Unit-4:

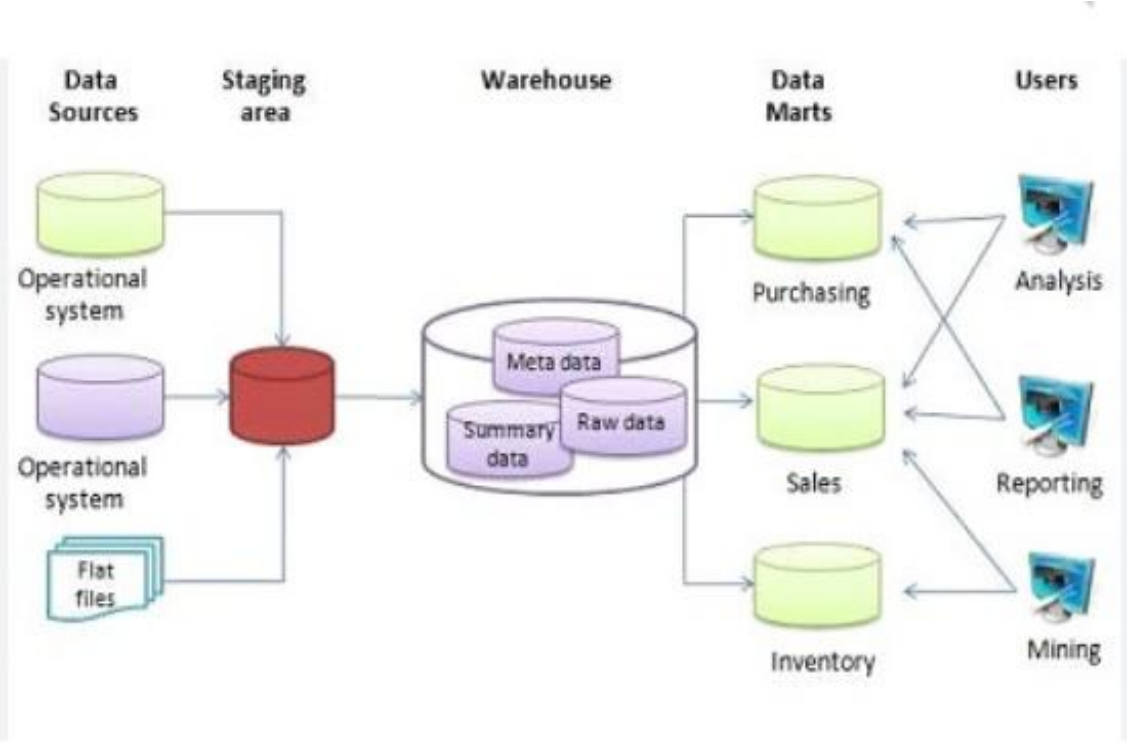
4.1 Concepts of Data Warehouse

Data warehousing is the process of constructing and using a data warehouse. A data warehouse is constructed by integrating data from multiple heterogeneous sources that support analytical reporting, structured and/or ad hoc queries, and decision making. Data warehousing involves data cleaning, data integration, and data consolidations.

Data Warehouse is a relational database management system (RDBMS) construct to meet the requirement of transaction processing systems. It can be loosely described as any centralized data repository which can be queried for business benefits. It is a database that stores information oriented to satisfy decision-making requests. It is a group of decision support technologies, targets to enabling the knowledge worker (executive, manager, and analyst) to make superior and higher decisions. So, Data Warehousing support architectures and tool for business executives to systematically organize, understand and use their information to make strategic decisions.

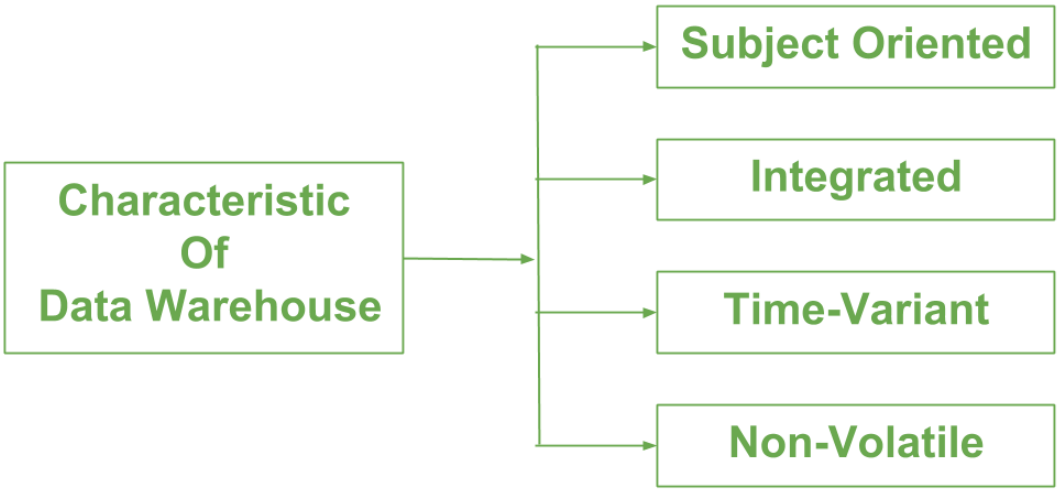
A Data Warehouse (DW) is a relational database that is designed for query and analysis rather than transaction processing. It includes historical data derived from transaction data from single and multiple sources.

A Data Warehouse provides integrated, enterprise-wide, historical data and focuses on providing support for decision-makers for data modeling and analysis.



Unit-4:

4.1.1 Features and Types of Data Warehouse



1. **Subject-oriented –** A data warehouse is always a subject oriented as it delivers information about a theme instead of organization's current operations. It can be achieved on specific theme. That means the data warehousing process is proposed to handle with a specific theme which is more defined. These themes can be sales, distributions, marketing etc.

   A data warehouse never put emphasis only current operations. Instead, it focuses on demonstrating and analysis of data to make various decision. It also delivers an easy and precise demonstration around particular theme by eliminating data which is not required to make the decisions.

2. **Integrated –** It is somewhere same as subject orientation which is made in a reliable format. Integration means founding a shared entity to scale the all similar data from the different databases. The data also required to be resided into various data warehouse in shared and generally granted manner.
   A data warehouse is built by integrating data from various sources of data such that a mainframe and a relational database. In addition, it must have reliable naming conventions, format and codes. Integration of data warehouse benefits in effective analysis of data. Reliability in naming conventions, column scaling, encoding structure etc. should be confirmed. Integration of data warehouse handles various subject related warehouse.

3. **Time-Variant –** In this data is maintained via different intervals of time such as weekly, monthly, or annually etc. It founds various time limit which are structured between the large datasets and are held in online transaction process (OLTP). The time limits for data warehouse is wide-ranged than that of operational systems. The data resided in data warehouse is predictable with a specific interval of time and delivers information from the historical perspective. It comprises elements of

time explicitly or implicitly. Another feature of time-variance is that once data is stored in the data warehouse then it cannot be modified, alter, or updated. Data is stored with a time dimension, allowing for analysis of data over time.

4. **Non-Volatile –** As the name defines the data resided in data warehouse is permanent. It also means that data is not erased or deleted when new data is inserted. It includes the mammoth quantity of data that is inserted into modification between the selected quantity on logical business. It evaluates the analysis within the technologies of warehouse. Data is not updated, once it is stored in the data warehouse, to maintain the historical data.
In this, data is read-only and refreshed at particular intervals. This is beneficial in analysing historical data and in comprehension the functionality. It does not need transaction process, recapture and concurrency control mechanism. Functionalities such as delete, update, and insert that are done in an operational application are lost in data warehouse environment. Two types of data operations done in the data warehouse are:

- Data Loading
- Data Access

1. **Subject Oriented:** Focuses on a specific area or subject such as sales, customers, or inventory.
2. **Integrated:** Integrates data from multiple sources into a single, consistent format.
3. **Read-Optimized:** Designed for fast querying and analysis, with indexing and aggregations to support reporting.
4. **Summary Data:** Data is summarized and aggregated for faster querying and analysis.
5. **Historical Data:** Stores large amounts of historical data, making it possible to analyze trends and patterns over time.
6. **Schema-on-Write:** Data is transformed and structured according to a predefined schema before it is loaded into the data warehouse.
7. **Query-Driven:** Supports ad-hoc querying and reporting by business users, without the need for technical support.

Types Of Data Warehouse

Three main types of Data Warehouses (DWH) are:

1. Enterprise Data Warehouse (EDW):

Enterprise Data Warehouse (EDW) is a centralized warehouse. It provides decision support service across the enterprise. It offers a unified approach for organizing and representing data. It also provide the ability to classify data according to the subject and give access according to those divisions.

2. Operational Data Store:

Operational Data Store, which is also called ODS, are nothing but data store required when neither Data warehouse nor OLTP systems support organizations reporting needs. In ODS, Data warehouse is refreshed in real

time. Hence, it is widely preferred for routine activities like storing records of the Employees.

3. Data Mart:

A data mart is a subset of the data warehouse. It specially designed for a particular line of business, such as sales, finance, sales or finance. In an independent data mart, data can collect directly from sources.

4.1.2 Difference among OLAP and OLTP

OLTP is an operational system that supports transaction-oriented applications in a 3-tier architecture. It administers the day to day transaction of an organization. OLTP is basically focused on query processing, maintaining data integrity in multi-access environments as well as effectiveness that is measured by the total number of transactions per second. The full form of OLTP is Online Transaction Processing.

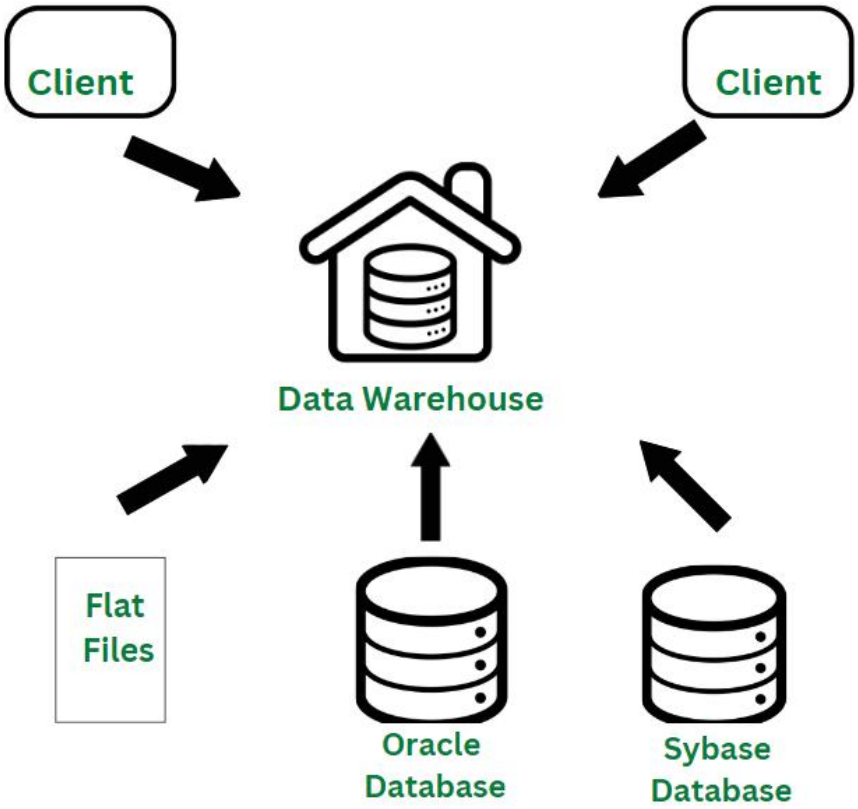
Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information. This chapter cover the types of OLAP, operations on OLAP, difference between OLAP, and statistical databases and OLTP.

OLAP	OLTP
Involves historical processing of information.	Involves day-to-day processing.
OLAP systems are used by knowledge workers such as executives, managers and analysts.	OLTP systems are used by clerks, DBAs, or database professionals.
Useful in analyzing the business.	Useful in running the business.
Contains historical data.	Contains current data.
Provides summarized and multidimensional consolidated data.	Provides primitive and highly detailed data.
Number or users is in hundreds.	Number of users is in thousands.
Number of records accessed is in millions.	Number of records accessed is in tens.
Database size is from 100 GB to 1 TB	Database size is from 100 MB to 1 GB.
Highly flexible.	Provides high performance.
Based on Star Schema, Snowflake, Schema and Fact Constellation Schema.	Based on Entity Relationship Model.

4.2 Integrating heterogeneous Database

Integration of heterogeneous databases in data warehousing refers to the process of combining data from multiple, disparate databases into a central repository, known as a data warehouse. This process involves extracting data from different sources, such as relational databases, NoSQL databases, and flat files, and then transforming, cleaning, and loading the data into the data warehouse.

The main goal of integrating heterogeneous databases in data warehousing is to make the data from different sources available in a consistent, unified format, allowing for easy querying and analysis of the data. This is particularly useful in organizations that have multiple databases with different structures and data models, as it allows for the integration of data from different systems, applications, and departments.



Need for Integration of Heterogeneous Databases

There are several reasons why the integration of heterogeneous databases is important in data warehousing:

Improved data governance: Integrating data from multiple sources allows for better data governance, as all data can be centrally managed and controlled. This can help to ensure data quality, security and compliance with regulations.

Increased efficiency: By integrating data from multiple sources, organizations can avoid the need to manually combine data from different databases, which can be time-consuming and error-prone.

Greater insights: Integrating data from different sources allows organizations to gain greater insights from their data by analyzing it in new ways and identifying patterns and trends that would not be visible with data from a single source.

Better Decision Making: Integrating heterogeneous data can help organizations to make better decisions by providing a more complete view of their operations, customers, and business performance.
Business continuity: In case of any system failure or unavailability of any particular data source, the data warehousing system can still be operational with the rest of the available data sources, this helps in business continuity.

4.2.1 Advantages and Dis-advantages of Query-driven and Updatedriven Apprroach.

1. Query-Driven Approach:

The query-driven approach for the integration of heterogeneous databases is a method of integrating data from different sources by using a central query processor to handle all data requests. With a query-driven strategy, several sophisticated queries will be created for each separate database. There will therefore be a requirement for filtering and integration of the queries as a query-driven technique will provide complex results. Therefore the query-driven approach is not preferable for the companies as it is an inefficient and expensive approach.

Advantages of Query-Driven Approach
Flexibility: The query-driven approach allows users to formulate complex queries that can retrieve data from multiple heterogeneous databases. This flexibility enables users to extract specific information or perform advanced analytics across different data sources.

Integration: Heterogeneous databases often have different data models, structures, and query languages. The query-driven approach provides a unified interface for querying diverse databases, enabling seamless integration of data from multiple sources. It eliminates the need for users to learn and interact with multiple database systems individually.

Efficient Data Retrieval: By leveraging the query-driven approach, users can execute queries that retrieve only the required data, minimizing data transfer and reducing network overhead. This can lead to improved performance and faster response times, especially when dealing with large datasets.

Scalability: The query-driven approach allows for scalability by distributing query processing across multiple heterogeneous databases. This can be advantageous when dealing with high-volume or high-velocity data sources, as it enables parallel execution of queries, enhancing overall system performance.

Disadvantages of Query-Driven Approach:

Performance issues: Query-driven integration can lead to poor performance, especially when dealing with large amounts of data or complex queries.

Expensive Approach: Query Driven Approach is expensive when tends to queries that require aggregations to perform.

Lack of data consistency: Query-driven integration can lead to data inconsistencies, as the integration process may not ensure that data is consistent across the different databases.

Inefficient: Query Driven Approach is Inefficient when dealing with the integration of heterogeneous databases as the databases exist from multiple sources.

2. Update Driven Approach:

The update-driven approach for the integration of heterogeneous databases in data warehousing is a method of integrating data from multiple databases by periodically updating the data in a central warehouse. The information from several heterogeneous sources is advanced combined and stored in a warehouse in an update-driven method. It is possible to directly query and analyze this saved data. As a result, many businesses utilize the update-driven strategy rather than the query-driven approach for integration because it is more effective and quick.

Advantages

Performance: Update-driven integration can improve performance by reducing the need for runtime translation of queries and allowing for optimized indexing and caching of data.

Data consistency: Update-driven integration can ensure data consistency by periodically updating the data in the warehouse, which can reduce data inconsistencies and errors.

Flexibility: Update-driven integration can be more flexible, as it allows for incremental updates and can adapt to changes in the data models or DBMS.
Efficient: Update driven approach is more efficient as the data are integrated and stored in advance in the data warehouse.

Disadvantages

Lack of Centralized Control: With an update-driven approach, there is no centralized control or oversight over the updates made to the heterogeneous databases. This can lead to inconsistencies in data across the different databases if proper synchronization mechanisms are not in place. It becomes challenging to enforce data integrity rules and ensure that updates are applied uniformly and correctly.

Data Synchronization Complexity: In a heterogeneous environment, databases may have different schemas, structures, and data models. Updating data directly in each database can introduce complexity when

synchronizing changes across the databases. Ensuring that updates are correctly propagated and applied consistently across all databases requires careful planning, coordination, and monitoring.

Increased Maintenance Effort: With an update-driven approach, maintenance and management of the heterogeneous databases become more complex. Each database may require individual attention, updates, and performance optimizations. This can increase the workload for database administrators and potentially lead to higher costs and resource utilization.

Difficulty in Enforcing Data Consistency: Since updates are made directly to the individual databases, it becomes challenging to enforce strict data consistency across all the databases. Inconsistencies may arise if updates are not properly synchronized or if conflicts occur during concurrent updates. Resolving data conflicts and ensuring data integrity becomes more challenging in such a decentralized approach.

Limited Scalability: The update-driven approach may face scalability limitations when dealing with large-scale heterogeneous databases. As the number of databases increases, the complexity of managing updates and ensuring data consistency grows exponentially. Coordinating and scaling the update processes across multiple databases can become a significant bottleneck.

Lack of Comprehensive Transaction Management: Transaction management becomes more complex with the update-driven approach. Maintaining atomicity, consistency, isolation, and durability (ACID properties) across multiple databases requires careful planning and coordination. Ensuring that updates are transactionally consistent and recoverable becomes more challenging in a heterogeneous environment.

4.2.2 Concepts of Data Warehouse Tools:

4.2.2.1 Extraction, Data Cleaning, Data Transformation

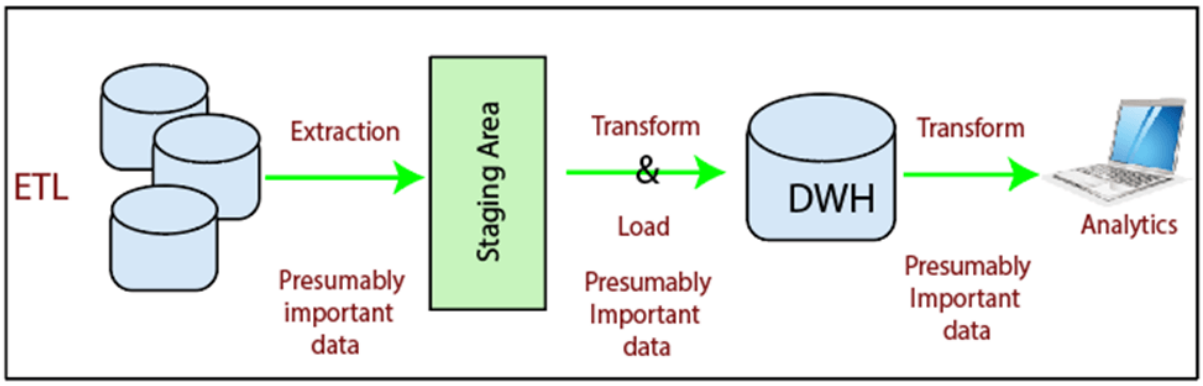
ETL (Extract, Transform, and Load) Process

What is ETL?

The mechanism of extracting information from source systems and bringing it into the data warehouse is commonly called ETL, which stands for Extraction, Transformation and Loading.

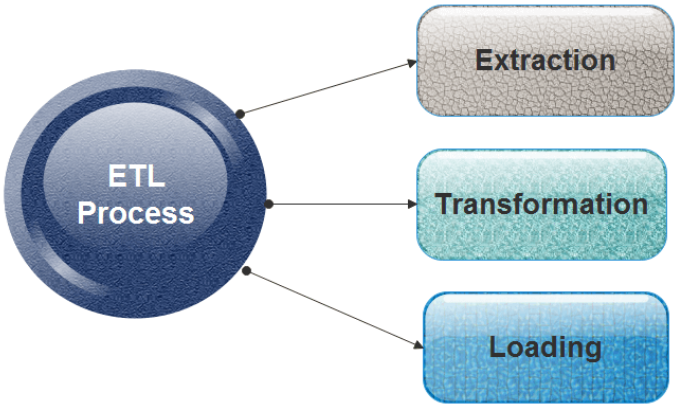
The ETL process requires active inputs from various stakeholders, including developers, analysts, testers, top executives and is technically challenging.

To maintain its value as a tool for decision-makers, Data warehouse technique needs to change with business changes. ETL is a recurring method (daily, weekly, monthly) of a Data warehouse system and needs to be agile, automated, and well documented.



How ETL Works?

ETL consists of three separate phases:



Extraction

- Extraction is the operation of extracting information from a source system for further use in a data warehouse environment. This is the first stage of the ETL process.
- Extraction process is often one of the most time-consuming tasks in the ETL.
- The source systems might be complicated and poorly documented, and thus determining which data needs to be extracted can be difficult.
- The data has to be extracted several times in a periodic manner to supply all changed data to the warehouse and keep it up-to-date.

Cleansing

The cleansing stage is crucial in a data warehouse technique because it is supposed to improve data quality. The primary data cleansing features found in ETL tools are rectification and homogenization. They use specific dictionaries to rectify typing mistakes and to recognize synonyms, as well as

rule-based cleansing to enforce domain-specific rules and defines appropriate associations between values.

The following examples show the essential of data cleaning:

If an enterprise wishes to contact its users or its suppliers, a complete, accurate and up-to-date list of contact addresses, email addresses and telephone numbers must be available.

If a client or supplier calls, the staff responding should be quickly able to find the person in the enterprise database, but this need that the caller's name or his/her company name is listed in the database.

If a user appears in the databases with two or more slightly different names or different account numbers, it becomes difficult to update the customer's information.

Transformation

Transformation is the core of the reconciliation phase. It converts records from its operational source format into a particular data warehouse format. If we implement a three-layer architecture, this phase outputs our reconciled data layer.

The following points must be rectified in this phase:

- Loose texts may hide valuable information. For example, XYZ PVT Ltd does not explicitly show that this is a Limited Partnership company.
- Different formats can be used for individual data. For example, data can be saved as a string or as three integers.

4.2.2.2 Data Loading

Loading is the ultimate step in the ETL process. In this step, the extracted data and the transformed data are loaded into the target database. To make the data load efficient, it is necessary to index the database and disable the constraints before loading the data. All three steps in the ETL process can be run parallel. Data extraction takes time and therefore the second phase of the transformation process is executed simultaneously. This prepared the data for the third stage of loading. As soon as some data is ready, it is loaded without waiting for the previous steps to be completed.

The loading process is the physical movement of the data from the computer systems storing the source database(s) to that which will store the data warehouse database.

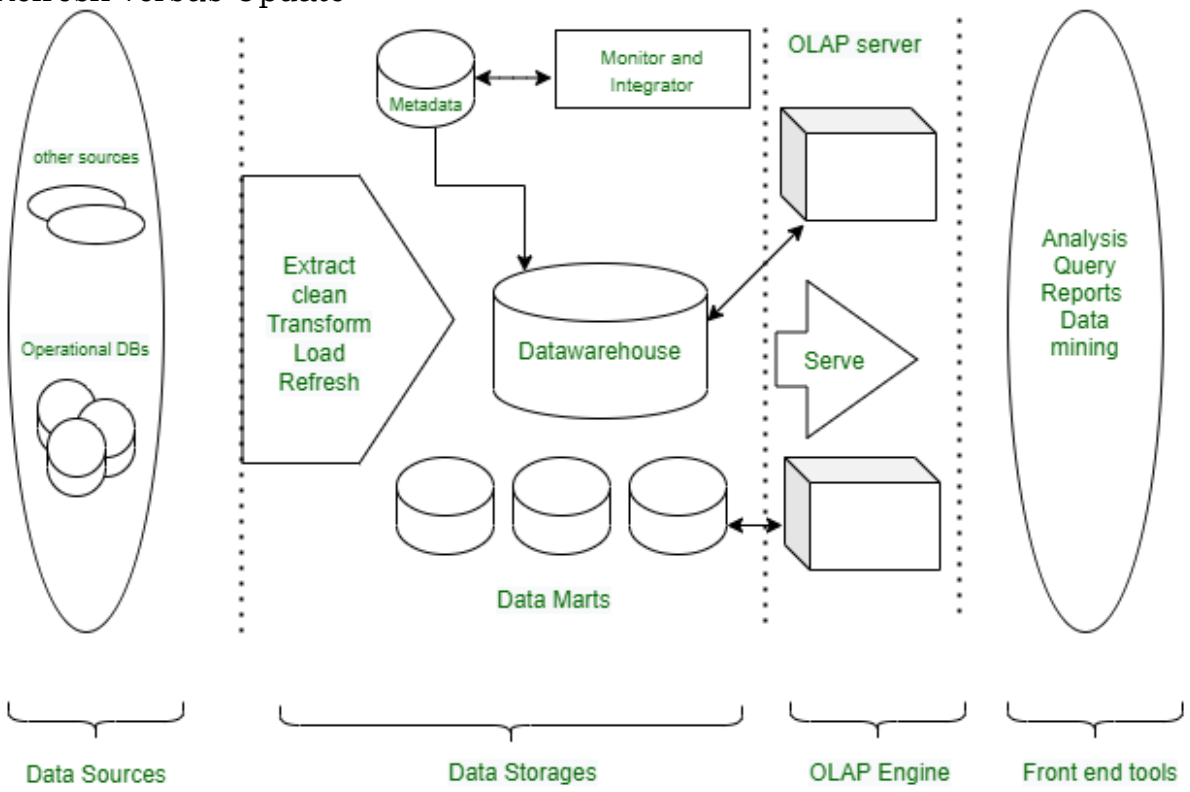
Initial Load: For the very first time loading all the data warehouse tables.

Incremental Load: Periodically applying ongoing changes as per the requirement. After the data is loaded into the data warehouse database, verify the referential integrity between the dimensions and the fact tables to

ensure that all records belong to the appropriate records in the other tables. The DBA must verify that each record in the fact table is related to one record in each dimension table that will be used in combination with that fact table.

Full Refresh: Deleting the contents of a table and reloading it with fresh data.

Refresh versus Update



4.2.3 Important terminologies of Data Warehouse:

4.2.3.1 MetaData, Metadata Repository

Metadata is simply defined as data about data. The data that is used to represent other data is known as metadata. For example, the index of a book serves as a metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data. In terms of data warehouse, we can define metadata as follows.

Metadata is the road-map to a data warehouse.

Metadata in a data warehouse defines the warehouse objects.

Metadata acts as a directory. This directory helps the decision support system to locate the contents of a data warehouse**.**

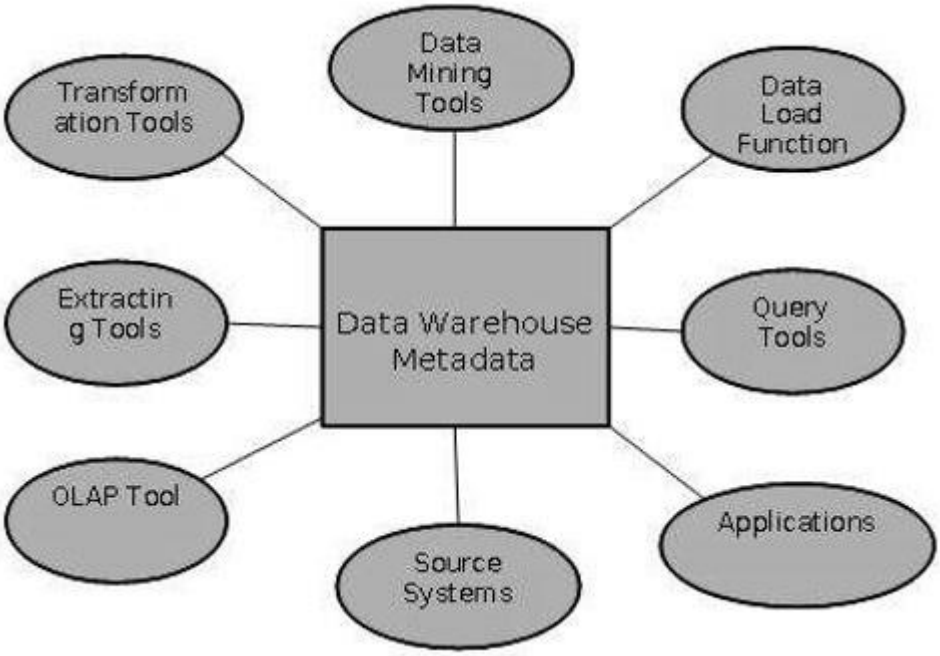
Metadata can be broadly categorized into three categories –

Business Metadata – It has the data ownership information, business definition, and changing policies.

Technical Metadata – It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural information such as primary and foreign key attributes and indices.

Operational Metadata – It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.

The following diagram shows the roles of metadata.



Metadata Repository

Metadata repository is an integral part of a data warehouse system. It has the following metadata –

Definition of data warehouse – It includes the description of structure of data warehouse. The description is defined by schema, view, hierarchies, derived data definitions, and data mart locations and contents.

Business metadata – It contains has the data ownership information, business definition, and changing policies.

Operational Metadata – It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.

Data for mapping from operational environment to data warehouse – It includes the source databases and their contents, data extraction, data partition cleaning, transformation rules, data refresh and purging rules.

Algorithms for summarization − It includes dimension algorithms, data on granularity, aggregation, summarizing, etc.

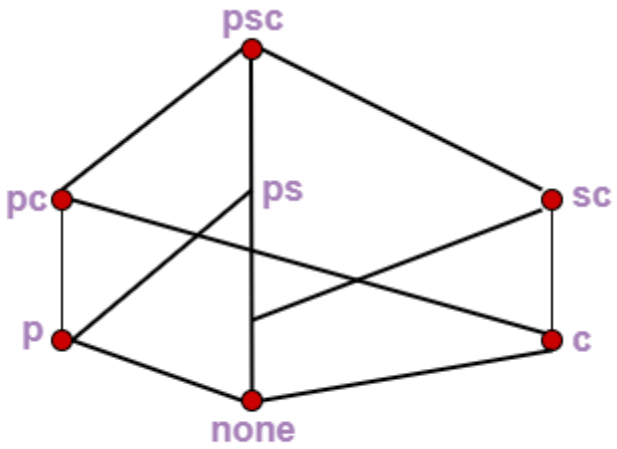
4.2.3.2 Data Cube, Data Mart

What is Data Cube?

When data is grouped or combined in multidimensional matrices called Data Cubes. The data cube method has a few alternative names or a few variants, such as "Multidimensional databases," "materialized views," and "OLAP (On-Line Analytical Processing)."

The general idea of this approach is to materialize certain expensive computations that are frequently inquired.

For example, a relation with the schema sales (part, supplier, customer, and sale-price) can be materialized into a set of eight views as shown in fig, where **psc** indicates a view consisting of aggregate function value (such as total-sales) computed by grouping three attributes part, supplier, and customer, **p** indicates a view composed of the corresponding aggregate function values calculated by grouping part alone, etc.



Eight views of data cubes for sales information.

A data cube is created from a subset of attributes in the database. Specific attributes are chosen to be measure attributes, i.e., the attributes whose values are of interest. Another attributes are selected as dimensions or functional attributes. The measure attributes are aggregated according to the dimensions.

For example, XYZ may create a sales data warehouse to keep records of the store's sales for the dimensions time, item, branch, and location. These dimensions enable the store to keep track of things like monthly sales of items, and the branches and locations at which the items were sold. Each dimension may have a table identify with it, known as a dimensional table, which describes the dimensions. For example, a dimension table for items may contain the attributes item_name, brand, and type.

Data cube method is an interesting technique with many applications. Data cubes could be sparse in many cases because not every cell in each dimension may have corresponding data in the database.

Techniques should be developed to handle sparse cubes efficiently.

If a query contains constants at even lower levels than those provided in a data cube, it is not clear how to make the best use of the precomputed results stored in the data cube.

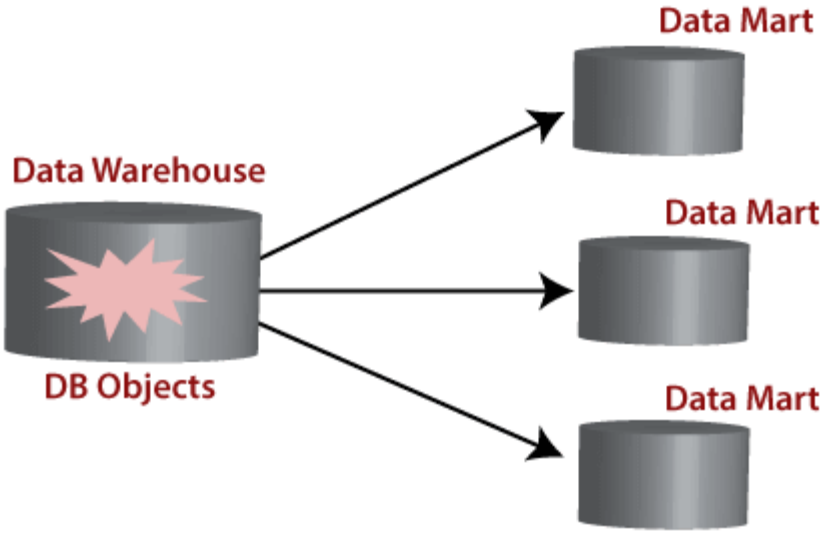
The model view data in the form of a data cube. OLAP tools are based on the multidimensional data model. Data cubes usually model n-dimensional data.

A data cube enables data to be modeled and viewed in multiple dimensions. A multidimensional data model is organized around a central theme, like sales and transactions. A fact table represents this theme. Facts are numerical measures. Thus, the fact table contains measure (such as Rs_sold) and keys to each of the related dimensional tables.

Data Mart

A **Data Mart** is a subset of a directorial information store, generally oriented to a specific purpose or primary data subject which may be distributed to provide business needs. Data Marts are analytical record stores designed to focus on particular business functions for a specific community within an organization. Data marts are derived from subsets of data in a data warehouse, though in the bottom-up data warehouse design methodology, the data warehouse is created from the union of organizational data marts.

The fundamental use of a data mart is **Business Intelligence (BI)** applications. **BI** is used to gather, store, access, and analyze record. It can be used by smaller businesses to utilize the data they have accumulated since it is less expensive than implementing a data warehouse.



Reasons for creating a data mart

- Creates collective data by a group of users
- Easy access to frequently needed data
- Ease of creation
- Improves end-user response time
- Lower cost than implementing a complete data warehouses
- Potential clients are more clearly defined than in a comprehensive data warehouse
- It contains only essential business data and is less cluttered.

Types of Data Marts

There are mainly two approaches to designing data marts. These approaches are

- Dependent Data Marts
- Independent Data Marts

Dependent Data Marts

A dependent data marts is a logical subset of a physical subset of a higher data warehouse. According to this technique, the data marts are treated as the subsets of a data warehouse. In this technique, firstly a data warehouse is created from which further various data marts can be created. These data mart are dependent on the data warehouse and extract the essential record from it. In this technique, as the data warehouse creates the data mart; therefore, there is no need for data mart integration. It is also known as a **top-down approach**.

Independent Data Marts

The second approach is Independent data marts (IDM) Here, firstly independent data marts are created, and then a data warehouse is designed using these independent multiple data marts. In this approach, as all the data marts are designed independently; therefore, the integration of data marts is required. It is also termed as a **bottom-up approach** as the data marts are integrated to develop a data warehouse.