

Modern Data Engineering in the Cloud

Dr. Christian Dollfus
Dozent

T direkt +41 41 228 22 54
christian.dollfus@hslu.ch

Luzern

Dr. Pavlin Mavrodiev
Dozent

T direkt +41 76 733 61 66
pavlin.mavrodiev@alumni.ethz.ch

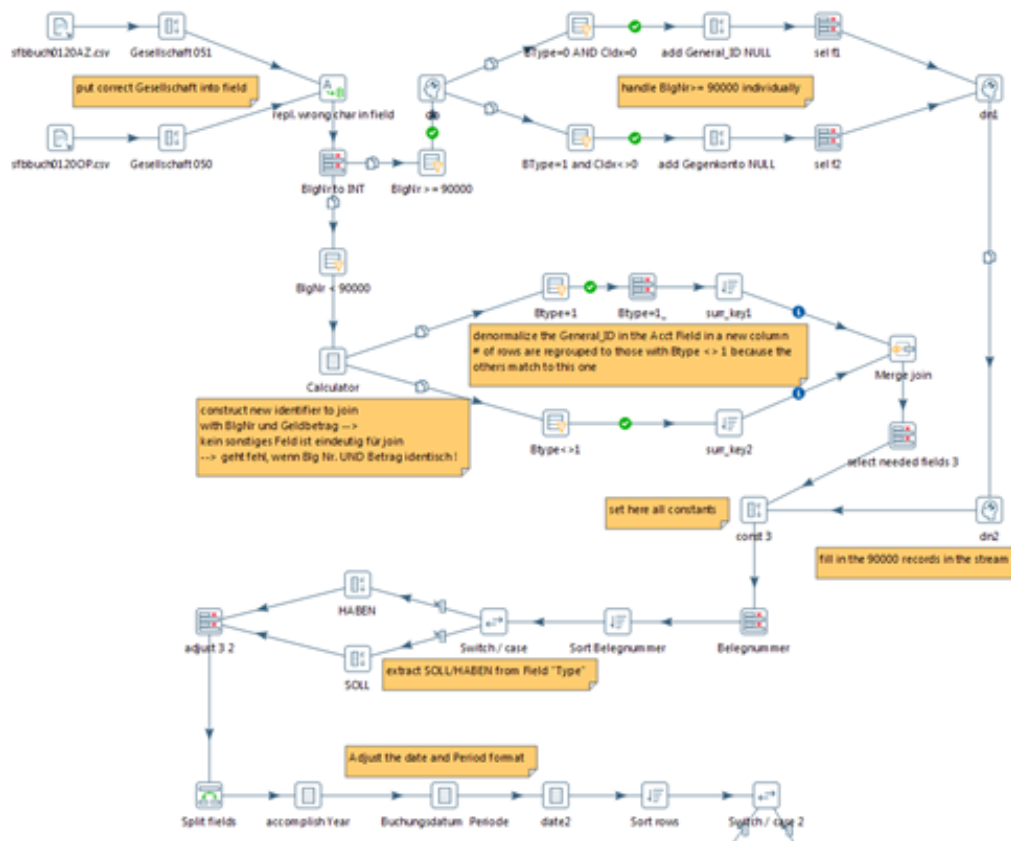
PART 3 : Data Engineering with the ETL/ELT -Tool PDI introduction

Data Engineering with PDI

- Introductory words
- What is Data Engineering?
- Motivation and Value Proposition - a historic overview
- How does Data Engineering look like in many companies?
- What are features and advantages of workflow-based ETL?
- Motivation for the use of ETL tools.
- Data Engineering and Business Process Automation - similarities and differences

- Processing Data is a consecutive Workflow of tasks that needs to be run; either continuously or in batch (every hour, every day, every week, ..)
- Data Processes can be more complex than a “linear” series of tasks: split into 2 tasks, performing a “loop” and reenter a stream :

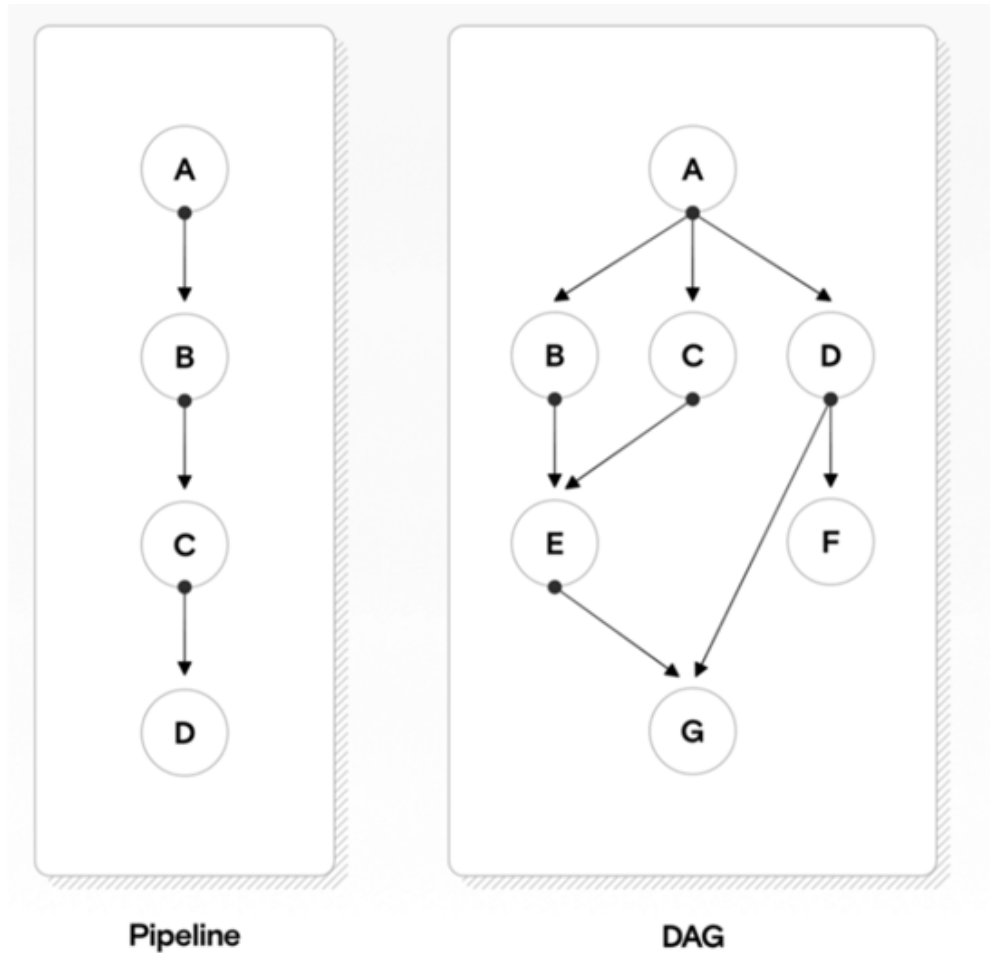
Directed Acyclic graph: DAG



Workflow orchestration tools allow you to define DAGs by specifying all of your tasks and how they depend on each other. The tool then executes these tasks on schedule, in the correct order, retrying any that fail before running the next ones. It also monitors the progress and notifies your team when failures happen.

Data Engineering with PDI

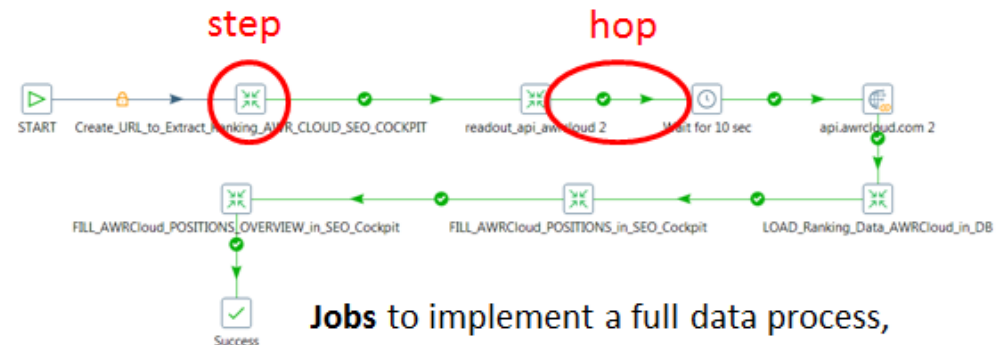
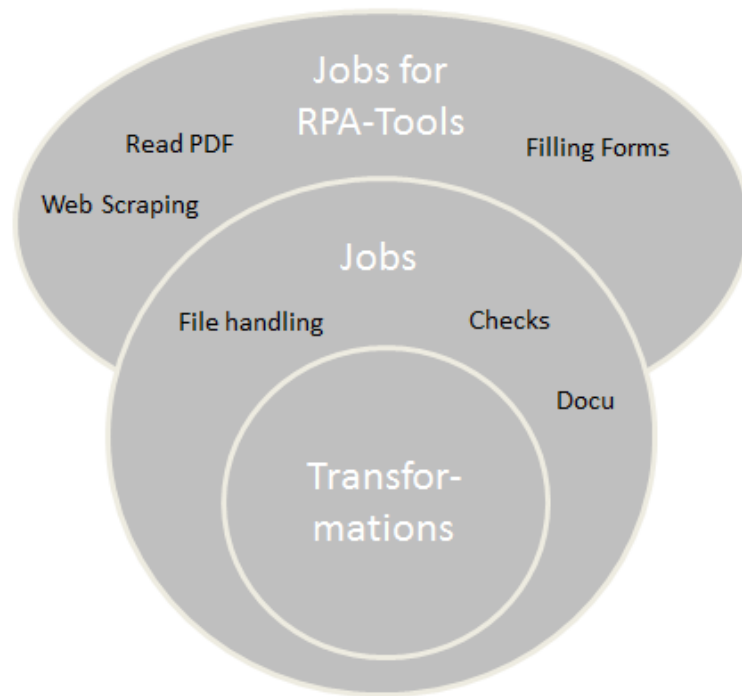
Overall, the focus of any orchestration tool is ensuring **centralized, repeatable, reproducible, and efficient** workflows: a virtual command center for all of your automated tasks.



Source: <https://towardsdatascience.com/airflow-vs-luigi-vs-argo-vs-mflow-vs-kubeflow-b3785dd1ed0c>

[Markus Schmitt](#)

Data Engineering with PDI



Jobs to implement a full data process, jobs collect a couple of functions (transformations) or other sub-jobs. They are the surrounding functions of a data pipeline: filehandling, downloading data, Webservice interfaces a.s.o.

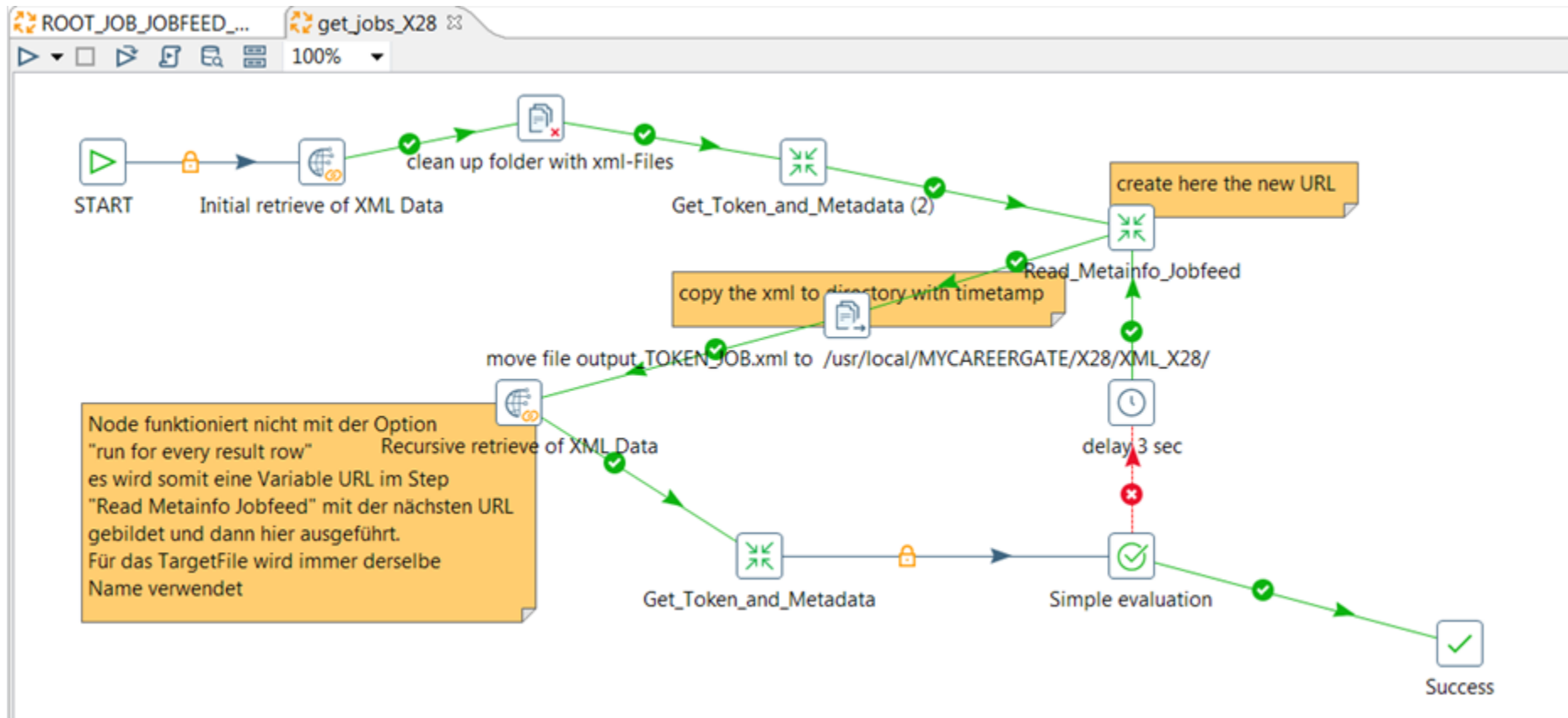


Transformation is a directed graph with logical tasks and can be built as a network. The elements are the STEPS.

Transformations are *dataflows* in the essence

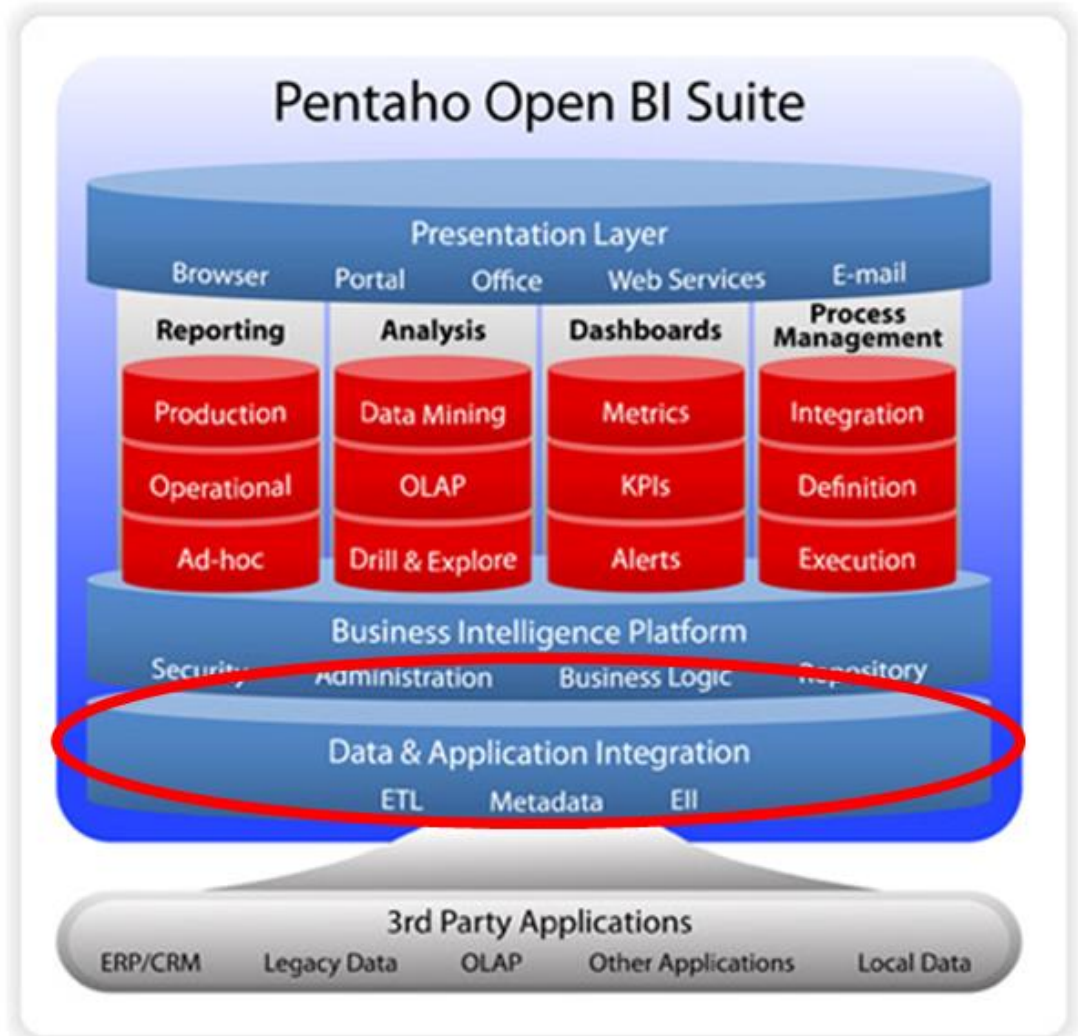
<https://help.pentaho.com/Documentation/7.0/OL0/OY0/O30/O10>

Data Engineering with PDI



Data Engineering with PDI

We use only one part of the
Whole Hitachi
Pentaho/Lumada suite:
The Pentaho Data
Integration part (former
«KETTLE»)



INTRODUCTION to ETL/ELT

1) Jobs and Transformations

There is an [Apache Software Foundation incubation](#) process starting soon, using a FORK of PDI:

A lot has changed behind the scenes, but don't worry, if you're familiar with Kettle/PDI, you'll feel right at home immediately

<https://hop.apache.org/manual/latest/>

<https://hop.apache.org/tech-manual/latest/hop-vs-kettle/index.html>

APANCHE HOP: <https://hop.apache.org/>



IMPORT KETTLE (PDI) PROJECTS IN APACHE HOP (INCUBATING)

Data Engineering with PDI

Help and Hints in Pentaho

Comprehensive Documentation
On the Web from Hitachi Vantara

<https://help.pentaho.com/Documentation/9.0>

Forum for Pentaho user

<https://forums.pentaho.com/forums/135-Pentaho-Data-Integration-Kettle/>

Comprehensive collection of
examples of functions, Jobs und
Transformations

INSTALL_DIRECTORY\NANE_PDI_DIRECTORY\data-
integration\samples\transformations

D:\Programme\pdi-ce-9.0.0.0.0-4323\data-
integration\samples\transformations

REGEXP Tester

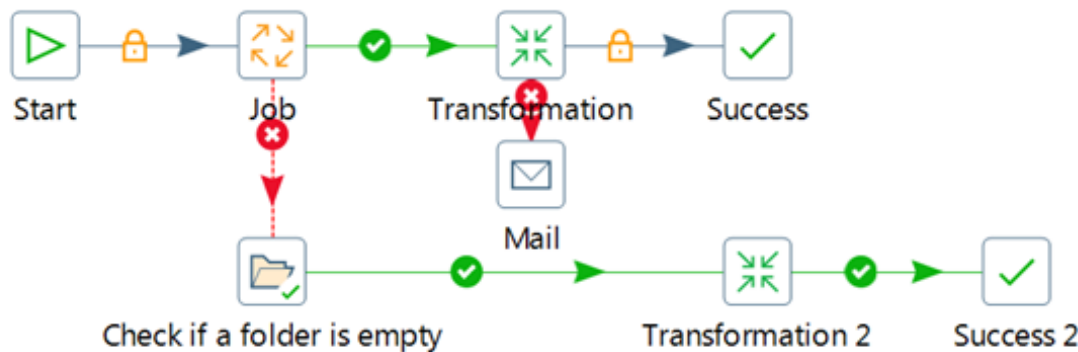
<https://regex101.com/>

Other ressources

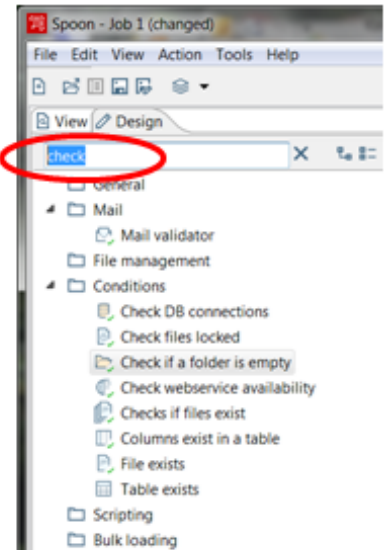
<https://etl-tools.info/pentaho-tutorial.html>

Data Engineering with PDI

Jobs and Transformations can be NESTED hierarchically (but not transformations in themselves): A Job contains several transformations and jobs (subjob)



Kann Begriff suchen



On Hobs in Jobs you can include a conditional logic of the pipeline flow:

Möglichkeit	Beschreibung
Bedingungslos	Gibt an, dass der nächste Jobeintrag unabhängig vom Ergebnis des ursprünglichen Jobeintrags ausgeführt wird
Folgen Sie, wenn das Ergebnis wahr ist	Gibt an, dass der nächste Jobeintrag nur ausgeführt wird, wenn das Ergebnis des ursprünglichen Jobeintrags wahr ist. Dies bedeutet eine erfolgreiche Ausführung wie "Datei gefunden", "Tabelle gefunden", ohne Fehler usw.
Folgen Sie, wenn das Ergebnis falsch ist	Gibt an, dass der nächste Jobeintrag nur ausgeführt wird, wenn das Ergebnis des ursprünglichen Jobeintrags falsch war. Dies bedeutet, dass die Ausführung nicht erfolgreich war, die Datei nicht gefunden wurde, die Tabelle nicht gefunden wurde, Fehler aufgetreten sind usw.

Data Engineering with PDI

Transformations

Jobs

Jobs

INPUT Steps

Transformations

TRANSF Steps

Transformations

OUTPUT Steps

Transformations

The following slides are giving a structured overview over the different nodes in PDI
grouped in this structure above:



denote important steps



denote VIP's

INTRODUCTION to ETL/ELT

Transformations

2.1) Handling INPUT in Data Pipelines:

Flat Files: CSV
Excel
XML
JSON

CSV file input
Text file input
GZIP CSV input
Property input

Microsoft Excel input

Get data from XML
XML input stream (StAX)

JSON input
Email messages input

SQL-Tables

Table input
Database lookup

Webservices

REST client
Web services lookup
HTTP post
HTTP client
RSS input

BigData / NoSQL
and Realtime Streaming

MongoDB input
HBase input
Cassandra input
Avro input
Hadoop file input
Kafka Consumer
MapReduce input
ORC input
Parquet input
YAML input

Special Input of Vendor

Google Spreadsheet Input
Google Analytics
HL7 input
LDIF input
Microsoft Access input
Mondrian input
OLAP input
S3 CSV input
SAS input
Salesforce input
XBase input
CouchDB input

Bulk Loader

MonetDB bulk loader
Elasticsearch bulk insert
Ingres VectorWise bulk loader
Greenplum load
MySQL bulk loader
Infobright loader
Oracle bulk loader
PostgreSQL bulk loader
Teradata Fastload bulk loader
Teradata TPT bulk loader
Vertica bulk loader

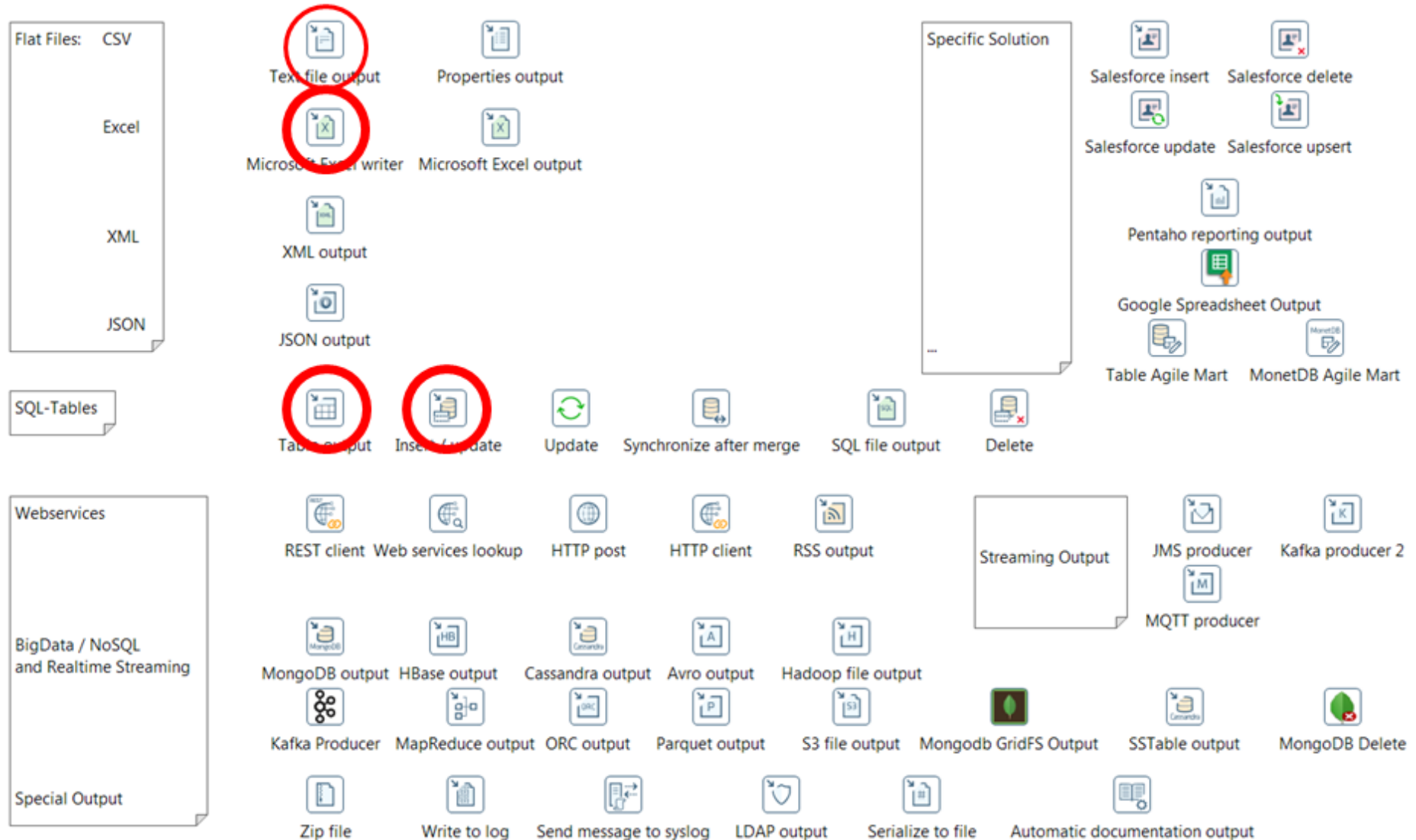
Streaming Input

Get records from stream
JMS consumer
MQTT consumer
Kafka consumer 2

INTRODUCTION to ETL/ELT

Transformations

2.2) Handling OUTPUT in Data Pipelines:

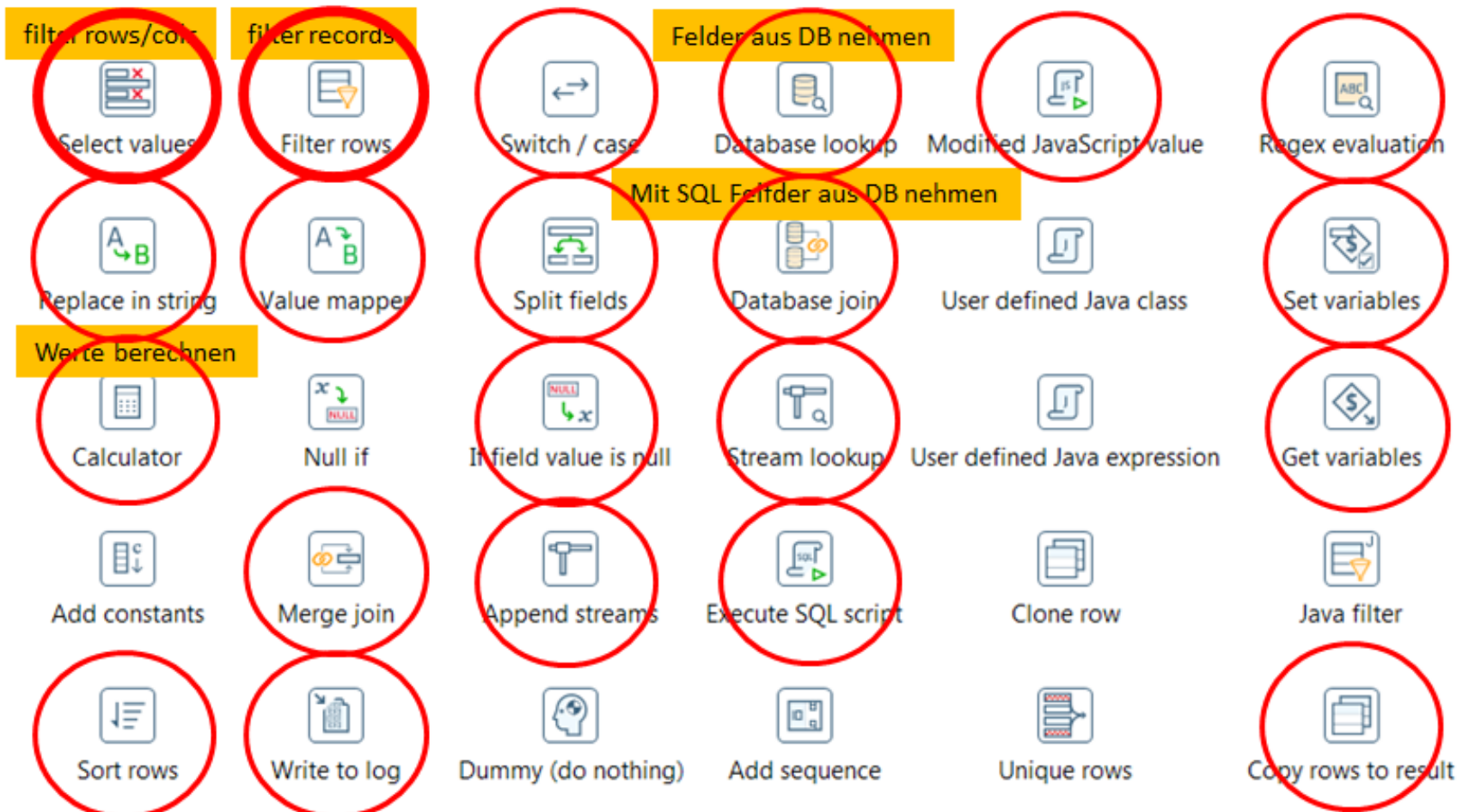


INTRODUCTION to ETL/ELT

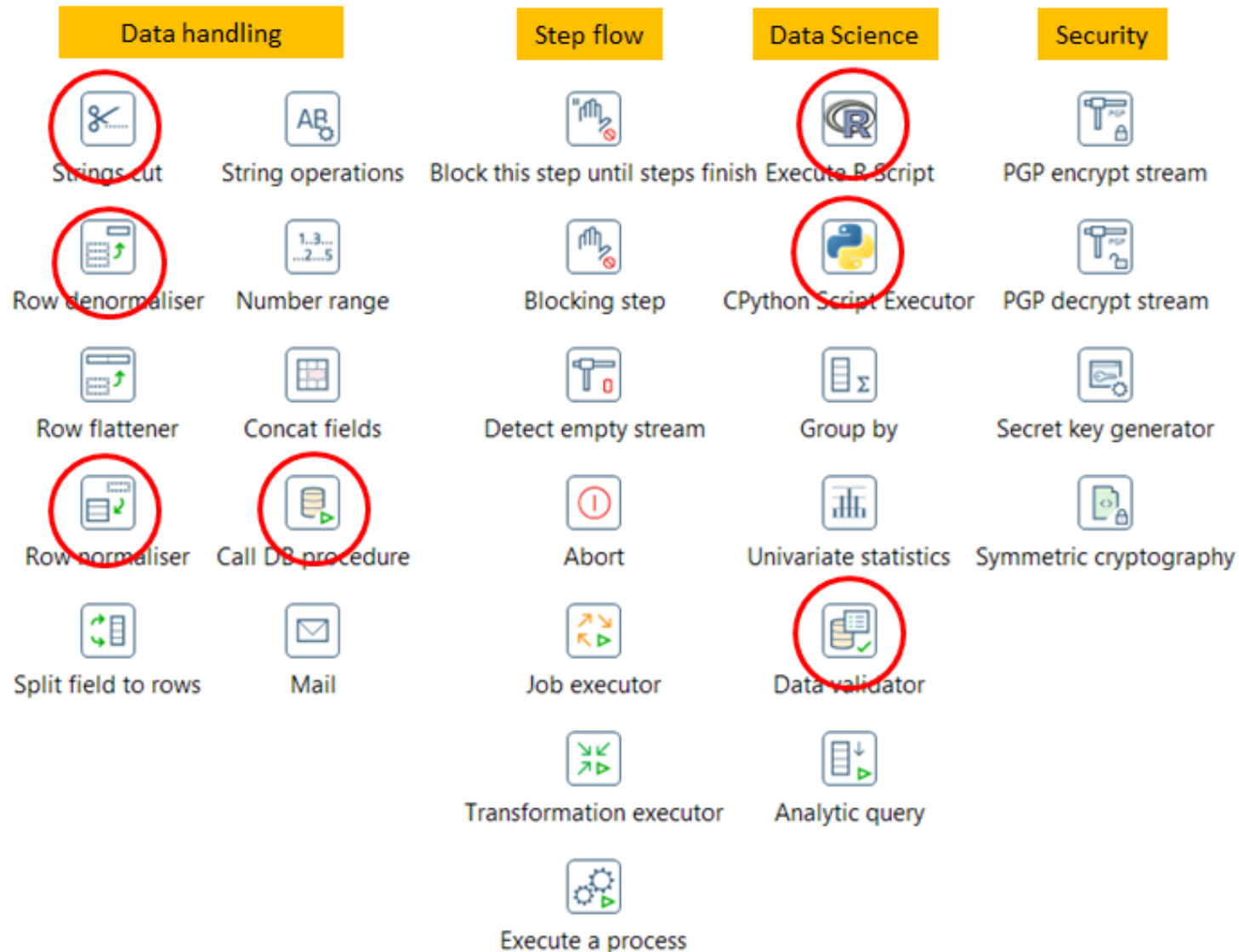
Transformations

3.1) Basic TRANSFORMATION functions in Data Pipelines:

Lookups (stream, DB), Joins, Sort, String handling, Filtering (fields/records), types, split fields, denormalization, normalization and much more



3.2) More TRANSFORMATION functions in Data Pipelines:



INTRODUCTION to ETL/ELT

Jobs

3.3) Basic JOB functions in Data Pipelines:

Basics

VIS

File handling

Checks



Start



Set variables



Add filenames to result



Create file



File compare



Get a file with FTP



Check DB connections



Check if XML file is well formed



Success



HTTP



Create a folder



Delete file



Move files



Get a file with SFTP



Table exists



DTD validator



Job



Mail



Check if a folder is empty



Copy files



Unzip file



Get a file with FTPS



Columns exist in a table



XSD validator



Transformation



Dummy



Delete folders



Add filenames to result 2



Zip file



Put a file with FTP



Evaluate rows number in a table



XSL transformation



Abort job



Compare folders



Delete filenames from result



Wait for file



Put a file with SFTP



Check webservice availability



Evaluate files metrics



Wait for



Write to file



Upload files to FTPS



Checks if files exist



Simple evaluation



Write to log


























FTP delete

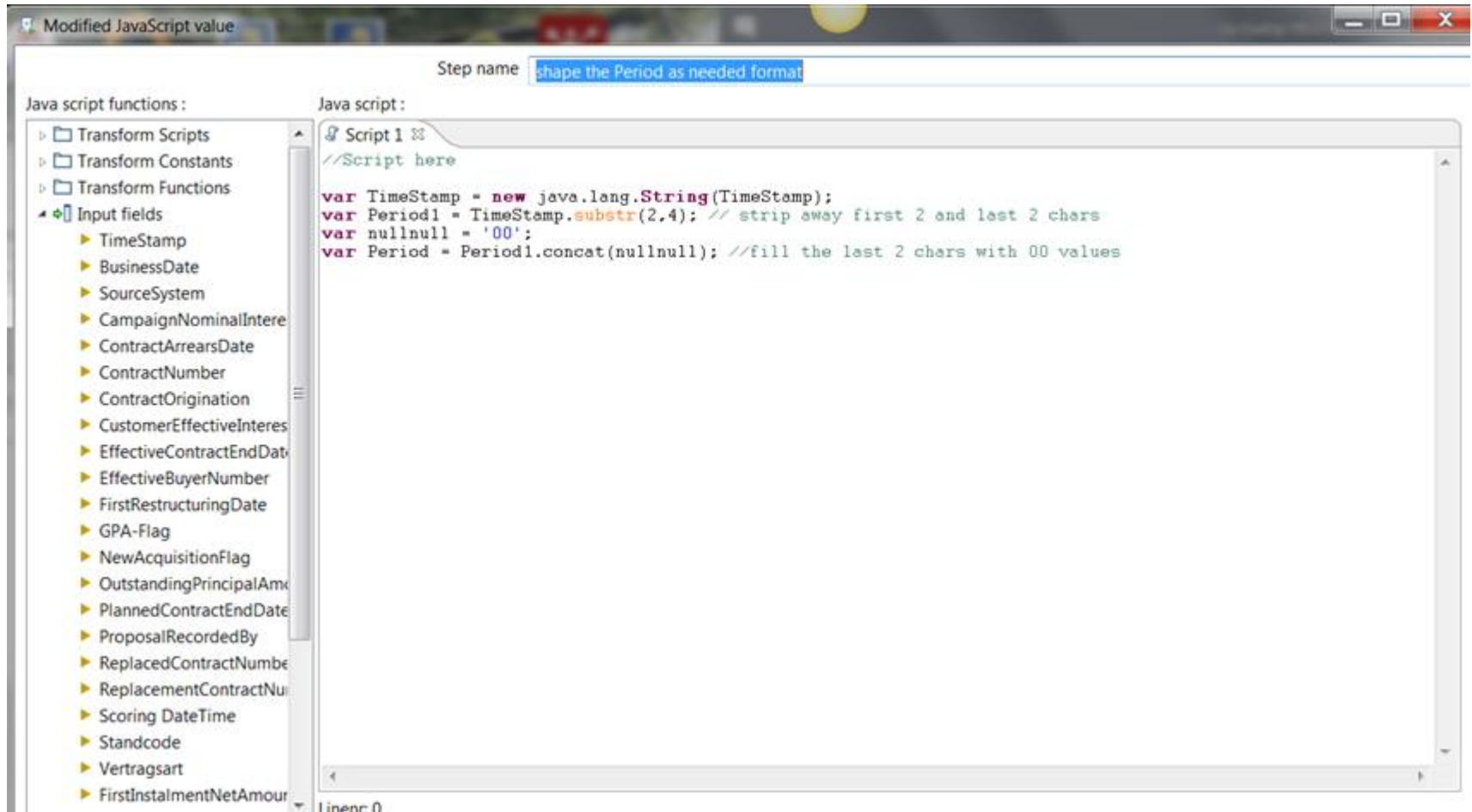


Check files locked

3.4) More JOB functions in Data Pipelines:

Big Data	Scripts	Security	Utilities		Bulk load
 Hadoop copy files	 JavaScript	 Decrypt files with PGP	 Ping a host	 Get mails (POP3/IMAP)	 Bulk load from MySQL into file
 Hadoop job executor	 Shell	 Encrypt files with PGP	 Telnet a host	 Mail validator	 Bulk load into MSSQL
 Spark submit	 SQL	 Verify file signature with PGP		 Truncate tables	 Bulk load into MySQL
 Sqoop export				 Wait for SQL	
 Sqoop import					
 Amazon EMR job executor					
	 Amazon Hive job executor				

More functions and features with PDI



More functions and features with PDI

- Automatic SQL-statements using Table Input and Table Output (CREATE TABLE) Statement
- SQL in PDI:
 - Table Input
 - SQL-Script
 - Database Join



- Variables in PDI → https://help.pentaho.com/Documentation/8.2/Products/Data_Integration/Data_Integration_Perspective/Run_Modifiers/Variables

Normalization and Denormalization

The screenshot displays the SAP Data Services interface with a workflow consisting of three steps: Data grid, Row denormaliser, and Row normaliser. The 'Data grid' step is selected, showing its configuration window. The 'Row denormaliser' step is also visible, showing its configuration window. The 'Row normaliser' step is also visible, showing its configuration window. The 'Examine preview data' window shows the data after the Row denormaliser step. The 'Results' window shows the execution history of the transformation.

Data grid configuration window:

Step name: Data grid

#	cust_id	cust_key	cust_value
1	101	first_name	Pavlin
2	101	last_name	Marvrodiev
3	101	age	35
4	102	first_name	Chris
5	102	last_name	Dollfeet
6	102	age	45
7	103	first_name	Hans
8	103	last_name	Muster
9	103	age	50
10	104	first_name	Marc
11	104	last_name	Friedrich
12	104	age	60

Row denormaliser configuration window:

Step name: Row denormaliser

The key field: cust_key

The fields that make up the grouping:

#	Group field
1	cust_id

Row normaliser configuration window:

Step name: Row normaliser

Type field: cust_key

#	Fieldname	Type	new field
1	target_firstname	cust_firstname	cust_val
2	target_lastname	cust_lastname	cust_val
3	target_age	cust_age	cust_val

Examine preview data window:

Rows of step: Row denormaliser (4 rows)

#	cust_id	target_firstname	target_lastname	target_age
1	101	Pavlin	Marvrodiev	35
2	102	Chris	Dollfeet	45
3	103	Hans	Muster	50
4	104	Marc	Friedrich	60

Results window:

Execution History

14:54:39 - Normalization_Denormalization - Dispatching started for transformation

14:54:39 - Data grid.0 - Finished processing (I=0, O=0, R=0, W=12, U=0, E=0)

14:54:39 - Row denormaliser.0 - Finished processing (I=0, O=0, R=12, W=4, U=0, E=0)

14:54:39 - Row normaliser.0 - Finished processing (I=0, O=0, R=4, W=12, U=0, E=0)

14:54:39 - Spoon - The transformation has finished!!

Regular Expression

- Regular Expressions: (Wikipedia: **regex** or **regexp**;^[1] also referred to as **rational expression**)^{[2][3]} is a sequence of characters that define a *search pattern*)

Example: 2013-(06|07|08|09|10|11|12)-.*_roi-event_TEST.txt

[!\"#\$%&'()*+,-.\\|/;<=>?@[\\]`{}^_`{|}~] → select all special chars

In PDI Regexp can be used in different applications :

- As selection of files
- As Filter for records
- String Operations (i.e. «replace in String»)

Get File with FTP

Get a file with FTP

Name of this job entry: Get a file with FTP LOCALHOST

General Files Advanced Socks Proxy

Remote

Remote directory: Display_Kampagnen_Goldbach_Account

Wildcard (regular expression): 2013-{06|07|08|09|10|11|12}-{^[^SEM]_roi-event.txt

Remove files after retrieval? ☐

Move files after retrieval? ☐

Move to folder

Create folder ☐

Name: MEDIAPLEX roi-input_TEST.txt Files 2

Directory

Add

Session

Session

Id files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders	
1	/var/local/file_staging_area/mediaplex/FTP_DOWNLOAD/	.*\..txt	.*_SEM_roi-event_TEST.txt	N	N	

Filling the SQL-statement automatically (CREATE TABLE)

