CHENNAI INSTITUTE OF TECHNOLOGY
(Autonomous)
(Affiliated to Anna University, Approved by AICTE, Accredited by NAAC & NBA)
Sarathy Nagar, Kundrathur, Chennai – 600069, India.

Lecture Notes

U N I T - I I I

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS341 DATA WAREHOUSING

III YEAR/VI SEMESTER

UNIT III META DATA, DATA MART AND PARTITION STRATEGY

Meta Data – Categories of Metadata – Role of Metadata – Metadata Repository – Challenges for Meta Management - Data Mart – Need of Data Mart- Cost Effective Data Mart- Designing Data Marts- Cost of Data Marts- Partitioning Strategy – Vertical partition – Normalization – Row Splitting– Horizontal Partition

3.1 META DATA:

- **Definition:**

  Metadata refers to data about data that provides information about other data. It includes details about the structure, content, and context of data, characteristics, attributes, and properties of other data.

- 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
- **Significance:** Metadata is crucial for data management, aiding in understanding, interpreting, and managing data effectively.

Metadata is data about the data or documentation about the information which is required by the users. In data warehousing, metadata is one of the essential aspects.

Metadata includes the following:

1. The location and descriptions of warehouse systems and components.
2. Names, definitions, structures, and content of data-warehouse and end-users views.
3. Identification of authoritative data sources.
4. Integration and transformation rules used to populate data.
5. Integration and transformation rules used to deliver information to end-user analytical tools.
6. Subscription information for information delivery to analysis subscribers.
7. Metrics used to analyze warehouses usage and performance.
8. Security authorizations, access control list, etc.

Metadata is used for building, maintaining, managing, and using the data warehouses. Metadata allow users access to help understand the content and find data.

Several examples of metadata are:

1. A library catalog may be considered metadata. The directory metadata consists of several predefined components representing specific attributes of a resource, and each item can have one or more values. These components could be the name of the author, the name of the document, the publisher's name, the publication date, and the methods to which it belongs.
2. The table of content and the index in a book may be treated metadata for the book.
3. Suppose we say that a data item about a person is 80. This must be defined by noting that it is the person's weight and the unit is kilograms. Therefore, (weight, kilograms) is the metadata about the data is 80.
4. Another examples of metadata are data about the tables and figures in a report like this book. A table (which is a record) has a name (e.g., table titles), and there are column names of the tables that may be treated metadata. The figures also have titles or names.

Why is metadata necessary in a data warehouses?

- First, it acts as the glue that links all parts of the data warehouses.
- Next, it provides information about the contents and structures to the developers.

- Finally, it opens the doors to the end-users and makes the contents recognizable in their terms.

Metadata is Like a **Nerve Center**. Various processes during the building and administering of the data warehouse generate parts of the data warehouse metadata. Another uses parts of metadata generated by one process. In the data warehouse, metadata assumes a key position and enables communication among various methods. It acts as a nerve centre in the data warehouse.

- For example, a digital image's metadata can include information related to the image's resolution, size, color depth, and time of creation.
- This information can be used for data classification, labeling, organization, sorting, tracking, searching, and analysis.
- This is part of a series of articles about data security.

File metadata:
This includes information about a **file, such as its name, size, type, and creation date**.
Image metadata:
This includes information about an **image, such as its resolution, color depth, and camera settings**.
Music metadata:
This includes information about a **piece of music, such as its title, artist, album, and genre**.
Video metadata:
This includes information about a **video, such as its length, resolution, and frame rate**.
Document metadata:
This includes information about a **document, such as its author, title, and creation date**.
Database metadata:
This includes information about a **database, such as its structure, tables, and fields.**
Web metadata:
This includes information about a **web page, such as its title, keywords, and description**.

3.2 CATEGORIES OF METADATA:

Structural Metadata:
Describes the structure of data, including database tables, field types, and relationships.
provides information about the relationships and organization of data, and may include elements such as links, tables of contents, and indices.
Structural metadata helps to organize and connect data and can be used to facilitate the navigation and discovery of data.
Descriptive Metadata:
Provides information about the content of data, such as titles, descriptions, and keywords.
This type of metadata provides information about the content, structure, and format of data, and may include elements such as title, author, subject, and keywords.
Descriptive metadata helps to identify and describe the content of data and can be used to improve the discoverability of data through search engines and other tools.
Reference Metadata: Establishes relationships between different data elements, like foreign keys in databases.
Administrative metadata:
Manages data, covering ownership, access controls, and versioning. This type of metadata provides information about the management and technical characteristics of data, and may include elements such as file format, size, and creation date. Administrative metadata helps to manage and maintain data over time and can be used to support data governance and preservation.

Provenance metadata: This type of metadata provides information about the history and origin of data, and may include elements such as the creator, date of creation, and sources of data. Provenance metadata helps to provide context and credibility to data and can be used to support data governance and preservation.
Rights metadata: This type of metadata provides information about the ownership, licensing, and access controls of data, and may include elements such as copyright, permissions, and terms of use. Rights metadata helps to manage and protect the intellectual property rights of data and can be used to support data governance and compliance.
Educational metadata: This type of metadata provides information about the educational value and learning objectives of data, and may include elements such as learning outcomes, educational levels, and competencies. Educational metadata can be used to support the discovery and use of educational resources, and to support the design and evaluation of learning environments.
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. Metadata can be stored in various forms, such as text, XML, or RDF.

3.3. ROLE OF METADATA:

Metadata has a very important role in a data warehouse. The role of metadata in a warehouse is different from the warehouse data, yet it plays an important role. The various roles of metadata are explained below.

- Metadata acts as a directory.
- This directory helps the decision support system to locate the contents of the data warehouse.
- Metadata helps in decision support system for mapping of data when data is transformed from operational environment to data warehouse environment.
- Metadata helps in summarization between current detailed data and highly summarized data.
- Metadata also helps in summarization between lightly detailed data and highly summarized data.
- Metadata is used for query tools.
- Metadata is used in extraction and cleansing tools.
- Metadata is used in reporting tools.
- Metadata is used in transformation tools.

Identification:

- Metadata provides essential information to identify and locate resources.
- It includes details such as title, author, date created, and unique identifiers.

Description:

- Metadata describes the content, context, and structure of resources.
- It includes elements such as keywords, abstracts, summaries, and classifications.

Discovery:

- Metadata facilitates resource discovery by enabling users to search, browse, and retrieve information efficiently.
- Search engines and databases use metadata to index and rank content, making it easier for users to find relevant resources.

Interoperability:

- Metadata enhances interoperability by standardizing the representation and exchange of data.
- Common metadata standards and formats enable systems and applications to share and integrate information seamlessly.

Preservation:

- Metadata supports long-term preservation and management of digital resources.
- It includes information about the provenance, versioning, rights management, and archival status of resources.

Access Control:

- Metadata can be used to manage access to resources by specifying permissions, restrictions, and usage policies.
- It includes details such as access rights, authentication requirements, and usage terms.

Quality Assurance:

- Metadata helps ensure the quality and integrity of data by documenting its source, accuracy, and reliability.
- It includes information about data validation, quality metrics, and lineage.

Workflow Management:

- Metadata facilitates workflow management by tracking the lifecycle of resources and recording actions taken on them. It includes details such as creation, modification, deletion, and transfer of resources.
- Overall, metadata serves as a critical foundation for organizing, managing, and accessing information in various domains, including libraries, archives, digital repositories, and content management systems.

Data Integration:

- Metadata helps in integrating data from multiple sources into the data warehouse.

- It provides information about the structure, format, and meaning of the data, enabling developers to understand
- how different datasets relate to each other.

Data Understanding and Documentation:

- Metadata provides documentation about the data stored in the warehouse, including its source, transformation rules, and business meaning.
- This documentation helps data analysts and business users understand the content and context of the data, facilitating its interpretation and analysis

Data Lineage:

- Metadata tracks the lineage of data, documenting its origins, transformations, and movements throughout the data warehouse environment.
- This lineage information is crucial for auditing, troubleshooting, and ensuring data quality and regulatory compliance.

Data Governance and Compliance:

- Metadata supports data governance initiatives by providing information about data ownership, usage policies, and compliance requirements.
- It helps organizations enforce data standards, privacy regulations, and security controls across the data warehouse ecosystem.

Query Optimization and Performance Tuning:

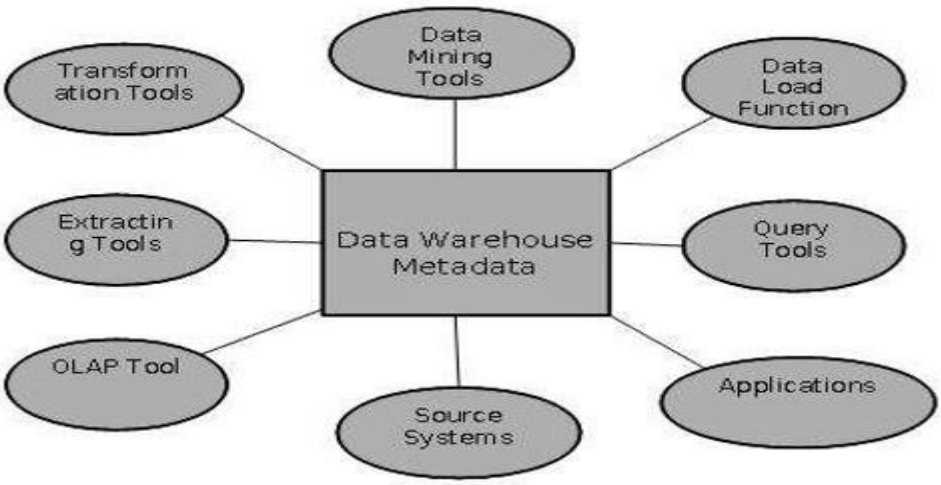
- Metadata stores statistics and indexes that describe the characteristics and distribution of data in the warehouse.
- Analyzing this metadata allows database administrators to optimize query execution plans, improve performance, and allocate resources effectively.

Data Quality Management:

- Metadata includes information about data quality metrics, validation rules, and cleansing processes applied to the data.
- Monitoring this metadata helps organizations identify and address data quality issues proactively, ensuring the accuracy and reliability of analytical insights.

Metadata Repository Management:

- Metadata repositories serve as centralized catalogs or databases that store metadata definitions, schemas, and relationships.
- Managing these repositories effectively is essential for maintaining data consistency, version control, and accessibility across the data warehouse environment.
- The following diagram shows the roles of metadata.



Metadata plays an important role in loading functions

Data Governance: Metadata is essential for establishing and enforcing data governance policies.

Data Lineage: Tracks the origin, transformation, and movement of data over its lifecycle.

Data Quality Management: Supports efforts to maintain and improve the quality of data.

Identification:

- Metadata provides essential information to identify and locate resources.
- It includes details such as title, author, date created, and unique identifiers.

Description:

- Metadata describes the content, context, and structure of resources.
- It includes elements such as keywords, abstracts, summaries, and classifications.

Discovery:

- Metadata facilitates resource discovery by enabling users to search, browse, and retrieve information efficiently.
- Search engines and databases use metadata to index and rank content, making it easier for users to find relevant resources.

Interoperability:

- Metadata enhances interoperability by standardizing the representation and exchange of data.
- Common metadata standards and formats enable systems and applications to share and integrate information seamlessly.

Preservation:

- Metadata supports long-term preservation and management of digital resources.
- It includes information about the provenance, versioning, rights management, and archival status of resources.

Access Control:

- Metadata can be used to manage access to resources by specifying permissions, restrictions, and usage policies.
- It includes details such as access rights, authentication requirements, and usage terms.

Quality Assurance:

- Metadata helps ensure the quality and integrity of data by documenting its source, accuracy, and reliability.
- It includes information about data validation, quality metrics, and lineage.

Workflow Management:

- Metadata facilitates workflow management by tracking the lifecycle of resources and recording actions taken on them. It includes details such as creation, modification, deletion, and transfer of resources.
- Overall, metadata serves as a critical foundation for organizing, managing, and accessing information in various domains, including libraries, archives, digital repositories, and content management systems.

3.4. METADATA REPOSITORY:

Definition: A metadata repository is a centralized database or system that stores and manages metadata.

- A metadata repository is a software tool that stores descriptive information about the data model used to store and share metadata.
- Metadata repositories combine diagrams and text, enabling metadata integration and change.

Function: It organizes metadata, facilitates search and retrieval, ensures integrity, and supports metadata security.

A metadata repository is a software tool that stores descriptive information about the data model used to store and share metadata.

- Metadata repositories combine diagrams and text, enabling metadata integration and change.
- Metadata repository serves as a centralized catalog or database that stores metadata related to the data warehouse environment.
- This repository plays a crucial role in managing and leveraging metadata for various data management and analytics tasks.

- Here's how a metadata repository functions within a data warehouse:
- **Centralized Storage:** The metadata repository serves as a centralized location for storing metadata related to data sources, data transformations, data models, business rules, and other artifacts within the data warehouse ecosystem. This centralized storage facilitates easy access to metadata by users and applications across the organization.
- **Metadata Capture:** The repository captures metadata from various sources within the data warehouse environment, including data sources, ETL (Extract, Transform, Load) processes, data models, reports, queries, and business glossaries. This metadata capture process may involve automated tools, manual entry, or a combination of both.
- **Metadata Management:** The repository provides capabilities for managing metadata definitions, schemas, relationships, and versioning. It allows users to define and customize metadata attributes, classifications, and hierarchies based on their specific requirements. Metadata management functionalities enable organizations to maintain data consistency, integrity, and governance across the data warehouse environment.
- **Metadata Search and Discovery**: The repository supports metadata search and discovery functionalities, allowing users to search for and explore metadata based on various criteria such as data source, data type, business domain, and usage context.

Metadata search capabilities help users locate relevant data assets, understand their characteristics, and assess their suitability for analytical and reporting purposes.

- **Metadata Lineage and Impact Analysis:** The repository facilitates metadata lineage and impact analysis, enabling users to trace the origins, transformations, and usage of data assets within the data warehouse environment. Metadata lineage functionalities help users understand how data flows through the system, while impact analysis capabilities allow them to assess the potential impact of changes to data sources, transformations, or schema structures.
- **Metadata Integration and Interoperability:** The repository supports integration with other data management tools and systems, enabling seamless exchange and interoperability of metadata across the organization. Integration with data integration tools, data modeling tools, BI (Business Intelligence) platforms, and data governance solutions ensures consistency and alignment of metadata definitions and processes.
- **Metadata Governance and Security:** The repository enforces metadata governance policies and security controls to ensure the confidentiality, integrity, and availability of metadata assets. It supports role-based access control, data masking, encryption, and audit logging to protect sensitive metadata and prevent unauthorized access or modification.

3.5 CHALLENGES FOR METADATA MANAGEMENT

The importance of metadata can not be overstated. Metadata helps in driving the accuracy of reports, validates data transformation, and ensures the accuracy of calculations. Metadata also enforces the definition of business terms to business end-users. With all these uses of metadata, it also has its challenges. Some of the challenges are discussed below.

- Metadata in a big organization is scattered across the organization. This metadata is spread in spreadsheets, databases, and applications.
- Metadata could be present in text files or multimedia files. To use this data for information management solutions, it has to be correctly defined.
- There are no industry-wide accepted standards. Data management solution vendors have narrow focus.
- There are no easy and accepted methods of passing metadata.

Metadata Interchange Initiative

The metadata interchange initiative was proposed to bring industry vendors and user together to address a variety of severe problems and issues concerning exchanging, sharing, and managing metadata. The goal of metadata interchange standard is to define an extensible mechanism that will allow the vendor to exchange standard metadata as well as carry along "proprietary" metadata. The founding members agreed on the following initial goals:

1. Creating a vendor-independent, industry-defined, and maintained standard access mechanisms and application programming interfaces (API) for metadata.
2. Enabling users to control and manage the access and manipulation of metadata in their unique environment through the use of interchange standards-compliant tools.
3. Users are allowed to build tools that meet their needs and also will enable them to adjust accordingly to those tools configurations.
4. Allowing individual tools to satisfy their metadata requirements freely and efficiently within the content of an interchange model.

5. Describing a simple, clean implementation infrastructure which will facilitate compliance and speed up adoption by minimizing the amount of modification.
6. To create a procedure and process not only for maintaining and establishing the interchange standard specification but also for updating and extending it over time.

Metadata Interchange Standard Framework

Interchange standard metadata model implementation assumes that the metadata itself may be stored in storage format of any type: ASCII files, relational tables, fixed or customized formats, etc.

It is a framework that is based on a framework that will translate an access request into the standard interchange index.

Several approaches have been proposed in metadata interchange coalition:

- Procedural Approach
- ASCII Batch Approach
- Hybrid Approach

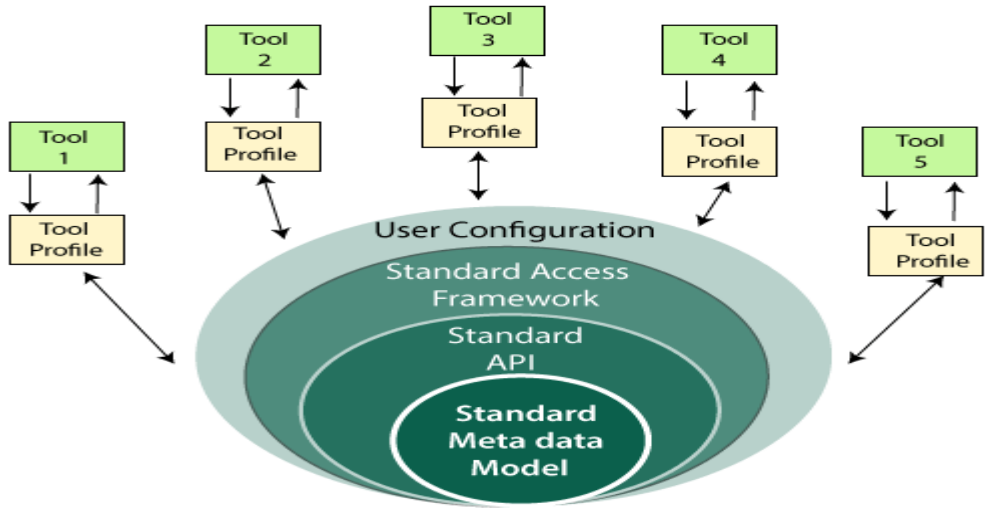
In a **procedural approach**, the communication with API is built into the tool. It enables the highest degree of flexibility.

In **ASCII Batch approach**, instead of relying on ASCII file format which contains information of various metadata items and standardized access requirements that make up the interchange standards metadata model.

In the **Hybrid approach**, it follows a data-driven model.

Components of Metadata Interchange Standard Frameworks

1) **Standard Metadata Model**: It refers to the ASCII file format, which is used to represent metadata that is being exchanged.



Metadata Interchange Standard Framework

2) The **standard access framework** that describes the minimum number of API functions.

3) Tool profile, which is provided by each tool vendor.

4) The user configuration is a file explaining the legal interchange paths for metadata in the user's environment.

Benefits of Metadata Repository

1. It provides a set of tools for enterprise-wide metadata management.
2. It eliminates and reduces inconsistency, redundancy, and underutilization.
3. It improves organization control, simplifies management, and accounting of information assets.
4. It increases coordination, understanding, identification, and utilization of information assets.
5. It enforces CASE development standards with the ability to share and reuse metadata.
6. It leverages investment in legacy systems and utilizes existing applications.
7. It provides a relational model for heterogeneous RDBMS to share information.
8. It gives useful data administration tool to manage corporate information assets with the data dictionary.

It increases reliability, control, and flexibility of the application development process

CHALLENGES FOR METADATA MANAGEMENT:

Metadata Management Challenges

- Disparate Information Sources
- Enforcing Business Rules for Metadata
- Data Quality and Accuracy
- Data Governance
- Effective Communication
- Additional Challenges
- The Bottom Line

Metadata Inconsistency: Inaccurate or outdated metadata across systems.

Metadata Sprawl: Proliferation of metadata repositories without proper governance.

Lack of Metadata Governance: Absence of policies for managing metadata lifecycle and quality.

Managing metadata poses several challenges, primarily due to the complexity and diversity of data environments. Here are some common challenges:

Volume and Variety:

With the proliferation of data sources, formats, and types, managing metadata for diverse data assets can be overwhelming.

Interoperability:

Ensuring interoperability between different metadata standards, schemas, and vocabularies is a significant challenge. Organizations may struggle to map and reconcile metadata across heterogeneous systems, tools, and platforms, leading to inconsistencies and gaps in metadata coverage.

Quality and Accuracy:

Maintaining the quality and accuracy of metadata is essential for effective data management and decision-making. However, ensuring data consistency, completeness, and relevance across metadata repositories can be difficult, particularly in dynamic and rapidly changing environments.

Metadata Discovery and Access:

Discovering and accessing relevant metadata assets can be challenging, especially in large and complex data environments. Organizations must invest in metadata search, cataloging, and navigation tools to facilitate metadata discovery and exploration for users across the organization.

Security and Privacy:

Protecting sensitive metadata from unauthorized access, disclosure, or misuse is a significant concern for organizations, particularly in regulated industries.

Implementing robust security controls, encryption mechanisms, and access policies is essential to safeguarding metadata assets.

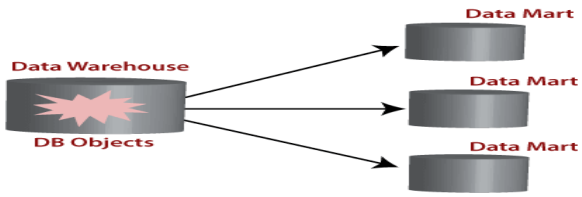
Scalability and Performance: As metadata volumes grow and data environments become more complex, scalability and performance become significant challenges. Organizations must invest in scalable metadata management solutions and infrastructure to handle increasing metadata workloads effectively.

3.6 DATA MART:

Definition: A data mart is a subset of a data warehouse, focusing on specific business areas or departments.

Purpose: It provides targeted data for departmental or specialized analytics, improving decision-making.

- A data ma rt is a simple form of **data warehouse focused on a single subject** or line of business.
- With a data mart, teams can access data and gain insights faster
- spend time searching within a more complex data warehouse or manually aggregating data from different sources.



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

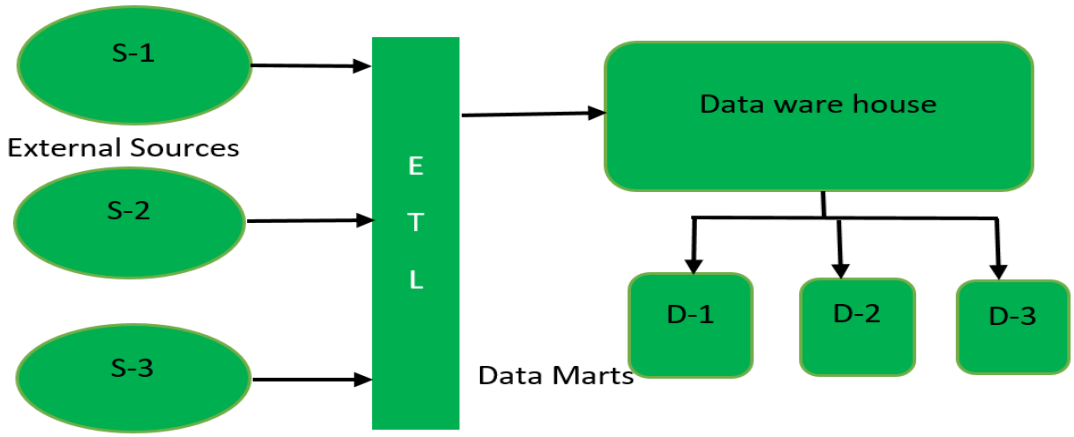
There are mainly two approaches to designing data marts. These approaches are

- 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**.

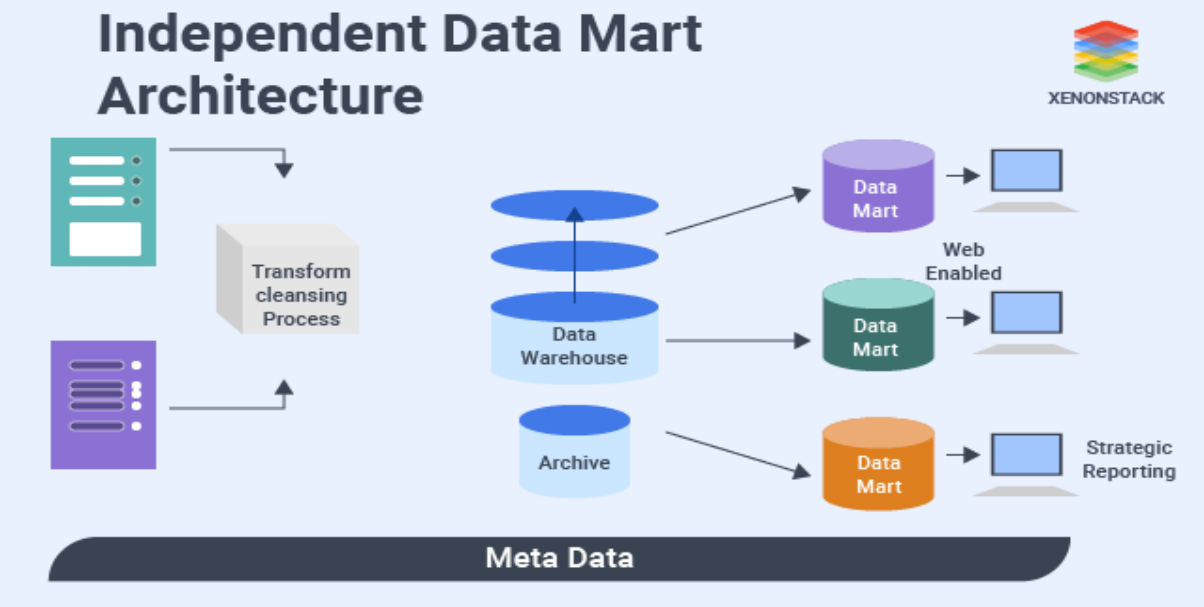
Dependent Data Mart



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.

Independent Data Mart



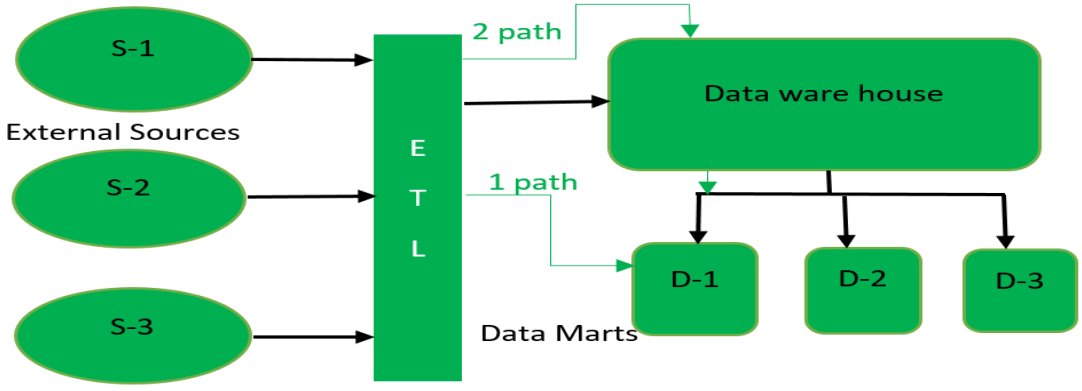
Other than these two categories, one more type exists that is called "**Hybrid Data Marts**."

Hybrid Data Marts

- It allows us to combine input from sources other than a data warehouse. This could be helpful for many situations; especially when Adhoc integrations are needed, such as after a new group or product is added to the organizations. This type of Data Mart is created by extracting data from operational source or from data warehouse.
- 1Path reflects accessing data directly from external sources and 2Path reflects dependent data model of data mart.

3.7. STEPS IN IMPLEMENTING A DATA MART

The significant steps in implementing a data mart are to design the schema, construct the physical storage, populate the data mart with data from source systems, access it to make informed decisions and manage it over time. So, the steps are:



Designing

The design step is the first in the data mart process. This phase covers all of the functions from initiating the request for a data mart through gathering data about the requirements and developing the logical and physical design of the data mart.
It involves the following tasks:

1. Gathering the business and technical requirements
2. Identifying data sources
3. Selecting the appropriate subset of data
4. Designing the logical and physical architecture of the data mart.

Constructing

This step contains creating the physical database and logical structures associated with the data mart to provide fast and efficient access to the data.
It involves the following tasks:

1. Creating the physical database and logical structures such as tablespaces associated with the data mart.
2. creating the schema objects such as tables and indexes describe in the design step.
3. Determining how best to set up the tables and access structures.

Populating

This step includes all of the tasks related to the getting data from the source, cleaning it up, modifying it to the right format and level of detail, and moving it into the data mart.
It involves the following tasks:

1. Mapping data sources to target data sources
2. Extracting data
3. Cleansing and transforming the information.
4. Loading data into the data mart
5. Creating and storing metadata

Accessing

This step involves putting the data to use: querying the data, analyzing it, creating reports, charts and graphs and publishing them.
It involves the following tasks:

1. Set up and intermediate layer (Meta Layer) for the front-end tool to use. This layer translates database operations and objects names into business conditions so that the end-clients can interact with the data mart using words which relates to the business functions.
2. Set up and manage database architectures like summarized tables which help queries agree through the front-end tools execute rapidly and efficiently.

Managing

This step contains managing the data mart over its lifetime. In this step, management functions are performed as:

1. Providing secure access to the data.

2. Managing the growth of the data.
3. Optimizing the system for better performance.
4. Ensuring the availability of data event with system failures.

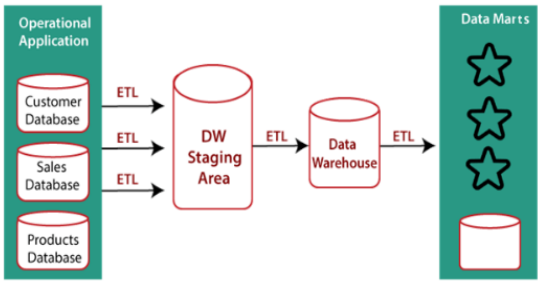
Advantages of Data Mart:

1. Implementation of data mart needs less time as compared to implementation of datawarehouse as data mart is designed for a particular department of an organization.
2. Organizations are provided with choices to choose model of data mart depending upon cost and their business.
3. Data can be easily accessed from data mart.
4. It contains frequently accessed queries, so enable to analyse business trend.

Disadvantages of Data Mart:

1. Since it stores the data related only to specific function, so does not store huge volume of data related to each and every department of an organization like datawarehouse.
2. Creating too many data marts becomes cumbersome sometimes.

Difference between Data Warehouse and Data Mart



Data Warehouse	Data Mart
A Data Warehouse is a vast repository of information collected from various organizations or departments within a corporation.	A data mart is an only subtype of a Data Warehouses. It is architecture to meet the requirement of a specific user group.
It may hold multiple subject areas.	It holds only one subject area. For example, Finance or Sales.
It holds very detailed information.	It may hold more summarized data.
Works to integrate all data sources	It concentrates on integrating data from a given subject area or set of source systems.
In data warehousing, Fact constellation is used.	In Data Mart, Star Schema and Snowflake Schema are used.
It is a Centralized System. It is a Decentralized System.	
Data Warehousing is the data-oriented.	Data Marts is a project-oriented.

3.8. NEED OF DATA MART:

Faster Access: Enables quicker access to relevant data for specific business units.
Improved Decision-Making: Enhances decision-making by tailoring data to departmental needs.
Cost-Effectiveness: Offers a more focused and cost-effective solution for specific analytical requirements.

Why Do We Need a Data Mart?

Listed below are the reasons to create a data mart −

- To partition data in order to impose **access control strategies.**
- To speed up the queries by reducing the volume of data to be scanned.
- To segment data into different hardware platforms.
- To structure data in a form suitable for a user access tool.

Note − Do not data mart for any other reason since the operation cost of data marting could be very high. Before data marting, make sure that data marting strategy is appropriate for your particular solution.

Cost of Data Marting

The cost measures for data marting are as follows −

- Hardware and Software Cost
- Network Access
- Time Window Constraints

Hardware and Software Cost

Although data marts are created on the same hardware, they require some additional hardware and software. To handle user queries, it requires additional processing power and disk storage. If detailed data and the data mart exist within the data warehouse, then we would face additional cost to store and manage replicated data.

Note − Data marting is more expensive than aggregations, therefore it should be used as an additional strategy and not as an alternative strategy.

Network Access

A data mart could be on a different location from the data warehouse, so we should ensure that the LAN or WAN has the capacity to handle the data volumes being transferred within the **data mart load process.**

Time Window Constraints

The extent to which a data mart loading process will eat into the available time window depends on the complexity of the transformations and the data volumes being shipped. The determination of how many data marts are possible depends on −

- Network capacity.
- Time window available
- Volume of data being transferred

Mechanisms being used to insert data into a data mart

3.9. COST EFFECTIVE DATA MART:

The cost of implementing data marts within a data warehouse can vary widely depending on several factors. Here are some key considerations that influence the cost:

1. **Infrastructure Costs**: This includes the hardware, software, and networking infrastructure required to support the data warehouse and data marts. Costs can vary based on factors such as the scale of the data warehouse, the technology stack chosen (e.g., on-premises vs. cloud-based), and the level of redundancy and scalability needed.
2. **Software Licenses**: Costs associated with licensing database management systems (DBMS), ETL tools, data modeling tools, analytics platforms, and other software required for implementing and managing the data warehouse and data marts.
3. **Data Integration Costs**: Expenses related to data integration, including data extraction, transformation, and loading (ETL) processes. This may involve purchasing ETL tools or developing custom ETL solutions, as well as ongoing maintenance and support costs.

4. **Data Storage Costs**: Costs associated with storing data within the data warehouse and data marts. This includes both primary storage for current data and archival storage for historical data. Cloud-based data warehousing solutions often charge based on storage usage, while on-premises solutions may require upfront investments in storage hardware.
5. **Data Governance and Security Costs**: Expenses related to implementing data governance policies, data security measures, and compliance initiatives within the data warehouse environment. This may include investments in access control mechanisms, encryption technologies, auditing tools, and compliance management solutions.
6. **Personnel Costs**: Costs associated with staffing and expertise required to design, implement, and maintain the data warehouse and data marts. This includes salaries for database administrators, data engineers, data architects, business analysts, and other IT professionals involved in managing the data infrastructure.
7. **Training and Professional Services**: Costs associated with training staff on data warehouse technologies, data modeling techniques, ETL processes, and analytics tools. Additionally, organizations may incur expenses for hiring external consultants or professional services firms to assist with data warehouse implementation, optimization, or troubleshooting.
8. **Scalability and Expansion Costs**: Considerations for future scalability and expansion of the data warehouse and data marts. This may include investments in additional hardware resources, software licenses, and infrastructure upgrades to accommodate growing data volumes and evolving business requirements.
9. **Operational Costs**: Ongoing operational expenses such as monitoring, maintenance, backup, and disaster recovery for the data warehouse environment. These costs ensure the reliability, availability, and performance of the data infrastructure over time.
10. Overall, the cost of implementing data marts within a data warehouse can vary significantly based on the organization's size, complexity of data sources, desired functionality, technology choices, and long-term strategic objectives. It's essential to carefully evaluate these factors and develop a comprehensive cost estimation and budgeting plan before embarking on a data warehouse initiative.

Cost-effective Data Marting

Follow the steps given below to make data marting cost-effective −

- Identify the Functional Splits
- Identify User Access Tool Requirements
- Identify Access Control Issues

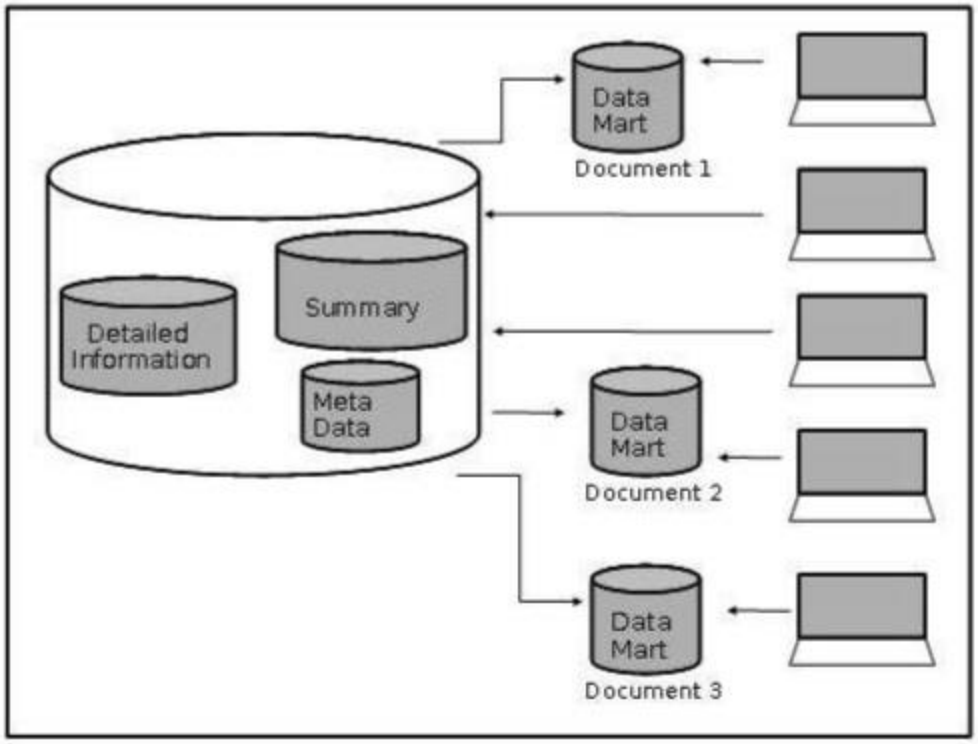
Identify the Functional Splits

In this step, we determine if the organization has natural functional splits. We look for departmental splits, and we determine whether the way in which departments use information tend to be in isolation from the rest of the organization. Let's have an example.

Consider a retail organization, where each merchant is accountable for maximizing the sales of a group of products. For this, the following are the valuable information −

- sales transaction on a daily basis
- sales forecast on a weekly basis
- stock position on a daily basis
- stock movements on a daily basis

As the merchant is not interested in the products they are not dealing with, the data marting is a subset of the data dealing which the product group of interest. The following diagram shows data marting for different users.



Given below are the issues to be taken into account while determining the functional split −

- The structure of the department may change.
- The products might switch from one department to other.
- The merchant could query the sales trend of other products to analyze what is happening to the sales.

Note − We need to determine the business benefits and technical feasibility of using a data mart.

Identify User Access Tool Requirements

We need data marts to support **user access tools** that require internal data structures. The data in such structures are outside the control of data warehouse but need to be populated and updated on a regular basis.

There are some tools that populate directly from the source system but some cannot. Therefore additional requirements outside the scope of the tool are needed to be identified for future.

Note − In order to ensure consistency of data across all access tools, the data should not be directly populated from the data warehouse, rather each tool must have its own data mart.

Identify Access Control Issues

There should to be privacy rules to ensure the data is accessed by authorized users only. For example a data warehouse for retail banking institution ensures that all the accounts belong to the same legal entity. Privacy laws can force you to totally prevent access to information that is not owned by the specific bank.

Data marts allow us to build a complete wall by physically separating data segments within the data warehouse. To avoid possible privacy problems, the detailed data can be removed from the data warehouse. We can create data mart for each legal entity and load it via data warehouse, with detailed account data.

3.10. DESIGNING DATA MARTS:

Data marts should be designed as a smaller version of starflake schema within the data warehouse and should match with the database design of the data warehouse. It helps in maintaining control over database instances.

The summaries are data marted in the same way as they would have been designed within the data warehouse. Summary tables help to utilize all dimension data in the starflake schema.

- Designing data marts involves several steps and considerations to ensure that they effectively serve the analytical needs of the organization. Here's a detailed overview of the process:
- **Understand Requirements**: Begin by gathering requirements from stakeholders, including analysts, business users, and decision-makers. Understand the specific questions they need to answer and the data they require to do so.
- **Identify Data Sources**: Determine the sources of data required to populate the data mart. These may include transactional databases, CRM systems, ERP systems, spreadsheets, external data feeds, etc. Assess the quality, completeness, and relevance of each data source.
- **Data Extraction, Transformation, and Loading (ETL)**:
- **Extraction**: Extract data from the identified sources using appropriate methods such as SQL queries, APIs, flat file exports, etc.
- **Transformation**: Cleanse, transform, and enrich the data as necessary to ensure consistency, accuracy, and usability. This may involve data cleaning, standardization, deduplication, data enrichment, etc.
- **Loading**: Load the transformed data into the data mart. This could be done using ETL tools, custom scripts, or database replication techniques.
- **Data Modeling**:
- **Dimensional Modeling**: Design the dimensional model for the data mart using techniques like star schema or snowflake schema. Identify dimensions (descriptive attributes) and facts (measurable metrics) based on the requirements gathered earlier.
- **Granularity**: Determine the level of detail or granularity at which data will be stored in the data mart. This should align with the analytical needs while considering performance and storage constraints.
- **Aggregation**: Identify and define aggregations to improve query performance for common analytical queries
- **Indexing and Partitioning**: Implement appropriate indexing and partitioning strategies to optimize query performance and manage data efficiently, especially for large datasets.
- **Data Security and Governance**: Implement security measures to ensure that sensitive data is protected appropriately. Define access controls and permissions to restrict data access based on user roles and responsibilities. Adhere to relevant data governance policies and regulations.
- **Metadata Management**: Establish a robust metadata management process to document the structure, content, and lineage of data within the data mart. This helps users understand the meaning and context of the data they are analyzing.

- **Data Quality Assurance**: Implement data quality checks and validation processes to ensure the accuracy, consistency, and completeness of data within the data mart. Monitor data quality over time and address any issues promptly.
- **Performance Tuning**: Continuously monitor and optimize the performance of the data mart. This may involve tuning database configurations, optimizing queries, adding hardware resources, or revising data models as needed.
- **Documentation and Training**: Document the design, architecture, and usage guidelines for the data mart. Provide training and support to users to ensure they can effectively leverage the data mart for their analytical needs.
- **Change Management**: Establish processes for managing changes to the data mart, including schema changes, data updates, and enhancements based on evolving business requirements.
- **Integration with BI Tools**: Integrate the data mart with business intelligence (BI) tools and reporting platforms to enable users to visualize and analyze data effectively. Ensure compatibility and seamless data access between the data mart and BI tools.
- By following these steps and considerations, you can design and implement data marts that provide valuable insights and support informed decision-making within your organization.
- **Requirements Gathering:** Identifying business objectives, user needs, and data sources.
- **Data Modeling:** Creating a dimensional model with fact tables, dimension tables, and relationships.
- **ETL Development:** Implementing Extract, Transform, Load processes for data integration.
- **Testing:** Validating data mart against user requirements to ensure quality.

3.11. Partitioning Strategy:

- Partitioning is done to enhance performance and facilitate easy management of data.
- Partitioning also helps in balancing the various requirements of the system.
- It optimizes the hardware performance and simplifies the management of data warehouse by partitioning each fact table into multiple separate partitions.

Why is it Necessary to Partition?

Partitioning is important for the following reasons −

- For easy management,
- To assist backup/recovery,
- To enhance performance.
- The fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of fact table is very hard to manage as a single entity. Therefore it needs partitioning.

Types of Partitioning:

Partitioning can be horizontal (rows are divided based on criteria such as ranges of values, list of discrete values) or vertical (columns are divided based on access patterns or usage frequency).

Example: In a sales database, you might partition the "Orders" table horizontally by date ranges (e.g., monthly partitions) to improve query performance for time-based analyses. Alternatively, you could vertically partition a large table into subsets of columns based on their access patterns to reduce storage overhead and optimize query performance.

1. Vertical partition
   - Normalization
   - Row Splitting
2. Horizontal partition

3.11.1. Vertical Partition - Normalization - Row Splitting

Vertical Partitioning: Splitting a table into smaller tables vertically based on columns or attributes.

- Vertical partitioning, also known as columnar partitioning or column-based partitioning, is a data partitioning technique to improve query performance and storage efficiency.
- Divides tables into rows based on specific criteria, vertical partitioning divides tables into columns.
- Each subset, or vertical partition, contains a distinct set of columns from the original table.
- For example, consider a large sales table with columns such as `order_id`, `customer_id`, `order_date`, `product_id`, `quantity`, `unit_price`, and `total_price`. Through vertical partitioning, you might create separate partitions for frequently accessed columns like `order_id`, `customer_id`, and `order_date`, while less frequently accessed or large columns like `product_id`, `quantity`, `unit_price`, and `total_price` are stored separately.

Benefits of Vertical Partitioning

- **Improved Query Performance**: By grouping related columns together in vertical partitions, queries can avoid scanning unnecessary columns, leading to faster query execution times.

  This is particularly beneficial when queries only access a subset of columns from large tables.
- **Storage Efficiency**: Vertical partitioning can reduce storage overhead by segregating frequently accessed columns from less frequently accessed ones.

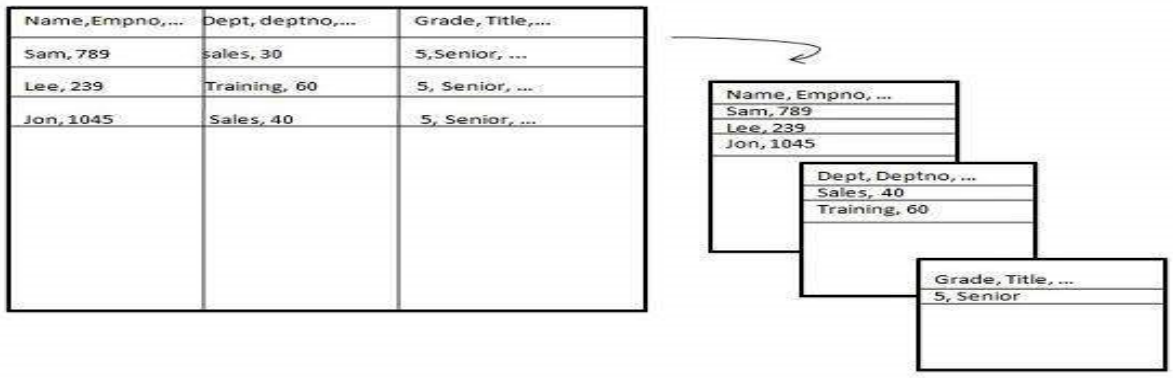
  This allows more efficient use of storage resources and can help optimize disk I/O operations.

  For example, highly repetitive or sparse columns may benefit from different compression algorithms, leading to further storage savings.
- **Simplified Data Maintenance**: Vertical partitioning can simplify data maintenance tasks such as backup and restore operations, as well as data loading and extraction processes.

  By separating columns into distinct partitions, you can perform these operations more selectively and efficiently.

Considerations for Vertical Partitioning

- **Query Patterns**: Analyze the typical **query patterns and access paths** to identify columns that are frequently accessed together.
- **Data Size and Distribution**: Consider the **size and distribution of data** within each column when deciding how to partition the data vertically.
- **Compression Requirements**: Evaluate **the compression requirements** for each column based on its data characteristics and usage patterns.
- **Data Model Complexity**: Vertical partitioning may introduce additional complexity to the **data model, especially when joining or aggregating** data across multiple partitions.
- **Database Support**: Check whether your database management system (DBMS) supports vertical partitioning natively or through custom implementation.



Implementation and Management

Vertical partitioning can be implemented using database-specific features such as column-oriented storage formats, databases, or custom partitioning schemes.

- In summary, vertical partitioning is a valuable technique in data warehousing for improving query performance, storage efficiency, and data maintenance.
- By selectively grouping columns into vertical partitions based on their access patterns and characteristics, organizations can optimize the storage and processing of large datasets, leading to faster and more efficient analytical workflows.
- Horizontal partitioning, also known as row partitioning or sharding, is a database design technique used in data warehousing to divide large tables into smaller subsets called partitions based on specific criteria.
- Each partition contains a subset of rows from the original table, allowing for more efficient data management, query performance, and scalability.

3.11.3NORMALIZATION:

Organizing data to reduce redundancy and improve data integrity.

Normalization partition in data warehouse

Normalization:

Definition: Normalization is the process of organizing data in a database to reduce redundancy and dependency by dividing large tables into smaller, related tables and defining relationships between them.

Goals: The primary goals of normalization are to eliminate data anomalies (such as update, insertion, and deletion anomalies) and to improve data integrity, consistency, and flexibility.

Normal Forms: Normalization is typically carried out according to a series of normal forms, such as First Normal Form (1NF), Second Normal Form (2NF),

Third Normal Form (3NF), Boyce-Codd Normal Form (BCNF), and Fourth Normal Form (4NF). Each normal form addresses specific types of data redundancy and dependency issues.
Example: Consider a denormalized "Customer Orders" table that stores both customer details (e.g., name, address) and order details (e.g., order date, product ID, quantity) in the same table. Normalization would involve splitting this table into separate "Customers" and "Orders" tables and establishing a one-to-many relationship between them.

- Data redundancy and improve data integrity through table restructuring and relationship definitions, partitioning focuses on improving query performance, manageability, and scalability by dividing tables or indexes into smaller segments based on specific criteria.
- Both normalization and partitioning are important techniques in database design and optimization,.

Normalization

Normalization is the standard relational method of database organization. In this method, the rows are collapsed into a single row, hence it reduce space. Take a look at the following tables that show how normalization is performed.

Table before Normalization

Product_id	Qty	Value	sales_date	Store_id	Store_name	Location	Region
30	5	3.67	3-Aug-13	16	sunny	Bangalore	S
35	4	5.33	3-Sep-13	16	sunny	Bangalore	S
40	5	2.50	3-Sep-13	64	san	Mumbai	W
45	7	5.66	3-Sep-13	16	sunny	Bangalore	S

Table after Normalization

Store_id	Store_name	Location	Region
16	sunny	Bangalore	W
64	san	Mumbai	S

Product_id	Quantity	Value	sales_date	Store_id
30	5	3.67	3-Aug-13	16
35	4	5.33	3-Sep-13	16
40	5	2.50	3-Sep-13	64
45	7	5.66	3-Sep-13	16

3.11.4.ROW SPLITTING:

Breaking down large rows into smaller rows to enhance query performance.

Row Splitting:

Row splitting tends to leave a one-to-one map between partitions. The motive of row splitting is to speed up the access to large table by reducing its size.

Note − While using vertical partitioning, make sure that there is no requirement to perform a major join operation between two partitions.

Identify Key to Partition

It is very crucial to choose the right partition key. Choosing a wrong partition key will lead to reorganizing the fact table. Let's have an example. Suppose we want to partition the following table.

Account_Txn_Table

transaction_id

account_id
transaction_type
value
transaction_date
region
branch_name
We can choose to partition on any key. The two possible keys could be

- region
- transaction_date

Suppose the business is organized in 30 geographical regions and each region has different number of branches. That will give us 30 partitions, which is reasonable. This partitioning is good enough because our requirements capture has shown that a vast majority of queries are restricted to the user's own business region.
If we partition by transaction date instead of region, then the latest transaction from every region will be in one partition. Now the user who wants to look at data within his own region has to query across multiple partitions. Hence it is worth determining the right partitioning key.

Metadata Management Software

Software for managing metadata makes it easier to assess, curate, collect, and store metadata. In order to enable data monitoring and accountability, organizations should automate data management. Examples of this kind of software include the following:

SAP Power Designer by SAP(System applications and Produts in Data Processing):

This data management system has a good level of stability. It is recognised for its ability to serve as a platform for model testing.

SAP Information Steward by SAP:

This solution’s data insights make it valuable.

IBM InfoSphere Information Governance Catalog by IBM:

The ability to use Open IGC to build unique assets and data lineages is a key feature of this system.

Alation Data Catalog by Alation:

This provides a user-friendly, intuitive interface. It is valued for the queries it can publish in Standard Query Language (SQL).

Informatica Enterprise Data Catalog by Informatica:

The technology used by this solution, which can both scan and gather information from diverse sources, is highly respected.

3.12.HORIZONTAL PARTITIONING

Definition of Horizontal Partitioning:

1. Horizontal partitioning involves dividing a large table into smaller partitions, each containing a subset of rows based on defined criteria. This criteria could include ranges of values, list of discrete values, or hash values.
2. For example, consider a large "Sales" table containing millions of rows. Through horizontal partitioning, divide the table into monthly partitions, with each partition containing sales data for a specific month.

Benefits of Horizontal Partitioning:

Improved Query Performance: Horizontal partitioning can improve query performance by reducing the amount of data that needs to be scanned or processed for a given query. Queries can target specific partitions based on the query predicates, leading to faster query execution times.

Efficient Data Management: Horizontal partitioning makes data management tasks such as data loading, backup and recovery, and maintenance operations more efficient.

Operations can be performed selectively on individual partitions rather than the entire table, leading to faster and more targeted processing.

Scalability: Horizontal partitioning facilitates horizontal scalability by allowing new partitions to be added as data volume grows, without impacting existing partitions or requiring expensive data reorganization. This makes it easier to manage and scale data warehouses as data volumes increase over time.

Types of Horizontal Partitioning:

1. **Range Partitioning**: Data is partitioned **based on ranges of values in one of the columns**. For example, the "Sales" table might be partitioned by sales date, with each partition containing sales data for a specific date range (e.g., monthly partitions).
2. **List Partitioning**: Data is partitioned based on **discrete values in one of the columns**. For example, you might partition a "Customers" table by geographic region, with each partition containing customers from a specific region.
3. **Hash Partitioning**: Data **is partitioned based on the result of applying a hash function to one or more columns**. This ensures even distribution of rows across partitions, which can help improve query performance and load balancing.

Considerations for Horizontal Partitioning

Partition Key Selection: Choose a suitable row or set of columns as the partition key based on access patterns, query requirements, and data distribution characteristics.

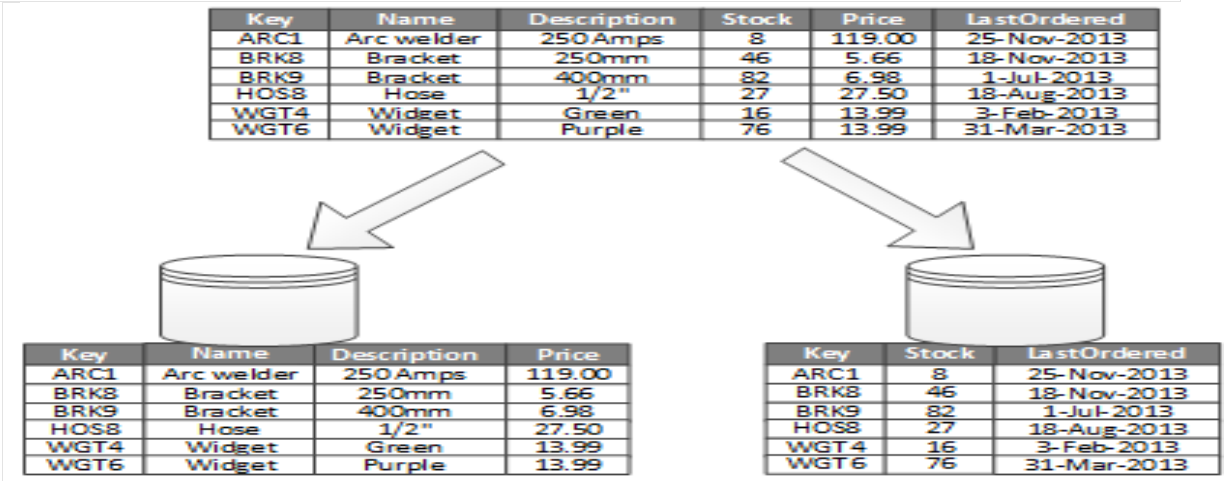
The partition key should be carefully selected to ensure even data distribution and efficient query performance.

Partition Size: Aim for a balanced partition size to ensure uniform data distribution and efficient resource utilization.

Avoid creating partitions that are too small or too large, as this can impact query performance and resource utilization.

Data Skew: Be aware of potential data skew issues, where certain partitions may become disproportionately larger than others.

Monitor partition sizes regularly and take corrective actions if necessary to ensure balanced data distribution.



Implementation and Management

Horizontal partitioning is typically implemented using database-specific features and syntax.

Most modern relational database management systems (RDBMS) provide support for horizontal partitioning, including Oracle, SQL Server, MySQL.

In summary, horizontal partitioning is a valuable technique in data warehousing for improving query performance, efficient data management, and scalability.

By dividing large tables into smaller partitions based on specific criteria, organizations can achieve faster query execution times, more efficient data loading and management, and better scalability as data volumes grow over time.

Feature	Vertical Partitioning	Horizontal Partitioning
Definition	Dividing a table into smaller tables based on columns.	Dividing a table into smaller tables based on rows (usually ranges of rows).
Purpose	Reduce the number of columns in a table to improve query performance and reduce I/O.	Divide a table into smaller tables to manage large volumes of data efficiently.
Data distribution	Columns with related data are placed together in the same table.	Rows with related data (typically based on a range or a condition) are placed together in the same table.
Query performance	Improves query performance when queries only involve specific columns that are part of a partition.	Improves query performance when queries primarily access a subset of rows in a large table.
Maintenance and indexing	Easier to manage and index specific columns based on their characteristics and access patterns.	Each partition can be indexed independently, making indexing more efficient.
Joins	May require joins to combine data from multiple partitions when querying.	Joins between partitions are typically not needed, as they contain disjoint sets of data.
Data integrity	Ensuring data consistency across partitions can be more challenging.	Easier to maintain data integrity, as each partition contains a self-contained subset of data.
Use cases	Commonly used for tables with a wide range of columns, where not all columns are frequently accessed together.	Commonly used for tables with a large number of rows, where data can be grouped based on some criteria (e.g., date ranges).
Examples	Splitting a customer table into one table for personal details and another for transaction history.	Partitioning a large sales order table by date, with each partition containing orders from a specific month or year.