- **Perform data quality checks :** Validate the loaded data to ensure its accuracy, completeness and conformity to predefined data quality standards. Identify and handle any data quality issues or discrepancies.

Q.5 List out advantages of ETL.

Ans. : Here are some key advantages of the ETL process :

1. Data integration
2. Data consistency and quality
3. Standardization and conformity
4. Data transformation and enrichment
5. Historical data tracking
6. Improved performance
7. Scalability and flexibility
8. Data security and governance
9. Automation and efficiency
10. Decision - making and insights

Q.6 What is top down design approach ?

Ans. : The top-down approach is a common methodology used in data warehouse design. It involves designing the data warehouse from a high - level perspective and gradually drilling down into more detailed components.

Q.7 What is bottom up approach ?

Ans. : The bottom-up approach is an alternative methodology used in data warehouse design. It involves building the data warehouse incrementally, starting from individual data sources and gradually integrating them into a comprehensive solution.

Q.8 What is data modeling life cycle ?

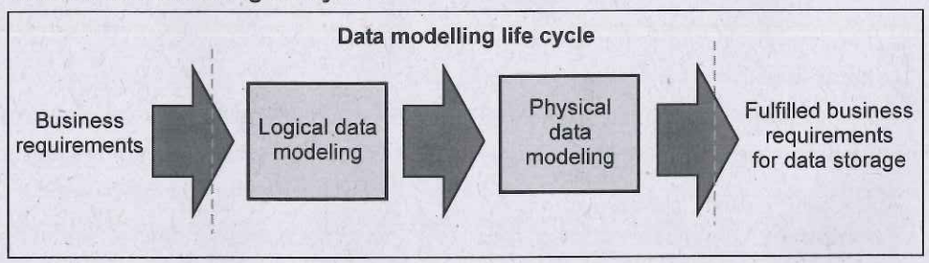

Fig. 2.13.1

Ans. : The data modeling life cycle encompasses the processes and activities involved in creating and managing data models throughout their lifecycle

❑❑❑

UNIT III

3 Meta Data, Data Mart and Partition Strategy

Syllabus

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.

Contents

3.1 Meta Data

- Metadata refers to the information about the data stored in a data warehouse. It provides details about the structure, meaning and usage of the data, allowing users to understand and effectively utilize the information contained within the data warehouse.

3.1.1 Categories of Metadata

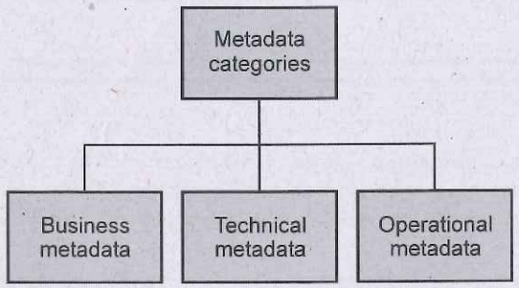

Fig 3.1.1 Broad categories of metadata

- In data warehousing, metadata can be categorized into several types based on their purpose and content. Here are the commonly recognized categories of metadata in data warehousing :

1. **Technical metadata :** This category includes metadata that describes the technical aspects of the data warehouse. It provides information about the physical structure and organization of the data, such as data source details, data types, file formats, database schemas, table structures, column definitions, indexes, partitions and data storage configurations.
2. **Business metadata :** Business metadata focuses on capturing the business context and meaning of the data. It includes metadata related to data definitions, business rules, data ownership, data lineage, data quality metrics, business glossaries and business process information. Business metadata helps users understand the semantics and relevance of the data in the context of the organization's operations.
3. **Operational metadata :** Operational metadata pertains to the processes and operations involved in managing and maintaining the data warehouse. It includes metadata about data extraction, transformation and loading (ETL) processes, data refresh schedules, data integration workflows, data validation rules, data lineage tracking, data monitoring and system-level metadata such as server configurations and performance metrics.

4. **Data quality metadata :** This category of metadata focuses on data quality-related information. It includes metadata about data profiling results, data quality rules, data validation and cleansing processes, data accuracy, completeness, consistency and timeliness metrics. Data quality metadata helps in assessing the trustworthiness and reliability of the data stored in the data warehouse.
5. **Usage metadata :** Usage metadata tracks how the data warehouse is utilized by users and applications. It includes metadata about data access patterns, query history, report usage statistics, user access permissions, security and privacy policies, data usage logs and auditing information. Usage metadata provides insights into data utilization patterns, helps optimize query performance and supports compliance and security requirements.
6. **Metadata relationships and dependencies :** This category includes metadata that captures relationships and dependencies between different data elements, such as data source-to-target mappings, transformation rules, data lineage and data dependencies. It helps in understanding data flows, impact analysis and maintaining the integrity and consistency of the data warehouse.

- These categories of metadata play a crucial role in data warehousing by enabling data governance, data integration, data quality management and effective utilization of the data assets stored in the data warehouse.

3.1.2 Role of Metadata

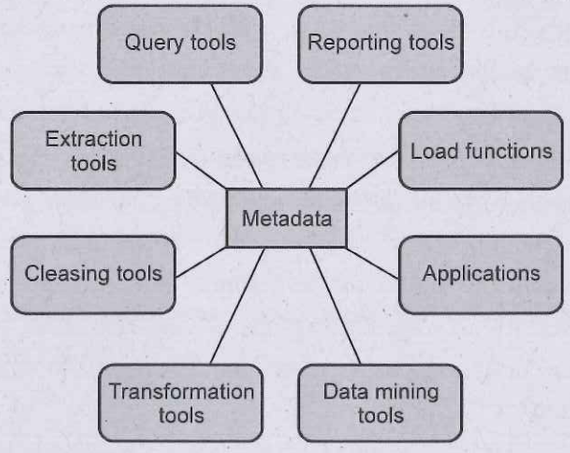

Fig. 3.1.2 Role of metadata

- Metadata plays a crucial role in data warehousing by providing valuable information about the data stored in a data warehouse. Metadata can be defined as data about data. It describes the characteristics, structure and organization of data, enabling users and systems to understand and effectively use the data in the data warehouse. Here are some key roles of metadata in data warehousing :

1. **Data discovery and understanding :** Metadata helps users to discover and understand the available data in the data warehouse. It provides information about the source of the data, its origin, meaning and structure. This enables users to determine the relevance and suitability of the data for their analytical needs.
2. **Data integration and transformation :** Metadata facilitates data integration by providing information about the relationships and dependencies between various data elements. It helps in mapping and transforming data from different sources into a unified format within the data warehouse. Metadata also assists in data cleansing, data quality assessment and data transformation processes.
3. **Data lineage and provenance :** Metadata tracks the lineage and provenance of data within the data warehouse. It records information about the transformations, processes, and calculations applied to the data at each stage. This lineage information is crucial for auditing, compliance and troubleshooting purposes, as it enables tracing back to the source of any data element and understanding how it has been derived.
4. **Data governance and compliance :** Metadata supports data governance initiatives by providing visibility into the data assets, their ownership and usage policies. It helps in enforcing data standards, data security and compliance requirements. Metadata can include information about data classification, sensitivity, retention periods and access controls, ensuring that the data in the data warehouse is managed appropriately and complies with regulatory guidelines.
5. **Query optimization and performance :** Metadata plays a role in optimizing query performance within the data warehouse. It includes statistics about the data distribution, indexes and partitions, which can be used by query optimizers to generate efficient query execution plans. By understanding the characteristics of the data, the query optimizer can make informed decisions to minimize data retrieval and processing time.
6. **Data documentation and collaboration :** Metadata serves as a documentation tool for data warehousing projects. It captures information about data models, data dictionaries, business glossaries and data transformation rules. This documentation facilitates collaboration among different stakeholders, such as business analysts, data

architects and data scientists, ensuring a common understanding of the data and promoting effective communication.

- Metadata in data warehousing provides essential context and information about the data stored in the data warehouse. It supports data discovery, integration, transformation, governance, compliance, query optimization and documentation, enabling users to effectively leverage the data for decision-making and analysis purposes.

3.1.3 Metadata Repository

- A metadata repository in data warehousing refers to a centralized storage system that houses and manages metadata associated with the data warehouse environment. It serves as a comprehensive and structured catalog of metadata, providing a holistic view of the data assets, structures, relationships and processes within the data warehousing ecosystem.
- A metadata repository in data warehousing typically includes the following components :

1. **Metadata storage :** The repository stores various types of metadata, including descriptive metadata (e.g., data definitions, business glossaries), structural metadata (e.g., data models, schemas), operational metadata (e.g., data lineage, transformations) and administrative metadata (e.g., access controls, ownership). The metadata is organized in a structured and searchable format.
2. **Metadata capture and registration :** The repository allows for the capture, registration, and ingestion of metadata from different sources and systems. This can be done manually by data stewards or automatically through metadata extraction processes. Metadata from data sources, ETL (Extract, Transform, Load) processes, data integration tools and other systems are captured and registered in the repository.
3. **Metadata integration and interoperability :** The repository can integrate with various data management systems and tools involved in the data warehousing ecosystem. This enables metadata exchange and interoperability between different components, such as data warehouses, data integration tools, data modeling tools and business intelligence platforms. The repository acts as a central hub for metadata management across these systems.
4. **Metadata search and discovery :** Users can search and explore the metadata repository to discover relevant data assets and understand their characteristics. The repository provides search capabilities based on different criteria, such as data element names, data models, tables, attributes or business concepts. This enables efficient data discovery and promotes reuse of existing data assets.

5. **Metadata lineage and impact analysis :** The repository captures and tracks the lineage of data elements within the data warehouse environment. It documents the flow of data from source systems to target tables, including the transformations applied at each stage. This lineage information helps in understanding data provenance, impact analysis and troubleshooting.
6. **Data governance and compliance :** The repository supports data governance initiatives by providing a framework for managing metadata standards, policies and data governance rules. It allows for defining and enforcing data quality controls, access controls, ownership information and compliance requirements. The repository ensures that metadata adheres to organizational standards and regulatory guidelines.
7. **Metadata documentation and collaboration :** The repository serves as a documentation hub for metadata information in the data warehousing environment. It allows for the documentation of data models, data dictionaries, data transformation rules and other relevant documentation. It facilitates collaboration among different stakeholders, enabling them to access, contribute and review the metadata documentation.

- A metadata repository in data warehousing is a centralized storage system that manages and organizes metadata associated with the data warehouse environment. It provides a consolidated view of the metadata, enabling efficient metadata management, data discovery, lineage tracking, compliance and collaboration within the data warehousing ecosystem.

3.1.3.1 Uses of Metadata Repository

- A metadata repository plays a vital role in managing and organizing metadata related to the data warehouse environment. Here is how a metadata repository is specifically used in data warehousing :

1. **Data source and integration :** The metadata repository stores information about the data sources used in the data warehouse. This includes details about the structure, format, location and extraction methods of the source data. It helps in identifying and integrating data from various sources into the data warehouse.
2. **Data transformation and ETL :** The repository stores metadata about the transformations and ETL (Extract, Transform, Load) processes applied to the data during the data integration and loading stages. It includes details about the transformation rules, mappings and dependencies. This metadata helps in understanding and managing the data transformation processes.

3. **Data modeling :** The repository holds metadata related to the data models used in the data warehouse. It includes information about the logical and physical data models, schema definitions, relationships between tables, attributes and indexes. This metadata provides a comprehensive view of the data structure and aids in data modeling and schema management.
4. **Data lineage and impact analysis :** Metadata in the repository captures the lineage and impact of data within the data warehouse. It tracks the origin of data elements, the transformations applied and the target data tables. This lineage information helps in understanding how data is derived, tracing back to its source and analyzing the impact of changes on downstream processes.
5. **Data quality and governance :** The metadata repository can store metadata related to data quality rules, metrics and monitoring. It helps in defining and enforcing data quality standards and policies for the data stored in the data warehouse. Metadata about data governance policies, access controls and ownership can also be stored in the repository.
6. **Performance optimization :** The metadata repository stores information about the performance characteristics of the data warehouse, such as indexes, partitions and statistics. This metadata helps in optimizing query performance by enabling the query optimizer to make informed decisions based on the data distribution and structure.
7. **Data documentation and collaboration :** The metadata repository acts as a documentation hub for the data warehouse environment. It stores metadata documentation such as data dictionaries, business glossaries and data lineage diagrams. It facilitates collaboration among different stakeholders, allowing them to access and contribute to the metadata documentation.

- By leveraging a metadata repository in data warehousing, organizations can effectively manage, govern and utilize the metadata associated with their data warehouse environment. It provides a centralized and comprehensive view of the data assets, facilitating data integration, transformation, modeling, lineage analysis, performance optimization, data quality management and collaboration among stakeholders.

3.1.4 Challenges for Meta Management

- Managing metadata in data warehousing can pose several challenges due to the complexity and scale of data warehouse environments. Here are some common challenges faced in metadata management for data warehousing :

1. **Metadata integration :** Data warehouses often integrate data from multiple sources, each with its own data models, formats and semantics. Integrating and harmonizing metadata from diverse sources can be challenging, requiring mapping and transformation efforts to establish a unified metadata model.
2. **Data lineage and impact analysis :** Tracking data lineage and understanding the impact of changes can be complex in data warehousing. As data goes through various transformations and aggregations, capturing and maintaining accurate lineage information becomes challenging. Changes in source data, transformations or schema modifications need to be tracked to ensure accurate impact analysis.
3. **Data governance and data quality :** Establishing and enforcing data governance policies and ensuring data quality in a data warehouse can be challenging. Metadata management plays a crucial role in supporting data governance initiatives, but it requires establishing governance frameworks, defining standards and monitoring data quality across different data sources and transformations.
4. **Metadata consistency and accuracy :** Maintaining consistency and accuracy of metadata across different components of the data warehouse is a significant challenge. Metadata needs to be updated and synchronized when changes occur in data sources, transformations or schema structures. Inconsistencies in metadata can lead to incorrect analysis, reporting or data integration processes.
5. **Metadata scalability and performance :** As data warehouses grow in size and complexity, managing metadata at scale becomes challenging. Large volumes of metadata can impact performance during metadata search, retrieval and updates. Efficient strategies and technologies, such as metadata caching, indexing and partitioning, need to be implemented to ensure scalability and performance.
6. **Metadata documentation and communication :** Documenting metadata effectively and facilitating communication among different stakeholders can be a challenge. Metadata needs to be documented in a clear and understandable manner, using standard definitions and terminology. Communication channels and collaboration tools need to be established to facilitate metadata discussions and knowledge sharing.
7. **Metadata security and access controls :** Metadata contains sensitive information about data structures, transformations and data lineage. Securing metadata and enforcing appropriate access controls to ensure only authorized users can view or modify metadata is crucial. Implementing security measures, such as role-based access control and encryption, is essential to protect sensitive metadata.

8. **Metadata maintenance and versioning :** Managing changes to metadata over time is critical for maintaining the accuracy and consistency of metadata. Metadata versions and history need to be tracked, allowing for rollback or comparison of metadata changes. Proper versioning and change management processes should be in place to avoid conflicts or unintentional metadata modifications.

- Addressing these challenges requires a combination of robust metadata management processes, effective governance frameworks, automated metadata management tools and collaboration among different stakeholders involved in data warehousing initiatives.

3.2 Data Mart

- Data mart is a subset of a data warehouse that focuses on a specific subject area or department within an organization. It is a smaller, more focused version of a data warehouse and is designed to meet the specific reporting and analytical needs of a particular business unit or user group.
- Here are some key characteristics and considerations related to data marts in data warehousing :

1. **Subject-oriented :** A data mart is typically organized around a specific subject area, such as sales, marketing, finance or human resources. It contains data that is relevant and tailored to the analytical requirements of that particular subject or department.
2. **Data granularity :** Data marts often store data at a more detailed and granular level compared to the data warehouse. This allows for more in-depth analysis and reporting within the specific subject area. Data aggregation and summarization may still occur in the data mart, but at a level suitable for the subject's requirements.
3. **Data integration :** Data marts can be created by extracting and transforming data from the central data warehouse or by directly integrating data from various source systems. Depending on the approach, data integration processes, such as ETL (Extract, Transform, Load), are performed to populate and update the data mart with the relevant data.
4. **Data modeling :** Data marts have their own data models tailored to the specific subject area. These models are designed to support the reporting and analytical needs of the subject, providing a simplified and focused view of the data. Dimensional modeling, such as star schema or snowflake schema, is commonly used in data mart design.

5. **User-focused :** Data marts are designed to cater to the needs of specific user groups or business units within an organization. They are optimized for the reporting and analysis requirements of those users, providing a user-friendly and intuitive interface to access and explore the data.
6. **Performance and scalability :** Data marts are designed to deliver high performance for queries and reporting within their subject area. By focusing on a specific subset of data, data marts can be optimized for faster response times and improved query performance. However, it is important to ensure that data marts are scalable to handle increasing data volumes and evolving analytical needs.
7. **Incremental development :** Data marts are often built incrementally, starting with the most critical or high-priority subject areas. This allows for faster implementation and quicker delivery of analytical capabilities to users. As the organization's data warehousing needs evolve, additional data marts can be created to cover other subject areas.
8. **Data governance :** Data governance practices and standards still apply to data marts, ensuring data quality, consistency and security. While data governance may be more focused within the scope of a specific data mart, it is important to align the data mart's practices with the overall data governance framework of the organization.

- Data marts provide a means to deliver targeted, subject-specific data and analytics capabilities to users, enabling them to gain insights and make informed decisions within their respective domains. They can be standalone entities or integrated with a larger data warehouse infrastructure, depending on the organization's requirements and data warehousing strategy.
- Data marts provide a localized and tailored solution for specific user groups, enabling them to access and analyze data that is directly relevant to their business area. They complement the central data warehouse by delivering focused analytical capabilities and empowering users to make data-driven decisions within their domain.

3.2.1 Cost Effective Data Mart

- Creating a cost-effective data mart within a data warehouse involves implementing strategies and best practices to optimize resources and minimize expenses. Here are some approaches to consider :

1. **Data mart consolidation :** Instead of creating multiple individual data marts for each subject area or department, consider consolidating related data marts into a single, unified data mart. This consolidation reduces the infrastructure and maintenance costs associated with managing multiple data marts.

2. **Incremental development :** Adopt an incremental development approach to build data marts gradually. Start with the most critical or high-priority subject areas and expand as needed. This approach allows you to prioritize resources and allocate budget based on the immediate needs and business value.
3. **Data mart virtualization :** Explore the possibility of virtualizing data marts rather than physically creating separate instances. Virtualization allows you to leverage the existing infrastructure and resources of the data warehouse, minimizing additional hardware and software costs associated with maintaining separate data marts.
4. **Data mart automation :** Implement automation tools and processes for data mart development, maintenance and updates. Automation reduces manual effort, saves time and lowers operational costs. This can include automating data integration, transformation, data model generation and metadata management tasks.
5. **Cloud-based solutions :** Consider leveraging cloud-based data warehousing solutions to create and manage data marts. Cloud providers often offer flexible pricing models, allowing you to pay for resources and usage on a per-need basis. This eliminates the upfront infrastructure costs and provides scalability options as your data mart requirements evolve.
6. **Open-source technologies :** Utilize open-source technologies for data mart development and management. Open-source tools often offer cost-effective alternatives to commercial software, reducing licensing costs. Additionally, the open-source community provides ongoing support, enhancements and a wide range of resources for implementation.
7. **Data compression and storage optimization :** Implement data compression techniques to reduce storage requirements in the data mart. Compressing data reduces disk space usage and lowers storage costs. Additionally, employ storage optimization techniques such as data partitioning, indexing and data archiving to improve query performance and minimize storage expenses.
8. **Performance monitoring and optimization :** Regularly monitor and optimize the performance of data marts to ensure efficient resource utilization. Identify and address performance bottlenecks, optimize queries and indexes and fine-tune the data mart's configuration. Optimizing performance reduces the need for additional hardware resources and can lead to cost savings.
9. **Resource sharing and collaboration :** Encourage resource sharing and collaboration among data mart users and stakeholders. By enabling users from different departments or subject areas to share common data and analytical models, you can reduce redundant efforts and costs associated with duplicating data marts for similar purposes.

10. **Scalability planning :** Develop a scalability plan for future growth and expansion of data marts. Consider the projected data volumes, user requirements and evolving business needs. Design the data mart architecture with scalability in mind, allowing for easy and cost-effective expansion as the demands increase over time.

- By implementing these cost-effective strategies, organizations can create and manage data marts within their data warehouse environment while optimizing resources and minimizing expenses. It is important to align these approaches with the specific requirements and constraints of the organization to achieve the desired cost efficiencies.

3.2.2 Designing Data Marts

- Designing data marts in a data warehouse involves several steps and considerations to ensure they effectively meet the analytical and reporting needs of specific user groups or departments. Here is a high-level overview of the design process :

1. **Identify business requirements :** Start by understanding the specific business requirements and objectives of the user group or department for which the data mart is being designed. This involves conducting interviews, workshops and discussions with stakeholders to gather information about their data needs, reporting requirements, and analytical goals.
2. **Define subject area :** Determine the subject area that the data mart will focus on. Identify the key entities, attributes and relationships within that subject area. This step helps in defining the scope and boundaries of the data mart and ensures that it aligns with the business objectives.
3. **Determine data sources :** Identify the relevant data sources that will provide the necessary data for the data mart. This can include data from the central data warehouse, operational databases, external systems, or other sources. Understand the structure, format and quality of the data from each source to assess its suitability for the data mart.
4. **Data modeling :** Design the data model for the data mart. Dimensional modeling techniques, such as star schema or snowflake schema, are commonly used in data mart design. Define the dimensions, which represent the various attributes or characteristics of the subject area and the fact tables, which contain the measures or metrics that will be analyzed.
5. **Data extraction and transformation :** Determine the extraction and transformation processes required to populate the data mart with data from the identified sources. This may involve data cleansing, data integration, data aggregation and other transformations to ensure the data is consistent, accurate and aligned with the data mart's requirements.

6. **Define granularity :** Determine the level of granularity at which the data will be stored in the data mart. This depends on the analytical needs of the user group and the level of detail required for reporting and analysis. Strike a balance between storing data at a granular level for detailed analysis and aggregating data for faster query performance.
7. **Establish data governance :** Implement data governance practices within the data mart design. This includes defining data quality standards, data lineage, data security and access controls. Ensure that the data mart design adheres to the organization's data governance framework and policies.
8. **Develop ETL processes :** Create the Extract, Transform, Load (ETL) processes to extract data from the source systems, transform it according to the data mart's requirements and load it into the data mart. This involves defining data mappings, transformations, data validation rules and error handling mechanisms.
9. **Design reporting and analytical capabilities :** Determine the reporting and analytical capabilities that will be provided by the data mart. This can include pre-defined reports, ad-hoc query capabilities, Online Analytical Processing (OLAP), data visualization and other tools or technologies to facilitate data analysis and decision-making.
10. **Performance optimization :** Optimize the performance of the data mart by creating appropriate indexes, defining partitions and implementing caching mechanisms. Fine-tune the data mart's configuration and query performance to ensure efficient and responsive data access.
11. **Test and validate :** Conduct thorough testing and validation of the data mart to ensure its accuracy, reliability and usability. Test the ETL processes, data integrity, query performance and reporting capabilities. Validate the data mart against the business requirements and gather feedback from users to make necessary refinements.
12. **Deploy and maintain :** Deploy the data mart into the production environment and establish a maintenance plan for ongoing updates, data refreshes and performance monitoring. Monitor the data mart's usage, performance and user feedback to identify areas for improvement and make iterative enhancements.

- The design of data marts should be an iterative and collaborative process, involving close collaboration between business stakeholders and data. The specific design process may vary depending on the organization's requirements, technologies used and available resources. It is essential to involve stakeholders, data architects and other relevant teams throughout the design process to ensure alignment with business objectives and optimal utilization.

3.2.3 Cost of Data Marts

- The cost of implementing data marts can vary depending on several factors, including the organization's size, complexity of requirements, chosen technologies and deployment options. Here are some cost considerations to be considered :

1. **Infrastructure costs :** Data marts require hardware and software infrastructure to support their storage and processing needs. This includes servers, storage devices, networking equipment and database management systems. The cost of these components can vary based on factors such as capacity, performance requirements and whether on-premises or cloud-based infrastructure is used.
2. **Licensing and software costs :** Depending on the technology stack chosen for implementing data marts, there may be licensing fees associated with commercial software, databases and analytical tools. The cost can vary depending on the number of users, features required and vendor pricing models. Open-source options can provide cost savings by eliminating licensing fees but may require additional resources for customization and support.
3. **Development and implementation costs :** The cost of developing and implementing data marts includes activities such as data modeling, data integration, ETL (Extract, Transform, Load) processes, data validation and testing. The level of complexity and customization required will impact the overall development costs. Hiring skilled resources or engaging external consultants can also add to the expenses.
4. **Data quality and governance costs :** Ensuring data quality and governance within data marts requires establishing processes, tools and personnel dedicated to data stewardship, data cleansing, metadata management and compliance. These activities may involve additional costs for data quality tools, data profiling, data lineage, data cataloging and regulatory compliance efforts.
5. **Maintenance and support costs :** Ongoing maintenance and support costs should be factored in, including activities such as monitoring data mart performance, applying patches and upgrades, resolving issues and providing user support. The level of complexity, the size of the user base and the need for ongoing enhancements or customizations will influence these costs.
6. **Training and user adoption costs :** Training users on how to utilize the data marts effectively can contribute to the overall cost. This includes training sessions, user documentation and continuous support to ensure user adoption and maximize the value derived from the data marts.

7. **Scalability and expansion costs :** If the organization plans to scale or expand the data marts in the future, additional costs may arise. This can include hardware upgrades, software licensing adjustments, data integration efforts for incorporating new data sources and accommodating additional user requirements.
8. **Cloud-based costs :** If the organization opts for cloud-based data marts, costs can be more flexible and based on usage. Cloud providers typically offer pricing models based on resources consumed, storage capacity, data transfer and computing power. It's important to monitor and optimize cloud costs to avoid unexpected expenses.

- It's worth noting that the cost of data marts can vary significantly depending on the organization's specific needs and circumstances. Conducting a thorough cost analysis, considering both upfront and ongoing expenses, will help in estimating and budgeting for data mart implementation and maintenance. Additionally, exploring cost optimization strategies such as infrastructure consolidation, open-source alternatives and efficient resource utilization can help minimize expenses.

3.2.4 Data Mart Versus Data Warehouse

- Data mart and data warehouse are both components of a Business Intelligence (BI) architecture, but they serve different purposes and have distinct characteristics. Here are the key differences between the two :

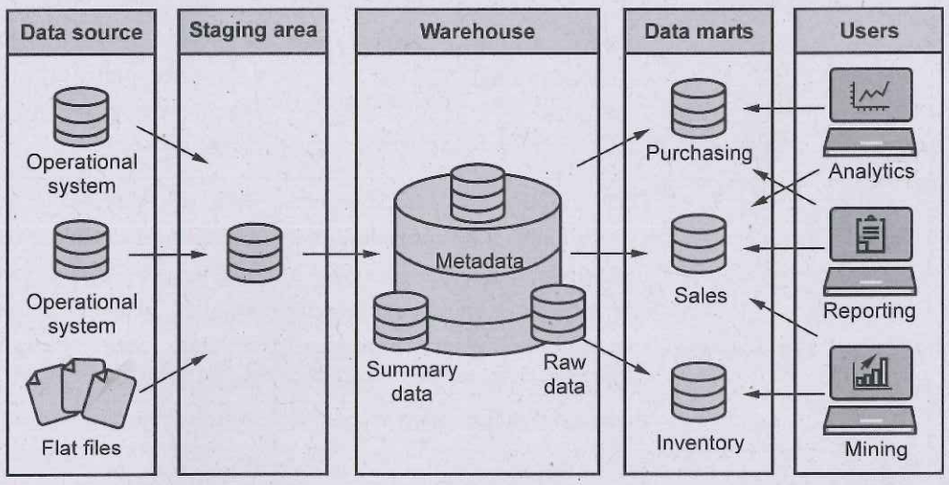

Fig. 3.2.1 Data mart versus data warehouse

Sr. No.	Parameters	Data warehouse	Data mart
1.	Scope	A data warehouse is a centralized repository that stores large volumes of integrated, historical data from multiple sources within an organization. It aims to provide a comprehensive view of the entire organization's data across different subject areas.	A data mart is a subset of a data warehouse and focuses on a specific subject area, department, or user group within the organization. It contains a more focused and tailored set of data that caters to the needs of a specific business function.
2.	Data integration	Data warehouses involve complex data integration processes, including data extraction, transformation and loading (ETL). They consolidate data from various operational systems and external sources, ensuring consistency and uniformity.	Data marts can be created by extracting and transforming data from the data warehouse or other relevant data sources. They are designed to support the specific reporting and analysis requirements of a particular business unit or domain.
3.	Granularity	Data warehouses typically store data at a detailed or transactional level. They contain highly detailed and granular data that supports complex analytics and ad-hoc reporting.	Data marts can have different levels of granularity depending on the requirements of the specific subject area. They can store data at a summarized or aggregated level to support specific reporting and analysis needs.
4.	Architecture	Data warehouses are designed using a dimensional or normalized data model. They often employ a star schema or snowflake schema to facilitate efficient querying and analysis across multiple dimensions and measures.	Data marts typically follow a dimensional modeling approach, such as a star schema, to simplify data structures and facilitate easier querying and analysis within the specific subject area.
5.	Usage	Data warehouses serve as a central hub for enterprise-wide reporting, analytics and decision-making. They support strategic, operational and tactical analysis,	Data marts are tailored to the needs of a specific business unit, department, or user group. They provide a focused view of data, supporting specific

		enabling business users to gain insights and perform historical trend analysis.	operational reporting, analysis, and decision-making within a particular domain.
6.	Data governance	Data warehouses typically have robust data governance practices in place to ensure data quality, consistency and compliance. They involve data stewardship, metadata management and data governance frameworks to maintain the integrity of the data.	Data marts may have their own data governance practices, but they often inherit the data governance policies and frameworks established at the data warehouse level.

- While a data warehouse serves as a centralized repository for integrated enterprise-wide data, a data mart is a specialized subset of a data warehouse that focuses on a specific subject area or user group. Data warehouses are designed for comprehensive and strategic analysis, while data marts cater to the specific needs of individual departments or business functions.

3.3 Partitioning : Introduction

- Partitioning is a critical technique in data warehousing that aims to improve performance, manageability and scalability of data. By dividing a fact table into multiple separate partitions, partitioning can optimize hardware utilization, simplify data management and balance system requirements.
- When selecting a partitioning strategy, it's crucial to consider factors such as data distribution, query patterns, hardware resources and performance goals. Analyzing query workloads, data access patterns and system requirements can help determine the most suitable partitioning strategy. Additionally, ongoing monitoring and maintenance of partitioned tables are necessary to ensure optimal performance and manageability within the data warehouse environment.
- **The partitioning is necessary for following reasons :**
  - For effortless management :

    A data warehouse's fact table might grow to be thousands of terabytes in size. It is quite difficult to maintain this enormous fact table as a single unit. Therefore, partitioning is required.

- To support backup / recovery :

  The fact table must first be partitioned and then it must be loaded entirely with all the data. Only the amount of data needed on a regular basis may be loaded thanks to partitioning. It speeds up system performance and cuts down on loading time.

- To increase performance :

  The query processes may be made better by dividing the fact table into sets of data. Because the query now just searches the pertinent partitions, query speed has improved. It does not have to scan the whole data.

3.3.1 Partition Strategy

- Partitioning strategy in a data warehouse involves determining how to divide data into partitions based on specific criteria to optimize query performance, manageability and scalability. The choice of partitioning strategy depends on factors such as data distribution, query patterns, hardware capabilities and business requirements. Here are some common partitioning strategies used in data warehousing :
  - **Range partitioning :** This strategy involves partitioning data based on a range of values. For example, a sales fact table can be partitioned by date ranges, where each partition contains data for a specific time period (e.g., monthly or quarterly). Range partitioning is effective when queries often filter data based on specific ranges, enabling efficient pruning of irrelevant partitions during query execution.
  - **List partitioning :** List partitioning involves dividing data based on specific values in a partition key column. For instance, a customer dimension table can be partitioned by region, where each partition contains data for a specific region. List partitioning is useful when queries commonly filter data based on discrete values, allowing for efficient partition elimination.
  - **Hash partitioning :** Hash partitioning distributes data across partitions based on a hash function applied to one or more columns. The hash function assigns each row to a specific partition based on the hash value, resulting in a uniform distribution of data. This strategy evenly distributes the data across partitions and can provide a balanced workload across the system, but it may not be suitable for range-based or list-based queries.
  - **Composite partitioning :** Composite partitioning combines multiple partitioning strategies to achieve finer granularity. For example, a large fact table can be partitioned first by range (e.g., date) and then within each range partition, further partitioned by hash or list. This approach allows for both efficient pruning of irrelevant partitions and improved data distribution within each range.

  - **Sub partitioning :** Sub partitioning is used to further divide partitions into smaller sub partitions. This strategy is particularly useful when dealing with large data volumes or when additional levels of data distribution are required for improved parallelism and query performance.
  - **Replication partitioning :** In replication partitioning, the entire dataset is replicated across multiple partitions. This strategy is primarily used for load balancing and fault tolerance, allowing for parallel processing across replicated partitions.
- The choice of partitioning strategy should consider the specific characteristics of the data and workload patterns in the data warehouse. It's important to assess the impact of partitioning on data loading, query performance, maintenance operations and resource utilization. Experimenting with different strategies and considering the trade-offs is crucial to determine the most effective partitioning strategy for a given data warehouse environment.

3.3.2 Vertical Partition

- Vertical partitioning in a data warehouse involves dividing a table vertically based on columns or attributes. It separates columns into multiple smaller tables, each containing a subset of the original columns. This partitioning technique offers several benefits, such as improved query performance, efficient data storage and simplified data management.
- Here's an example to illustrate vertical partitioning in a data warehouse :

  Consider a customer dimension table with the following columns :

  | Customer ID | Name | Age | Gender | Address | City | State |

In vertical partitioning, the table can be divided into two smaller tables based on column subsets :

Table 3.3.1 Customer Info

Customer ID	Name	Age	Gender
1	John Smith	35	M
2	Emma Brown	28	F

Table 3.3.2 Customer location

Customer ID	Address	City	State
1	123 Main Street	New York	NY
2	456 Oak Avenue	Chicago	IL

- In this example, the original customer dimension table is vertically partitioned into two tables. Table 3.3.1 (Customer Info) contains personal details such as customer ID, name, age and gender. Table 3.3.2 (Customer Location) contains information related to address, city and state. Each partition focuses on a specific subset of columns, allowing for more efficient storage and retrieval of data.
- Vertical partitioning splits the data vertically. The following image depicts how vertical partitioning is done.

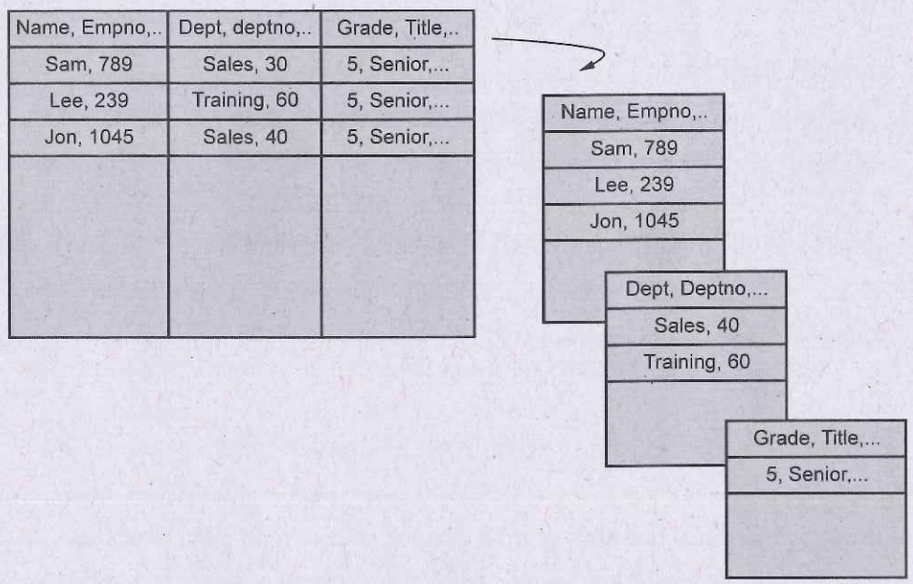

Fig. 3.3.1 Vertical partition

There are two types of Vertical portioning

1. Row splitting
2. Normalization.

3.3.2.1 Vertical Partition Advantages

- Here are some advantages of vertical partitioning in a data warehouse :

1. **Improved query performance :** Vertical partitioning allows for selective column retrieval, which means that queries only access the columns needed for analysis or reporting. This reduces the amount of data read from disk and improves query response times, as it eliminates the need to process and transfer unnecessary columns.
2. **Efficient data storage :** By storing only relevant columns in each partition, vertical partitioning reduces storage requirements. This can be especially beneficial when dealing with wide tables that contain a large number of columns, many of which are not frequently used in queries. The reduced storage footprint can lead to cost savings and optimize data storage resources.
3. **Enhanced data compression :** Vertical partitioning can improve data compression ratios. By partitioning columns based on their data characteristics (e.g., data types, value distributions), it becomes possible to apply different compression techniques to each partition, maximizing compression efficiency. This can result in reduced storage requirements and improved data transfer speeds.
4. **Simplified data management :** Vertical partitioning facilitates data management by allowing different partitions to have distinct maintenance requirements. For example, frequently updated or volatile columns can be placed in a separate partition, making data updates more efficient. Partition-specific maintenance operations, such as backup, restore, or indexing, can be performed independently, reducing the impact on the entire table.
5. **Enhanced security and privacy :** Vertical partitioning can help improve data security and privacy by providing better control over access to sensitive or confidential data. It allows for restricting access to specific partitions containing sensitive columns, reducing the risk of unauthorized access.
6. **Flexibility in data loading and ETL processes :** Vertical partitioning allows for loading or refreshing specific partitions independently, which can speed up data loading processes. In an ETL (Extract, Transform, Load) workflow, it enables selective transformations and data cleansing on specific partitions, reducing the overall processing time.
7. **Better performance in joins and aggregations :** When a query involves joins or aggregations, vertical partitioning can improve performance. Since each partition contains a subset of columns, the query optimizer can eliminate unnecessary column-to-column or row-to-column operations, resulting in faster join and aggregation processing.

- It's important to note that the effectiveness of vertical partitioning depends on factors such as data characteristics, query patterns and the capabilities of the underlying database or data warehouse system. Careful analysis and consideration of the data usage patterns and performance requirements are crucial when deciding on whether to implement vertical partitioning in a data warehouse.

3.3.2.2 Vertical Partition Disadvantages

- While vertical partitioning in a data warehouse offers several advantages, it also has some potential disadvantages. Here are a few limitations or challenges associated with vertical partitioning :

1. **Increased complexity :** Vertical partitioning introduces complexity to the data warehouse design and management. Breaking a table into multiple partitions requires careful planning and consideration of the appropriate column subsets for each partition. This complexity can add overhead to maintenance tasks, such as schema changes, data loading and query optimization.
2. **Limited query flexibility :** Vertical partitioning restricts the ability to perform queries that involve multiple columns from different partitions. If a query requires accessing columns from multiple partitions, it may result in increased data retrieval and processing overhead due to the need for additional join or union operations. This can impact query performance and overall system efficiency.
3. **Potential for data skew :** In vertical partitioning, there is a risk of data skew when certain partitions contain significantly more data than others. This can lead to imbalance in resource utilization, query performance degradation and uneven data distribution across storage devices or servers. Addressing data skew may require additional efforts in monitoring and rebalancing partitions to ensure optimal performance.
4. **Data integrity challenges :** Vertical partitioning may introduce complexities in maintaining data integrity, especially when updates or modifications affect multiple partitions. Coordinating updates across partitions and ensuring consistent data can become more challenging. Transaction management and concurrency control mechanisms need to be carefully considered and implemented to maintain data integrity in a vertically partitioned environment.

5. **Increased metadata complexity :** Managing metadata and data dictionaries becomes more complex with vertical partitioning. Maintaining accurate and up-to-date metadata information for each partition can be cumbersome. Ensuring that metadata systems and tools are designed to handle partitioned data appropriately becomes crucial.
6. **Impact on ETL processes :** Vertical partitioning may require modifications to ETL (Extract, Transform, Load) processes. If transformations or data cleansing operations are performed on the entire table, they need to be adjusted to operate on specific partitions. This can lead to additional development and maintenance efforts for ETL pipelines.
7. **Limited scalability :** While vertical partitioning can improve query performance and storage efficiency, it may not provide the same scalability benefits as horizontal partitioning. Adding more partitions may not inherently distribute the workload across multiple servers or nodes as effectively as horizontal partitioning does.

- It's important to carefully assess the specific characteristics of the data, query patterns and overall requirements of the data warehouse before deciding to implement vertical partitioning. A thorough evaluation of the trade-offs and potential challenges will help determine if vertical partitioning is the most suitable approach for a given data warehousing scenario.

3.3.3 Normalization

- The relational database organization process is known as **normalization,** which is the industry standard. This technique reduces space since the rows are combined into one row. The normalization process is demonstrated in the following tables.
- Normalization in a data warehouse refers to the process of designing the database schema in such a way that it eliminates redundancy and improves data integrity. The main goal of normalization is to minimize data redundancy by organizing data into multiple related tables, reducing data duplication and ensuring data consistency.
- Normalization follows a set of rules called **normal forms,** which define the levels of normalization. The most commonly used normal forms in data warehousing are :
  - **First Normal Form (1NF) :** In 1NF, the data is organized into tables with rows and columns and each column contains atomic values (indivisible). There should be no repeating groups or arrays within a single attribute.

- **Second Normal Form (2NF) :** In 2NF, the table is in 1NF and all non-key attributes are fully dependent on the entire primary key. If any non-key attribute depends on only part of the primary key, it should be moved to a separate table.
- **Third Normal Form (3NF) :** In 3NF, the table is in 2NF and there are no transitive dependencies. Transitive dependencies occur when a non-key attribute depends on another non-key attribute rather than directly on the primary key.

- Normalization helps to eliminate data anomalies and inconsistencies, improves query performance and simplifies database maintenance. However, in a data warehouse environment, where the focus is on analytical processing and complex queries, denormalization is often applied to enhance performance by aggregating and storing data in a more flattened structure.
- Denormalization involves combining tables, duplicating data and introducing redundancy to optimize query performance. It aims to strike a balance between performance and data integrity, considering the specific reporting and analysis requirements of the data warehouse.
- It's important to note that the level of normalization or denormalization applied in a data warehouse depends on factors such as the type of analysis, query patterns, data volume and performance requirements. Designing an effective data warehouse schema involves making informed decisions based on these factors to achieve the desired balance between normalization and denormalization.
- Table before normalization

Produc_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-Aug-13	16	Sunny	Bangalore	S
40	5	2.50	3-Aug-13	64	San	Mumbai	W
45	7	5.66	3-Aug-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-Aug-13	16
40	5	2.50	3-Aug-13	64
45	7	5.66	3-Aug-13	16

3.3.3.1 Advantages of Normalization

- Normalization in a data warehouse offers several advantages that contribute to improved data quality, flexibility and maintainability. Here are some key benefits of normalization in a data warehouse :

1. **Data integrity :** Normalization helps maintain data integrity by minimizing data redundancy and inconsistencies. By organizing data into separate tables with well-defined relationships, normalization reduces the chances of data anomalies, such as update anomalies or inconsistent data. This ensures that the data warehouse contains accurate and reliable information.
2. **Flexible data model :** Normalized data models provide flexibility in accommodating changes to the data warehouse structure. As new data sources are added or existing ones evolve, normalization allows for easier integration of new data elements and modifications to the schema. This flexibility enables the data warehouse to adapt to evolving business requirements without significant disruptions.
3. **Efficient storage :** Normalization eliminates data redundancy by storing data in a more compact and efficient manner. Reducing data duplication can result in significant storage savings, especially for large-scale data warehouses. Additionally, normalized structures are optimized for efficient indexing and retrieval, leading to improved query performance.
4. **Simplified data maintenance :** With a normalized data model, data maintenance tasks become more manageable. Updates, inserts and deletes can be performed with precision, as changes only need to be made in one place, ensuring data consistency. This simplifies data management processes, reduces the likelihood of errors and facilitates easier system maintenance.
5. **Enhanced data consistency :** Normalization promotes data consistency across the data warehouse. With well-defined relationships and constraints, data anomalies and discrepancies are minimized. This consistency ensures that data across different tables remains synchronized, supporting accurate reporting and analysis.

6. **Easier querying and analysis :** Normalized data models simplify query development and analysis. By separating data into logical tables based on their attributes, it becomes easier to understand and navigate the data structure. Queries involving joins across related tables can be executed more efficiently, allowing for complex analysis and reporting.
7. **Scalability and extensibility :** Normalization supports the scalability and extensibility of the data warehouse. As data volumes increase or new data sources are added, the normalized structure enables efficient data integration and management. The ability to add new tables or modify existing ones without impacting the entire data warehouse makes it easier to accommodate growth and change.

- While normalization brings several advantages, it's important to strike a balance between normalization and denormalization in the data warehouse design. Over-normalization can lead to excessive complexity, increased query complexity and reduced query performance. Therefore, a thoughtful approach is required to determine the appropriate level of normalization based on the specific requirements and characteristics of the data warehouse environment.

3.3.3.2 Disadvantages of Normalization

- While normalization offers several advantages, it also has certain disadvantages when applied in a data warehouse context. Here are some potential drawbacks of normalization :

1. **Increased complexity of queries :** Normalization often involves breaking data into multiple tables and establishing relationships between them. As a result, querying normalized data requires more complex join operations across multiple tables. Complex queries can be harder to write, understand and optimize, which can impact query performance and increase the complexity of the overall data warehouse system.
2. **Reduced query performance :** Normalization can lead to reduced query performance, particularly in scenarios where complex joins are required to retrieve data from multiple tables. Joining large tables with numerous relationships can introduce additional overhead and impact query response times. Denormalization techniques may be necessary to optimize query performance in such cases.
3. **Data integrity challenges :** While normalization improves data integrity, it can introduce challenges related to data integrity management. Maintaining referential integrity across multiple tables requires careful attention and validation to ensure data consistency. Updates, inserts and deletes need to be properly managed across related tables, which can increase the complexity of data maintenance operations.

4. **Increased storage and disk I/O :** Normalization can increase storage requirements and disk I/O. Since data is distributed across multiple tables, it may lead to more disk reads and writes during query execution. In scenarios where data duplication is avoided, retrieving data from multiple tables can result in additional disk I/O operations, impacting overall system performance.
5. **Difficulty in analyzing historical data :** Normalized data models may pose challenges when analyzing historical data. As new data is added over time, it may require joining multiple tables with historical and current data. Constructing historical views or tracking changes over time can be more complex, as it involves navigating through multiple related tables.
6. **Impact on ETL processes :** Extract, Transform, Load (ETL) processes can be more complex and time-consuming in a normalized data warehouse. Data transformation and consolidation from multiple sources into normalized tables can require additional effort and resources. ETL pipelines may need to perform complex join operations and ensure data integrity across various tables during the loading phase.
7. **Limited flexibility for Ad hoc analysis :** Normalized data models can limit the flexibility for ad hoc analysis or agile reporting. The structure of normalized tables may not align with the specific needs of ad hoc queries or flexible reporting requirements. In such cases, denormalization or the use of data marts or dimensional modeling techniques may be necessary to provide more flexibility for agile analysis.

- It's important to strike a balance between normalization and denormalization in data warehousing, considering the specific requirements of the system and the nature of the data being stored. Evaluating the trade-offs and considering factors such as query patterns, performance requirements and data complexity can help determine the appropriate level of normalization to apply in a data warehouse.

3.3.4 Row Splitting

- Row splitting in a data warehouse refers to the process of splitting a single row of source data into multiple rows in the data warehouse. This technique is typically used when dealing with multi-valued or repeating attributes in the source data that need to be expanded into separate rows for analysis and reporting purposes.

Here is an example to illustrate row splitting :

- Let's say you have a source table with the following structure :

Source Table :

0	Order ID	Products
1	100	A,B,C
2	101	D,E

- In this example, the "products" column contains multiple values separated by commas. To analyze and report on individual products, you may need to split these values into separate rows in the data warehouse. The resulting table after row splitting might look like this :

Data warehouse table :

Customer ID	Order ID	Products
1	100	A
1	100	B
1	100	C
2	101	D
2	101	E

- As you can see, each product from the source table is now represented as a separate row in the data warehouse table. This allows for more granular analysis and reporting on individual products.
- Row splitting can be performed during the ETL (Extract, Transform, Load) process, where the transformation step includes splitting the rows based on the multi-valued attribute. This can be done using various techniques, such as using scripting languages, SQL queries or ETL tools that provide capabilities for data manipulation.
- It's important to note that row splitting should be used judiciously, as it can lead to an increase in the number of rows and potentially impact performance and storage requirements in the data warehouse. The decision to perform row splitting should be based on the specific reporting and analysis needs of the data warehouse and should be balanced with the overall design and performance considerations.

3.3.4.1 Advantages of Row Splitting

- Row splitting, also known as **vertical partitioning,** is a technique used in data warehousing to divide a table's rows into multiple smaller partitions based on specific criteria. This approach offers several advantages in terms of data management and query performance. Here are some benefits of row splitting in a data warehouse :

1. **Improved query performance :** Row splitting can enhance query performance by allowing for parallel processing of queries across multiple partitions. Queries executed on a subset of data within a partition are typically faster than scanning the entire table. Parallel processing enables efficient utilization of computing resources, resulting in quicker query response times.
2. **Efficient data loading :** Row splitting facilitates efficient data loading processes. Instead of loading the entire table, data can be loaded partition by partition. This allows for faster and more targeted loading, as only the relevant partition needs to be updated. It reduces the impact on other partitions and minimizes downtime during data loading operations.
3. **Reduced storage requirements :** Row splitting can help optimize storage space by eliminating the need to store all data in a single table. By dividing the table into smaller partitions, storage requirements can be reduced. This is particularly beneficial when dealing with large tables that contain a significant amount of data. It allows for more efficient utilization of storage resources.
4. **Simplified data management :** Managing data in smaller partitions can simplify administrative tasks. Partition-specific maintenance operations, such as backup, restore, or indexing, can be performed independently, reducing the impact on the entire table. It also enables more granular data management, making it easier to address specific subsets of data within the data warehouse.
5. **Scalability and parallel processing :** Row splitting enables scalability in a data warehouse environment. As the data volume grows, new partitions can be added without impacting the existing ones. This allows for horizontal scaling, distributing the workload across multiple servers or nodes. Parallel processing across partitions allows for increased processing power and improved system performance.
6. **Data security and privacy :** Row splitting can enhance data security and privacy. It allows for restricting access to specific partitions, ensuring that sensitive data is only accessible to authorized users or roles. This fine-grained access control helps protect confidential information and ensures data privacy compliance.

7. **Optimized data analysis :** Row splitting can improve data analysis capabilities. By dividing the data into meaningful partitions based on specific criteria (e.g., time intervals, geographical regions), analytical queries can focus on a specific subset of data relevant to the analysis. This targeted analysis can result in more efficient and accurate insights.

- It's important to consider the nature of the data and the query patterns when deciding on row splitting as a partitioning strategy. Factors such as data distribution, partitioning criteria and query requirements should be carefully analyzed to determine the optimal partitioning approach for a given data warehouse scenario.

3.3.4.2 Disadvantages of Row Splitting

- While row splitting, in a data warehouse offers several advantages, it also has certain disadvantages or challenges. Here are some potential drawbacks of row splitting :

1. **Increased complexity of queries :** Row splitting can introduce complexity in query design and execution. Queries that need to access data from multiple partitions may require complex join operations or union operations to combine the results. This complexity can make query development, optimization and troubleshooting more challenging.
2. **Data skew :** In row splitting, there is a risk of data skew, where certain partitions contain significantly more data than others. Data skew can result in imbalance across partitions, uneven resource utilization and performance degradation. Addressing data skew may require additional monitoring, rebalancing, or redistribution of data to ensure optimal performance.
3. **Limited joins and relationships :** Horizontal partitioning can restrict the ability to perform joins and establish relationships across partitions. Queries involving data from multiple partitions may require additional processing steps, such as joining intermediate results, which can impact query performance and overall system efficiency.
4. **Data integrity challenges :** Maintaining data integrity can become more complex with row splitting. Updates, inserts or deletes that affect multiple partitions require careful coordination to ensure data consistency. Transaction management and concurrency control mechanisms need to be properly implemented to avoid data integrity issues across partitions.

5. **ETL complexity :** Row splitting can introduce complexities in Extract, Transform, Load (ETL) processes. Loading data into multiple partitions may require additional logic and transformations to distribute the data correctly. Managing data updates or inserts across partitions can also add complexity to the ETL workflows.
6. **Limited flexibility in data analysis :** Row splitting can restrict certain types of data analysis that require access to the entire dataset. Analytical operations that involve aggregating or correlating data across partitions may become more challenging. Analysis involving historical or trend analysis spanning multiple partitions can also be more complex to perform.
7. **Partition maintenance overhead :** Managing and maintaining multiple partitions adds overhead to administrative tasks. Backup and restore operations need to be performed on a partition-by-partition basis, which can increase the complexity and time required for data management operations. Monitoring and managing partition metadata can also become more demanding.

- It's important to carefully consider the trade-offs and evaluate the specific requirements of the data warehouse before implementing row splitting as a partitioning strategy. The choice of partitioning technique should align with the characteristics of the data, query patterns, scalability needs and overall performance goals of the data warehouse system.

3.3.5 Comparison of Normalization and Row Splitting

Sr. No.	Parameters	Normalization	Row splitting
1.	Purpose	The primary purpose of normalization is to eliminate data redundancy, improve data integrity, and ensure data consistency by organizing data into well-structured, atomic tables.	The purpose of row splitting, is to divide a table's rows into multiple partitions based on specific criteria. The goal is to improve query performance, data loading efficiency and scalability.
2.	Data organization	Normalization focuses on organizing data within a table by breaking it down into multiple related tables. It aims to reduce data redundancy and ensure that each table stores a unique set of data attributes.	Row splitting organizes data within a table by dividing rows into multiple partitions based on predetermined criteria, such as time intervals, geographical regions, or other relevant factors.

3.	Data redundancy	Normalization aims to eliminate or minimize data redundancy by breaking data into separate tables and establishing relationships between them.	Row splitting does not directly address data redundancy. However, by dividing data into partitions, it can indirectly reduce redundancy by storing relevant data together within each partition.
4.	Query performance	Normalized tables can require more complex joins across multiple tables, which can impact query performance, especially for queries that involve retrieving data from multiple related tables.	Row splitting can improve query performance by allowing parallel processing of queries across multiple partitions. Queries executed on a subset of data within a partition can be faster than scanning the entire table.
5.	Data loading and maintenance	Normalization can make data loading and maintenance operations more complex, as data needs to be inserted, updated, or deleted across multiple related tables. It may require more effort for data integration and management.	Row splitting simplifies data loading by allowing for efficient loading of individual partitions. Maintenance operations can be performed on specific partitions, reducing the impact on the entire dataset.
6.	Flexibility and analytical capabilities	Normalization provides flexibility in accommodating changes to the data model and allows for efficient data integration. It is well-suited for transactional systems and supports data integrity. It may limit certain types of flexible analysis, such as ad hoc querying and reporting.	Row splitting can improve scalability, enable parallel processing and support targeted analysis on specific partitions. It may offer more flexibility for certain types of analysis and reporting needs.

3.3.6 Horizontal Partition

- Horizontal partitioning, also known as **data partitioning or data sharding,** is a technique used in data warehousing to divide a large table horizontally into smaller, more manageable parts called **partitions.** Each partition contains a subset of the table's rows based on a defined criteria or range. The goal of horizontal partitioning is to improve performance, manageability and scalability of data warehouse operations.

- Here are some key aspects of horizontal partitioning in a data warehouse :
  - **Partitioning criteria :** The table is divided into partitions based on a specific column or set of columns, often referred to as the partition key. Common partitioning criteria include ranges of values (e.g., based on date ranges or numeric ranges), list of discrete values (e.g., based on specific regions or customer IDs), or hash values (e.g., based on a hash function applied to a specific column). The choice of partitioning criteria depends on the data distribution, query patterns and performance requirements.
  - **Partition management :** Each partition is stored separately, either physically or logically, with its own file or tablespace. This allows for independent management of each partition, such as separate backup and restore operations, data archiving, and partition-level optimizations.
  - **Query optimization :** Horizontal partitioning can improve query performance by reducing the amount of data accessed. When queries involve the partitioning criteria, the system can prune irrelevant partitions, minimizing disk I/O and improving response times. For example, if a query filters data by a specific date range, only relevant partitions need to be accessed, avoiding a full table scan.
  - **Load balancing and scalability :** By distributing data across multiple partitions, horizontal partitioning can help achieve load balancing and scalability in a data warehouse environment. It allows for distributing data and processing across multiple servers or storage devices, enabling parallel processing and accommodating larger data volumes.
  - **Partition maintenance :** Horizontal partitioning may require additional maintenance operations compared to a non-partitioned table. These operations include adding or removing partitions as data grows or changes, ensuring data integrity across partitions and managing the partitioning strategy as the business requirements evolve.
- It's important to note that horizontal partitioning is typically transparent to end users and applications. The partitioning details are handled by the Database Management System (DBMS) or data warehouse platform and queries can be written without explicit consideration of the partitions. Horizontal partitioning is a valuable technique in data warehousing to optimize query performance, improve manageability and support scalability. However, its effectiveness depends on the specific characteristics of the data, workload patterns and the capabilities of the underlying database or data warehouse system.

3.3.7 Advantages of Horizontal Partition

- Horizontal partitioning, also known as **sharding,** is a technique used in data warehousing to divide a large database table into smaller, more manageable parts. Each partition contains a subset of the table's rows, typically based on a defined criteria such as a range of values or a specific attribute. Horizontal partitioning offers several advantages in a data warehouse environment :

1. **Improved performance :** By dividing a large table into smaller partitions, queries and data retrieval operations can be distributed across multiple nodes or servers, allowing for parallel processing. This can significantly enhance query performance and reduce response times, especially for complex queries that involve large data sets.
2. **Enhanced scalability :** Horizontal partitioning enables the data warehouse to handle larger volumes of data by distributing the load across multiple servers or storage devices. As the data grows, new partitions can be added to accommodate the increased volume, allowing for seamless scalability without impacting existing partitions.
3. **Efficient data maintenance :** Partitioning can simplify and optimize data maintenance tasks, such as data loading, updates and deletions. Instead of performing these operations on the entire table, they can be targeted to specific partitions, reducing the overhead and improving overall efficiency.
4. **Cost optimization :** Horizontal partitioning can help optimize storage costs by allowing different partitions to be stored on different types of storage devices based on their usage patterns and requirements. Frequently accessed partitions can be stored on high-performance storage systems, while less frequently accessed partitions can be stored on lower-cost storage devices, thus balancing performance and cost.
5. **Increased availability and fault tolerance :** Partitioning can improve the availability and fault tolerance of the data warehouse. If one partition or server fails, the remaining partitions can still be accessed, ensuring continuous availability of the data. Additionally, partitioning allows for easier backup and recovery operations at the partition level, reducing the impact of failures.
6. **Data segregation and security :** Partitioning can be used to enforce data segregation and enhance security. Different partitions can be assigned different access controls and security policies based on the sensitivity or ownership of the data they contain. This allows for fine-grained control over data access and helps protect sensitive information.

7. **Data archiving and retention :** Partitioning facilitates data archiving and retention strategies. Historical or infrequently accessed data can be moved to separate partitions or storage systems, freeing up resources and improving the performance of the active data set. It also simplifies the process of purging or deleting expired data, ensuring compliance with data retention policies.

- Horizontal partitioning offers advantages such as improved performance, scalability, efficient data maintenance, cost optimization, increased availability and fault tolerance, data segregation and security and data archiving and retention. These benefits make it a valuable technique in data warehousing, particularly for handling large and growing data volumes.

3.3.8 Disadvantages of Horizontal Partition

- While horizontal partitioning in a data warehouse offers several advantages, it also has some potential disadvantages that should be considered :

1. **Query complexity :** Partitioning a table horizontally can introduce complexity when querying the data. Queries that require accessing multiple partitions may need additional logic or co-ordination to retrieve and combine the results. This complexity can affect query performance and require more sophisticated query optimization techniques.
2. **Data skew :** Horizontal partitioning relies on a defined partitioning criteria, such as a range of values or attribute values. If the data distribution is uneven or skewed, meaning that certain partitions have significantly more data than others, it can lead to performance issues. Imbalanced data distribution may result in uneven resource utilization, slower queries on heavily loaded partitions and challenges in maintaining data consistency.
3. **Data integrity :** Maintaining data integrity can be more challenging with horizontal partitioning. Operations such as updates, inserts and deletions that affect multiple partitions require coordination and synchronization across partitions to ensure data consistency. Failure to properly manage data integrity can lead to inconsistencies and data corruption.
4. **Partition management overhead :** Managing multiple partitions introduces additional administrative overhead. Partition creation, modification and deletion require careful planning and co-ordination. The partitioning strategy needs to be designed upfront and may require ongoing monitoring and adjustments as data volumes and access patterns change. This adds complexity to the data warehouse management tasks.

5. **Increased storage overhead :** Horizontal partitioning can lead to increased storage overhead. Each partition typically requires additional metadata and storage space for storing the partitioning criteria and boundaries. While partitioning can improve query performance, it comes at the cost of additional storage requirements.
6. **Limited cross-partition queries :** Some types of queries that involve joining or aggregating data across multiple partitions may become more complex and less efficient. These queries often require data movement between partitions or coordination across multiple nodes, which can impact performance and increase query execution time.
7. **Data redistribution challenges :** If the partitioning strategy needs to be modified due to changes in data distribution or query patterns, redistributing the data across partitions can be a complex and time-consuming process. It may require moving and reorganizing large amounts of data, which can impact the availability and performance of the data warehouse during the redistribution process.
8. **Impact on indexing :** Horizontal partitioning can affect the effectiveness of indexing strategies. Partitioning may require indexing to be applied at the partition level, which can impact the overall efficiency of index usage. Designing and managing indexes in a partitioned environment requires careful consideration to ensure optimal query performance.

- It's important to carefully evaluate these disadvantages against the specific requirements and characteristics of your data warehouse environment before implementing horizontal partitioning. Depending on the nature of your data and workload, the benefits of partitioning may outweigh these disadvantages, or alternative partitioning strategies may be more suitable.

3.3.9 Comparison of Horizontal and Vertical Partition

- Horizontal and vertical partitioning are two different techniques used in data warehousing to divide large database tables into smaller, more manageable parts. Let's compare these two approaches :

Sr. No.	Parameters	Horizontal partition	Vertical partition
1.	Structure :	In horizontal partitioning, the rows of a table are divided into partitions based on a defined criteria, such as a range of values or specific attributes. Each partition contains a subset of rows but includes all columns of the original table.	In vertical partitioning, the columns of a table are divided into partitions. Each partition contains a subset of columns but includes all rows of the original table.
2.	Data distribution :	Horizontal partitioning distributes the rows of a table across multiple partitions or storage devices. Each partition can reside on a different server or node, allowing for parallel processing and improved performance.	Vertical partitioning divides the columns of a table, keeping all rows together. It allows for the segregation of columns with different access patterns or data types. The partitions may be stored within the same server or node.
3.	Performance :	Horizontal partitioning enables parallel processing, as queries can be executed concurrently on different partitions. This can result in improved query performance, especially for complex queries involving large data sets. However, cross-partition queries may introduce additional complexity and overhead.	Vertical partitioning can improve query performance by reducing the data volume accessed for each query. Queries that only require a subset of columns can skip unnecessary columns, leading to faster execution. However, vertical partitioning may introduce additional joins or lookups to retrieve the complete data.
4.	Scalability :	Horizontal partitioning offers better scalability as new partitions can be added to accommodate increasing data volumes. It allows for distributed processing and storage, enabling the system to handle larger data sets and growing workloads.	Vertical partitioning does not inherently address scalability concerns. Adding new columns to a table does not automatically distribute the load. It may require redistributing the entire table or modifying the partitioning strategy.

5.	Maintenance and flexibility :	Horizontal partitioning simplifies data maintenance tasks, such as data loading, updates and deletions. These operations can be targeted to specific partitions, reducing the overhead. However, managing partitioning boundaries and redistributing data can introduce additional administrative tasks.	Vertical partitioning simplifies maintenance tasks related to specific columns. For example, adding or dropping a column only affects the partition containing that column. However, managing and modifying vertical partitions may require schema changes and careful coordination.
6.	Data integrity :	Ensuring data integrity across partitions can be more complex in horizontal partitioning. Operations that affect multiple partitions, such as updates or deletes, require co-ordination to maintain data consistency.	Vertical partitioning does not introduce significant challenges to data integrity since all rows remain intact. However, consistency needs to be maintained across the different vertical partitions.
7.	Storage overhead :	Horizontal partitioning may introduce additional storage overhead, as each partition requires metadata and storage space for partitioning criteria. However, the storage overhead can be reduced by distributing the partitions across different storage devices.	Vertical partitioning reduces storage overhead by only storing the necessary columns in each partition. It can save storage space, especially when dealing with tables containing many columns.

Review Questions

1. *Explain categories of metadata.* **(Refer section 3.1.1)** **Marks 7**
2. *Explain data mart versus data warehouse.* **(Refer section 3.2.4)** **Marks 6**
3. *Explain partition strategy.* **(Refer section 3.3.1)** **Marks 6**
4. *Write down advantages of vertical partitioning.* **(Refer section 3.3.2.1)** **Marks 6**
5. *Write down disadvantages of vertical partitioning.* **(Refer section 3.3.2.2)** **Marks 6**
6. *Explain normalization.* **(Refer section 3.3.3)** **Marks 5**

7. *Explain advantages of normalization.* **(Refer section 3.3.3.1)** **Marks 6**
8. *Explain disadvantages of normalization.* **(Refer section 3.3.3.2)** **Marks 6**
9. *Comparison of normalization and row splitting.* **(Refer section 3.3.5)** **Marks 7**
10. *Comparison of horizontal and vertical partition.* **(Refer section 3.3.9)** **Marks 7**

3.4 Two Marks Questions with Answers

Q.1 What is meta data ?

Ans. : Metadata refers to the information about the data stored in a data warehouse. It provides details about the structure, meaning and usage of the data, allowing users to understand and effectively utilize the information contained within the data warehouse.

Q.2 What is data mart ?

Ans. : Data mart is a subset of a data warehouse that focuses on a specific subject area or department within an organization. It is a smaller, more focused version of a data warehouse and is designed to meet the specific reporting and analytical needs of a particular business unit or user group.

Q.3 What involves in designing of data mart ?

Ans. : Designing data marts in a data warehouse involves several steps and considerations to ensure they effectively meet the analytical and reporting needs of specific user groups or departments.

Q.4 What is partitioning ?

Ans. : Partitioning is a critical technique in data warehousing that aims to improve performance, manageability and scalability of data. By dividing a fact table into multiple separate partitions, partitioning can optimize hardware utilization, simplify data management and balance system requirements.

Q.5 What is vertical partition ?

Ans. : Vertical partitioning in a data warehouse involves dividing a table vertically based on columns or attributes. It separates columns into multiple smaller tables, each containing a subset of the original columns. This partitioning technique offers several benefits, such as improved query performance, efficient data storage and simplified data management.

Q.6 What is normalization ?

Ans. : The relational database organization process is known as normalization, which is the industry standard. This technique reduces space since the rows are combined into one row.

Q.7 What is horizontal partition ?

Ans. : Horizontal partitioning, also known as data partitioning or data sharding, is a technique used in data warehousing to divide a large table horizontally into smaller, more manageable parts called partitions. Each partition contains a subset of the table's rows based on a defined criteria or range. The goal of horizontal partitioning is to improve performance, manageability and scalability of data warehouse operations.

❑❑❑

UNIT IV

4 Dimensional Modeling and Schema

Syllabus

Dimensional Modeling - Multi - Dimensional Data Modeling - Data Cube - Star Schema - Snowflake schema - Star Vs Snowflake schema - Fact constellation Schema - Schema Definition - Process Architecture - Types of Data Base Parallelism - Datawarehouse Tools.

Contents