

Lecture 6

Q1: What is a Conformed Dimension? Explain with an example.

Answer:

A **Conformed Dimension** is a dimension table that is shared across multiple fact tables in a data warehouse, maintaining the same meaning and structure in each context. It ensures consistency in reporting and analysis by providing a uniform reference for dimensional attributes. A classic example is the **Date dimension**, which can be used in sales, inventory, and shipping fact tables without ambiguity. This eliminates redundancy and supports integrated analysis across business processes.

Q2: What is a Degenerate Dimension, and how is it represented in a fact table?

Answer:

A **Degenerate Dimension** is a dimension key that does not have a corresponding dimension table. It is stored directly in the fact table because its attributes are already captured in other analytical dimensions or are not needed for separate analysis. For example, in an order fact table, **OrderID** may appear as a degenerate dimension if there is no separate Order dimension table. It helps in grouping and filtering transactions without complicating the dimensional model.

Q3: What is a Junk Dimension, and why is it used?

Answer:

A **Junk Dimension** is a single dimension table that combines multiple low-cardinality attributes (such as flags, codes, or statuses) that are unrelated but frequently used together. It is used to reduce clutter in the fact table and minimize the number of dimension tables. By consolidating these attributes into one table and using a surrogate key, it simplifies ETL processes, saves storage, and improves query performance by reducing joins.

Q4: List three benefits of using a Junk Dimension.

Answer:

1. **Simplifies ETL:** Combines multiple low-cardinality attributes into one table, making data transformation cleaner.

2. **Reduces Storage:** Replaces several small dimension tables with one, lowering data warehouse storage needs.
 3. **Improves Query Performance:** Fewer joins are required when querying, leading to faster report generation.
-

Q5: Explain the concept of a Role-Playing Dimension with an example.

Answer:

A **Role-Playing Dimension** is a single dimension table that is used multiple times in the same fact table, each time representing a different business role. For example, a **Date dimension** can play the roles of **Order Date**, **Ship Date**, and **Delivery Date** in a sales fact table. Instead of creating three separate date tables, the same table is linked multiple times via different foreign keys, allowing flexible analysis from different temporal perspectives without data duplication.

Q6: How would you design a Junk Dimension for a retail sales system with the following attributes?

- Payment Method (Credit Card, Cash, Check)
- Promotion Type (Discount, BOGO, None)
- Delivery Type (In-Store Pickup, Home Delivery)
- Customer Type (New, Returning)

Answer:

You would create a **Junk Dimension table** with a surrogate key (JunkKey) and columns for each attribute. Each unique combination of attribute values gets a row. For example:

JunkKey	PaymentMethod	PromotionType	DeliveryType	CustomerType
---------	---------------	---------------	--------------	--------------

1	Credit Card	Discount	In-Store	New
2	Cash	BOGO	Home Delivery	Returning

In the fact table, you would replace the four attribute columns with a single **JunkKey** foreign key.

Q7: What are the alternatives to using a Junk Dimension, and what are their drawbacks?

Answer:

Alternatives include:

1. **Putting them in an existing dimension:** May cause grain mismatch and Cartesian products.
 2. **Leaving them in the fact table:** Increases row size, clutters the model, and hurts performance.
 3. **Creating separate dimensions:** Leads to too many tables, complexity, and maintenance issues.
 4. **Eliminating them:** Not possible if business requires them for analysis.
-

Q8: How does a Junk Dimension support business intelligence applications like dashboards?

Answer:

A Junk Dimension centralizes codes and categories in a table, which can be used to populate picklists, dropdowns, or filters in dashboards and reports. This replaces hard-coded values in applications, making it easier to maintain, update, and share across teams. For example, promotion status codes stored in a Junk Dimension can be dynamically loaded into a BI tool without modifying the application code.

Q9: Describe a practical scenario where a Role-Playing Dimension would be useful outside of date analysis.

Answer:

In a **hospital system**, a **Staff dimension** could play multiple roles in a patient treatment fact table:

- Admitting Doctor
- Attending Physician
- Discharging Nurse

The same Staff table is linked multiple times via different foreign keys, allowing analysis of patient care from different clinical roles without duplicating staff data.

Q10: How does a Junk Dimension improve data governance and consistency?

Answer:

By storing low-cardinality attributes in a central table, a Junk Dimension ensures that codes, flags, and categories are consistent across the data warehouse. It prevents the same attribute from being defined differently in separate tables or applications, supports easier auditing, and simplifies change management—such as adding a new status code—with impacting fact tables or ETL logic significantly.

Lecture 7

Q1: What is an Outrigger Dimension? Provide an example from the lecture.**Answer:**

An **Outrigger Dimension** is a secondary dimension table that extends a primary dimension table by storing additional, less frequently used attributes. It is connected to the main dimension table in a star or snowflake schema.

Example: In a retail data warehouse, the **Products** dimension may have an outrigger dimension for **Product Specifications**, containing details like screen size, processor, and storage, linked via the product_key.

Q2: Why are Outrigger Dimensions used in dimensional modeling? List three reasons.**Answer:**

1. **Detail Separation:** Keeps primary dimensions clean and focused on core attributes.
 2. **Scalability:** Allows addition of new attributes without altering the primary dimension structure.
 3. **Query Performance:** Reduces unnecessary joins for queries that don't require detailed attributes.
-

Q3: What is a Snowflake Dimension? How does it differ from a Star Schema?**Answer:**

A **Snowflake Dimension** is a dimension table that is normalized into multiple related tables, forming a hierarchy. For example, a Product dimension may be split into separate tables for Category, Subcategory, and Manufacturer.

Difference from Star Schema: In a star schema, dimensions are denormalized into single tables. In a snowflake schema, dimensions are normalized, reducing redundancy but increasing join complexity.

Q4: What is a Slowly Changing Dimension (SCD)? Why is it important in data warehousing?

Answer:

A **Slowly Changing Dimension (SCD)** is a dimension where attributes change over time infrequently (e.g., customer address, product price).

Importance: It allows historical tracking of changes, supporting time-based analysis and reporting without losing past context, which is crucial for accurate trend analysis and compliance.

Q5: Describe the differences between SCD Type 1, Type 2, and Type 3.

Answer:

- **Type 1 (Overwrite):** Updates existing row with new values; no history is kept.
 - **Type 2 (Add Row):** Creates a new row for each change, preserving history with effective dates.
 - **Type 3 (Add Column):** Adds a new column to store previous value; limited to tracking one change.
-

Q6: What are the disadvantages of using SCD Type 1?

Answer:

1. **Loss of History:** Previous attribute values are overwritten and lost.
 2. **Aggregation Issues:** Summary tables and OLAP cubes may need to be rebuilt if underlying data changes.
 3. **No Audit Trail:** Cannot track when or how often changes occurred.
-

Q7: How does SCD Type 2 preserve historical data? Provide an example table.

Answer:

SCD Type 2 adds a new row each time an attribute changes, along with effective/termination dates and a current flag.

Example:

cust_id customer_name location effective_date termination_date is_current

2001	Ahmed	Cairo	2023-10-20	2023-10-30	False
2001	Ahmed	London	2023-10-30	2023-11-20	False
2001	Ahmed	Paris	2023-11-20	Null	True

Q8: What is SCD Type 4, and when would you use it?

Answer:

SCD Type 4 splits dimension data into two tables: one for **current** data and one for **historical** data.

Use Case: When you need to optimize query performance for current values while still maintaining a full history for reporting and compliance.

Q9: What is a Fast Changing Dimension? How is it handled?

Answer:

A **Fast Changing Dimension** is a dimension where one or more attributes change very frequently (e.g., product price multiple times a day).

Solution: Use **Mini-Dimensions** (also called Fast Changing Dimensions), where rapidly changing attributes are separated into their own dimension table linked via a bridge table with effective dates.

Q10: Describe how a Mini-Dimension (Fast Changing Dimension) is structured using the example from the lecture.

Answer:

The main dimension (e.g., Product) retains stable attributes, while fast-changing attributes (e.g., Price) are moved to a separate **Mini-Dimension**. A bridge table links them with start and end dates.

Example:

Product Dimension:

product_id product_name

1000 HP Laptop

Price Mini-Dimension:

price_key price

1 30000

2 35000

3 40000

Bridge Table:

product_id price_key start_date end_date

1000 1 2023-10-20 2023-10-30

1000 2 2023-10-30 2023-11-20

1000 3 2023-11-20 Null

Q11: How does an Outrigger Dimension improve maintainability?

Answer:

By localizing detailed attributes to separate tables, changes (like adding a new product specification) can be made in the outrigger dimension without affecting the primary dimension or existing queries. This modular structure simplifies updates and reduces risk of errors.

Q12: In what scenario would you choose SCD Type 3 over Type 2?

Answer:

SCD Type 3 is suitable when you only need to track the **previous value** of an attribute (not full history) and when changes are rare. For example, tracking a customer's previous address after a single move. It is simpler to implement than Type 2 but offers limited historical depth.

Lecture 8

Q1: What is the ETL process and what is its primary purpose in a data warehouse?

Answer:

The **ETL (Extract, Transform, Load)** process is a series of steps used to collect data from various sources, transform it into a consistent and usable format, and load it into a data warehouse. Its primary purpose is to reshape raw data from operational systems into structured, high-quality information that supports strategic decision-making, reporting, and analytics in the data warehouse.

Q2: List and briefly explain the three main phases of the ETL process.

Answer:

1. **Extract:** Data is pulled from various source systems such as databases, spreadsheets, or external applications. This can be a one-time full extraction or an ongoing incremental extraction.
 2. **Transform:** Raw data is cleaned, standardized, enriched, and restructured to meet data warehouse requirements. This includes tasks like filtering, deduplication, merging, and conversion.
 3. **Load:** The transformed data is loaded into the data warehouse tables. This can be done via initial load, incremental load, or full refresh.
-

Q3: What are some key considerations when extracting data for a data warehouse?

Answer:

Key considerations include:

- **Source Identification:** Determining which applications and data structures to extract from.
 - **Method of Extraction:** Choosing between manual or automated/tool-based extraction.
 - **Extraction Frequency:** Deciding how often to extract (daily, weekly, etc.).
 - **Job Sequencing:** Ensuring dependent extraction jobs run in the correct order.
 - **Exception Handling:** Defining how to manage records that cannot be extracted.
-

Q4: Why is data transformation important in ETL? Provide three examples of transformation tasks.

Answer:

Data transformation is crucial because raw extracted data is often inconsistent, incomplete, or unusable. Transformation ensures data quality, consistency, and readiness for analysis.

Examples:

1. **Cleansing:** Filling in missing values or correcting errors.
 2. **Key Restructuring:** Converting source system keys into surrogate keys for the data warehouse.
 3. **Summarization:** Aggregating detailed data to a higher level for performance and usability.
-

Q5: What is the difference between an initial load, incremental load, and full refresh?

Answer:

- **Initial Load:** The first population of all tables in the data warehouse with historical data.
 - **Incremental Load:** Periodic updates that apply only new or changed data since the last load.
 - **Full Refresh:** Complete replacement of data in one or more tables, typically used when data corruption occurs or when a full reload is required.
-

Q6: Describe the four methods of applying data during the load phase: Load, Append, Destructive Merge, and Constructive Merge.

Answer:

1. **Load:** Overwrites all existing data in the target table with new data.
 2. **Append:** Adds new data to the target table without deleting existing records; duplicates may be allowed or rejected.
 3. **Destructive Merge:** Updates existing records if keys match; adds new records if no match is found.
 4. **Constructive Merge:** Preserves existing records and adds new versions when keys match, marking the new record as superseding the old one (useful for historical tracking).
-

Q7: What is data staging and why is it used in ETL?

Answer:

Data staging is an intermediate storage area where extracted data is temporarily held before transformation and loading. It is used to:

- Isolate the transformation process from source systems and the data warehouse.
 - Allow for complex transformations without impacting performance of source or target systems.
 - Provide a checkpoint for data validation and error handling before final loading.
-

Q8: How does data quality impact the data warehouse?

Answer:

Poor data quality can lead to incorrect insights, flawed strategic decisions, and loss of trust in the data warehouse. High-quality data ensures accuracy, consistency, and reliability in reporting and analytics, which is essential for effective business intelligence.

Q9: What are some common data transformation types mentioned in the lecture?

Answer:

Common transformation types include:

- Format revisions
 - Splitting or merging fields
 - Character set conversion
 - Unit of measurement conversion
 - Date/time conversion
 - Summarization
 - Key restructuring
 - Deduplication
 - Calculated and derived values
-

Q10: What role do ETL tools play, and can you name a few examples?

Answer:

ETL tools automate and streamline the extraction, transformation, and loading processes, reducing manual effort, improving accuracy, and enabling scheduling and monitoring.

Examples: Informatica, Talend, Apache NiFi, and Microsoft SSIS.

Q11: Explain the concept of “key restructuring” in data transformation with an example.

Answer:

Key restructuring involves converting source system keys into surrogate keys that are more suitable for the data warehouse environment.

Example: A production system product code like 12 W1 M53 1234 69 (containing embedded codes for country, warehouse, territory, etc.) may be transformed into a simple numeric surrogate key like 12345678 in the data warehouse for consistency and performance.

Q12: Why is exception handling important during data extraction?

Answer:

Exception handling ensures that extraction processes can gracefully manage errors such as missing files, corrupted data, or connectivity issues. Proper handling prevents entire ETL jobs from failing and allows for logging, alerts, and corrective actions to maintain data pipeline reliability.

Q13: In what scenarios would you use a constructive merge instead of a destructive merge?

Answer:

Constructive merge is used when historical tracking is required. For example, in a customer dimension where address changes need to be preserved over time, a constructive merge retains the old record and adds a new version, whereas a destructive merge would overwrite the old record, losing history.

Lecture 9 & 10

Q1: What is OLAP and how does it differ from OLTP?

Answer:

OLAP (Online Analytical Processing) is a technology that enables users to analyze multidimensional data interactively from various perspectives. It supports complex queries for

business intelligence, reporting, and data mining.

Difference from OLTP (Online Transaction Processing): OLAP is used for historical analysis and decision-making, while OLTP is used for day-to-day transaction processing. OLAP is read-intensive, uses aggregated data, and follows a star/snowflake schema; OLTP is write-intensive, uses detailed data, and follows a normalized relational schema.

Q2: Describe the main characteristics of OLAP applications.

Answer:

OLAP applications are characterized by:

- **Multidimensional analysis:** Viewing data across multiple dimensions (e.g., time, product, region).
 - **Interactive analysis:** Users can drill down, roll up, slice, dice, and pivot data.
 - **Ad-hoc querying:** Supports unplanned, user-driven exploration.
 - **Aggregated data:** Pre-calculated summaries for fast query performance.
 - **Iterative analysis:** Answers lead to further questions in a cyclical process.
-

Q3: What is an OLAP cube? How does it organize data?

Answer:

An **OLAP cube** is a multidimensional data structure that organizes data into dimensions and measures. It allows users to analyze data across multiple axes (e.g., product, time, location). Each cell in the cube contains aggregated values (measures), and dimensions are organized hierarchically (e.g., year → quarter → month).

Q4: Explain the four basic OLAP operations: Roll-up, Drill-down, Slice and Dice, and Pivot.

Answer:

1. **Roll-up (Consolidation):** Aggregates data to a higher level (e.g., monthly to quarterly sales).
2. **Drill-down:** Breaks data into more detailed levels (e.g., quarterly to monthly sales).
3. **Slice and Dice:**

- **Slice:** Filters data along one dimension (e.g., sales in Q1 only).
 - **Dice:** Filters data across multiple dimensions (e.g., sales in Q1 for Product A and Region B).
4. **Pivot (Rotate):** Changes the orientation of the data view (e.g., swapping rows and columns).
-

Q5: What are the three main types of OLAP architectures? Briefly describe each.

Answer:

1. **MOLAP (Multidimensional OLAP):** Stores data in a pre-computed multidimensional cube for fast query performance. Best for predefined queries but less scalable.
 2. **ROLAP (Relational OLAP):** Uses relational databases to store data. Supports large datasets and ad-hoc queries but may have slower performance.
 3. **HOLAP (Hybrid OLAP):** Combines MOLAP and ROLAP; stores aggregated data in cubes and detailed data in relational tables. Balances speed and scalability.
-

Q6: What are the advantages and disadvantages of MOLAP?

Answer:

Advantages:

- Fast query performance due to pre-aggregation.
- Easy to use for non-technical users.
- Optimized for multidimensional analysis.

Disadvantages:

- Limited scalability with large datasets.
 - Data redundancy and high storage requirements.
 - Difficult to update dimensions without rebuilding cubes.
-

Q7: How does ROLAP differ from MOLAP in terms of data storage and query processing?

Answer:

ROLAP stores data in relational tables and uses SQL to compute aggregates on the fly. It is more flexible and scalable but slower for complex queries.

MOLAP stores pre-aggregated data in multidimensional arrays, allowing direct access to aggregated values, resulting in faster queries but less flexibility.

Q8: What is a “data cube” in OLAP? Provide a simple example.

Answer:

A **data cube** is a multidimensional representation of data. For example, a sales cube might have:

- **Dimensions:** Product, Time, Store
- **Measures:** Sales Amount
- **Hierarchy:** Time → Year → Quarter → Month

A cell might represent: Sales of *Laptop* in *Q1* at *Store A* = \$10,000.

Q9: Why is OLAP considered “interactive” and “iterative”?

Answer:

OLAP is **interactive** because users can dynamically change queries, dimensions, and levels of detail without pre-programmed reports. It is **iterative** because the analysis process is cyclical: each answer leads to new questions, prompting further drill-down, roll-up, or pivoting.

Q10: What is the role of aggregation in OLAP? Why is it important?

Answer:

Aggregation in OLAP involves pre-calculating summary data (e.g., totals, averages) across dimensions. It is important because:

- Speeds up query response times.
 - Supports trend analysis and high-level reporting.
 - Enables efficient roll-up and drill-down operations.
-

Q11: Describe a scenario where HOLAP would be preferred over MOLAP or ROLAP.

Answer:

HOLAP is preferred when an organization needs both fast query performance on aggregated data (like MOLAP) and the ability to query detailed transactional data (like ROLAP). For example, a retail chain might use HOLAP to:

- Quickly view regional sales summaries (via cubes).
 - Drill down to individual store transactions (via relational tables).
-

Q12: How does OLAP support decision-making in business?

Answer:

OLAP enables decision-makers to:

- Analyze trends and patterns over time.
 - Compare performance across products, regions, or time periods.
 - Perform “what-if” analysis and forecasting.
 - Identify root causes of issues through drill-down.
 - Generate ad-hoc reports without IT assistance.
-

Q13: What are some common OLAP tools? Name a few examples.

Answer:

Common OLAP tools include:

- **Oracle Essbase (MOLAP)**
 - **Microsoft SQL Server Analysis Services (SSAS)** (supports MOLAP, ROLAP, HOLAP)
 - **IBM Cognos**
 - **SAP BusinessObjects**
 - **Tableau** (for visualization with OLAP backends)
-

Q14: What is meant by “multidimensional analysis” in OLAP?

Answer:

Multidimensional analysis refers to the ability to view and analyze data across multiple

dimensions simultaneously (e.g., sales by product, time, and location). It allows users to explore relationships and patterns that are not visible in flat, two-dimensional tables.

Q15: How does OLAP handle sparse data in cubes?

Answer:

In OLAP cubes, **sparse data** refers to empty cells where no data exists for certain dimension combinations (e.g., a product not sold in a region). MOLAP systems use compression techniques to store only non-empty cells, optimizing storage and performance.