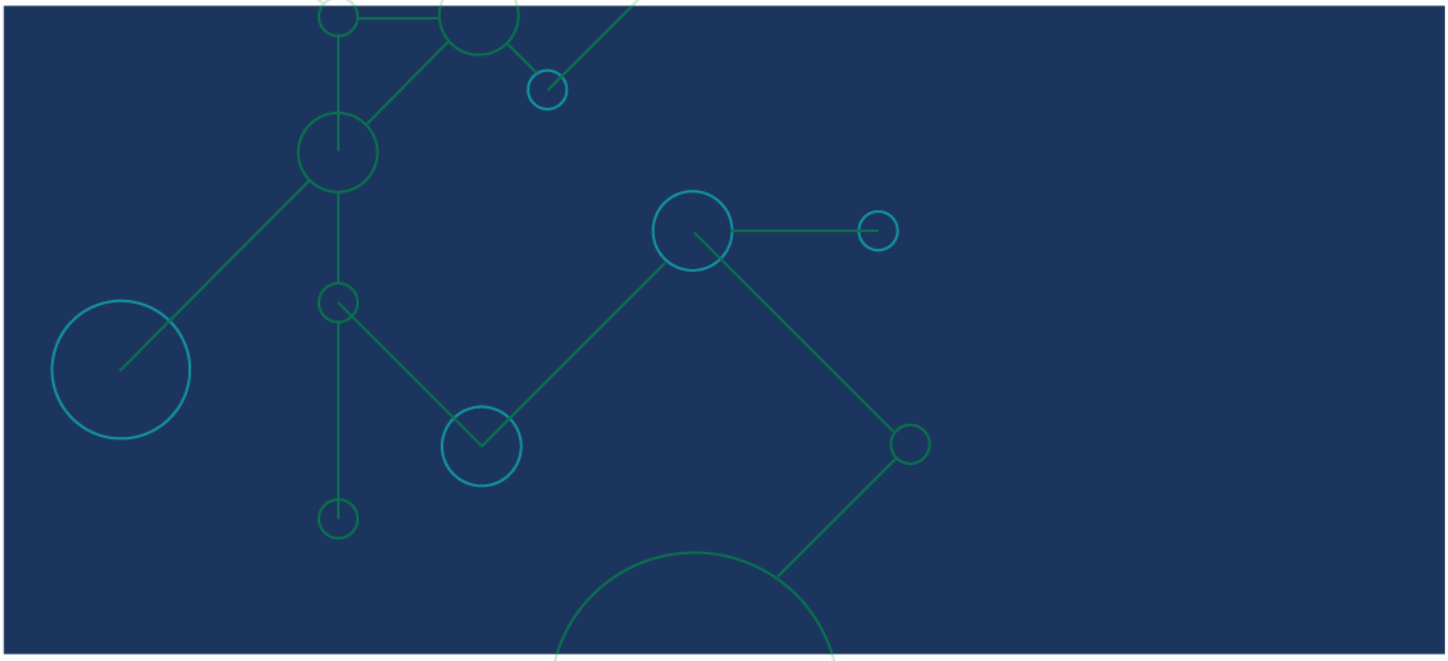
WHITE PAPER

# Defining Data Models and Mappings for Pivoting Data

Data Warehouse Best Practices



## TABLE OF CONTENTS

## EXECUTIVE SUMMARY


- Data warehouse ingestion and loading often requires transforming (or pivoting) data from a source data structure to a target data warehouse model.
- Pivoting data is often a batch-oriented method that requires repeated processing of the full data set.
- The Qlik Data Integration Platform provides out of the box capabilities that support both batch and incremental change data processing to pivot source data sets to data warehouse models.


## INTRODUCTION

Operational systems of record often provide data that is not readily consumable by analytics projects and frequently require complex extract, transform and load (ETL) processing to conform the information into a standard data warehouse model.

One common transformation is to pivot multiple rows into a single row with multiple columns to provide a flattened data set for analysis. Most ETL technologies provide a capability to perform that transformation when processing data in batch; however, the Qlik Data Integration platform provides methods that support **both** batch and near real-time pivoting of source data into the data warehouse.

This paper provides real-world examples that explain the purpose of a pivoted data set, describes how to model the data structures, and how to implement transformation logic for batch and near real-time processing.

# Pivoting Data Use Case

Core operational systems are often modeled in third normal form (3NF) for faster inserts, updates, and deletes. However, while this is an appropriate structure for a transactional system it does not lend itself to business intelligence (BI), dashboarding, and reporting. Although ETL tools and manual SQL scripting are potential solutions, many find them time consuming to implement, even for common data transformation tasks such as pivoting data.

Consider the simple example below. An application data source has a *CustomerContact* table that stores a distinct contact type and value for each customer. Additionally, a customer with multiple contact types are stored as multiple rows. In our example customer *ALFKI* has 3 contact types records which for AltPhone, Email and Website rows.

| CustomerID | AlternateContactType | Contact |
|---|---|---|
| ALFKI | Email | Maria.Jones@AlfredsFutterkiste.com |
| ALFKI | Home | 9999 |
| ALFKI | Website | www.AlfredsFutterkiste.com |
| ANATR | Email | Ms..Ana.Trujillo@AnaTrujilloEmparedadosyhelados.... |
| ANATR | Home | (5) 555-471 |
| ANATR | Website | www.AnaTrujilloEmparedadosyhelados.com |

*Figure 1. CustomerContact Table*

But what happens when our business users request reports that consist of just one row per customer as below?

| ID | CustomerID | WebsiteContact | EmailContact | AltPhoneContact |
|---|---|---|---|---|
| 1 | ALFKI | www.AlfredsFutterkist... | Maria.Jones@AlfredsF... | 030-007439 |
| 2 | ANATR | www.AnaTrujilloEmpar... | Ms..Ana.Trujillo@AnaT... | (5) 555-471 |
| 3 | ANTON | www.AntonioMorenoT... | Antonio.Moreno@Ant... | (5) 555-393 |
| 4 | AROUT | www.AroundtheHorn.... | Thomas.Hardy@Arou... | (171) 555-77P |
| 5 | BERGS | www.Berglundssnabb... | Christina.Berglund@B... | 0921-12 34 2 |
| 6 | BLAUS | www.BlauerSeeDelikat... | Hanna.Moos@BlauerS... | 0621-0846 |
| 7 | BLONP | www.Blondelpèreetfils... | Frédérique.Citeaux@B... | 88.60.15.0 |
| 8 | BOLID | www.BólidoComidasp... | Martín.Sommer@Bóli... | (91) 555 22 3 |

*Figure 2. A Pivoted Data Set*

The solution is to convert the data using a PIVOT transformation where one record per CustomerID consists of multiple columns for each *AlternateContactType* value. This common real-world requirement

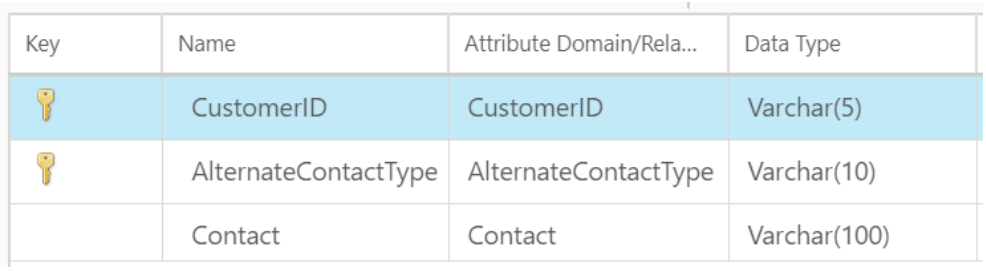
is traditionally managed by ETL tools and a set of complex transformation components that are difficult to manage and configure. These processes are not only difficult to write, but also require maintenance whenever your data warehouse structures change.

### Pivoting Data in Qlik Data Integration

The Qlik Data Integration solution is an alternative approach. Qlik's philosophy is extract, load and transform (E-LT). ELT leverages the data warehouse engine to perform the transformation vs a middle-tier server favored by traditional ETL solutions. The benefit of the ELT approach is that we can leverage the native PIVOT capabilities of the target data warehouse platform for transformations.

## Defining the Warehouse Data Model

Qlik provides multiple methods to help you implement the logical data structure for your warehouse. The most common method is to use the solution to intelligently discover data models from source systems, an example of which is below.



| Key | Name | Attribute Domain/Rela... | Data Type |
|---|---|---|---|
| 🔑 | CustomerID | CustomerID | Varchar(5) |
| 🔑 | AlternateContactType | AlternateContactType | Varchar(10) |
| | Contact | Contact | Varchar(100) |

*Figure 3. Source Data Models*

However, the discovered model needs amending because we want to manage a pivoted data set with one record per customer. To adjust the discovered model, we use the Contact attribute domain. Attribute Domains define data types and allow for consistent type management across multiple columns.

To change the discovered structures, we do the following:

- Delete the *AlternateContactType* and *Contact* attributes
- Add attributes for each *AlternateContactType* we want the warehouse to manage. There are 3 in our example: Website, Phone and Email.

- Re-use the *Contact* attribute domain to ensure the data type matches the source system and is consistent for all the attributes created.

The following diagram depicts the use of the *Contact* attribute domain to create an attribute. The Prefix has been used to better define the attribute as a ***Website****Contact.*

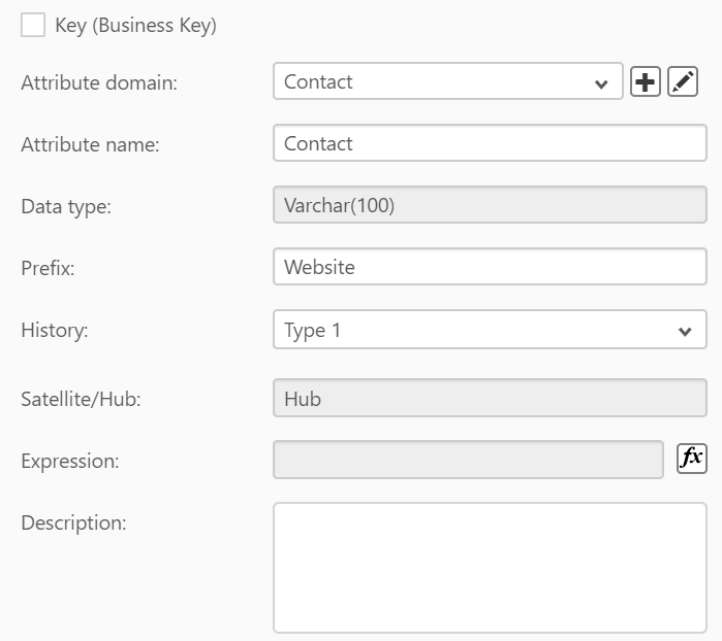



*Figure 4. Attribute Domain Details*

Once the attributes have been defined the *AlternateContact* entity should look like the diagram below:

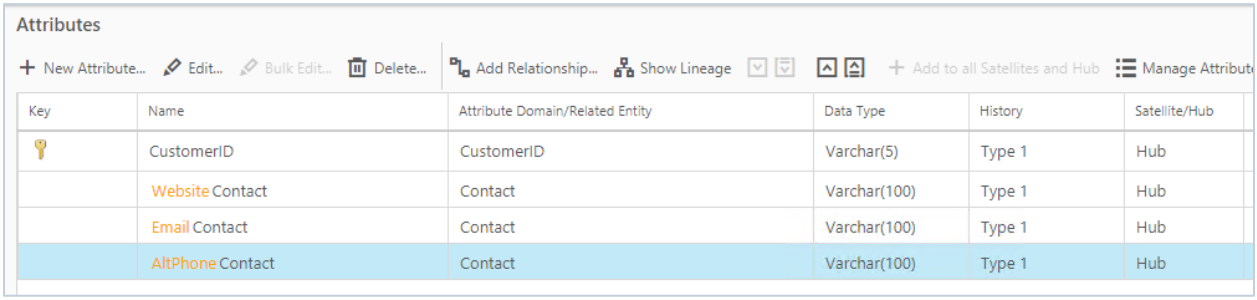



*Figure 5. AlternateContact Entity Details*

The history management for each attribute can be altered based on your business requirements. In the example we only wish to maintain the current (Type 1) version of each contact. The final step of data model definition is to use Qlik's functionality to **CREATE** or **VALIDATE** the model to automatically generate, and then execute, the table creation SQL in the target data warehouse.

# Define Mappings for PIVOT

Now that the model has been defined, we must create appropriate data mappings to transform the source data into our target warehouse structures. Remember, Qlik is an Extract Load and Transform (E-LT) based solution. ELT solutions leverage the data warehouse environment as the compute and set-based SQL to process and transform the data (as opposed to ETL solutions which leverage a mid-tier, proprietary engine for transformation logic).

## Mappings and ETL Set Types

Qlik Compose for Data Warehouses describes mappings as the business logic that loads a single target entity. However, it's more than that. The business logic is an automated pattern that detects differences between source and target data, applies Type 1 or Type 2 processing to the entity, and relates entities via their surrogate keys. Qlik Compose also defines a collection of mappings that run together as an ETL set.

Qlik Compose defines two types of ETL sets**: BULK** and **CDC** and the table below describes the differences:

### What makes real-time PIVOT transformations difficult for traditional ETL tools?

Processing data in real-time complicates any ETL process. Real-time methods are designed to be more efficient because they are only triggered in response to changes in source records.

Consider the example data set. A change to a single row, e.g. a customers website contact would require traditional ETL to processes every single contact record for the given customer to ensure a complete data set. Processing only the changed record would result in partial records because it does not have all the data (NULL values in contact columns that are NOT "Website").

Qlik Compose provides a unique feature to automatically detect the last NOT-NULL value for a column thus completing the data set and ensure values in every column. This feature is critically important to efficiently perform real-time PIVOT transformations.

| | BULK ETL Set | CDC ETL Set |
|---|---|---|
| **Data Source Used** | Source tables / views / queries selected in the mapping. | Replicate **Store Change (__ct tables)**. |
| **What It Does** | Processes all data in the source tables / views / query, detecting changes between the source and data warehouse via ELT processing. | Process records in the Replicate **Store Changes** ( __ct tables) into the data warehouse, then deletes or archives the **__ct** rows it has processed |
| **When to Use It** | • Initial load of the data warehouse<br>• Full load only data sources (from Replicate or other load processes)<br>• When all data needs to be processed (e.g. snapshot of inventory loaded to the dw on a daily basis) | • Near real-time / intraday data loads<br>• Batch oriented data loads that should process changed data only (instead of all data) |
| **Limitations / Items to Consider** | Bulk loads process all the source data and could have an impact on batch SLA's. | CDC sets only support table-based mappings. |

In most cases you can leverage the same mapping for BULK ETL and CDC-based ETL processing. While the business logic and mappings can be re-used, the generated SQL code will be different for each method of execution. There are cases however where defining a specific BULK mapping and CDC mapping make sense – mostly for performance reasons. Pivoting data is one of those cases where it is recommended to define a specific mapping for each execution methodology.

The illustration below highlights the mapping editor features. Pivoting uses the *Data Source*, *Column Mapping / Transformation*, *Function* and *NULL Handling Configuration* components.

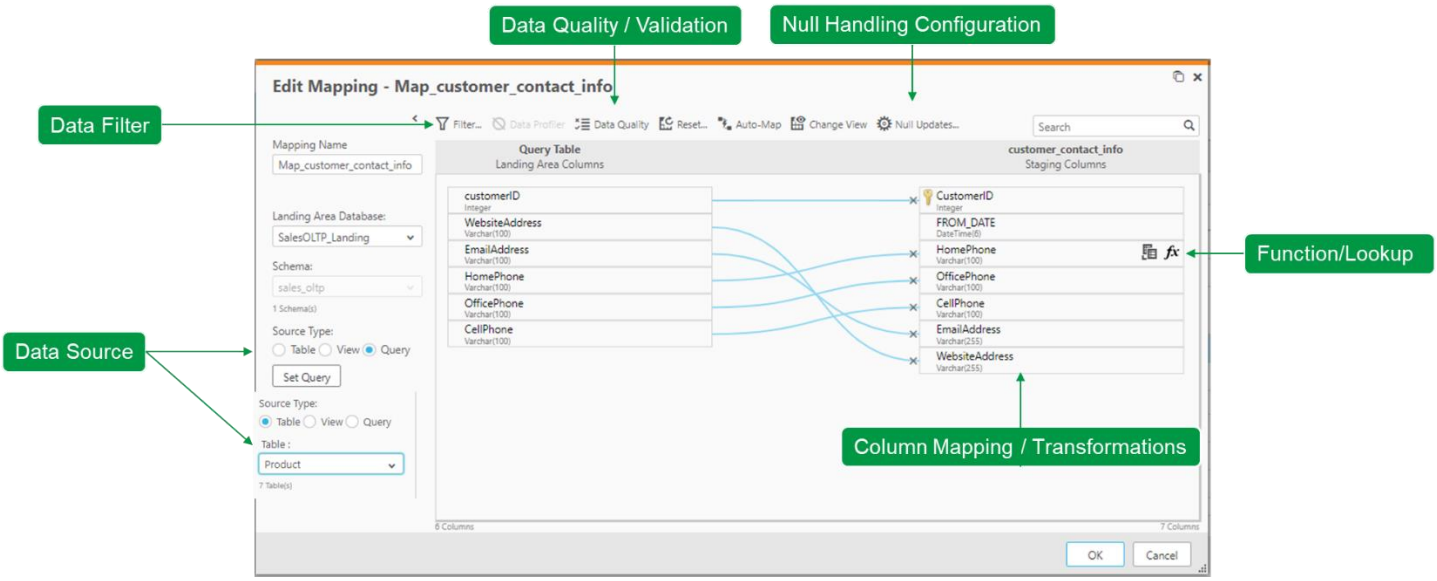



*Figure 6. Mapping Editor*

## Defining a BULK Mapping for PIVOT

Mappings can use *Tables*, *Views* or a *Query* as the source. Use *View* or *Query* as the source when pivoting data or performing a bulk / initial load. In the example that follows we will choose to use *Query* so that all the logic is encapsulated within Qlik Compose.

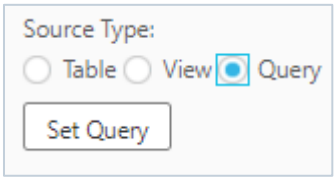



*Figure 7. Source Type Selection*

The bulk mapping should reflect your target data warehouse SQL language. Note in our example case we will use the PIVOT syntax for Microsoft SQL Server. To create a bulk mapping, perform the following steps:

- Create a new mapping for the target entity and set the **Source** to *Query*.
- Click Set Query button
- Enter the SQL query to pivot the source data dialog box.

```
SELECT CustomerID, [WebSite],[AltPhone],[Email]
FROM SalesODS.sales.customer_alt_contact c
PIVOT ( MAX(Contact)
FOR AlternateContactType IN ([WebSite],[AltPhone],[Email] )) p
```

- The former query uses SQL Server's PIVOT function to convert rows to columns. MAX or MIN must functions are used for PIVOT to enforce correct aggregation by ensuring each column receives only one value (in the event there are multiple). For our example use case – since we know there is only one value, it doesn't matter if we select MIN or MAX. The IN clause in PIVOT commands provide the filter for rows to pivot into columns. In our example if there were an *AlternateContactType* of "HomePhone" it would be excluded from the dataset. Each value in the IN clause becomes a column in the output of the query.
- Click **OK** and map each source column to the appropriate target column.

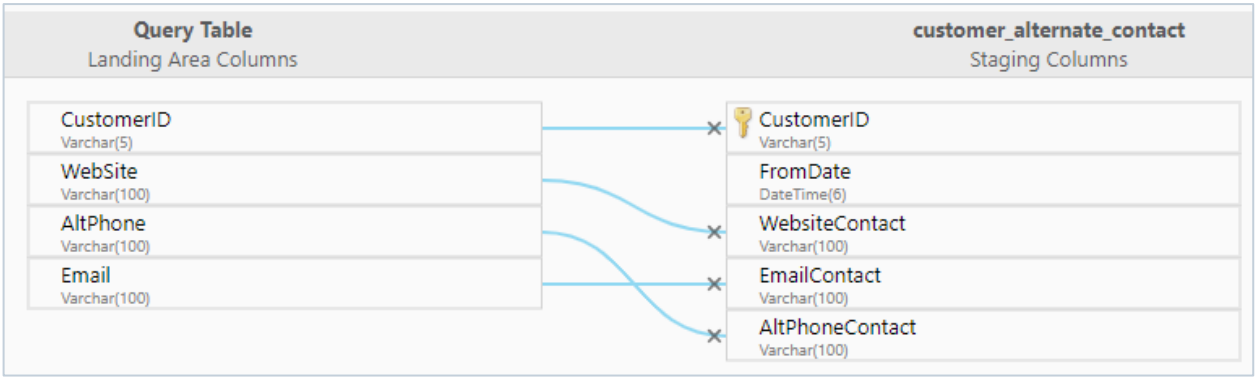



*Figure 8. Field Mapping*

- Finally assign this mapping to a BULK ETL set, generate and execute the mapping.

The SQL query to PIVOT provides the best performance for transforming the entire source data set and performing an initial load of data into the target data warehouse. This bulk method can also be used if the data source only supports bulk refreshes of data as the BULK ETL set will detect changes and apply them to the data warehouse.

## Defining a CDC Mapping for PIVOT

Qlik Data Integration is architected to deliver data into your data warehouse in near real-time. This is accomplished by Qlik Replicate delivering change data to change tracking (__**ct** ) tables and Qlik Compose CDC ETL set functionality processing those changes. (If you are not aware of Qlik Replicate's **Store Changes** setting please review the *Using Change Tables* section in the Qlik Replicate User Guide.)

As mentioned, CDC ETL sets generate code differently to BULK ETL sets – using the __ct tables as the source instead of the defined source table. CDC ETL sets are only applicable to mappings defined with a *Table* as the source. A mapping with a query or view can entail complex logic with multiple source tables, table valued functions and any other SQL construct supported in the warehouse environment.  A CDC ETL set does not know how to calculate changes from that type of mapping due to the unlimited features that could be leveraged in the query or view.  Thus, the BULK mapping cannot be used for CDC processing.  Also, of note is that the __ct tables contain only changes to the source since the last execution of the Qlik Compose ETL set.

Since a query or view cannot be used as a source, we must define the CDC mapping differently.

- Create a new mapping for the target entity and set the **Source** to *Table*.
- Select the **customer_alternate_contact** table as the source
- Map the *CustomerID* source column to the target column.
- For each pivoted attribute, we will use an expression to map appropriate values. (To open the expression editor, click the **fX** symbol).


*Figure 9. Launching Expression Editor*

- Enter the appropriate CASE statement for the target column.
- Statements for the example are shown in the table below

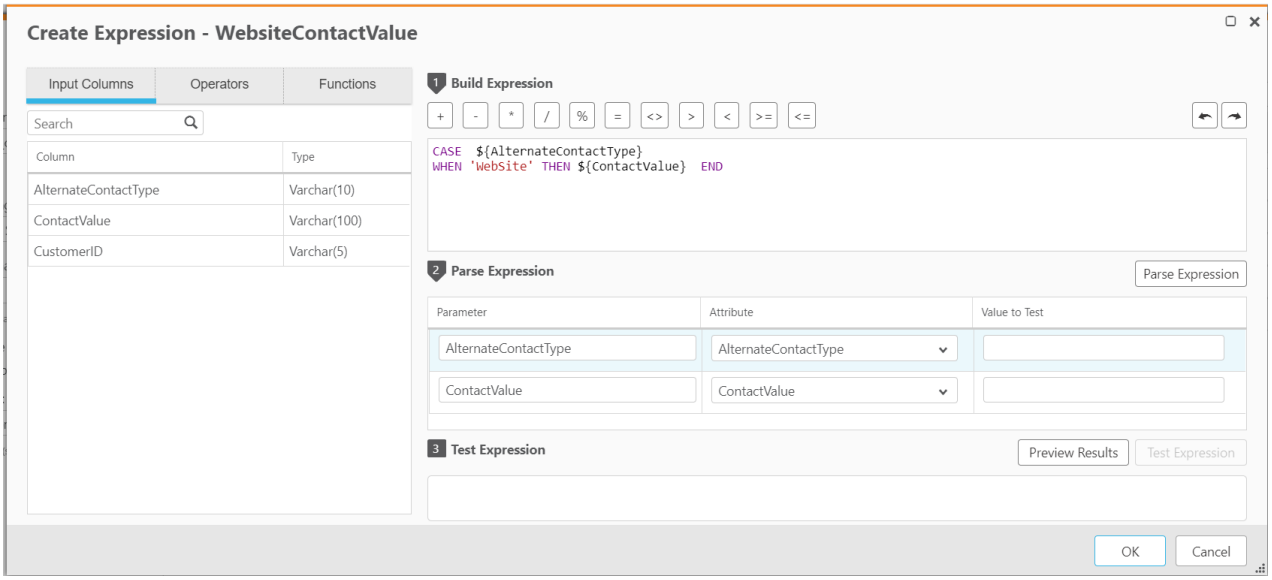



*Figure 10. Expression Editor*

| Column | Expression |
|---|---|
| **WebSiteContact** | CASE ${AlternateContactType}<br>WHEN 'WebSite' THEN ${Contact} END |
| **EmailContact** | CASE ${AlternateContactType}<br>WHEN 'Email' THEN ${Contact} END |
| **AltPhoneContact** | CASE ${AlternateContactType}<br>WHEN 'AltPhone' THEN ${Contact} END |

Your mapping should now look like the diagram below with an expression for each field and the *CustomerID* column mapped.


*Figure 11. Mapped Fields*

Let's consider our example use case and the mapping logic.

The initial set of data for CustomerID ALFKI produced the below result in the data warehouse

.

| CustomerID | AlternateContactType | Contact |
|---|---|---|
| ALFKI | Email | Maria.Jones@AlfredsFutterkiste.com |
| ALFKI | Home | 9999 |
| ALFKI | Website | www.AlfredsFutterkiste.com |

*Figure 12. Table Before Pivot*

| ID | CustomerID | WebsiteContact | EmailContact | AltPhoneContact |
|---|---|---|---|---|
| 1 | ALFKI | www.AlfredsFutterkist... | MariaJones@AlfredsF... | 030-007439 |

*Figure 13. Table After Pivo*

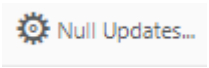
If a change occurs to a the AltPhone record the __ct table contains only that change as below.

| Operation | CustomerID | AlternateContactType | Contact |
|---|---|---|---|
| **UPDATE** | ALFKI | AltPhone | 555-5566 |

However, the currently defined CDC ETL set would only see that single record and the resulting mapping logic produces an output with an incomplete record.

| CustomerID | WebsiteContact | EmailContact | AltPhoneContact |
|---|---|---|---|
| ALFKI | **NULL** | **NULL** | 555-5566 |

If Qlik Compose simply applied this to the target, then the data warehouse would show NULL values for *WebsiteContact* and *EmailContact* columns. This would be incorrect. However, Qlik has a feature that allows us to leverage both data it is processing and existing data in the data warehouse to detect the **last not-null value**. This capability allows us to understand the prior value for *WebsiteContact* and *EmailContact* to fill the gaps and produce complete records.

To use this feature in the mapping, click 

and select *Do not change the target value*.

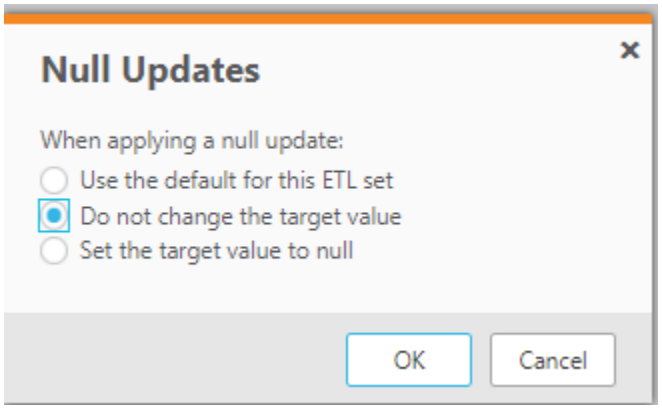



*Figure 14. Null Update Handling*

When processing the change Qlik will determine the correct values for *WebsiteContact* and *EmailContact* to process Type 1 or Type 2 changes appropriately as we see in the following table.

| CustomerID | WebsiteContact | EmailContact | AltPhoneContact |
|---|---|---|---|
| **ALFKI** | www.AlfredsFutterkiste.com | Maria.Jones@AlfredsFutterkiste.com | 555-5566 |

The last step to define a CDC mapping is to assign the mapping to the CDC ETL set, ensuring you select “Handle Duplicate” and then generate the SQL code. Handle Duplicates will ensure Qlik Compose reacts appropriately to multiple changes to the same customer within a micro-batch.

The CDC mapping will now efficiently process source changes as they are delivered by Qlik Replicate. If your use case requires historical management of the data change (i.e. slowly changing dimension type 2) simply alter the model and define each attribute as Type 2. Qlik Compose CDC processing will appropriately detect the changes, fill out the partial record with its NULL processing capabilities and perform Type 2 operations on the target data warehouse tables.

# Conclusion

Qlik Compose for Data Warehouses provides capabilities to support complex modeling and transformation requirements for real-time data warehouses. Mappings often work in both BULK and CDC mode with no changes, but there are use cases that require some customization. In addition, Qlik Compose provides unique features to support complex transformations in near real-time which are often difficult to accomplish in traditional ETL tools.

# Appendix: Test Scripts

The following Microsoft SQL Server SQL Statements help you create the example use case in your own environment.

1. Create source table.

```
CREATE TABLE sales.customer_alt_contact(CustomerID varchar(5) NOT NULL,
   AlternateContactType varchar(10) NOT NULL,Contact varchar(100) NULL,
CONSTRAINT customer_alt_contact_PRIMARY PRIMARY KEY (
   CustomerID ,
   AlternateContactType
))
```

2. Insert an initial sample set of records

```
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ALFKI', N'Email', N'Maria.Jones@AlfredsFutterkiste.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ALFKI', N'AltPhone', N'9999') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ALFKI', N'Website', N'www.AlfredsFutterkiste.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ANATR', N'Email', N'Ms..Ana.Trujillo@AnaTrujilloEmparedadosyhelados.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ANATR', N'Website', N'www.AnaTrujilloEmparedadosyhelados.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ANTON', N'Email', N'Antonio.Moreno@AntonioMorenoTaquerÃ-a.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ANTON', N'AltPhone', N'(5) 555-393') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'ANTON', N'Website', N'www.AntonioMorenoTaquerÃ-a.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'AROUT', N'Email', N'Thomas.Hardy@AroundtheHorn.com') ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES
(N'AROUT', N'AltPhone', N'(171) 555-77P') ;
```

3. Develop the solution described in the paper in Compose and execute the BULK load. To assist with tracking changes, set the attributes to Type 2 in the Qlik Compose model.

4. Execute the update and insert statements to simulate changes to the source system.

```
UPDATE sales.customer_alt_contact
SET Contact = '555-5566'
WHERE CustomerID = 'ALFKI' ;
INSERT sales.customer_alt_contact (CustomerID, AlternateContactType, Contact) VALUES (N'ANATR',
N'AltPhone', N'555-6655') ;
```

5. Execute the CDC ETL set to see the changes processed incrementally.


## 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 serves over 50,000 customers around the world.

.