Unit 2: Data Warehousing Processes and Architectures

1. System Processes in Data Warehousing

1.1 Definition

System processes in a data warehouse refer to the various tasks and workflows that ensure **data extraction, transformation, loading (ETL), storage, and retrieval** for analytical processing. These processes help maintain **data integrity, consistency, and accessibility**.

1.2 Key System Processes

1. **Data Extraction**: Collecting data from multiple sources (databases, files, web services).
2. **Data Transformation**: Cleaning, filtering, and converting data into a standardized format.
3. **Data Loading**: Storing the transformed data into the data warehouse.
4. **Indexing and Partitioning**: Optimizing data storage for efficient querying.
5. **Query Processing**: Managing user queries and generating reports.
6. **Data Backup and Recovery**: Ensuring data security and disaster recovery.

---

2. Query Management Process

2.1 Definition

The **Query Management Process (QMP)** is responsible for optimizing and handling queries efficiently within a data warehouse. It ensures that users receive **fast, accurate, and optimized results** for analytical processing.

2.2 Components of Query Management

1. **Query Parsing**: Validates the syntax and structure of a query.
2. **Query Optimization**: Finds the most efficient way to execute a query.
3. **Query Execution**: Runs the optimized query on the warehouse.
4. **Query Caching**: Stores frequently used query results for faster access.
5. **Load Balancing**: Distributes query processing across multiple servers.

2.3 Importance of Query Management

- Enhances **query performance** and reduces response time.
- Prevents **system overload** by balancing query execution.
- Ensures **data consistency** by managing concurrent queries.

---

3. Process Architecture of Data Warehousing

3.1 Definition

The **Process Architecture** of a data warehouse defines the structured workflow of how data is extracted, processed, stored, and retrieved for business intelligence applications.

3.2 Components of Data Warehouse Architecture

1. **Source Layer**:
   - Includes **OLTP databases, external APIs, and files** where data originates.
2. **ETL Layer (Extract, Transform, Load)**:
   - Extracts data, cleans it, and loads it into the warehouse.
3. **Data Warehouse Storage**:
   - Central repository that contains **fact and dimension tables**.
4. **OLAP Engine (Online Analytical Processing)**:
   - Enables multidimensional data analysis.
5. **Reporting & Visualization**:
   - Tools like **Power BI, Tableau** help generate reports and dashboards.
6. **Metadata Management**:
   - Stores **data definitions, relationships, and user access logs**.

3.3 Types of Data Warehouse Architectures

1. **Single-Tier Architecture**: Data warehouse and source systems in the same database (less common).
2. **Two-Tier Architecture**: Separate ETL and analytical processing layers.
3. **Three-Tier Architecture**: Most common; includes **source systems, data warehouse, and front-end tools**.

---

4. Database Schema in Data Warehousing

4.1 Definition

A **Database Schema** is the logical structure that defines how data is stored and organized in a data warehouse.

4.2 Types of Tables

1. **Fact Table**:
    - Stores **measurable business data** (e.g., sales amount, revenue).
    - Contains **foreign keys** referencing dimension tables.
    - Example:

Date	Product_ID	Store_ID	Sales_Amount
01-01-2024	P101	S202	$1000

2. **Dimension Table**:
    - Stores **descriptive attributes** related to facts.
    - Example:

Product_ID	Product_Name	Category
P101	Laptop	Electronics

---

5. Database Schemas in Data Warehousing

5.1 Star Schema

- The **simplest and most common** schema.
- **Fact table** at the center, connected to multiple **dimension tables**.
- **Advantages**:
    - Simple and easy to query.
    - Faster query execution due to denormalization.
- **Disadvantages**:

- Data redundancy in dimension tables.

- **Example**:

```
    Customers   Products   Time   Stores
          \         |         /
           \       |       /
             Fact Table
```

5.2 Snowflake Schema

- Extension of the **Star Schema**, where dimension tables are **normalized**.
- Reduces **data redundancy** but increases **complexity**.
- **Advantages**:
  - Saves storage space.
  - More structured.
- **Disadvantages**:
  - Slower query performance due to joins.

5.3 Star Flake Schema

- **Hybrid approach** combining Star and Snowflake schemas.
- Some dimensions are **denormalized (Star Schema)**, while others are **normalized (Snowflake Schema)**.
- Balances **performance and storage optimization**.

5.4 Multi-Dimensional Schema

- Extends the above schemas to handle **complex hierarchies** and **multiple fact tables**.
- Used in **large-scale data warehouses**.

---

6. Data Partitioning in Data Warehousing

6.1 Definition

Partitioning refers to **dividing a large dataset** into smaller, manageable chunks to improve **query performance and storage efficiency**.

6.2 Types of Partitioning

(a) Horizontal Partitioning

- Divides a table into **rows** based on criteria (e.g., time-based partitions).
- Example:
  - **Sales_2023**: Contains sales data from 2023.
  - **Sales_2024**: Contains sales data from 2024.
- **Advantages**:
  - Faster data retrieval for specific time periods.
  - Improves query performance.
- **Disadvantages**:
  - Complexity in managing multiple partitions.

(b) Vertical Partitioning

- Divides a table into **columns** to store frequently accessed data separately.
- Example:
  - Table 1: **Product_ID, Product_Name** (used for searching).
  - Table 2: **Product_ID, Price, Stock** (used for inventory management).
- **Advantages**:
  - Improves query speed for specific attributes.
  - Reduces memory usage.
- **Disadvantages**:
  - Requires additional joins for full data retrieval.

(c) Hardware Partitioning

- Distributes data across **multiple physical servers or storage devices**.
- Uses **RAID (Redundant Array of Independent Disks)** for fault tolerance.
- **Advantages**:
  - Enhances performance through parallel processing.
  - Improves fault tolerance and data availability.
