WHITE PAPER

Managing Data Warehouse Deletes

Best Practices for Managing SOFT and HARD Deletes

QLIK.COM

LEAD WITH DATA Qlik

TABLE OF CONTENTS

SUMMARY

- Data Warehouse requirements may include reacting to delete operations in source systems
- Qlik Replicate can be configured to ensure that Qlik Compose can perform soft deletes in the data warehouse
- Qlik Compose can support hard or soft deletes on data not delivered by Qlik Replicate via its custom code feature

INTRODUCTION

Data warehouses are optimized to store historical data for complex analytics and reporting. Historical data can include transactional history (e.g. store all transactions for seven years), or slowly changing dimensions attribute history (e.g. manage changes to specific attributes like name or address, over time). Consequently, we rarely purge data from the data warehouse.

However, operational systems that feed the data warehouse, frequently delete records. Some systems allow for complete record removal, while others impose restrictions. As a result, the data warehouse can experience side effects of record deletion form the source systems.

In short, managing record deletion in a data warehouse is complicated, and this paper describes the best practices for handling various data deletion scenarios.

Deletion Methods

The introduction mentioned how data warehouses are often structured to store historical data for complex analytics and reporting. And we explained that there are many business reasons that dictate how we handle the retention and deletion of historical data. Sometimes there are no obligations to retain data, whilst other times regulatory requirements prevent us from doing so. As a result, there are generally two strategies we employ to handle record deletion:

1. Soft Deletes - A soft delete is the process of marking a record as “deleted”, but not actually removing the record from the database. Soft deletes are often accomplished by adding a “deleted” flag or a “deleted timestamp” column that is populated when the delete occurs.

2. Hard Deletes – Hard deletion it the process of physically purging the data from database tables. This deletion method is often permanent and cannot be rolled back or undone. Generally, data warehouses encounter hard deletes less frequently.

Deletion Side Effects

Why are different types of deletions important? Operational systems that feed the data warehouse can treat data deletion in both ways and the method is often chosen based on the business requirement. For example, closing a bank account rarely removes the account from the system of record because certain banking regulations may stipulate that the records must be retained for a period. In this instance the record is updated with the date that the account was closed, and a flag is set that the account has been deleted.

Other transactional systems support true deletes. For example, an inventory system may delete stock item records as they are depleted and shipped. In this scenario the item records are purged and no longer available for reference.

In either case, there are business conditions where deletes in source systems (or data suddenly missing from a bulk delivered source) need to be reflected in the data warehouse and we need to perform a soft or hard delete. The remainder of this paper describes methodologies to handle soft and hard deletes when Qlik Replicate is sourcing the data for Qlik Compose, and for those situations where Qlik Compose is processing bulk data sources.

Qlik Data Integration Record Deletion Strategy

Qlik Compose ignores deletes by default since deleting records in the data vault can significantly impact the integrity of the data and relationships it manages. (See the Soft Delete Scenario side bar for an example where hard deletes would cause issues). Therefore, performing a soft delete is the recommended approach to for managing deletes in the data warehouse.

There are always specific business conditions that do require a hard delete however, you should **carefully** consider the impact to your data warehouse and data integrity before opting to delete records from the data vault.

When there is a delete requirement and Qlik Replicate is delivering data for Qlik Compose to process, then you can configure Qlik Replicate to automatically provide soft delete flags, which Qlik Compose will subsequently update in the data vault. If a hard delete is required, the soft delete flags can be used to filter deletes either immediately or after a specific time frame has passed, using Qlik Compose Post Loading ETL features.

If data is not delivered by Qlik Replicate, or Qlik Replicate is delivering a bulk load only to the data warehouse, then there are design patterns to custom code soft or hard deletes.

Soft Delete Scenario

Let’s consider a scenario for soft deletes.

Your data warehouse manages seven years of sales transactions with reference data on products and customers. In the operational system a product is deleted because it is no longer sold (and has not been sold for 3 years).

Deleting the product in the data warehouse would result in the inability to view accurate analytics for sales by product characteristics (whether at the product level or a higher grain of dimensionality like product category).

In this situation the best practice is to mark the product as deleted in source, perhaps with a delete date, to allow for future deletion once any remaining sales transaction data has “rolled-off” in the data warehouse.

Note

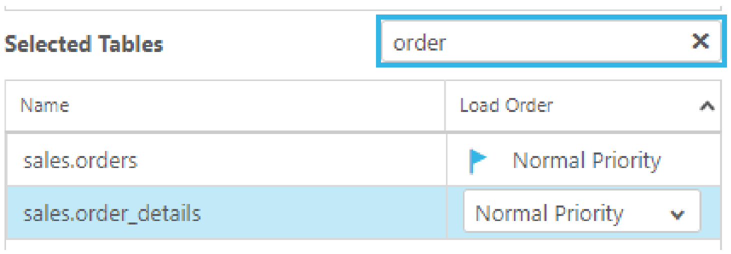
Performing deletes are always recommended instead of truncating Qlik Compose data vault tables due to the relationship management and data integrity automation provided by Qlik Compose.

Configuring Qlik Replicate to Perform Soft Deletes

Qlik Replicate can provide a soft delete flag to Compose in the bulk and change tracking (_ct) tables. Soft deletes can be issued per table, groups of tables or for the entire set of data in a replication task. In this example the configuration will be applied to a single table.

To configure Qlik Replicate to provide a soft delete flag perform the following in your replicate task:

- In the Qlik Replicate console, open your replication task
- Under the **Selected Tables** (if specific tables have been selected) or **Full Table List** (if a pattern of tables has been selected) find the table to apply the soft delete.
- Double click on the table open the **Table Settings** (in this case below - *order_details*)



Selected Tables — order

Name	Load Order
sales.orders	Normal Priority
sales.order_details	Normal Priority

- Click on **Transform**
- In the Transform screen you can apply record level transformations to the replication process.
- In this case you will create 2 additional columns with following data types
  - DELETED_FLAG STRING(1)
  - DELETED_DATETIME DATETIME(6)
- Click **Add Column** and add the 2 columns as described above.

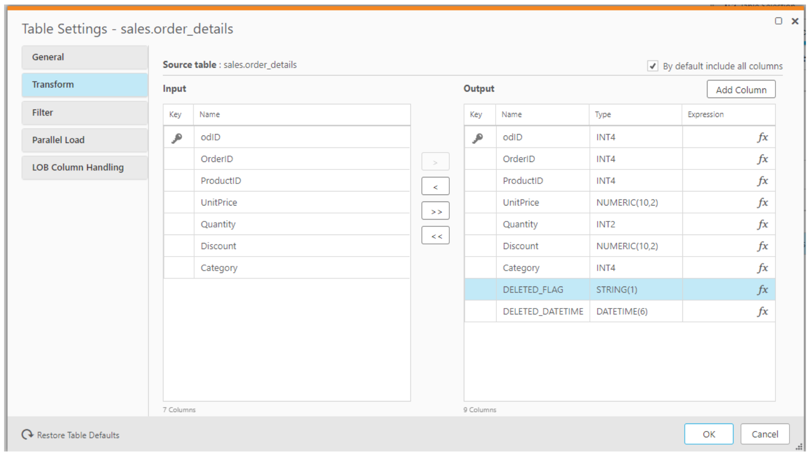


Table Settings - sales.order_details

Source table : sales.order_details

By default include all columns [✓]

Input

Key	Name
🔑	odID
	OrderID
	ProductID
	UnitPrice
	Quantity
	Discount
	Category

7 Columns

Output

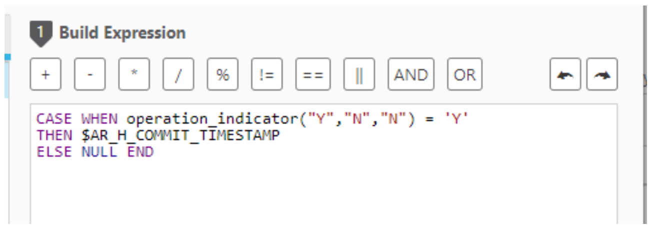
Key	Name	Type	Expression
🔑	odID	INT4	*fx*
	OrderID	INT4	*fx*
	ProductID	INT4	*fx*
	UnitPrice	NUMERIC(10,2)	*fx*
	Quantity	INT2	*fx*
	Discount	NUMERIC(10,2)	*fx*
	Category	INT4	*fx*
	DELETED_FLAG	STRING(1)	*fx*
	DELETED_DATETIME	DATETIME(6)	*fx*

9 Columns

- Click the **fx** button to open the expression editor and enter the formula to calculate the value for each column
- Enter the expressions below for each column.

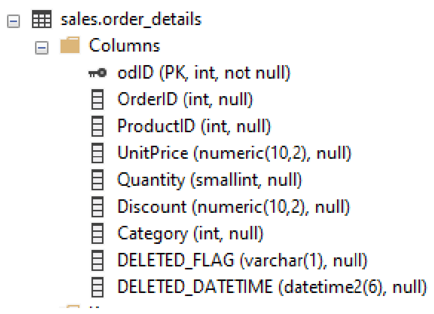
Column	Expression to Enter	Comments
DELETED_FLAG	Ifnull(operation_indicator("Y","N","N"), “N”)	Flag records deleted on source with a “Y”. Other valid records will have an “N’. The ifnull condition ensures a ‘N’ value for Replicates initial load. The values for the function can be altered based on your specific requirements.
DELETED_DATETIME	*CASE WHEN operation_indicator("Y","N","N") = 'Y'* *THEN $AR_H_COMMIT_TIMESTAMP* *ELSE NULL END*	Use the source commit timestamp as the deleted date for any deleted records. Other, valid records will have NULL. The values for the function can be altered based on your specific requirements.

- The image below depicts the DELETED_DATETIME expression



```
CASE WHEN operation_indicator("Y","N","N") = 'Y'
THEN $AR_H_COMMIT_TIMESTAMP
ELSE NULL END
```

- Click **OK** to close the dialog box.
- Save the task and perform a full load to the target environment.
- Review the replicated and change tracking (_ct) target table structures. Both sets of tables will now have two additional fields that depict delete operations in the source which can be used by Qlik Compose to denote soft deletes.


```
sales.order_details
  Columns
    odID (PK, int, not null)
    OrderID (int, null)
    ProductID (int, null)
    UnitPrice (numeric(10,2), null)
    Quantity (smallint, null)
    Discount (numeric(10,2), null)
    Category (int, null)
    DELETED_FLAG (varchar(1), null)
    DELETED_DATETIME (datetime2(6), null)
```

Qlik Replicate is now configured for soft deletes for the *order_details* table.

Note

Your use case may not require both *DELETED_FLAG* and *DELETED_DATETIME* columns and you may of course, alter the column names. If the configuration needs to be applied to multiple tables, it is recommended to use Qlik Replicate's **GLOBAL TRANSFORMATION** feature to add the columns to a subset or all of tables.

Please review **GLOBAL TRANSFORMATIONS** in the Qlik Replicate users guide for further details on this feature.

Operation_indicator function

When the *operation_indicator* function is invoked on its own or as part of an expression, records deleted from the source endpoint will not be deleted from the target endpoint. Instead, the corresponding target record will be flagged (with a user-provided value) to indicate that it was deleted from the source.

The *operation_indicator* function also requires you to provide values to indicate records that were inserted or updated in the source endpoint. When used, the operation_indicator function forces Replicate to convert a DELETE into and UPDATE operation when delivering to the target.

Function Syntax: *operation_indicator*(value_on_delete, value_on_update, value_on_insert)

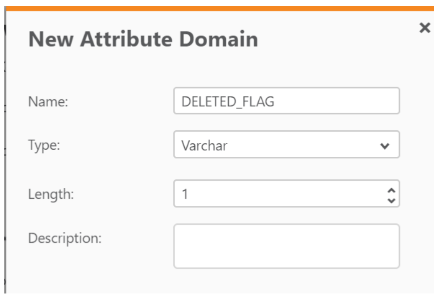
Configuring Qlik Compose for Soft Deletes

To support soft deletes the Qlik Compose model and mappings must be configured to manage the DELETED_FLAG and DELETED_DATETIME columns.

Model Soft Delete Flags

The soft delete flags should be added to the model. When discovering the model from the source system, Compose reads the source system metadata. Since the soft delete flags are added in the replication pipeline the columns will not be defined in the model. The model can be manually edited to add the columns to the tables that require soft deletes. This can be accomplished by:

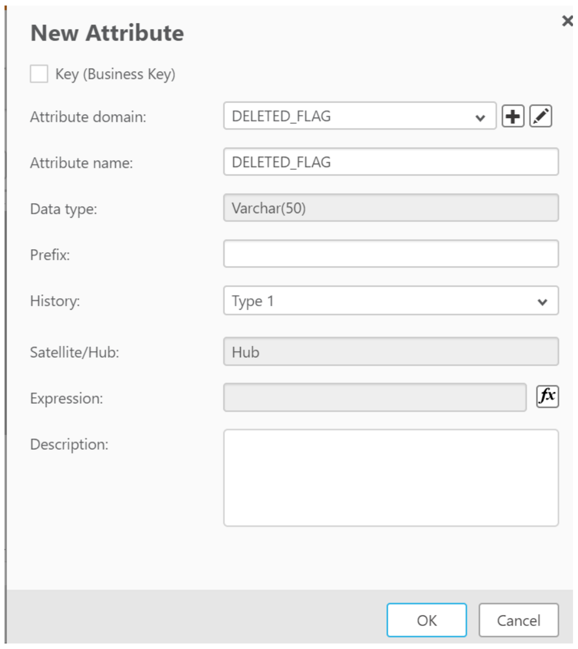
- Navigate to the Compose console and open your project
- Click **Manage** under the **Model** layer
- Select your entity (in this case *order_details*) and click **New Attribute** to add the required attributes. If required create a new Attribute Domain for DELETED_FLAG (VARCHAR(1)) and DELETED_DATETIME DATETIME



New Attribute Domain

Name: DELETED_FLAG
Type: Varchar
Length: 1
Description:

- It is recommended to set the History to *Type 1* which denotes if that instance is currently deleted
    - If you have a requirement to detect history of deletes, it is recommended to create 2 attributes for each flag column – 1 for history and 1 for the current value to easily detect and filter currently deleted data


New Attribute

Key (Business Key)	☐
Attribute domain:	DELETED_FLAG
Attribute name:	DELETED_FLAG
Data type:	Varchar(50)
Prefix:	
History:	Type 1
Satellite/Hub:	Hub
Expression:	
Description:	

- Once the attributes have been defined (either by manually or automatically discovered) you can validate or create the data warehouse model and deploy the tables to the data warehouse.

Now that the model has been defined any mappings must also be edited to support updating the soft delete flags.

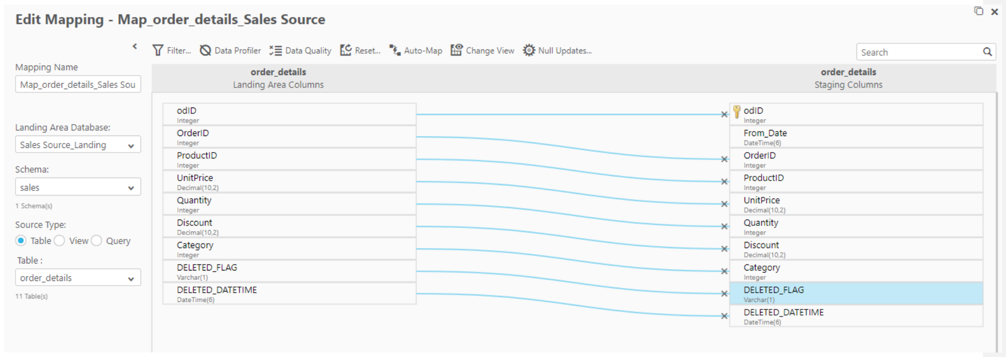
Mapping Configuration for Qlik Replicate Source

The Qlik Compose model definitions dictate how updating the *DELETED_FLAG* and *DELETED_DATETIME* columns are managed. When Qlik Replicate is delivering data to the data warehouse the deleted columns will be available in the landing area but will not automatically be defined in the Qlik Compose mapping.

To define the column mappings:

- Click **Manage** under the *Data Warehouse* section of Compose to open the mappings
- Double-click the table you added the *DELETED_FLAG* and *DELETED_DATETIME* columns to.

- Drag the source *DELETED_FLAG* and DELETED_DATETIME columns to the targets



Edit Mapping - Map_order_details_Sales Source

Mapping Name: Map_order_details_Sales Sou

Landing Area Database: Sales Source_Landing

Schema: sales

1 Schema(s)

Source Type: [✓ Table] View Query

Table : order_details

11 Table(s)

order_details Landing Area Columns		order_details Staging Columns	
odID	Integer	odID	Integer
OrderID	Integer	From_Date	DateTime(6)
ProductID	Integer	OrderID	Integer
UnitPrice	Decimal(10,2)	ProductID	Integer
Quantity	Integer	UnitPrice	Decimal(10,2)
Discount	Decimal(10,2)	Quantity	Integer
Category	Integer	Discount	Decimal(10,2)
DELETED_FLAG	Varchar(1)	Category	Integer
DELETED_DATETIME	DateTime(6)	DELETED_FLAG	Varchar(1)
		DELETED_DATETIME	DateTime(6)

- Close the mapping and **Generate** the ETL code for any ETL Sets that contain the mapping.

- If Qlik Replicate is not delivering data to the warehouse, but the source does contain a soft delete flag of some kind then a similar process can be used to update the data in the vault.

Qlik Compose will now update the soft delete flags based on the data fed to it via Qlik Replicate.

Managing Soft Delete for Non-Replicate Sources

Using Qlik Replicate to deliver data for Qlik Compose to process is a best practice because the two products are tightly integrated. Qlik Replicate processes changes as they are delivered to the data warehouse. However there are scenarios where the source system does not provide change data capture capabilities and Qlik Replicate is used to batch load the data only (for example csv files). This is commonly known as performing a “Full Load”. You may also use Qlik Compose to process other data sources that are not delivered by Qlik Replicate too (for example when Qlik Catalog loads a json file to the data warehouse). In these scenarios Qlik Compose is not notified of a delete via replication, but must detect it.

In these cases, there are some custom code requirements to support soft delete processing. Qlik Compose does not automatically identify deletes when detecting changes between bulk sources and the data warehouse. You must manually code an update statement that can be executed in a single-table ETL set or in a post loading ETL step to update the data vault hub or satellite tables. As with many scenarios, this can be managed in a variety of ways. Two possible options are defined as follows:

- Option 1 – LastActiveDateTime
    - Use Qlik Compose to update a "LastActiveDateTime" column to the current datetime for each row that is present in the bulk data set.
    - Use Post Load ETL to update the DELETED_FLAG and DELETED_DATETIME attributes based on the value in the LastActiveDateTime attribute. (Update the flags when LastActiveDateTime does not have the max value)
- Option 2 – Manual Update
    - Write an Update statement to be executed in a Post Loading ETL to update the DELETED_FLAG or DELETED_DATETIME columns when the record exists in the hub, but not in the source system

A comparison between these two approaches is provided below:

	LastActiveDateTime	Manual Update
Description	Use Compose to update the value of a "LastActiveDateTime" column for each row that is present in the bulk data set. Derive the *DELETED_*FLAG and *DELETED_*DATETIME attributes based on the value in the *LastActiveDateTime* attribute. (Update the flags when LastActiveDateTime does not have the max value).	Write an UPDATE statement to be executed either in Post Loading or Single-Table ETL to manually update the *DELETED_FLAG* and *DELETED_DATETIME* columns.
Modelling Requirements	Add a LastActiveDateTime attribute to the entity to be processed	No additional modelling requirements beyond the DELETED_FLAG / DELETED_DATETIME

Soft Delete Update Requirements	Set the expression for the attribute to the data warehouse platforms "current timestamp" function. (e.g. getdate() for sql server).
	
	Add a Post Loading ETL to simply update the entities DELETED_FLAG and DELETED_DATETIME where the LastActiveDateTime <> MAX(LastActiveDateTime) from the HUB
		Write a POST LOADING or Single-Table ETL statement to manually update the DELETED_FLAG & DELETED_DATETIME columns
		e.g. UPDATE order_details_HUB
		SET DELETED_FLAG = 'Y', DELETED_DATETIME = getdate()
		WHERE NOT EXISTS (SELECT 1 FROM <source_table> s WHERE s.key1 = order_details_HUB.key1)

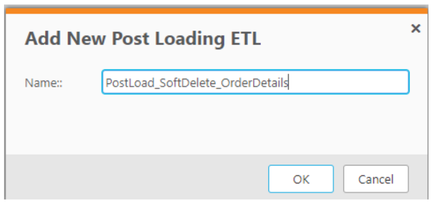
Pro's	Simplified post-load processing as Compose ETL takes care of updating the LastActiveDateTime each time the ETL set executes. Provides a LastActiveDateTime to know the last time the record appeared in the source system. Manual update statement is the same for every entity	No additional attributes to manage for the entity Compose processing is simplified
Con's	Additional attribute to manage in the entity. Compose will update every record that is in the bulk data set for every ETL run.	Post loading / single-table etl is hand coded per entity where soft-deletes must be applied.

Non-Replicate Soft Delete Example

Let's work through an example using the *Manual Update* method above and the *order_details* table already defined. Assuming the source table for *order_details* provides only current order information, you must write an update statement as a Post Load ETL step to mark records as deleted when they don't exist in the source tables. Note the below sample code is written for SQL Server. If using Snowflake or another data warehouse platform the syntax may need to be adjusted.

To configure Qlik Compose:

- Navigate to your project in Qlik Compose
- Click **Manage** under the Data Warehouse section to open the ETL mappings
- Click **Post Loading ETL** and **+ New Post Loading ETL…**
- Enter a name for the post load process

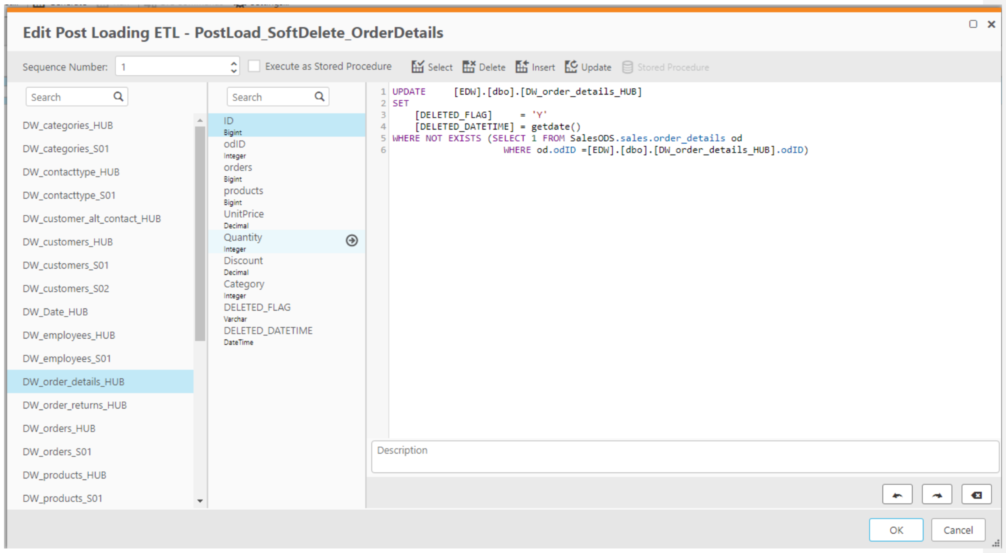


Add New Post Loading ETL

Name:: PostLoad_SoftDelete_OrderDetails

- Click **OK**
- Select the table to perform the soft delete operation on (in this case its *DW_order_details_HUB*)
- Clicking **UPDATE** will provide a skeleton update statement that you can alter. Alternatively type in the update statement to update the *DELETED_FLAG* and *DELETED_DATETIME* fields
- In the example below, odID is the business key for the entity. The update statement updates any record that does not have a corresponding entry in the SalesODS.sales.order_details table (the source table). Note that there could be additional logic needed here. For example if the source only provides orders for the last six months then the update statement would need to take that into account (only mark records where the order date is within the last six months).

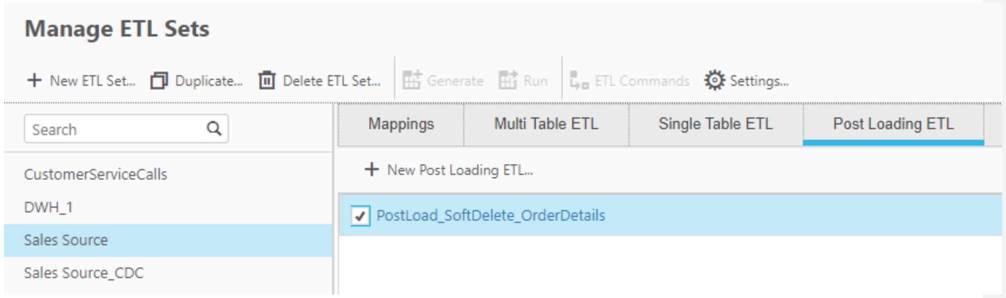
Commented [CB1]: You didn't finish your thought!


Edit Post Loading ETL - PostLoad_SoftDelete_OrderDetails

Sequence Number: 1

```
UPDATE     [EDW].[dbo].[DW_order_details_HUB]
SET
    [DELETED_FLAG]     = 'Y'
    [DELETED_DATETIME] = getdate()
WHERE NOT EXISTS (SELECT 1 FROM SalesODS.sales.order_details od
                  WHERE od.odID =[EDW].[dbo].[DW_order_details_HUB].odID)
```

- If there are multiple Post Load ETL steps, ensure you define the correct *sequence number* to sequence the tasks.
- Click **OK** once the code is correct
- Assign the post load step to the correct ETL Sets by selecting the ETL set in the left and then activating the step (check the check box)

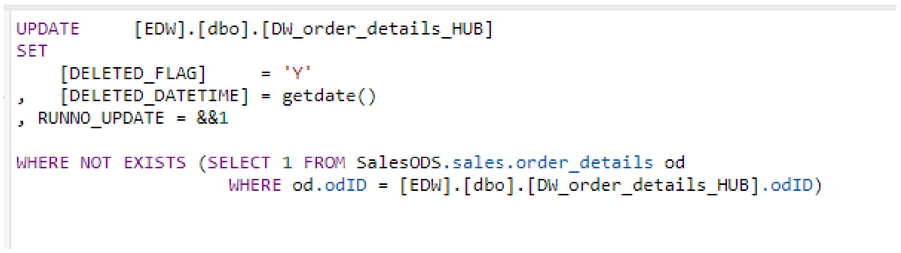


- Generate the code for the ETL Sets impacted.

Execute the ETL sets to test and ensure the update is applied correctly for data that no longer appears in the bulk data source.

Note that any updates handled via Post-Load custom code will not automatically be cascaded to the data marts via Compose. To ensure Compose Data Mart processes pick up custom coded updates you must update the RUNNO_UPDATE for the appropriate HUB tables to the current ETL run no. This can be done using the internal Compose parameter '&&1'.

For example the sample process defined above would be adjusted to the below. The RUNNO_UPDATE operational column will then be updated to the current value – which will trigger data mart processes to pick up the altered column.



```
UPDATE     [EDW].[dbo].[DW_order_details_HUB]
SET
    [DELETED_FLAG]     = 'Y'
,   [DELETED_DATETIME] = getdate()
, RUNNO_UPDATE = &&1

WHERE NOT EXISTS (SELECT 1 FROM SalesODS.sales.order_details od
                    WHERE od.odID = [EDW].[dbo].[DW_order_details_HUB].odID)
```

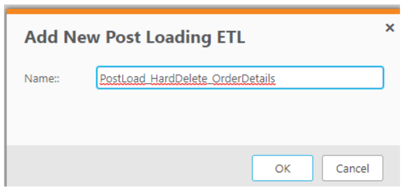
Hard Delete Operations

Qlik Compose manages relationships between entities as part of its data warehouse automation features. As such hard deletion of data within a specific entity could cause issues when populating the data vault and when processing data marts. Consider a scenario where an order_return is related to an order_detail record. Depending on the Qlik Compose model, deletion of the order_detail could invalidate associated records in order_returns. You should **carefully** consider the impact to your data warehouse and data integrity before opting to physically / hard delete records from the data vault.

It is recommended to perform deletion using the soft delete method mentioned earlier in this paper, however there are certain business conditions that require physical deletion of data. Even for these scenarios, it is recommended to leverage the soft delete methodology to mark records for deletion and then deleting based upon approved retention policies / validation that deletion of the record will not adversely impact the data warehouse. (The *DELETED_DATETIME* attribute can be used to determine retention of soft delete records).

Qlik Compose does not provide a facility for automating the deletion of data in the vault. Manual code must be written as part of a Post Loading ETL Step to perform a physical delete of data. To configure this process in Qlik Compose:

- Navigate to your project in Qlik Compose
- Click **Manage** under the Data Warehouse section to open the ETL mappings
- Click **Post Loading ETL** and **+ New Post Loading ETL…**
- Enter a name for the post load process

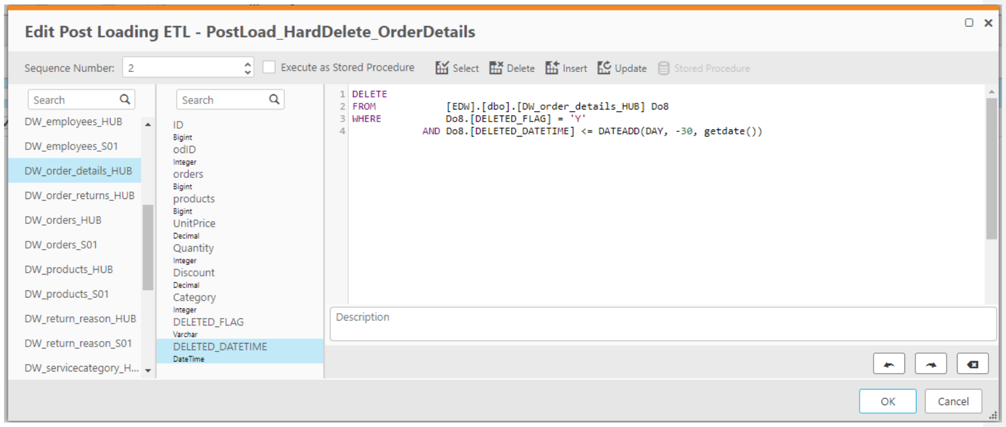


Add New Post Loading ETL

Name:: PostLoad_HardDelete_OrderDetails

- Click **OK**
- Select the table to perform the hard delete operation on (in this case its *DW_order_details_HUB*)
- Clicking **DELETE** will provide a skeleton delete statement that you can alter. Alternatively type in the delete statement to delete records based on your criteria.

- In the example below, records are only deleted once they have been in that state for 30 days.



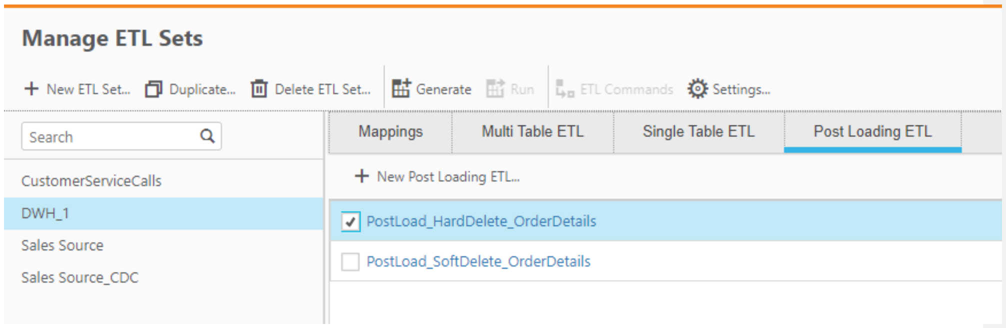
Edit Post Loading ETL - PostLoad_HardDelete_OrderDetails

Sequence Number: 2

Execute as Stored Procedure

```
DELETE
FROM            [EDW].[dbo].[DW_order_details_HUB] Do8
WHERE           Do8.[DELETED_FLAG] = 'Y'
            AND Do8.[DELETED_DATETIME] <= DATEADD(DAY, -30, getdate())
```

- Note that deletion of data from the vault will not automatically delete records from the data mart and the post load step may need to additionally delete data from the corresponding fact / dimension table in the data mart.
- If there are multiple Post Load ETL steps, ensure you define the correct *sequence number* to sequence the tasks.
- Click **OK** once the code is correct
- Assign the post load step to the correct ETL Sets by selecting the ETL set in the left and then activating the step (check the check box)


Manage ETL Sets

CustomerServiceCalls
DWH_1
Sales Source
Sales Source_CDC

✓ PostLoad_HardDelete_OrderDetails
PostLoad_SoftDelete_OrderDetails

- Generate the code for the ETL Sets impacted.
- Execute the ETL Set to perform the hard delete operations

Qlik Compose is now configured to perform hard deletes of data from the data vault.

Data Mart Considerations

The above defines how to implement a delete strategy for the central data vault, however you must consider how deletes should be handled in the data mart also. If the DELETE_FLAG is processed by Compose ETL mappings then any update to a DELETE_FLAG will cascade to the data mart. This will simply update a “DELETE_FLAG” column in the data mart, it will not delete the record from the mart.

If the DELETE_FLAG has been manually set (for example via a Post-Load ETL steps) then you must ensure you update the RUNNO_UPDATE for Compose to recognize the changed records (see the example for non-Replicate data sources). This ensures Compose cascades the update to the DELETE_FLAG.

Note that Compose will **not** automatically delete the data in the data mart. To perform a hard delete in the Data Mart you can leverage the Post-Load steps for data marts.

Note also that leveraging a filter for the data mart “WHERE DELETED_FLAG = ‘Y’” will not remove existing records that exist in the data mart.

Conclusion

While propagation of deletes are sometimes required in a data warehouse, you should carefully consider soft versus hard deletes and the potential impact on data integrity within the data vault before customizing Qlik Compose to handle those operations. It is not recommended to simply truncate data vault tables due to the referential integrity managed by Qlik Compose. Qlik Replicate and Qlik Compose provide the features necessary to automate the processing required to load a data warehouse and they can be easily configured to support the soft delete requirements for your data warehouse.

About Qlik

Qlik’s vision is a data-literate world, one where everyone can use data to improve decision-making and solve their most challenging problems. Only Qlik offers end-to-end, real-time data integration and analytics solutions that help organizations access and transform all their data into value. Qlik helps companies lead with data to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 50,000 customers around the world. **qlik.com**

© 2020 QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

LEAD WITH DATA