Introduction to ETL

In modern organizations, data stored in operational systems often falls short of the strategic information required for informed decision-making. **Data warehousing** is designed to address this issue by using data from operational systems and reshaping it into valuable insights. This reshaping happens through the **ETL process**, which stands for **Extraction**, **Transformation**, and **Loading**.

ETL functions reshape relevant data from source systems into useful information that is stored in a data warehouse. The ETL process consists of three core steps:

- **Extraction**: Extracts data from various source systems.
- **Transformation**: Cleanses and converts the extracted data into a format that can be easily queried.
- **Loading**: Loads the transformed data into the data warehouse.

Each step ensures the data is accurate, clean, and ready for analysis. If the source data is not properly extracted, cleaned, and integrated in the correct format, query processing — the backbone of the data warehouse — could fail.

Functional Areas of ETL

ETL covers three key functional areas:

1. **Data Collection**: Captures and extracts data from source systems.
2. **Data Storage**: Stores the cleaned and transformed data in the warehouse.
3. **Information Delivery**: Makes the stored data available for query and analysis.

The **data collection** and **data storage** processes encompass the **extraction**, **transformation**, and **loading** of data. These back-end processes involve extracting data from source systems, transforming it into suitable formats, and storing it in the warehouse database. Once the data is properly stored, it becomes available for query processing, which allows decision-makers to derive strategic insights.

Importance of Requirements Definition

Before starting the ETL process, it’s crucial to define what data needs to be extracted and from which source systems. This ensures that only the relevant data is processed and transformed for the warehouse, reducing inefficiencies and errors.

Common ETL Challenges

ETL processes can face several challenges, particularly when dealing with diverse source systems. Some of these issues include:

- **Diverse Source Systems**: Data often comes from multiple platforms and operating systems, making extraction complex.
- **Legacy Systems**: Many older systems use obsolete database technologies, making data extraction difficult.

- **Lack of Historical Data**: Source systems often don't retain historical data, which is critical for data warehouses.
- **Inconsistent Data Quality**: Older systems may have evolved over time, leading to poor data quality.
- **Changing Source Systems**: As business conditions change, source systems may evolve, requiring ETL processes to adapt as well.
- **Inconsistency Among Source Systems**: Different systems may represent the same data in different ways (e.g., salary as monthly, weekly, or bimonthly). This inconsistency must be addressed during data transformation.
- **Cryptic Data Representation**: Some source systems may use ambiguous or unclear data types, making transformation essential to make the data meaningful to users.

Major Steps in the ETL Process

To build an effective data warehouse, follow these major steps in the ETL process:

1. **Determine Target Data**: Identify all the data required in the data warehouse.
2. **Identify Data Sources**: Determine all internal and external data sources.
3. **Prepare Data Mapping**: Map the source data elements to target elements in the warehouse.
4. **Establish Data Extraction Rules**: Define rules for extracting data from the sources.
5. **Set Transformation and Cleansing Rules**: Establish guidelines for cleaning and transforming the data.
6. **Plan for Aggregate Tables**: Prepare for the creation of aggregate tables in the warehouse.
7. **Organize Staging Area**: Set up a staging area for processing and testing the data.
8. **Write Data Load Procedures**: Document procedures for loading data into the warehouse.
9. **ETL for Dimension Tables**: Handle the extraction, transformation, and loading of dimension tables.
10. **ETL for Fact Tables**: Manage the ETL process for fact tables, which store the quantitative data for analysis.

Data Extraction

Data extraction plays a critical role in both operational systems and data warehouses, but the processes differ significantly in complexity and scope. For operational systems, data extraction is usually a one-time process that involves moving data from an older system to a new one. In contrast, data extraction for a data warehouse is more complex for two primary reasons:

1. **Disparate Data Sources**: Unlike operational systems, where data is usually extracted from one source, data warehouses require data from various, often disparate, sources. This could include multiple databases, external systems, or files. Coordinating these extractions to ensure consistency and accuracy is a challenging task.
2. **Ongoing Data Updates**: For operational systems, extraction is typically a one-time effort during system implementation. However, data warehouses require both an initial full load of data and

ongoing incremental updates to reflect changes in source systems over time. These incremental updates capture new or changed data and ensure the data warehouse stays up to date.

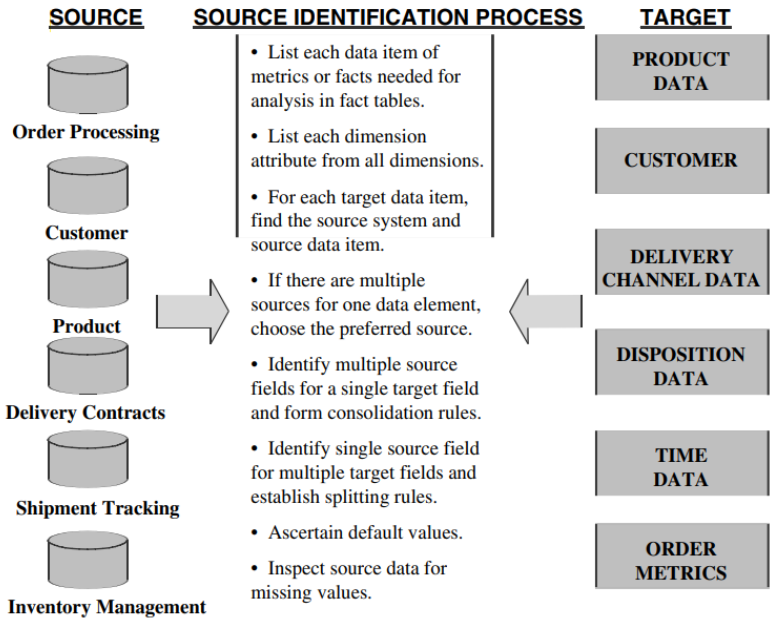
Because of this added complexity, data extraction in data warehousing often requires third-party extraction tools that come with built-in flexibility and record metadata. In contrast, in-house programs might be difficult and costly to maintain, especially when source systems change frequently. A robust data extraction strategy is essential for ensuring the success of a data warehouse.

Key Data Extraction Issues for Data Warehouses:

1. **Source Identification**: Identify all source applications and data structures that provide necessary data for the warehouse.
2. **Extraction Method**: Decide whether manual or tool-based extraction will be used for each data source.
3. **Frequency**: Establish how often data should be extracted—daily, weekly, quarterly, etc.
4. **Time Window**: Define the specific time windows for performing the data extraction.
5. **Job Sequencing**: Ensure that the extraction jobs are executed in the correct sequence if they depend on one another.
6. **Exception Handling**: Define how to handle records that cannot be extracted successfully.

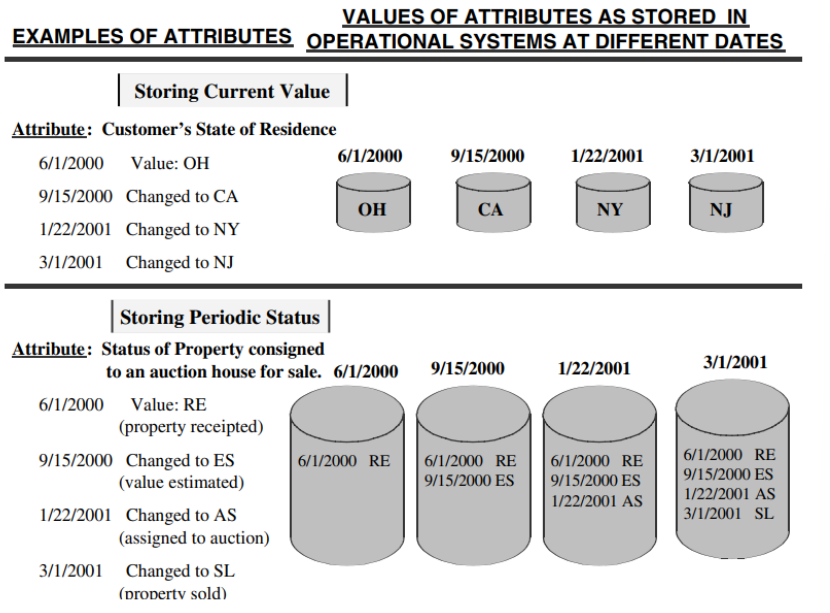
For example, in an order fulfillment system, various attributes such as order amount, discounts, commissions, delivery time, and order statuses need to be extracted from the source systems. Proper source identification and verification ensure that all necessary data is captured for analysis and decision-making.

Data Extraction Techniques



Data in operational systems can be divided into two categories:

1. **Current Value Data**: This includes attributes where values change frequently based on business transactions, such as customer addresses or account balances. These values represent the state of the data at the current moment, and tracking historical changes for data warehousing can be challenging.
2. **Periodic Status Data**: In this category, data changes are recorded with time stamps, preserving the history of changes. Examples include insurance policies or order statuses, where each change is recorded with a time reference.



Data extraction techniques generally involve two types:

- **Static Data Capture**: A snapshot of the data at a specific point in time, used for the initial load of the data warehouse or a full refresh of certain data.
- **Incremental Data Capture**: Capturing only the changes (revisions) since the last extraction, which can be immediate or deferred, based on how frequently the data changes and needs to be updated in the data warehouse.

Major Incremental Data Capture Methods

Data extraction is a critical step in building a data warehouse, and choosing the right extraction method depends on factors such as the source system, data update frequency, and performance needs. There are two main approaches: **Immediate Data Extraction** and **Deferred Data Extraction**. Each method has its specific use cases and techniques.

1. Immediate Data Extraction

This method involves capturing data as soon as changes occur in the source system. It is ideal for situations where real-time or near real-time data is required in the data warehouse for reporting,

analysis, or other business functions. There are three common techniques for immediate extraction:

a. Capture through Transaction Logs

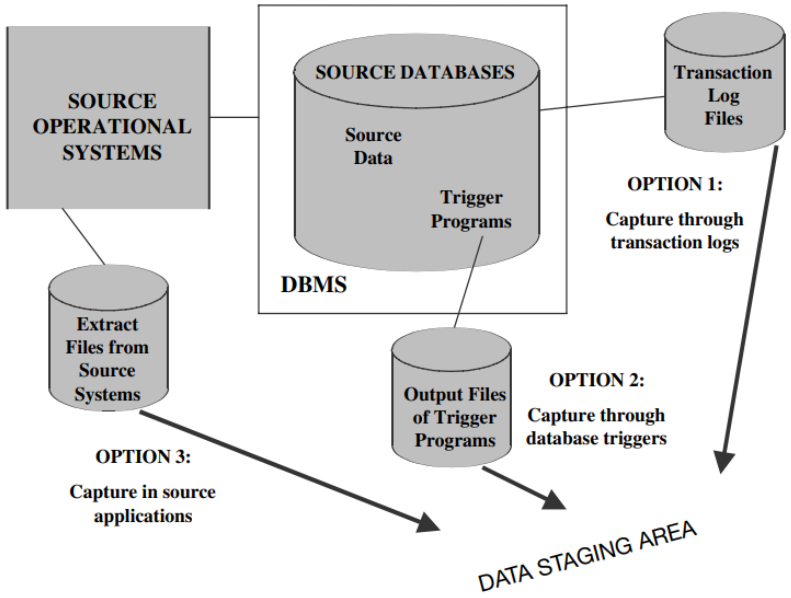
- **Overview**: Transaction logs are files maintained by database systems to record all changes made to the data (e.g., inserts, updates, deletes). These logs are primarily used for recovery in case of system failure, but they can also be tapped for data extraction.
- **How It Works**: In this method, the extraction process reads the transaction logs in real-time or periodically and captures any changes to the data (also known as Change Data Capture or CDC). Every change in the database is recorded in these logs, allowing extraction systems to detect and capture new, modified, or deleted data without querying the main database directly.
- **Advantages**:
  - **Real-time data capture**: Ensures that data in the data warehouse is almost immediately up-to-date with changes in the source system.
  - **Minimal impact on performance**: Because transaction logs are already maintained by the system for recovery purposes, reading from them has minimal impact on the database's performance compared to querying the live system.
- **Disadvantages**:
  - **Complexity**: Requires access to the transaction log files and specialized tools to interpret them.
  - **Database-dependent**: The structure and format of transaction logs vary across different databases (e.g., SQL Server, Oracle), making it less flexible across platforms.

b. Capture through Database Triggers

- **Overview**: Triggers are special types of stored procedures that automatically execute when specific events (insert, update, or delete) occur in the database.
- **How It Works**: In this method, triggers are set up on the source database tables. Whenever a relevant event occurs (such as inserting a new record, modifying an existing record, or deleting a record), the trigger captures the event and either logs the data change or sends it directly to the data extraction system.
- **Advantages**:
  - **Event-driven**: It captures data changes immediately as they happen, making it ideal for near real-time data extraction.
  - **Fine-grained control**: Triggers can be customized to capture only relevant data changes, reducing the volume of data extracted.
- **Disadvantages**:
  - **Performance overhead**: Triggers can introduce additional overhead on the source database, especially in high-volume transaction environments, as they execute in real-time.
  - **Maintenance complexity**: Managing and maintaining a large number of triggers across different tables can become cumbersome, especially in large databases.

c. Capture in Source Applications

- **Overview**: **Capture in Source Applications** is a method of immediate data extraction where changes are captured directly within the source application, typically as part of the application's normal operations. This method is often integrated into the application itself, ensuring that data is captured and transferred to the data warehouse in real-time or near real-time.
- **How It Works**: The source application is designed or modified to capture changes in data as they occur. This means that as transactions happen or as data is entered or updated, the application simultaneously logs these changes.
- **Advantages**:
  - **Real-time Updates**: This method allows for real-time or near real-time data extraction, ensuring that the data warehouse has the most up-to-date information.
  - **Less Dependence on External Tools**: Since the extraction process is embedded within the source application, there's no need for external data extraction tools or complex setups.
- **Disadvantages**:
  - **Increased Complexity in Application**: Embedding the data extraction process in the source application can increase the complexity of the application code, making it harder to maintain and troubleshoot.
  - **Performance Impact**: Since the extraction occurs within the application, it can impact the performance of the source system.



2. Deferred Data Extraction

In this method, data is not captured immediately when changes occur in the source system. Instead, the extraction process is delayed, often scheduled at intervals (e.g., daily, weekly). This method is

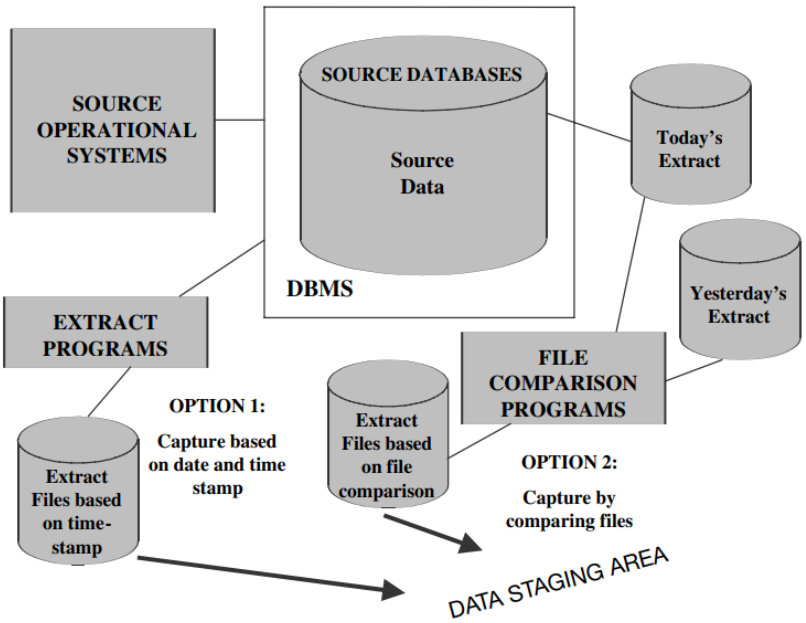
useful for less time-sensitive data or when real-time extraction isn't feasible due to system performance constraints.

a. Capture Based on Date and Time Stamp

- **Overview**: Most source systems record timestamps (e.g., "last modified" dates) for each record whenever data is updated or inserted. This timestamp can be used for data extraction.
- **How It Works**: The extraction process queries the source system to identify records that have changed since the last extraction, based on the date or time stamp. For example, if the extraction runs daily, it can retrieve all records modified within the past 24 hours.
- **Advantages**:
  - **Simplicity**: This method is straightforward and easy to implement as most databases natively store timestamps.
  - **Low performance impact**: Since only changed records are extracted, this method avoids putting unnecessary load on the system by querying for all data.
- **Disadvantages**:
  - **Not real-time**: Data may not be immediately available in the data warehouse, as it relies on scheduled extractions.
  - **Data consistency issues**: There can be timing issues if the timestamp doesn't capture all changes accurately, especially in systems with frequent updates.

b. Capture by Comparing Files

- **Overview**: This method involves comparing two versions of data files or database tables to identify changes (new, modified, or deleted records).
- **How It Works**: The extraction process compares two snapshots of the data, typically taken at different points in time (e.g., the current day's snapshot vs. the previous day's snapshot). The differences between these snapshots represent the changes, which are then extracted and processed for the data warehouse.
- **Advantages**:
  - **Broad applicability**: It can be used in systems where transaction logs or timestamps are not available, or when source systems are not databases (e.g., file-based systems).
  - **Accuracy**: As it involves comparing complete snapshots, all data changes are captured accurately.
- **Disadvantages**:
  - **High overhead**: Comparing entire datasets can be resource-intensive, especially for large databases, as it requires reading and comparing each record.
  - **Delay**: Like other deferred methods, the data is only available after the extraction process completes, leading to potential delays in data availability.



Capture of static data

Good flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
Can be used on legacy systems.
Can be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture in source applications

Good flexibility for capture specifications.
Performance of source systems affected a bit.
Major revisions to existing applications.
Can be used on most legacy systems.
Can be used on file-oriented systems.
High internal costs because of in-house work.

Capture through transaction logs

Not much flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
Can be used on most legacy systems.
Cannot be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture based on date and time stamp

Good flexibility for capture specifications.
Performance of source systems not affected.
Major revisions to existing applications likely.
Cannot be used on most legacy systems.
Can be used on file-oriented systems.
Vendor products may be used.

Capture through database triggers

Not much flexibility for capture specifications.
Performance of source systems affected a bit.
No revisions to existing applications.
Cannot be used on most legacy systems.
Cannot be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture by comparing files

Good flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
May be used on legacy systems.
May be used on file-oriented systems.
Vendor products are used. No internal costs.

Data Transformation

Data transformation is a critical part of the Extract, Transform, Load (ETL) process in data warehousing. It refers to the process of converting raw data from various sources into a format that can be effectively used in a data warehouse or data analytics system. This process ensures that data from disparate systems is consistent, accurate, and structured for analysis.

Basic Tasks of Data Transformation

Regardless of the complexity of the data sources or the extent of the data warehouse, data transformation often involves the following fundamental tasks:

1. **Selection**: This is the first step, where relevant data is selected from the source systems. Sometimes, entire records are selected, while in other cases, only specific parts of the data are extracted. In situations where the source structure does not allow precise selection during extraction, the whole record is extracted, and the selection happens during transformation.
2. **Splitting/Joining**: This involves breaking down records into smaller parts or combining data from different sources. Joining is more common in data warehouses, where multiple records or datasets need to be combined for analysis.
3. **Conversion**: Data fields are standardized across different systems or made understandable for end users. This may involve changing data types, standardizing formats, or making data uniform across different source systems.
4. **Summarization**: In some cases, detailed data may not be required for analysis. Data summarization reduces granularity by aggregating data (e.g., summarizing daily sales rather than storing each transaction).
5. **Enrichment**: This task involves rearranging or simplifying data fields to make them more useful for analysis. For instance, multiple fields can be combined to create a new, enriched view of the data.

Major Transformation Techniques

The following are common techniques or types of transformations performed on data before loading it into the warehouse:

1. **Format Revisions**: This involves changing the format, data types, or lengths of fields. Standardization of formats ensures that data from different systems can be compared and understood by users.
2. **Decoding Fields**: Source systems often use cryptic or inconsistent codes for the same data items. Decoding these fields ensures that data is presented in a user-friendly way. For example, gender might be coded as "1" and "2" in one system and "M" and "F" in another. The transformation process decodes and standardizes these values.
3. **Calculated and Derived Values**: Certain fields in the data warehouse may not exist in the source systems and need to be calculated. For example, profit margins or total costs might need to be derived from sales and cost data.
4. **Splitting Fields**: Sometimes, older systems store multiple pieces of information (such as names or addresses) in a single field. For data warehouse analysis, these fields need to be split into their components, such as splitting a full name into first name and last name.
5. **Merging Information**: This involves combining data from different sources into a single entity. For instance, information about a product may come from various systems, and it needs to be merged into a unified record in the data warehouse.
6. **Character Set Conversion**: When data is transferred between systems that use different character encodings (e.g., from EBCDIC to ASCII), it needs to be converted to a standard format.

7. **Conversion of Units of Measurement**: When dealing with international data, measurements (e.g., metric vs. imperial units) must be converted to a consistent standard.
8. **Date/Time Conversion**: Different systems use different date formats (e.g., U.S. vs. British formats). Date and time fields must be converted to a standardized format for consistency.
9. **Summarization**: In some cases, granular data is not required, and summary data can be used instead. For instance, storing the daily total of sales rather than each transaction helps reduce data storage and increases processing efficiency.
10. **Key Restructuring**: When legacy systems use complex or meaningful keys (e.g., product codes containing location data), these keys may need to be restructured into simpler, system-generated keys to avoid issues with changes in business processes.
11. **Deduplication**: In many cases, duplicate records (such as customer records) exist in the source systems. Deduplication ensures that only one record for each entity is kept in the data warehouse.

Data Integration and Consolidation

A major challenge in data transformation is integrating data from multiple disparate systems. These systems may have different formats, data representations, and business rules, making integration complex. Two common issues in data integration are:

1. **Entity Identification Problem**: When dealing with multiple systems, entities such as customers, vendors, or products may be represented differently across the systems. Identifying which records correspond to the same entity across systems is a significant challenge. For instance, a customer may have different IDs in different systems, and matching these records requires complex algorithms.
2. **Multiple Sources Problem**: A single data element might come from multiple systems, and these systems may have conflicting or inconsistent values. Deciding which value to use in the data warehouse requires resolving these conflicts, often by establishing business rules or priorities for different sources.

Issues in Data Transformation

Several issues may arise during data transformation:

- **Data Loss**: Sometimes, data transformation can lead to the loss of important details, especially during summarization or when data is incorrectly decoded or merged.
- **Performance Overhead**: Complex transformations, especially on large datasets, can be time-consuming and resource-intensive. Efficient algorithms and optimization techniques are necessary to manage this.
- **Data Quality**: Inconsistent or erroneous data from source systems may lead to incorrect transformations, resulting in poor data quality in the data warehouse.
- **Business Rule Conflicts**: Source systems may follow different business rules, making it difficult to apply a uniform transformation process.

Data Loading

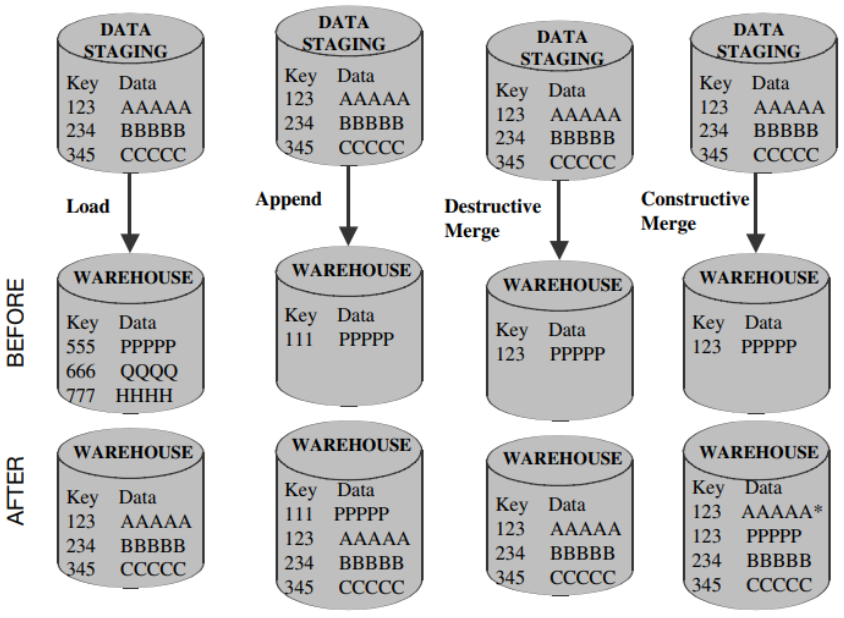
Data loading is the process of transferring transformed and integrated data from the staging area into the data warehouse. This is the final stage of the ETL (Extract, Transform, Load) process, where prepared data is applied to the warehouse, populating its tables and making it accessible for analysis.

The data loading process can be referred to in various terms, such as **applying the data**, **loading the data**, or **refreshing the data**. These terms denote different aspects of the data loading process, and it's important to differentiate between them:

1. **Initial Load**: Populates the data warehouse tables for the very first time.
2. **Incremental Load**: Periodically applies changes (updates, new records) to the data warehouse.
3. **Full Refresh**: Completely erases the contents of one or more tables and reloads them with fresh data.

Data can be applied to the warehouse in different modes:

1. **Load**: Wipes existing data and loads new data from the incoming file.
2. **Append**: Adds incoming data while keeping existing data. Duplicate records can either be added or rejected.
3. **Destructive Merge**: Updates matching records and adds new ones.
4. **Constructive Merge**: Keeps existing records and adds new ones, marking them as superseding the old ones.



Types of Loads

1. Initial Load

- **Definition**: The initial load is the first step in populating all the tables in the data warehouse.

- **Process**: It involves extracting data from source systems, integrating and transforming it, and creating load images. These load images are then applied to the data warehouse tables.
- **Challenges**: This process can take a significant amount of time, so it's essential to plan carefully and test the initial load to estimate the running time. The data warehouse may need to be taken offline during the load process to ensure no interference with user queries.

2. Incremental Load

- **Definition**: This refers to updating the data warehouse with ongoing changes from source systems. We will use this strategy when less than 10% data of table is being updated.
- **Process**: Instead of reloading the entire dataset, incremental loads apply only the data that has changed since the last load. This involves extracting only the updated or new records from the source systems, transforming them, and applying them to the relevant tables in the warehouse.
- **Example**: If a product's price changes, the incremental load would apply only the updated product record instead of reloading all product records.
- **Challenges**: Proper handling of keys and timestamps is important to ensure that only the appropriate changes are applied. This load type can be done on a scheduled basis (e.g., daily or weekly) depending on business needs.

3. Full Refresh

- **Definition**: A full refresh involves wiping out the data in one or more tables and reloading it completely from scratch. We will use this strategy when more than 10% data of table is being updated.
- **Process**: Like the initial load, the data is extracted, transformed, and applied to the tables in the data warehouse. However, the key difference is that existing data is erased before loading the new data.
- **When to Use**: Full refreshes are typically done periodically or when the entire dataset needs to be replaced with a fresh version, such as when there are significant structural changes to the data.
- **Challenges**: The time and resources required for a full refresh are similar to those of the initial load, and ensuring minimal disruption to users is crucial.

4. Trickle Feed (Continuous Load)

- **Definition**: Trickle Feed refers to continuously loading small batches of data into the data warehouse as new data becomes available from source systems, ensuring near real-time updates.
- **Process**: Unlike traditional batch processing, this method loads data incrementally and in real-time. As soon as new or updated data is detected, it is extracted, transformed, and loaded into the warehouse in small, continuous batches. The process happens throughout the day, keeping the warehouse updated.
- **Example**: In an online retail system, as soon as a customer places an order, the order details are immediately loaded into the data warehouse, allowing real-time order tracking and analytics.

- **Challenges**: Continuous data loading can increase system complexity, requiring efficient monitoring to avoid performance bottlenecks. Synchronizing data between source and warehouse in real-time is crucial to maintain consistency.

Applying Data in Different Scenarios

Let's examine how the different modes of data application fit into the types of loads:

1. Initial Load:

- The **Load** mode is most commonly used, especially when loading the entire data warehouse in one go. However, if multiple runs are needed to complete a single table, the **Append** mode might be used after the first run.

2. Incremental Load:

- The **Constructive Merge** mode is commonly used for incremental loads, where updates need to be tracked over time. This allows for the addition of new records while maintaining the history of existing records.
- In some cases, **Destructive Merge** is used, such as when correcting errors or updating Type 1 slowly changing dimensions (where old records are replaced by updated ones).

3. Full Refresh:

- This process is similar to the initial load but requires the existing data to be erased before loading new data. The **Load** and **Append** modes are typically used for full refreshes.

Question 2: (5 Points)
Assume that 50,000 rows out of 10 million rows in Account dimension table changes on each data refresh. Which loading strategy should you follow? Explain the reasons for your selection. Also suggest some practical steps that expedite the data loading process.

Ans:
As the data changes on each refresh is low (i.e. 0.5%) which is less than 10%, so Incremental data refresh loading strategy is more efficient.

Difference Between ETL and ELT

- **ETL (Extract, Transform, Load)**: In ETL, data is **first extracted** from source systems, **then transformed** (cleaned, filtered, and structured) in a staging area, and finally **loaded** into the data warehouse. This process is good when you need to process and transform data before storing it. It's typically used when the warehouse cannot handle large-scale transformations efficiently.
- **ELT (Extract, Load, Transform)**: In ELT, data is **first extracted** from the source systems and **loaded directly** into the data warehouse without any transformation. The transformation happens **after the data is loaded** into the warehouse. ELT is useful when your warehouse is powerful enough to handle large transformations and when you want to load data quickly for analysis without delay.

Aspect	ETL (Extract, Transform, Load)	ELT (Extract, Load, Transform)

Data Processing Order	Extract → Transform → Load	Extract → Load → Transform
Transformation Stage	Happens before loading into the warehouse	Happens after loading into the warehouse
Speed	Slower loading due to transformations before load	Faster loading, as transformations happen later
System Requirements	Requires an external ETL tool for transformation	Uses the data warehouse's processing power
Use Case	Best for structured data, smaller transformations	Ideal for large datasets and complex transformations