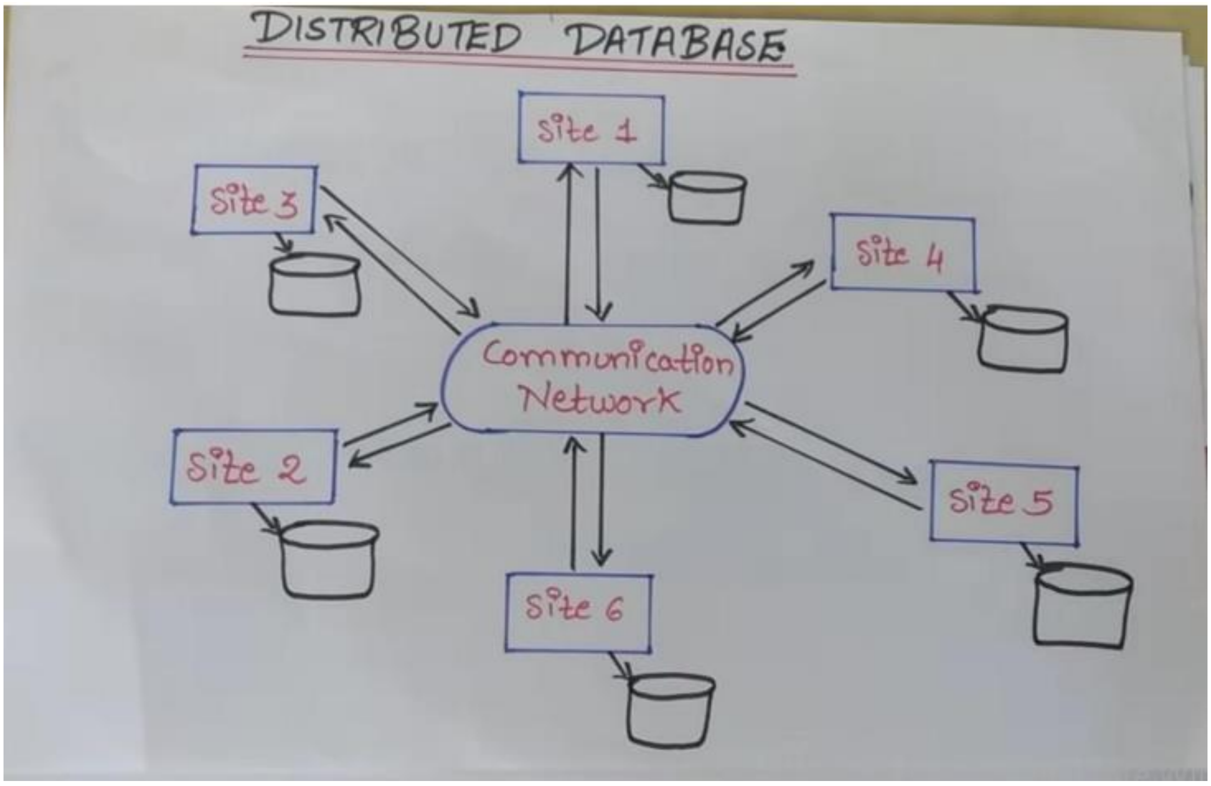
1) Define distributed database. Also explain the architecture of distributed database in detail.

A **Distributed Database System** is a database spread across multiple computers or locations connected by a network, meaning it's not limited to just one physical location or computer. This setup enables users from different parts of the world to access the database as if it were a single, unified system. Even though the data is physically distributed, users interact with it as if it's one database, which enhances accessibility, performance, and reliability.



Types of Distributed Database Systems

Distributed Database Systems come in two primary types: **Homogeneous** and **Heterogeneous**.

1. **Homogeneous Database System**:
   - In a homogeneous system, all sites (locations) store the same database and use identical software, data structures, and operating systems.
   - Each location has a copy of the database, making data management simpler and ensuring consistency across all sites.
   - This uniformity makes it easy to manage and integrate data as the same database structure and software are used throughout the system.
2. **Heterogeneous Database System**:
   - In this system, each site might use a different DBMS, operating system, or data structure, creating a more complex setup.

- Each location may have a unique database structure, so a special translation scheme (middleware) is needed to make sure data can be shared and managed across different sites.
- Due to differences in software and data structure, it can be challenging to maintain consistent communication and integration between sites.

Distributed Data Storage Techniques

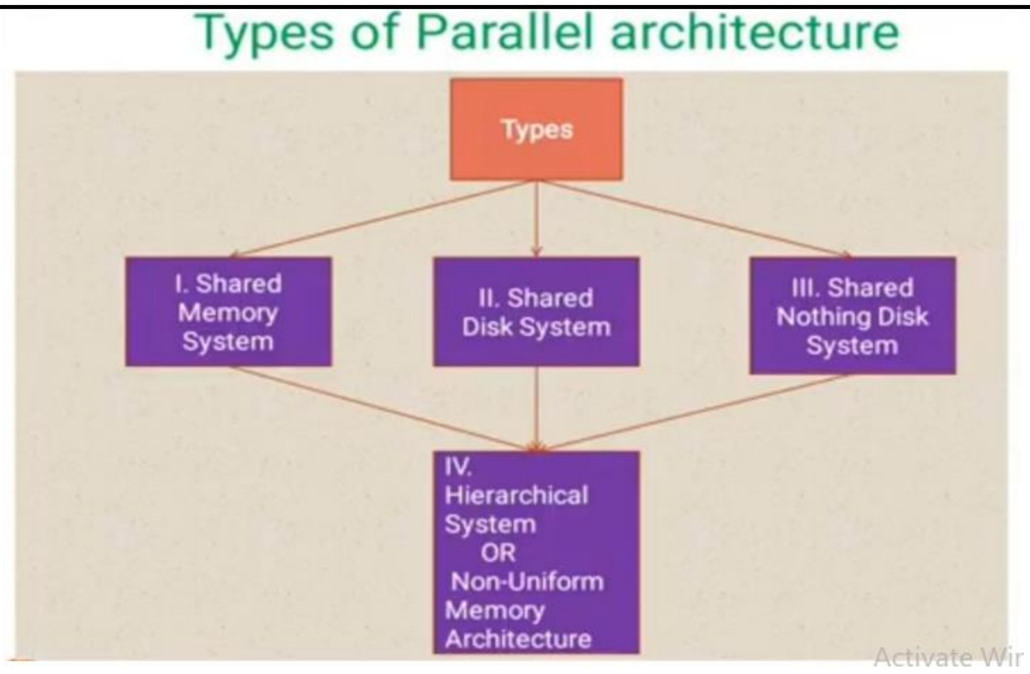
Data in distributed databases can be stored across multiple sites using two main methods: **Replication** and **Fragmentation**.

1. **Replication**:
   - Replication involves storing full copies of the database or specific tables at multiple locations. For example, a user in one country could access the entire database stored at their local site, while another user in a different location could access the same data locally.
   - **Advantages**: Replication improves data availability and enables faster access because users can access a local copy of the data.
   - **Challenges**: When data is updated in one location, the changes must be applied to all copies to maintain consistency. Managing these updates across sites adds complexity and overhead to the system.
2. **Fragmentation**:
   - Fragmentation divides the database into smaller parts, with each fragment stored at different locations. Unlike replication, this approach doesn’t involve duplicate data storage.
   - **Horizontal Fragmentation**: Divides the data by rows. For instance, customer data might be stored by region so that each location only has data relevant to its users.
   - **Vertical Fragmentation**: Divides the data by columns. For example, a customer table could be split so that contact information is stored in one fragment while financial information is stored in another. A common key, like a customer ID, links these fragments together.
   - **Advantages**: Fragmentation helps avoid redundancy, reduces data storage requirements, and is efficient for large databases.

2) Explain parallel database architecture.

A **Parallel Database Architecture** is designed to boost database processing speed and manage large data sets by distributing tasks across several processors. In this system, multiple requests can be processed at the same time, allowing high performance and scalability. Each processor handles a specific portion of the data or task, so tasks are completed much faster than if only one processor were working.

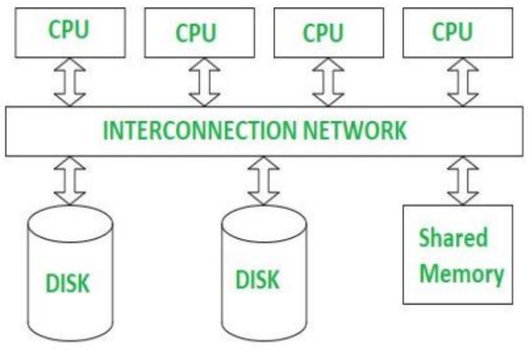
Parallel processing database divides a large task into many **smaller tasks**. Then it executes the smaller tasks concurrently on several separate processor on a separate machine.


Types of Parallel architecture
Types
I. Shared Memory System
II. Shared Disk System
III. Shared Nothing Disk System
IV. Hierarchical System OR Non-Uniform Memory Architecture
Activate Wir

Shared Memory Architecture:

In a shared memory system, all processors use the same central memory and storage. This setup allows fast and easy data access, as processors can directly share data.

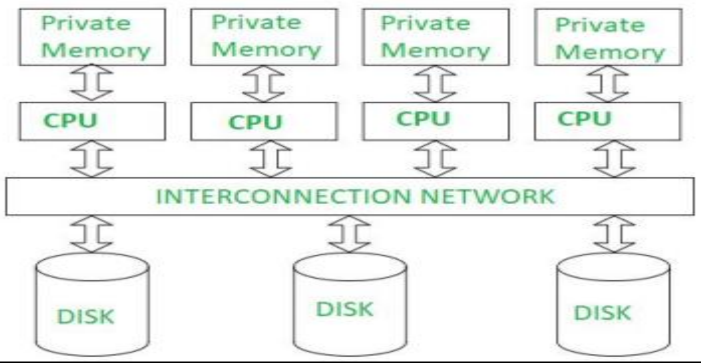
However, if many processors try to access the memory at once, it can slow down the system, creating a "bottleneck." This system works best for smaller applications that don't require many processors, as adding more can reduce performance.



Shared Disk Architecture

In a **shared disk architecture**, each processor has its own memory but shares a common disk storage with other processors. This setup allows multiple processors to work on tasks at the same time without interfering with each other's memory.

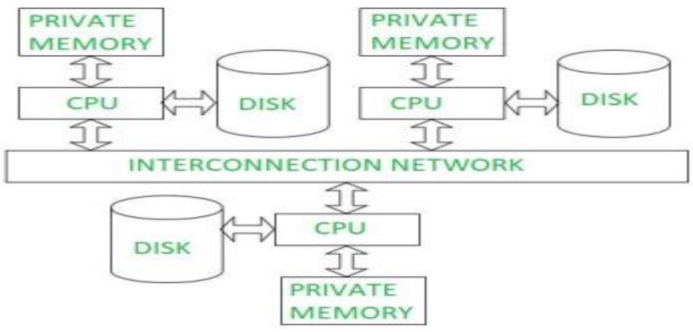
However, since they all access the same disk, delays can happen if many processors try to access data at once. This architecture is useful for mid-sized systems that need some scalability but don't have extremely large databases.



Shared Nothing Architecture

In a **shared nothing architecture**, each processor has its own memory and disk, working completely independently. There are no shared resources, so processors don’t have to wait for each other to access data. This makes the system very scalable and efficient for large databases.

You can easily add more processors to grow the system, making it suitable for large applications like big data analysis. However, managing these systems can be tricky because it requires careful coordination to ensure that all processors can access the necessary data without conflicts.



3. Abstract Data types / What is the Abstract Data type. Explain with suitable example / Define Abstract data type. Discuss the Operations on Structured Data.

An **Abstract Data Type (ADT)** in a Database Management System (DBMS) is a custom data type that groups related attributes to represent complex data structures, similar to classes in programming. ADTs, also called structured types, enable more organized data management by allowing the creation of user-defined types that mirror real-world objects.

In an **Object-Relational Database Management System (ORDBMS)**, ADTs allow users to define new data types directly in the database, providing flexibility and making data handling easier. This approach supports complex data needs by allowing databases to store user-defined types, such as addresses or contact details, as single objects instead of multiple separate fields.

Structured Data Types in ORDBMS

A **structured type** is a user-defined type that contains multiple attributes, each with its own data type, like CHAR, NUMBER, or another user-defined type. The syntax in SQL to create a structured type is similar to defining a class in programming:

```
          CREATE TYPE type_name AS OBJECT(
             Attribute1 data_type,
             Attribute2 data_type,
             ... );
```

Example of a Structured Type

Here’s an example to define a structured type for a phone number:

```
CREATE TYPE phone AS OBJECT
(
   Country_code NUMBER(4),
   STD_Code NUMBER(5),
   Phone_Number NUMBER(10)
);
```

This type can then be used in a **contact** table:

```
CREATE TABLE contact
(
   Contact_name VARCHAR(25),
   Street VARCHAR(25),
   City VARCHAR(25),
   Ph phone
);
```

Here, Ph uses the phone type, allowing the storage of a complete phone number object in one column.

Benefits of ADTs

1. **Modularity**: Structured types group related attributes, making data easier to manage and reuse across tables.
2. **Data Organization**: Data models reflect real-world objects more closely, improving readability and efficiency.
3. **Flexibility**: Complex data types can be defined and managed directly within the database, improving consistency and ease of use.

Operations on Structured Data

With structured types, you can perform operations like:

- **Insert**: Add new structured data (like a complete phone number) into a table.
- **Update**: Change specific attributes within a structured type, like updating only the Phone_Number.
- **Select**: Retrieve specific attributes within a structured type, such as querying only the Country_code.

4. Explain different data storing techniques in distributed DBMS

A **Distributed Database System** is a database spread across multiple computers or locations connected by a network, meaning it's not limited to just one physical location or computer. This setup enables users from different parts of the world to access the database as if it were a single, unified system. Even though the data is physically distributed, users interact with it as if it's one database, which enhances accessibility, performance, and reliability.

1. Fragmentation

Fragmentation involves dividing a database into smaller, manageable pieces or fragments, which can be stored across different locations. There are three main types of fragmentation:

- **Horizontal Fragmentation**:
  - This technique divides a table into subsets of rows, where each fragment contains a subset of the data. For instance, customer data can be fragmented based on geographical regions, such as North, South, East, and West. This approach optimizes data retrieval based on specific queries targeting particular subsets.
- **Vertical Fragmentation**:
  - This technique divides a table into subsets of columns. Each fragment contains a specific set of columns from the original table. For example, in a customer table, one fragment might include personal information (name, address) while another fragment contains transaction details (order history, payment method). This allows for efficient access when only certain attributes are needed.
- **Mixed Fragmentation**:
  - A combination of both horizontal and vertical fragmentation. This approach allows for more complex data distribution strategies tailored to specific application needs.

2. Replication

Replication involves storing copies of the same data at multiple locations to enhance availability and reliability. There are two primary replication strategies:

- **Synchronous Replication**:
  - Updates to data are immediately propagated to all replicas. This ensures data consistency across replicas but can introduce latency due to the time required for updates to be applied to all copies.
- **Asynchronous Replication**:
  - Updates are propagated to replicas at a later time. This can improve performance and reduce latency since updates can be processed without waiting for all replicas to synchronize. However, it may lead to temporary inconsistencies among replicas.

3. Data Distribution

Data distribution refers to the strategy used to place data fragments across different nodes in a distributed system. Techniques include:

- **Hash Partitioning**:
  - Data is distributed across nodes based on a hash function applied to a key attribute. This method ensures an even distribution of data but may complicate range queries, as related data may be stored on different nodes.
- **Range Partitioning**:
  - Data is divided based on ranges of values in a particular attribute. For example, customer IDs could be partitioned into ranges (1-1000, 1001-2000, etc.), which can simplify range queries but may lead to unbalanced loads if data is skewed.
- **List Partitioning**:
  - Specific values are assigned to particular partitions. For instance, customer records could be stored in different partitions based on the customer's country (USA, Canada, etc.). This technique allows for targeted data retrieval but can lead to complexity if the list of values changes frequently.

5 Differentiate between ORDBMS and OODBMS

Here's a table that outlines the key differences between Object-Relational Database Management Systems (ORDBMS) and Object-Oriented Database Management Systems (OODBMS):

Feature	ORDBMS	OODBMS
Definition	Combines relational and object-oriented features.	Focuses exclusively on storing objects.
Data Model	Relational model extended to include objects.	Pure object model.
Data Structure	Tables with rows and columns, plus objects.	Objects with attributes and methods.

Query Language	SQL with object-oriented extensions.	Object Query Language (OQL) or custom queries.
Inheritance Support	Limited inheritance (subtypes) allowed.	Full inheritance support for objects.
Complex Data Types	Supports complex data types (e.g., arrays, structs).	Direct support for complex data types.
Relationships	Foreign keys and references to link tables.	Objects reference other objects directly.
Schema Evolution	Schema changes can be complex due to tables.	Easier schema evolution due to dynamic nature.
Performance	Can be slower due to relational overhead.	Optimized for performance with object handling.
Transactions	ACID properties are maintained using SQL.	ACID properties maintained but may vary.
Data Access	Data accessed through SQL queries.	Data accessed through methods on objects.
Normalization	Normalization is a key concept.	Denormalization is common to reduce complexity.
Application Domain	Suitable for traditional applications needing complex data handling.	Ideal for applications with complex data relationships (e.g., CAD, multimedia).
Object Identity	Rows have a unique identifier (primary key).	Objects have a unique identity through references.
Development Complexity	Typically simpler for developers familiar with SQL.	More complex, requiring understanding of object-oriented principles.

6. Distinguish between parallel database and distributed database.

Here's a table that highlights the key differences between Parallel Databases and Distributed Databases:

Feature	Parallel Database	Distributed Database
Definition	A single database system that uses multiple processors to execute queries in parallel.	A database system that stores data across multiple locations, which may be physically or logically distributed.
Architecture	Typically has a shared storage architecture.	Can have shared or distributed storage architecture.
Data Location	Data is stored in a single location but processed in parallel.	Data is stored in multiple locations, potentially across different networks.

Goal	Aims to improve performance by utilizing parallel processing for faster query execution.	Aims to improve availability, reliability, and scalability by distributing data across multiple sites.
Query Processing	Uses parallel execution strategies to process queries faster.	Queries can be processed independently at different sites, often involving network communication.
Data Access	Accessed through a unified interface, typically via a single DBMS.	Accessed through various interfaces, depending on the location of the data.
Scalability	Scaling involves adding more processors to a single system.	Scaling involves adding more nodes or sites to the network.
Failure Handling	More susceptible to single points of failure due to the shared architecture.	Designed to handle failures by providing data redundancy and replication.
Data Redundancy	Typically does not have redundancy; focuses on parallel processing of data.	Often includes data redundancy for fault tolerance and availability.
Use Cases	Suitable for data-intensive applications requiring high performance (e.g., data warehousing).	Suitable for applications requiring high availability, geographic distribution, and fault tolerance (e.g., online transaction processing).
Complexity	Simpler in terms of management due to a unified system.	More complex due to data distribution and synchronization challenges.
Communication	Less communication overhead as all processes work on the same data set.	More communication overhead due to data exchange across different nodes.

Module 02

1.Define OLAP. Explain the different OLAP models with a suitable diagram

Online Analytical Processing (OLAP) is a category of software technology that enables analysts, managers, and executives to gain insight into data through fast, consistent, interactive access. OLAP allows users to perform multidimensional analysis of business data, helping in decision-making processes by providing the ability to view data from different perspectives.

Online Analytical Processing (OLAP) is a technology that helps users examine data from different angles and perspectives. It’s designed to make complex calculations, find trends, and create data models, allowing users to quickly and easily gain insights from large amounts of data. OLAP systems are important tools for decision-making and business intelligence, as they allow users to perform complex queries and create reports quickly and efficiently.

OLAP Models

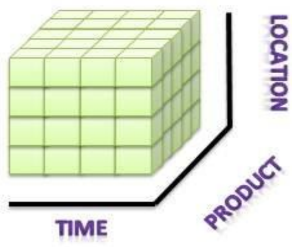
There are several OLAP models, but the two most commonly discussed are **ROLAP (Relational OLAP)** and **MOLAP (Multidimensional OLAP)**. Below is a detailed explanation of each model, including their characteristics, advantages, and diagrams.

1. ROLAP (Relational OLAP)

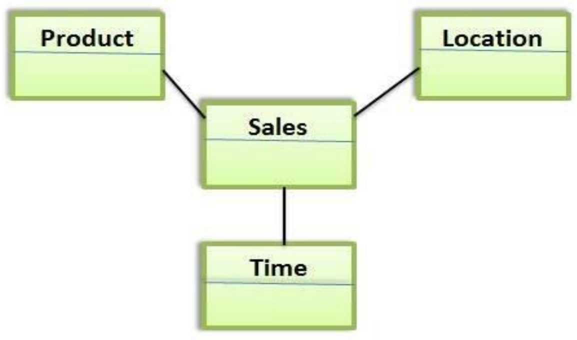
Definition:
ROLAP stands for Relational Online Analytical Processing. It stores data in relational databases and uses SQL for query processing. ROLAP systems rely on the underlying relational database management system (RDBMS) to store and manage the data, utilizing standard relational tables.

Characteristics:

- **Data Storage**: ROLAP stores data in traditional relational databases (e.g., Oracle, MySQL, SQL Server).
- **Data Model**: It uses star or snowflake schemas to organize data into facts and dimensions.
- **Query Language**: ROLAP primarily uses SQL for querying data.
- **Scalability**: ROLAP can handle large amounts of data efficiently, leveraging the capabilities of relational databases.
- **Dynamic Aggregation**: Calculated aggregations are created on-the-fly during query execution, allowing for flexibility.



MOLAP



ROLAP

2. MOLAP (Multidimensional OLAP)

Definition:
MOLAP stands for Multidimensional Online Analytical Processing. It stores data in a cube format, which makes it easy to retrieve and analyze data quickly.

Multidimensional OLAP (MOLAP) is a classical OLAP system that uses array-based storage engines to present data in a multidimensional format, often organized in an OLAP cube. MOLAP stores data that has been pre-calculated, pre-summarized, and arranged into a multidimensional array, which allows for rapid access and analysis. This structure provides users with the ability to view data from different perspectives and with various detail levels. Since MOLAP has all possible combinations of data pre-stored, it can quickly retrieve information, making it faster than Relational OLAP (ROLAP), which relies on dynamic queries. This speed advantage makes MOLAP ideal for applications that need quick insights from complex data.

Characteristics:

- **Data Storage**: Data is stored in multidimensional cubes that are designed for fast access and efficient storage of aggregated data.
- **Data Structure**: Data is organized into dimensions (categories like time, geography, product) and measures (numerical values).
- **Query Language**: Uses specialized query languages like MDX (Multidimensional Expressions) instead of standard SQL.
- **Pre-aggregation**: Data is often pre-aggregated, which speeds up query performance.

2 OLAP Vs OLTP

Here's an easy comparison between OLAP (Online Analytical Processing) and OLTP (Online Transaction Processing):

Feature	OLAP	OLTP
Definition	Used for analyzing and reporting data from multiple perspectives.	Used for managing daily transactions.
Data Source	Collects data from multiple sources, often data warehouses.	Uses operational, current data from databases.
Method Used	Multidimensional analysis.	Transactional, processing one record at a time.
Application	Business intelligence, decision support, and reporting.	Banking, order processing, and inventory management.
Normalization	Often denormalized to speed up complex queries.	Highly normalized to avoid redundancy.
Usage of Data	Historical data for trend analysis.	Real-time data for immediate processing.
Task	Complex analysis and reporting.	Short, simple tasks with frequent reads and writes.
Purpose	To analyze data and gain insights for decision-making.	To capture and manage current transactional data.
Volume of Data	Large volumes of data; often historical.	Smaller, current data sets.
Queries	Complex queries with large aggregations.	Simple, quick queries with fewer joins.
Update	Not frequently updated, mostly read-only.	Updated regularly, as it records transactions.
Backup and	Less frequent backups needed as	Regular backups are essential

Recovery	data is generally read-only.	to prevent data loss.
Processing Time	Takes longer due to complex calculations.	Fast, as it is designed for quick transaction processing.
Types of Users	Managers, analysts, and business intelligence users.	Clerks, frontline staff, and customers.
Operations	Read-heavy, involving aggregations and analysis.	Write-heavy, with frequent data insertions and updates.
Updates	Less frequent, as the data is historical and mainly for analysis.	Frequent updates, as new transactions are continuously recorded.

3 Star flake, snowflake and fact constellation Schema

A schema is the structural blueprint of a database, defining how data is organized, tables are related, and information is stored for querying and analysis. In data warehousing, schemas help organize data for efficient querying and analysis, supporting different analytical needs.

The Schema is divided into 3 types

1 **Star Flake 2. Snowflake 3. Fact constellation**

1. Star Schema

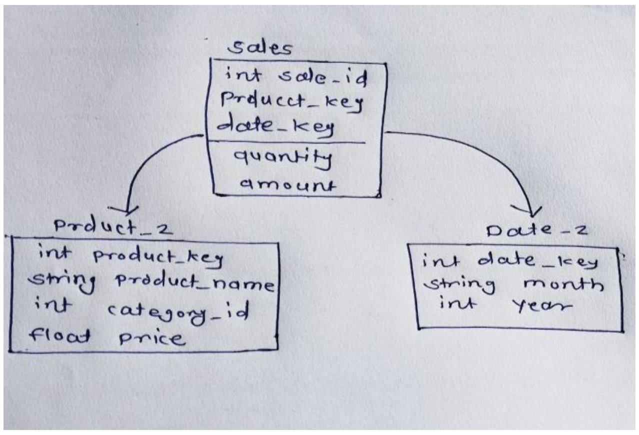
Definition:
The Star Schema is the simplest form of a data warehouse schema and is used widely in dimensional modeling. It is called a star schema because the diagram resembles a star, with a central fact table surrounded by dimension tables.

- **Fact Table**: Contains the measurable data, typically numerical values, that represent business processes, like sales amounts or quantities sold.
- **Dimension Tables**: Contain descriptive attributes, such as dates, products, customers, or locations, which help categorize and describe facts.

Structure:

- In a star schema, there is one central fact table connected to multiple dimension tables. Each dimension table is directly linked to the fact table without any intermediate tables.

Example:

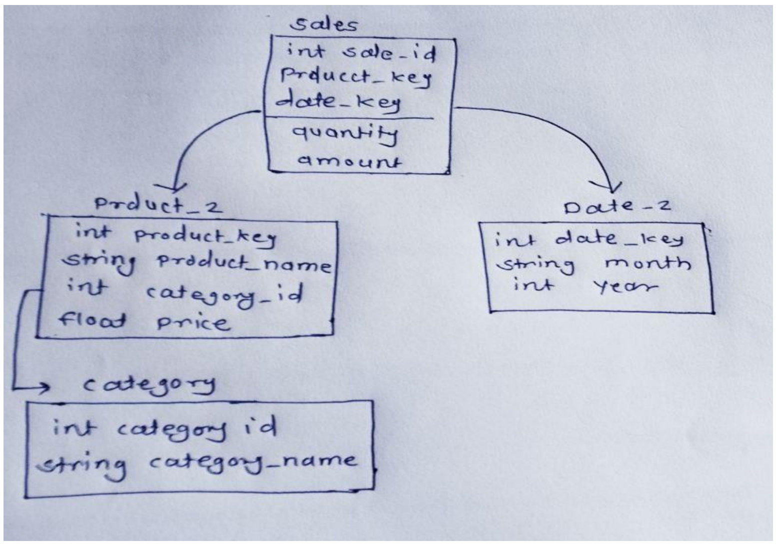


2. Snowflake Schema

Definition:
The Snowflake Schema is an extension of the star schema and contains additional layers of dimension tables that are normalized, meaning that dimension tables are divided into related tables to eliminate redundancy.

- **Fact Table**: Similar to the star schema, it contains numerical data for analysis.
- **Dimension Tables**: Dimension tables in a snowflake schema are normalized into multiple related tables, creating a snowflake-like structure around the central fact table.

Structure:

- In a snowflake schema, each dimension table is connected to the fact table, and some dimensions may also link to additional sub-dimensions.
- 

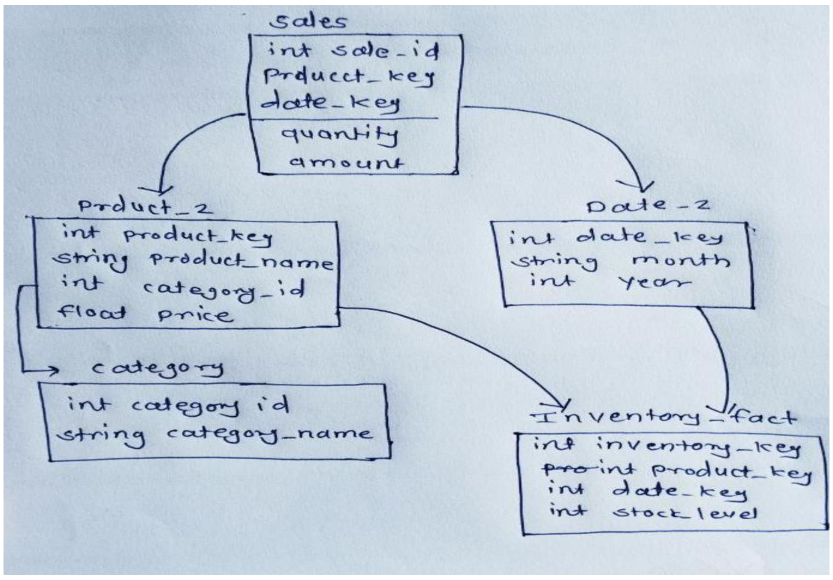
3. Fact Constellation Schema (Galaxy Schema)

Definition:
The Fact Constellation Schema, also known as the Galaxy Schema, is a schema that contains multiple fact tables sharing dimension tables. This schema is used when there are multiple business processes in a data warehouse, each represented by its own fact table. These fact tables can share common dimensions.

- **Fact Tables**: Multiple fact tables store data from different business processes.
- **Dimension Tables**: Shared and unique dimension tables help categorize data across different fact tables.

Structure:

- In a fact constellation schema, there are multiple fact tables that may be linked to both shared and individual dimension tables.



4. Describe in detail Data Warehouse architecture and ETL Process

A **Data Warehouse Architecture** is a system that defines how data is collected, processed, and presented for analysis and reporting within an organization. This architecture makes data easily accessible and organized, enabling users to analyze and interpret data to make informed decisions. Though each data warehouse setup varies, all follow some key principles and components.

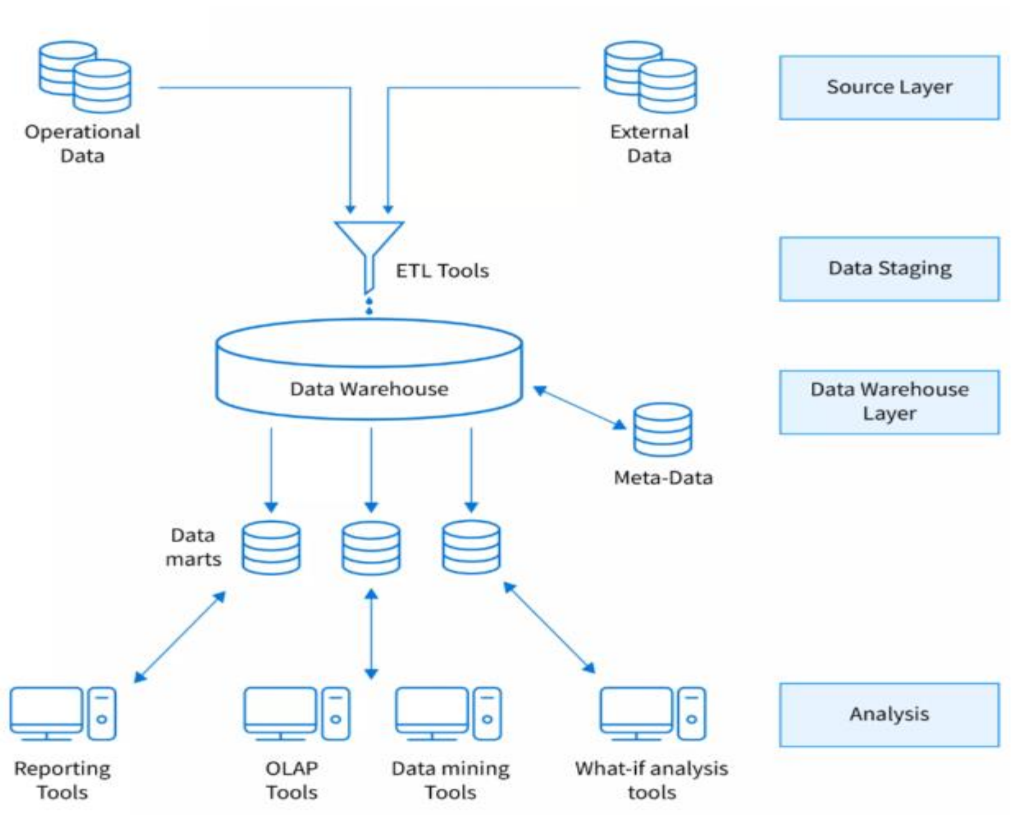
Data Warehouse Architecture defines how data is collected, processed, stored, and accessed within an organization to support analysis and decision-making. The architecture typically consists of three key layers

- **External Sources**:
  - Data is collected from various sources, both internal and external. These sources can contain different types of data, such as structured (databases), semi-structured (XML/JSON), or unstructured (emails, social media).
- **Staging Area**:
  - Data from these sources is initially stored in a temporary area known as the **staging area**. Here, the data is processed through **ETL (Extract, Transform, Load)** to ensure consistency and readiness for the data warehouse.
  - **Extract (E)**: Data is taken (extracted) from the various sources.
  - **Transform (T)**: The data is cleaned and converted into a common, standardized format.
  - **Load (L)**: The transformed data is then loaded into the data warehouse.
- **Data Warehouse**:
  - The data warehouse is a **central repository** where cleaned and organized data is stored. It holds data in its raw, detailed form and includes metadata (data about the

data). The data warehouse acts as the single source of truth for the organization, providing a unified view of data across departments.

 Data Marts:

- **Data marts** are created from the data warehouse to serve the needs of specific departments, such as finance or HR. They contain only a **subset** of the data stored in the data warehouse, focusing on information relevant to each area of the organization.



ELT process

The **ETL (Extract, Transform, Load) process** is crucial for filling and updating a data warehouse, ensuring that data is accurate, consistent, and useful for analysis. The ETL process has three main steps:

1. **Extract**:
   - In this phase, raw data is gathered from various sources. These could be databases, APIs, or simple flat files.
   - The extraction can involve gathering all the data (known as **full extraction**) or just the new data that has changed since the last extraction (known as **incremental extraction**).
2. **Transform**:
   - In the transformation stage, data is cleaned and processed to ensure accuracy and readiness for analysis.
   - **Common transformations** include data cleaning (removing duplicates and fixing errors), normalization (standardizing values), aggregation (such as

summing totals or finding averages), and format conversion (such as adjusting dates or standardizing units).
    - This phase also applies **business rules** so the data aligns with company standards.
3. **Load**:
    - The final phase is to load the transformed data into the data warehouse.
    - Loading can be done in bulk (all at once) or incrementally (only loading new or changed data).
    - Once loaded, the data is now available for analysis and reporting.

Example of the ETL Process: Suppose an e-commerce company wants to analyze customer data from various sales channels.

- **Extract**: Data is extracted from customer orders, website logs, and social media.
- **Transform**: The data is cleaned (duplicates are removed), addresses are standardized, and data is grouped by region.
- **Load**: The transformed data is loaded into the data warehouse, where business intelligence (BI) tools can analyze it to find customer purchasing patterns.

5. Define OLAP. Illustrate with an example different OLAP operations: Roll-Up, Drill- Down, Slice, Dice and Pivot

Online Analytical Processing (OLAP) is a category of software technology that enables analysts, managers, and executives to gain insight into data through fast, consistent, interactive access. OLAP allows users to perform multidimensional analysis of business data, helping in decision-making processes by providing the ability to view data from different perspectives.

Online Analytical Processing (OLAP) is a technology that helps users examine data from different angles and perspectives. It’s designed to make complex calculations, find trends, and create data models, allowing users to quickly and easily gain insights from large amounts of data. OLAP systems are important tools for decision-making and business intelligence, as they allow users to perform complex queries and create reports quickly and efficiently.

 Roll-Up:

- Roll-up is also known as "consolidation" or "aggregation."
- The Roll-up operation can be performed in two ways: by reducing dimensions and by climbing up the concept hierarchy.
- The concept hierarchy is a system of grouping things based on their order or level.
- The ROLLUP operation allows a SELECT statement to calculate multiple levels of subtotals across a specified group of dimensions and also calculates a grand total.
- ROLLUP is a simple extension to the GROUP BY clause, making its syntax straightforward and easy to use.
- The ROLLUP extension is highly efficient, adding minimal overhead to a query, which improves performance in data analysis.

 Drill-Down:

- Drill-down is the process of navigating from less detailed data to more detailed data.

- It allows users to view the data at a finer granularity, providing deeper insights into the dataset.
- Drill-down can be achieved by moving down a concept hierarchy or by breaking down aggregated data into its constituent parts.
- This operation enables users to analyze specific segments of data, such as examining monthly sales after viewing yearly totals.
- It enhances decision-making by allowing users to explore the underlying factors that contribute to higher-level summaries.

 Slice:

- The slice operation selects a single dimension from a multidimensional data cube, allowing users to focus on a specific data subset.
- It essentially creates a new sub-cube by fixing one dimension while keeping the others intact.
- The slice operation helps in analyzing a particular aspect of the data, such as sales performance for a specific product category over a defined time period.
- This operation simplifies data exploration by narrowing down the dataset to relevant information, making analysis more straightforward.

 Dice:

- Dice is a more complex operation that involves selecting a specific subset of data from a data cube by applying multiple filters on multiple dimensions.
- This operation enables users to create a smaller data cube that contains only the desired dimensions and values.
- For example, users can dice the data to view sales figures for a specific product category across multiple regions during certain time periods.
- Dice allows for a more targeted analysis, helping organizations to focus on specific metrics that matter most to their objectives.

 Pivot (Rotate):

- Pivot, also known as rotation, involves rearranging the data presentation in a data cube to provide a different perspective on the data.
- This operation allows users to switch the rows and columns in a data view, facilitating the exploration of data relationships.
- Pivoting helps in visualizing trends and patterns that may not be apparent in the original data layout.
- By changing the orientation of the data, users can gain new insights, making it easier to compare metrics across different dimensions.

  (Refer notbook example for all operations)

Module 03

1. Explain different pre-processing techniques in detail

Data preprocessing is the first step in data mining, where raw data is organized and transformed into a format that's ready for analysis. Often, data is collected from different sources and may contain issues such as noise, missing values, or inconsistencies. These issues need to be resolved to ensure that the data is clean and structured for effective analysis.

This preprocessing phase is crucial because the quality of the data directly impacts the accuracy and usefulness of the final mining results. Properly preprocessed data allows for more reliable insights and patterns to be discovered, making the data mining process both accurate and effective.

Steps of Data Preprocessing

Data cleaning

Data cleaning help us remove inaccurate, incomplete and incorrect data from the dataset. Some techniques used in data cleaning are −

Handling missing values

This type of scenario occurs when some data is missing.

- Standard values can be used to fill up the missing values in a manual way but only for a small dataset.
- Attribute's mean and median values can be used to replace the missing values in normal and non-normal distribution of data respectively.
- Tuples can be ignored if the dataset is quite large and many values are missing within a tuple.
- Most appropriate value can be used while using regression or decision tree algorithms

Noisy Data

Noisy data are the data that cannot be interpreted by machine and are containing unnecessary faulty data. Some ways to handle them are −

- **Binning** − This method handle noisy data to make it smooth. Data gets divided equally and stored in form of bins and then methods are applied to smoothing or completing the tasks. The methods are Smoothing by a bin mean method(bin values are replaced by mean values), Smoothing by bin median(bin values are replaced by median values) and Smoothing by bin boundary(minimum/maximum bin values are taken and replaced by closest boundary values).

- **Regression** − Regression functions are used to smoothen the data. Regression can be linear(consists of one independent variable) or multiple(consists of multiple independent variables).
- **Clustering** − It is used for grouping the similar data in clusters and is used for finding outliers.

Data integration

The process of combining data from multiple sources (databases, spreadsheets,text files) into a single dataset. Single and consistent view of data is created in this process. Major problems during data integration are Schema integration(Integrates set of data collected from various sources), Entity identification(identifying entities from different databases) and detecting and resolving data values concept.

Data transformation

In this part, change in format or structure of data in order to transform the data suitable for mining process. Methods for data transformation are −

- **Normalization** − Method of scaling data to represent it in a specific smaller range(-1.0 to 1.0)
- **Discretization** − It helps reduce the data size and make continuous data divide into intervals.
- **Attribute Selection** − To help the mining process, new attributes are derived from the given attributes.
- **Concept Hierarchy Generation** − In this, the attributes are changed from lower level to higher level in hierarchy.
- **Aggregation** − In this, a summary of data gets stored which depends upon quality and quantity of data to make the result more optimal.

Data reduction

It helps in increasing storage efficiency and reducing data storage to make the analysis easier by producing almost the same results. Analysis becomes harder while working with huge amounts of data, so reduction is used to get rid of that.

Steps of data reduction are −

- ***Data Compression*** : Data is compressed to make efficient analysis. Lossless compression is when there is no loss of data while compression. loss compression is when unnecessary information is removed during compression.

- ***Numerosity Reduction***

There is a reduction in volume of data i.e. only store model of data instead of whole data, which provides smaller representation of data without any loss of data.

- ***Dimensionality reduction***

  In this, reduction of attributes or random variables are done so as to make the data set dimension low. Attributes are combined without losing its original characteristics.

2. Explain Knowledge Discovery Process (KDD) in detail. What is the role of data mining in KDD process?

In computer science, "Data Mining" is the process of discovering valuable information and patterns in large datasets. It's also known as knowledge extraction, pattern analysis, data archaeology, or data dredging. Data Mining, often called Knowledge Discovery in Databases (KDD), involves finding hidden, previously unknown, and potentially useful information from databases.

The purpose of data mining is to analyze large datasets and find meaningful patterns that can help in making predictions or supporting better decisions. Today, data mining is widely used in many areas where large amounts of data are stored, such as banking, retail, and network security.

KDD Process

The Knowledge Discovery in Databases (KDD) process involves finding useful, unknown, and valuable information from large datasets. This process is iterative, meaning it repeats several times to accurately extract knowledge from the data. Through steps like data selection, cleaning, transformation, data mining, and pattern evaluation, KDD refines and organizes data into valuable insights.

Here's a detailed breakdown of the KDD process:

1. Data Selection

- This initial step identifies and selects relevant data from a larger pool, focusing only on the data needed for analysis.
- The selected data could come from various sources, such as databases, data warehouses, or external sources.
- The aim is to focus on data that is relevant to the problem at hand, which helps reduce processing time and improves efficiency.

2. Data Preprocessing (Cleaning and Integration)

- **Data Cleaning**: The data is cleaned to address any errors or inconsistencies, like missing values, outliers, or duplicate data. Cleaning ensures the data is accurate, complete, and consistent.
- **Data Integration**: Data from multiple sources is integrated to create a single dataset. This is important when the data comes from different sources with different formats or structures.

3. Data Transformation

- The next step is transforming the data into a suitable format for mining. This often includes:
    - **Normalization**: Scaling numerical data to a consistent range (e.g., 0 to 1) for easier analysis.
    - **Discretization**: Converting continuous variables into categories or bins to simplify the data.
    - **Attribute Creation**: Creating new attributes or features that may be helpful for analysis based on existing attributes.

4. Data Reduction

- Here, the data is reduced to retain only the most relevant information, reducing the amount of data to be processed while preserving essential patterns and trends.
- Techniques include **feature selection** (choosing only the relevant features), **feature extraction** (creating new features based on original data), and **dimensionality reduction** (like PCA) to simplify data.

5. Data Mining

- This is the core step where specific algorithms and techniques are applied to extract patterns, trends, or relationships from the processed data.
- Data mining can be divided into:
    - **Classification**: Sorting data into predefined categories (e.g., spam vs. not spam).
    - **Clustering**: Grouping data into clusters based on similarity, without predefined labels.
    - **Association**: Finding relationships between variables, commonly used for market basket analysis.
    - **Regression**: Estimating relationships between variables, often used for predictive analytics.

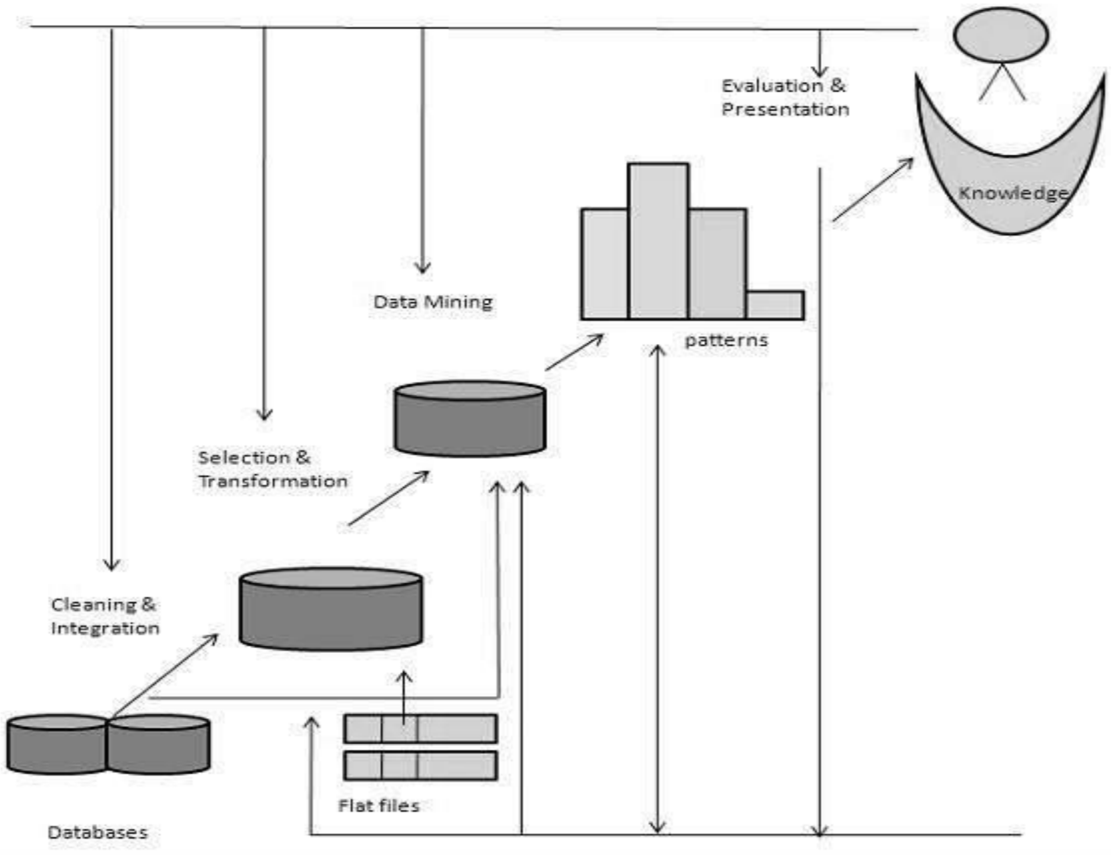
6. Pattern Evaluation

- In this step, the discovered patterns are evaluated to determine their relevance and usefulness.
- Only patterns that provide meaningful insights are selected, ensuring that the final knowledge gained is relevant to the business or research objectives.
- Metrics like accuracy, relevance, and novelty are used to evaluate the patterns.

7. Knowledge Presentation

- The final step involves presenting the extracted knowledge in a form that stakeholders can easily interpret and use.
- Common formats include **charts, graphs, reports, or dashboards**. Data visualization techniques make complex patterns easier to understand.
- This step is crucial for turning raw data insights into actionable knowledge that can be used for decision-making.

Architecture of KDD:



3. Explain Data reduction techniques.

Data reduction techniques are essential in data mining and data analysis as they help reduce the volume of data while preserving its integrity and essential characteristics. This is crucial for improving the efficiency of data processing and analysis, especially when dealing with large datasets. Here are some common data reduction techniques:

Methods of Data Reduction

Data reduction techniques help simplify large datasets while retaining essential information. Here are some of the key methods:

1. Data Cube Aggregation

Data cube aggregation simplifies data by summarizing it into a more manageable form. For example, if you have revenue data for a company every quarter from 2012 to 2014, instead of analyzing quarterly figures, you could aggregate this data to show total annual sales. This means you summarize the data to get a yearly total rather than dealing with more detailed quarterly data.

2. Dimension Reduction

Dimension reduction eliminates unnecessary or redundant data attributes that are not important for analysis, thus reducing the overall data size. Here are two techniques for dimension reduction:

- **Step-wise Forward Selection**: This method starts with no attributes and gradually adds the most relevant ones based on their importance. For example:
    - Start with an empty set.
    - Add attributes one by one based on their significance.
    - Final reduced set might include only the most relevant attributes.
- **Step-wise Backward Selection**: This technique starts with all attributes and removes the least important ones. For example:
    - Start with the full set of attributes.
    - Remove the least relevant attributes one by one.
    - The final reduced set will consist of only the most useful attributes.
- **Combination of Forward and Backward Selection**: This method combines both techniques, allowing for the removal of the least useful attributes while selecting the best ones to optimize the dataset efficiently.

3. Data Compression

Data compression reduces the size of files using various encoding methods. There are two main types:

- **Lossless Compression**: This type retains all original data, allowing for perfect restoration after compression. Techniques include Run-Length Encoding (RLE), where repeated data is stored in a more compact form.
- **Lossy Compression**: This method reduces file size by removing some data permanently, which might not significantly impact usability. An example is JPEG image compression, where some details are lost, but the image remains useful.

4. Numerosity Reduction

In numerosity reduction, actual data is replaced with mathematical models or simpler representations. This approach can involve:

- **Clustering**: Grouping similar data points together.
- **Histograms**: Summarizing data distributions.
- **Sampling**: Using a subset of data for analysis rather than the entire dataset.

5. Discretization & Concept Hierarchy Operation

Discretization divides continuous data into intervals, making it easier to analyze. This can involve:

- **Top-Down Discretization**: Starting with a few key points (breakpoints) to split the data and refining it progressively.

- **Bottom-Up Discretization**: Beginning with all constant values as potential split points and then merging them based on neighboring values to form meaningful intervals.