Merge Transform

Contents:

Merges two or more columns in your dataset to create a new column of String type. Optionally, you can insert a delimiter between the merged values.

> **NOTE:** This transform applies to String columns or other columns that can be interpreted as strings (for example, Zip codes could be interpreted as five-digit strings). To concatenate arrays, use the `ARRAYCONCAT` function. See *ARRAYCONCAT Function*.

Basic Usage

Column example:

```
merge col:Column1,Column2 as:'MergedCol'
```

Output: Merges the contents of `Column1` and `Column2` in that order into a new column called `MergedCol`.

Column and string literal example:

```
merge col:'PID',ProdId with:'-'
```

Output: Merges the string `PID` and the values in `ProdId` together. The string and the value are separated by a dash. Example output value: `PID-00123`.

Syntax and Parameters

```
merge col:column_ref [with:string_literal_pattern] [as:'new_column_name']
```

Token	Required?	Data Type	Description
merge	Y	transform	Name of the transform
col	Y	string	Source column name or names
with	N	string	String literal used in the new column as a separator between the merged column values
as	N	string	Name of the newly generated column

For more information on syntax standards, see *Language Documentation Syntax Notes*.

Copyright © 2021 Trifacta Inc.

col

Identifies columns or range of columns as source data for the transform. You must specify multiple columns.

To specify multiple columns:

- Discrete column names are comma-separated.
- Values for column names are case-sensitive.

```
merge col: Prefix,Root,Suffix
```

Output: Merges the columns `Prefix`, `Root`, and `Suffix` in that order into a new column.

Usage Notes:

Required?	Data Type
Yes	String (column name)

with

Merge transform: Specifies the delimiter between columns that are merged. If this parameter is not specified, no delimiter is applied.

Replace transform: Specifies the replacement value.

```
merge col: CustId,ProdId with:'-'
```

Output: Merges the columns `CustId` and `ProdId` into a new column with a dash (`-`) between the source values in the new column.

Usage Notes:

Required?	Data Type
No	String (column name)

as

Name of the new column that is being generated. If the `as` parameter is not specified, a default name is used.

```
merge col: CustId,ProdId with:'-' as:'PrimaryKey'
```

Output: Merges the columns `CustId` and `ProdId` into a new column with a dash (`-`) between the source values in the new column. New column is named, `PrimaryKey`.

Usage Notes:

Required?	Data Type
No	String (column name)

Copyright © 2021 Trifacta Inc.

Examples

> **Tip:** For additional examples, see *Common Tasks*.

Example - Merging date values

You have date information stored in multiple columns. You can merge columns together to form a single date value.

Source:

OrderId	Month	Day	Year
1001	2	14	2008
1002	7	22	2009
1003	11	22	2010
1004	12	25	2011

Transform:

```
merge col:Month~Year with:'/' as:'Date'
```

Results:

When you add the transform and move the generated `Date` column, your dataset should look like the following. Note that the generated column is automatically inferred as Datetime values.

OrderId	Month	Day	Year	Date
1001	2	14	2008	2/14/2008
1002	7	22	2009	7/22/2009
1003	11	22	2010	11/22/2010
1004	12	25	2011	12/25/2011

Example - Use merge and settype to clean up numeric data that should be treated as other data types

This example illustrates how to clean up data that has been interpreted as numeric in nature, when it is actually String or a structured string type, such as Gender. This example uses:

- `settype` - defines the data type for a column or columns. See *Settype Transform*.
- `merge` - merges two String type columns together. See *Merge Transform*.

Source:

The following example contains customer ID and Zip code information in two columns. When this data is loaded into the Transformer page, it is initially interpreted as numeric, since it contains all numerals.

The four-digit `ZipCode` values should have five digits, with a `0` in front.

CustId	ZipCode
4020123	1234

Copyright © 2021 Trifacta Inc.

2012121	94105
3212012	94101
1301212	2020

Transform:

CustId column: This column needs to be retyped as String values. You can set the column data type to String through the column drop-down, which is rendered as the following transform:

```
settype col:CustId type:'String'
```

While the column is now of String type, future transforms might cause it to be re-inferred as Integer values. To protect against this possibility, you might want to add a marker at the front of the string. This marker should be removed prior to execution.

The basic method is to create a new column containing the customer ID marker (`C`) and then merge this column and the existing `CustId` column together. It's useful to add such an indicator to the front in case the customer identifier is a numeric value that could be confused with other numeric values. Also, this merge step forces the value to be interpreted as a String value, which is more appropriate for an identifier.

```
merge col:'C', CustId
```

You can now drop the `CustId` columns and rename the new column as `CustId`.

ZipCode column: This column needs to be converted to valid Zip Code values. For ease of use, this column should be of type String:

```
settype col:ZipCode type:'Zipcode'
```

The transform below changes the value in the `ZipCode` column if the length of the value is four in any row. The new value is the original value prepended with the numeral `0`:

```
set col: ZipCode value: if(len($col) == 4, merge(['0',$col]), $col)
```

This column might now be re-typed as Zipcode type.

Results:

CustId	ZipCode
C4020123	01234
C2012121	94105
C3212012	94101
C1301212	02020

Remember to remove the `C` marker from the `CustId` column. Select the `C` value in the `CustId` column and choose the `replace` transform. You might need to re-type the cleaned data as String data.

Copyright © 2021 Trifacta Inc.