ETL (Extract Transform Load)

**Content:**

1. What is ETL?

2. Difference between ETL and ELT.

3. Introduction to OLTP and OLAP.

4. Difference between OLTP and OLAP.

5. What is ACID property?

6. What is Data Warehouse? Need of Data Warehouse?

7. Data Warehouse Architecture

8. What are Data Marts and its types?

9. What is ODS (Operational Data Store)?

10. What is Dimensional Modelling?

11. What is Fact and Dimension?

12. What is Fact Table and Dimension Table?

13. Star, Snowflake and Galaxy Schema

14. What is Normalization and De-normalization?

15. What are the different Normal Forms?

16. Slowly Changing Dimension

1.What is ETL?

ETL stands for Extract, Transform and Load, which is a process used in data integration and data warehousing. It involves three key steps:

1. **Extract:** The first step is extracting data from various source systems. These sources could include databases, flat files, APIs, cloud storage, or external data systems. The goal is the collect the raw data from these diverse systems.
2. **Transform:** Once the data is extracted, it is transformed into a usable format. This step involves cleaning, filtering, enriching, aggregating and formatting the data to make it suitable for analysis or reporting.
3. **Load:** The final step is loading the transformed data into a target system such as a data warehouse, database or data lake, where it can be analysed, queried and used by business users.

**Use Cases:** ETL processes are commonly used in data warehousing projects and business intelligence (BI) systems, where large volumes of data need to be consolidated, cleaned and prepared for analysis.

2. Difference between ETL and ELT

* ETL:

1. Extract: Data is extracted from various source systems like data warehouse, files, APIs, etc.
2. Transform: The extracted data is cleaned, formatted and transformed to fit the structure required for the target system (e.g., data warehouse).
3. Load: The transformed data is loaded into the targeted system, typically a data warehouse, databases, etc.

**Where the transformation happens:** Transformation occurs before the data is loaded into the target system.

**Use Cases:** ETL is traditionally used in data warehousing and reporting systems, where data is pre-processed before it enters the data warehouse to ensure the data is clean, consistent and ready for analysis.

* ELT:

1. Extract: Data is extracted from the various source systems.
2. Load: The extracted data is directly loaded onto the target system (such as cloud data warehouse like Snowflake, Google Big-Query or Amazon Redshift)
3. Transform: Transformation happens after the data is loaded into the target system, typically using the processing power of the target system itself.

**Where the transformation happens:** Transformation happens after the data is loaded into the target system.

**Use Cases:** ELT is more common in modern cloud-based data architectures. It works well when the target system has strong processing capabilities (such as cloud data warehouse), allowing for the transformations to be done at scale on the raw data once it is stored.

|  |  |  |
| --- | --- | --- |
| **Feature** | **ETL (Extract Transform Load)** | **ELT (Extract Load Transform)** |
| Transformation Location | Before loading (target system) | After loading (target system) |
| Processing Power | External tools (ETL Servers) | Data Warehouse/target system(cloud-based) |
| Complexity | More complex in intermediate steps | Simplified with powerful target systems |
| Performance | Slower for large datasets | Faster, leverages data warehouse’s power |
| Best For | Business Intelligence (BI), Reporting systems, | Modern cloud-based data architectures. |

3. Introduction to OLTP and OLAP

* OLTP:

Online Transaction Processing (OLTP) is a class of system that manages transaction-oriented applications. OLTP systems are widely used in industries such as banking, retail, telecommunications and other sector where real-time transactional data is processed. OLTP systems are designed to handle a large number of short online transactions, such as database insert, updates and deletes, typically involving high-frequency, small-scale operations.

Key Features:

1. Real-Time Processing: OLTP systems handles transactions as they occur, providing immediate results and responses to users. For example, when a customer makes a purchase, the system processes the transaction and updates the inventory and financial records in real time.
2. Transactional Integrity: OLTP system must ensure the ACID (Atomicity, Consistency, Isolation, Durability). This ensures the transactions processed reliably, even in the case of failures or system crashes.
3. Short Transactions: Transactions in OLTP systems are short, often involving a few records at a time. For example, a customer may make a payment, which only involves updating a few database tables (account balance, transaction log, etc)
4. High Transaction Volume: systems need to handle a high volume of transactions, which can range from thousands to millions of transactions per day. This requires efficient database operations and optimized data storage to ensure speed and scalability.

Use Cases:

1. Banking: Processing customer transactions such as deposits, withdrawals, and transfers in real time.
2. Retail: Managing customer orders, inventory updates, and payment processing.
3. E-commerce: Handling product orders, payments, customer account information, and shipping status updates.

* OLAP:

Online Analytical Processing (OLAP) refers to systems that allow users to analyse and interact with large volumes of data in complex ways, typically for decision-making, business intelligence, or reporting purposes. OLAP is designed for querying and analysing multi-dimensional data and is commonly used in data warehousing environments.

Key Features:

1. Complex queries: OLAP systems are optimized for complex queries and calculations, such as aggregations and multi-dimensional analysis.
2. Read heavy workload: OLAP systems are designed to support heavy read operations rather than frequent updates or insertions.
3. Data warehousing: OLAP is typically used in conjunction with data warehousing, where historical data is stored and analysed.
4. Multi-dimensional analysis: OLAP allows for slicing and dicing of data across multiple dimensions (e.g., time, geography, product categories).

Use Cases:

1. Business Intelligence: Analysing sales data, customer behaviour, and market trends.
2. Financial Reporting: Generating executive-level reports on profitability, expenses, and financial forecasting.
3. Market Research: Analysing survey data or sales patterns across different dimensions.
4. Difference between OLTP and OLAP

|  |  |  |
| --- | --- | --- |
| **Feature** | **OLTP (Online Transaction Processing)** | **OLAP (Online Analytical Processing)** |
| Purpose | Transaction-oriented (daily operations) | Analytical (business intelligence, decision support) |
| Data Volume | Handles a high number of small transactions | Handles large amounts of aggregated historical data |
| Query Complexity | Simple, fast queries (insertion, update, delete) | Complex, multi-dimensional queries (aggregation, slicing) |
| Database Structure | Highly normalized for efficiency in updates | Denormalized for efficient querying and reporting |
| Transaction Type | Frequent small transactions (e.g., order processing) | Periodic large data analysis (e.g., trend reports) |
| Performance | Optimized for fast data entry and retrieval | Optimized for read-heavy, complex queries |
| Example System | Banking, e-commerce, reservation systems | Data warehouses, business intelligence platforms |

1. What is ACID property?

ACID properties guarantee that database transactions are processed reliably and ensures data integrity.

1. Atomicity:

* Definition: Either all transactions are completed successfully, or none are.
* Example: If you transfer money from one bank account to another, both steps must succeed together. If one step fails (e.g., insufficient balance) the entire transaction is rolled back.

1. Consistency:

* Definition: A transaction must leave the database in a valid state.
* Example: Database has a rule that every order must have at least one item, consistency ensures that transaction doesn’t leave the database in a state where an order exists without an item.

1. Isolation:

* Definition: Isolation ensures that the operations of a transaction are isolated from the other transactions.
* Example: If two users are simultaneously transferring money from the same account, isolation ensures that one user’s transaction does not affect the other’s, it ensures that one transaction is completed before the other begins.

1. Durability:

* Definition: Durability ensures that once a transaction has been committed, it is permanently recorded in the database and will survive any subsequent system crashes or failures.
* Example: After a successful bank transaction, if the system crashes immediately afterward, the transfer will still be reflected in the database once the system is restored.

6. What is Data Warehouse?

Data Warehouse is a central repository that stores large amounts of data that receives from multiple sources, typically used to support decision-making, business intelligence (BI), and reporting needs within an organization. Data warehouses store historical and current data from various operational systems, transforming it into a consistent and consolidated format to enable efficient analysis.

**Benefits of a Data Warehouse:**

* Improved Decision-Making: By consolidating data from multiple sources, a data warehouse provides a comprehensive view of business operations, helping organizations make more informed decisions.
* Historical Analysis: Data warehouses store historical data, enabling organizations to track trends, analyse patterns over time, and make forecasts based on past behaviour.
* Faster Query Performance: Data warehouses are optimized for complex analytical queries, allowing users to quickly retrieve data for reporting and analysis, even when dealing with large datasets.
* Data Consistency: The ETL process ensures that data is cleansed, transformed, and standardized, leading to consistent and accurate reporting.

**Example Use Cases:**

* Business Intelligence (BI): A company can use a data warehouse to generate comprehensive reports on sales, customer behaviour, and inventory levels across various time periods.
* Financial Reporting: Financial departments can analyse historical transactions, compare performance across fiscal periods, and forecast future revenue.
* Marketing Analytics: Marketing teams can analyse customer demographics, spending patterns, and campaign effectiveness to refine marketing strategies.

7. Data Warehouse Architecture

1. **Data Source Layer (Data Sources)**:

Data comes from various operational systems such as databases, CRM software, ERP systems, and external data sources (social media, market research, etc.).

1. **ETL Layer (Extract, Transform, Load)**:

Data from different sources is extracted, transformed (cleaned and standardized), and loaded into the data warehouse. The transformation step ensures that data from different sources is in a consistent format before it is stored in the warehouse.

1. **Data Storage Layer (Data Warehouse)**:

Data is stored in the data warehouse in a structured format. Typically, data is organized in star schemas, snowflake schemas, or fact and dimension tables, allowing for efficient querying.

1. **OLAP (Online Analytical Processing) Layer**:

OLAP tools allow users to perform multidimensional analysis on the data warehouse. These tools provide fast, interactive querying and the ability to "slice and dice" the data to uncover insights.

1. **Data Marts**:

A data mart is a subset of a data warehouse that focuses on a specific business area or department (e.g., sales, finance, or marketing). It contains a more focused dataset that is easier and faster for specific teams to analyse.

1. **Data Presentation Layer:**

This is the layer where end-users, analysts, and business intelligence tools access the data for reporting, dashboards and analytics. The data can be visualized and analysed using business intelligence tools.

8. Data Marts and its types

A Data Mart is a subset of a data warehouse that is focused on a specific business area, department, or subject. It is a smaller, more specialized version of the data warehouse, designed to meet the specific needs of a particular group of users or a business function (like sales, marketing, or finance). Data marts are used to improve the efficiency of query processing and to enable faster and more relevant reporting for individual business units.

**Types of Data Marts**

1. **Dependent Data Mart:**

A dependent data mart is created from an existing data warehouse. In this architecture, data is extracted from the central data warehouse and then loaded into the data mart. This ensures that the data in the data mart is consistent with the data in the data warehouse.

**Example Use Case:**

A finance department may create a dependent data mart using the sales and transaction data already stored in the data warehouse, filtering out unnecessary data for their reporting needs.

1. **Independent Data Mart:**

In independent data mart is created without relying on a central data warehouse. Data is sourced directly from operational systems or external data sources and stored in the data mart. This type of data mart is self-contained and does not require a data warehouse for its creation or operation.

**Example Use Case:**

A marketing department might create an independent data mart using customer transaction data from a CRM system to analyse customer behaviour without needing the entire company’s data.

1. **Hybrid Data Mart:**

A hybrid data mart combines elements of both dependent and independent data marts. It pulls data from both a data warehouse and operational systems or external sources. This type of data mart offers flexibility by integrating different types of data sources.

**Example Use Case:**

A sales department might build a hybrid data mart that pulls customer and sales data from the data warehouse while also including real-time data from the company's point-of-sale system.

9. What is ODS (Operational Data Store)?

An Operational Data Store (ODS) is a type of database used to store and manage real-time, current data from various operational systems within an organization. Unlike a data warehouse, which is designed for historical and analytical data, an ODS is focused on providing up-to-date, integrated data from multiple sources to support day-to-day operational decision-making.

The ODS acts as an intermediary or staging area that collects data from different transactional systems (such as ERP, CRM, or other operational databases) and stores it in a more accessible and usable format for reporting and operational analysis.

**Use Cases for ODS:**

1. Operational Reporting:

The ODS is commonly used for real-time or near-real-time operational reporting. For example, customer service representatives can query the ODS to check the latest order status or inventory levels.

1. Short-Term Decision Making:

The ODS enables management to make day-to-day decisions based on current data. For example, sales managers may use the ODS to track daily sales performance and make decisions on pricing or promotions.

1. Data Integration for Operational Systems:

The ODS can consolidate data from various operational systems to provide a unified view. For example, integrating customer data from sales, marketing, and service departments can help provide a complete picture of customer interactions.

1. Staging Area for Data Warehousing:

Before data is moved to a data warehouse for historical analysis, it can first be loaded into the ODS for integration, transformation, and quality checks. This makes sure that only clean and consistent data is passed to the data warehouse.

Presentation / Front End

CRM

BI Tools

Reports

Dashboards

Data Mart

Data Warehouse

ODS

ERP

Data Mart

ETL

E-Commerce

Data Mart

External

10. What is Dimensional Modelling?

Dimensional modelling is a design technique used to structure data in a way that is optimized for querying and reporting in data warehouses and business intelligence systems. The goal is to create a database schema that makes it easy for business users (such as analysts and executives) to retrieve, analyse, and understand the data quickly and intuitively.

In dimensional modelling, data is organized into dimensions and facts to support complex analytical queries, like trend analysis, sales performance, and customer segmentation.

**Example of Dimensional Model**

Let’s say we are modeling a sales analysis system for a retail store:

* **Fact Table: Sales**
  + Measures: SalesAmount, QuantitySold, DiscountAmount
  + Foreign Keys: TimeKey, ProductKey, CustomerKey, StoreKey
* **Dimension Tables:**
  + **Time**: Contains attributes like Date, Month, Quarter, Year
  + **Product**: Contains attributes like ProductID, ProductName, ProductCategory
  + **Customer**: Contains attributes like CustomerID, CustomerName, CustomerRegion
  + **Store**: Contains attributes like StoreID, StoreLocation, StoreType

11. What are Facts and Dimensions?

1. **Facts**:
   * **Definition:** Facts are quantitative data points that represent measurable business events or transactions, such as sales, revenue, profit, or units sold.
   * **Example:** A sales fact table might include data like the number of units sold, sales revenue, or discounts applied for each transaction.
   * **Characteristics:**
     + Typically stored in **fact tables**.
     + They are numeric and additive, meaning they can be summed, averaged, or aggregated.
     + They are typically the focal point for analysis.
2. **Dimensions**:
   * **Definition:** Dimensions are descriptive attributes that provide context to the facts. They answer "who," "what," "where," and "when" questions related to the facts.
   * **Example:** A sales report might be analysed based on dimensions like **time**, **product**, **location**, and **customer**.
   * **Characteristics:**
     + Typically stored in **dimension tables**.
     + They are textual or categorical and provide descriptive details that make the facts meaningful (e.g., "January" for the time dimension or "Laptop" for the product dimension).
     + Dimensions are used for filtering, grouping, and categorizing data.

12. What is Fact Table and Dimension Table?

**1.Fact Tables**:

* + **Definition:** A fact table is a central table in a dimensional model that contains the facts (measurable data) for business processes. It often contains numeric data and keys to the related dimension tables.
  + **Characteristics:**
    - Contains **foreign keys** linking to dimension tables.
    - Contains **measures** (facts) like sales amounts, quantities, or costs.
    - Can be large because they often store transactional or detailed data.
    - May include aggregated data, depending on the level of granularity.

**2.Dimension Tables**:

* + **Definition:** Dimension tables store the descriptive information about each dimension. They provide context for the facts in the fact table.
  + **Characteristics:**
    - Contain descriptive fields that describe a dimension (e.g., **customer name**, **product category**, **region**).
    - Dimension tables are typically smaller in size compared to fact tables.
    - Used for filtering and grouping in queries (e.g., "sales by product" or "sales in the North region").

13. Star, Snowflake and Galaxy Schema

**1. Star Schema**:

* + **Definition:** The star schema is the simplest and most commonly used structure in dimensional modelling. It consists of a **fact table** at the centre, surrounded by **dimension tables** (like a star).
  + **Structure:**
    - The fact table is connected to the dimension tables via foreign keys.
    - Each dimension table contains attributes that describe different aspects of the business process (e.g., time, customer, product).
    - **Example:**
      * Fact table: Sales (with measures like units sold, revenue, etc.)
      * Dimension tables: Time, Customer, Product, Location.

**2. Snowflake Schema**:

* + **Definition:** The snowflake schema is a more complex variation of the star schema where dimension tables are **normalized**. This means that dimension data is divided into additional related tables, creating a more intricate structure.
  + **Structure:**
    - The fact table remains at the centre, but dimension tables are split into additional tables to reduce data redundancy.
    - For example, in a snowflake schema, the **Product** dimension might be broken down into **Product Category** and **Product Subcategory** tables.
    - **Example:**
      * Fact table: Sales
      * Dimension tables: Time, Customer, Product (split into Product Category and Product Subcategory), Location.
  + **Advantages:** Reduces data redundancy and storage.
  + **Disadvantages:** Increases complexity in querying and slower performance due to the need for multiple joins.

**3. Galaxy Schema (or Fact Constellation)**:

* + **Definition:** The galaxy schema is a more complex schema that involves multiple fact tables sharing dimension tables. It is used when there are multiple business processes that require different sets of facts but share the same dimensions.
  + **Structure:**
    - Multiple fact tables, each representing a different business process, such as **sales**, **inventory**, and **returns**, all sharing common dimensions like **time**, **customer**, or **product**.
  + **Example:**
    - Fact tables: Sales, Inventory, Returns.
    - Shared dimension tables: Time, Product, Customer.
  + **Advantages:** Suitable for complex business environments where multiple business processes share common dimensions.
  + **Disadvantages:** More complex to design and manage.

14. What is Normalization and De-normalization?

* Normalization:
  + Normalization is the technique of dividing the data into multiple tables to reduce data redundancy.
  + Normalization is used in OLTP systems, which emphasizes on making the insert, update and delete anomalies faster.
  + Normalization increases the number of tables and joins.
  + Disc space is optimized in a normalised table.
* De-normalization:
  + De-normalization is exact opposite of normalization. Denormalization is the technique of combining the data into a single table to make data retrieval faster.
  + Denormalization is used in OLAP system, which emphasizes on making the search and analysis faster.
  + Denormalization reduces the number of tables and joins.
  + Disc space is wasted in denormalization.

15. What are the different Normal Forms?

There is total 6 defined Normal Forms,

1. **First Normal Form (1NF):**

* A relation will be in 1NF if it contains an atomic value which means that a column/attribute of a table cannot hold multiple values. Table must hold only single valued column/attribute.

1. **Second Normal Form (2NF):**

* In 2NF, relation must be in 1NF.
* It should not have partial dependency.
* If the proper subset of any candidate key determines the non-prime attributes, it is called partial dependency.

1. **Third Normal Form (3NF):**

* In 3NF, relation must be in 2NF and not contain any transitive functional dependency.
* If y depends on x, it ok’s no issues but if z depends on y and y depends on x then it’s a transitive functional dependency.

1. Boyce-Codd Normal Form (BCNF):

* In BCNF, relation must be in 3NF.
* Ensure that every determinant (a column that determines other column) is a candidate key.

1. Fourth Normal Form (4NF):

* In 4NF, relation must be in BCNF.
* It has no multivalued dependencies.

1. Fifth Normal Form (5NF):

* In 5NF, relation must be in 4NF.
* It has no join dependencies that are not implied by the candidate key.

16. Slowly Changing Dimension

**Slowly Changing Dimensions (SCDs)** are a concept in data warehousing that refer to how the attributes of dimensions (the descriptive data that characterizes facts, like Customer, Product, Employee, etc.) change over time. In many business scenarios, the values of certain dimension attributes change slowly, rather than quickly. For example, a customer’s address or a product's price might change, but these changes don’t happen frequently.

There are different methods for managing Slowly Changing Dimensions (SCDs), depending on the specific requirements of how historical changes should be tracked and stored.

There are 6 main types of Slowly Changing Dimensions (SCD)

**Type 1 (SCD 1): Overwrite**

* In Type 1, when a change occurs in the dimension attribute, the old value is overwritten with the new one. This means that the previous historical data is lost.
* **Use Case**: This is appropriate for data that doesn't require historical tracking (e.g., correcting a misspelled customer name).
* **Example**: A customer's address changes, and the old address is replaced with the new one in the data warehouse.

**Type 2 (SCD 2): Add New Row (Historical Tracking)**

* Type 2 dimensions maintain full history by adding new rows in the dimension table whenever a change happens. The new row typically includes an effective date (start date) and sometimes an end date.
* **Use Case**: This is used when you need to keep historical data for all changes to the dimension. It's useful for cases where understanding the state of data at a particular time is important.
* **Example**: A customer changes their address, and a new row is added to the dimension table with the new address, marking the previous address as no longer valid.

**Type 3 (SCD 3): Add New Column (Limited Historical Tracking)**

* Type 3 stores both the old and new values in the same row but in separate columns. Typically, this allows you to keep only a limited amount of history (e.g., the previous and current values).
* **Use Case**: This is useful when only a small number of historical changes need to be tracked, and you don't need the full history.
* **Example**: A customer changes their address, and both the current and the previous address are stored in separate columns in the same row.

**Type 4: History Table (Separate Historical Table)**

* **Concept**: In Type 4, the historical data is maintained in a separate table from the current dimension table. The main table contains the current values, and all historical changes are stored in a separate history table.
* **How it works**: The current dimension table only holds the latest data, while the historical dimension table records all changes. This allows for efficient querying of the current data while keeping the historical data separate for historical analysis.
* **Use Case**: Useful when you want to store historical data but do not want to clutter the main dimension table with many rows.
* **Example**: In a customer dimension, the main table would store only the current customer details (name, address, etc.), while the history table would store previous addresses, names, or other attributes that have changed over time.

**Type 6: Hybrid Approach (Combination of Type 1, Type 2, and Type 3)**

* **Concept**: Type 6 is a hybrid approach that combines elements of **Type 1**, **Type 2**, and **Type 3** Slowly Changing Dimensions. This type typically maintains the current value in the current column (like Type 1), keeps historical changes in new rows (like Type 2), and also includes additional columns to track the previous value (like Type 3).
* **How it works**: A dimension table in Type 6 might store the current value in one column, the historical value in another, and also add a mechanism to track changes over time (such as effective dates, start and end dates).
* **Use Case**: Useful when a business needs to track full historical data (like Type 2), maintain only some previous historical data (like Type 3), and also keep the current state (like Type 1) all in the same dimension table.
* **Example**: A product dimension may track the current price (Type 1), keep the full price history (Type 2), and retain the previous price value (Type 3).

**Type 0: Fixed Dimension (No Changes Allowed)**

* **Concept**: Type 0 is used when the dimension data is **fixed** and does **not** change over time. This means that once a record is created in the dimension table, it cannot be modified or updated, and no historical tracking is needed.
* **How it works**: The dimension table is considered to be static, and any change to the dimension data (e.g., an update to a customer's name or address) is prevented. This is used for data that will never change once it is recorded.
* **Use Case**: Typically used for **reference data** that is constant and stable, such as country codes, product IDs, or other non-volatile attributes.
* **Example**: A dimension table for countries might contain a list of countries and their respective country codes. These attributes are not likely to change, so no historical tracking is needed.