Delta-exam-topics-Data-Warehouse

- Delta-exam-topics-Data-Warehouse
  - 1. OLAP vs. OLTP
  - 2. Facts
  - 3. Dimensions
  - 4. Star Schema
    - Characteristics of a Star Schema
    - Advantages of Star Schema
    - Disadvantages of Star Schema
    - Conclusion
  - 5. Snowflake Schema
    - Characteristics of a Snowflake Schema
    - Advantages of Snowflake Schema
    - Disadvantages of Snowflake Schema
  - 6. Star Schema vs. Snowflake Schema – Which is Better?
  - 7. Fact Table and Its Types
  - 8. Granularity
  - 9. Data Virtualization
    - Think of it like a TV remote!
  - 10. Data Marts
    - Think of it like a small store inside a shopping mall!
  - 11. Data Lakes
    - Think of it like a big lake!
  - 12. Relational OLAP
    - Think of it like a Library with an Index!
  - 13. Multi dimensional OLAP
    - Think of it like a Vending Machine!
  - 14. Data Cubes
    - 1. Slice (Filtering one dimension)
    - 2. Dice (Filtering multiple dimensions)

    - 3. Drill Down (Going deeper into details)
    - 4. Roll Up (Summarizing the data)
    - 5. Pivot (Rotating the view)
    - Summary Table
- 15. Dimensional Data
- 16. Difference between table and cube data type
    - 1️⃣ Table Data Type (Relational Data Model)
        - 🔹 Key Characteristics of Tables:
    - 2️⃣ Cube Data Type (Multidimensional Model)
        - 🔹 Key Characteristics of Cubes:
        - 🧊 Example of a Sales Cube (Multidimensional Data)
    - 3️⃣ Key Differences Between Table & Cube Data
- 17. Centipede Fact table
    - 🛠️ Characteristics of a Centipede Fact Table
    - 📌 Example: Centipede Fact Table in an E-commerce Business
    - Linked Dimension Tables
    - 📌 When to Use a Centipede Fact Table?
- 18. Factless Fact Table
    - 1️⃣ Why Use a Factless Fact Table?
        - 📌 Example: Event Tracking
    - 2️⃣ When to Use a Factless Fact Table?
- 19. Three-Tier Data warehouses
    - Three-Tier Data Warehouse Architecture
- 1️⃣ Bottom Tier: Data Source Layer (Raw Data Storage)
- 2️⃣ Middle Tier: Processing Layer (ETL & Data Storage)
- 3️⃣ Top Tier: Presentation & Reporting Layer
- 🔄 Summary Table: Three-Tier Data Warehouse
- 20. Junk dimension
    - 1️⃣ Why Do We Need a Junk Dimension?
        - 🛑 Without a Junk Dimension (Messy Fact Table)
    - 2️⃣ How Junk Dimension Solves This?
        - ✅ With a Junk Dimension (Clean Fact Table)
        - Junk Dimension Table (📦 Groups Unrelated Attributes Together)

        - Fact Table (Now Cleaner)
        - 3️⃣ When to Use a Junk Dimension?
    - 21. ODS and its types
    - 22. Major Parallel computing platforms
    - 23. Quantifiable Data Indicator
        - Examples of Quantifiable Data Indicators
        - QDI vs. Qualitative Data Indicator
    - 24. Cross Tab
        - Example of Cross Tab Analysis
    - 25. Metadata in datawarehouse
    - 26. Conformed Dimensions
    - 26. Domain
    - 27. Hierarchy in Data Warehousing
    - 28. Role playing dimension
      - Example: Order Processing
      - Table Structure:
    - 29. Degenerate dimension
      - Example: Order Transactions
      - Table Structure:

1. OLAP vs. OLTP

- 🔹 **OLTP (Online Transaction Processing)** – Handles day-to-day transactions.
- 🔹 **OLAP (Online Analytical Processing)** – Used for complex queries and reporting.

Feature	OLTP (e.g., Banking App)	OLAP (e.g., Data Warehouse)
Purpose	Fast transactions	Data analysis & reporting
Data Size	Small (few MBs/GBs)	Huge (TBs/PBs)
Operations	Insert, Update, Delete	Read-heavy (Aggregation, Grouping)
Example	ATM withdrawal	Monthly sales report

📌 **Example:**

- A shopping website (**OLTP**) records purchases in a database.

- A sales manager checks **yearly sales trends** using an OLAP system.

2. Facts

A **fact** is a measurable event stored in a data warehouse.

📌 **Example:**
For an **e-commerce store**, facts could be:
✅ **Total Sales**
✅ **Number of Orders**
✅ **Revenue Generated**

Facts are usually stored in **Fact Tables**.

3. Dimensions

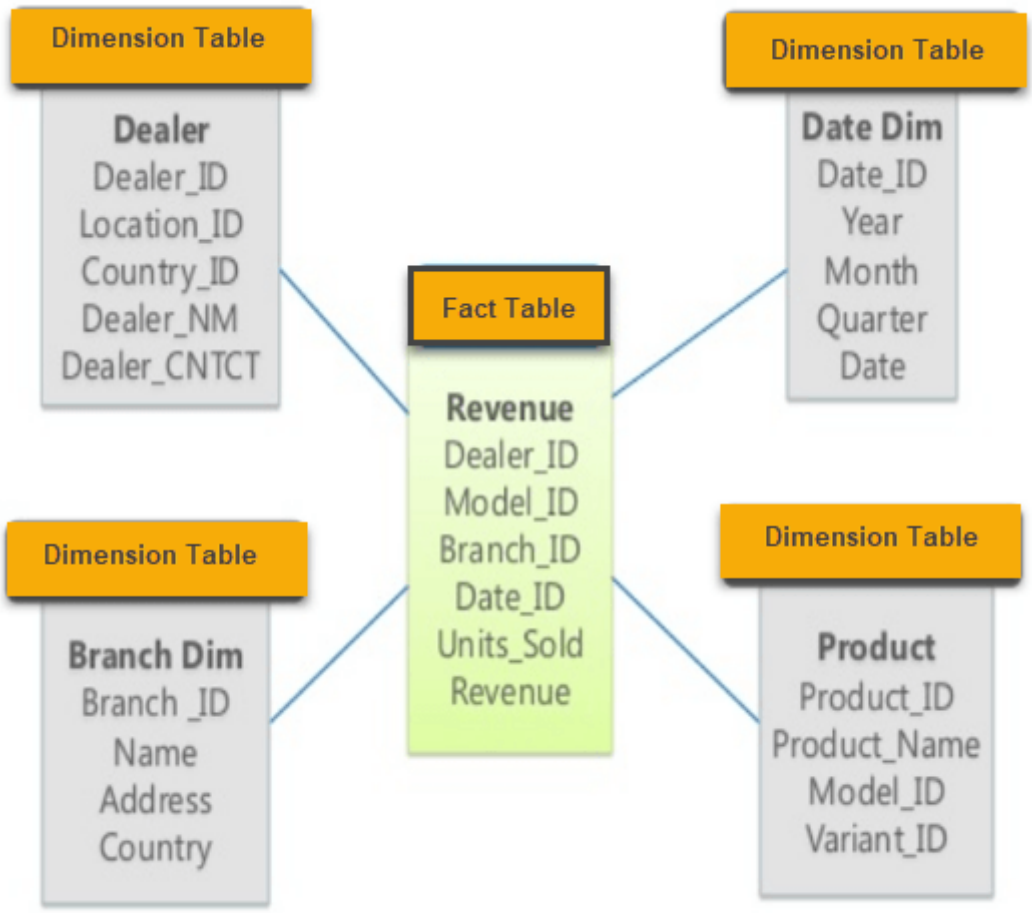
A **dimension** provides context to facts. It describes **who, what, where, and when**.

📌 **Example:**
For sales data, dimensions could be:
✅ **Customer** (Name, Age, Location)
✅ **Product** (Category, Price)
✅ **Time** (Year, Quarter, Month)

Dimensions are stored in **Dimension Tables** and linked to fact tables.

4. Star Schema

A **Star Schema** organizes data into:
✅ **A central Fact Table** (stores measurable data)
✅ **Multiple Dimension Tables** (descriptive details)



Example of Star Schema Diagram

Characteristics of a Star Schema

✔️ **Single-level dimension tables** (no normalization).
✔️ **Dimension tables connect only to the fact table** (not to each other).
✔️ **Faster query performance** due to simple joins.
✔️ **Easy to maintain and expand**.

Advantages of Star Schema

✅ **Faster Query Performance** – Simple structure enables efficient indexing and aggregation.
✅ **Easier for Reporting & Analysis** – Widely used in **BI tools** like Power BI, Tableau.
✅ **Supports OLAP Cubes** – Star schema helps in creating **multi-dimensional data cubes**.

Disadvantages of Star Schema

❌ **Data Redundancy** – Since dimensions are not normalized, some data gets repeated.
❌ **Not Ideal for Complex Relationships** – Cannot efficiently handle **many-to-many relationships**.
❌ **Limited Flexibility** – Adding new analysis fields might require schema redesign.

Conclusion

📌 **Star Schema** is the most popular **data warehouse model**, used in OLAP and reporting systems.
📌 It consists of **one fact table** and **multiple dimension tables** forming a star-like structure.
📌 Provides **fast querying**, **easy maintenance**, and **efficient OLAP cube design**.

🚀 **Ideal for**: Business Intelligence (BI), Data Warehousing, and Decision Support Systems.

5. Snowflake Schema

A **Snowflake Schema** is a more structured version of the Star Schema.

📌 **Example:**
Instead of storing a **Location** directly in the Customer table, we split it into:
✅ **Customer** → CustomerID, Name, LocationID
✅ **Location** → LocationID, City, Country

Characteristics of a Snowflake Schema

✔️ **Dimension tables are normalized** into multiple sub-tables.
✔️ **Minimizes redundancy** by reducing duplicate data storage.
✔️ **More complex queries** due to multiple joins.
✔️ **Better for hierarchical data** (e.g., Country → State → City).

Advantages of Snowflake Schema

✅ **Saves Disk Space** – Normalization reduces storage requirements.
✅ **No Data Redundancy** – Each value is stored only once.
✅ **Scalability** – Works well with large, complex datasets.

Disadvantages of Snowflake Schema

❌ **Slower Query Performance** – More joins lead to **longer execution times**.
❌ **Complex to Maintain** – Schema changes require updating multiple related tables.
❌ **Harder to Understand** – More complex relationships make querying difficult.

6. Star Schema vs. Snowflake Schema – Which is Better?

Feature	Star Schema	Snowflake Schema
Storage	Takes more space	Uses less space
Query Speed	Faster	Slower (More Joins)
Simplicity	Easy to understand	More complex

📌 **Example:**

- **Star Schema** is better for fast queries (e.g., sales reports).
- **Snowflake Schema** is better for **optimized storage**.

7. Fact Table and Its Types

A **Fact Table** stores numerical data (sales, revenue, etc.).

Types:
✅ **Transaction Fact Table** – Stores business events (e.g., a purchase)
✅ **Snapshot Fact Table** – Stores periodic data (e.g., monthly balance)
✅ **Accumulating Fact Table** – Tracks the status of a process (e.g., order lifecycle)

8. Granularity

🔹 **Granularity** refers to the **level of detail** in a data warehouse, particularly in **fact tables**.

📌 **Example:**

- **High Granularity** → Data is very detailed (e.g., per second sales).
- **Low Granularity** → Data is summarized (e.g., monthly sales).

Choosing the right granularity affects **storage size and query speed**.

**9. Data Virtualization

👉 **Access data from multiple sources without actually moving or copying it.**

Think of it like a TV remote!

Just like a TV remote lets you switch between different channels without moving to a different room, **data virtualization** allows users to access and query data from different databases without physically transferring it.

🔹 **Example:**
A company has data stored in:

- MySQL (customer data)
- PostgreSQL (sales data)
- Google Sheets (marketing data)

Instead of copying all the data into one database, **data virtualization** lets employees **query all sources at once** without moving the data.

✅ **Pros:** No duplication, real-time access
❌ **Cons:** May be slower for big queries

10. Data Marts

👉 **A small, focused version of a data warehouse for a specific department.**

Think of it like a small store inside a shopping mall!

A **data warehouse** is like a giant mall with everything. A **data mart** is like a small store inside that mall, serving a specific group of people.

🔹 **Example:**
A company has a large data warehouse, but different departments only need specific data:

- 📈 **Sales Team:** Needs sales performance data
- 🎯 **Marketing Team:** Needs customer engagement data
- 💰 **Finance Team:** Needs revenue and expense data

Instead of each team searching through the entire data warehouse, they get their own **data mart** with just the relevant information.

✅ **Pros:** Faster, easier to manage for teams
❌ **Cons:** Needs proper setup and maintenance

Type	How It's Built ⚒️	Where It Gets Data 📥	Best For ✅
1️⃣ **Dependent Data Mart** 🏗️	Built from a central data warehouse	From the **main data warehouse**	Large organizations that already have a data warehouse
2️⃣ **Independent Data Mart** 🚀	Created separately, without a data warehouse	Directly from **operational databases**	Smaller businesses or when a data warehouse doesn't exist
3️⃣ **Hybrid Data Mart** 🔀	Combination of Dependent & Independent	From both **data warehouses & external sources**	Companies that need a mix of internal & external data

11. Data Lakes

👉 **A huge storage system that keeps raw data in its original format until needed.**

Think of it like a big lake!

Just like a lake holds water from different sources (rivers, rain, streams) without filtering, a **data lake** stores all types of data—structured (tables), semi-structured (JSON, XML), and unstructured (videos, images).

🔹 **Example:**
An e-commerce company stores:

- 🛍️ Transaction data (structured)
- 📧 Customer emails (semi-structured)

- 🎥 Product demo videos (unstructured)

A **data lake** keeps all this data **as-is** until it's needed for analysis.

✅ **Pros:** Flexible, stores all data types
❌ **Cons:** Can become messy (a "data swamp") if not managed well

12. Relational OLAP

👉 **Uses traditional relational databases (SQL) to store and analyze data.**

Think of it like a Library with an Index!

A library has books (data) stored in different sections (tables), and an **index** (SQL queries) helps find the information quickly.

🔹 **How it works:**

- Stores data in relational databases (e.g., MySQL, PostgreSQL).
- Uses complex SQL queries to analyze large datasets.
- Works well with large and dynamic data

13. Multi dimensional OLAP

👉 **Uses pre-built cubes for fast analysis.**

Think of it like a Vending Machine!

Instead of searching for ingredients (raw data) and cooking, you get a ready-to-eat snack (pre-calculated data).

🔹 **How it works:**

- Stores data in a **multidimensional cube** instead of tables.
- Pre-calculates summaries for quick access.
- Best for fast reporting and interactive analysis.

✅ **Pros:** Super fast queries since data is pre-aggregated.
❌ **Cons:** Takes up more storage space and is less flexible.

14. Data Cubes

A **data cube** is a multi-dimensional structure used for analyzing data efficiently, especially in **OLAP (Online Analytical Processing)**. It allows users to explore data from different perspectives, just like how we can look at a physical cube from different angles.

Now, let's break down the **key operations** used to analyze a data cube:

1. Slice (Filtering one dimension)

👉 **Think of it as cutting a single slice from a cake.**

- It selects a single value from **one** dimension and shows the remaining data.
- Example:
  - A data cube contains **sales data** with three dimensions: **Region, Product, and Time**.
  - If we take **only the sales data for "2024"** (fixing the "Time" dimension), we get a **slice** of the cube showing only "Region vs. Product" sales.

2. Dice (Filtering multiple dimensions)

👉 **Like cutting a smaller block out of the cake.**

- It selects specific values from **two or more** dimensions.
- Example:
  - From the same cube, we want **sales data for "2024" and only for "Mobile Phones" in "Asia".**
  - This creates a **smaller sub-cube** focusing on those conditions.

3. Drill Down (Going deeper into details)

👉 **Like zooming in on a map.**

- It increases the level of detail by moving from **higher-level summaries** to **lower-level details**.
- Example:
  - We have **total sales per country** in a region.
  - If we "drill down" into India, we now see **sales per state** instead of just the country.

4. Roll Up (Summarizing the data)

👉 **Like zooming out on a map.**

- It decreases the level of detail by **aggregating** data into higher-level summaries.
- Example:
  - Instead of **sales per city**, we roll it up to show **total sales per country**.
  - This is the reverse of **Drill Down**.

5. Pivot (Rotating the view)

👉 **Like turning a Rubik's cube to see a different side.**

- It changes the way data is presented by **swapping rows and columns**.
- Example:
  - Suppose our table shows **Products in rows** and **Regions in columns**.
  - If we **pivot**, it can now show **Regions in rows** and **Products in columns**, giving a different perspective.

Summary Table

Operation	Action	Example
Slice	Filters one dimension	Show sales data only for 2024
Dice	Filters multiple dimensions	Show sales only for "Mobile Phones" in "Asia" during 2024
Drill Down	Increases detail	See sales per **state** instead of country
Roll Up	Summarizes	See sales per **region** instead of state
Pivot	Changes view	Swap "Regions" and "Products" in a table

15. Dimensional Data

Dimensional data is a way of organizing data to make it easier to analyze in a **data warehouse**. It focuses on how business users **query data** rather than how databases store it.

When data is structured for **reporting and analysis**, it is stored in two main types of tables:

1. **Fact Tables** (store business events & numbers)
2. **Dimension Tables** (store descriptive information)

16. Difference between table and cube data type

1️⃣ Table Data Type (Relational Data Model)

A **table** is a structured collection of rows and columns, like a spreadsheet. It follows the **relational database model** and is commonly used in databases like **PostgreSQL, MySQL, and SQL Server**.

🔹 Key Characteristics of Tables:

✅ **Two-dimensional** (rows & columns).
✅ Stores **transactional data** in a normalized format.
✅ Uses **SQL queries** (JOIN, SELECT, WHERE, etc.).

Tables are **good for transactional systems (OLTP)** but **slow for complex analytics**, especially when dealing with large datasets.

2️⃣ Cube Data Type (Multidimensional Model)

A **cube** is a **multidimensional data structure** used in OLAP for fast data analysis. Instead of rows & columns, data is stored in multiple **dimensions**.

🔹 Key Characteristics of Cubes:

✅ **Multi-dimensional** (more than just rows & columns).
✅ Optimized for **aggregated reporting and analytics**.
✅ Supports **OLAP operations** (Slice, Dice, Drill Down, Roll Up, Pivot).
✅ Pre-aggregated data for fast querying.

🧊 Example of a Sales Cube (Multidimensional Data)

A **Sales Cube** could have three dimensions:

- **Time (Year, Month, Day)**
- **Product (iPhone, MacBook, iPad)**
- **Region (USA, India, Europe)**

3️⃣ Key Differences Between Table & Cube Data

Feature	Table (Relational)	Cube (Multidimensional)
Structure	Rows & Columns (2D)	Multiple Dimensions (3D+)
Use Case	Storing transactional data	Fast analytical queries
Speed	Slower for complex queries	Faster due to pre-aggregated data
Operations	SQL (JOIN, WHERE, GROUP BY)	OLAP (Slice, Dice, Drill, Roll Up)
Optimization	For CRUD operations	For Business Intelligence (BI)

- Use **tables** when working with **transactional databases** (OLTP) for **storage and retrieval**.
- Use **cubes** when performing **fast analytical queries** on large datasets (OLAP).

17. Centipede Fact table

A **Centipede Fact Table** is a special type of fact table in a data warehouse that has **many foreign keys** pointing to multiple dimension tables. This structure makes it look like a centipede 🐛 because of its **long list of foreign key columns** acting like "legs."

🛠️ Characteristics of a Centipede Fact Table

1. **Many Dimension Tables** 🏗️
   - A centipede fact table has **a large number of dimension tables** linked to it.
   - Each dimension provides more details about the facts.
2. **Wide Table (Many Foreign Keys)** 📊
   - The fact table has **many foreign key columns**, making it **wide**.
   - Each row in the table represents a business event (e.g., a sale, transaction, or shipment).
3. **Optimized for Analysis, Not Performance** ⏳
   - While centipede fact tables provide **detailed** information, they can be **slower** because of many **joins** needed for querying.

📌 Example: Centipede Fact Table in an E-commerce Business

Imagine an **E-commerce Sales Fact Table** with many dimension tables:

Order ID	Date Key	Customer Key	Product Key	Region Key	Payment Key	Shipment Key	Sales Amount	
1001	20240301	501	101	301	701	801	₹5000	
1002	20240302	502	102	302	702	802	₹3000	

Linked Dimension Tables

- **Date Dimension** (Date Key → Year, Month, Day)
- **Customer Dimension** (Customer Key → Name, Age, Location)
- **Product Dimension** (Product Key → Name, Category, Brand)
- **Region Dimension** (Region Key → Country, State, City)
- **Payment Dimension** (Payment Key → Payment Type, Bank Name)
- **Shipment Dimension** (Shipment Key → Courier, Delivery Time)

Because there are **so many foreign keys**, the fact table **looks like a centipede** with many "legs" (references to dimension tables).

📌 When to Use a Centipede Fact Table?

✅ When you need **highly detailed data** for in-depth analytics.
✅ When your data warehouse supports **many dimensions** for better reporting.
❌ Avoid if performance is a concern—consider **denormalization** to reduce joins.

---

18. Factless Fact Table

A **Factless Fact Table** is a fact table that **does not have measurable numeric facts** (like sales amount, revenue, or quantity). Instead, it records **events or relationships** between dimensions without any numerical metrics.

📌 **Think of it as a fact table that captures "what happened" but without actual values.**

1️⃣ Why Use a Factless Fact Table?

🔹 To track **events or occurrences** (e.g., student attendance, product promotions).
🔹 Useful for **analyzing trends and patterns** without needing numerical data.

📌 Example: Event Tracking

- Captures **something that happened** but has no numerical measures.
- Example: **Student Attendance** in a university.

Fact Table: Attendance Fact

Date Key	Student Key	Course Key	Professor Key
20240301	101	201	301
20240302	102	202	302

What this tells us?

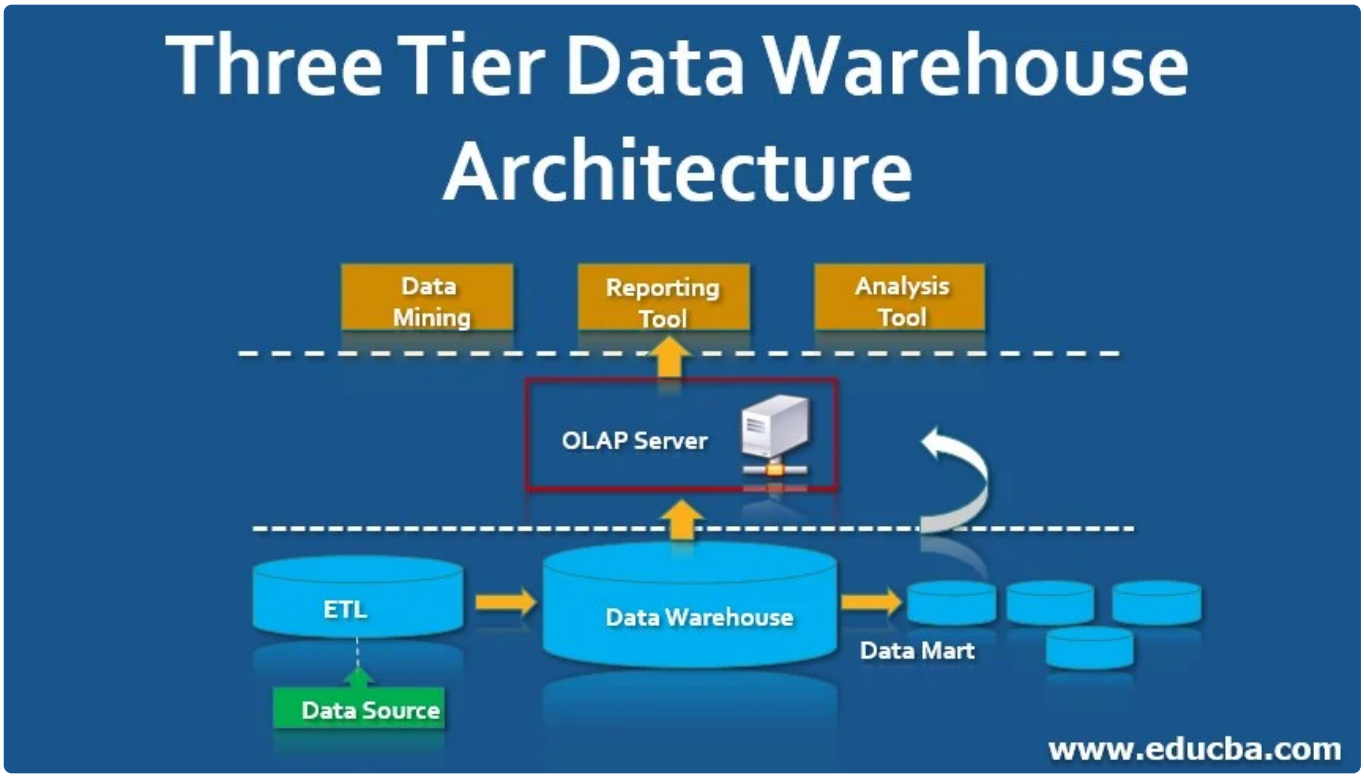
- A student (101) attended a course (201) on a specific date (20240301).
- There’s no measurable fact like "hours attended"—just the **event itself**.

2️⃣ When to Use a Factless Fact Table?

✅ When tracking **events or actions** without measurable facts.
✅ When modeling **relationships** between dimensions (many-to-many).
✅ When analyzing **patterns, trends, or coverage data**.

---

19. Three-Tier Data warehouses



Three-Tier Data Warehouse Architecture

A **Three-Tier Data Warehouse** is a structured way of organizing a data warehouse, dividing it into three layers:

1️⃣ **Bottom Tier (Data Source Layer)** – Where data is collected
2️⃣ **Middle Tier (Processing Layer)** – Where data is transformed and stored
3️⃣ **Top Tier (Presentation Layer)** – Where users access the data

📌 **Think of it like a restaurant:**

- **Kitchen (Bottom Tier)** → Raw ingredients (data) are stored.
- **Chef (Middle Tier)** → Prepares and organizes the food (transforms data).
- **Waiter & Menu (Top Tier)** → Serves the final dishes (reports & dashboards).

1️⃣ Bottom Tier: Data Source Layer (Raw Data Storage)

🔹 This is where the data warehouse gets **raw data** from multiple sources.
📌 **Example:** A retail company collects data from **sales, inventory, and customer transactions** in various databases.

2️⃣ Middle Tier: Processing Layer (ETL & Data Storage)

🔹 This layer does:

- **ETL (Extract, Transform, Load)** → Cleans, formats, and loads data into the warehouse.
- **Data Storage** → Stores processed data in **Star Schema or Snowflake Schema**.
- **OLAP Processing** → Enables multidimensional data analysis (cubes, aggregations).

📌 **Example:**

- Sales data from different stores is cleaned and converted into **fact and dimension tables**.
- The data warehouse organizes the data for fast querying.

3️⃣ Top Tier: Presentation & Reporting Layer

🔹 This is where users **query and analyze** the data.
🔹 Includes:

- **BI Tools & Dashboards** (Power BI, Tableau, Looker, etc.).
- **SQL Query Interfaces** (for data analysts).
- **Reports & Visualizations** (for business users).

📌 **Example:**

- A sales manager runs a **dashboard report** to see monthly revenue trends.
- A data analyst runs an **SQL query** to find the top-selling products.

🔄 Summary Table: Three-Tier Data Warehouse

Tier	Purpose	Example Technologies
Bottom Tier	Collect & store raw data	PostgreSQL, MySQL, CSV files
Middle Tier	Process & organize data (ETL, OLAP)	Apache Spark, Snowflake, Amazon Redshift
Top Tier	User access (reports, dashboards)	Power BI, Tableau, Looker, SQL Queries

20. Junk dimension

A **Junk Dimension** is a dimension in a data warehouse that combines **low-cardinality** attributes (flags, indicators, or miscellaneous data) into a single table.

📌 **Think of it as a "dumping ground" for small, unrelated attributes** that don't fit well in other dimensions.

1️⃣ Why Do We Need a Junk Dimension?

- 🔹 To **reduce clutter** in the fact table (avoid too many foreign keys).
- 🔹 To **improve performance** by grouping unrelated attributes together.
- 🔹 To **reduce storage space** by avoiding multiple small dimension tables.

🛑 Without a Junk Dimension (Messy Fact Table)

Order ID	Product Key	Customer Key	Discount Applied?	Promo Code Used?	Return Status	Gift Wrapped?
1001	101	501	Yes	BLACKFRIDAY	No	Yes
1002	102	502	No	-	Yes	No

❌ Problems:

- Too many small attributes in the fact table.
- Increases the number of **columns** and makes queries slower.
- Harder to manage flags and indicators.

2️⃣ How Junk Dimension Solves This?

Instead of keeping all flags and indicators in the fact table, we **combine them into one Junk Dimension** table.

✅ With a Junk Dimension (Clean Fact Table)

Junk Dimension Table (📦 Groups Unrelated Attributes Together)

Junk Dimension Key	Discount Applied?	Promo Code Used?	Return Status	Gift Wrapped?
1	Yes	BLACKFRIDAY	No	Yes
2	No	-	Yes	No

Fact Table (Now Cleaner)

Order ID	Product Key	Customer Key	Junk Dimension Key
1001	101	501	1
1002	102	502	2

✅ **Benefits:**

- Fewer columns in the fact table.
- Faster queries due to indexing on **Junk Dimension Key**.
- Easier to maintain and extend in the future.

3️⃣ When to Use a Junk Dimension?

✅ When you have **small attributes** (flags, indicators, status fields) that don't belong to any specific dimension.
✅ When you want to **optimize storage** and **improve performance**.
✅ When attributes **don't change frequently** (if they do, consider a Slowly Changing Dimension instead).

21. ODS and its types

An **Operational Data Store (ODS)** is a type of database that **stores current and integrated data** from multiple sources for **real-time or near-real-time reporting**.

📌 **Think of it as a "mini-data warehouse"** that provides fresh data for operational decision-making.

Classification of ODS

Classifications	Comment
Class 1	• Refresh cycle: Real-time • Degree of transformation: Low due to compressed timeframes
Class 2	• Refresh cycle: ½ - 1 hour store and forward mechanism • Degree of transformation: Medium
Class 3	• Refresh cycle: Daily. Traditional batch process • Degree of transformation: High
Class 4	• Refresh cycle: Ad hoc, often involving preprocessed, value-added information from a data warehouse • Degree of transformation: Highest

- **Class I** – Updates of data from operational systems to ODS are synchronous.
- **Class II** – Updates between operational environment & ODS occurs between 2-3 hour frame.
- **Class III** – synchronization of updates occurs overnight.
- **Class IV** – Updates into the ODS from the DW are unscheduled.

22. Major Parallel computing platforms

1️⃣ **Cluster Computing** – A group of connected computers that work together as a single system. They are located in the same place and connected through a high-speed network. Used for AI, simulations, and databases.
Example: Google's data centers, Hadoop clusters.

2️⃣ **Grid Computing** – A collection of computers spread across different locations, working together on a shared problem. Each computer can have different hardware and operating systems. Used in scientific research and distributed computing.
Example: CERN's computing grid, Folding@home.

3️⃣ **Massively Parallel Processing (MPP)** – A system where many processors work independently on different parts of a large problem. Each processor has its own memory and

communicates with others through a network. Used in big data analytics and data warehousing.
Example: Snowflake, Amazon Redshift, Teradata.

4️⃣ **High-Performance Computing (HPC)** – Using supercomputers or large clusters to solve highly complex problems at extreme speeds. Requires specialized processors and high-speed networking. Used in scientific simulations, weather forecasting, and AI training.
Example: Summit Supercomputer, Fugaku, Cray systems.

23. Quantifiable Data Indicator

A **Quantifiable Data Indicator (QDI)** is a measurable value that helps track performance, trends, or progress over time. These indicators are based on **numerical data** and can be analyzed to make informed decisions.

Examples of Quantifiable Data Indicators

📌 **Business:**

- Monthly sales revenue ($50,000 in March).
- Customer retention rate (85% customers returned).
- Website traffic (100,000 visitors per month).

📌 **Healthcare:**

- Patient recovery rate (92% success).
- Hospital bed occupancy rate (75% occupied).

📌 **Education:**

- Student pass percentage (90% passed).
- Average test scores (85 out of 100).

📌 **Manufacturing:**

- Defect rate in production (2% defective items).
- Number of units produced per hour (500 units/hour).

QDI vs. Qualitative Data Indicator

- **QDI (Quantifiable)** → Numerical, measurable (e.g., "500 units sold").
- **Qualitative Indicator** → Descriptive, subjective (e.g., "Customer satisfaction is high").

24. Cross Tab

A **Cross Tab (Cross Tabulation)** is a method of organizing and analyzing data by comparing two or more variables in a table format. It helps identify patterns, trends, and relationships between data points.

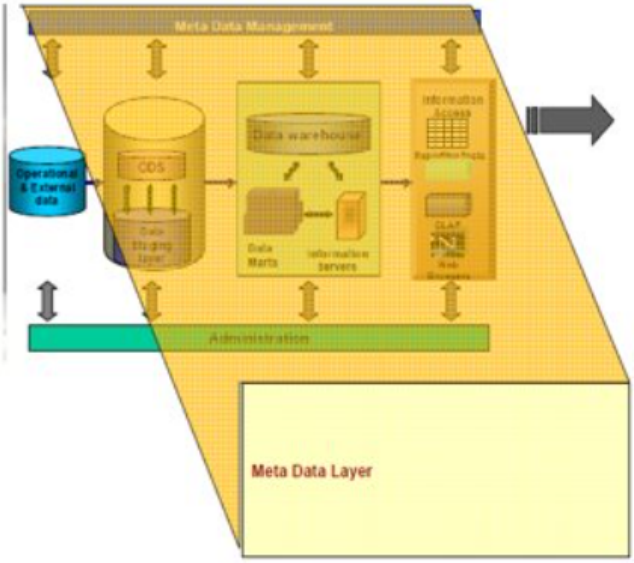
Example of Cross Tab Analysis

Scenario: A company surveys 1,000 customers about their preferred beverage (Tea or Coffee) based on age groups.

Age Group	Prefer Tea	Prefer Coffee	Total
18-25	200	300	500
26-40	150	250	400
41+	50	50	100

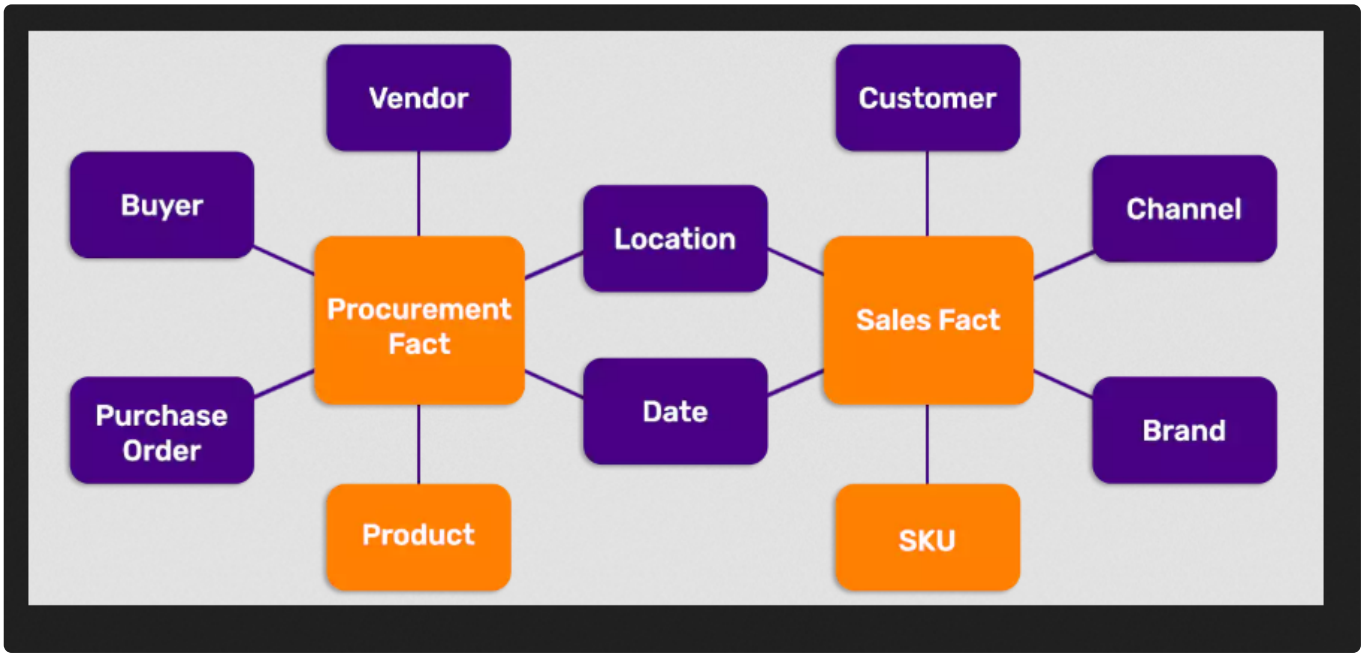
Insights from this Cross Tab:
✅ **Young people (18-25) prefer Coffee more than Tea**.
✅ **Older people (41+) have an equal preference for both beverages**.
✅ **Total survey participants = 1,000**.

25. Metadata in datawarehouse



- Metadata is data about data.
- Stored in a repository.
- Contains all corporate metadata resources: database catalogs and data dictionaries.

26. Conformed Dimensions



A **Conformed Dimension** is a **dimension table that is shared across multiple fact tables** in a data warehouse. It allows different fact tables to be analyzed using the same dimension data, ensuring **consistency** across reports and queries.

📌 **Key Characteristics:**
✔️ **Shared across multiple fact tables**
✔️ **Consistent across different business processes**
✔️ **Enables accurate cross-domain reporting**

📌 **Example:**
Imagine a retail company with **Sales** and **Returns** fact tables. Both need **Date** and **Customer** dimensions. Instead of creating separate tables, a single **conformed dimension** is used.

```
Sales_Fact   ←——— Date_Dim
Returns_Fact ←——— Date_Dim
```

This ensures that the **Date dimension** is consistent for both sales and returns analysis.

📌 **Benefits of Conformed Dimensions:**
✅ Eliminates redundancy by reusing dimensions.
✅ Ensures consistency across different fact tables.
✅ Supports enterprise-wide reporting.

26. Domain

A **Domain** in a data warehouse defines the **valid set of values** for a specific data field. It ensures **data consistency and quality** by restricting values to a predefined set.

📌 **Example:**

- **Customer_Gender** domain: {Male, Female, Other}
- **Order_Status** domain: {Pending, Shipped, Delivered, Canceled}

💡 A **domain** prevents invalid data entry, ensuring uniformity across the data warehouse.

27. Hierarchy in Data Warehousing

A **Hierarchy** defines a **parent-child relationship** within a dimension, allowing **drill-down analysis** from a broad to a detailed level.

📌 **Example of a Geographic Hierarchy:**

```
Continent → Country → State → City → Store
```

A report can be analyzed at different levels:
✔️ Global sales (Continent Level)
✔️ Country-wise sales (Country Level)
✔️ City-wise sales (City Level)

28. Role playing dimension

A **role-playing dimension** is when **one dimension table is used multiple times** in the same fact table, but for **different roles**.

Example: Order Processing

A sales order might have:
✅ **Order Date** (When the order was placed)
✅ **Shipment Date** (When the order was shipped)
✅ **Delivery Date** (When the order reached the customer)

All these dates come from the **same "Date Dimension" table**, but they represent **different roles** in the fact table.

Table Structure:

Fact Table: `Order_Fact`

Order_ID	Order_Date_ID	Shipment_Date_ID	Delivery_Date_ID	Total_Amount
101	20240201	20240203	20240205	500
102	20240202	20240204	20240206	700

Dimension Table: `Date_Dim`

Date_ID	Full_Date	Month	Year
20240201	01-Feb-2024	Feb	2024
20240203	03-Feb-2024	Feb	2024

- 🔹 **Why is this useful?**

- Instead of creating separate tables like `Order_Date_Dim` and `Shipment_Date_Dim`, we **reuse the same date table** with different column names in the fact table.
- Helps maintain **consistency** and **reduces redundancy**.

29. Degenerate dimension

A **degenerate dimension (DD)** is a **dimension that exists in the fact table but does NOT have a separate dimension table**.

Example: Order Transactions

Every order has a **unique order number**, but we **don't need a separate table for it**. Instead, it is stored **directly in the fact table**.

Table Structure:

Fact Table: `Order_Fact`

Order_Number	Date_ID	Customer_ID	Product_ID	Total_Amount
ORD1001	20240201	501	301	500
ORD1002	20240202	502	302	700

🚫 **No separate `Order_Dim` table!**

🔹 **Why is this useful?**

- Some values, like **Order Number or Invoice Number**, don't have additional attributes, so creating a separate table is unnecessary.
- Improves **query performance** because we avoid unnecessary joins.