

Web Data Integration

Introduction to

MapForce



Agenda

1. Project Phase Overview
2. Altova MapForce
 - i. Altova MapForce Introduction
 - ii. Creating an Integrated Target Schema
 - iii. Loading your data into MapForce
 - CSV
 - Excel
 - JSON
 - iv. Creating Correspondences
 - v. Get the Translated Data Out
3. Hands-on: Schema Mapping with Altova MapForce

1. Project Phase Overview

Project Phase 1: Data Collection, Schema Mapping, Data Translation

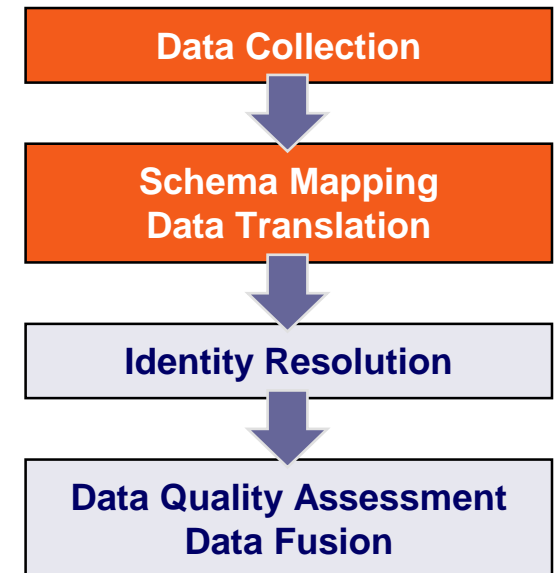
Duration: now – October 16th

Tasks:

1. Decide on a use case
2. Collect data from the Web
3. Profile your data
4. Generate integrated schema (target schema)
5. Convert all your data into the integrated schema using MapForce

Result: All data is represented using a single unified schema

- one XML file per data source



Project Phase Overview: Requirements

You should integrate:

1. at least **3 different data sets**
2. at least **2,500 entities** described in total (in joint dataset)
 - but more are better, good: >10,000 but <100,000
3. at least **1000 entities** should be contained in at least **two datasets**
 - please estimate based on small sample
4. at least **10 attributes** in joint dataset
 - entities should be identifiable by attribute combinations, e.g. name+birthdate
5. at least **5 attributes** should be contained in at least **two datasets**
 - some attributes should be contained in three datasets (for fusion by voting)
6. ideally, that datasets should contain a **list attribute**
 - actors of a movie, directors of a company, songs on a CD

2. Altova MapForce

- Visual Schema Mapping Tool
 - Supports many data formats such as CSV, XML, JSON, EXCEL, ...
 - Build-in mapping functions which can be used by *drag & drop*
- How to get and run *MapForce*
 - Download & Install: <http://www.altova.com/download.html>
 - Run MapForce for the first time and enter the product key, provided in ILIAS
 - Documentation: <http://manual.altova.com/Mapforce/mapforceenterprise/>
- Most tools of Altova can be tested for 30-days for free

Altova MapForce Interface

Insert

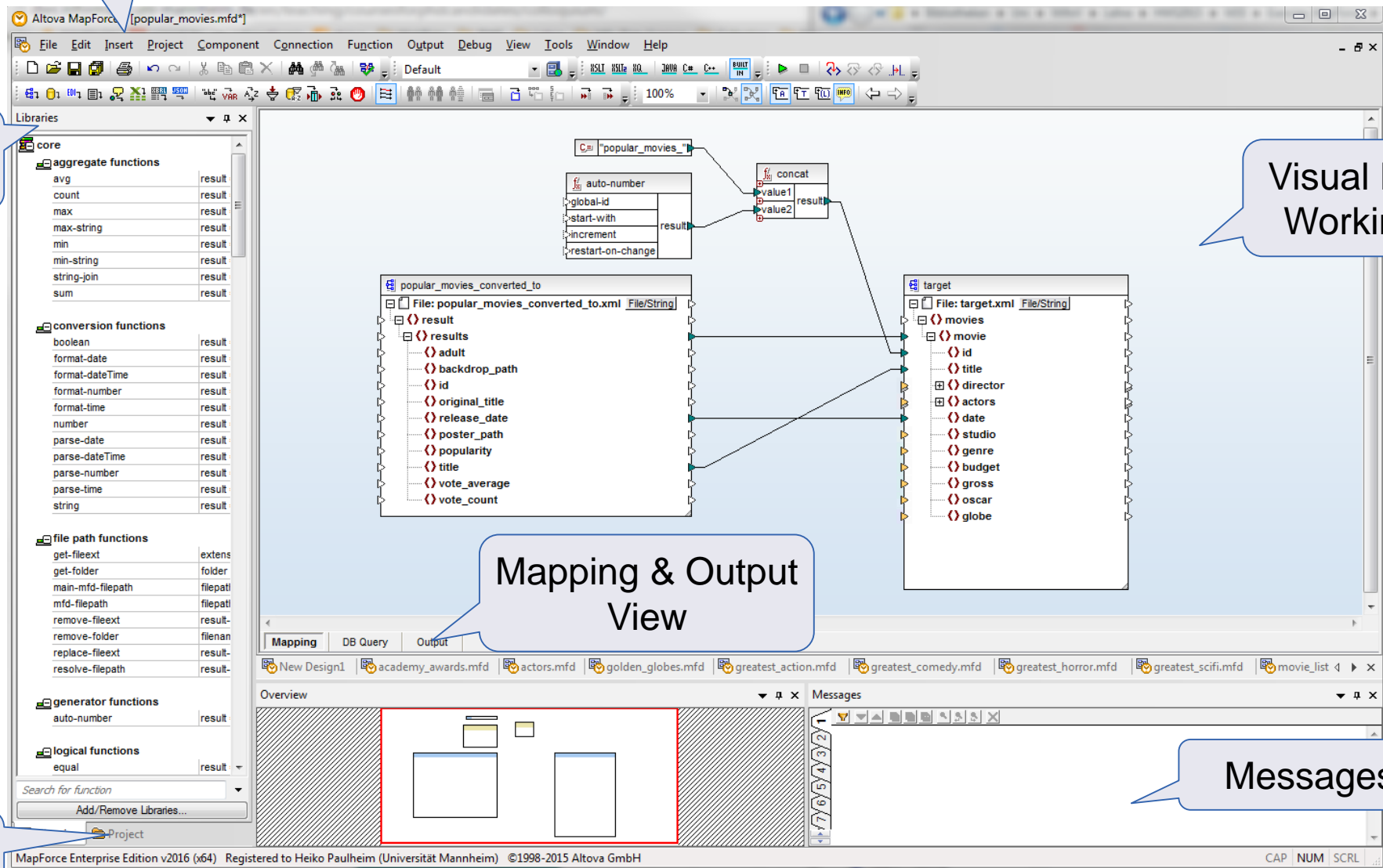
Build-In
Functions

Visual Mapping
Working Area

Mapping & Output
View

Messages/Log

Project
Overview



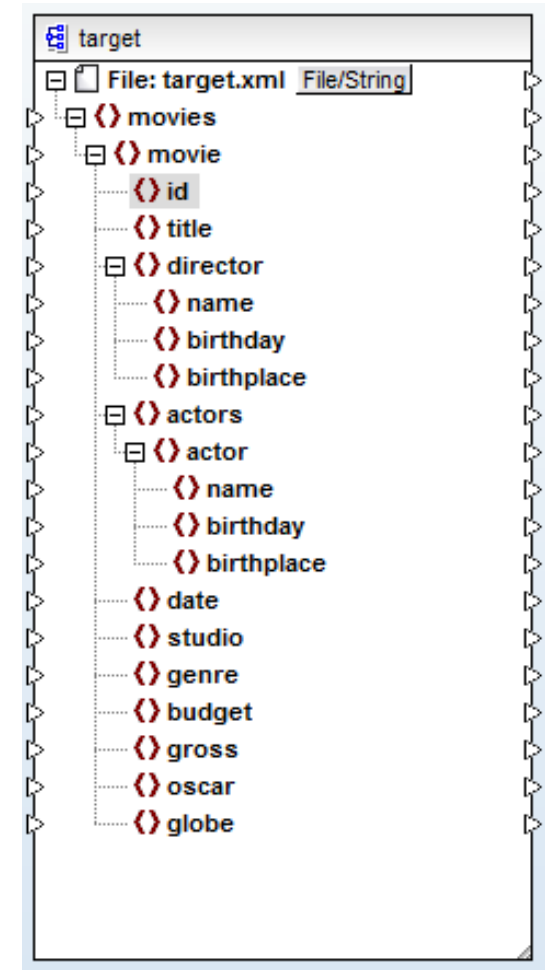
Creating an Integrated Target Schema

- Options:
 - Create XML schema by hand
 - Retrieve the XML schema from a XML file (example)
- The latter is encouraged...
- Example file:

```
<movies>
  <movie id="m1">
    <title>The Shining</title>
    <director>Stanley Kubrick</director>
  </movie>
  <movie id="m2">
    <title>Pi</title>
    <director>Darren Aronofsky</director>
  </movie>
</movies>
```

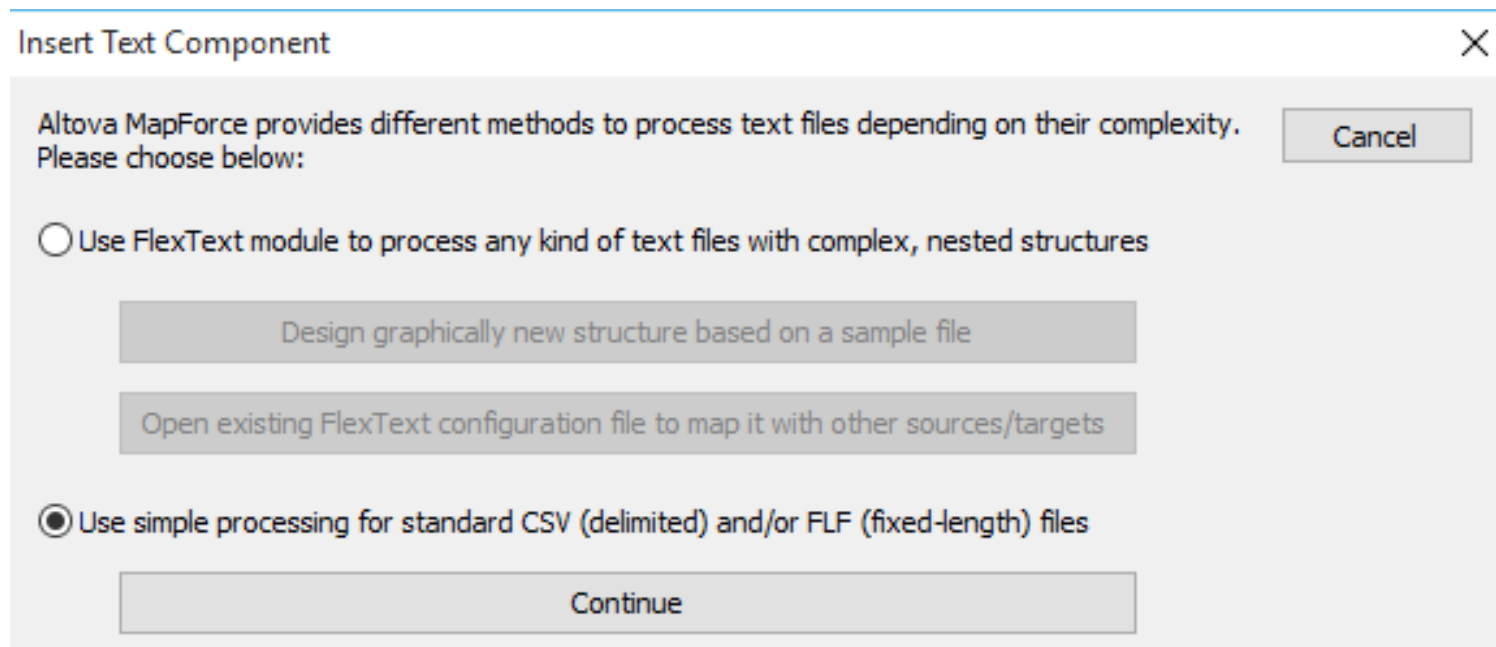
Creating an Integrated Target Schema

1. Open MapForce
2. Choose Insert → XML Schema/File...
3. Let MapForce create a schema for you
4. Edit the schema, if necessary, e.g.
 1. Adjust cardinalities
 2. Change component name
 3. ...



Loading Your Data into MapForce: CSV

1. To import your csv, choose `Insert → Text File...`,
2. **Select** `use simple processing for standard csv`
3. Continue to configure the file importer



Loading Your Data into MapForce: CSV

Component Settings

Component name:

OK Cancel

Input / Output File

☒ Save all file paths relative to MFD file

Input / Output Encoding

Encoding name:

Byte order: ☐ Include byte order mark

CSV Settings

Field delimiter: ☐ Tab ☐ Semicolon ☒ Comma ☐ Space ☐ Custom:

Quote character: ☐ None ☐ ' ☒ "

☒ First row contains field names ☒ Treat empty fields as absent

☒ Add when needed ☐ Add always

☒ CSV ☐ Fixed

Year	Category	Nominee	Additional Info	Won?	F
2010 (83rd)	Actor -- Leading Role	Javier Bardem	Biutiful {'Uxbal'}	NO	
2010 (83rd)	Actor -- Leading Role	Jeff Bridges	True Grit {'Rooster Cogburn'}	NO	
2010 (83rd)	Actor -- Leading Role	Jesse Eisenberg	The Social Network {'Mark Zuckerberg'}	NO	

<<"/>

Name

select file

select encoding

select CSV layout

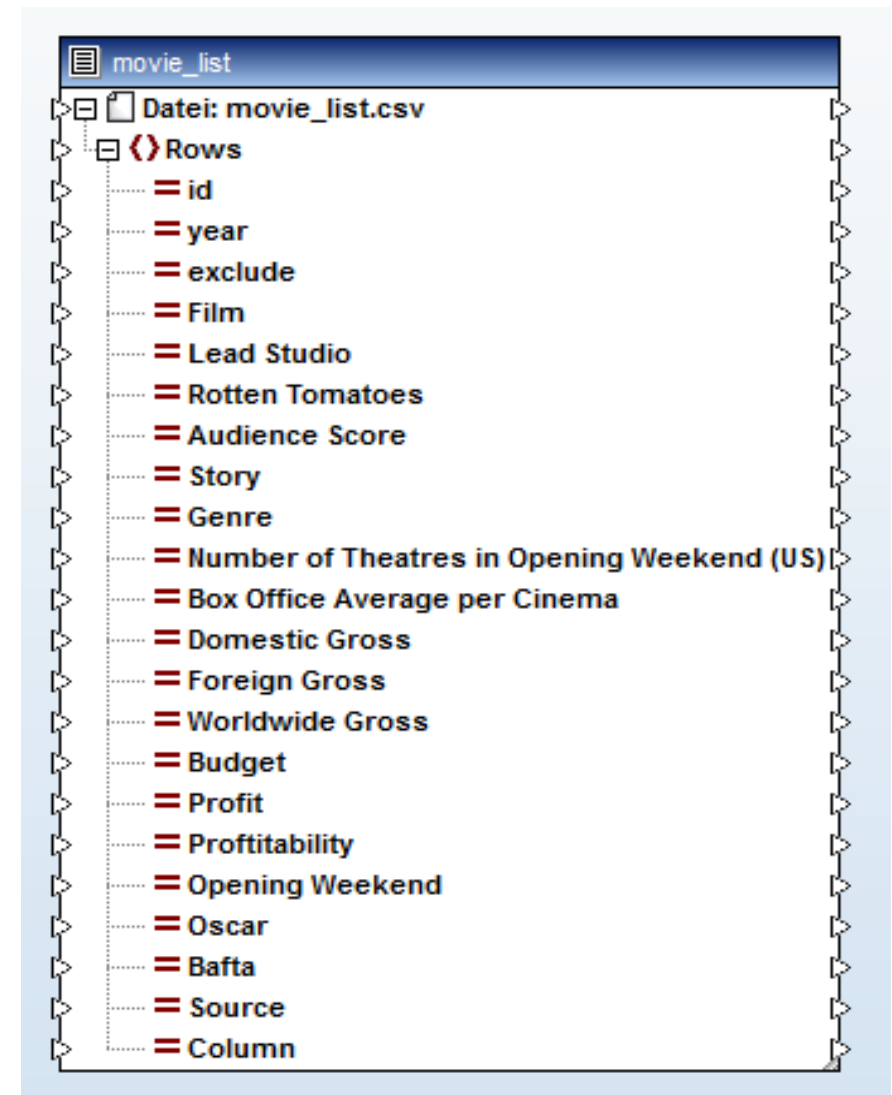
are there headlines?

adjust datatypes

control output

Loading Your Data into MapForce: CSV

1. Review the final output
2. Check for missing attributes or not correctly imported fields



Loading Your Data into MapForce: Excel

1. To import Excel files, choose `Insert` → `Excel 2007+ File...`

MapForce



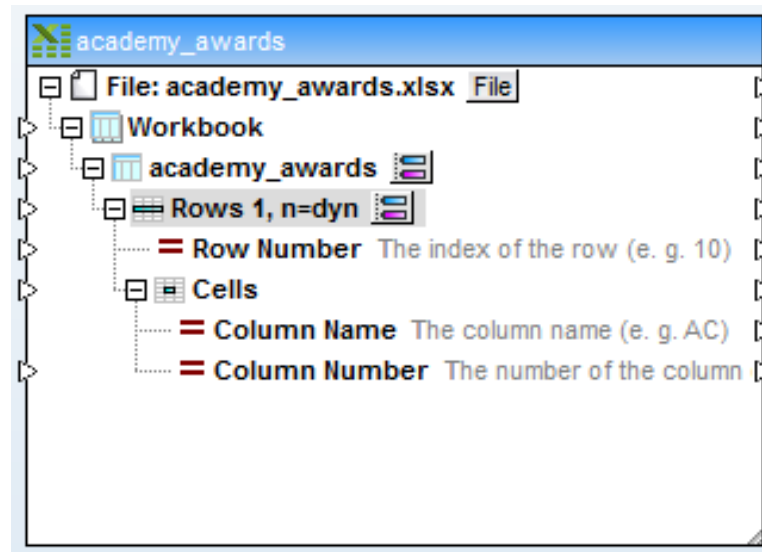
MapForce allows you to define Excel workbooks as source and target. For a source workbook, you might want to provide a sample XLSX file or global resource to preview your transformation.

Do you want to supply a sample XLSX file, a global resource, or not supply any at all?

Browse...

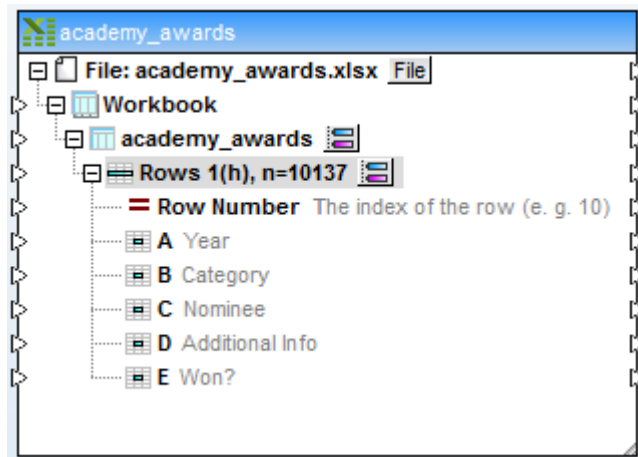
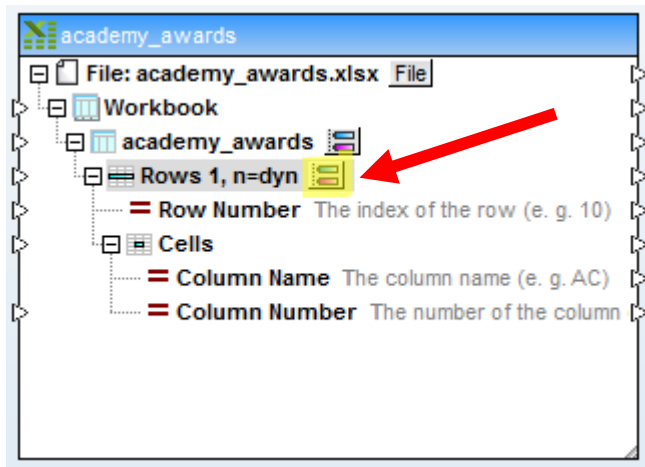
Skip

2. Select `Skip` (if you want an Excel file as output)
3. Select `Browse` (if you want to specify an Excel file as input)



Loading Your Data into MapForce: Excel

– Configure the Import



Select Range of Cells

Load Range from Excel Input File

Select Cell Range to Load... Refresh

Starting Row

☒ Row 1 ☐ Previous range with offset 1

Row Count

☒ Count 10137 ☐ Dynamic

Columns

☐ Show a single Cells item for all columns

Data type: string

☒ Show separate items for columns

Column range: from A to E

☒ First row is header with column names Reload

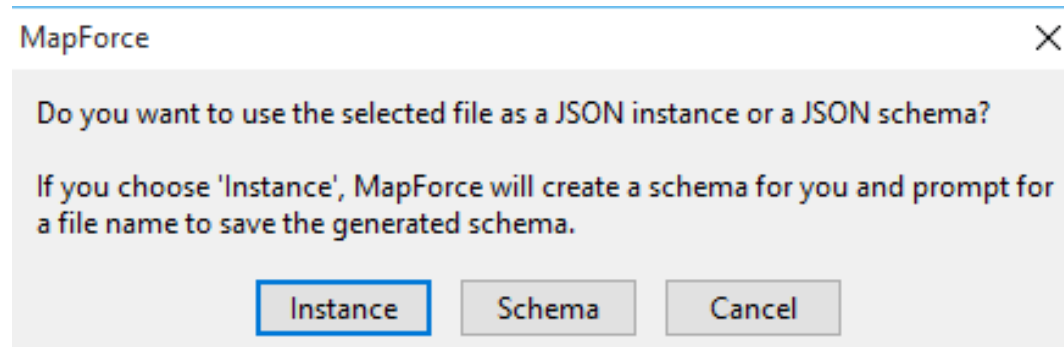
Select types for the cells and enter column names:

	Type	Column Name
A	string	Year
B	string	Category
C	string	Nominee
D	string	Additional Info
E	string	Won?

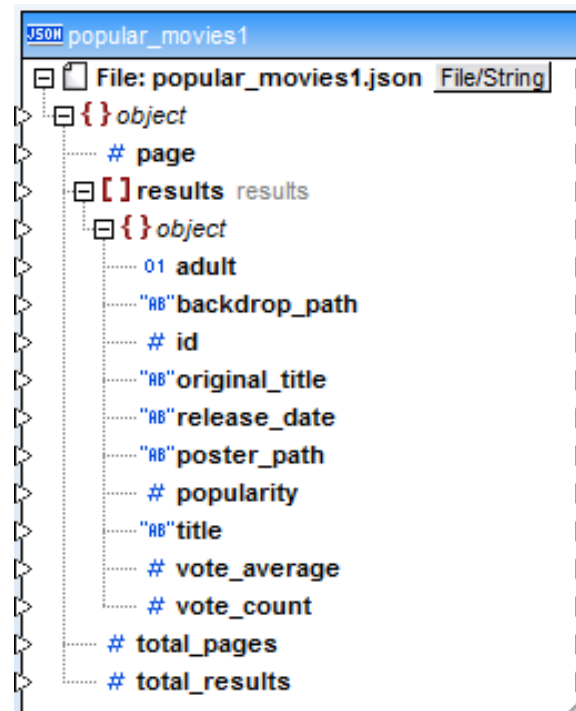
OK Cancel

Loading Your Data into MapForce: JSON

- To import a JSON file, choose Insert → JSON Schema/File...

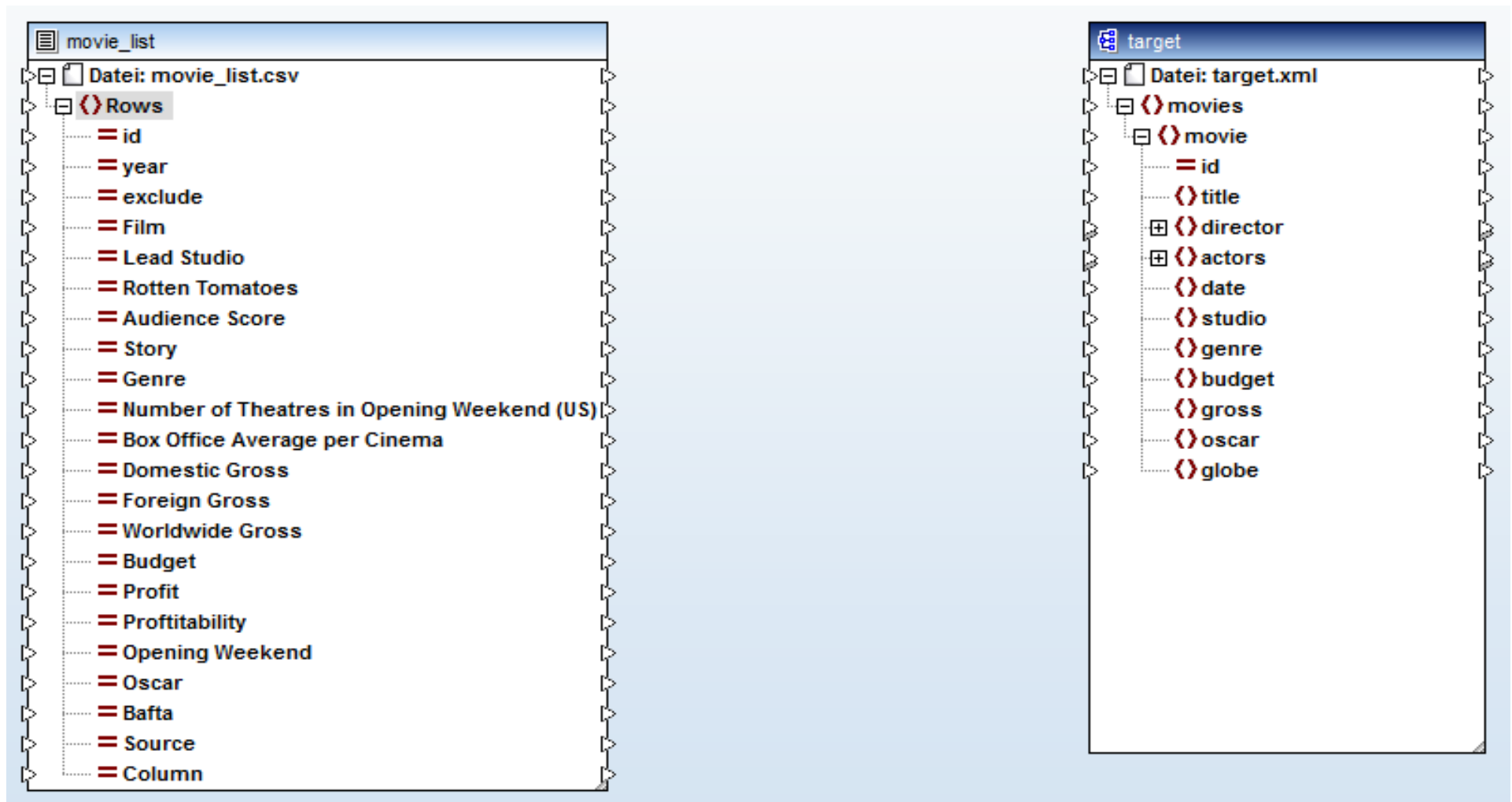


- Choose Instance



Creating Correspondences

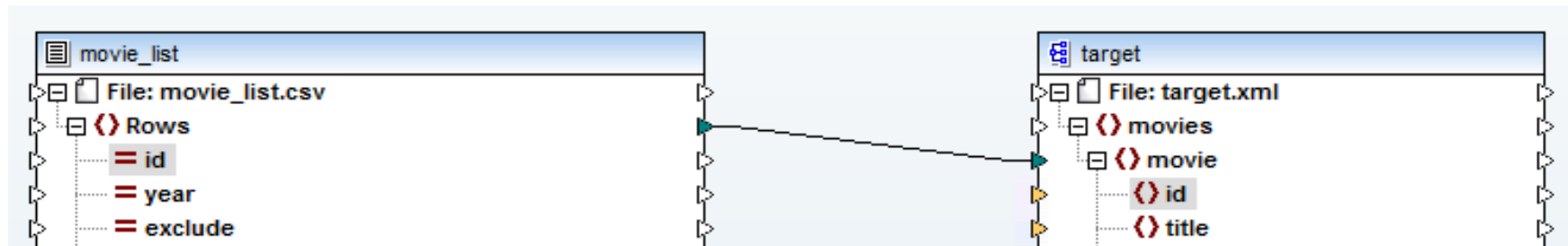
- Now, you have two schemata (source: left, target: right) in your MapForce view



Creating Correspondences

- Map the top level elements
 - each movie instance (row in CSV file) in the input becomes a movie instance in the output

Mapping DB Query Output



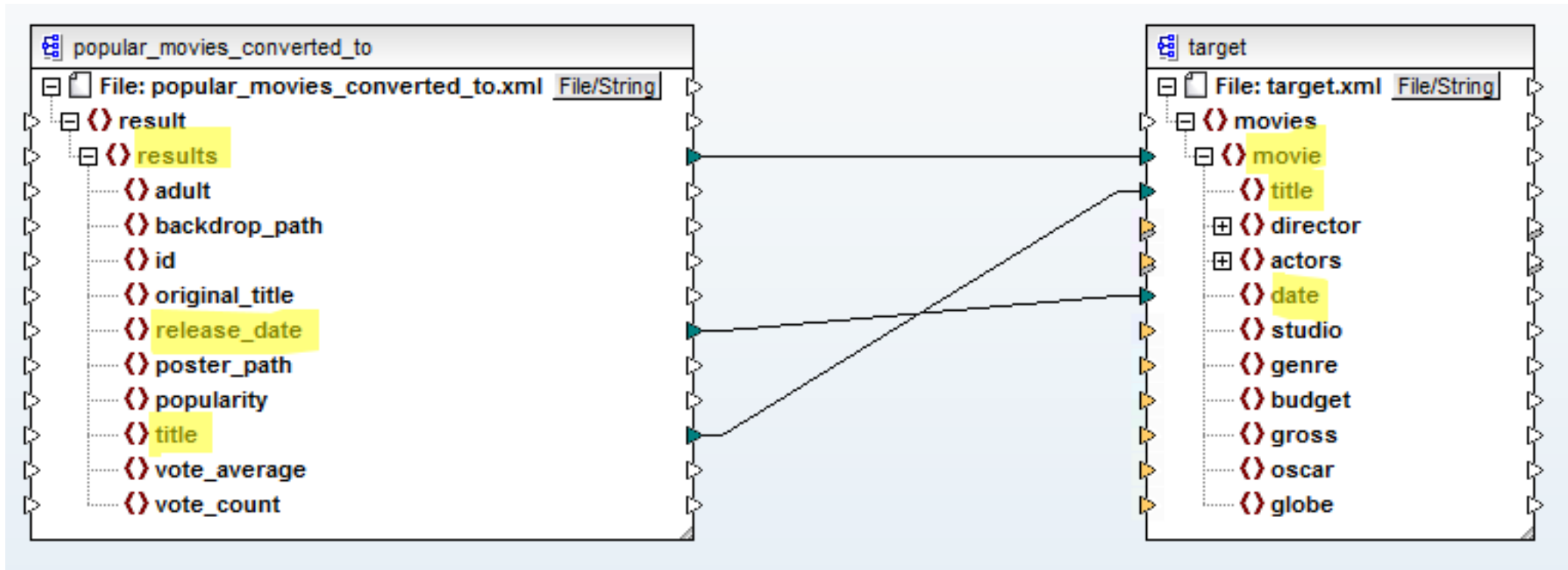
- Output View
 - a list of (still empty) movie elements

Mapping DB Query Output

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <movies xsi:noNamespaceSchemaLocation="C:/Work_Volha/Teaching/HWS201~3/04_EXE~1/examples/movies/input/target.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <movie/>
4   <movie/>
5   <movie/>
6   <movie/>
7   <movie/>
```

Creating Correspondences

- Simple 1:1 correspondences are created by drawing arrows from left to right

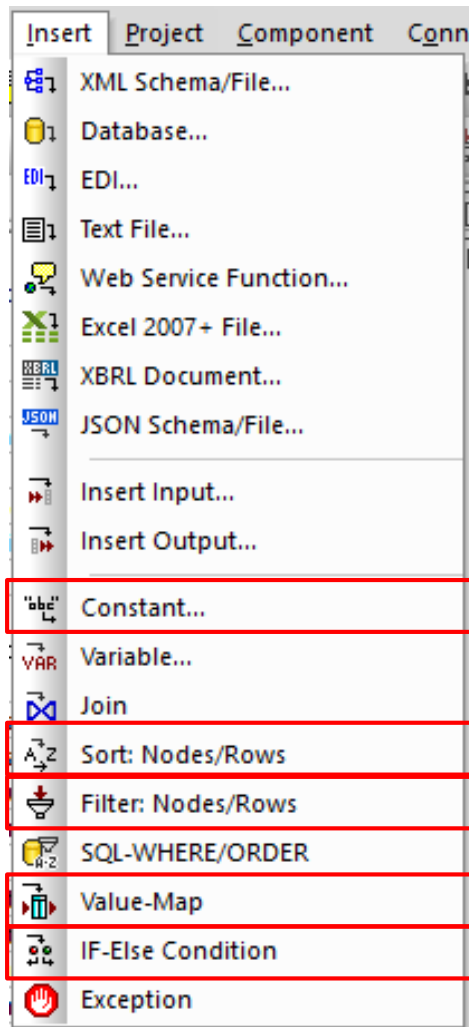


Creating Correspondences

- Simple 1:1 correspondences are created by drawing arrows from left to right

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <movies xsi:noNamespaceSchemaLocation="C:/Users/Oliver/Documents/Lehre/WEBDAT~1
3  <movie>
4      <title>Rififi</title>
5      <date>1955-04-13</date>
6  </movie>
7  <movie>
8      <title>Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb</title>
9      <date>1964-01-29</date>
10 </movie>
11 <movie>
12     <title>The Pink Panther</title>
13     <date>1963-12-19</date>
14 </movie>
15 <movie>
16     <title>For a Few Dollars More</title>
17     <date>1965-12-18</date>
18 </movie>
19 <movie>
20     <title>The Jazz Singer</title>
21     <date>1927-10-06</date>
22 </movie>
23 <movie>
24     <title>The Lady Vanishes</title>
25     <date>1938-11-01</date>
26 </movie>
27 <movie>
28     <title>Lethal Weapon</title>
29     <date>1987-03-05</date>
30 </movie>
31 </movies>
```

Built-in Functions



Built-in functions allow for the creation of more complex mappings:

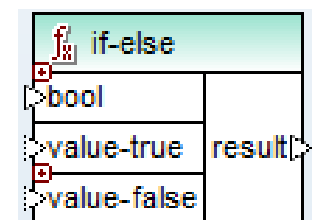
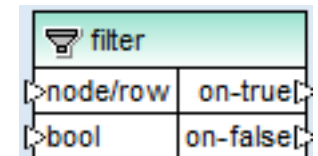
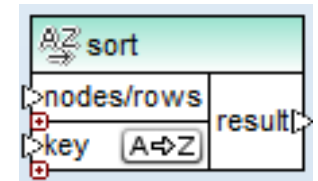
Sort: Sort a set of records by the specified key

Filter: Filter a set of records by the Boolean input for each record

Constant: A constant value of type “String”, “Number” or “All other”

Value-Map: Specify values to be replaced by other values

If-Else: Output values is conditional on Boolean input



Build-In Functions Example: Filter

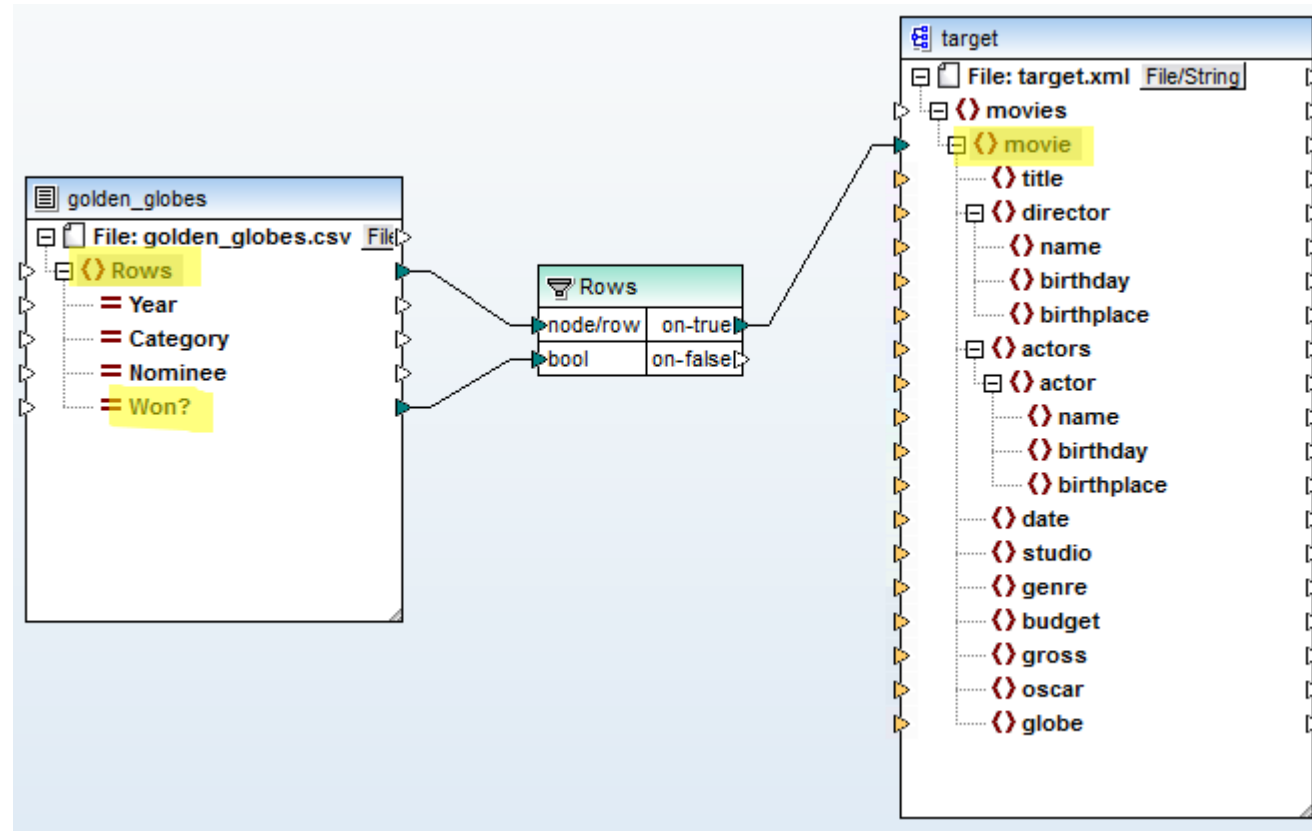
- Goal: Only use data rows that won a Golden Globe

- **Insert Filter**

- Specify which rows to filter as first input
- Condition as second input

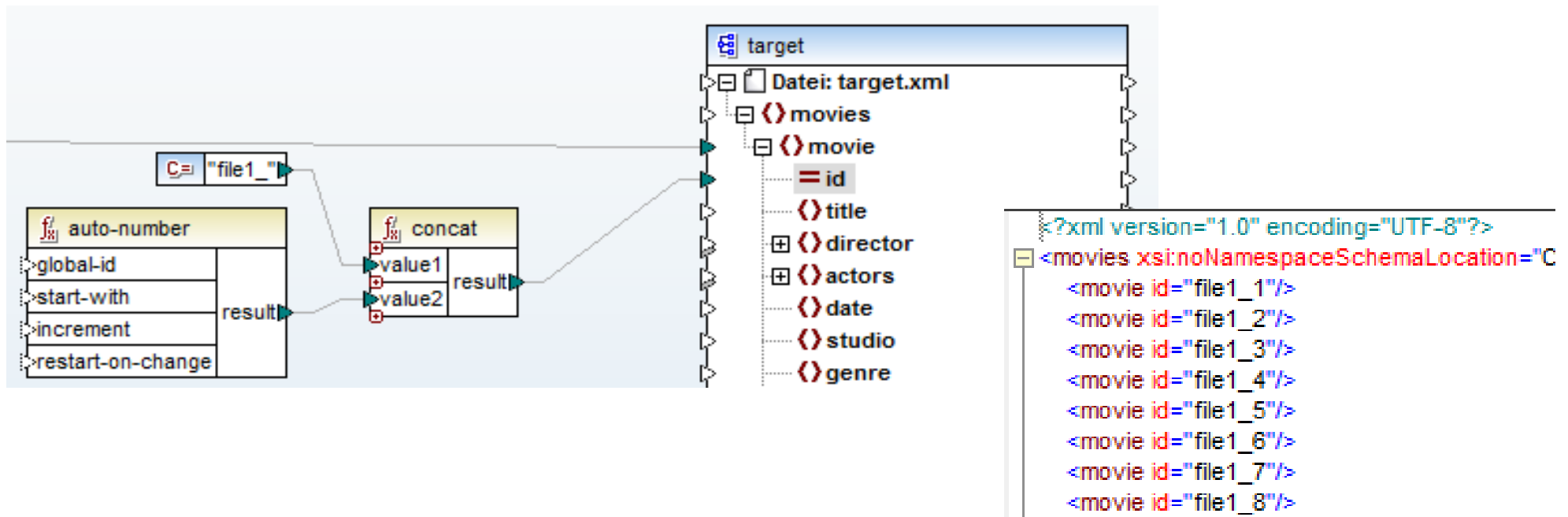
- **Output**

- on-true: All records for which the condition is true
- on-false: All records for which the condition is false



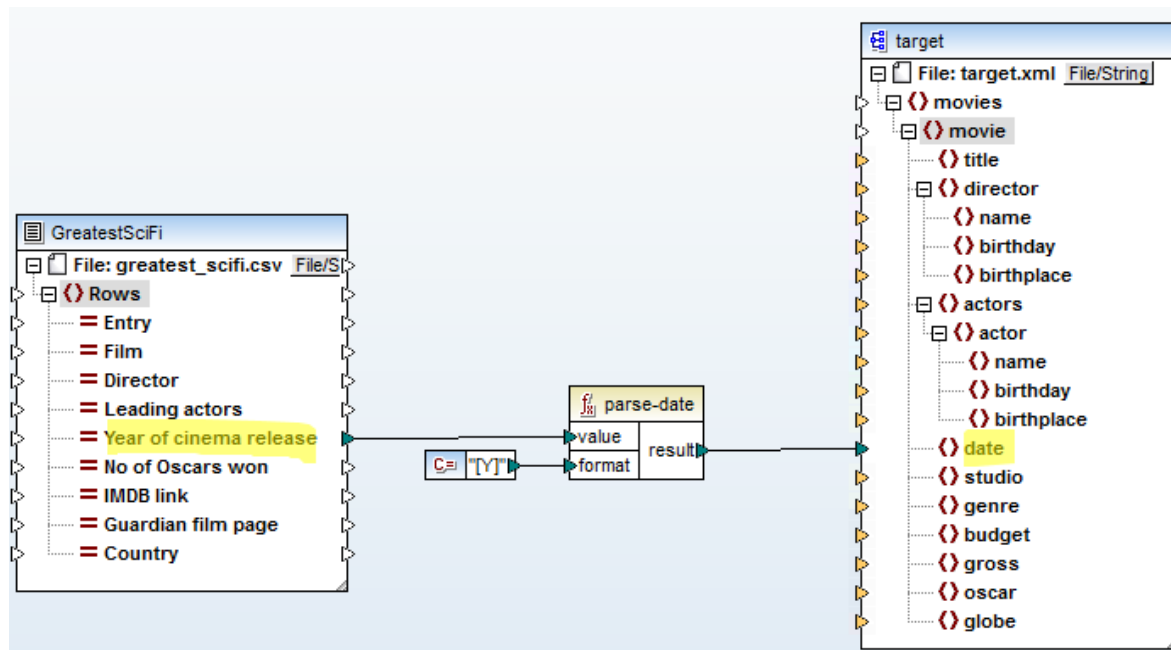
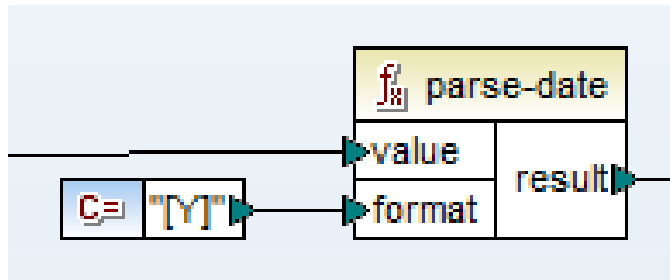
Creating Correspondences

- Generating identifiers (ids) (*obligatory step*)
 - Using the `auto-number` function
 - Caution: your ids should be unique across all generated files
 - Thus: rather use prefix (e.g. file name) + auto-number
 - Using the `concat` function
 - Insert constant with right click → insert constant



Conversions: parse-date / parse-dateTime

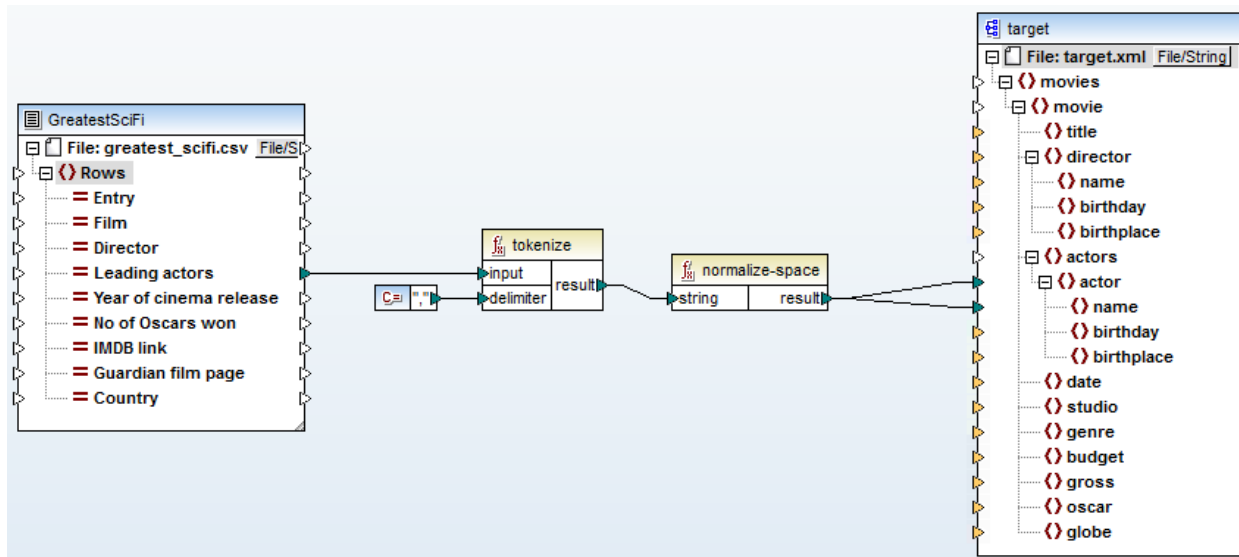
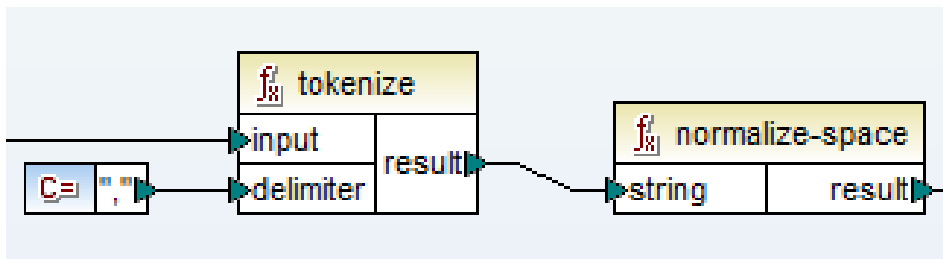
- Converts a string into a date / dateTime
- See also: format-date / format-dateTime



Y	year (absolute value)
M	month of the year
D	day of month
d	day of year
F	day of week
W	week of the year
w	week of month
H	hour (24 hours)
h	hour (12 hour)
P	A.M. or P.M.
m	minutes in hour
s	seconds in minute
f	fractional seconds
Z	timezone as a time offset from UTC
z	timezone as a time offset using GMT

String Functions

- Example: list of actors in string with commas
 - `normalize-space` removes leading and trailing space

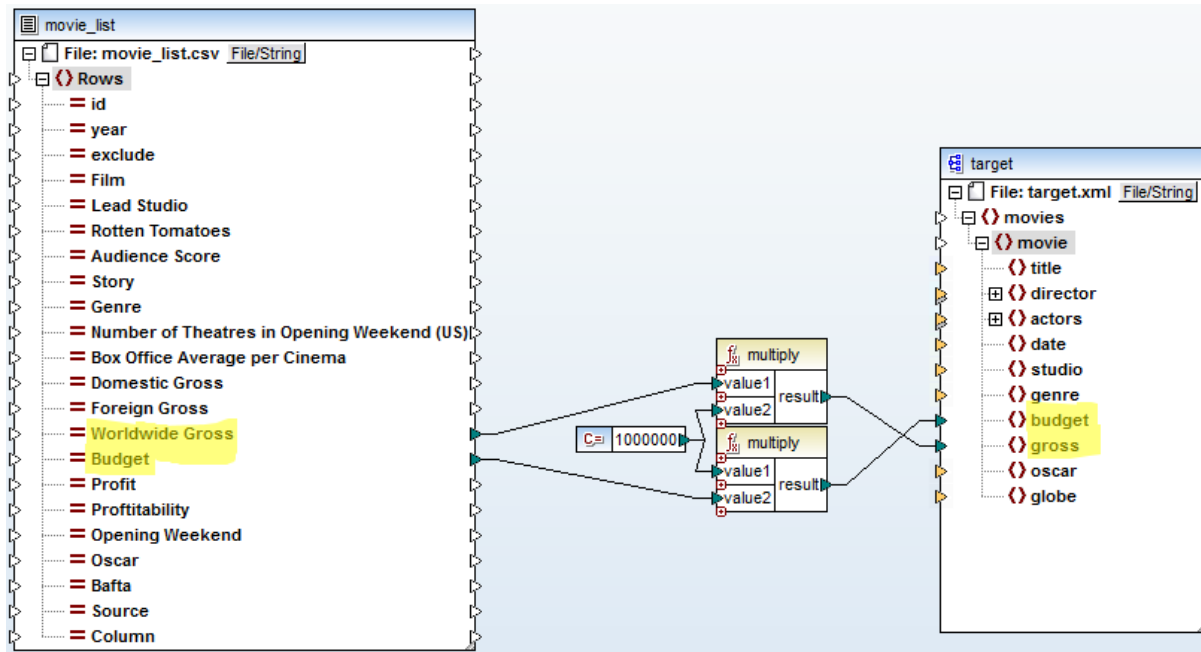
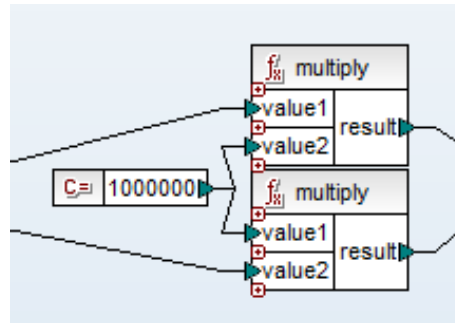


string functions	
char-from-code	result = char-from-code(value)
code-from-char	result = code-from-char(value)
concat	result = concat(string1, string2)
contains	result = contains(value, substring)
normalize-space	result = normalize-space(string)
starts-with	result = starts-with(string, substr)
string-length	result = string-length(string)
substring	result = substring(string, start [,length])
substring-after	result = substring-after(string, substr)
substring-before	result = substring-before(string, substr)
tokenize	result = tokenize(input, pattern)
tokenize-by-length	result = tokenize-by-length(input, length)
tokenize-regexp	result = tokenize-regexp(input, pattern, flags)
translate	result = translate(value, string1, string2)

string functions	
capitalize	result = capitalize(value)
count-substring	result = count-substring(string, substr)
empty	result = empty(value)
find-substring	result = find-substring(string, substr [,startindex])
format-guid-string	formatted_guid = format-guid-string(unformatted_guid)
left	result = left(string, number)
left-trim	result = left-trim(string)
lowercase	result = lowercase(string)
match-pattern	result = match-pattern(string, substr)
pad-string-left	result = pad-string-left(string, final-length, pad-string)
pad-string-right	result = pad-string-right(string, final-length, pad-string)
repeat-string	result = repeat-string(string, count)
replace	result = replace(value, oldstring, newstring)
reversefind-substring	result = reversefind-substring(string, substr [,startindex])
right	result = right(string, number)
right-trim	result = right-trim(string)
string-compare	result = string-compare(string1, string2)
string-compare-ignore-case	result = string-compare-ignore-case(string1, string2)
uppercase	result = uppercase(string)

Mathematical Functions

- Example: Multiply original values with a constant-value

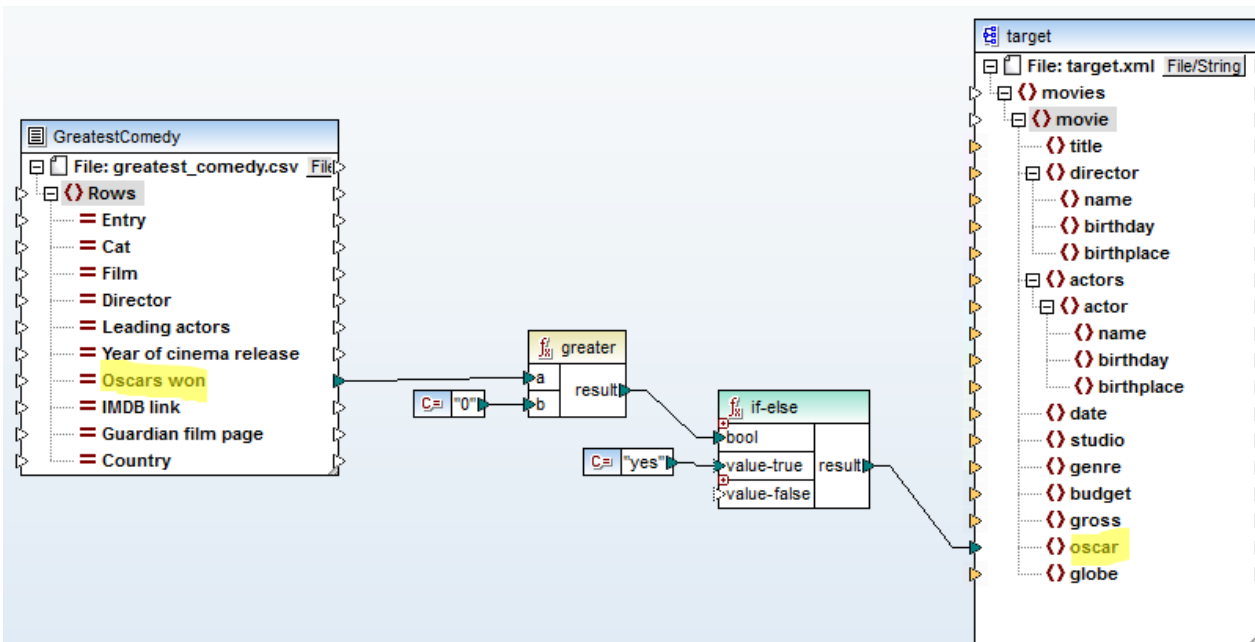
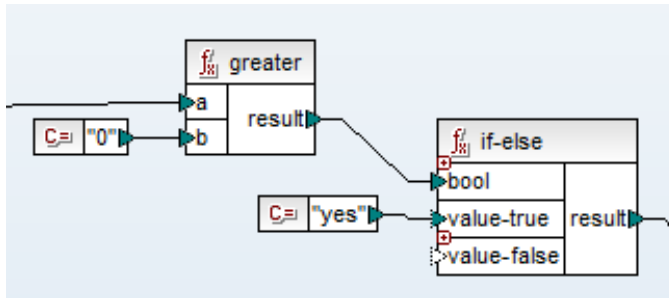


math functions	
add	result = value1 + value2
ceiling	result = ceiling(value)
divide	result = value1 / value2
floor	result = floor(value)
modulus	result = value1 mod value2
multiply	result = value1 * value2
round	result = round(value)
round-precision	result = round-precision(value, decimals)
subtract	result = value1 - value2

math functions	
abs	result = abs(value)
acos	result = acos(value)
asin	result = asin(value)
atan	result = atan(value)
cos	result = cos(value)
degrees	result = degrees(value)
divide-integer	result = value1 div value2
exp	result = exp(value)
log	result = log(value)
log10	result = log10(value)
max	result = max(value1, value2)
min	result = min(value1, value2)
pi	result = pi()
pow	result = a ^ b
radians	result = radians(value)
random	result = random()
sin	result = sin(value)
sqrt	result = sqrt(value)
tan	result = tan(value)
unary-minus	result = -value

Logical Functions

- Example: Transform integer to Boolean value

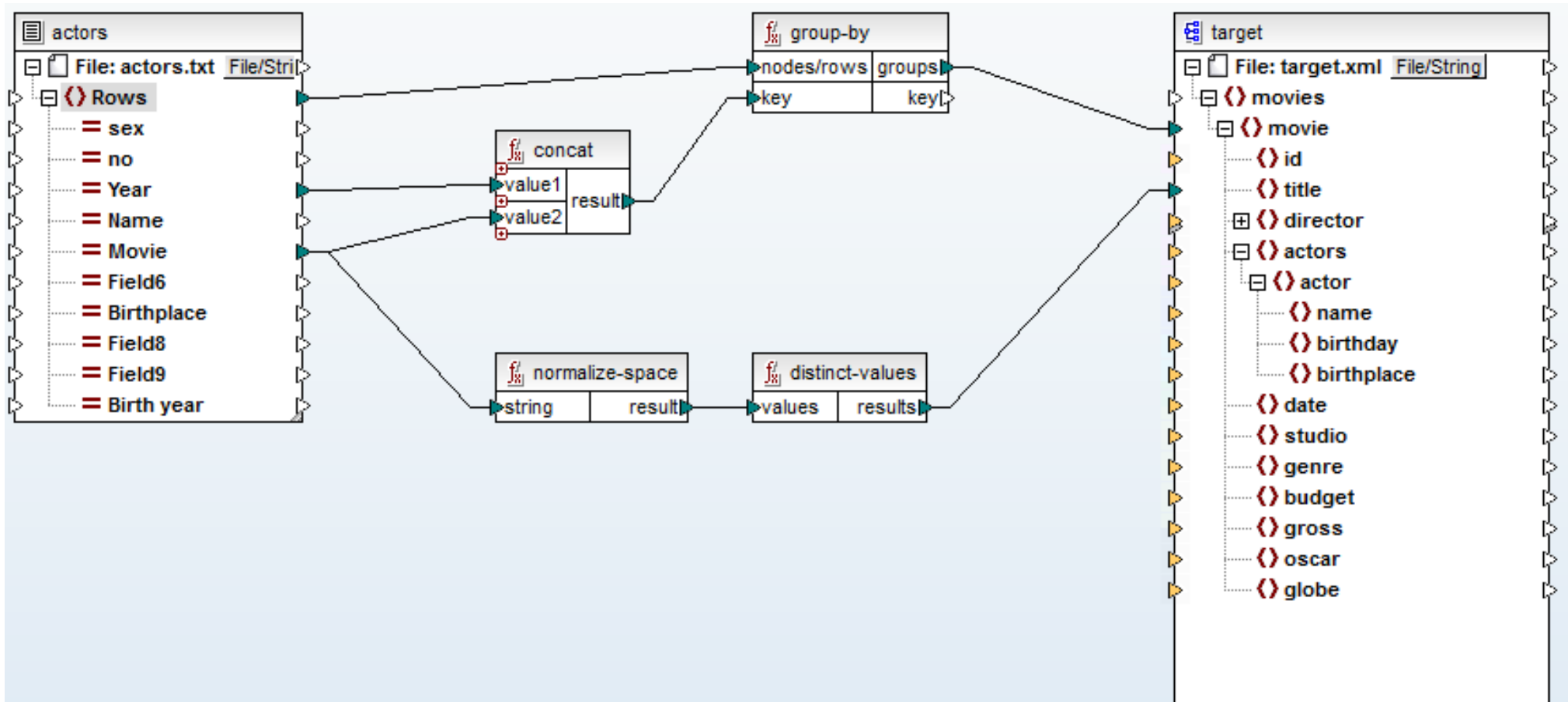


logical functions	
equal	result = a equal b
equal-or-greater	result = a >= b
equal-or-less	result = a <= b
greater	result = a > b
less	result = a < b
logical-and	result = logical-and(value1, value2)
logical-not	result = logical-not(value)
logical-or	result = logical-or(value1, value2)
not-equal	result = logical-not(a equal b)

logical functions	
logical-xor	result = logical-xor(value1, value2)
negative	result = value < 0
numeric	result = numeric(value)
positive	result = value >= 0

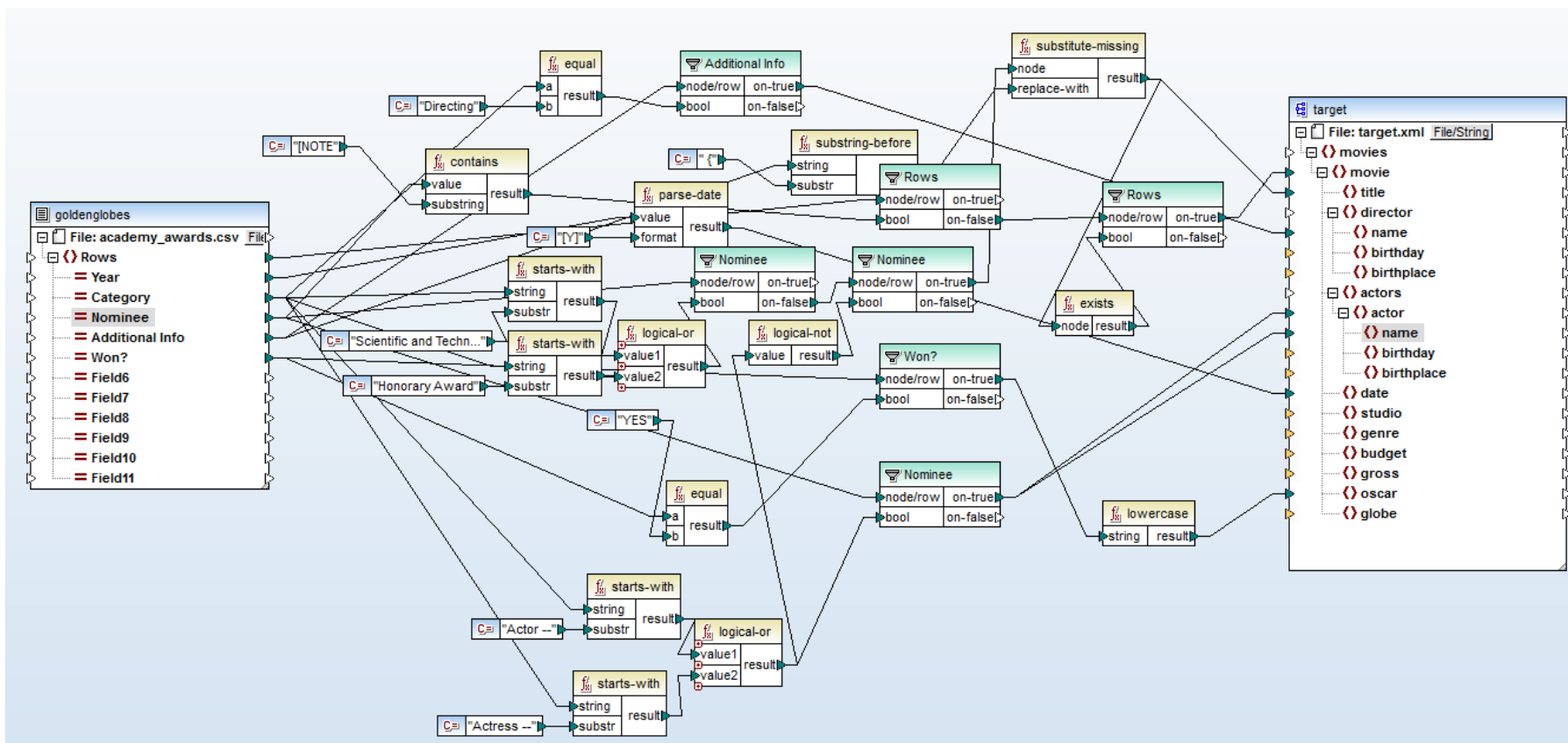
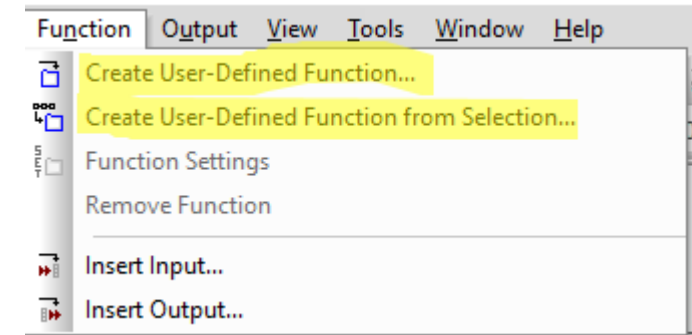
Sequence Functions

- Aggregate Actors to Movies
 - Input: one line per actor/movie combination
 - Output: one node per movie



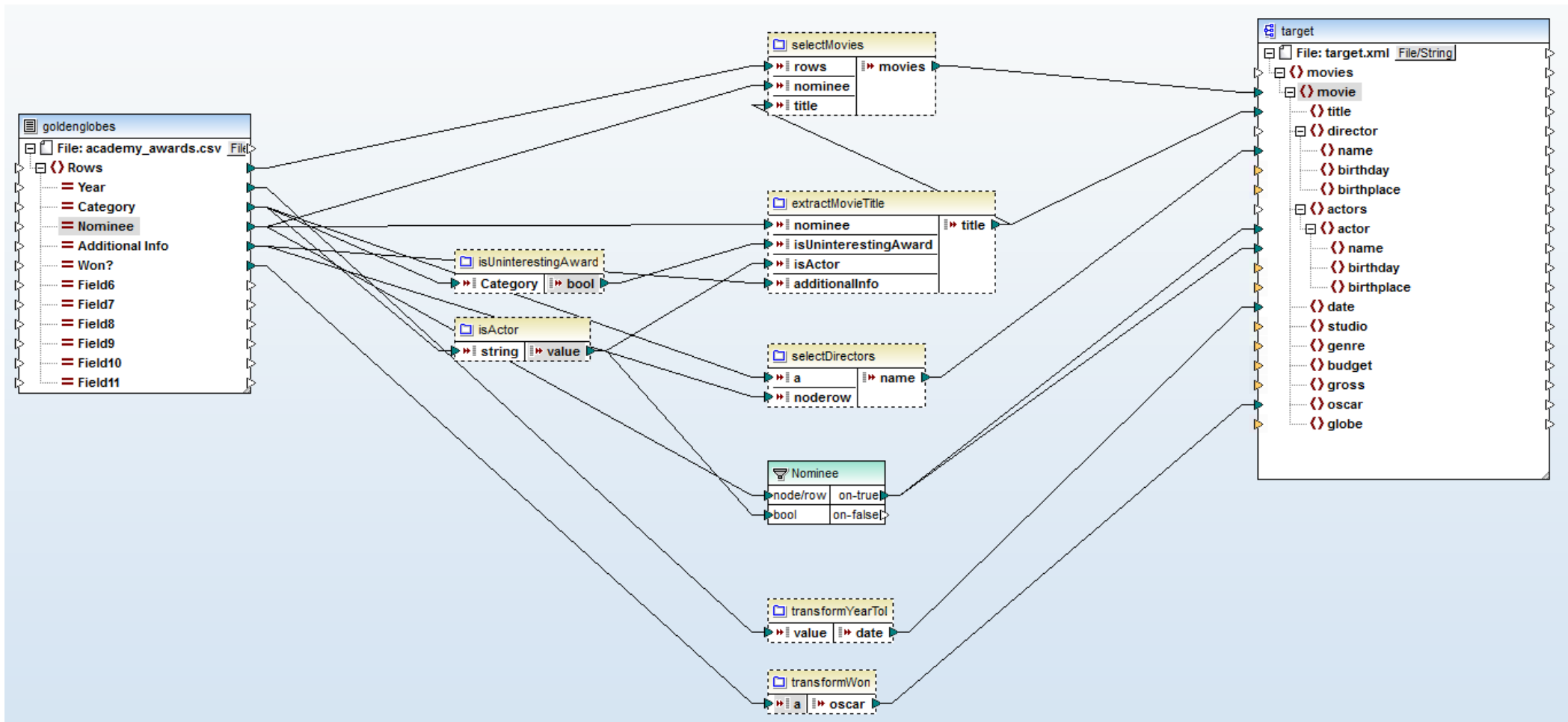
User-Defined Functions

- The mapping view quickly becomes confusing



User-defined Functions

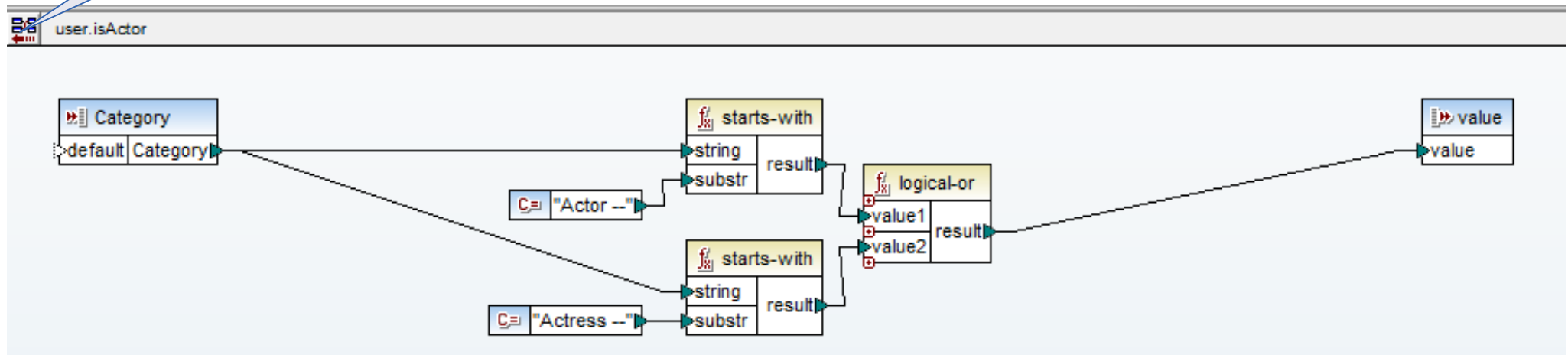
- Make use of User-Defined Functions (UDF) to organize your functions



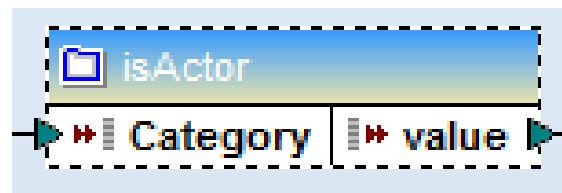
User-Defined Functions

- A UDF maps any number of input parameters to output parameters

Click to return to the overall mapping

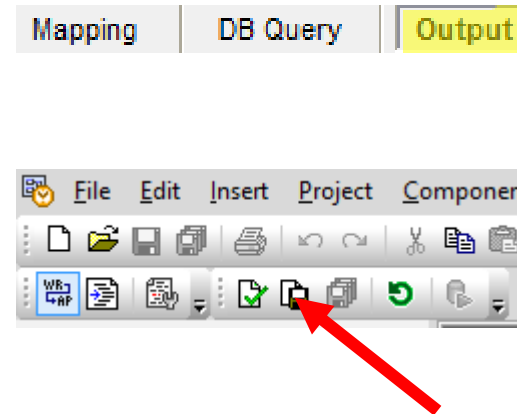


- It is represented by a single function in the mapping view



Get the Translated Data Out

- Go to the Output View
- Select Save generated output



Hands-on: Schema Mapping with MapForce

- Your task: Create mappings between different datasets from the movie domain using MapForce.
- Data
 - target.xsd: An XML schema file providing the target schema
 - movie_list.csv: Dataset describing movies with 22 features such as year, story and genre stored as csv
 - actors.csv: Dataset describing actors with 10 different features such as name, movie in which they acted and birthplace stored as a tab separated file

Hands-on: Schema Mapping with MapForce - Movies

1. Load files

- movie_list.csv as input
- target.xml as target schema

2. Assign IDs

- The id should start with the prefix *movie_list_id* followed by an increasing number which starts at 1000.

Increment: **auto-number** function

Add prefix to incrementing number: **concat** function

Foreign Gross: **if** the value is numeric add it **else** add **0**

Date: Add prefix, define the **format**, parse the concat values as date

3. Create attribute correspondences from **source** to **target**

- Map **Film** and **Genre** to the corresponding attributes of the target schema
- Set the **gross** attribute value as a summation of the **Domestic** and the **Foreign Gross**. (*hint*: You can only perform summation if all the involved values are numeric)
- Map **year** to **date** and set the day and month to the first of June.

Check if exclude value **equals** y. On **false** map the source row to the target movie

4. Filter

- Exclude the movies where the exclude attribute is set to y

Hands-on: Schema Mapping with MapForce - Actors

1. Load files

- actors.csv as input
- target.xml as target schema

The input file is **tab** separated. Define the **datatype** and **names** of the attributes.

2. Aggregate by Movie

- Identify an appropriate mapping key (the name of the movie is not enough as there might be movies with the same title)
- Group by the created key. Use the key as id for the target dataset.
- Map the aggregated rows to the movies in the target schema

Use **normalize space** function to remove the spaces. **Concatenate** the birthday day, month, year in a parsable **format** like **[D].[M].[Y]**

3. Create actor correspondences

- Create the missing correspondences for the name, birthplace and the full birthday
- In case the values contain spaces, remove them

File Format:

Gender	No	MovieYear	ActorName	Title	Reviews	BirthPlace	BirthMonth	BirthDay	BirthYear
--------	----	-----------	-----------	-------	---------	------------	------------	----------	-----------

...and now

1. Collect your data
2. Profile your data
3. Generate your target schema
4. Convert all your data into the integrated schema using MapForce

