Understanding Slowly Changing Dimensions (SCD)

Slowly Changing Dimensions are the methods used to monitor changes in the dimension attributes, manage updates, helping businesses preserve historical data and ensure accuracy in reporting. In data warehousing, a common problem is to manage changes in dimensional data over time. This is where we use the concept of Slowly Changing Dimensions (SCD). This reading will provide a brief explanation on the types of SCD and also discuss their benefits, usage and considerations in designing a data warehouse.

Various types of SCDs:

There are four primary types of SCDs:

Type 0: Retain Original Value

Type 1: Overwrite the Existing Data

Type 2: Preserve Historical Data

Type 3: Add New Attribute

However, in most of the advanced implementations, the below types are also used, which combines or extend some of the basic types.

Type 4: Historical Table

Type 6: Hybrid Approach

Type 0: Retain Original Value

This can be used on Static Dimension which means once a value is inserted, it will remain static. No changes will be made to the dimension data in Type 0. Also, the historical data is not updated. This approach is beneficial for data which should remain constant over time. Examples like product codes or account numbers. The major advantage with Type 0 is, it is simple to implement and more effective for dimensions that rarely change.

Type 1: Overwrite the Existing Data

Type 1 i.e., overwrite the existing data applies changes to the dimension directly by overwriting the existing data. This way does not maintain a record of historical changes, so if an attribute value is updated, the old value will be lost. For example, when only the current state of the data is important, such as correcting spelling mistakes or updating any contact information.

Pros:

Easy to implement.

Saves storage space.

Cons:

No historical data is retained.

May lead to inaccurate historical reporting.

Example: If a customer changes their address, the new address overwrites the old one

Type 2: Preserve Historical Data (Row Versioning)

Type 2 i.e., preserve historical data allows you to track changes by adding new rows in the dimension table whenever there's an update. Each of the row will have the current and the historical versions of the data and start/end dates (or) flags are used to indicate whether the row is the current version. For example, when it is essential to retain a full history of changes, such as tracking customer address changes for legal compliance/auditing.

Pros:

In Type 2, full historical data will be preserved.

It is easier to retrieve data using queries as it existed at any point in time.

Cons:

Type 2 increases the size of the dimension table.

This mainly requires careful management of versioning fields.

Example: A new row will be created with the new address when a customer updates their address, while the old row will be marked as historical.

Type 3: Add New Attribute (Tracking Limited History)

Type 3 adding a new attribute will track the historical changes by adding new columns to the dimension table. Each of the column represents a different version of the attribute. This is helpful when only a limited amount of the historical data needs to be stored, such as the previous and current values. For example, when you need to track a small number of changes and also when it is only necessary to compare the previous and the current states.

Pros:

Type 3 is easy to implement.

This requires very less space than Type 2.

Cons:

Type 3 can only track a limited amount of history.

This does not maintain a total history of changes.

Example: Storing a customer's current address and previous address in separate columns in the same row.

Type 4: Historical Table (Tracking Historical Data in a Separate Table)

In Type 4, historical data will be stored in a separate table from the current dimension data. In this, the main dimension table holds only the current data, while a separate historical table stores all the previous versions of the data. For example, when you like to separate current data from historical data to improve performance and to simplify the design.

Pros:

Type 4 will maintain a complete historical record.

This usually separates current and historical data.

Cons:

Type 4 is more complex to implement.

This mainly requires additional storage for historical tables.

Example: A current customer table having only the latest updated information, while an associated historical customer table holding the older records.

Type 6: Hybrid Approach

Type 6 is a hybrid approach that combines aspects of all the **Type 1**, **Type 2**, and **Type 3**. This retains the full history like Type 2, will have a current flag like Type 1, and also it will track the previous versions like Type 3. This method helps in accessing the current data, compare it with the previous versions, and also maintains a complete historical record. For example, if you need a flexible solution to track both the current and the historical versions of data, and also requires the comparisons of previous values.

Pros:

Type 6 combines the advantages of other types like Type 1, Type 2 and Type 3.

This will track the complete history by maintaining the current state.

Cons:

Type 6 is more complex to manage.

This mainly requires more storage.

Example: When a customer changes their address, the dimension table has a current address field (Type 1), new rows in the table to track full historical changes (Type 2), and a previous address field (Type 3).

Key Considerations in Implementing SCD:

. **Business Requirements:** Before choosing an SCD type, assess the business requirements. Do you need to track historical changes? If so, how much history do you need to keep?

. **Versioning:** As mentioned earlier, type 2 often requires a start date, end date, and a current flag to manage the different versions of the same dimension row. Ensure to handle these fields carefully in order to avoid errors in version control.

. **Storage and Performance:** Tracking the historical data can increase the size of the dimension tables. Always consider the performance impact on queries that access the dimension tables.

. **Extract, Transform, Load (ETL) Process:** The ETL process should be designed properly to fit the type of SCD in use. For example, Type 1 ETL only updates existing rows while the Type 2 ETL needs to detect changes and insert new rows.

Conclusion:

Slowly Changing Dimensions (SCDs) always provide very strong way to manage changes in dimension data over time. Organizations can ensure accurate reporting, maintain historical data, and optimize the performance of their data warehouses by carefully selecting the proper SCD type based on their business requirements. Whether you require simple overwriting (Type 1) or full

historical tracking (Type 2) or a hybrid solution (Type 6), the right SCD strategy will help you achieve long-term data management success.