Labs 5: Dimension, measures, star schema, snow flake, shared connection managers and package tasks

1. What are the different database design techniques for OLAP?

**ROLAP (Relational OLAP):**

ROLAP operates directly on relational tables without reorganizing data into a cube.

SQL is commonly used for multidimensional queries, reporting, and analysis1.

**MOLAP (Multidimensional OLAP):**

MOLAP stores data in a multidimensional cube format.

Efficient for complex calculations and aggregations.

Examples include Microsoft Analysis Services (SSAS) and IBM Cognos TM1.

**HOLAP (Hybrid OLAP):**

Combines features of both ROLAP and MOLAP.

Stores summary data in a cube while detailed data remains in relational tables.

Provides a balance between performance and flexibility.

**DOLAP (Desktop OLAP):**

Designed for individual users or small workgroups.

Runs on desktop machines and doesn’t require a separate server.

Limited scalability but suitable for ad hoc analysis.

**WOLAP (Web OLAP):**

Web-based OLAP accessible via browsers.

Enables collaboration and sharing of insights across teams.

Examples include web-based dashboards and reporting tools.

**IOLAP (Intelligent OLAP):**

Integrates OLAP with artificial intelligence (AI) techniques.

Enhances decision-making by providing intelligent recommendations.

1. Why is normalisation design not good for OLAP database?

The debate over whether to normalize or denormalize databases for OLAP (Online Analytical Processing) workloads has been ongoing for years. Let’s explore the reasons behind this discussion:

OLTP (Online Transaction Processing) vs. OLAP:

OLTP databases are designed for transactional systems, where data integrity and operational efficiency are critical. In OLTP systems, normalization is highly recommended to eliminate redundancy and maintain data consistency1.

OLAP databases, on the other hand, serve analytical purposes. They focus on flexible and rapid data analysis. Here, the trade-off between normalization and denormalization becomes more nuanced.

**Normalization in OLAP**:

Advantages:

Data Integrity: Normalization ensures data consistency and reduces anomalies.

Storage Efficiency: Normalized tables occupy less space due to reduced redundancy.

Easier Maintenance: Updates and modifications are straightforward.

Challenges:

Complex Queries: Highly normalized schemas can lead to complex joins and queries, impacting performance.

Aggregation Overhead: Aggregating data across multiple normalized tables can be resource-intensive.

Read Performance: Normalized structures may not be optimal for read-heavy OLAP workloads.

**Denormalization in OLAP:**

Advantages:

Query Performance: Denormalization simplifies queries by reducing joins and improving read performance.

Aggregation Efficiency: Pre-aggregated data in denormalized tables speeds up complex calculations.

User Experience: Faster query response times enhance user satisfaction.

Trade-offs:

Data Redundancy: Denormalization introduces redundancy, which can lead to data inconsistencies if not managed carefully.

Maintenance Complexity: Updates become more complex due to redundant data.

Storage Overhead: Denormalized tables consume more storage space.

Striking the Balance:

The ideal strategy depends on the context:

Highly Transactional OLTP Systems: Prioritize high normalization for data integrity and operational efficiency.

Analytical OLAP Systems: Lean towards denormalization for flexible analysis, with some level of normalization for efficient storage2.

Hybrid Approaches (HOLAP): Combine elements of both normalization and denormalization to achieve a balance.

In summary, while normalization is essential for OLTP systems, OLAP databases often benefit from a judicious mix of both approaches. The choice should align with your specific use case, performance requirements, and data modeling goals.

1. What are dimensions and measures?

Dimension: Qualitative Information: Dimension tables contain qualitative or categorical data. Examples include names, dates, or geographical data.

Quantitative Information: Measure tables provide quantitative or numerical data. They contain numeric values that you can measure.

* Population: Dimension, because it is used to categorise a group of persons and cannot be measured.
* Currency Code: It is a dimension because it is to categorise a specific group of money currency.
* Age (Hint all numbers are not measures): In this context, Age is a dimension because Age is used to categorise a specific group of age (ex:18-30). In case the user would like to know the average of people’s age within a company then Age is measure.
* Employee Name: It is a dimension because it is qualitative, and describes a specific individual. It can not be measured.

1. What is the difference between star schema and snow Flake?

Two common data warehouse modeling techniques:

**Star Schema:**

Definition: Star schema is a type of multidimensional model used for data warehouses.

**Structure:**

Contains a central fact table representing the business transactions or events (e.g., sales, orders).

Dimension tables surround the fact table, providing context and descriptive attributes (e.g., time, product, location).

**Join Complexity:**

In star schema, fewer foreign-key joins are used.

Forms a star shape with the fact table at the center and dimension tables radiating outwards.

**Design Approach:**

Top-down model.

Space Usage:

Consumes more space due to denormalization.

Query Execution Time:

Queries execute faster in star schema.

**Normalization:**

Not used in star schema.

**Design Simplicity:**

Star schema has a simple design.

**Query Complexity:**

Query complexity is low.

**Understanding:**

Easy to understand.

**Foreign Keys:**

Has fewer foreign keys.

**Data Redundancy:**

High data redundancy.

**Snowflake Schema:**

Definition: Snowflake schema is another type of multidimensional model used for data warehouses.

**Structure:**

Contains the same central fact table and dimension tables as star schema.

Additionally, it includes sub-dimension tables (normalized tables) that further break down attributes (e.g., subcategories within a product category).

**Join Complexity:**

In snowflake schema, more joins are involved due to normalization.

Forms a snowflake shape with fact tables, dimension tables, and sub-dimension tables.

**Design Approach:**

Bottom-up model.

**Space Usage:**

Uses less space due to normalization.

**Query Execution Time:**

Takes more time than star schema for query execution.

**Normalization:**

Both normalization and denormalization are used.

**Design Complexity:**

Snowflake schema design can be complex.

**Query Complexity:**

Query complexity is higher than star schema.

**Understanding:**

Requires more effort to understand.

**Foreign Keys:**

Has more foreign keys.

**Data Redundancy:**

Low data redundancy due to normalization12.

In summary, choose the schema that best aligns with your specific data analysis needs, performance requirements, and understanding of the data

1. How to share connection manager across different tasks?

We share connection manager across different tasks by creating a connection manager on the solution explorer then we click on the connection manager.

1. How to disable and enable Tasks on a package ?

In order to disable or enable a task, we should right click the task and select disable or enable options (on the displayed menu).

1. How to execute a package ?

To execute a package it is necessary to double left click the package that we want to execute, and then click on the run button.

In case we have more than one package, we want to run a specific package as a start up, we should right click the package that we would like to run first, then select the menu RUN AS START UP OBJECT.

Labs 6: SCD, Type 0, Type 1, OLEDB command and Unicode conversions

In summary:

* Fixed attributes remain constant.
* Changing attributes are updated over time.
* Historical attributes save changed values in new records while marking previous records as outdated.

Unicode and Non-unicode

Certainly! Let’s explore the differences between Unicode and non-Unicode characters:

**Unicode:**

Definition: Unicode is a universal character encoding standard that represents characters from various languages and scripts.

Character Set: Unicode includes roughly 100,000 characters, covering a wide range of languages, symbols, and special characters.

Storage Size: Unicode uses multiple bytes (typically 2 or 4 bytes) to represent each character.

Variants: It has three common variants: UTF-8, UTF-16, and UTF-32.

Usage:

Widely used in web development, databases, and international applications.

Default encoding for many programming languages.

Supports emojis and diverse character sets.

**Non-Unicode:**

Definition: Non-Unicode (also known as ANSI or legacy encoding) is an older character encoding system.

Character Set: It supports a limited set of characters (usually 256) based on the specific encoding (e.g., ISO-8859-1, Windows-1252).

Storage Size: Non-Unicode uses single bytes (8 bits) per character.

Limitations: Due to its limited character set, it cannot represent all languages and symbols.

Usage:

Legacy systems that predate Unicode adoption.

Local applications with specific language requirements.

Slowly Changing Dimension (SCD) task is used when we want to insert or update a database (To avoid errors that don’t allow insertion of records that contains the same Business Key).

To use SCD for auditoria purposes, it is important to remove the primary key on the destination table, to allow the insertion of two records with the same business key to audit changes between them. Setting type 2 (Historical attribute) is crucial to ensure auditing. Additionally, it is imperative to create a Boolean column within the destination table that would be used to show the current and expired records.

In Microsoft Visual Studio 2022, when using Slowly Changing Dimension Wizard, in the historical Attribute Options, the “variable to set date values” option does not act as users would expect due to the fact it does not give the possibility to use the drop-down menu. The alternative option is to navigate with the mouse’s pad to solve the issue. Additionally, to connect to the local SQL server 2022 database, it is important to use Microsoft OLE DB Provider for SQL Server. Version: Microsoft Visual Studio Community 2022 (64-bit) – Current Version 17.9.7.

1. What is SCD and why is the name slowly changing dimensions?

Slowly Changing Dimensions (SCD) refer to how data in a data warehouse changes over time. These dimensions have the same natural key (such as a product ID or customer ID), but other data columns associated with them may or may not change over time. The term “slowly changing” emphasizes that these changes occur gradually rather than frequently.

Here are some key points about SCDs:

Purpose of SCDs:

SCDs are essential for tracking how records evolve over time.

They allow us to maintain historical context while accommodating changes.

By understanding SCDs, businesses can analyze trends, monitor performance, and make informed decisions.

Types of Slowly Changing Dimensions:

There are different types of SCDs based on how they handle changes:

Type 0: These dimensions never change. Examples include mapping tables (e.g., states, zip codes) and static data like social security numbers.

Type 1: Data is overwritten by new data without keeping a historical record. No tracking of changes over time occurs. For instance, customer addresses often fall into this category.

Type 2: Historical records are preserved by creating new rows for each change. This type allows tracking of changes over time. For example, product features or customer attributes.

Type 3: Only limited history is maintained by adding new columns to represent changes. It’s a compromise between Type 1 and Type 2.

Avoiding Costly Mistakes:

Choosing the wrong SCD type can impact business metrics.

For instance, if historical attributes are not tracked, gaps in reporting may occur.

Understanding SCDs helps prevent such mistakes.

In summary, SCDs play a crucial role in data analytics by allowing us to observe how data evolves over time, ensuring accurate reporting and informed decision-making.

1. What is the need of business key in SCD?

The business key (also known as the natural key) plays a crucial role in Slowly Changing Dimensions (SCD). Let’s explore why it is needed:

Uniqueness and Identification:

The business key uniquely identifies each dimension record.

It ensures that each entity (such as a customer, product, or location) has a distinct representation in the data warehouse.

Without a business key, it would be challenging to differentiate between similar records.

Tracking Changes Over Time:

SCDs capture how data changes over time.

By using the business key, we can track historical changes for a specific entity.

For example, if a customer’s address changes, the business key (such as the customer ID) allows us to maintain a history of both old and new addresses.

Joining and Integration:

When integrating data from multiple sources, the business key serves as a common link.

It enables joins between fact tables and dimension tables.

For example, in a sales transaction, the business key (e.g., product ID) connects the sales fact table with the product dimension.

Efficient Lookups and Queries:

Using the business key for lookups (e.g., finding a customer’s details) is more efficient than searching by other attributes.

Indexes are often created on business keys to optimize query performance.

In summary, the business key ensures uniqueness, facilitates historical tracking, supports data integration, and improves query efficiency in SCDs.

1. Explain the difference between fixed, changing and historical attributes?

Certainly! Let’s delve into the differences between fixed attributes, changing attributes, and historical attributes in the context of Slowly Changing Dimensions (SCD):

Fixed Attributes:

Definition: Fixed attributes represent characteristics of a dimension that remain constant over time.

Purpose: These attributes provide stable information about a dimension member.

Examples:

In a product dimension, the SKU (Stock Keeping Unit) is a fixed attribute. Once assigned, the SKU rarely changes.

Fixed attributes are typically used for descriptive or categorization purposes.

Changing Attributes:

Definition: Changing attributes are those that can be updated or modified over time.

Purpose: These attributes capture dynamic information about a dimension member.

Examples:

In the same product dimension:

The price of a product is a changing attribute. Prices can fluctuate due to promotions, inflation, or other factors.

Changing attributes are often used for performance metrics or operational data.

Historical Attributes:

Definition: Historical attributes represent changes in a dimension member over time while preserving historical context.

Purpose: These attributes allow us to track changes and maintain a history of values.

Examples:

Consider a customer dimension:

The address of a customer is a historical attribute.

When a customer moves to a new address, a new record is created with the updated address, while the previous address remains in the system (marked as outdated).

Historical attributes are crucial for trend analysis and accurate reporting.

In summary:

Fixed attributes remain constant.

Changing attributes are updated over time.

Historical attributes save changed values in new records while marking previous records as outdated

1. What is the difference between Unicode and Non-unicode types ?

Unicode provides a broader character repertoire and better international support, while non-Unicode systems are more limited in scope.

1. What is the difference between type1 and type 0 SCD’s?

Type 0 SCDs retain the original data without any updates.

Type 1 SCDs overwrite existing data with new values, without tracking historical changes.

1. What is the difference between DT\_STR and DT\_WSTR?

DT\_STR:

Definition: The DT\_STR data type represents a null-terminated ANSI character string.

Character Set: It uses the single-byte character encoding (such as ASCII or ISO-8859-1).

Storage Size: The length is variable, and the maximum length is typically 8000 bytes.

Usage:

Used for non-Unicode (ASCII) character data.

Suitable for legacy systems or when dealing with specific code pages.

Example: Storing names, addresses, or other text data.

DT\_WSTR:

Definition: The DT\_WSTR data type represents a null-terminated Unicode character string.

Character Set: It uses the double-byte character encoding (UTF-16).

Storage Size: The length is variable, and the maximum length is also typically 8000 characters.

Usage:

Used for Unicode character data (supporting multiple languages and special characters).

Suitable for modern applications, internationalization, and data exchange.

Example: Storing multilingual text, emojis, or any data requiring Unicode support.

Implications:

When working with expressions in SSIS:

You can use either DT\_STR or DT\_WSTR as the return type.

Inside an expression, only DT\_WSTR is supported, and DT\_STR values are automatically converted to DT\_WSTR values.

In summary:

DT\_STR is for non-Unicode (ASCII) character data.

DT\_WSTR is for Unicode character data (UTF-16) supporting multiple languages and special characters.

1. Can destination component do UPDATES and DELETES?

Certainly! Let’s explore how destination components in SQL Server Integration Services (SSIS) handle updates and deletes:

Updates:

Destination components can indeed handle updates. When using this approach, you typically follow these steps:

Save the rows to temporary tables (created specifically for this purpose) within the destination component.

After the data flow task, use an Execute SQL task to perform bulk update operations against the destination table from the temporary tables.

This allows you to efficiently update existing records in the destination based on changes from the source.

Deletes:

Similarly, destination components can handle deletes. Here’s how:

Save the rows to temporary tables (again, created for this purpose) within the destination component.

After the data flow task, use an Execute SQL task to perform bulk delete operations against the destination table from the temporary tables.

This ensures that any records marked for deletion in the source are removed from the destination.

Overall Approach:

The general pattern involves identifying inserts, updates, and deletes in the source data.

You then push the updates and deletes to staging tables.

Finally, you use set-based SQL statements (UPDATE and DELETE) to apply these changes to the destination from the staging tables.

In summary, destination components can handle both updates and deletes, allowing you to efficiently synchronize data between source and destination systems in SSIS.

1. What is the need of OLEDB command component?

The OLE DB Command component in SQL Server Integration Services (SSIS) serves a specific purpose and is useful in certain scenarios. Let’s explore its need and functionality:

Purpose:

The OLE DB Command component is used for row-level operations within an SSIS data flow.

Unlike other destination components (such as OLE DB Destination or Flat File Destination), which handle bulk inserts, the OLE DB Command component allows you to execute individual SQL commands for each row of data.

Use Cases:

The OLE DB Command component is typically used when:

You need to perform customized updates, deletes, or other operations on a row-by-row basis.

The transformation logic cannot be achieved using standard SSIS transformations.

You want to apply business rules or complex calculations during data flow execution.

How It Works:

For each input row, the OLE DB Command component executes a parameterized SQL command against the specified database connection.

You define the SQL command (usually an UPDATE or DELETE statement) with placeholders for parameters.

The component maps input columns to these parameters, allowing you to customize the command for each row.

Performance Considerations:

The OLE DB Command component can be inefficient for large datasets because it executes individual SQL commands sequentially.

It involves round-trips to the database for each row, which can impact performance.

Use it judiciously and consider alternatives (such as staging tables and set-based operations) for better performance.

In summary, the OLE DB Command component is valuable when you need fine-grained control over row-level operations during data flow execution. However, due to its potential performance impact, use it selectively and optimize your SSIS packages accordingly.

1. What are params and sequence in OLEDB command?  
   Params allow you to customize SQL commands for each row using placeholders.

Sequence determines the order in which SQL commands are executed for individual rows in the OLE DB Command component.

Labs 7

What is the use of the lookup component?

The **Lookup Transformation** in **SQL Server Integration Services (SSIS)** is a powerful tool used for comparing data between the source and destination. Let’s explore its use cases and functionality:

1. **Purpose of the Lookup Transformation**:
   * The Lookup Transformation allows you to:
     + Compare data from the input (source) with data in a reference dataset (such as a table or query).
     + Filter out matched and unmatched rows based on specified conditions.
     + Perform lookups to access additional information in related tables.
2. **Common Use Cases**:
   * Here are some scenarios where the Lookup Transformation is useful:
     + **Data Enrichment**:
       - You have a fact table with foreign keys (e.g., product IDs, customer IDs).
       - You want to enrich the fact table by looking up additional information (e.g., product names, customer details) from dimension tables.
     + **Data Cleansing**:
       - You need to validate data against reference data (e.g., validating customer ZIP codes against a master ZIP code table).
       - Rows with invalid or unmatched data can be redirected to an error output.
     + **Incremental Data Loading**:
       - When loading data incrementally (e.g., from a staging table to a data warehouse), you can use the Lookup Transformation to identify new, updated, or deleted records.
3. **Configuration**:
   * In the Lookup Transformation, you configure:
     + The connection manager to the reference dataset (usually an OLE DB connection).
     + The columns to compare (input columns vs. reference columns).
     + The lookup mode (full cache, partial cache, or no cache).
     + The output (matched, unmatched, or both).
4. **Performance Considerations**:
   * The Lookup Transformation can be resource-intensive, especially when using full cache mode.
   * Consider using partial cache or no cache for large datasets.
   * Optimize your package design to minimize lookups when possible.

In summary, the Lookup Transformation is essential for data enrichment, validation, and incremental loading scenarios in SSIS. [It allows you to compare data efficiently and make informed decisions during ETL processes](https://www.sqlshack.com/an-overview-of-the-lookup-transformation-in-ssis/)

What is the difference between compiled and debug mode in SSIS?

Certainly! Let’s explore the differences between **compiled mode** and **debug mode** in **SQL Server Integration Services (SSIS)**:

1. **Compiled Mode**:
   * **Definition**: Compiled mode refers to the state of an SSIS package after it has been successfully built (compiled) without any errors.
   * **Purpose**:
     + In compiled mode, the SSIS package is ready for deployment and execution.
     + It ensures that the package is syntactically correct and all components are properly connected.
   * **Characteristics**:
     + No breakpoints are active.
     + No debugging information is generated.
     + The package is optimized for execution.
2. **Debug Mode**:
   * **Definition**: Debug mode is used during development and testing to identify and fix issues in the SSIS package.
   * **Purpose**:
     + Debugging allows you to step through the package, inspect variables, and identify errors.
     + It helps locate logic errors, data flow issues, and unexpected behavior.
   * **Characteristics**:
     + Breakpoints can be set to pause execution at specific points.
     + Debugging information (such as variable values) is available.
     + The package runs slower due to additional overhead for debugging.
3. **Workflow**:
   * During development, you work in debug mode to identify and resolve issues.
   * Once the package is stable and error-free, you switch to compiled mode for deployment.
4. **Switching Modes**:
   * In Visual Studio, you can switch between compiled mode and debug mode:
     + Use the **Start Debugging** button (or press F5) to run the package in debug mode.
     + Use the **Start Without Debugging** button (or press Ctrl+F5) to run the package in compiled mode.

In summary, compiled mode is for production deployment, while debug mode is for development and troubleshooting. [Debugging provides detailed insights into package behavior, whereas compiled mode ensures optimal execution](https://stackoverflow.com/questions/2651243/what-is-the-difference-between-debug-and-compile)

Lab 8L Sort, Merge and Merge joins.

Lab 9: Creating SSAS Cube (SSAS)

Issue: When Installing SQL server for first time, I chose tabular mode on my MSQLSERVER instance for Analysis Services, which made it impossible for me to launch my SSAS cube (which is set to Multidimensional mode). I had to create a new instance named SSAS\_MultiDimen, select analysis services again and finally choosing multidimensional mode.

I created new directories (just like the first installation for good practices for saving data about log, data, temp and backup files) with SSAS as a suffix at the end.

I went to the SSASCustomer projects, clicked on properties, then the deployment tab, I changed the server from localhost to EMMAN\SSAS\_MultiDimen (YourServer\SSAS\_MultiDimen). I rebuilt the projects and clicked on process which managed to solve the issue.

Here’s the steps I followed which were recommended by chatGPT:

To verify the mode of your SQL Server Analysis Services (SSAS) instance in SQL Server Management Studio (SSMS), follow these steps:

1. **Open SQL Server Management Studio (SSMS)**:
   * Launch SSMS from your Start menu or desktop.
2. **Connect to the Analysis Services Instance**:
   * In the "Connect to Server" dialog box, select "Server type" as "Analysis Services".
   * Enter the server name (e.g., localhost if it's installed locally).
   * Click "Connect".
3. **Check the Server Mode**:
   * After connecting, right-click on the Analysis Services instance name in the Object Explorer.
   * Select "Properties" from the context menu.
4. **View the Server Mode**:
   * In the "Properties" window, go to the "Information" section.
   * Look for the "Server Mode" property. This will indicate the current mode of the SSAS instance (e.g., Multidimensional, Tabular)

Since your existing SQL Server Analysis Services (SSAS) instance is set to "Tabular" mode and it’s not possible to change it directly, you’ll need to install a new SSAS instance in "Multidimensional and Data Mining" mode. Here are the steps to do this:

**Installing a New SSAS Instance in Multidimensional Mode**

1. **Launch SQL Server Installation Center**:
   * Open the SQL Server Installation Center from your Start menu or search for "SQL Server Installation Center".
2. **Start a New Installation**:
   * Click on "New SQL Server stand-alone installation or add features to an existing installation".
3. **Setup Support Rules**:
   * The installer will run a setup support rules check. Ensure all checks pass, then click "Next".
4. **Specify a New SQL Server Instance**:
   * On the "Instance Configuration" page, select "Named instance" and provide a unique name for this new instance (e.g., SSAS\_Multidimensional).
5. **Select Analysis Services**:
   * On the "Feature Selection" page, select "Analysis Services" and click "Next".
6. **Server Configuration**:
   * On the "Server Configuration" page, set the appropriate service account and collation settings, then click "Next".
7. **Select Multidimensional Mode**:
   * On the "Analysis Services Configuration" page, select "Multidimensional and Data Mining Mode".
8. **Configure SSAS Security**:
   * Add users who will have administrative rights to the Analysis Services instance.
9. **Complete the Installation**:
   * Follow the remaining prompts to complete the installation. Once the installation is complete, you will have a new SSAS instance running in Multidimensional mode.

**Update Visual Studio to Use the New SSAS Instance**

1. **Open Your Visual Studio Project**:
   * Open the project containing your cube.
2. **Update Deployment Server**:
   * Right-click on your project in Solution Explorer and select "Properties".
   * Go to the "Deployment" tab.
   * Change the "Server" property to the new SSAS instance name you specified during installation (e.g., YourServerName\SSAS\_Multidimensional).
3. **Deploy the Cube**:
   * Right-click the project and select "Deploy". This will deploy the cube to the new SSAS instance running in Multidimensional mode.
4. **Process the Cube**:
   * After deploying, right-click the cube and select "Process". Ensure the processing completes without errors.

Second Encountered Issue: Internal error: The operation terminated unsuccessfully. The following system error occurred: The user name or password is incorrect.

Solution: I opened the data source, I went to impersonation information, and edit the login detail by providing my email address (which is the admin) and my password. In conclusion, set the impersonation information tab within the data source designer dialog box with the admin credential for the appropriate privilege.