Pentaho
Data Integration

Good Morning Folks.. Grab a cup of coffee / tea.. We kick off at 9.00am!

Welcome…

- Introduction
- Schedule

- Verify Training Environment:
  - https://halo.labs.hds.com
  - Pentaho Training Environment

- Course Outline

Schedule

- 9:00 am – 5:00 pm
- 1 hour lunch break
- Other breaks as needed

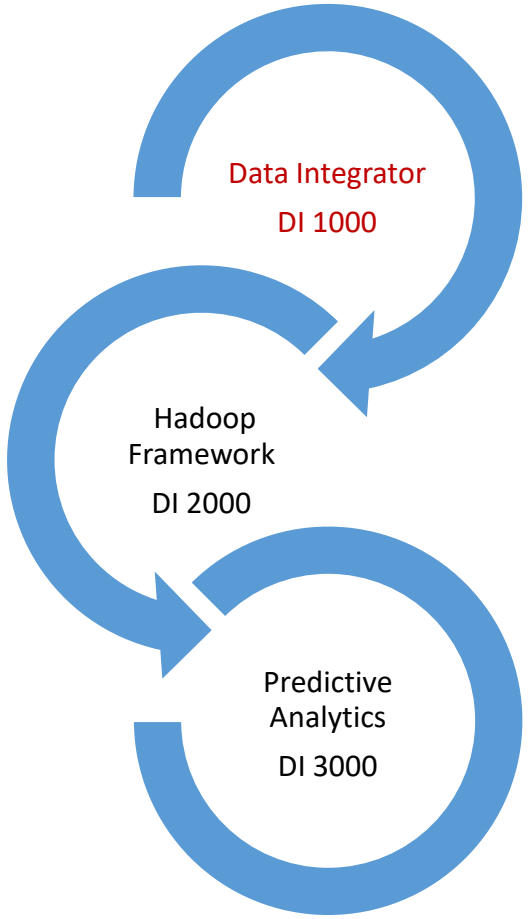
- Times are approximate and the actual times for breaks and lunch is guided by the pace of the exercises and discussions

- Please ask questions to make this an interactive and fun learning experience..!

Course Outline

Day 1

- PDI Platform & Components
  - Spoon
  - PDI Repository
- PDI Concepts & Terminology
  - Transformations
  - Metadata
  - Data Explorer
  - Jobs
- Data sources
  - Working with Files
    - CSV, TEXT
    - Excel
    - XML
    - JSON

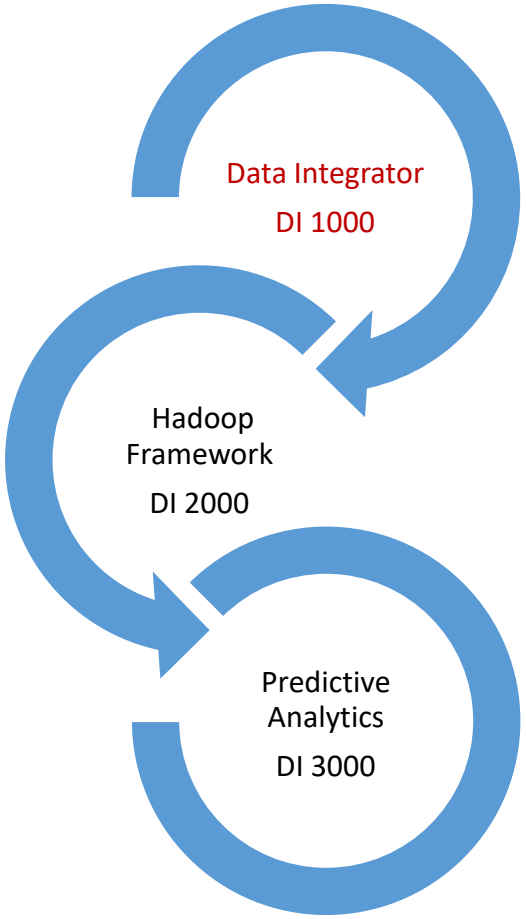


Course Outline

Day 2

- Data sources
  - Working with Databases
    - Connect to database
    - Getting data from a CSV file into a database
    - Inserting / Updating records
    - Dimension Lookup / Update
    - Deleting records
    - Passing parameters

- Data Enrichment
  - Merge
  - Joins
  - Lookups
  - Scripting

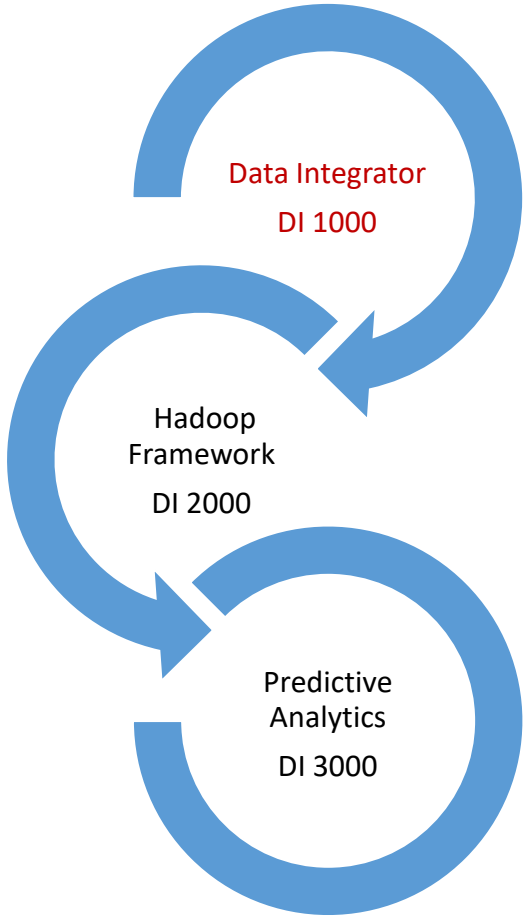


Course Outline

Day 3

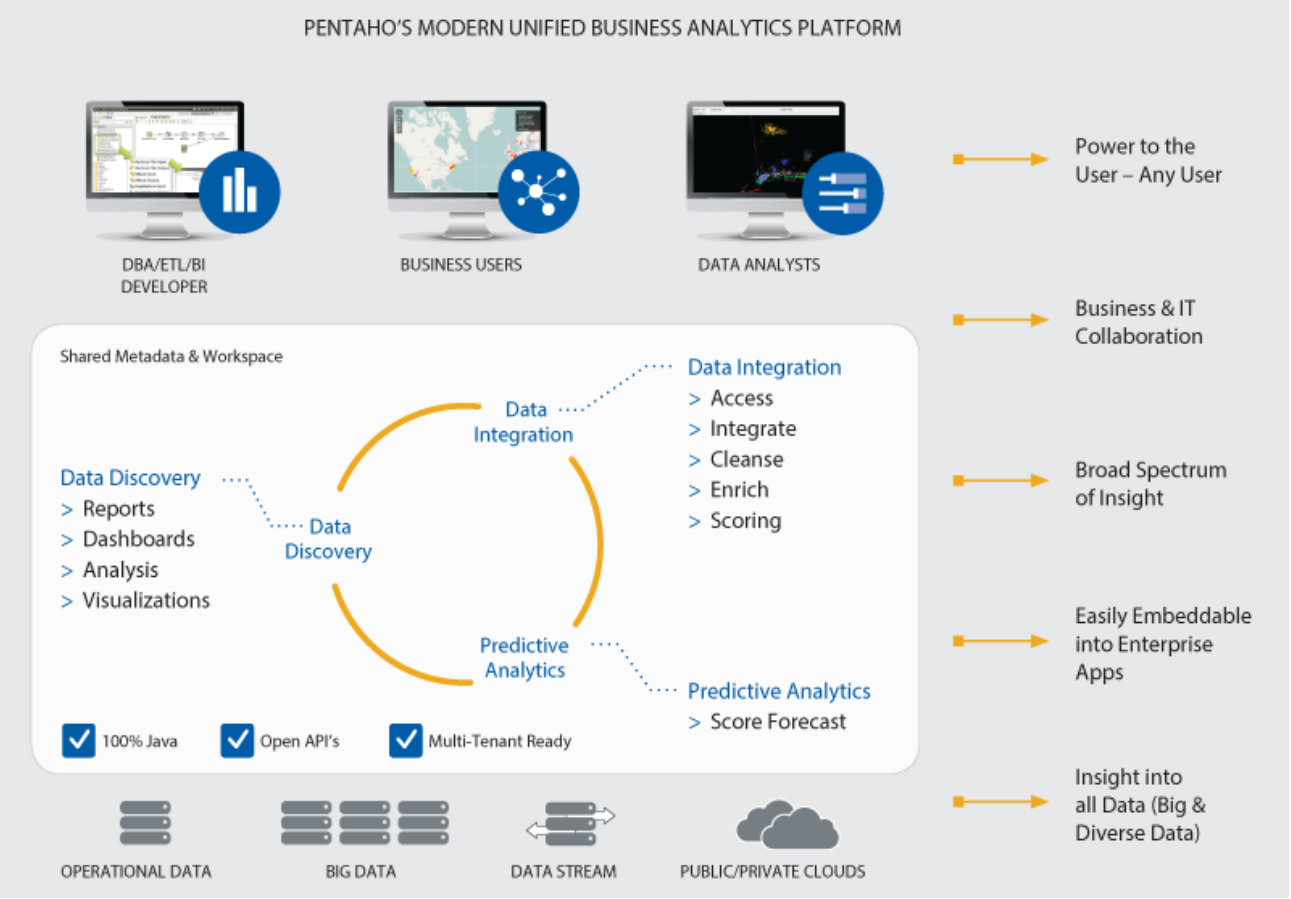
- Data Enrichment
  - Merge
  - Joins
  - Lookups
  - Scripting

- Enterprise Solution
  - Jobs
  - Scaling – Master / Slave nodes
  - Monitoring
  - Scheduling
  - Logging



Pentaho Data Integration

Module 1: Platforms & Components



Topics

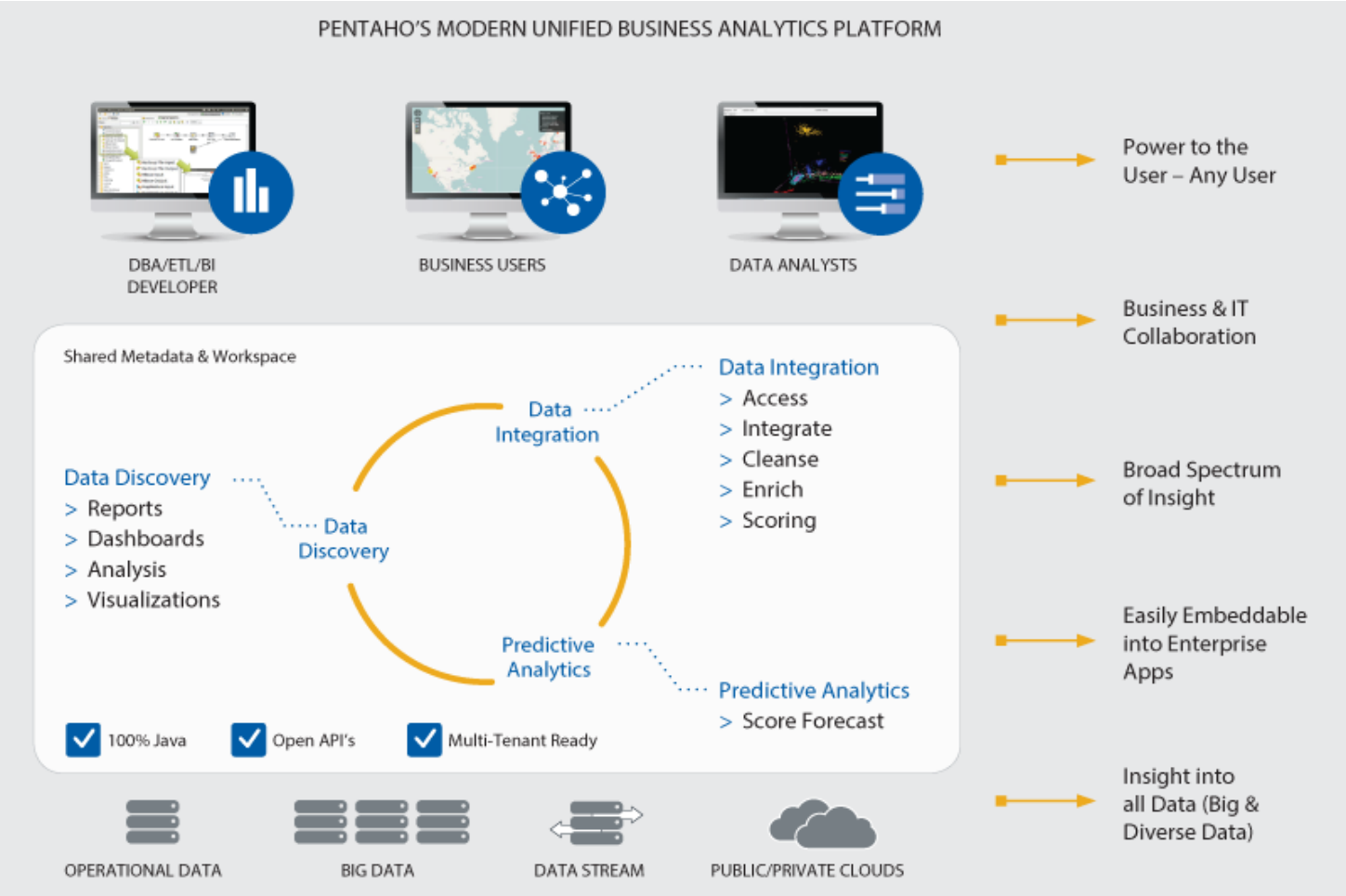
- Pentaho BA Platform
- Enterprise Architecture
- Pentaho Data Integration

Pentaho Analytics Platform

Pentaho
Data Integration

Pentaho Business Analytics

- Overview of Pentaho Business Analytics Platform



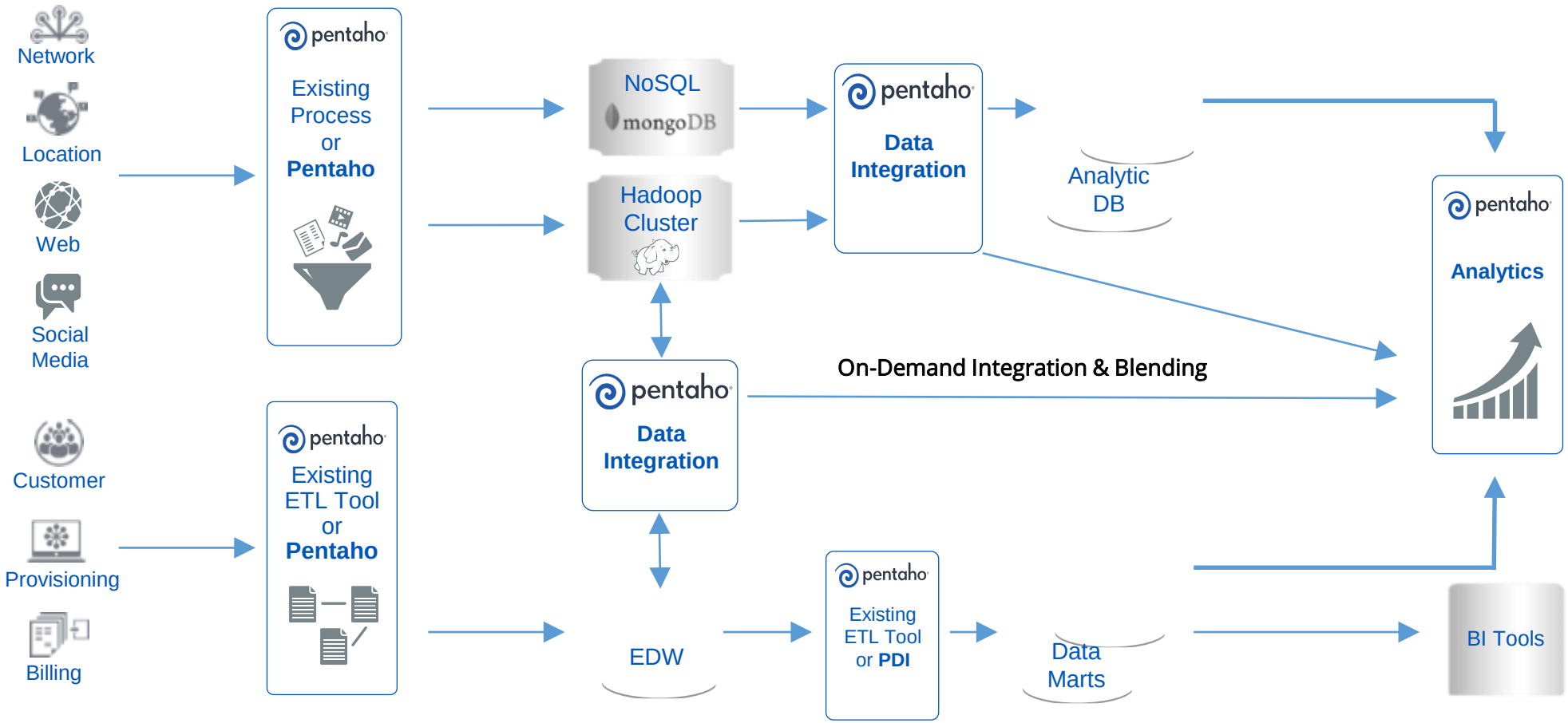
- BA Server and User Console
- Web-Based Tools and Plugins
- Client-Based Design Tools

Analyzer
Interactive Reporting
Dashboard Designer
Mobile

Report Designer

pentaho

Enterprise Data Architecture



- Blending Big Data and Traditional Architectures into a Data Lake..

pentaho

PDI Components

Client Tools

Data Integration (Spoon)

Designer
Engine
WEKA
Repo Browser
Scheduler
DB Browser

Pan
Kitchen

.ktr, .kjb

Pentaho Server

Data Integration (DI) Topology

Engine
Repository
Quartz
Security
Monitoring

Repository

pentaho

PDI Components

Spoon

- Graphical modeling environment for developing, testing, debugging and monitoring jobs and transformations

Data Integration Server

- Dedicated ETL server used for remote execution providing scheduling, security integration and content management capabilities

Kitchen, Pan

- Command-line driven job and transformation runners used for OS-level scheduling

Carte

- Light-weight HTTP server used for remote execution and parallel execution of jobs and transformations on a scale-out cluster (slave nodes)

pentaho

Spoon

Pentaho
Data Integration

Topics

Guided Demo 1-2-1: Spoon & Documentation

- Reviewing the PDI installation directory
- Starting Spoon
- Touring Spoon's user interface
- Changing Spoon's user interface options

Guided Demo 1-3-1: KETTLE Configuration

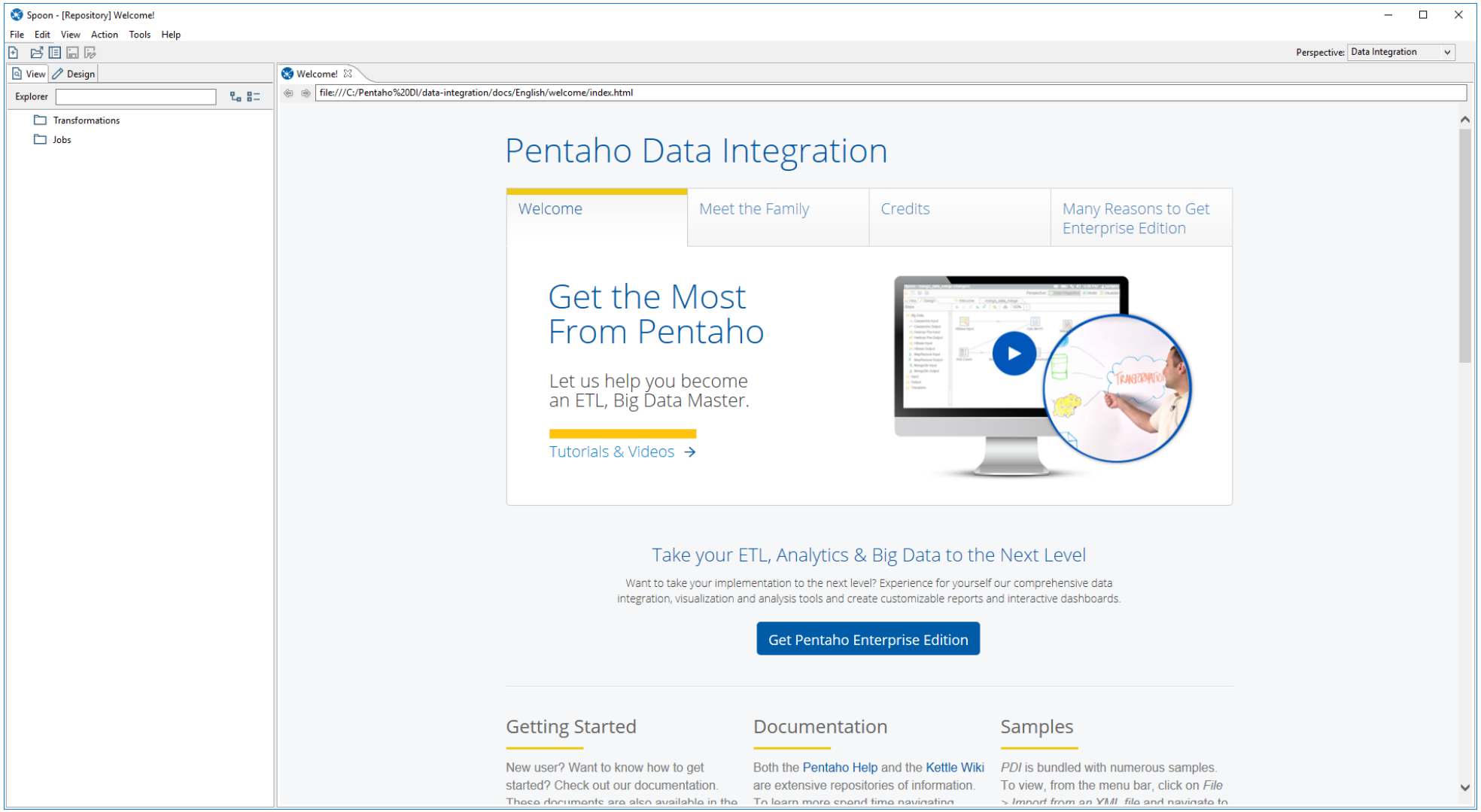
- Configuring PDI

Guided Demo 1-3-2: KETTLE variables

- Edit kettle.properties file and set kettle variables

pentaho

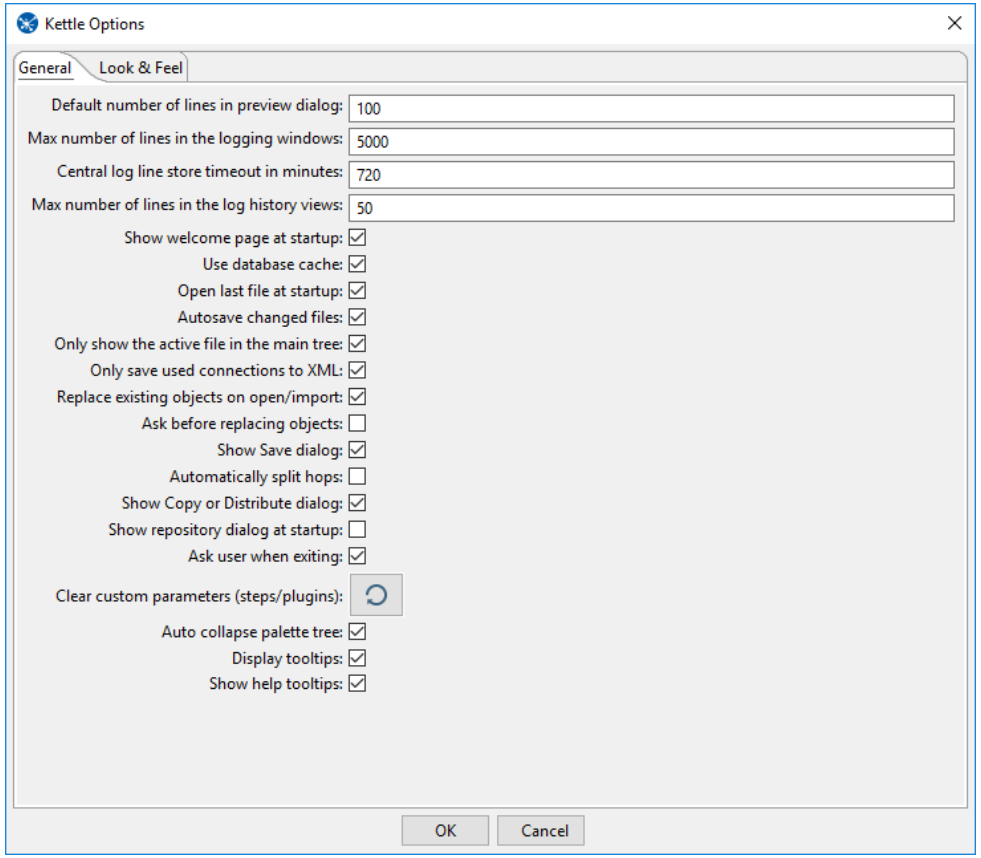
PDI Welcome

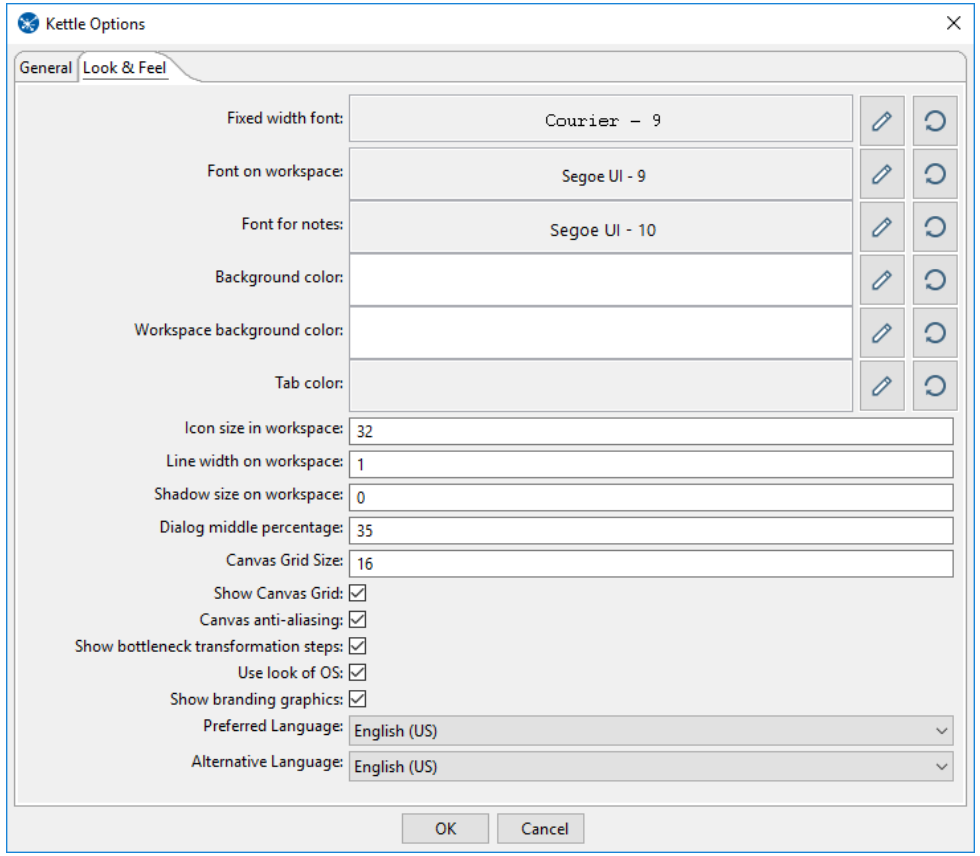


GD1-2-1: Look and Feel

This guided demo introduces some of the Kettle key look and feel options:

Tools > Options





Configuration

kettle.properties

Pentaho Data Integration

PDI Installation

All Kettle programs can be started using shell scripts located in the Kettle home directory.

Directory / File	Windows / Unix	Action
Shell script	spoon.bat / spoon.sh	Starts Spoon
Shell script	kichen.bat / kitchen.sh	Starts command line for Jobs
Shell script	pan.bat / pan.sh	Starts command line for Transformations
Samples	..\data-integration\samples	samples of .ktr and .ktj
..\lib	.jar files	add 'driver.jar' file mysql-connector-java-5.1.35-bin.jar
..\server\data-integration-server\tomcat\lib	.jar files	add 'driver.jar' file mysql-connector-java-5.1.35-bin.jar

pentaho

GD1-3-1: PDI Configuration

Kettle home directory (.kettle)

- Configuration files that control the behavior of PDI jobs and transformations
- Located at user's home directory by default → user dependency
- KETTLE_HOME variable can be used set the location

kettle.properties	Default properties file for variables
shared.xml	Default shared objects file
db.cache	The database cache for metadata
repositories.xml	The local repositories file
.spoonrc	User interface settings, last opened transformation/job
.languageChoice	User language (delete to revert language)

- Use a different KETTLE_HOME variable for each unique combination of customer, project and environment

pentaho

PDI Configuration

kettle.properties

- The kettle.properties file is where you set the global variables for Kettle e.g. hold database connections, paths to directories, constants that occur in your transformation or jobs.
- Each property is denoted by key/value pair
- Reference notation: ${<*propertyname*>} or %%<*propertyname*>%%

- Located: C:\ users\<username>\.kettle\kettle.properties

PDI Configuration

connection properties for db server

DB_HOST=dbhost.domain.org.com

DB_NAME= whatever

DB_USER=dblogin

DB_PASSWORD=db_password

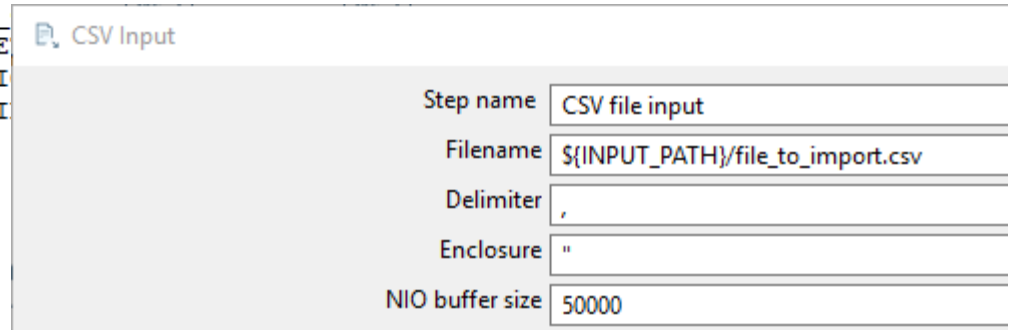
#path to read input files

INPUT_PATH=path to files

#path for error reports

ERROR_PATH=path to error reports

```
## This file was generated by Pentaho Data Integration version 6.0.1.0-386.
#
# Here are a few examples of variables to set:
#
# PRODUCTION_SERVER = hercules
# TEST_SERVER = zeus
# DEVELOPMENT_SERVER = thor
#
# Note: lines like these with a # in front of it are comments
#
#
#Sat Apr 02 12:51:59 BST 2016
KETTLE_COMPATIBILITY_IMPORT_PATH_ADDITION_ON_VARIABLES=N
KETTLE_REDIRECT_STDERR=N
KETTLE_SHARED_OBJECTS=
KETTLE_METRICS_LOG_DB=
KETTLE_DEFAULT_DATE_FORMAT=
KETTLE_JOB_LOG_SCHEMA=
KETTLE_DEFAULT_INTEGER_FORMAT=
KETTLE_AGGREGATION_MIN_NULL_IS_VALUED=N
KETTLE_PLUGIN_CLASSES=
KETTLE_LOG_MARK_MAPPINGS=N
KETTLE_CORE_
KETTLE_ROWSE
KETTLE_METRI
KETTLE_PLUGI
```

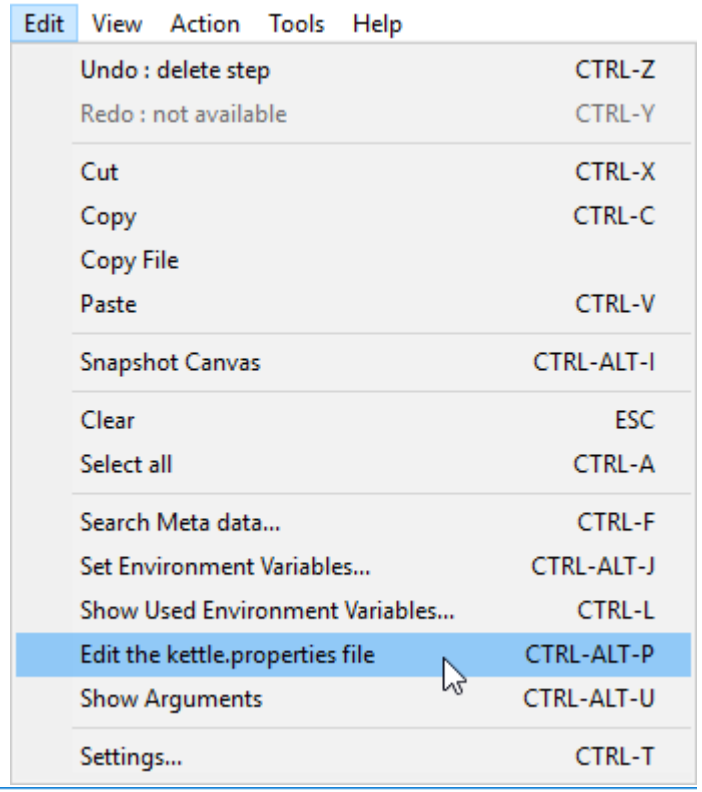


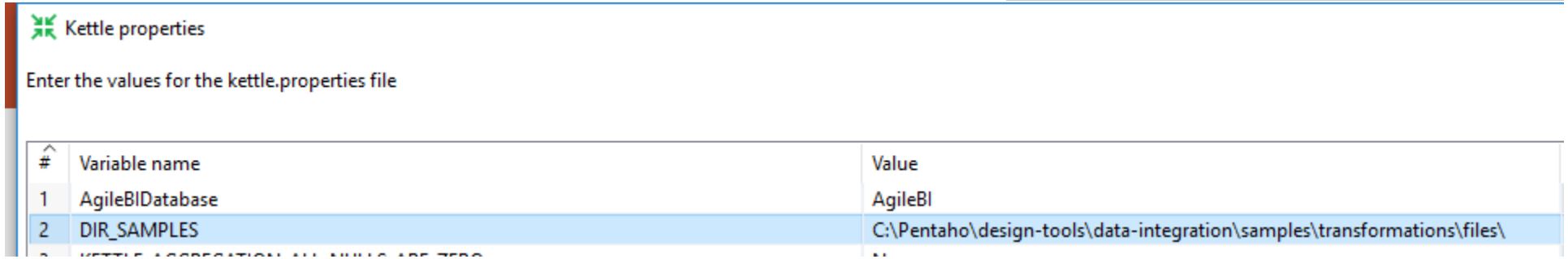
pentaho

GD1-3-2: Edit kettle.properties

To edit the kettle.properties

- Add DIR_SAMPLES
  C:\pentaho\design-tools\data-integration\samples
  \transformations\files\





Summary

Guided Demo 1-2-1: Spoon & Documentation

- Reviewing the PDI installation directory
- Starting Spoon
- Touring Spoon's user interface
- Changing Spoon's user interface options

Guided Demo 1-3-1: KETTLE Configuration

- Configuring PDI

Guided Demo 1-3-2: KETTLE variables

- Edit kettle.properties file and set kettle variables

pentaho

Repository

Pentaho Data Integration

Topics

- Pentaho Repositories

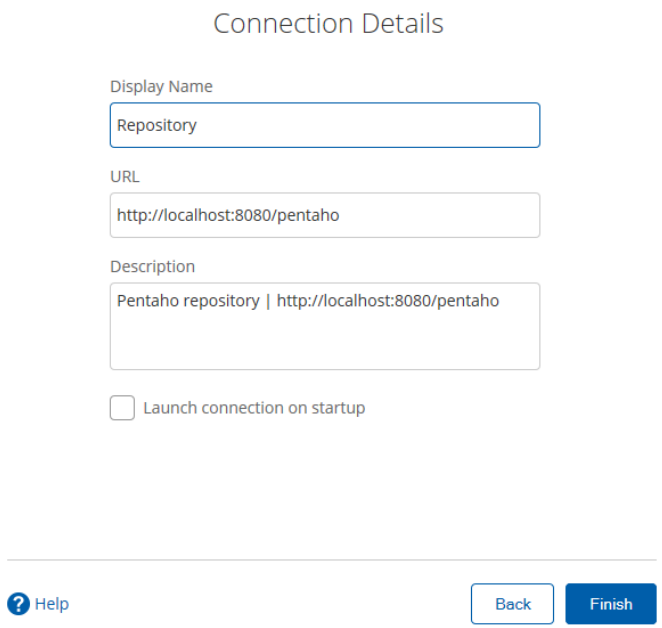
- Pentaho Enterprise Repository
  - Security
  - Content Management
  - Scheduling & Monitoring

- Guided Demo 1-4-1: Repository
  - Configure the Repository

- Guided Demo 1-4-2: Upload to the Repository
  - Upload ‘Objects’ to the Repository

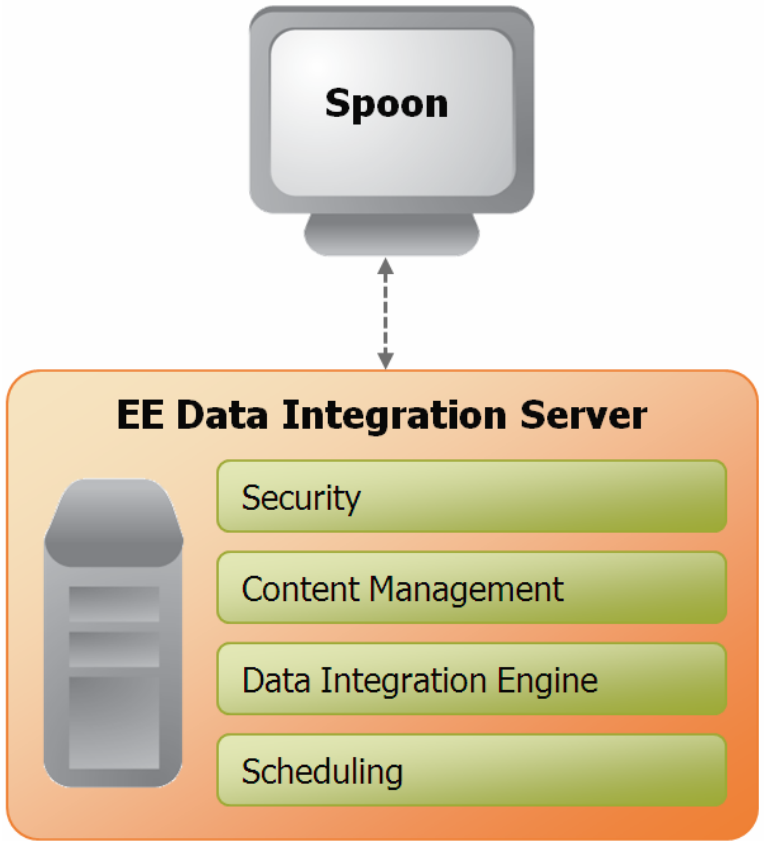
Pentaho Repositories

- There are 3 Repository types:
  - Enterprise Repository - runs on Data Integration Server, open source CMS Apache Jackrabbit
  - Database Repository - can run a script MS SQL Server, Oracle, MySql,
  - File Repository - stored as xml



PDI Repository

- Based on the security and content management modules in the EE Data Integration Server:



Security – Allows you to manage users and roles (default security) or integrate security into your existing security provider (such as LDAP or Microsoft Active Directory)

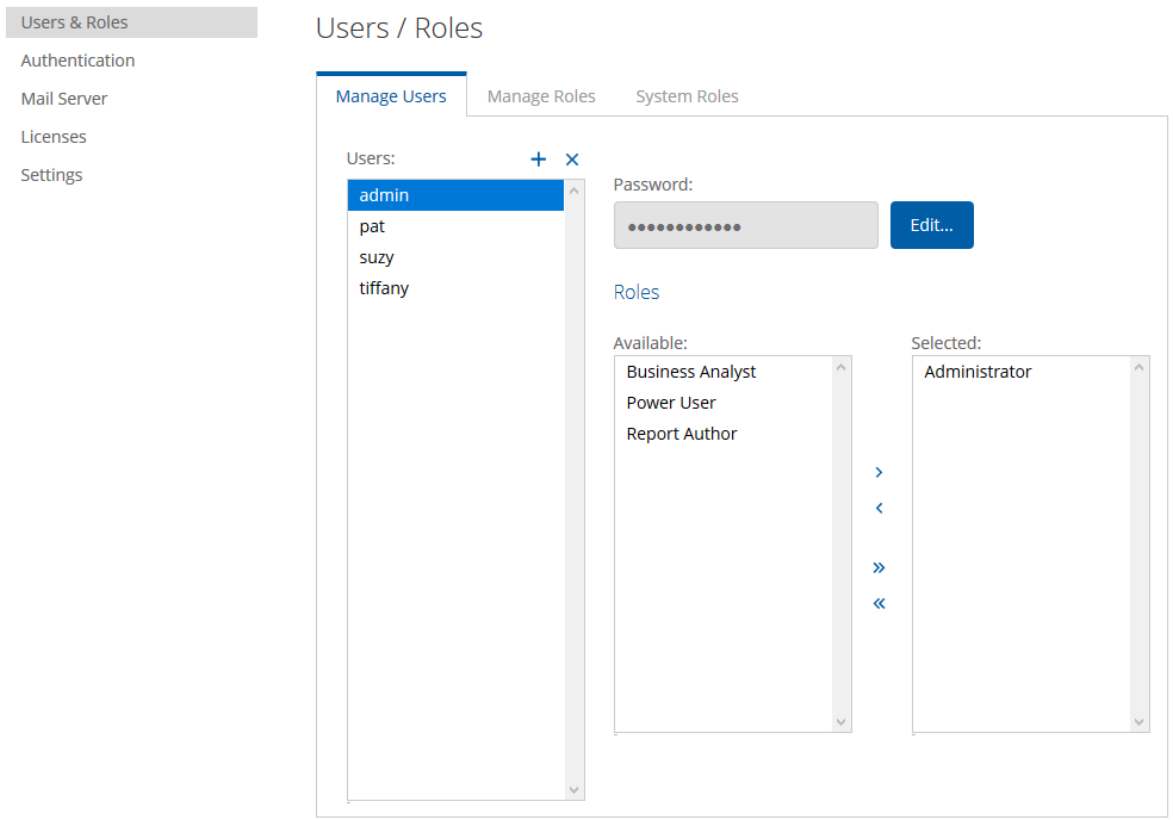
Content management – Provides the ability to centrally store and manage ETL jobs and transformations (includes full revision history of content and features such as sharing and locking for collaborative development environments)

Data Integration Engine – This is a Carte instance. Carte is also used in clustering (covered later in this course).

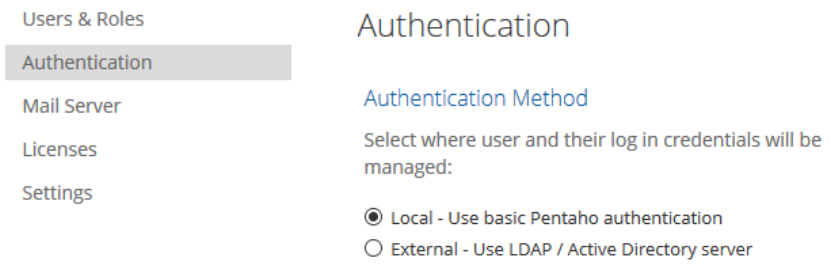
Scheduling – The Quartz scheduler is used internally, and the tasks are executed in the data integration engine.

Repository Security

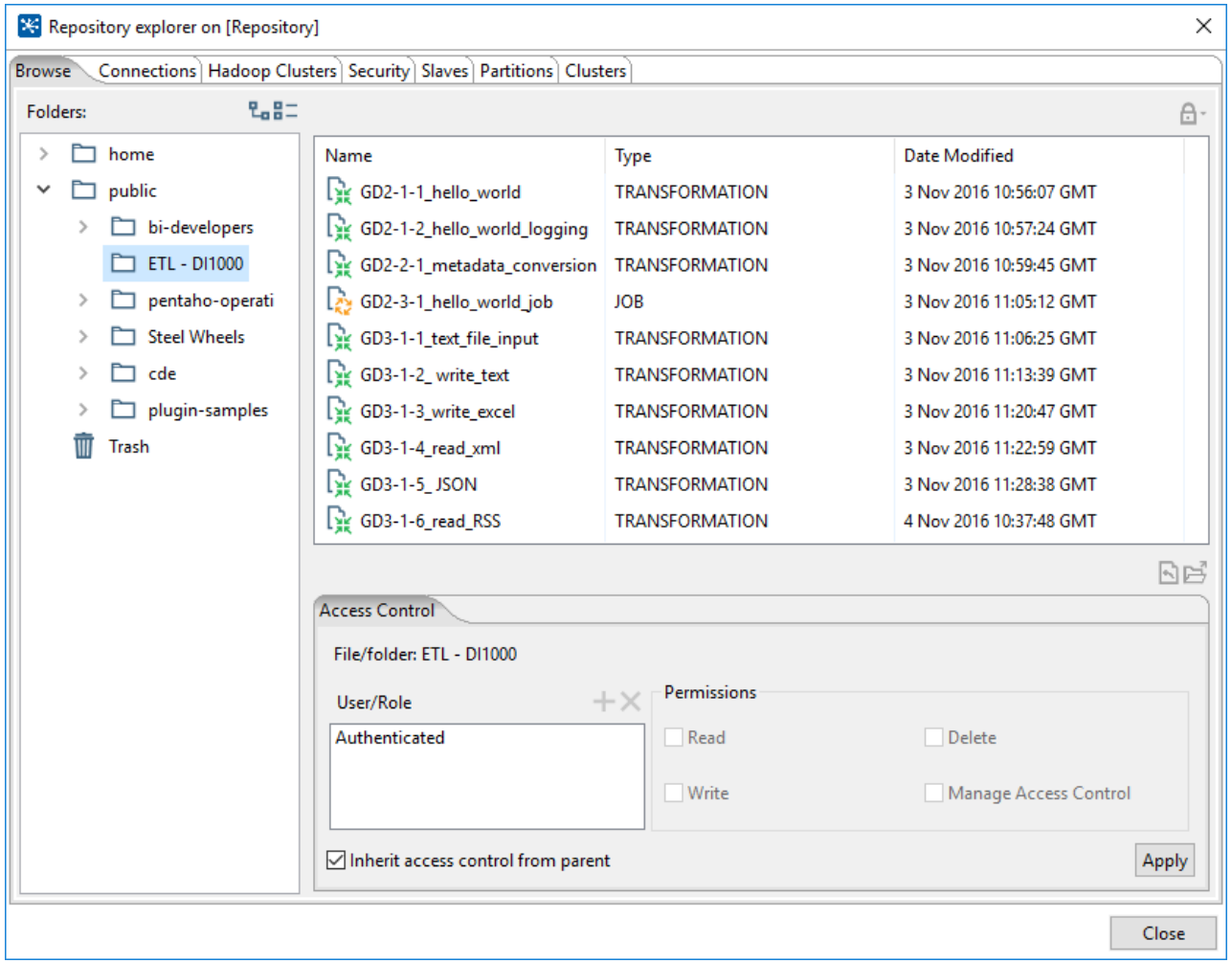
- The Data Integration Server is configured (by default) to use the Pentaho default security provider.
  - Pre-populated with a set of sample users and roles:



See the security guide in the Pentaho Documentation for details on configuring security with an existing security provider (such as LDAP or Microsoft Active Directory).



Content Management

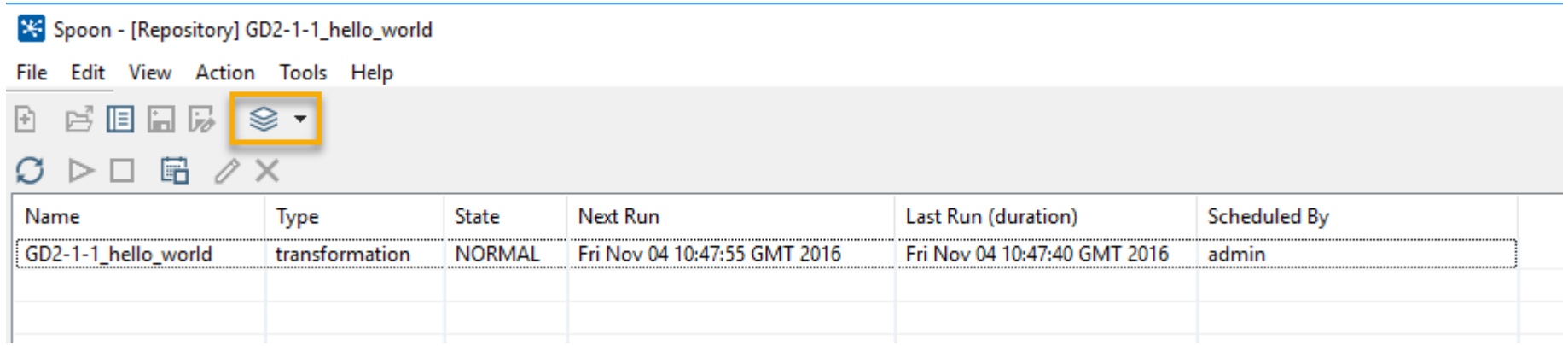


- Repository based on JCR (Content Repository API for Java)
- Enterprise security:
  - Configurable authentication including
    - support for LDAP and MSAD
  - Task permissions defining what actions a user/role can perform such as read/execute content, create content, and administer security
  - Granular permissions on individual files / folders
- Full revision history (not enabled by default)
- Ability to lock transformations/jobs

pentaho

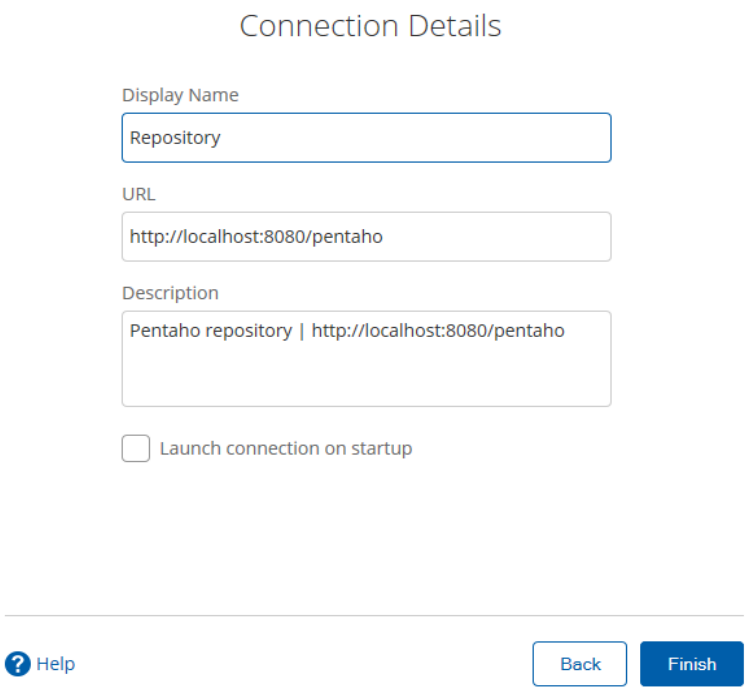
Scheduling & Monitoring

- Switch to Schedule Perspective



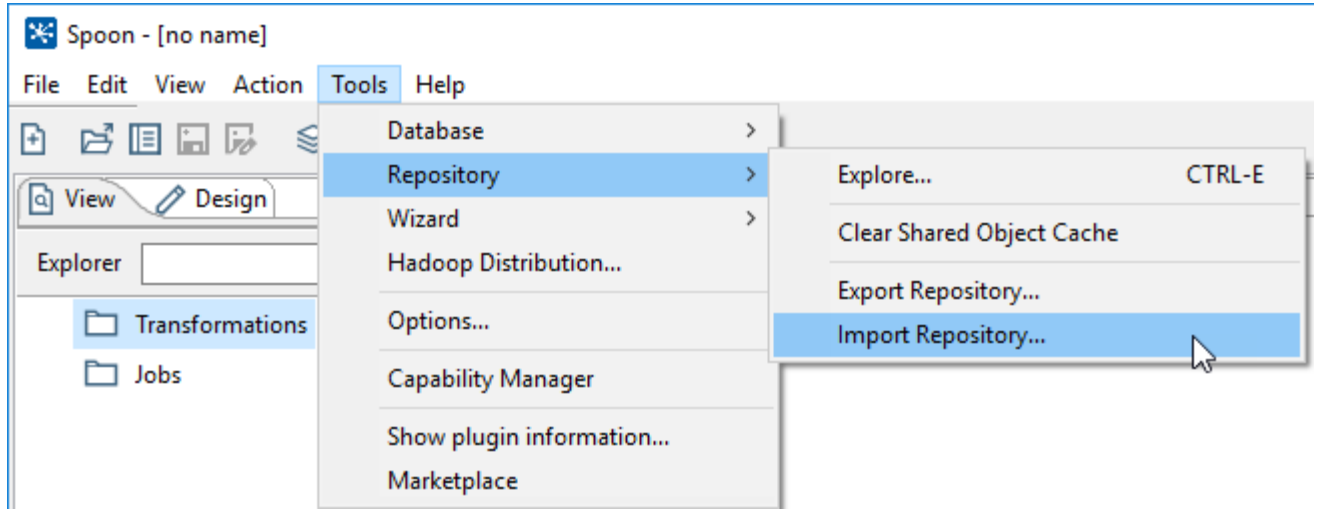
GD1-4-1: Define Enterprise Repository

- If you don’t have a Repository



GD1-4-2: Import Repository

- Import all the Transformations / Jobs for the course.
- Just an XML file..!



- Repository.xml located in Resources folder
- No Rules, import to Root.

pentaho

Summary

- Pentaho Repositories

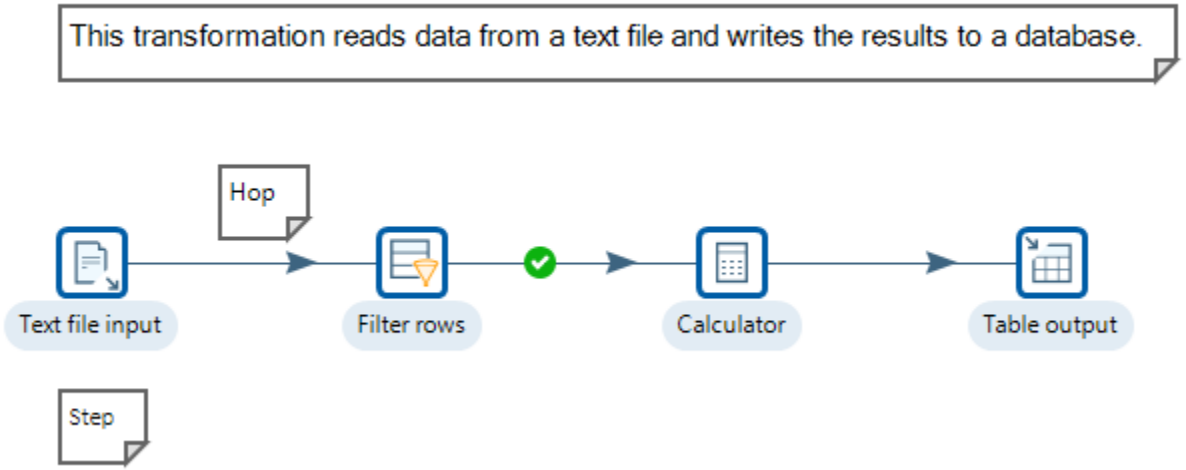
- Pentaho Enterprise Repository
  - Security
  - Content Management
  - Scheduling & Monitoring

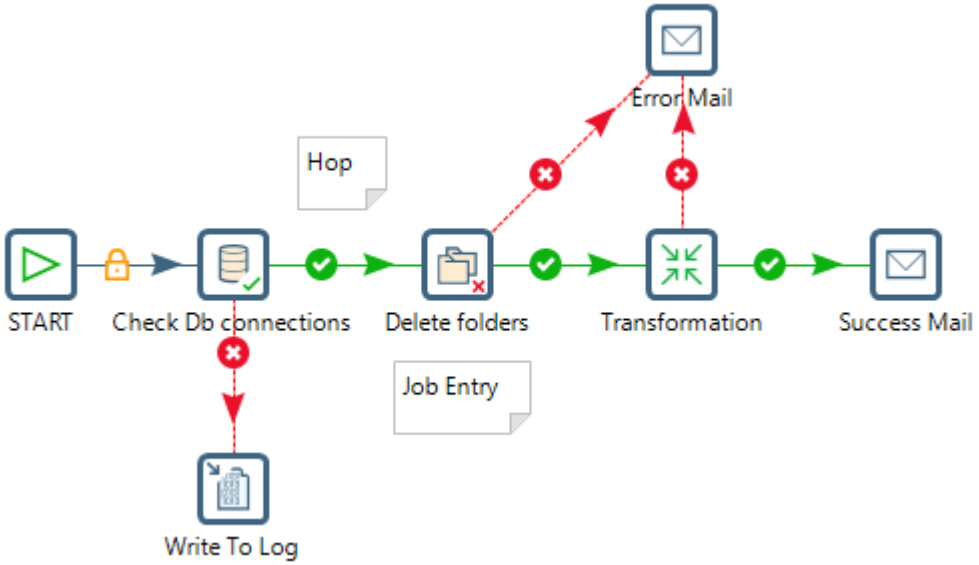
- Guided Demo 1-4-1: Repository
  - Configure the Repository

- Guided Demo 1-4-2: Upload to the Repository
  - Upload ‘Objects’ to the Repository

Pentaho Data Integration

Module 2: Concepts & Terminology





Topics

- Transformations
  - Steps
  - Transformation Hops
- Guided Demo 2-1-1: Hello World
- Guided Demo 2-1-2: Hello World (Logging)
- Parallelism

- Metadata
- Data Types
- Guided Demo 2-2-1: Metadata Conversion

- Data Explorer
- Guided Demo 2-3-1: Data Explorer

- Jobs
  - Job Entries
  - Job Hops
- Guided Demo 2-4-1: Hello World (Job)

Transformations

Parallelism

Basic Logging

Metadata

Data Explorer

Jobs

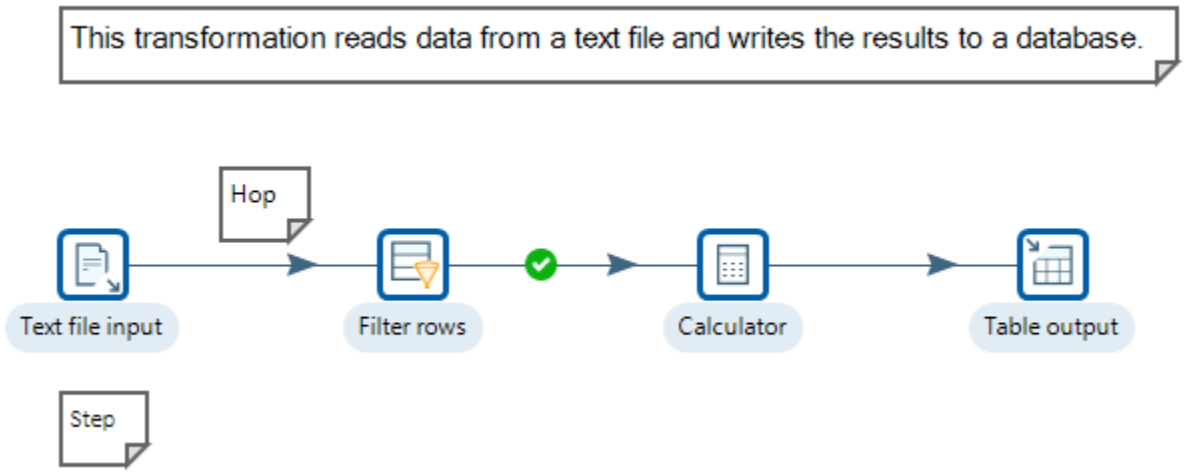
Pentaho Data Integration

Topics

- Transformations
  - Steps
  - Transformation Hops
- Guided Demo 2-1-1: Hello World
- Guided Demo 2-1-2: Hello World (Logging)

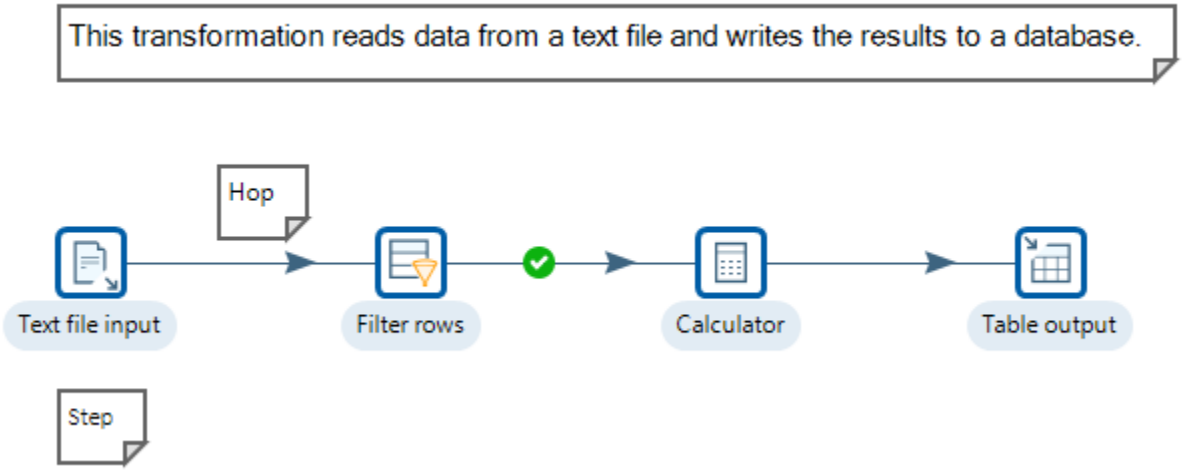
- Parallelism

Transformations



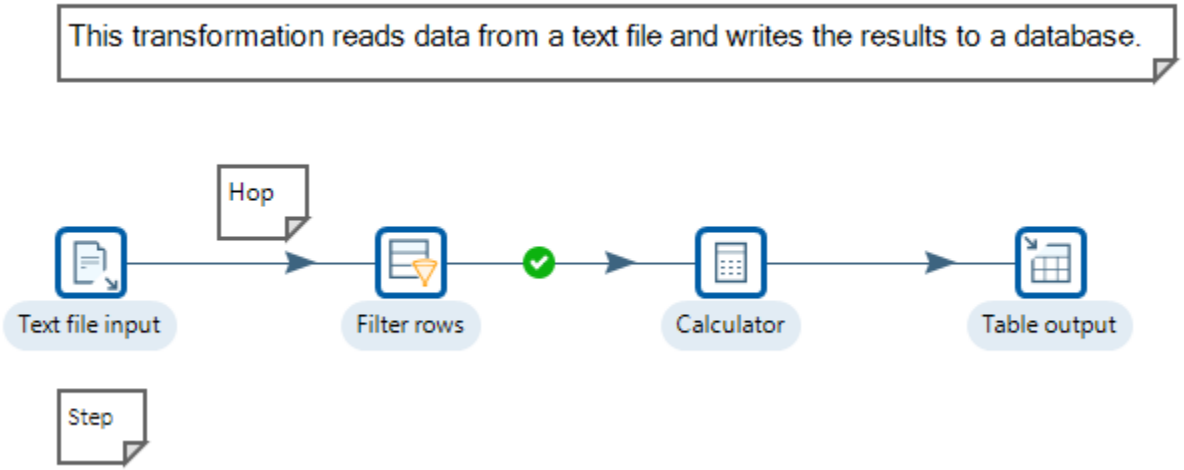
- Workhorses of your ETL solution
- Handle the manipulation of rows of data
- Consist of steps that perform the core work
- Steps are connected by hops
- Data Flow is the movement of rows from one step to another

Steps



- Core building blocks in a transformation, each having their distinct functionality
- Read data from incoming hops, write data to outgoing hops
- Can have multiple outgoing hops: copy or distribute (round robin)
- Each step is started simultaneously and runs in its own thread
  - Parallelism
  - A thread is a concurrently running task

Hops



- Define the data path between steps
- Hops also represent a row buffer called the row set (5,000 – 50,000)
  - When row set is full, the step that writes rows halts
- Different types of hops available

Parallelism

- The hops (buffer) allow steps to be executed in parallel
  - Work independently, at their own speed, in separate threads
- Once a buffer is full, parallelism gets reduced
  - Steps will operate at virtually the same speed
- Not possible to define an order of execution in a transformation
  - Every step is started simultaneously
  - Rows are being forced through the step network
- Functionally a transformation does have a start and end
- If you need to perform tasks in a specific order, refer to jobs

pentaho

GD2-1-1: Hello World

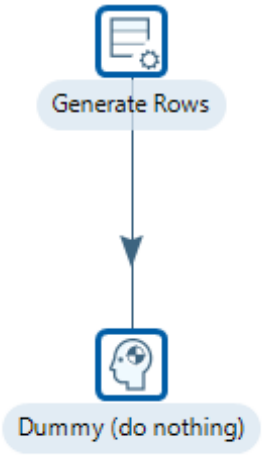
This transformation generates rows, 'hello world'. Illustrates some of the key features:

Step1: Generate Rows
Limit: 10 rows

Name: message
Type: String
Value: hello world

click: Preview

Step2: Dummy



- Generate ‘hello world’ data
- Explore execution results

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows Last rows Off

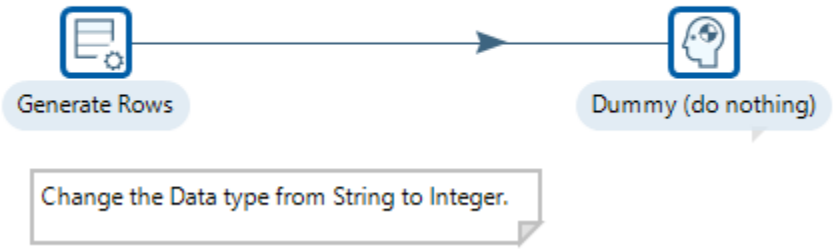
#	message
1	hello world
2	hello world
3	hello world
4	hello world
5	hello world

pentaho

GD2-1-2: Hello World - Basic Logging

Change the data type in ‘hello world’ from String to Integer

Changing the format from String to Integer results in an error.



Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

```
2016/10/26 16:00:18 - Spoon - Launching transformation [GD2-1-2_hello_world_logging]...
2016/10/26 16:00:18 - Spoon - Started the transformation execution.
2016/10/26 16:00:18 - GD2-1-2_hello_world_logging - Dispatching started for transformation [GD2-1-2_hello_world_logging]
2016/10/26 16:00:18 - Generate Rows.0 - ERROR (version 7.0-QAT-376, build 1 from 2016-10-06 21.16.11 by buildguy) : Couldn't parse Integer field [message] with value [hello world] --> org.pentaho.di.core.exception.KettleValueException:
2016/10/26 16:00:18 - Generate Rows.0 - Unexpected conversion error while converting value [message String] to an Integer
2016/10/26 16:00:18 - Generate Rows.0 -
2016/10/26 16:00:18 - Generate Rows.0 - message String : couldn't convert String to Integer
2016/10/26 16:00:18 - Generate Rows.0 -
2016/10/26 16:00:18 - Generate Rows.0 - message String : couldn't convert String to number : non-numeric character found at position 1 for value [hello world]
2016/10/26 16:00:18 - Generate Rows.0 - ERROR (version 7.0-QAT-376, build 1 from 2016-10-06 21.16.11 by buildguy) : Error initializing step [Generate Rows]
2016/10/26 16:00:18 - GD2-1-2_hello_world_logging - ERROR (version 7.0-QAT-376, build 1 from 2016-10-06 21.16.11 by buildguy) : Step [Generate Rows.0] failed to initialize!
2016/10/26 16:00:18 - Spoon - ERROR (version 7.0-QAT-376, build 1 from 2016-10-06 21.16.11 by buildguy) : GD2-1-2_hello_world_logging: preparing transformation execution failed
2016/10/26 16:00:18 - Spoon - ERROR (version 7.0-QAT-376, build 1 from 2016-10-06 21.16.11 by buildguy) : org.pentaho.di.core.exception.KettleException:
2016/10/26 16:00:18 - Spoon - We failed to initialize at least one step.  Execution can not begin!
```

Summary

- Transformations
  - Steps
  - Transformation Hops
- Guided Demo 2-1-1: Hello World
- Guided Demo 2-1-2: Hello World (Logging)

- Parallelism

Metadata

Pentaho Data Integration

Topics

- Metadata
- Data Types
- Guided Demo 2-2-1: Metadata Conversion

Metadata

Metadata

- Name, type, length, precision, format, group, decimal, currency
- Describes the fields in a row
- Not enforced beyond name and data type
  - e.g. strings are not cut to specified length
  - e.g. numbers are not rounded to specified precision
  - This functionality is explicitly available in dedicated steps

- All the rows in a row set need to have the same layout
  - Same fields, same data type, same order

pentaho

Rows of Data

- Unit of data in a transformation is a row
- A row is a collection of zero or more fields
- Each step is capable of describing the input and output fields (row metadata)
  - Name – name of the field
  - Type – data type of the field
  - Length – length of string or number
  - Precision – decimal precision of number
  - Mask – representational format (used in data type conversions)
  - Format – data conversion mask
  - Decimal – decimal symbol in number
  - Group – grouping symbol in number
- Metadata can change from step to step depending on the step's functionality

pentaho

Data Types

String	Any type of character data
Integer	A signed long integer (64-bit)
Number	Double precision floating point (up to 15 significant digits)
BigNumber	Number with unlimited precision
Date	Date-time value with millisecond precision
Timestamp	Date-time value with nanosecond precision
Boolean	True/false
Binary	Array of bytes
Internet Address	

📖 Data types are pluggable

pentaho

Data Type Rules

- All rows need to have the same layout / structure.
  - Esp: when you merge rows

- Beyond data type and name, field metadata is not enforced during the execution of a transformation, e.g. a *string* is not automatically cut to the specified length.

- By default, empty *strings* (“ ”*)* are considered NULL

pentaho

Guided Demo: 2-2-1 Metadata Conversion

Guided Demonstration: 2-2-1 Metadata Conversion

This transformation illustrates CSV File input step to read the sales_data.csv file.

Step1: CSV File input- Read Sales Data
Browse for the ..\samples\transformations\files\sales_data.csv
Delimiter: ,
Select: Lazy conversion & Head row present
Click: Get fields
Click: Preview

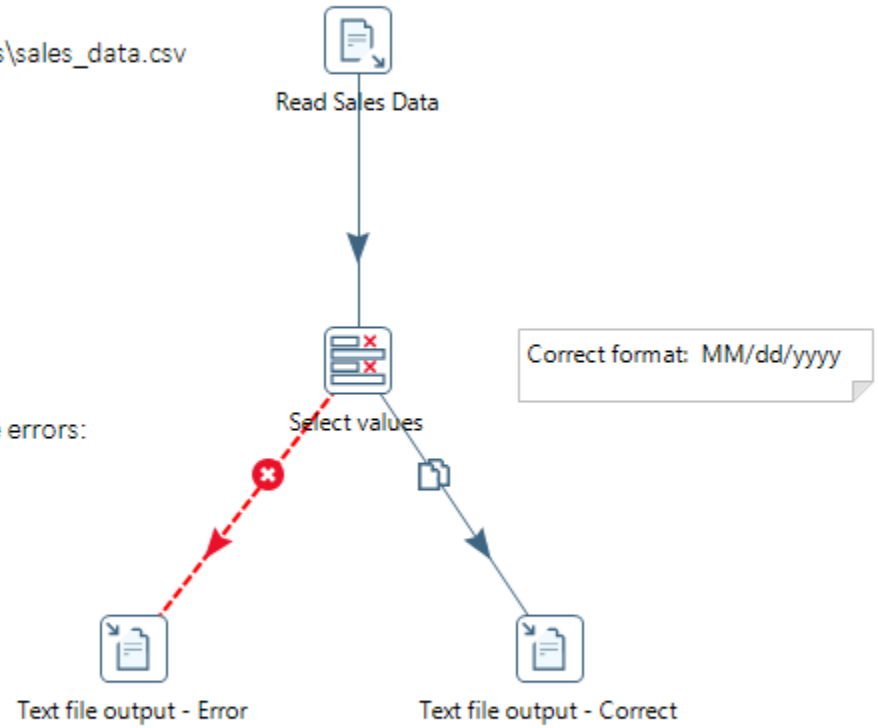
Step2: Select values
Select the Meta-data tab:
Click: Get fields to change
Change the format of the ORDERDATE to create errors:

Type:	Format:
Date	yyyy/MM/dd

This will cause errors..!

Step3: Text File Output- Correct
Used to Preview correct data

Step4: Text File Output- Error
Configure the hop for error logging



Remember to configure for errors:
Double click on Hop ..

Select Copy for record distribution

pentaho

Summary

- Metadata
- Data Types
- Guided Demo 2-2-1: Metadata Conversion

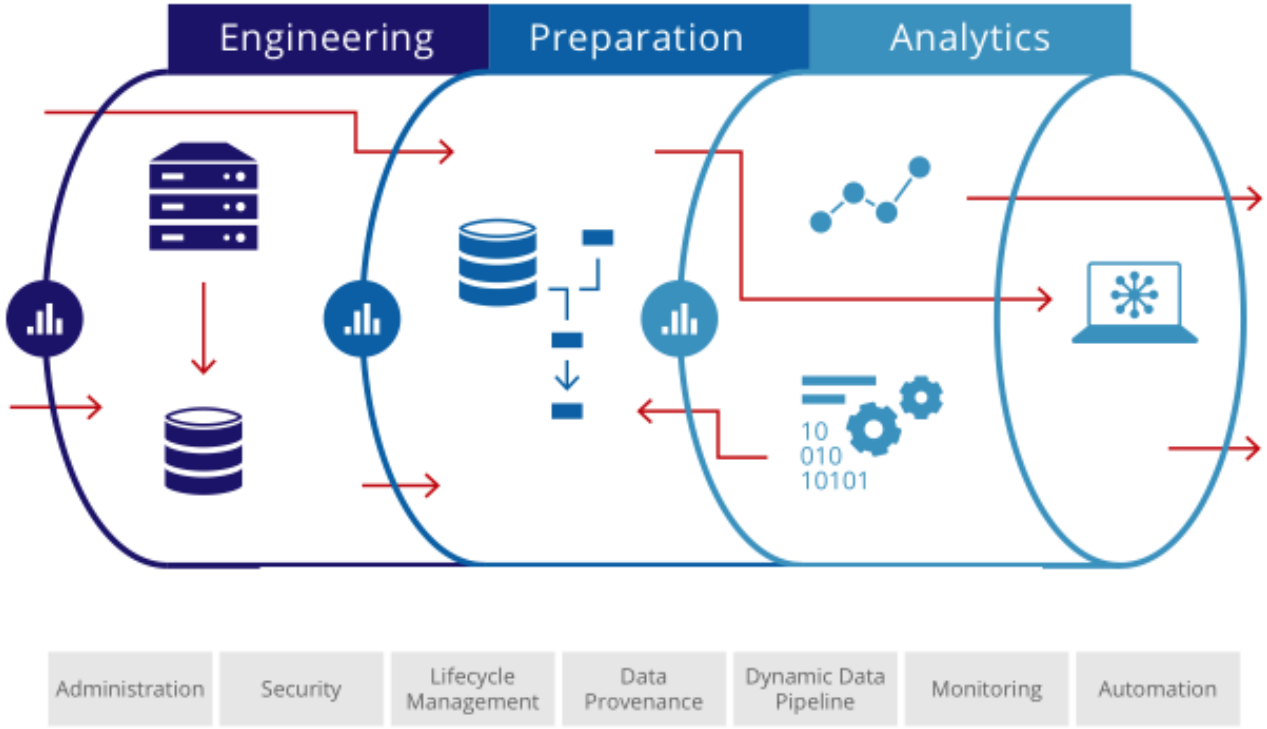
Data Explorer

Pentaho Data Integration

Data Explorer

Analytics anywhere in the data pipeline:

- Run and Inspect data
- Inspect data (cached)
- Not available in Hadoop



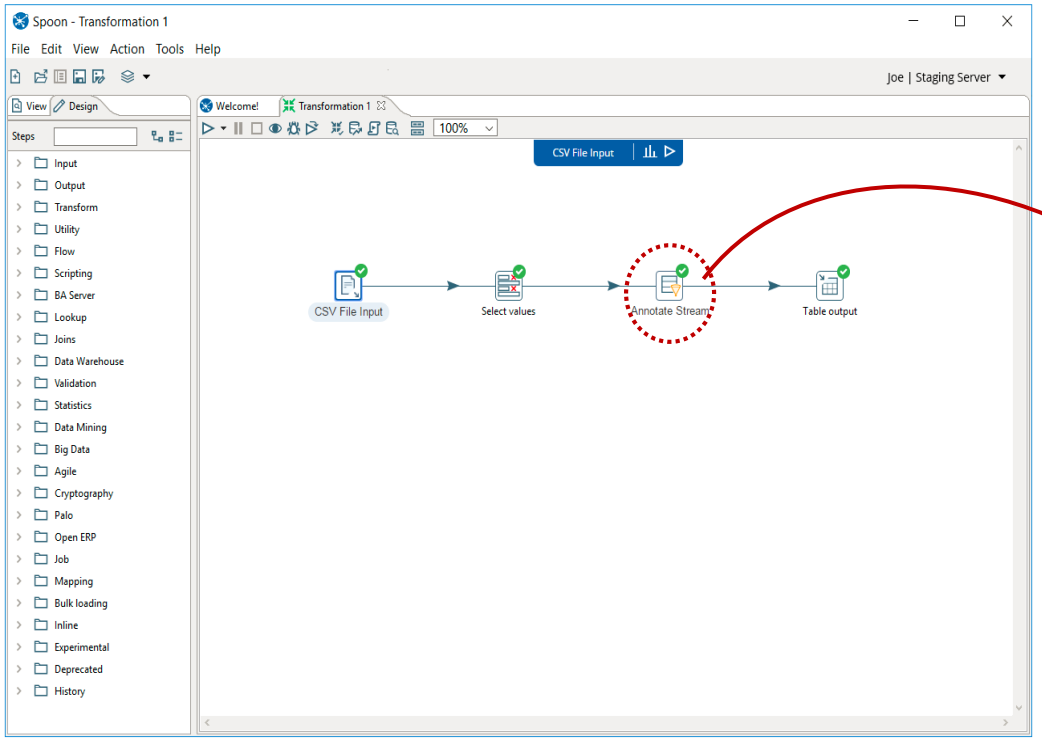
pentaho

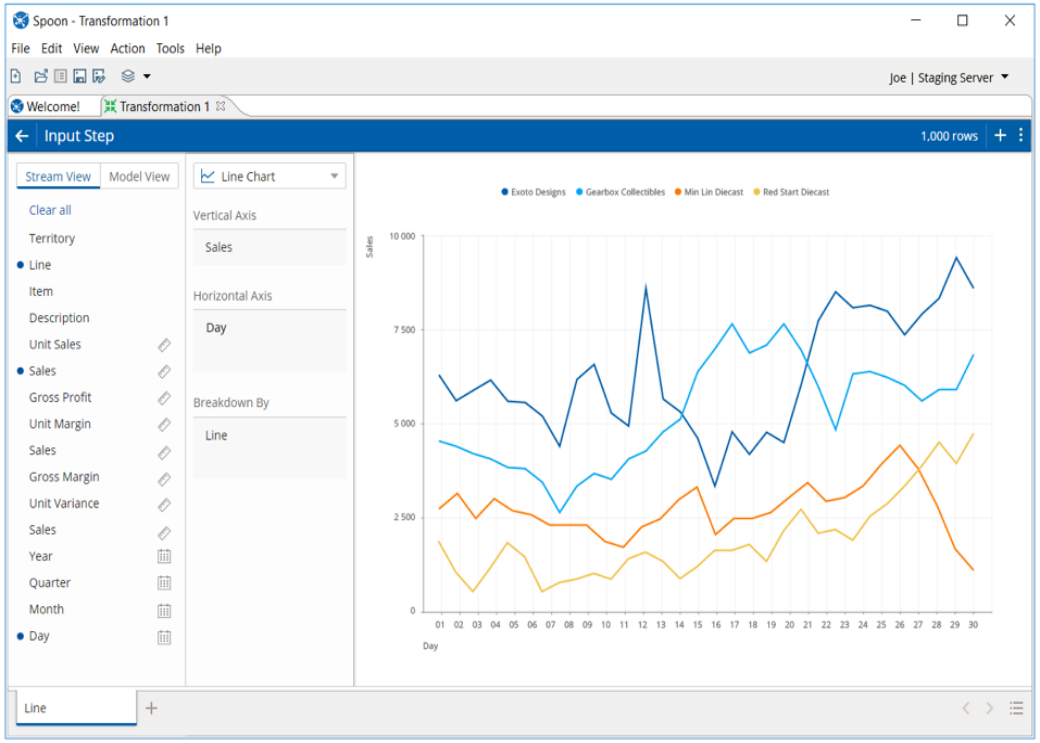
Data Explorer

- Data Explorer:
  - Access any analytics, including charts, visualizations, and reporting, from any step in data prep– shortening the cycle from data to analytics
  - ETL developers and data prep staff can easily spot check analytics in-flight
  - Directly publish data sources for the business user, creating a more collaborative process between business and IT
  - Data services to virtualize transformations without staging, making data sets immediately available to reports and applications
  - Set up a self-service data prep environment with governed, on-demand data sets.

pentaho

Data Explorer





pentaho

GD3-2-1: Tab Persistency

1. Persist (keep) a Data Explorer tab, which creates an icon on canvas
2. Show how changes to fields upstream will cause existing downstream visualizations with those fields not to render – and how to mitigate this by replacing the fields in the layout pane
3. Show that when you close Data Explorer on a given tab, that tab remains in view when you reopen Data Explorer
4. Show how to revert a step to the default table view – through changing the visualization type to the flat table or adding a new flat table tab

GD3-2-2: Visualizations & Hierarchies

1. Background: Show what Data Explorer generates in auto model – measures, attributes, and basic geo hierarchies
2. Background: Show how Annotate Stream step is necessary to create a complete model for BI prototyping in Data Explorer, with correct dimensions, measures, formatting, aggregation, etc.
3. Show new visualizations – Heat Grid, Sunburst, Geo Map
4. Show drill-down behavior in several visualizations – highlight which layout fields drill down first, which drill down after (order of operations for drill down)
5. Show how to find specific fields from the field list with the ‘Search Fields’ box in left pane of Data Explorer

pentaho

GD3-2-3: Geo Capabilities

1. Background: Show how auto model guesses at basic geo hierarchies (territory, country, state, city, etc) – and guesses at locations by attaching latitude and longitude fields to the field that immediately precedes them in the PDI data stream
2. Show how to directly configure latitude and longitude properties for a location attribute (such as an address or store) – in Annotate Stream
3. Show drilling down into a geo hierarchy on a map; also show sequential dragging of geo levels onto map with panning/zooming to focus
4. Show example of using Analyzer to visualize a Data Explorer data source on a map – i.e. data published from Data Explorer to PUC

GD3-2-4: Performance Improvements

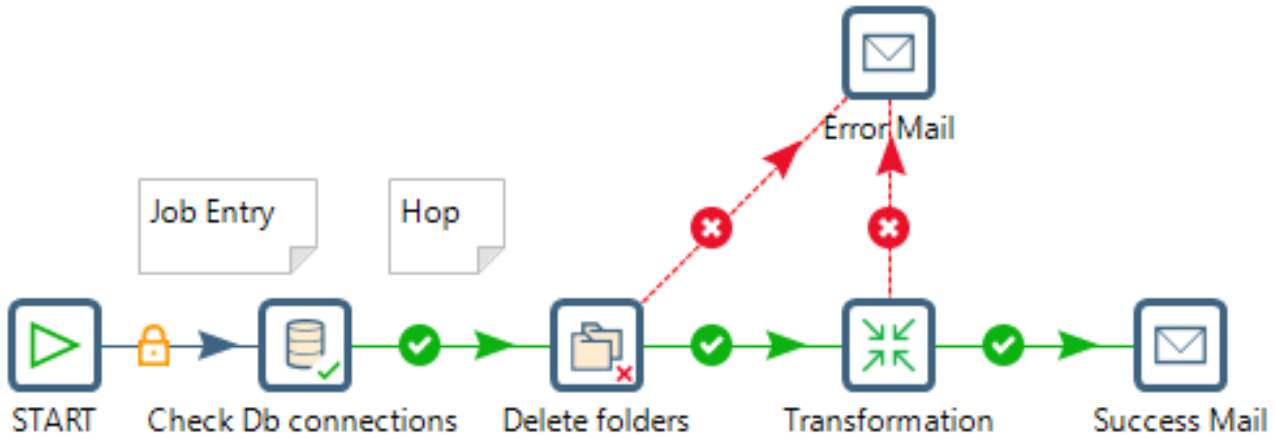
1. Show that it is now possible to view rows incrementally in Data Explorer (no longer to load entire data set first) – enables easier scrolling and jumping through a data set
2. Show that Data Explorer re-opens faster after it has been loaded once
3. Background – Show it is possible to configure row limit, increasing it above 50k; *NOTE – increasing row limit can adversely impact performance*

pentaho

Jobs

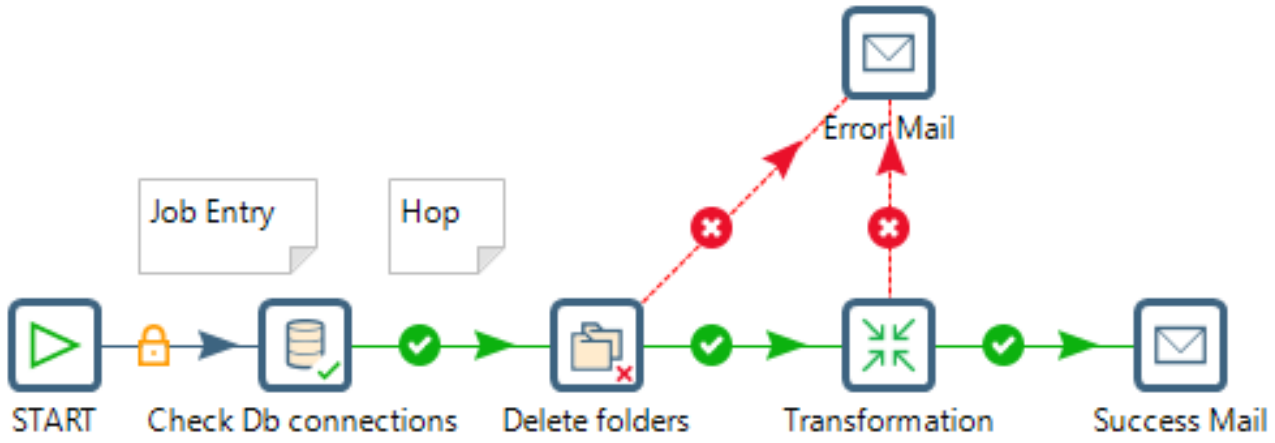
Pentaho Data Integration

Jobs



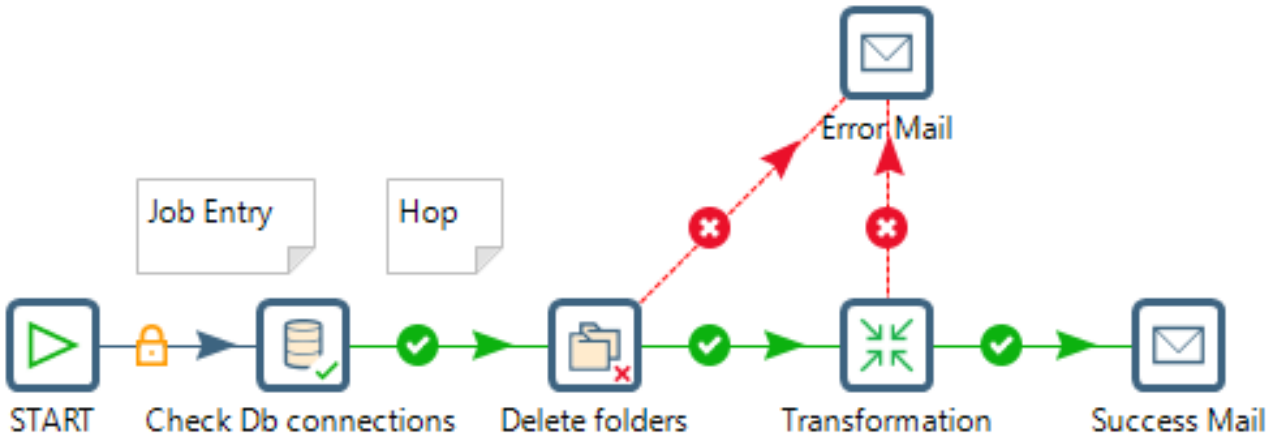
- Jobs are workflow-like models for coordinating resources, execution, and dependencies of ETL activities
- Consist of job entries
- Aggregate individual transformations to a process
- Perform all sorts of maintenance tasks

Jobs



- Core building blocks in a job, each having their distinct functionality
- Pass a result object
  - Can contain rows, but not in streaming fashion, batch mode
- Because job entries are executed sequentially you need to define a starting point
- Can be executed in parallel

Jobs



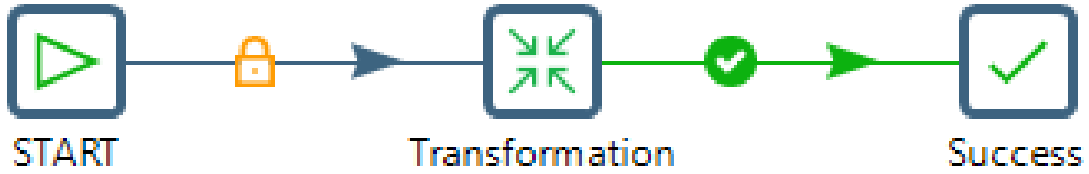
- Define the execution path
- Can be conditional
  - Unconditional
  - Follow when result is true
  - Follow when result is false

Job File Management

- File management
  - HTTP
  - Unzip file
  - Write to file
  - Copy Files
  - Delete file
  - Move Files
  - Delete filenames from result
  - Delete folders
  - File Compare
  - Wait for file
  - Zip file
  - Process result filenames
  - Create a folder
  - Delete files
  - Add filenames to result
  - Convert file between Windows and Unix
  - Create file
  - Compare folders

GD2-4-1: Hello World (Job)

A simple Job that runs the Hello World Transformation.



Transformation: GD2-4-1_hello_world.ktr

- We will come back to Jobs on our Project day.

Summary

- Transformations
  - Steps
  - Transformation Hops
- Guided Demo 2-1-1: Hello World
- Guided Demo 2-1-2: Hello World (Logging)

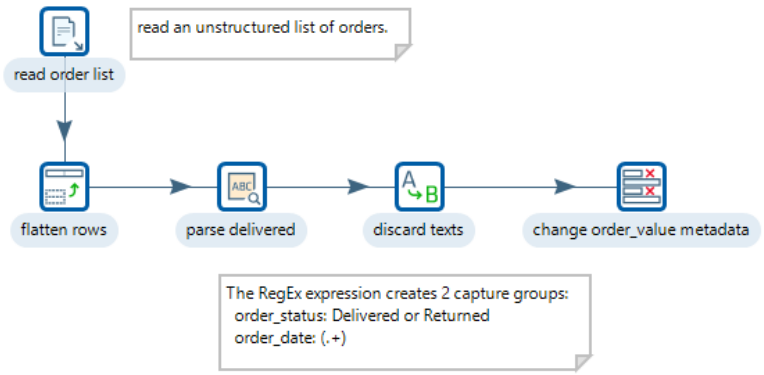
- Parallelism
- Rows of Data
- Data Types
- Guided Demo 2-2-1: Metadata Conversion

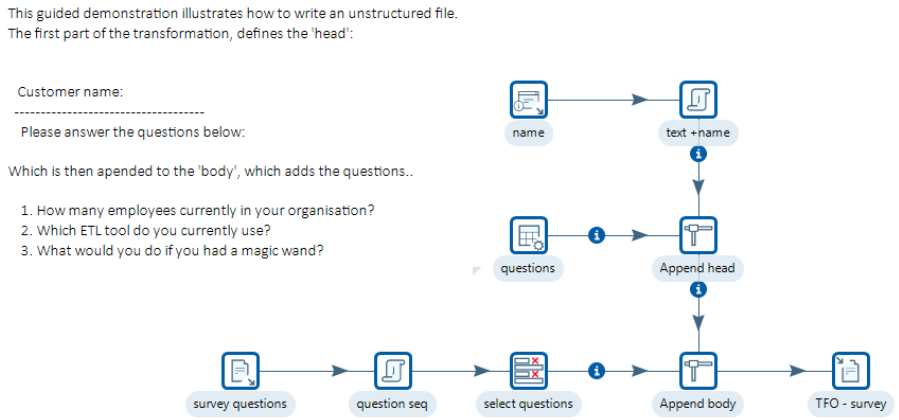
- Jobs
  - Job Entries
  - Job Hops
- Guided Demo 2-3-1: Hello World (Job)

pentaho

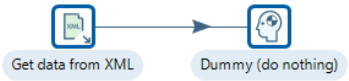
Pentaho Data Integration

Module 3: Data Sources





A simple example of reading an xml file



Reading xml from a URL

Topics

- Working with Files
  - Guided Demo 3-1-1: Text File Input
  - Guided Demo 3-1-2: Text File Output
  - Guided Demo 3-1-3: Write an Excel spreadsheet
  - Guided Demo 3-1-4: Reading XML
  - Guided Demo 3-1-5: Reading JSON
  - Guided Demo 3-1-6: Reading RSS Feed

Text File Input

Write a Text File

Write to Excel

XML

JSON

RSS Feed

Pentaho Data Integration

GD3-1-1: Text File Input

- Create an ETL workflow that will write the data to a database table:
- List of

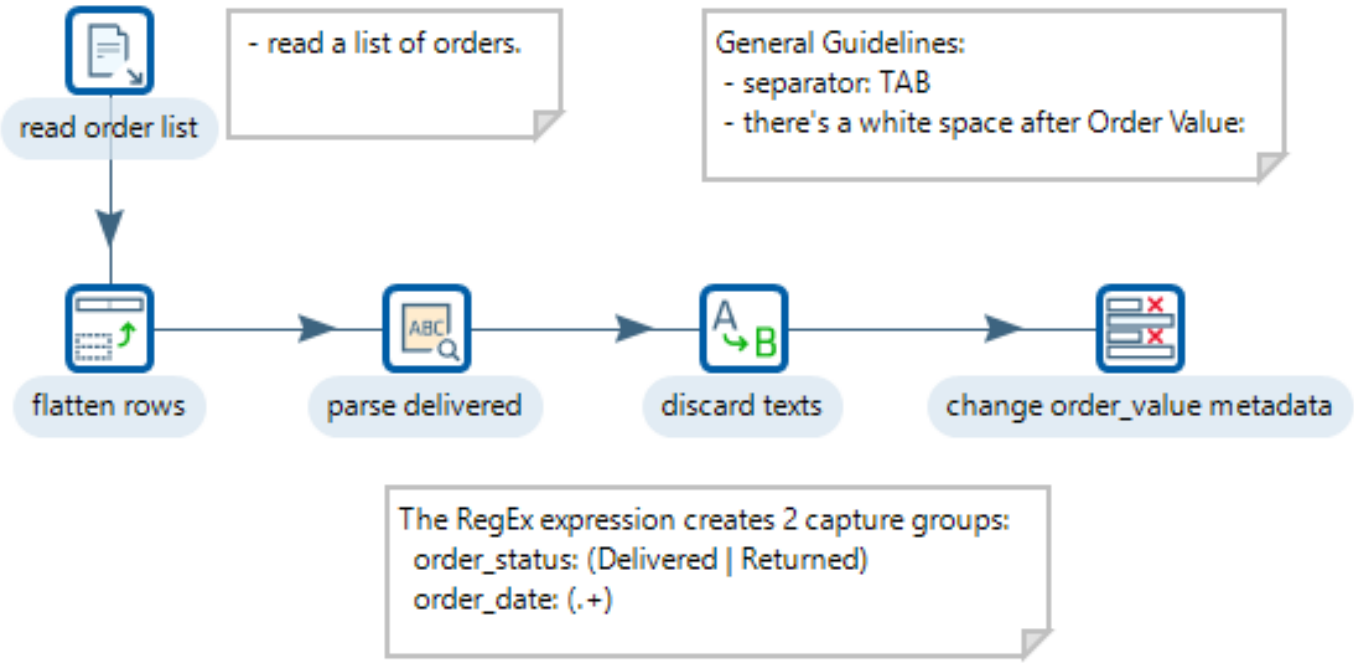
```
Productline: Classic Cars
Customer: Christine Loomis
Delivered: January 2004
Order Value: $21.99

Productline: Classic Cars
Customer: Mary L. Peachin
Delivered: November 2008
Order Value: $24.99

Productline: Trains
Customer: Bob Italia
Delivered: July 1994
Order Value: $14.99

Productline: Planes
Customer: Scott M. Ascher
Delivered:  March 2014
Order Value: $27.99

Productline: Motorcycles
Customer: Monty Halls
Returned: April 2007
Order Value: $29.99

Productline: Trains
Customer: Paul McCallum
Returned: June 2017
Order Value: $34.99

Productline: Boats
Customer: Jill Robinson
Delivered: November 2014
Order Value: $19.99
```

So what approach would you recommend?

Flatten the layout for each data stream column:
Productline Customer Status Order_Value Order_Date

Status can have a value of either: Delivered | Returned

GD3-1-1: Text File Input



GD3-1-2: Write to a Text File

- Create a survey based on questions in a text file.

```
text
----------------------------------------------
Please answer the questions below:
```

Head

Stream Append

```
How many employees currently in your organisation?
Which ETL tool do you currently use?
What would you do if you had a magic wand?
```

Body

So what approach would you take?

In the ‘head’ workflow you have the input for the Customer Name.
In the ‘body’ workflow you have the questions.

Then append the ‘header’ stream to the ‘body’ stream

GD3-1-2: Write to a Text File

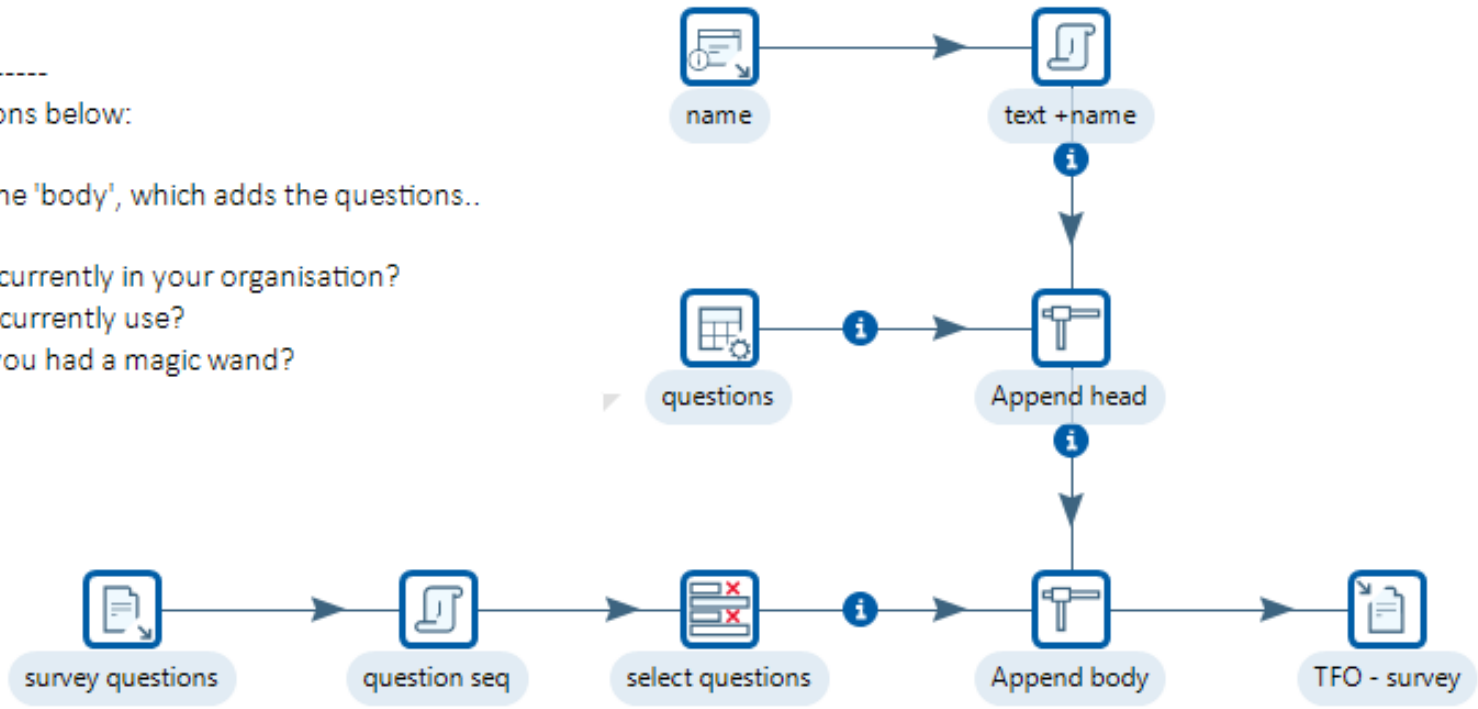
This guided demonstration illustrates how to write an unstructured file.
The first part of the transformation, defines the 'head':

Customer name:

Please answer the questions below:

Which is then apended to the 'body', which adds the questions..

1. How many employees currently in your organisation?
2. Which ETL tool do you currently use?
3. What would you do if you had a magic wand?



GD3-1-3: Writing to Excel

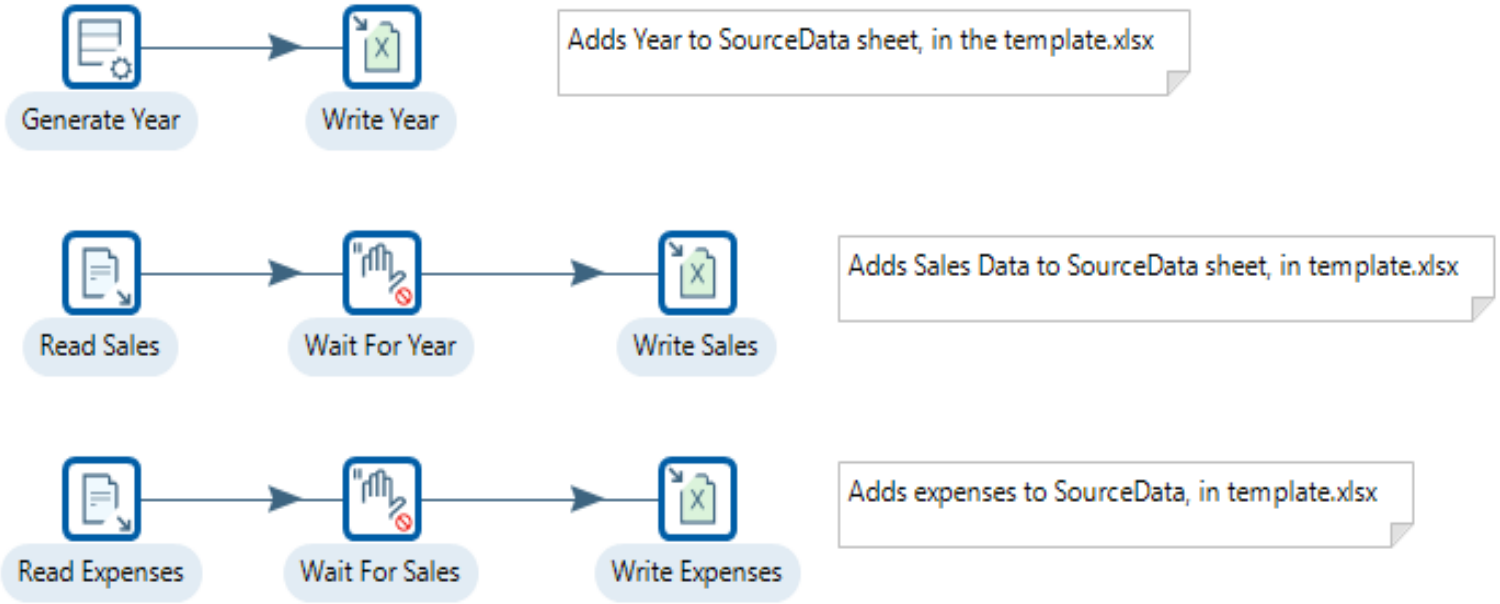
- Steel Wheels wish to automate their Half Yearly Sales and Expenses Report (Excel).
  - Sales and Expenses are text files.
  - Can use a template

	A	B	C	D	E	F	G	H	I	J	K
1	Year	2016									
2											
3		JANUARY	FEBRUARY	MARCH	APRIL	MAY	JUNE	6-MONTH TOTAL	MEAN	MINIMUM	MAXIMUM
4	PRODUCTLINE										
5	Classic Cars	23,455.22€	25,442.11€	24,222.89€	20,233.11€	19,876.22€	19,233.56€	132,463.11€	22,077.19€	19,233.56€	25,442.11€
6	Motorcycles	11,244.21€	12,987.69€	13,954.09€	13,006.33€	13,065.31€	12,087.74€	76,345.37€	12,724.23€	11,244.21€	13,954.09€
7	Trains	1,231.29€	1,227.98€	1,395.33€	1,399.90€	1,335.90€	1,376.98€	7,967.38€	1,327.90€	1,227.98€	1,399.90€
8	Planes	956.12€	834.56€	457.76€	765.32€	898.11€	667.49€	4,579.36€	763.23€	457.76€	956.12€
9											
10	Total Sales	35,655.55€	39,264.36€	38,634.74€	34,004.76€	33,839.64€	31,988.79€	213,387.84€	35,564.64€	31,988.79€	39,264.36€
11		2,055.22€	2,542.11€	2,422.89€	2,033.11€	1,986.22€	1,933.56€				
12	EXPENSES	100.32€	103.23€	140.23€	130.23€	120.33€	121.34€				
13	Advertising	11,020.80€	11,020.80€	11,020.80€	9,350.10€	9,350.10€	12,350.60€	64,113.20€	10,685.53€	9,350.10€	12,350.60€
14	Cost of Goods	223.23€	223.23€	223.23€	223.23€	223.23€	223.23€	1,339.38€	223.23€	223.23€	223.23€
15	Salary	10.30€	0.00€	209.99€	3.99€	0.00€	12.23€	236.51€	39.42€	0.00€	209.99€
16	Lease	90.23€	90.23€	78.90€	90.23€	78.90€	0.00€	428.49€	71.42€	0.00€	90.23€
17	Miscellaneous	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€
18	Overhead	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€	0.00€
19	Total Expenses	11,344.56€	11,334.26€	11,532.92€	9,667.55€	9,652.23€	12,586.06€	66,117.58€	11,019.60€	9,652.23€	12,586.06€
20											
21	PROFIT	24,310.99€	27,930.10€	27,101.82€	24,337.21€	24,187.41€	19,402.73€	147,270.26€	24,545.04€	19,402.73€	27,930.10€

GD3-1-3: Writing to Excel

Guided Demonstration: 3-1-3 Write Excel File

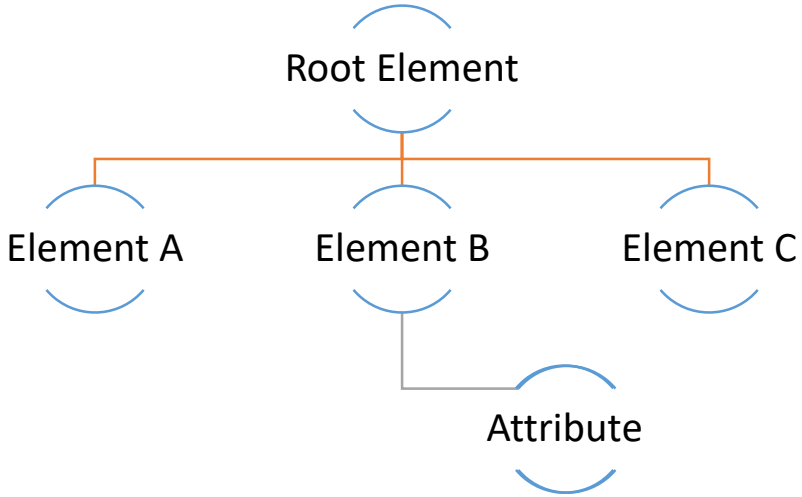
This guided demonstration creates an Excel workbook based on a template that is populated from several Excel spreadsheets, with text files as their datasource.



XML

- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML documents are used to not only store data, but exchange data between systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Hugo</to>
  <from>James</from>
  <heading>Reminder</heading>
  <body>Don't forget to do your homework</body>
</note>
```

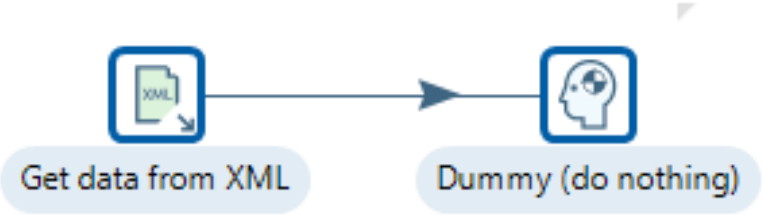


GD3-1-4: Read XML

"Get data From XML" can read data from 3 kind of sources:

- file
- url
- stream

A simple example of reading an xml file

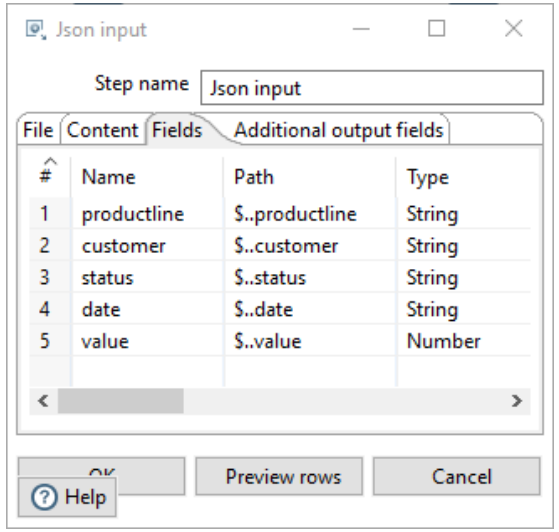


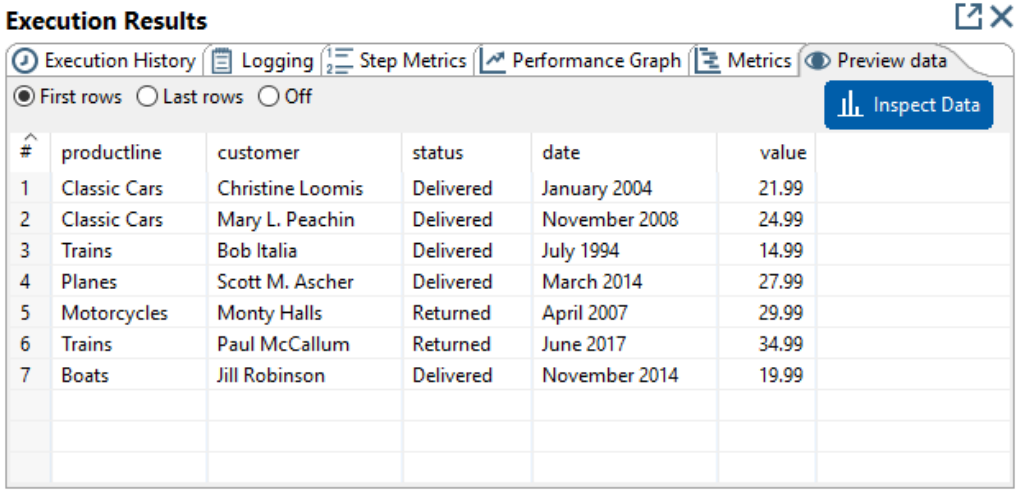
Reading xml from a URL

GD3-1-5: JSON

Read Json file and extract portions data out of structure.


```
{ "document": {
    "order": [
      { "productline": "Classic Cars",
        "customer": "Christine Loomis",
        "status": "Delivered",
           "date": "January 2004",
        "value": 21.99
      },
      { "productline": "Classic Cars",
        "customer": "Mary L. Peachin",
        "status": "Delivered",
           "date": "November 2008",
        "value": 24.99
      },
      { "productline": "Trains",
        "customer": "Bob Italia",
        "status": "Delivered",
        "date": "July 1994",
        "value": 14.99
      },
      { "productline": "Planes",
        "customer": "Scott M. Ascher",
        "status": "Delivered",
        "date": "March 2014",
        "value": 27.99
      },
```



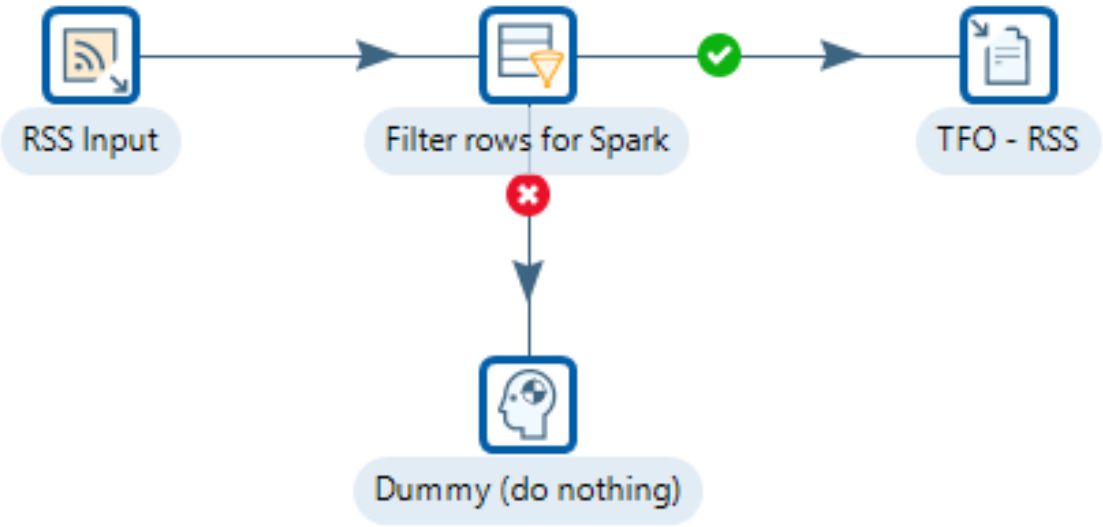
#	productline	customer	status	date	value
1	Classic Cars	Christine Loomis	Delivered	January 2004	21.99
2	Classic Cars	Mary L. Peachin	Delivered	November 2008	24.99
3	Trains	Bob Italia	Delivered	July 1994	14.99
4	Planes	Scott M. Ascher	Delivered	March 2014	27.99
5	Motorcycles	Monty Halls	Returned	April 2007	29.99
6	Trains	Paul McCallum	Returned	June 2017	34.99
7	Boats	Jill Robinson	Delivered	November 2014	19.99

pentaho

GD3-1-6: RSS Feed

- RSS (Rich Site Summary) is a format for delivering regularly changing web content. Many news-related sites, weblogs and other online publishers syndicate their content as an RSS Feed to whoever wants it.
- Filter for ‘Titles that contain Spark’.

Guided Demonstration that illustrates RSS input.



Summary

- Working with Files
  - Guided Demo 3-1-1: Text File Input
  - Guided Demo 3-1-2: Write to a File
  - Guided Demo 3-1-3: Write an Excel spreadsheet
  - Guided Demo 3-1-4: Reading XML
  - Guided Demo 3-1-5: Reading JSON
  - Guided Demo 3-1-6: Reading RSS Feed

Databases

Pentaho Data Integration

Topics

- Working with Databases
- Overview of Steel Wheels Database
- Connecting to Databases
  - Guided Demo 3-3-1: Connect to Database
  - Guided Demo 3-3-2: Reading from a Database
  - Guided Demo 3-3-3: Write to Database
  - Guided Demo 3-3-4: Updating Records
  - Guided Demo 3-3-5: Inserting / Updating Records
  - Guided Demo 3-3-6: Dimension Lookup / Update
  - Guided Demo 3-3-7: Deleting Records
  - Guided Demo 3-3-8: Passing Parameters

pentaho

Steel Wheels

DB Connection

Pentaho Data Integration

Steel Wheels Inc

Steel Wheels buys collectable model cars, trains, trucks, etc, from manufacturers and sells to distributors across the globe.

Overview of Steel Wheels Database

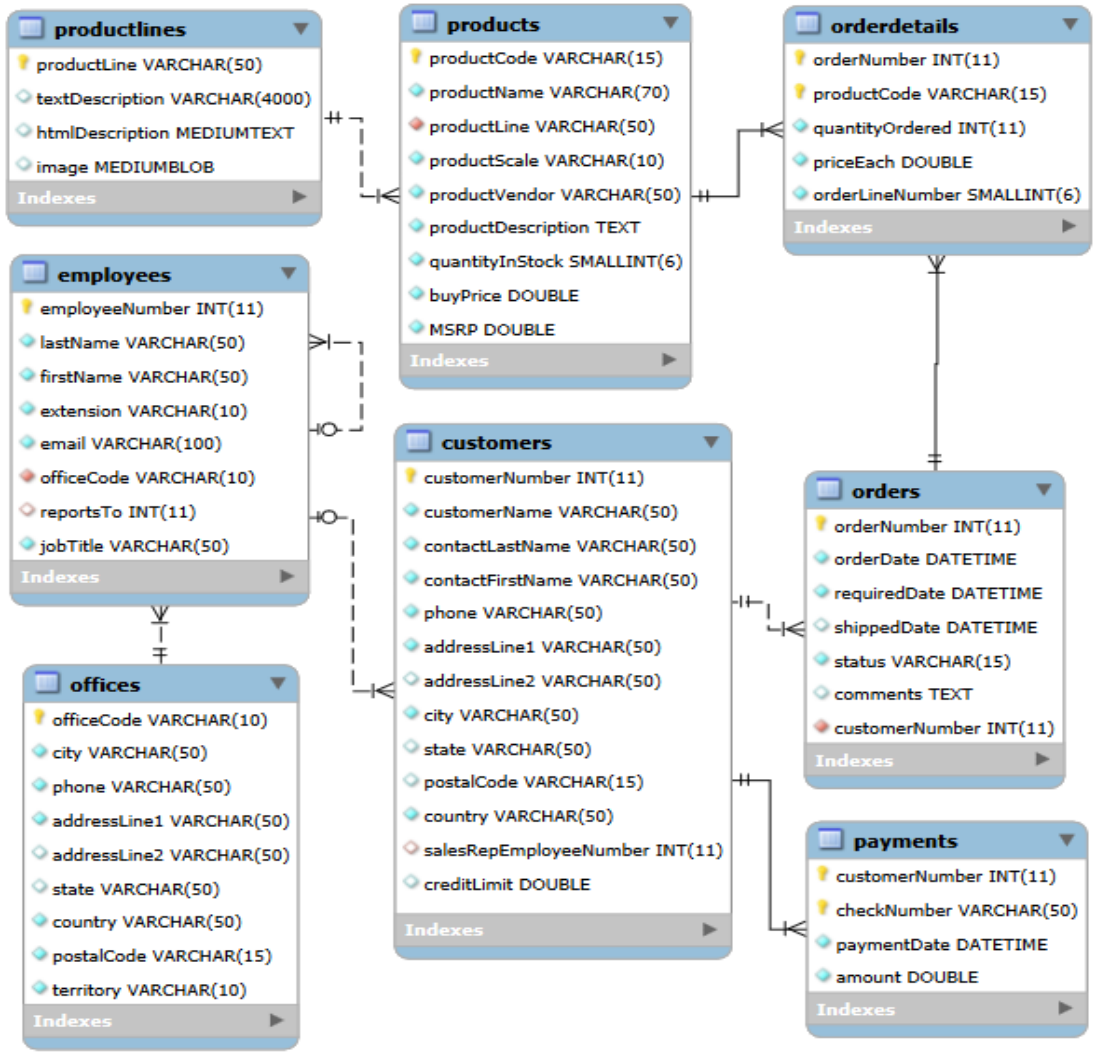


Table	Description
CUSTOMERS	Steel Wheels' customers
EMPLOYEES	All employee information, organization structure such as who reports to whom
PRODUCTS	Products sold by Steel Wheels
PRODUCTLINES	List of product line categories.
OFFICES	Steel Wheels' offices
ORDERS	Information about sales orders
ORDERDETAILS	Sales order line items for each sales order.
PAYMENTS	Payments made by customers based on their accounts.

pentaho

Database Connections

Working file-based

- DB connections are specific to job or transformation

Working repository-based (file, DB or DI)

- DB connections are stored centrally in repository
- Defined connections are readily available to transformations and jobs
- DB connections can be secured

JDBC (Native) Access

- Database drivers must be added to Spoon (\lib directory) and DI server
- Dialect-specific SQL support for listed data sources
- Generic database connection available for non-listed data sources
  - Generic SQL dialect used for SQL-92 compliant data sources

pentaho

Other DB Connections

- ODBC connections are possible
  - ODBC connections must be defined in Windows
  - ODBC connections made via ODBC-JDBC-Bridge
    - Limitations on SQL syntax
    - Slower than JDBC due to the additional layer
- Use a JNDI connection to connect to a data source defined in an application server like Tomcat or JBoss
- Plugin specific access methods are supplied by a specific database driver (like SAP R/3)

📖 The Connection information for the repository is stored in repositories.xml. It is located in the user's home folder at, ~/.kettle/repositories.xml.

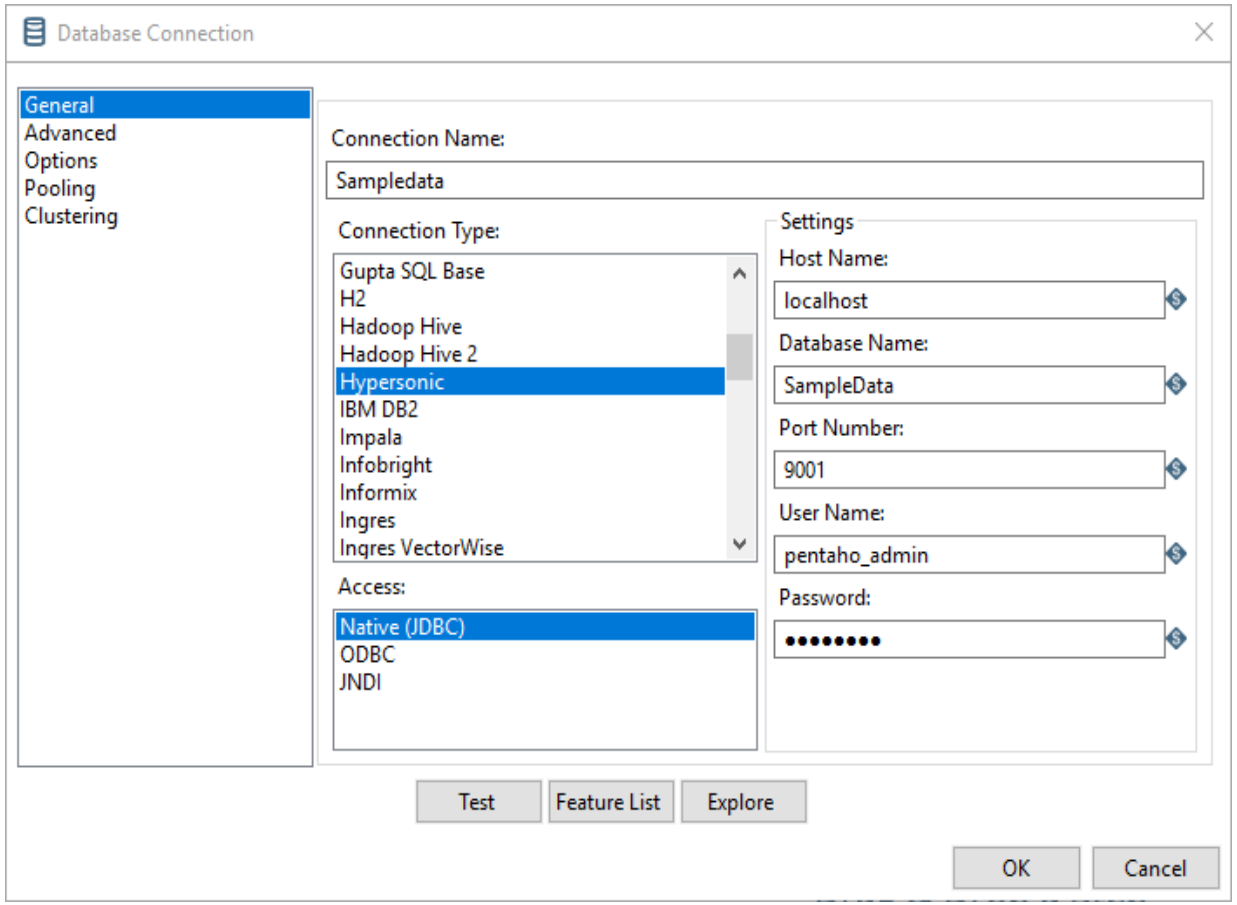
Connecting to Databases

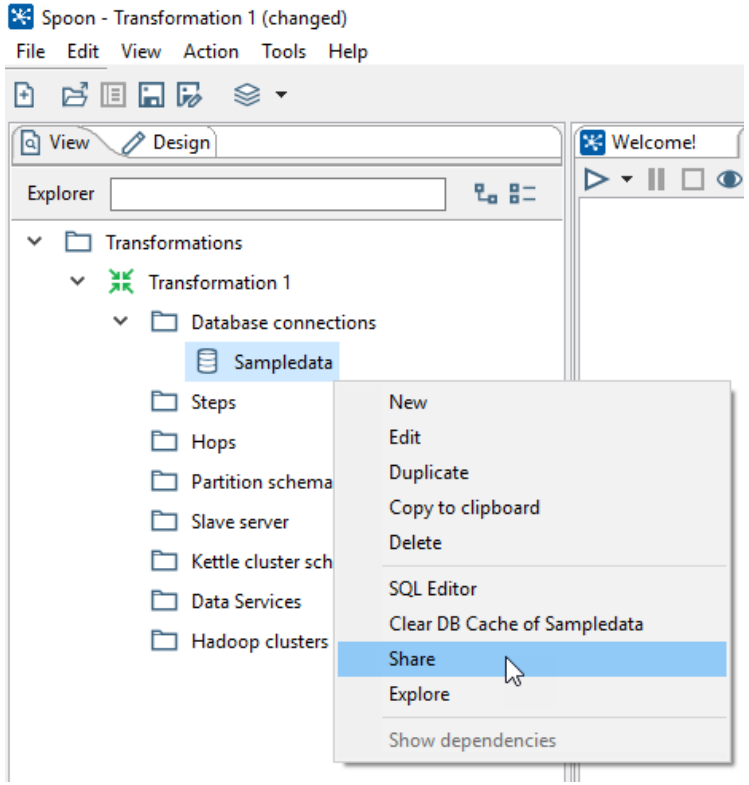
Once you have the correct driver, copy it to these directories.

- Pentaho / DI Server: /pentaho/server/data-integration-server/tomcat/webapps/pentaho/WEB-INF/lib/ .
- Spoon: data-integration/lib

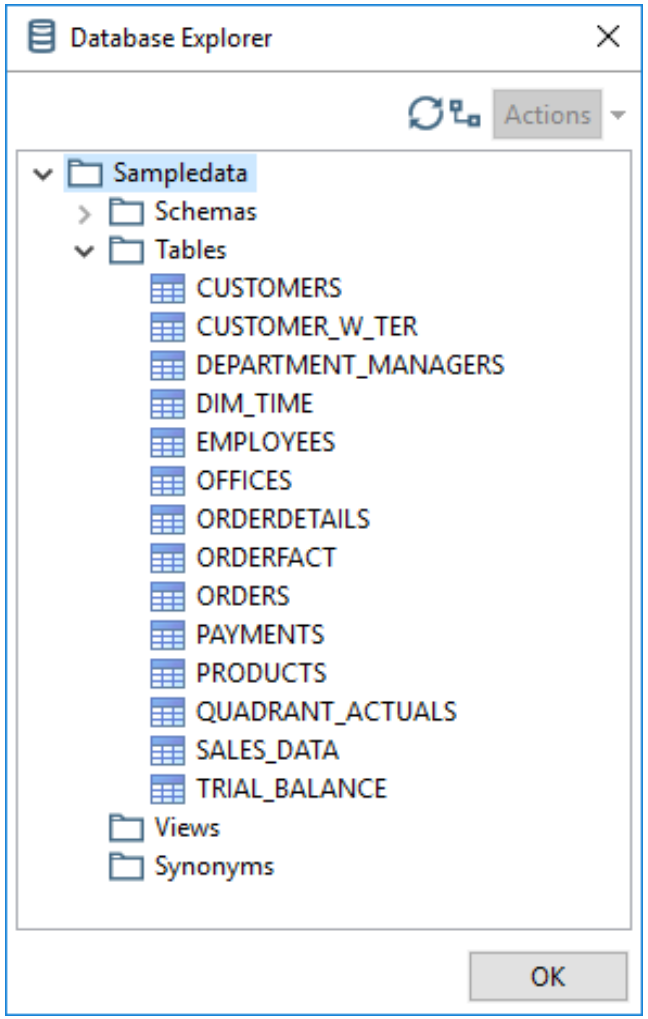
You must restart Spoon for the driver to take effect.

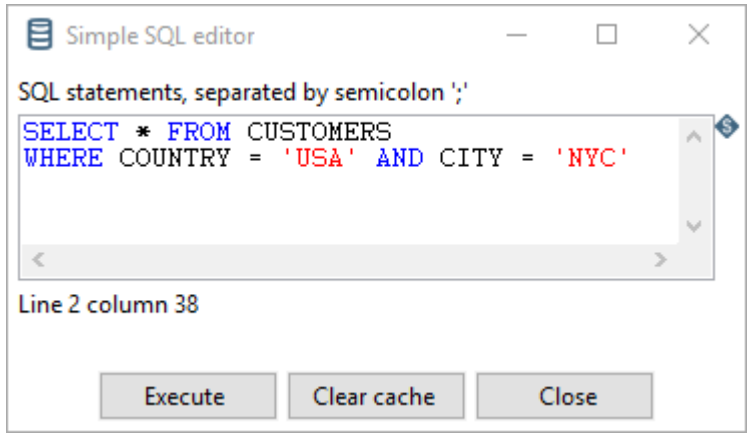
GD3-2-1: Connect to Database





GD3-2-1: Connect to Database





Client Tools:

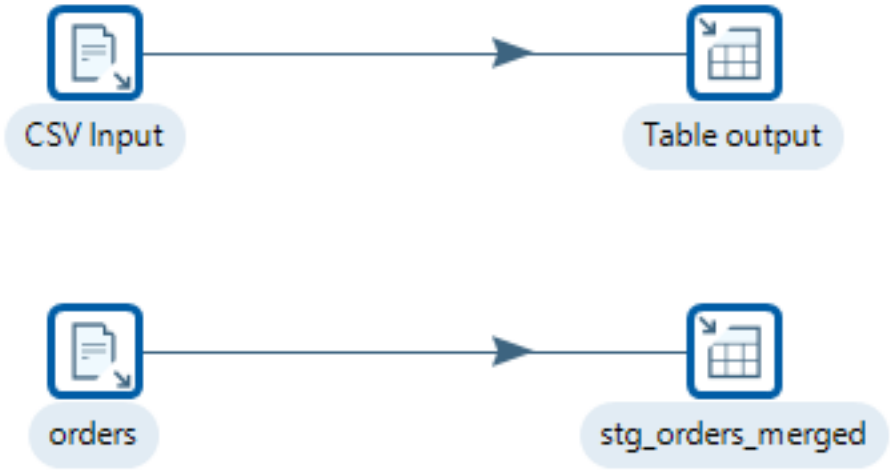
- RazorSQL
- TOAD
- Navicat Premium
- Squirrel (Open source)
- DBeaver (Open Source)

- https://en.wikipedia.org/wiki/Comparison_of_database_tools

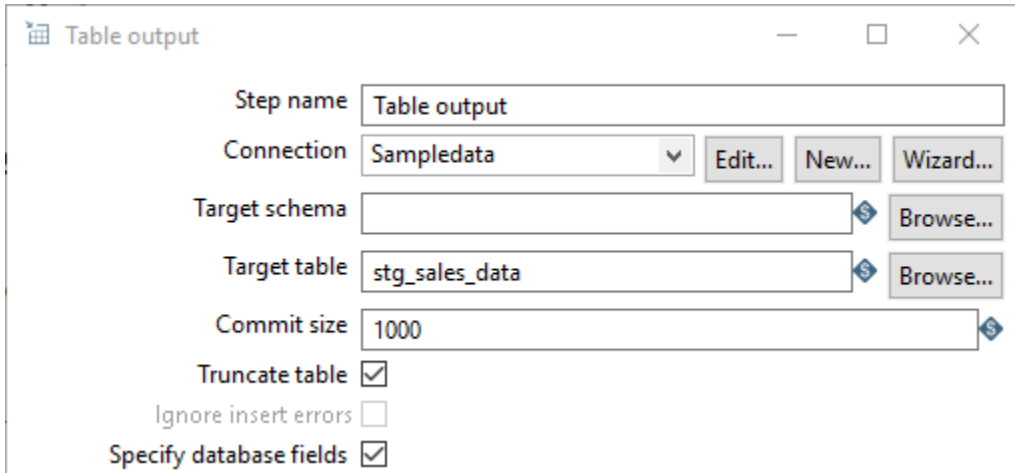
GD3-2-2: Write to Table

- If you work with databases, one of the main objectives will be to extract, load and transform your data. Steel Wheels has several data sources that require loading into a database to discover, cleanse, conform, enrich and validate the data for reports.

Simple transformation that writes the sales_data.csv to a table

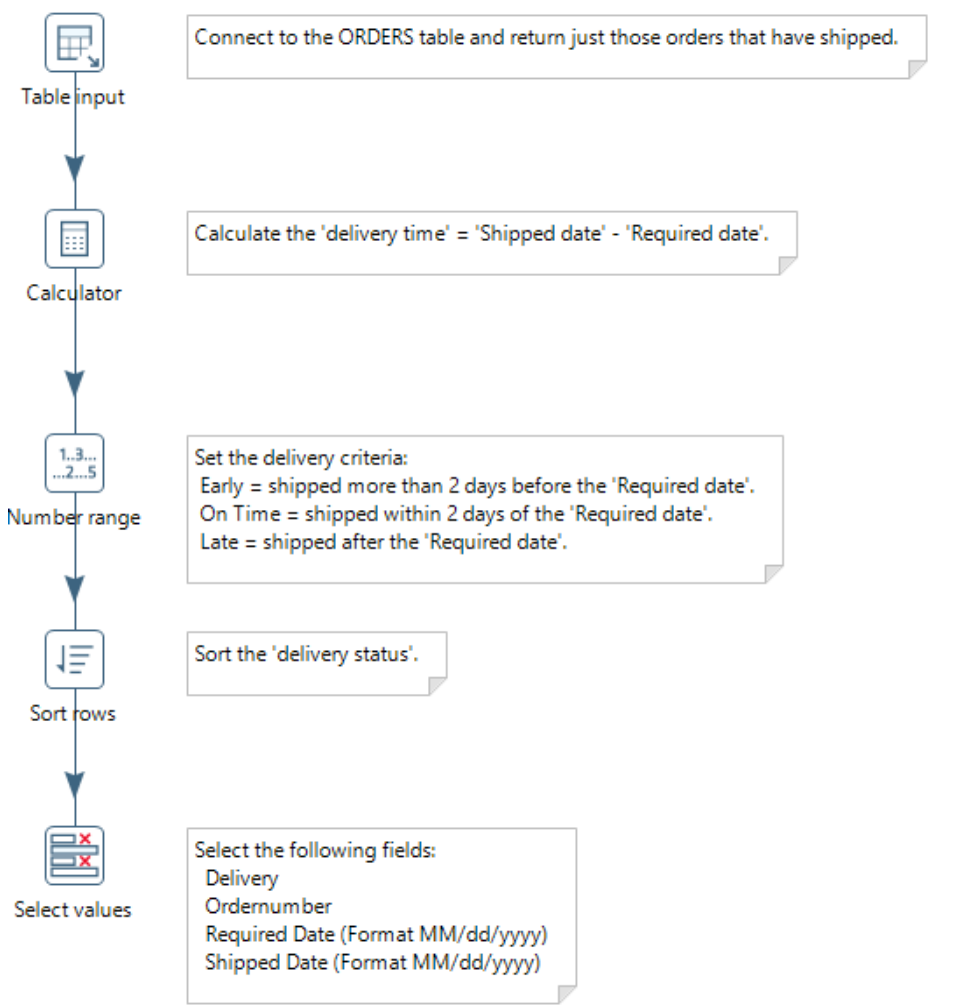


Required for GD 4-1-2 Merged Rows (diff) Database



GD3-2-3: Reading from a Database Table

- So far you have just connected to a database..



Steel Wheels wish to produce a report Tracking the 'Delivery Status' of each order.

#	delivery	ORDERNUMBER	REQUIREDDATE	SHIPPEDDATE
1	Late	10165	10/31/2003	12/26/2003
2	On Time	10121	05/13/2003	05/13/2003
3	On Time	10160	10/17/2003	10/17/2003
4	On Time	10240	04/20/2004	04/20/2004
5	On Time	10251	05/24/2004	05/24/2004
6	On Time	10331	11/23/2004	11/23/2004
7	On Time	10339	11/30/2004	11/30/2004
8	On Time	10358	12/16/2004	12/16/2004
9	On Time	10111	03/31/2003	03/30/2003
10	On Time	10128	06/12/2003	06/11/2003
11	On Time	10133	07/04/2003	07/03/2003
12	On Time	10149	09/18/2003	09/17/2003
13	On Time	10157	10/15/2003	10/14/2003

Updates

Insert / Update

Dimension Lookup / Update

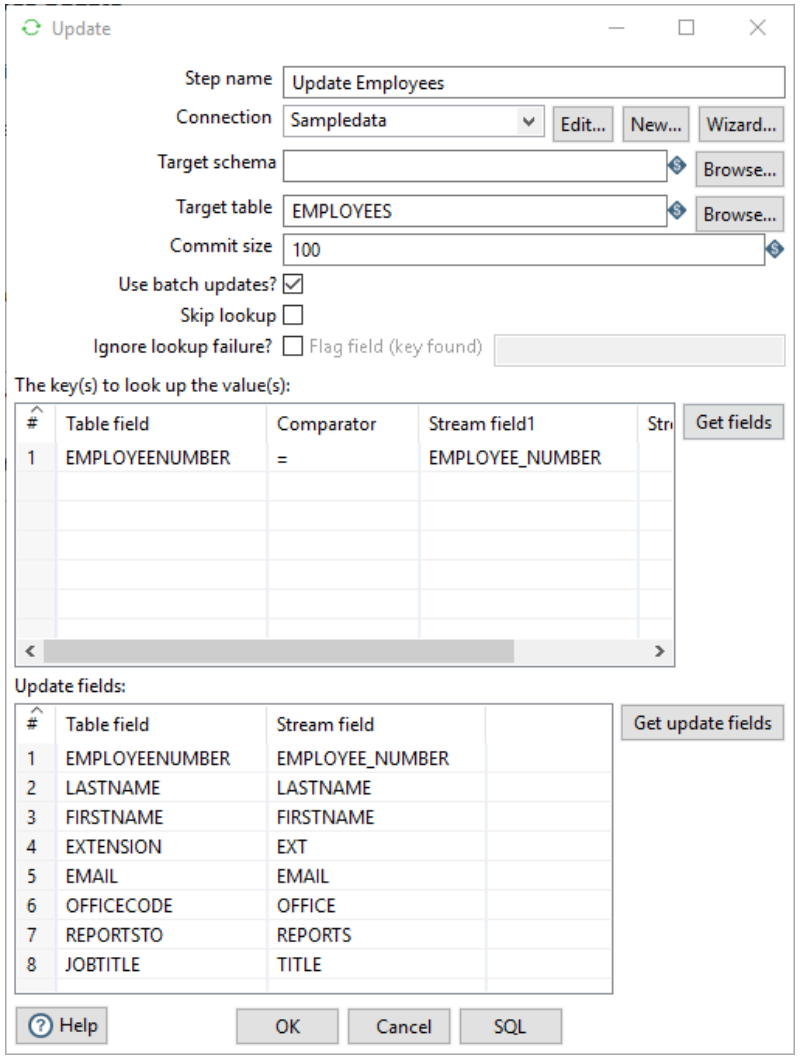
Delete

Pentaho Data Integration

GD3-2-4: Updating Records

The stream fields are mapped to the database table by ensuring:
EMPLOYEE_NUMBER = EMPLOYEENUMBER

Ensure all the Field mappings are correct



GD3-2-5: Inserting / Updating Records

In this demonstration the EMPLOYEE table is updated and new records are inserted.

Inspect the EMPLOYEE Table and compare the records against the employee.txt

The stream fields are mapped to the database table by ensuring:
EMPLOYEE_NUMBER = EMPLOYEENUMBER

Ensure all the Field mappings are correct

Slowly Changing Dimensions

SCD management methodologies referred to as Type 0 through 6. Type 6 SCDs are also sometimes called Hybrid SCDs.

- A type 1 slowly changing dimension is the most basic one and doesn't require any special modelling or additional fields.
  SCD type 1 columns just get **overwritten** with new values when they come into the data warehouse.

- The Type 2 method tracks historical data by creating multiple records for a given natural key in the dimensional tables with separate surrogate key (technical key) and/or different version numbers. With Type 2, we have unlimited history preservation as a new record is inserted each time a change is made.

pentaho

Type I SCD

- **Type 1** - Overwriting the old value. In this method, no history of dimension changes is kept in the database. The old dimension value is simply overwritten with the new one. This type is easy to maintain and is often use for data which changes are caused by processing corrections (e.g. removal special characters, correcting spelling errors).

Before the change:

Customer_ID	Customer_Name	Customer_Type
1	Cust_1	Corporate

After the change:

Customer_ID	Customer_Name	Customer_Type
1	Cust_1	Retail

Type II SCD

- **Type 2** - Creating a new additional record. In this methodology, all history of dimension changes is kept in the database. You capture attribute change by adding a new row with a new surrogate key (technical key) to the dimension table. Both the prior and new rows contain as attributes the natural key (or another durable identifier).
- Also ‘current version' and ‘effective date' columns are used in this method. There could be only one record with current version set to ‘1‘; incrementing everytime a new record is inserted.
- For 'effective date' columns, i.e. start_date and end_date, the end_date for current record usually is set to value 9999-12-31. Introducing changes to the dimensional model in type 2 could be very expensive database operation so it is not recommended to use it in dimensions where a new attribute could be added in the future.

Type II SCD

Before the change:

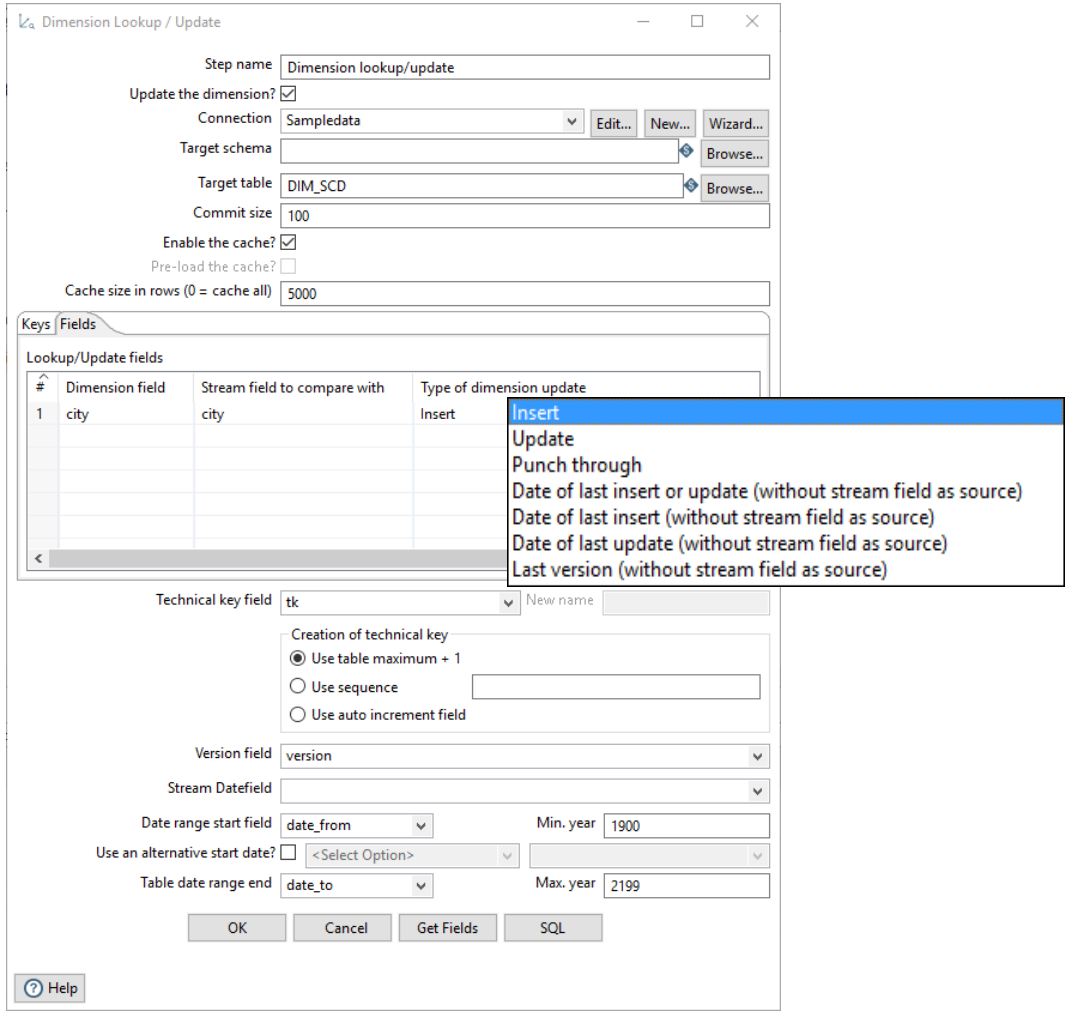
Customer_ID	Customer_Name	Customer_Type	Start_Date	End_Date	Version
1	Cust_1	Corporate	22-07-2010	31-12-9999	1

After the change:

Technical Key	Customer_ID	Customer_Name	Customer_Type	Start_Date	End_Date	Version
1	1	Cust_1	Corporate	22-07-2010	17-05-2012	1
2	1	Cust_1	Retail	17-05-2012	31-12-9999	2

GD3-2-6: Dimension Lookup / Update Type 1

- Operates in 2 modes:



If the Update option is selected, with no 'Stream Datefield', the step operates in Type I Update /Insert mode.

If the Update option is left unchecked, with no 'Stream Datefield', the step operates in Type 1 Lookup mode.

pentaho

GD3-2-6: Type 1 Insert / Update

Add constant rows

Step name: Data Grid

Meta | Data

#	id	city
1	1	London

Help | OK

Dimension Lookup / Update

Step name: Dimension lookup/update

Update the dimension? ☑

Connection: Sampledata | Edit... | New... | Wizard...

Target schema: | Browse...

Target table: DIM_SCD | Browse...

Commit size: 100

Enable the cache? ☑

Pre-load the cache? ☐

Cache size in rows (0 = cache all): 5000

Keys | Fields

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	id	id

Keys | Fields

Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	city	city	Insert

- Insert
- Update
- Punch through
- Date of last insert or update (without stream field as source)
- Date of last insert (without stream field as source)
- Date of last update (without stream field as source)
- Last version (without stream field as source)

pentaho

GD3-2-6: Type 1 Insert /Update

Technical key field: tk
New name:

Creation of technical key
- (●) Use table maximum + 1
- ( ) Use sequence
- ( ) Use auto increment field

Version field: version
Stream Datefield:
Date range start field: date_from
Min. year: 1900
Use an alternative start date? ☐ <Select Option>
Table date range end: date_to
Max. year: 2199

OK | Cancel | Get Fields | SQL

Help

Execution Results

Performance Graph | Metrics

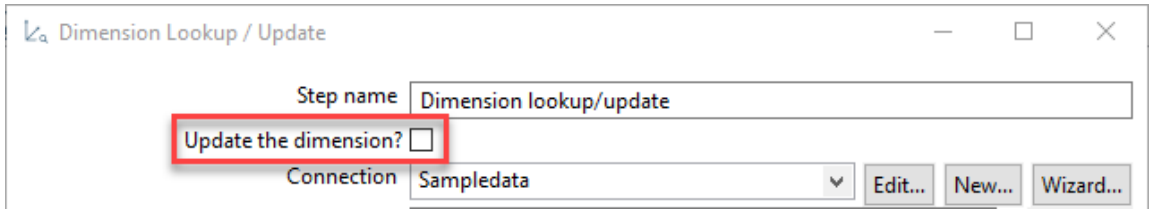
(●) First rows ( ) Last rows ( ) Off — Inspect Data

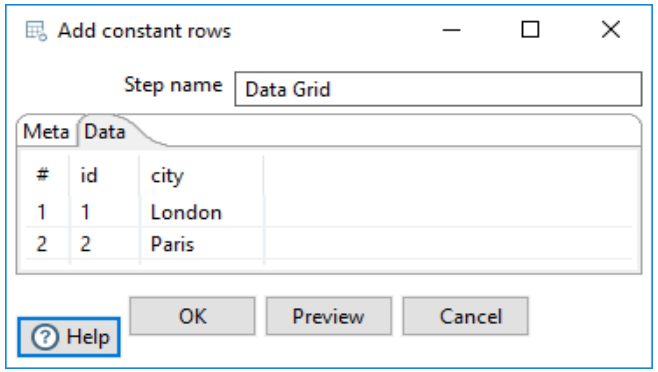
#	id	name	tk
1	1	London	1

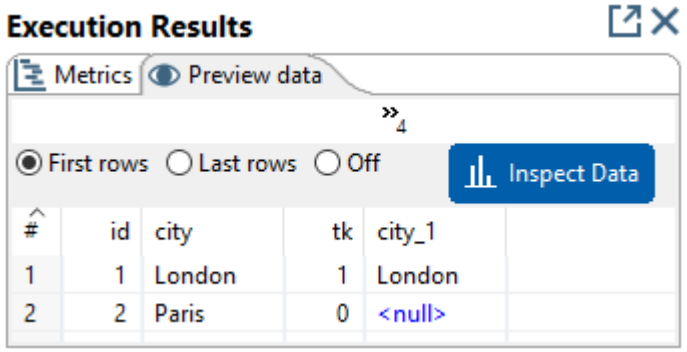
	TK	VERSION	DATE_FROM	DATE_TO	ID	CITY
1	0	1				
2	1	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	1	London

pentaho

GD3-2-6: Type 1 Lookup

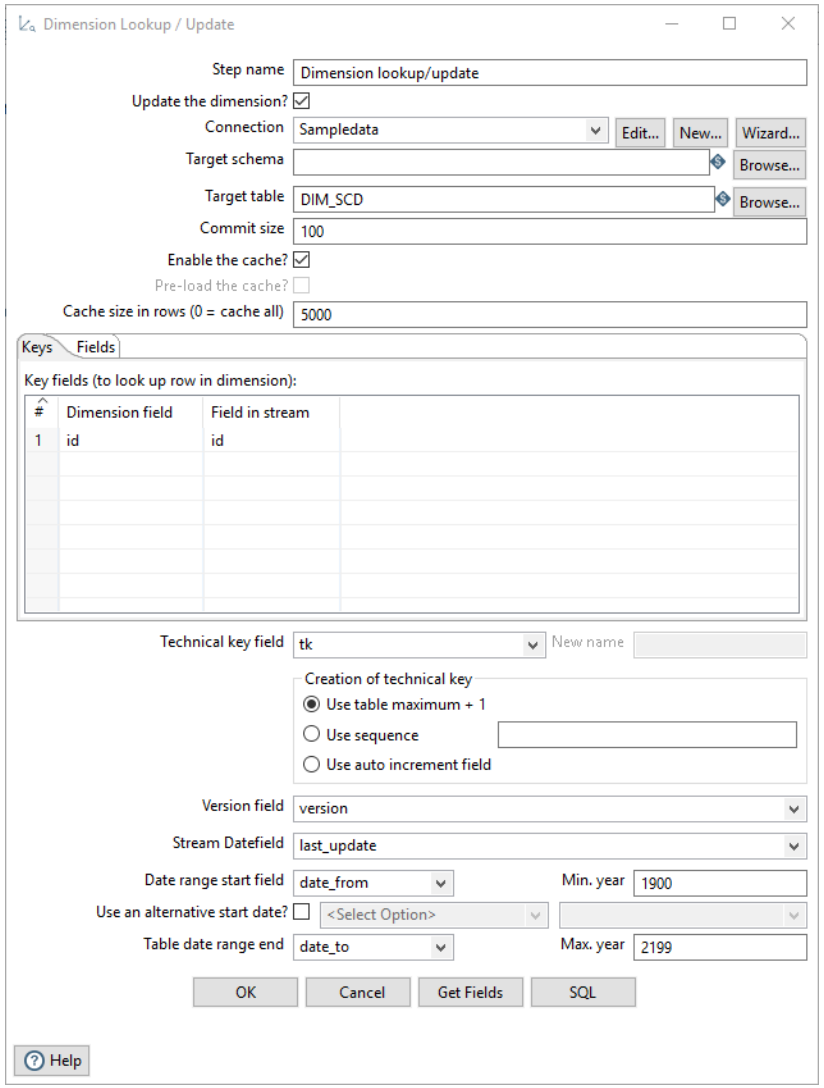






- The record is not written to the table

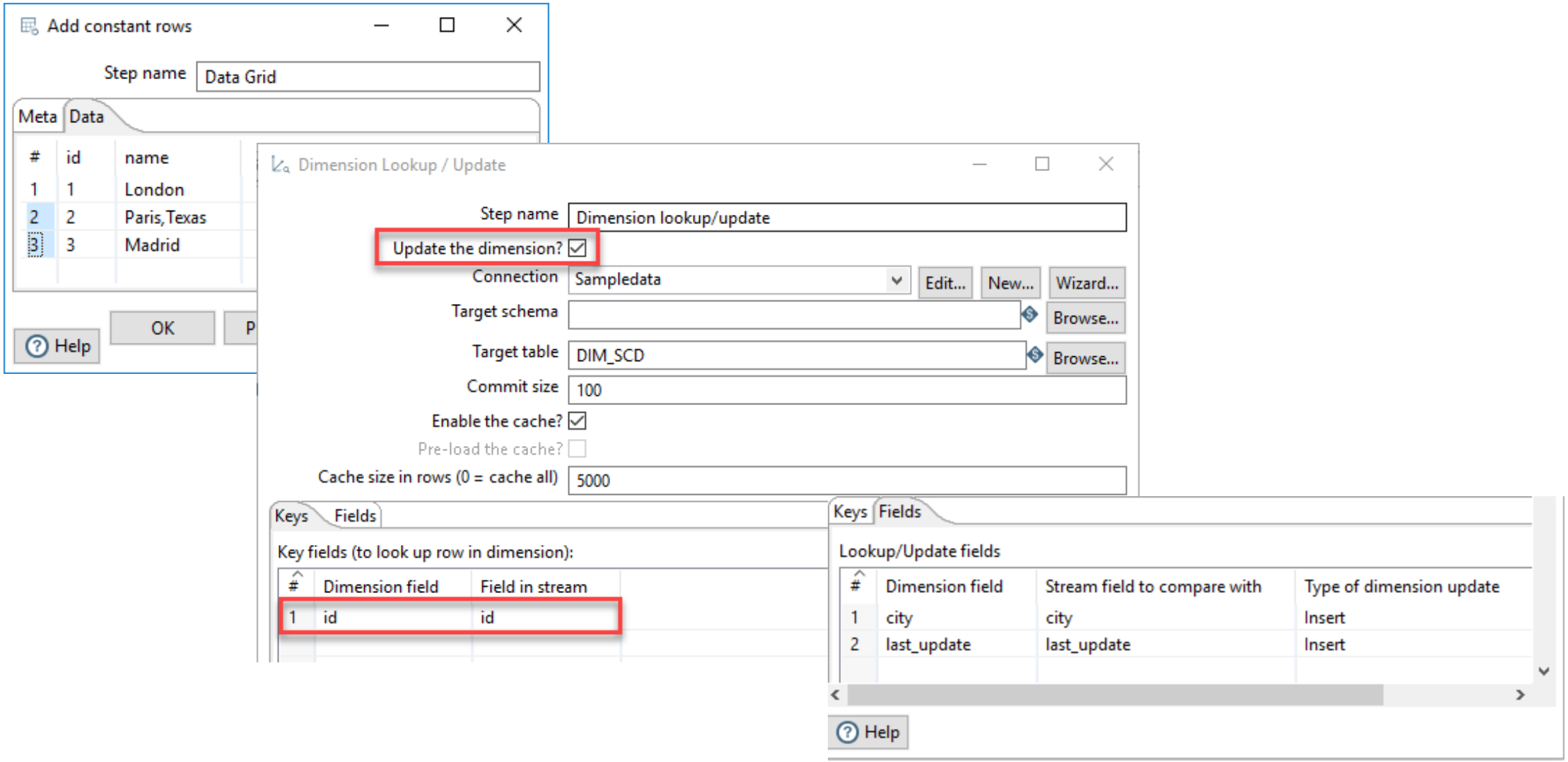
GD3-2-6: Dimension Lookup / Update Type 2



- If the Update option is selected with a Stream Datefield the step operates in Type 2 mode (Update /Insert)
- Historical record is preserved as updating the last_update, forces a new record to be inserted.

pentaho

GD3-2-6: Dimension Insert / Update Type 2



Dimension Insert / Update Type 2

Version field: version

Stream Datefield: last_update

Date range start field: date_from

TK	VERSION	DATE_FROM	DATE_TO	ID	CITY	LAST_UPDATE
0	1					
1	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	1	London	2016-11-24 00:13:13.883000
2	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	2	Paris,Texas	2016-11-24 00:13:13.883000
3	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	3	Madrid	2016-11-24 00:13:13.883000

Add constant rows

Step name: Data Grid

Meta | Data

#	id	name
1	1	London
2	2	Paris
3	3	Madrid

Help | OK

Keys | Fields

Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	city	city	Insert
2	last_update	last_update	Update

Help

TK	VERSION	DATE_FROM	DATE_TO	ID	CITY	LAST_UPDATE
0	1					
1	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	1	London	2016-11-24 00:26:18.652000
2	1	1900-01-01 00:00:00.000000	2016-11-24 00:26:18.652000	2	Paris,Texas	2016-11-24 00:25:24.899000
3	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	3	Madrid	2016-11-24 00:26:18.652000
4	2	2016-11-24 00:26:18.652000	2199-12-31 23:59:59.999000	2	Paris	2016-11-24 00:26:18.652000

pentaho

Delete Columns / Rows

- Sometimes you might have to delete data from a table. If the operation to do it is simple, for example:

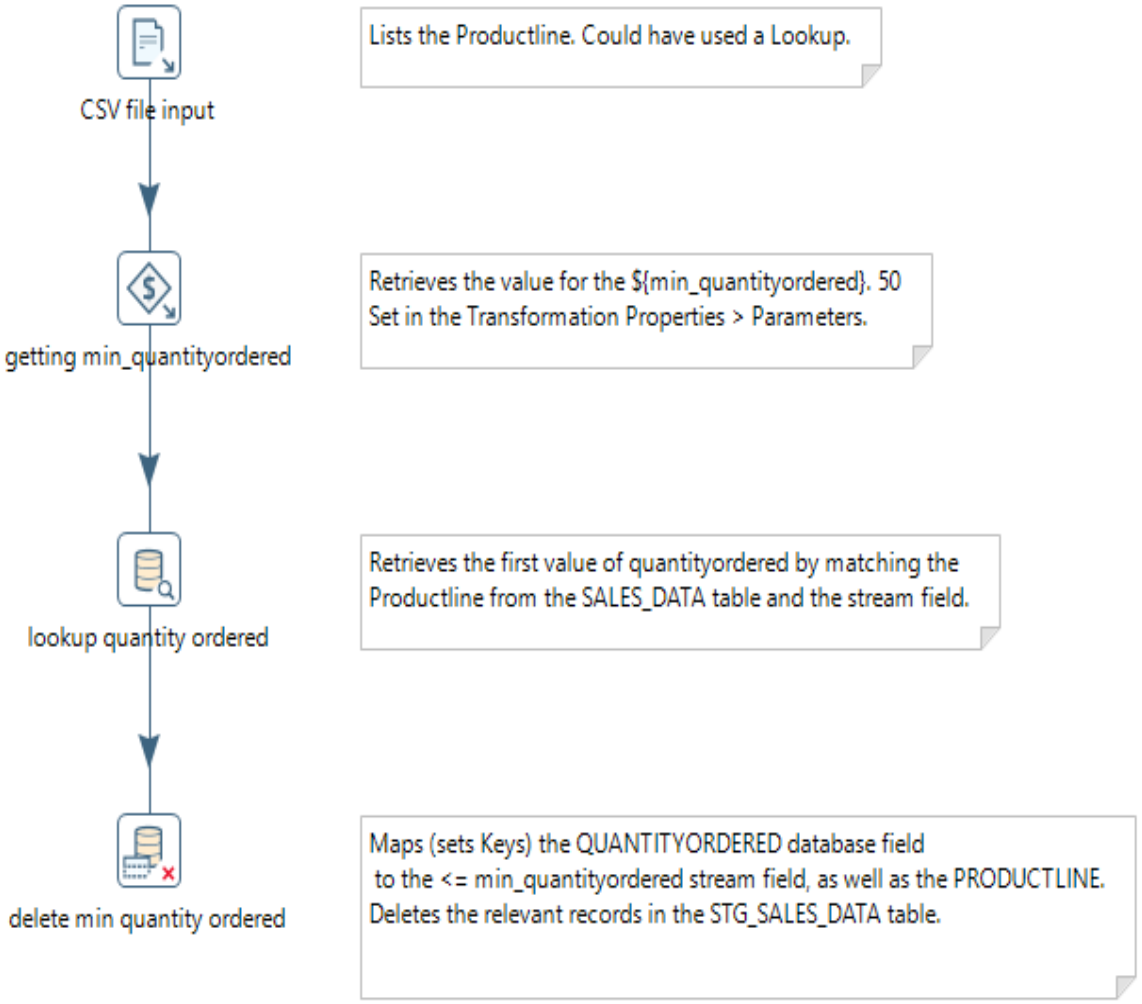
```
DELETE FROM LOG_TABLE WHERE VALID='N'
```

Or

```
DELETE FROM TMP_TABLE
```

- You could simply execute it by using an SQL job entry or an Execute SQL script step. If you face the second of the above situations, you can even use a Truncate table job entry.
- For more complex situations, you should use the Delete step.

GD3-2-7: Delete



- Steel Wheels are launching a campaign, focusing on Customers who have ordered more than 50 of each of their various Productlines.

Parameters

Variables

Pentaho Data Integration

Variables, Parameters

When you Run a Transformation or Job, there are several ways you can define the required settings.

- Variables allow you to dynamically enter the required setting.
- Parameters statically sets the condition of the Transformation or Job.

Parameters

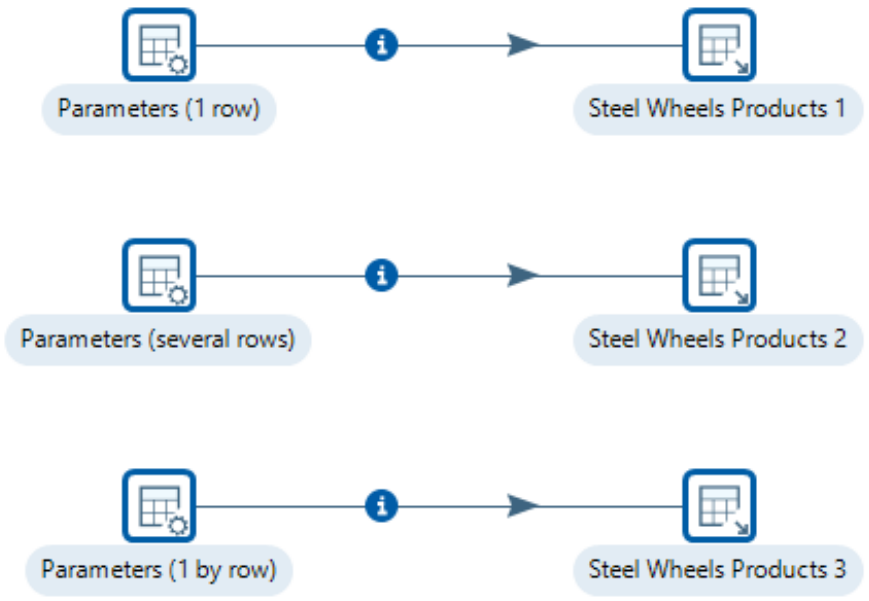
- Named parameters are special in the sense that they are explicitly named command line arguments. If you pass on a lot of arguments to your Kettle Job or Transformation, it might help to assign those values to an explicitly named parameter.
- Named Parameters have following advantages:
  - On the commandline you assign the value directly to a parameter, hence there is zero chance of a mix-up.
  - A default value can be defined for a named parameter
  - A description can be provided for a named parameter
  - No need for an additional transformation that sets the variables for the job

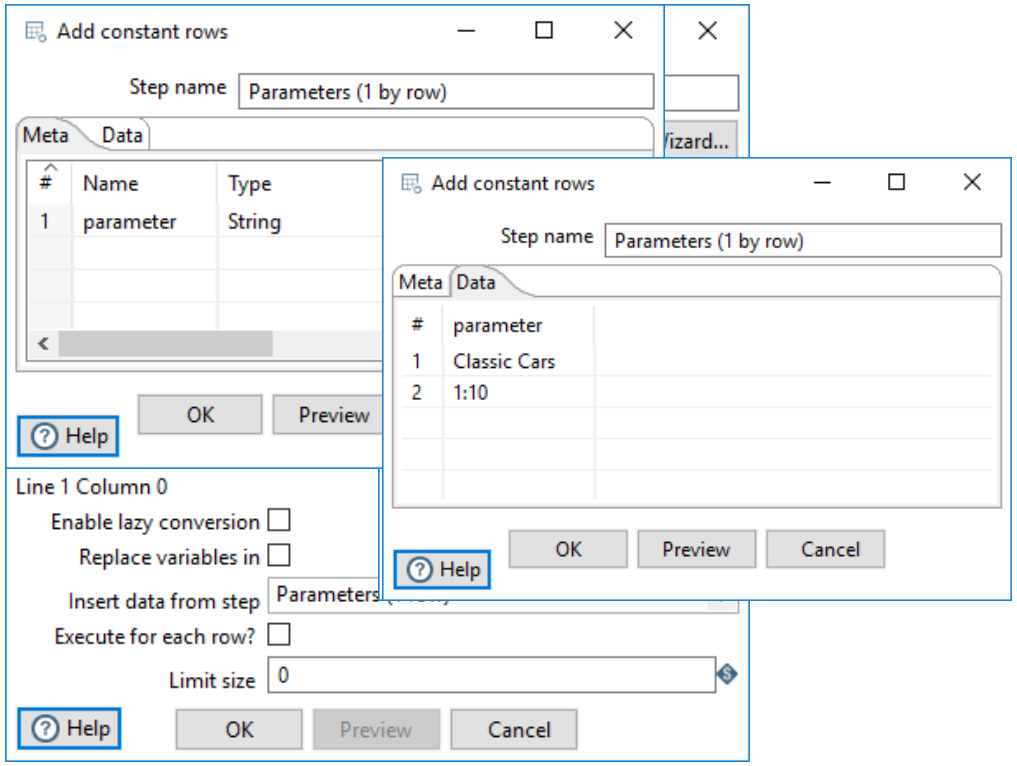
pentaho

GD3-2-8: Parameters

- One of the ways you have to make your queries more flexible is by passing it through some parameters.

This demonstration illustrates the different options for passing parameters.





pentaho

Variables

- Variables can be used throughout Pentaho Data Integration, including in transformation steps and job entries. You define variables by setting them with the Set Variable step in a transformation or by setting them in the kettle.properties file in the directory:
  - $HOME/.kettle (Unix/Linux/OSX)
  - C:\Documents and Settings\<username>\.kettle\ (Windows XP)
  - C:\Users\<username>\.kettle\ (Windows Vista, 7 and later)

- The way to use them is either by grabbing them using the Get Variable step or by specifying meta-data strings like:

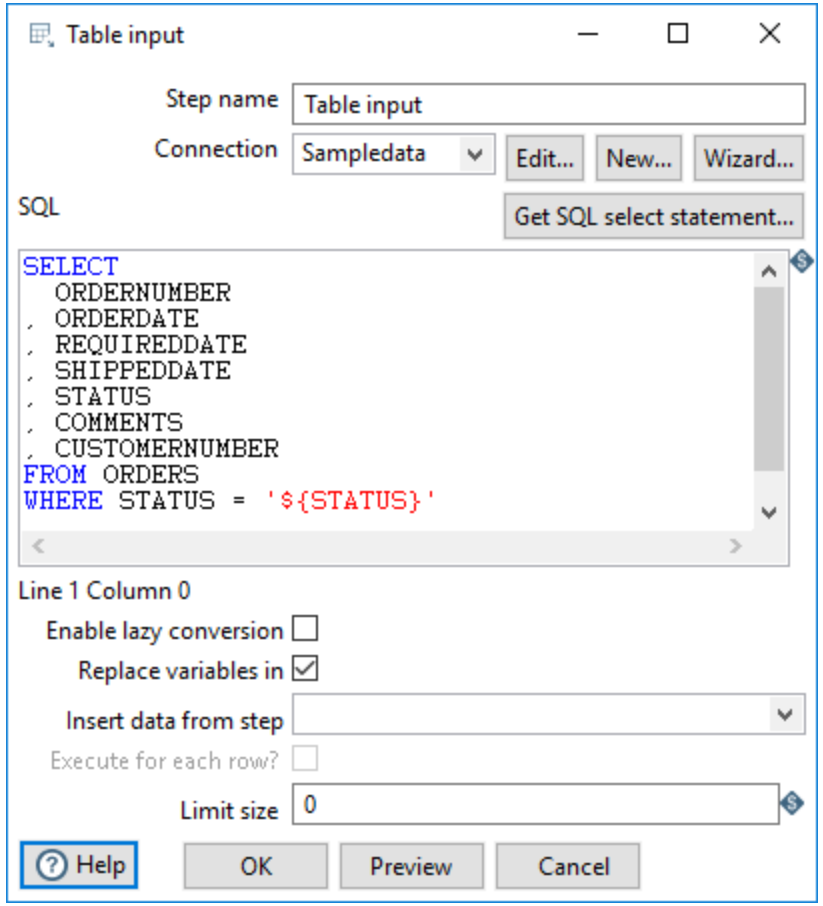
  ${VARIABLE}

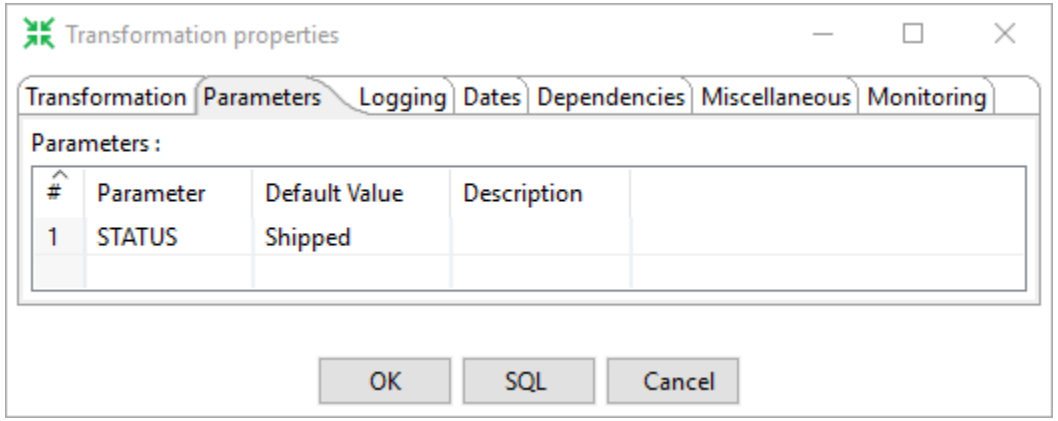
  or:

  %%VARIABLE%%

GD3-2-8: Variables

- So lets modify an existing Transformation ..
- Open GD3-2-3: Reading from a Database Table





Summary

- Working with Databases
- Overview of Steel Wheels Database
- Connecting to Databases
  - Guided Demo 3-3-1: Connect to Database
  - Guided Demo 3-3-2: Reading from a Database
  - Guided Demo 3-3-3: Write to Database
  - Guided Demo 3-3-4: Updating Records
  - Guided Demo 3-3-5: Inserting / Updating Records
  - Guided Demo 3-3-6: Dimension Lookup / Update
  - Guided Demo 3-3-7: Deleting Records
  - Guided Demo 3-3-8: Passing Parameters

pentaho

Pentaho Data Integration

Module 4: Data Enrichment

Topics

- Stream Operations
  - Guided Demonstration 4-1-1: Merge rows
  - Guided Demonstration 4-1-2: Merge rows (diff)

- Lookups
  - Guided Demonstration 4-2-1: Database Lookups

- Joins
  - Guided Demonstration 4-3-1: Join rows
  - Guided Demonstration 4-3-2: Merge Join
  - Guided Demonstration 4-3-3: Database Join

Merge Rows

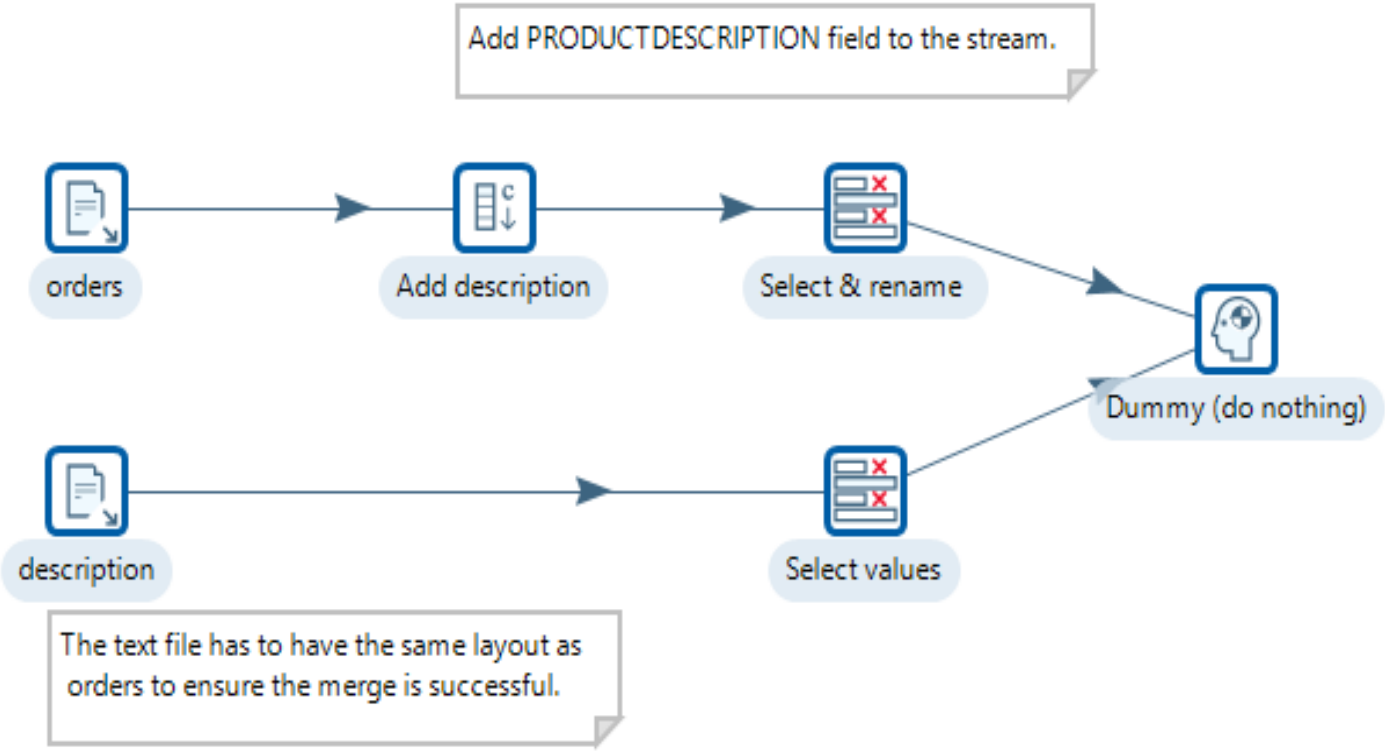
Merge Rows (diff)

Pentaho Data Integration

Topics

- Stream Operations
  - Guided Demonstration 4-1-1: Merge rows
  - Guided Demonstration 4-1-2: Merge rows (diff)

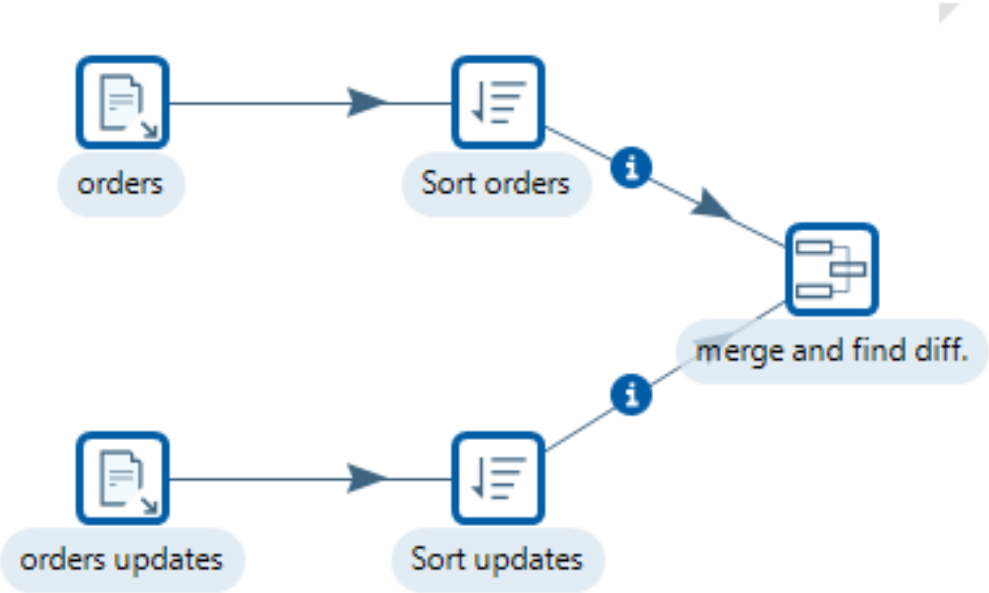
GD4-1-1: Merge Streams



- Each data stream must have the same layout / structure

GD4-1-2: Merge Rows (diff)

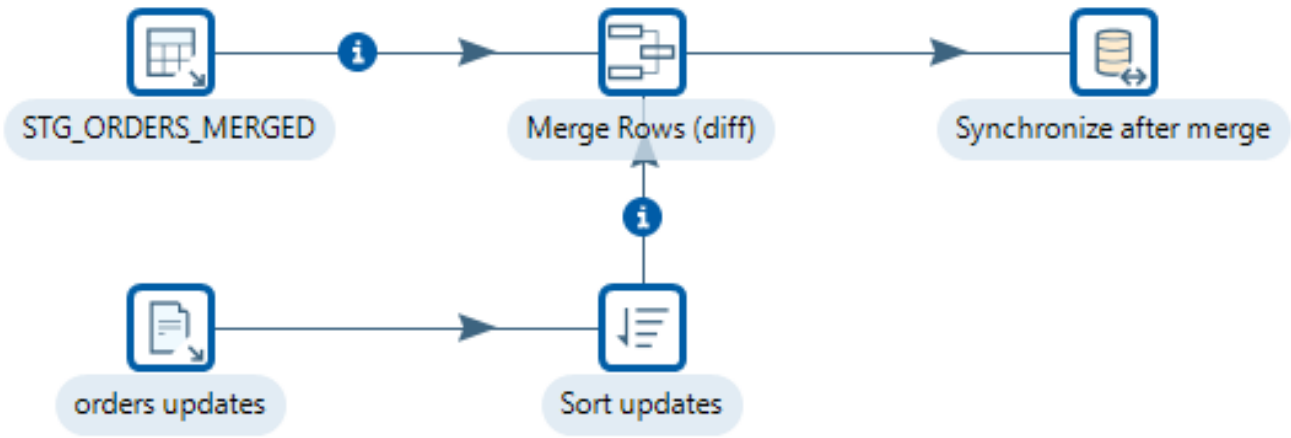
The streams are mergred and the difference 'flagged' as:
identical
changed
new
deleted



- Depending on which stream fields are compared, then the 'flag' indicates:
  - **identical -** both streams values are the same
  - **changed** – if the stream field value has changed.
  - **new** - new record in update stream
  - **deleted** - if the stream field value exists in the 'original' stream, but not in the 'update' stream, it will be flagged deleted.

GD4-1-2: Merge Rows Database (diff)

If the Merge rows (diff) step is used in conjunction with the Synchronize after merge, then based on the value of the flag, the database can be updated.



- The advanced tab of the Synchronize after merge allows various database operations based on the flag value.

Join Rows

Merge Join

Database Join

Pentaho Data Integration

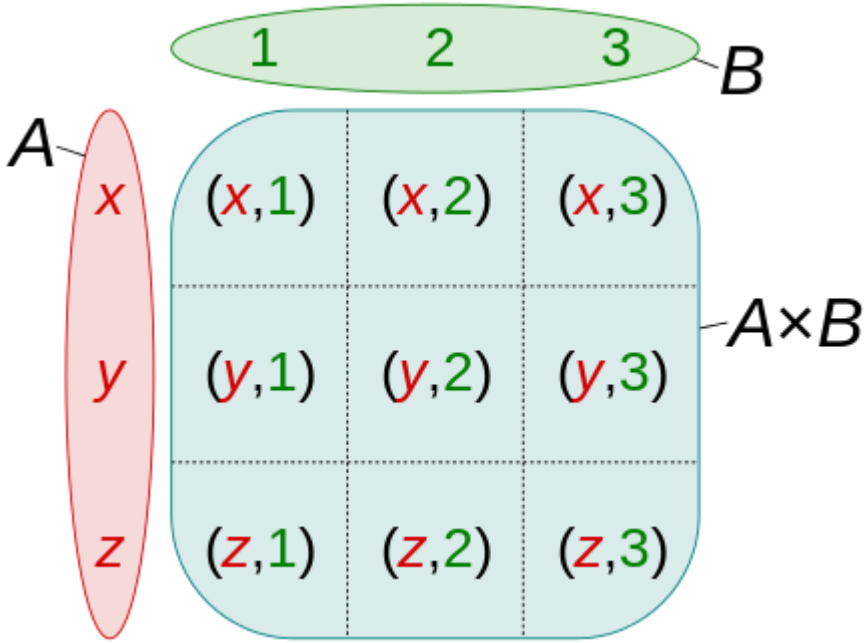
Topics

- Join Rows
  - Guided Demo 4-2-1: Join Rows – Baby name Selector
- Merge Join
  - Guided Demo 4-2-2: Merge Join - Overview
  - Guided Demo 4-2-2: Merge Join – Orders
- Database Join
  - Guided Demo 4-2-3: Database Join - Conditions

pentaho

Join Rows (Cross | Cartesian)

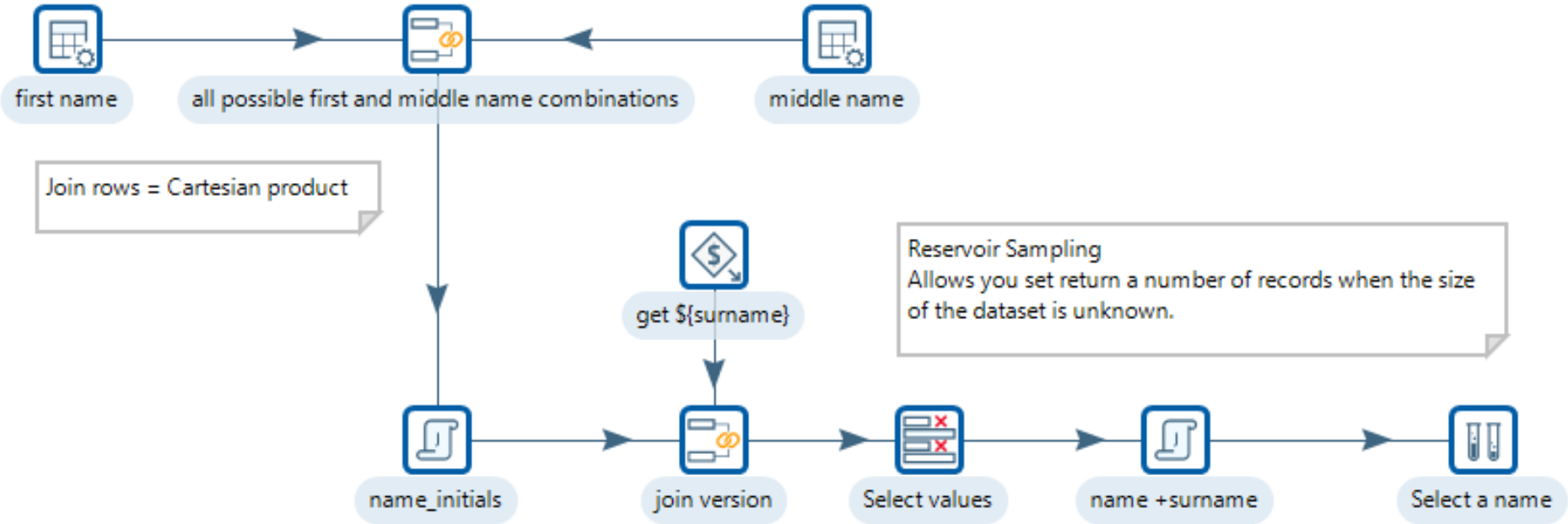
- The Join rows step allows you to produce combinations (Cartesian product) of all rows in the input streams. This normally happens when no matching **join** columns are specified. For example, if table A with 10 rows is joined with table B with 10 rows, a (**Cross) Join** will return 100 rows (Cartesian Product).
- An alternative with a higher performance in most cases is the Merge Join step.



9 Rows

GD4-2-1: Join Rows (Baby Name Selector)

This demo is for fun...!!
Your instructor will take you through the various steps. Notes are also in the manual.



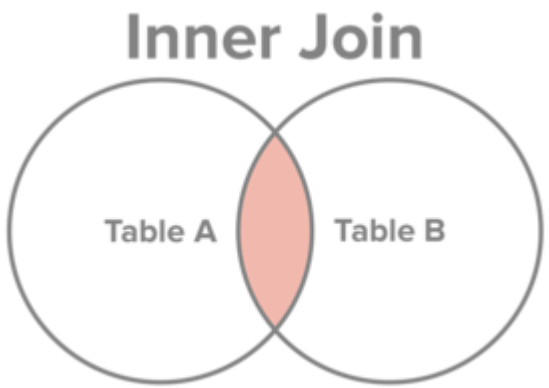
Merge Join

- Merge join step performs a merge join between data sets using data from two different input steps.
  - Join options: INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER

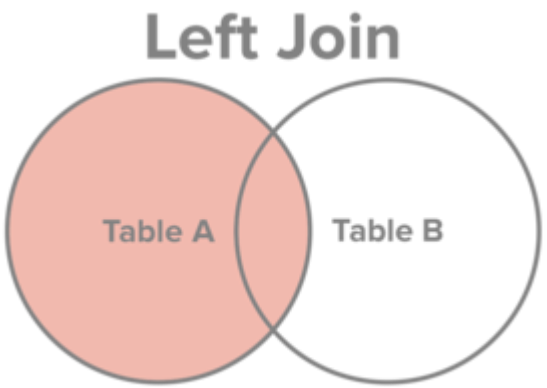
- **Options include**:
  - **Step name:** Unique name of step
  - **First Step**: First input step to the merge join
  - **Second Step:** Second input step to the merge join
  - **Join Type:** INNER, LEFT OUTER, RIGHT OUTER, or FULL OUTER
  - **Keys for 1st step:** Key fields on which incoming data is sorted
  - **Keys for 2nd step:** Key fields on which incoming data is sorted

pentaho

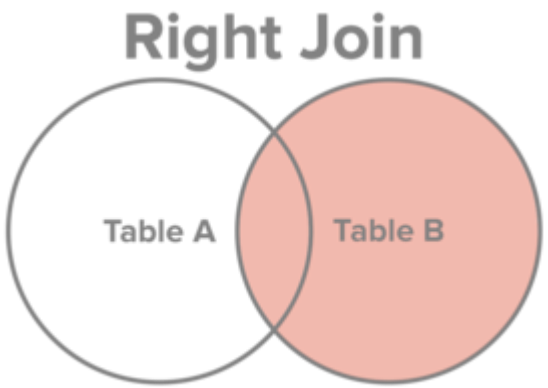
Examples of SQL Joins



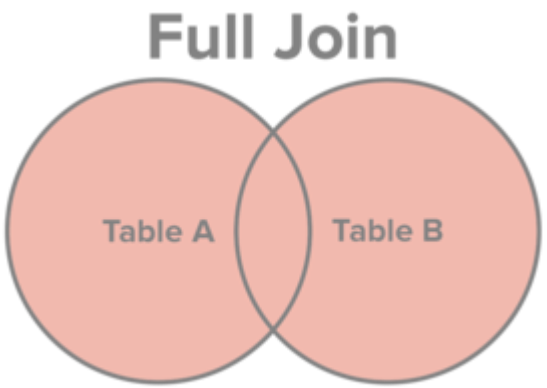
Select all records from Table A and Table B, where the join condition is met.



Select all records from Table A, along with records from Table B for which the join condition is met (if at all).



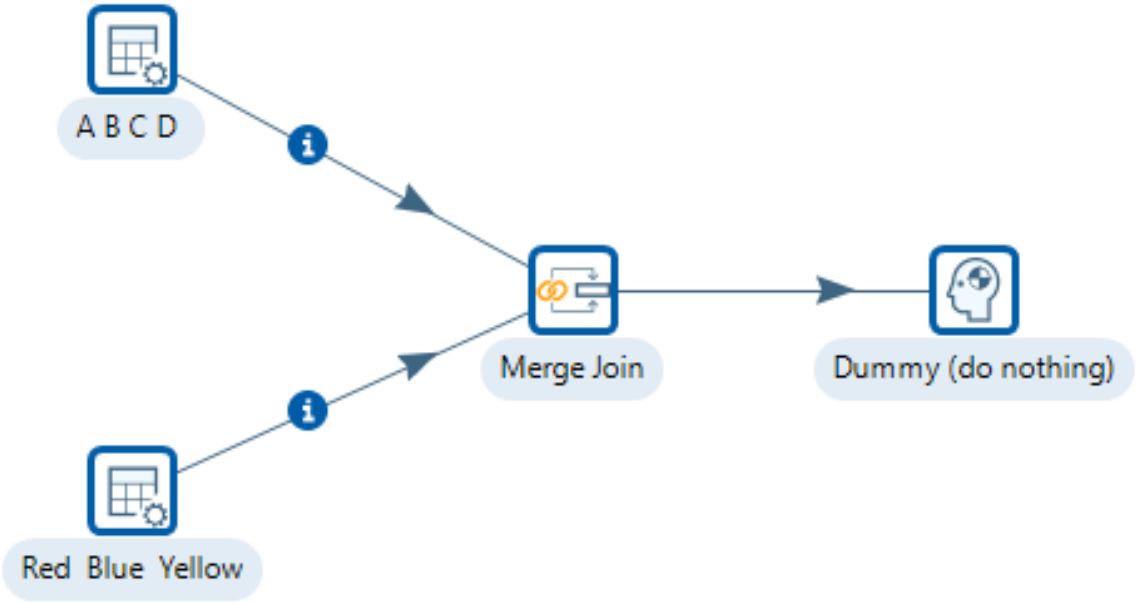
Select all records from Table B, along with records from Table A for which the join condition is met (if at all).

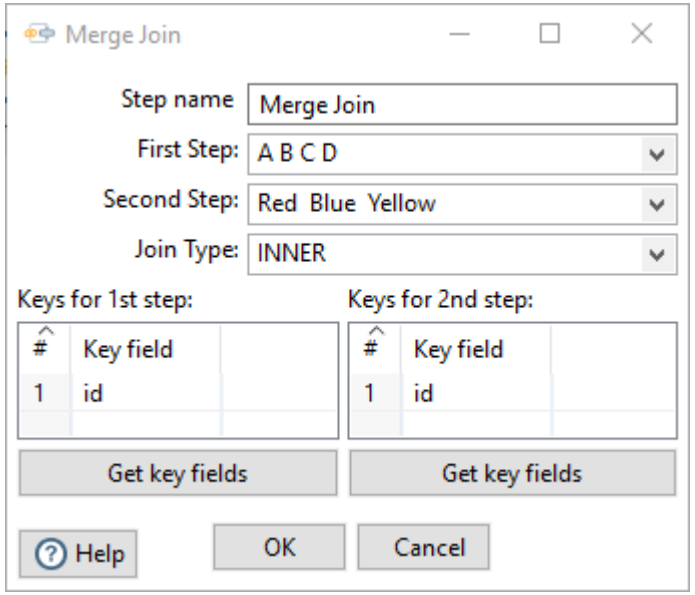


Select all records from Table A and Table B, regardless of whether the join condition is met or not.

GD4-2-2: Merge Join - Overview

Simple merge join demonstration.





Merge Join refresher..

Table a

ID	Value
1	A
2	B
3	C
4	D

Table b

ID	Value
1	Red
3	Blue
5	Yellow

a	b
A	Red
C	Blue

A regular INNER join between two tables will produce a result set with only the common values from both tables.

a	b
A	Red
B	NULL
C	Blue
D	NULL

A LEFT OUTER join will display all the values from the left table, matching values from the right table, and inserting NULL values for non-matching values.

pentaho

Merge Join refresher..

Table a

ID	Value
1	A
2	B
3	C
4	D

Table b

ID	Value
1	Red
3	Blue
5	Yellow

a	b
A	Red
C	Blue
NULL	Yellow

A RIGHT OUTER join will display all the values from the right table, matching values from the left table, and inserting NULL values for non-matching values.

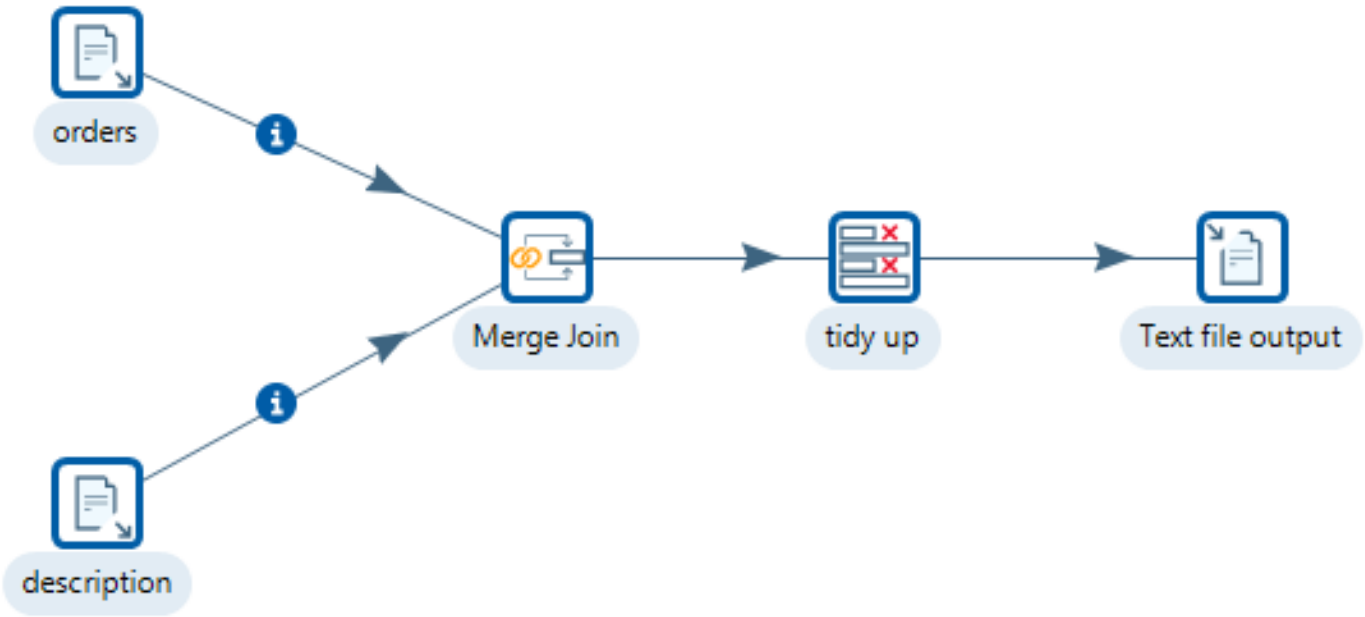
a	b
A	Red
B	NULL
C	Blue
D	NULL
NULL	Yellow

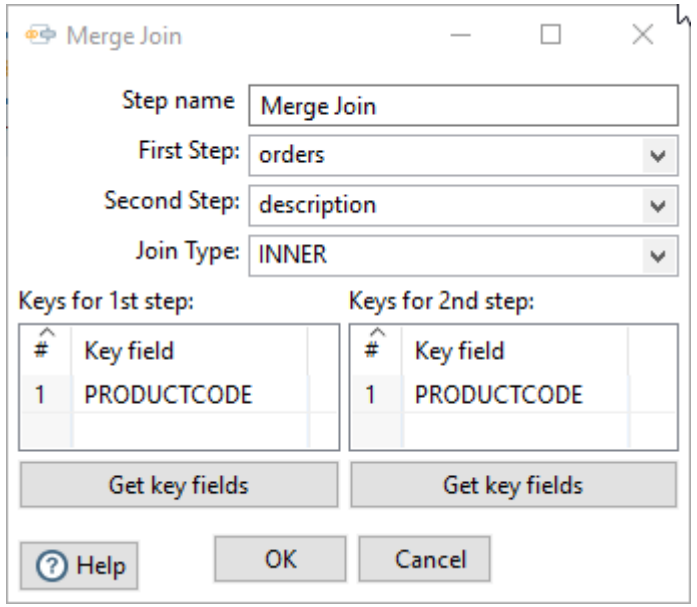
A full OUTER join is not available in MySQL. It takes the values from both tables, inserting NULL values for non-matching values.

pentaho

GD4-2-2: Merge Join - Orders

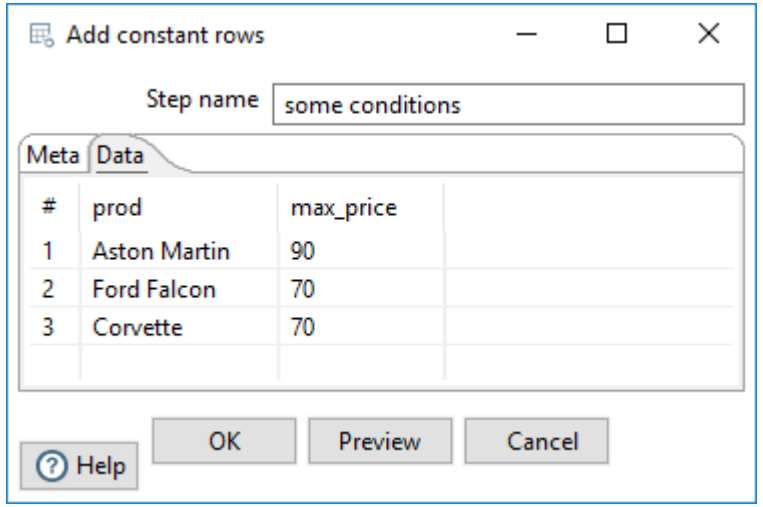
Solves the problem of merge rows.. GD4-1-1
With a 'join' you dont have to ensure that each data stream has the same structure / layout.

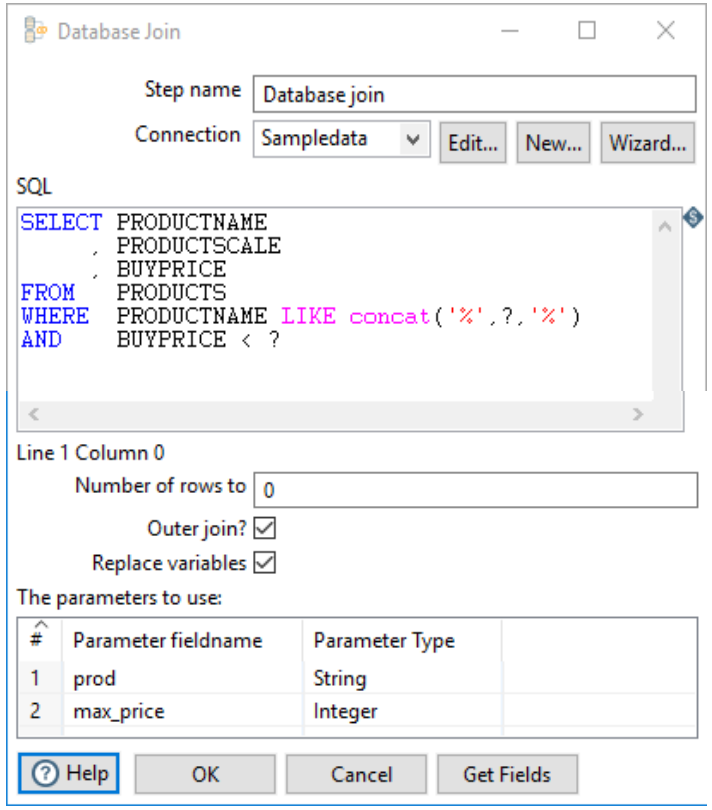




GD4-2-3: Database Join

- Searching for information in databases, text files, web services, and so on, is a very common task.
- The Database Join compares a dataset of records against the table using variables / parameters.





As its uses an Outer Join,
All the rows from the datasets are returned.

GD4-2-3: Database Join

- The first record

```
WHERE PRODUCTNAME LIKE concat ('%','Aston Martin','%') AND BUYPRICE < 90
```

The other database fields are returned as stream fields.

- The second record

```
WHERE PRODUCTNAME LIKE concat ('%',‘Ford Falcon','%') AND BUYPRICE < 70
```

No ‘Ford Falcon’ values were found with a BUYPRICE <70, so the fields returned NULL values.

Summary

- Stream Operations
  - Guided Demonstration 4-1-1: Merge rows
  - Guided Demonstration 4-1-2: Merge rows (diff)

- Joins
  - Guided Demonstration 4-2-1: Join rows
  - Guided Demonstration 4-2-2: Merge Join
  - Guided Demonstration 4-2-3: Database Join

- Lookups
  - Guided Demonstration 4-3-1: Database Lookups

pentaho

Lookups

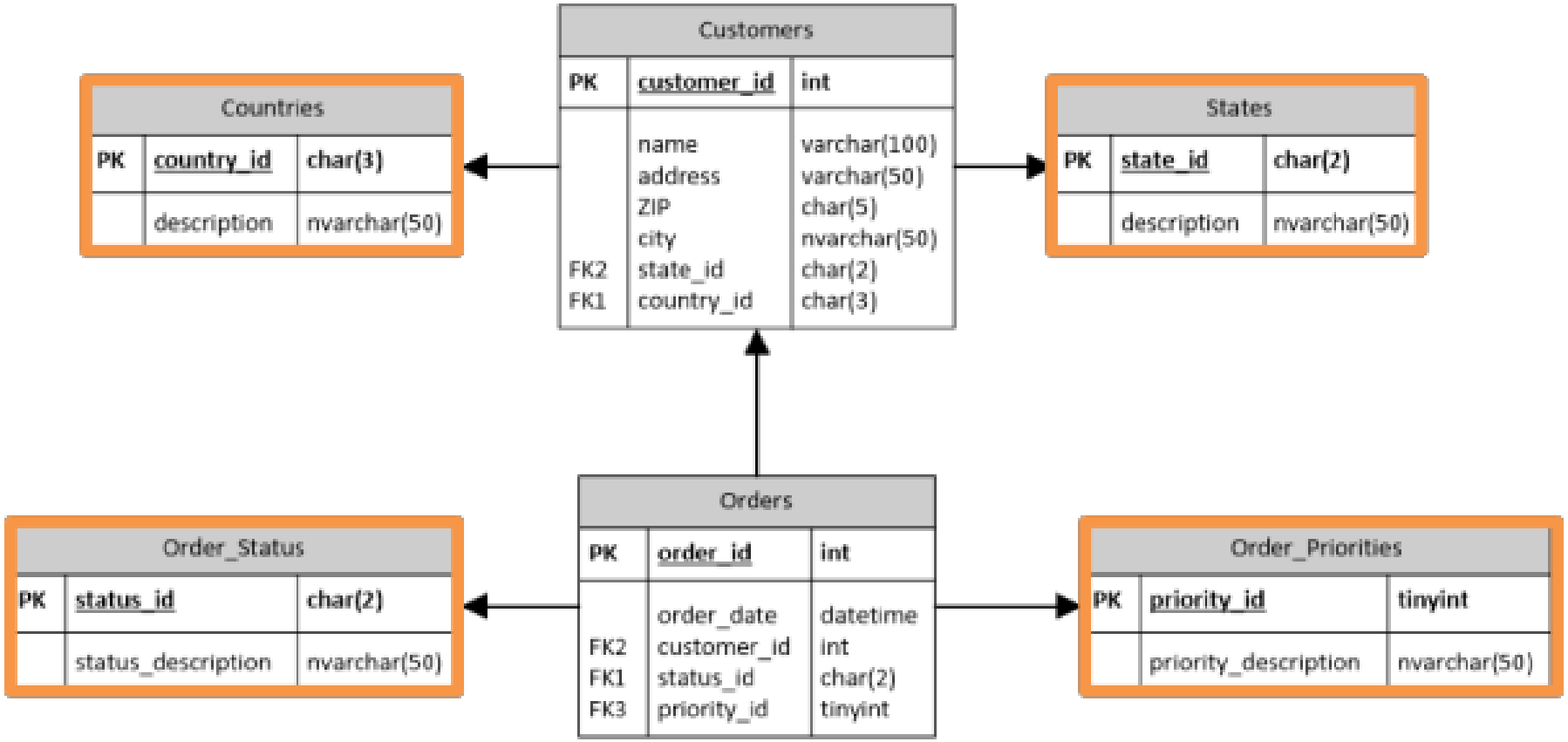
Pentaho Data Integration

Topics

- Lookups
  - Guided Demonstration 4-3-1: Lookup

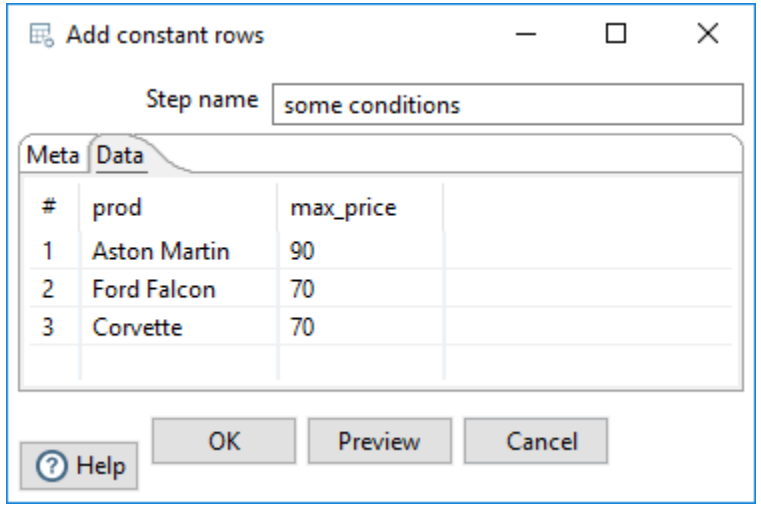
Database Lookup Tables

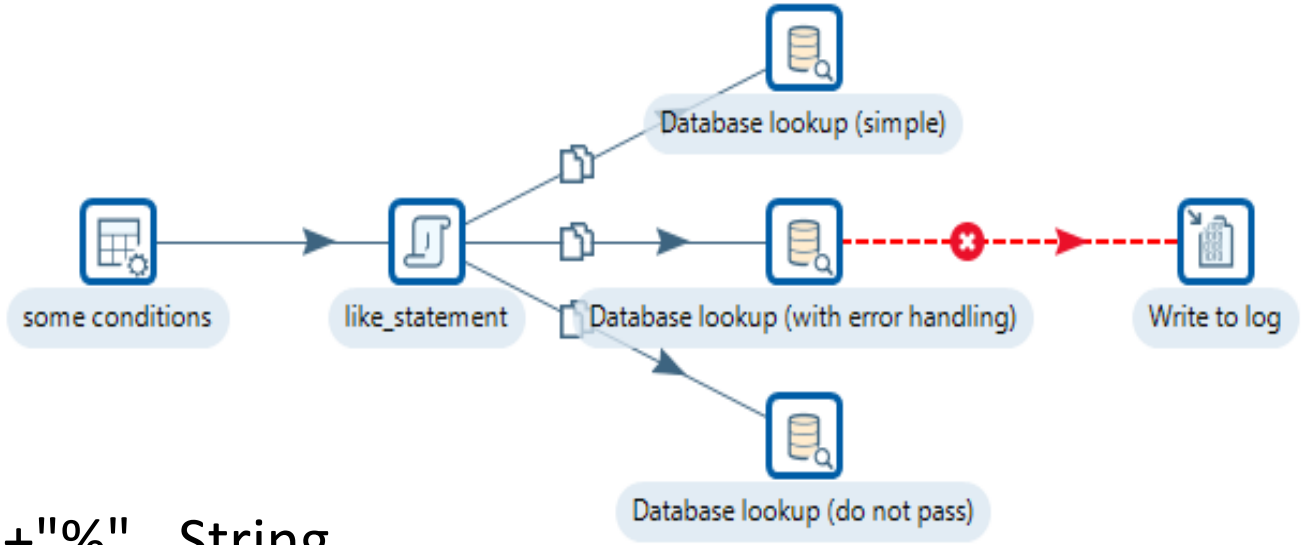
- Besides transforming the data, you may need to search and bring data from other sources.



GD4-2-1: Database Lookup

- Simple - straightforward lookup, returns rows inc. rows that don't compare
- Error - rows that don't compare are streamed to error step
- Do not pass – don't pass error

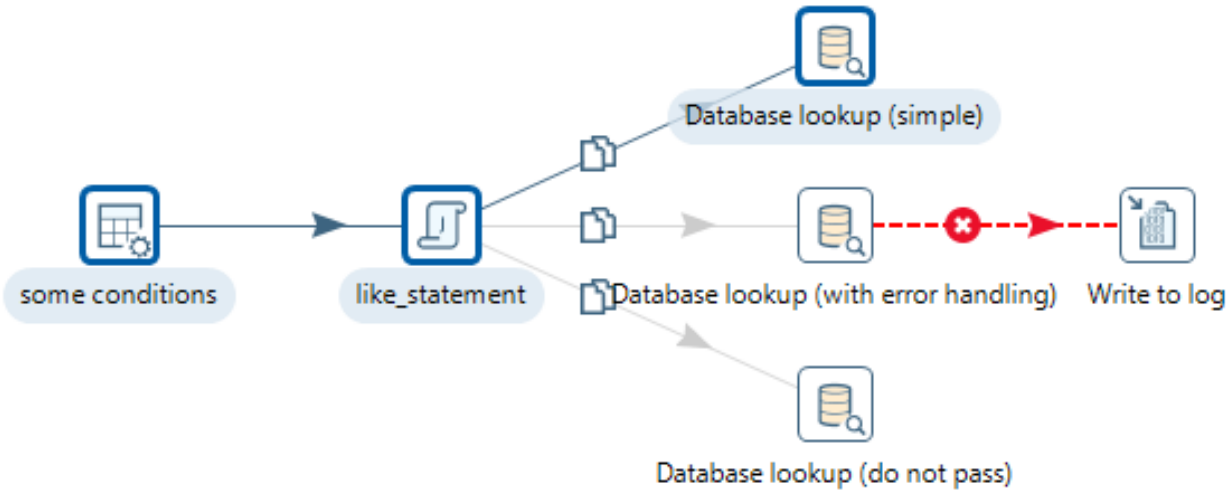




- like_statement "%"+prod+"%" String

GD4-3-1: Database Lookup

- Simple



Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

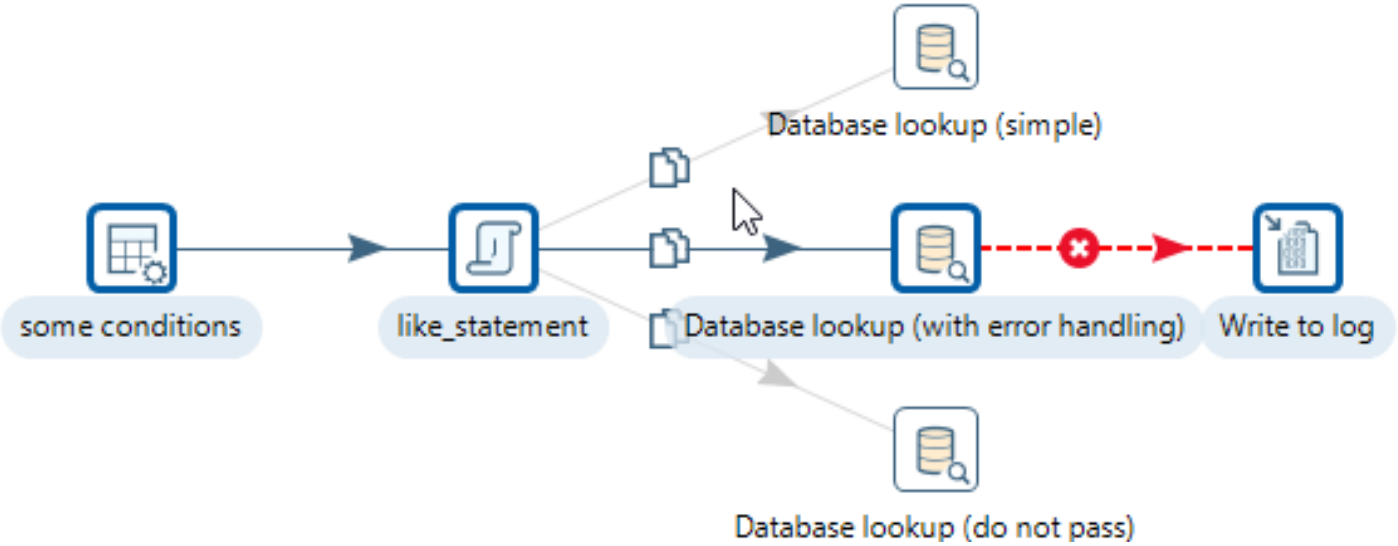
First rows Last rows Off Inspect Data

#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Ford Falcon	70	%Ford Falcon%	not available	<null>
3	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

pentaho

GD4-3-1: Database Lookup

- error




```
2016/12/10 18:26:48 - Spoon - Launching transformation [GD4-2-1_databse_lookup]...
2016/12/10 18:26:48 - Spoon - Started the transformation execution.
2016/12/10 18:26:48 - GD4-2-1_databse_lookup - Dispatching started for transformation [GD4-2-1_databse_lookup]
2016/12/10 18:26:48 - some conditions.0 - Finished processing (I=0, O=0, R=0, W=3, U=0, E=0)
2016/12/10 18:26:48 - like_statement.0 - Finished processing (I=0, O=0, R=3, W=3, U=0, E=0)
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - ------------> Linenr 1----------------------------
2016/12/10 18:26:48 - Write to log.0 - no productname match at that price
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - Ford Falcon
2016/12/10 18:26:48 - Write to log.0 - 70
2016/12/10 18:26:48 - Write to log.0 - %Ford Falcon%
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - ====================
2016/12/10 18:26:48 - Database lookup (with error handling).0 - Finished processing (I=2, O=0, R=3, W=2, U=0, E=1)
2016/12/10 18:26:48 - Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2016/12/10 18:26:48 - Spoon - The transformation has finished!!
```

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

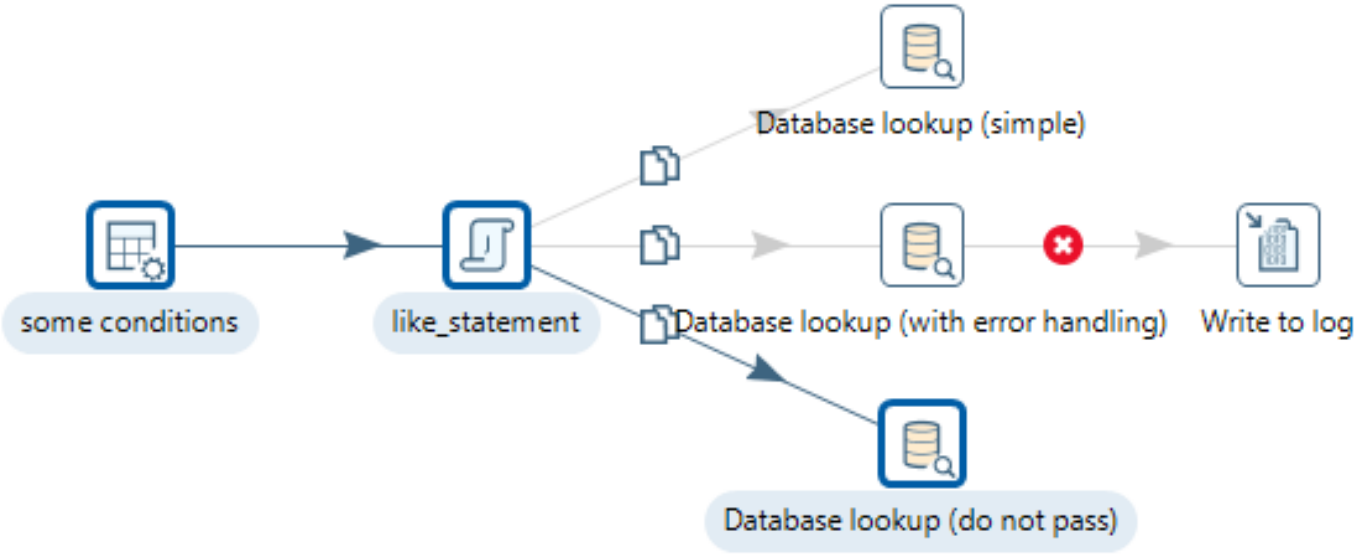
First rows | Last rows | Off | Inspect Data

#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

pentaho

GD4-3-1: Database Lookup

- Do not pass



Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off

Inspect Data

#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

pentaho

Formula

Modified JavaScript Value

Pentaho Data Integration

Topics

- Formula
  - Guided Demonstration 4-4-1: Booking Reservation
- Modified JavaScript Value
  - Guided Demonstration 4-4-2: Replace in String
- User Defined Java Class

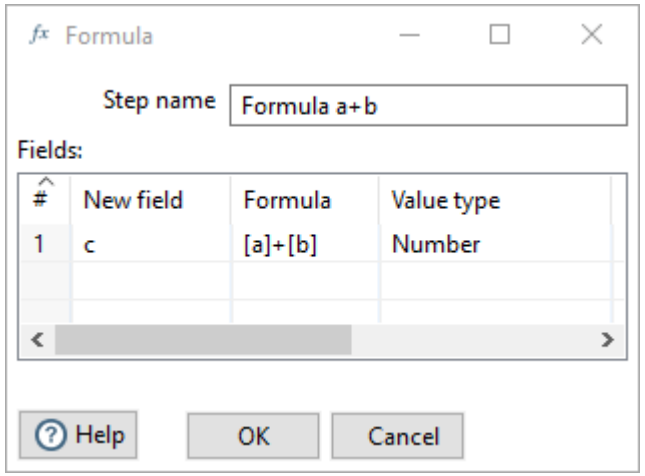
Scripting

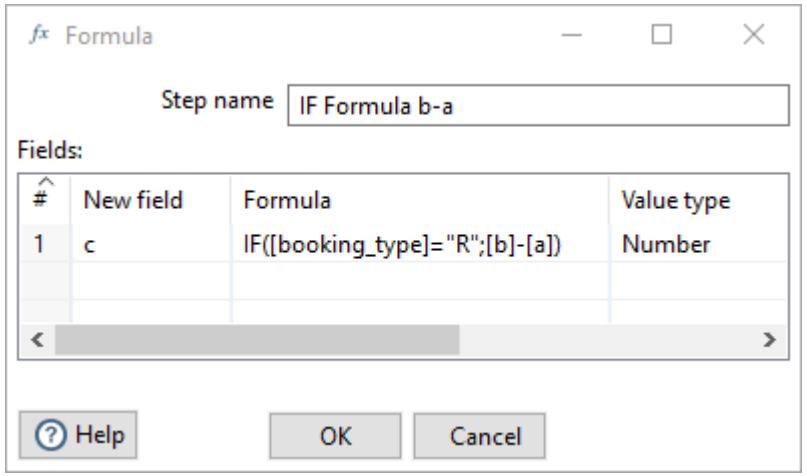
- Love-hate relationship: maintainability vs. power and flexibility
- Historically, Java Script step was PDI's "duct tape", taking care of complex transformation work
- Over the years more standard steps and job entries got introduced
- As a general rule of thumb, avoid using scripting altogether

pentaho

GD4-4-1: Formula Step

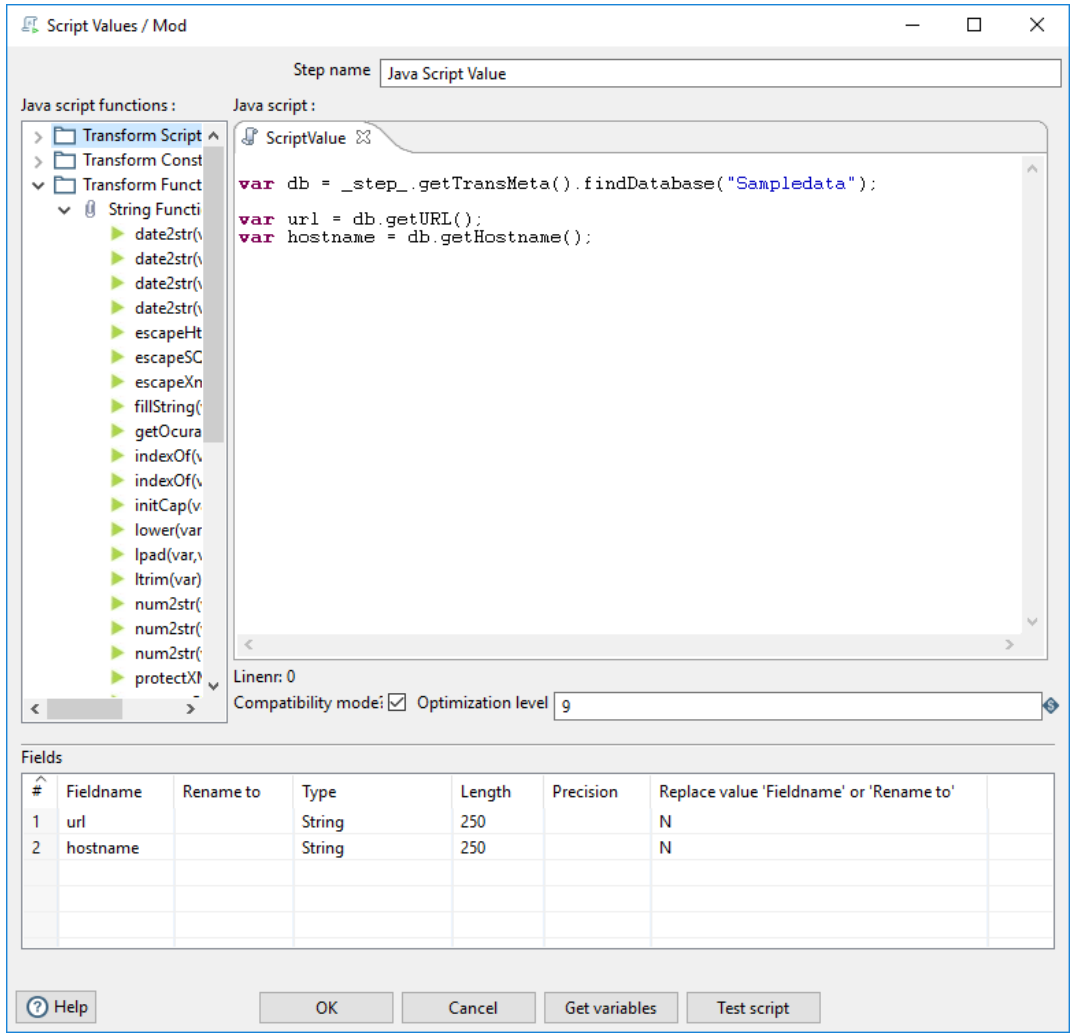
- Faster than JavaScript step
- Not a real scripting step
- Oasis OpenFormula syntax (also used in Calc, OpenOffice)
  http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part2.html
- Allows for more flexible formulas than the predefined ones in the Calculator step
- Conditional logic
- Note
  - No fields selector





Modified JavaScript Value

- Using JavaScript in a transformation.
- Lots of inbuilt functions..
- Look in the samples folder ..



Compatibility Mode

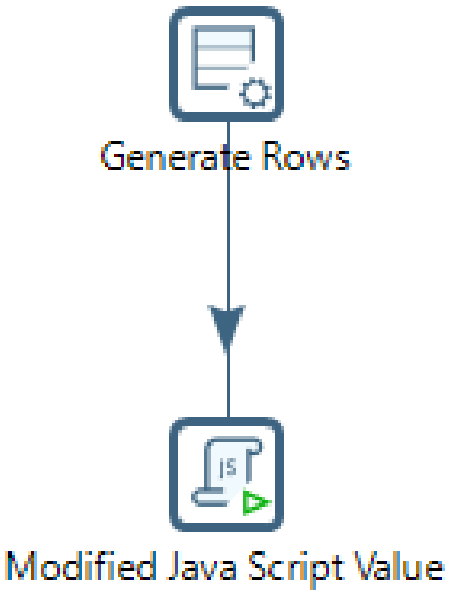
What does the compatibility switch do?

- There are two version of the JavaScript engine: the 2.5 version and the 3 version. If "compatibility mode" is checked (and by default it is), JavaScript works like it did in version 2.5. Obviously the new version should be used if possible so uncheck "compatibility mode" if you can.

- The big difference between the two versions is that in 2.5, value objects are directly modifiable and their type can be changed (a date variable can be converted into a string). This can cause errors. Because this is no longer possible in the 3.0 version, the JavaScript should also be faster.

pentaho

GD4-4-2: Modified JavaScript Value



Script Values / Mod

Step name: Modified Java Script Value

Java script functions:

ltrim(var)
num2str(
num2str(
num2str(
protectXl
removeC
replace(v
rpad(var,
rtrim(var
str2RegE
substr(va
substr(va
trim(var)

Java script:

ScriptValue

```
var result=field.replace("/", "").replace("\\", "").replace("-", "");
```

Position: 1, 69

Compatibility mode: ☐ Optimization level: 9

Fields

#	Fieldname	Rename to	Type	Length	Precision	Replace value 'Fieldname' or 'Rename to'
1	result		String			N

Help | OK | Cancel | Get variables | Test script

pentaho

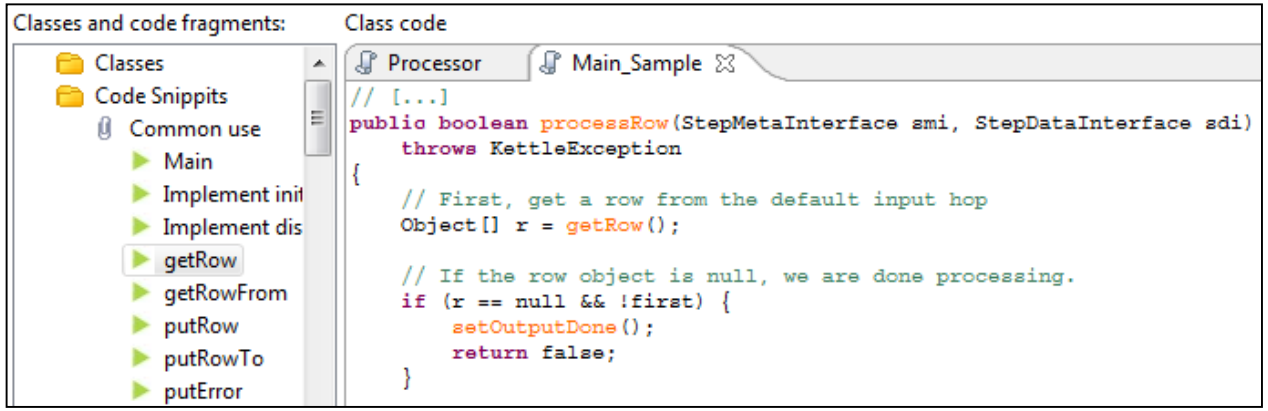
User Defined Java Class

- Instead of just a single expression, this step lets you define a complete class, which allows you to write a Kettle plugin as a step.

  http://rpbouman.blogspot.co.uk/2009/11/pentaho-data-integration-javascript.html

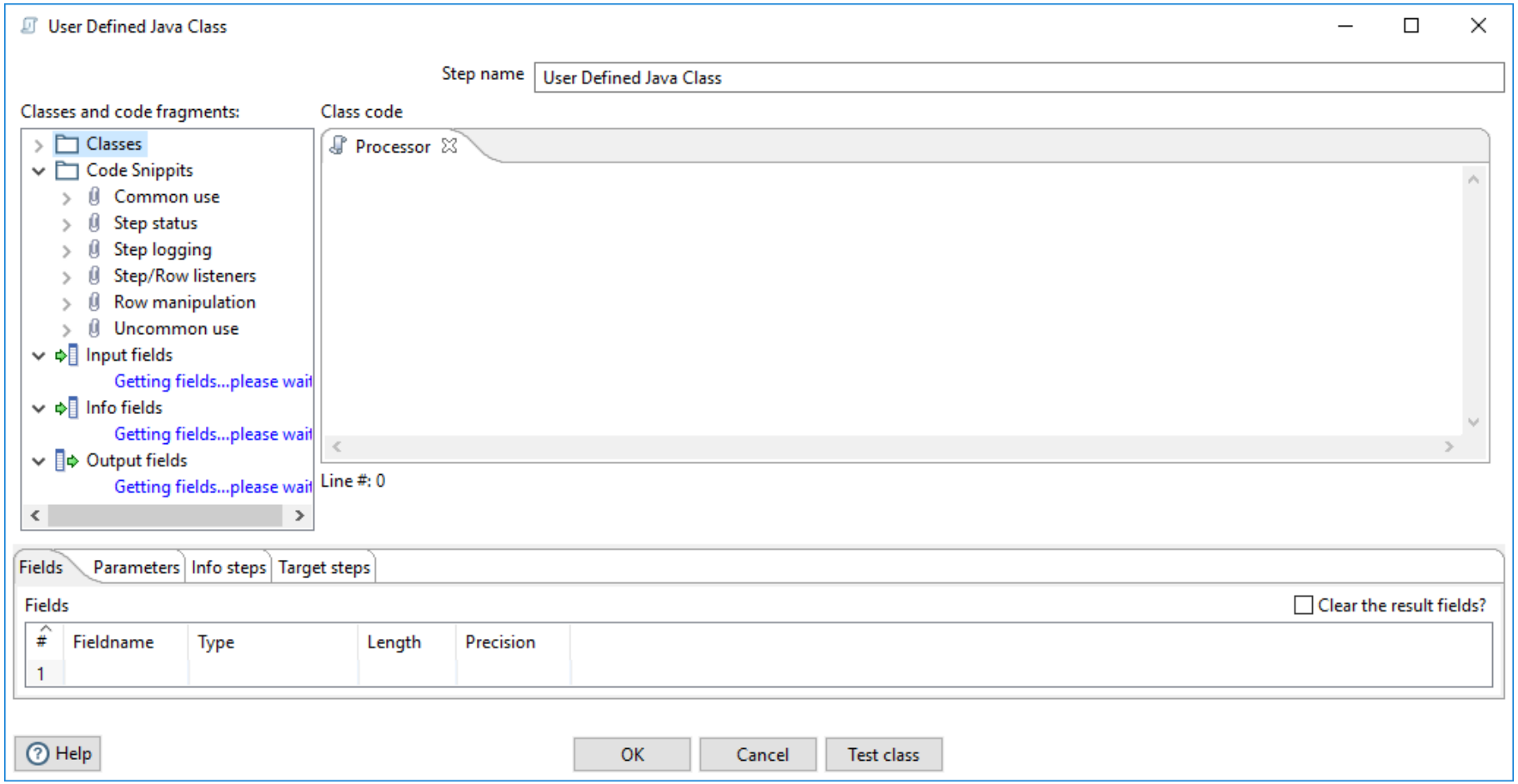
  http://wiki.pentaho.com/display/EAI/Writing+your+own+Pentaho+Data+Integration+Plug-In

- The benefit of User Defined Java Class is to simplify the deployment process.
- Code snippets provide samples:



User Defined Java Class

- Code snippet samples



Summary

- Formula
  - Guided Demonstration 4-4-1: Booking Reservation
- Modified JavaScript Value
  - Guided Demonstration 4-4-2: Replace in String
- User Defined Java Class

Pentaho Data Integration

6. Enterprise Solution

Topics

- Jobs
  - Guided Demo 6-1-1: Hello World
- Scalability
  - Guided Demo 6-2-1: Configuring a Slave Server
  - Guided Demo 6-2-2: Scheduling & Monitoring
- Logging
  - Guided Demo 6-3-1: Set up Logging

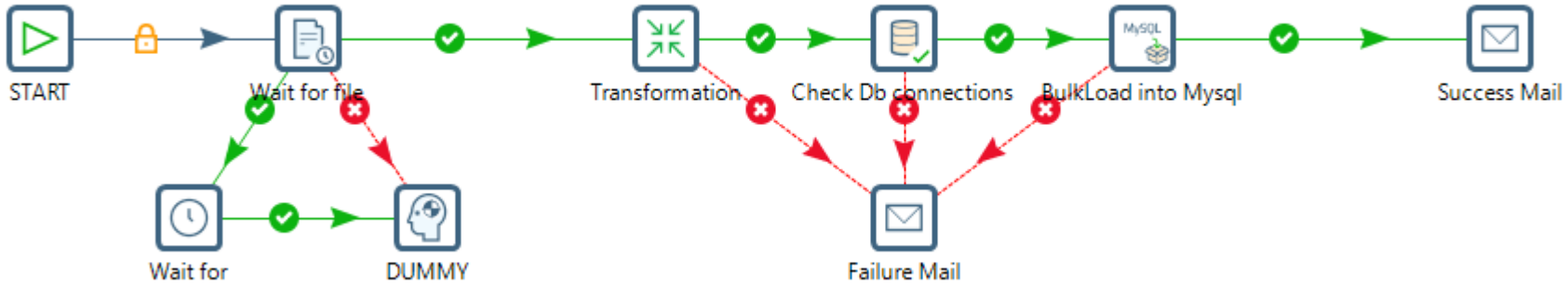
Jobs

Scalability

Logging

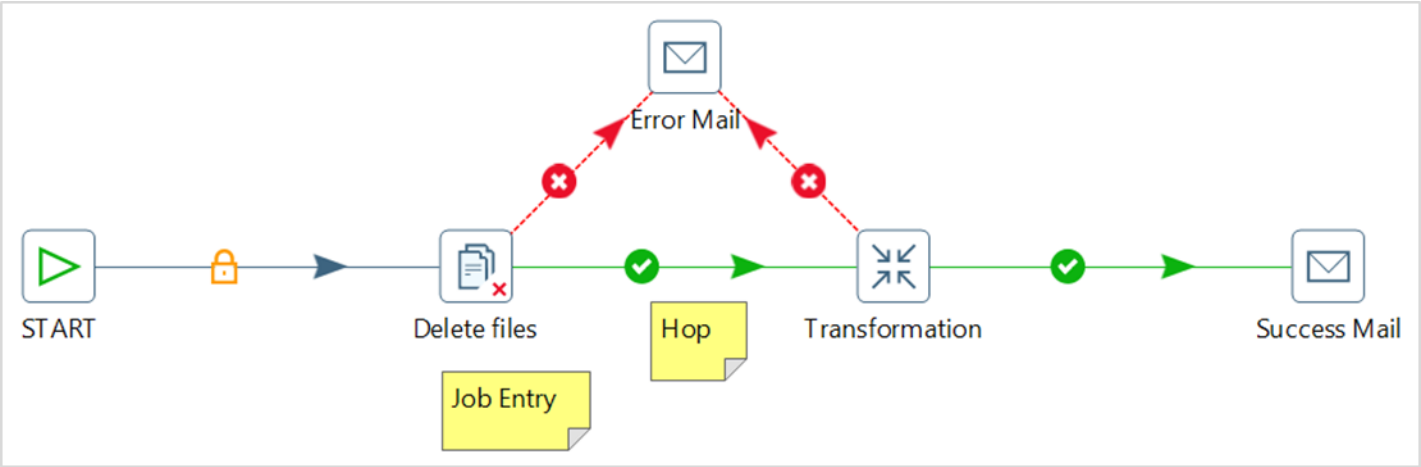
Pentaho Data Integration

Jobs Workflow



- Create a new job
- Adding and configuring job entries
- Job Properties
- Using file management job entries
- Implementing error handling in jobs
- Reviewing execution results
- Saving a job

Jobs



- Jobs are workflow-like models for coordinating resources, execution and dependencies of ETL activities
- Consist of job entries, hops and notes
- Aggregate individual transformations to a process and introduce order
- Perform all sorts of maintenance tasks
- Hops can be conditional

Start → Check → Watch → Execute → Notify → Finish

Job Orchestration

- In an IT environment, orchestration is the ongoing task of ensuring that technical processes run reliably and according to schedule while producing the correct outputs using acceptable amounts of resources

- Orchestration includes:
  - Ensuring the availability of resources
  - Executing a series of tasks in the right order
  - Detecting and recovering from errors
  - Logging the results
  - Notifying people and other applications

pentaho

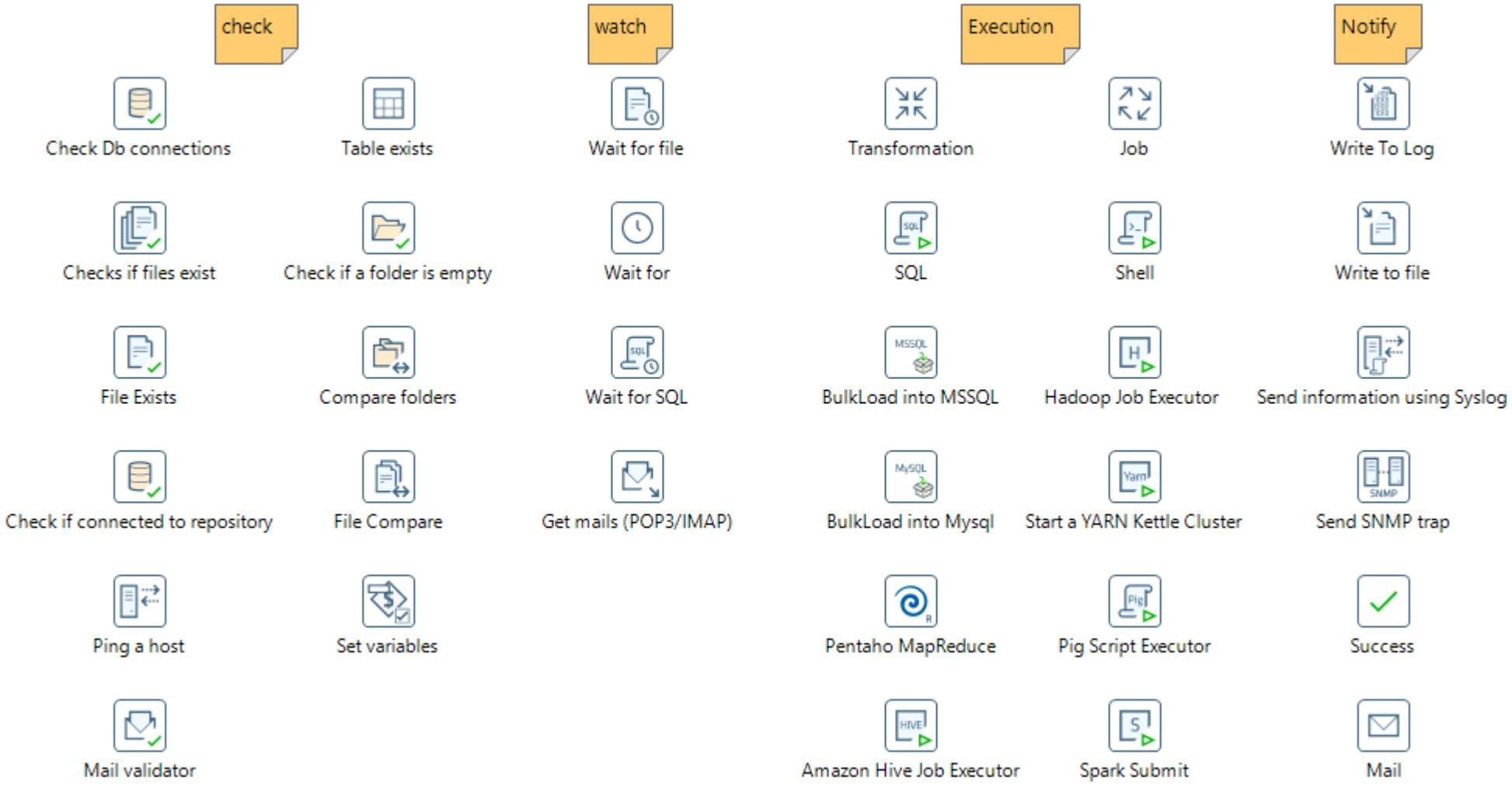
Job Management

- Always check the status of external systems
  - Host systems
  - Databases / tables
  - File systems / directories
  - FTP sites
  - Web services

- Execute tasks with retry logic
- Notify others when a problem has occurred
- Store variables in a central location and use them consistently

pentaho

Job Entry Toolkit



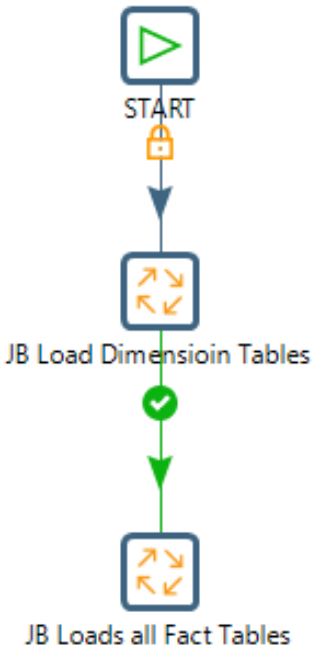
Job Entry

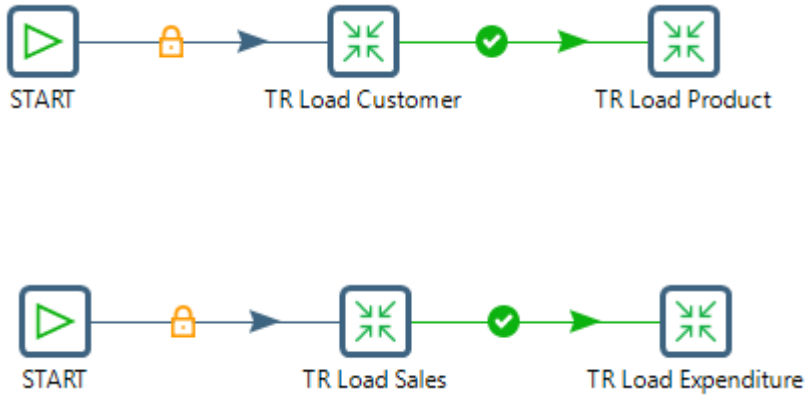
- A job (Job / Transformation) entry is the primary building block of a job.
  - Execute jobs / transformations, retrieve files, generate email, and so on
- Each job can only have one job entry.

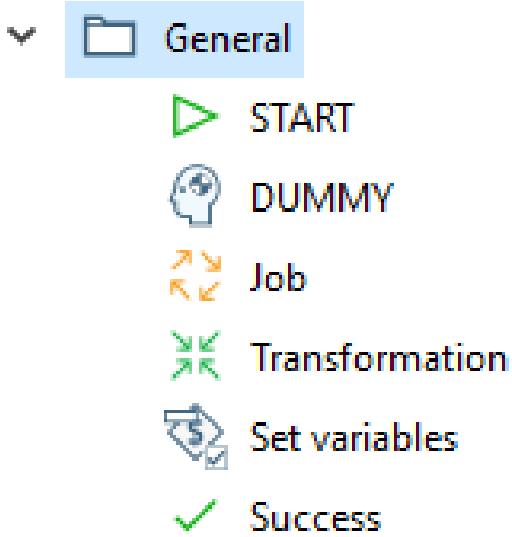
Job entries can pass a result object containing rows

Rows can be exchanged in memory – result rows

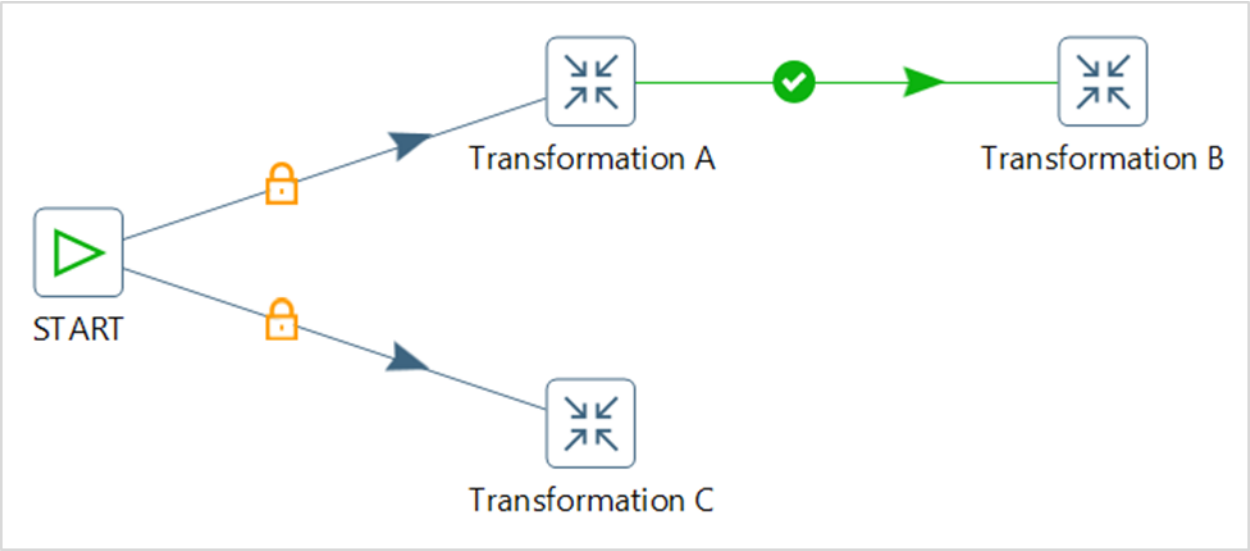
Rows are passed in batch mode, not in a streaming fashion (transformations)







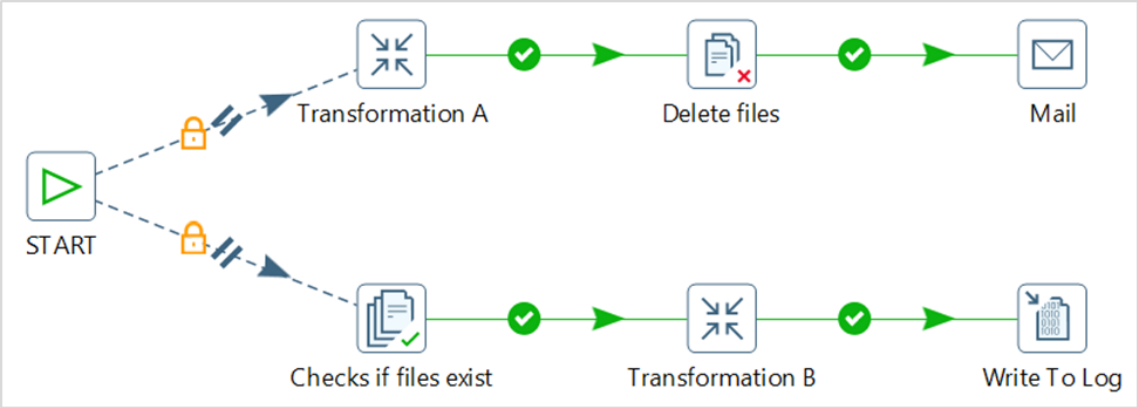
Multiple Paths & Backtracking



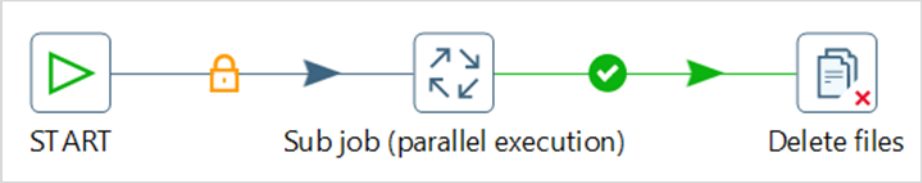
- Jobs are executed using a so called backtracking algorithm
- The path itself is defined by the actual outcome (success or failure) of the job entries
- However, a backtracking algorithm means that a path is always followed until the very end before a next possibility is considered
- Order depends on creation of job entries
- The result of the job (success or failure) depends on the last job entry

Parallel Execution

- A job entry can be told to execute the next job entries in parallel
  - Running in separate threads
- When you have a number of sequential job entries, these are executed in parallel as well

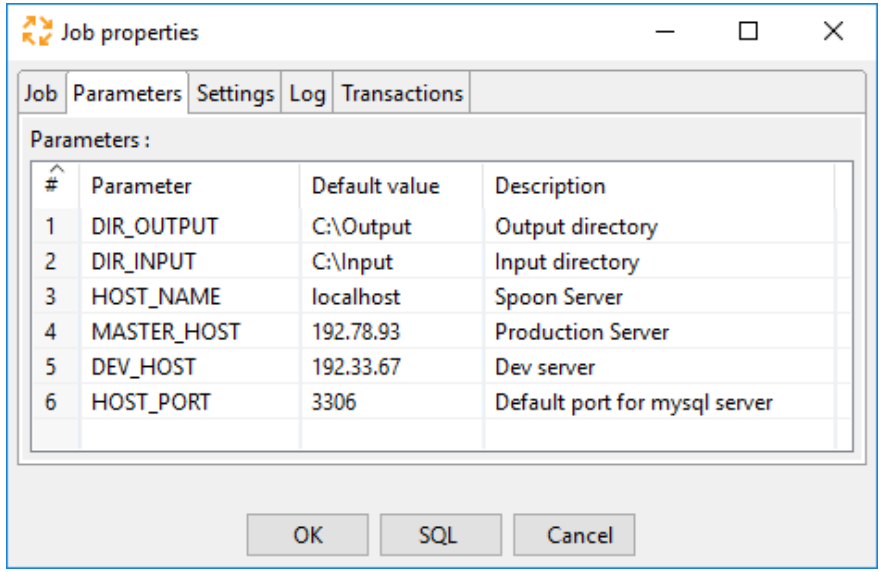


- If you want to continue in sequence after this parallel execution you need to put the parallel part of the job in a sub-job



Job Properties

- One of the handy features available to Job properties are, Named Parameters.
- If there are variables that you consider to be parameters for your transformation or job, you can declare them as a 'Named Parameter'.
- Advantage of using Named Parameters is that they are explicitly listed, documented with a description, and carry a default value.
- Once defined you can declare in all possible locations.. ${named_parameter}



#	Parameter	Default value	Description
1	DIR_OUTPUT	C:\Output	Output directory
2	DIR_INPUT	C:\Input	Input directory
3	HOST_NAME	localhost	Spoon Server
4	MASTER_HOST	192.78.93	Production Server
5	DEV_HOST	192.33.67	Dev server
6	HOST_PORT	3306	Default port for mysql server

Job File Management

- File management
  - HTTP
  - Unzip file
  - Write to file
  - Copy Files
  - Delete file
  - Move Files
  - Delete filenames from result
  - Delete folders
  - File Compare
  - Wait for file
  - Zip file
  - Process result filenames
  - Create a folder
  - Delete files
  - Add filenames to result
  - Convert file between Windows and Unix
  - Create file
  - Compare folders

GD5-1-1: Hello World - Job

Create a Job for your Project

Scalability

Pentaho Data Integration

Topics

- Pentaho Carte Cluster
  - Master / Slave Nodes
  - Guided Demo 5-2-1: Configuration of Master Node

Carte Configuration

Master DI server

- Default configuration for low volume and process mix

Master DI server with Independent number Carte servers

- Used for large amount of transformations of lesser volume
- Master acts as a load balancer for transformations

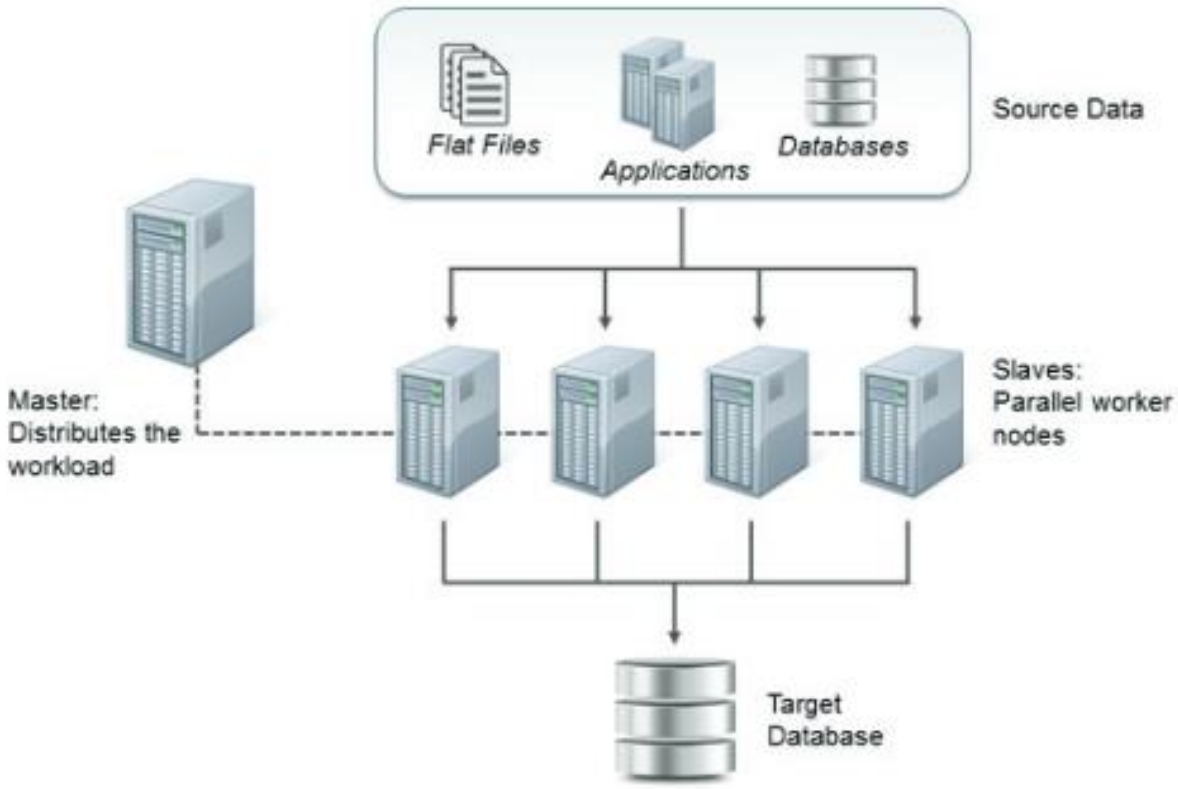
Master DI server with slave servers (Clustering)

- Used for large volumes with fewer transformations
- Distributes set of rows across slave servers
- Can be fixed cluster size or dynamic

pentaho

Clustering Carte Slave servers

- Clustering is a technique that can be used to scale out transformations to make them run on multiple servers, in parallel.
- The Cluster is defined with a Schema, with one master server acting as the controller for the cluster.
- The Cluster schema also contains metadata on how master and slaves pass data back and forth.



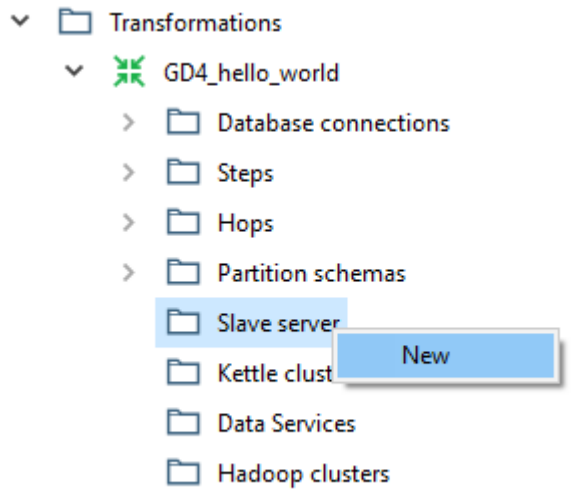
Carte as a Slave Server

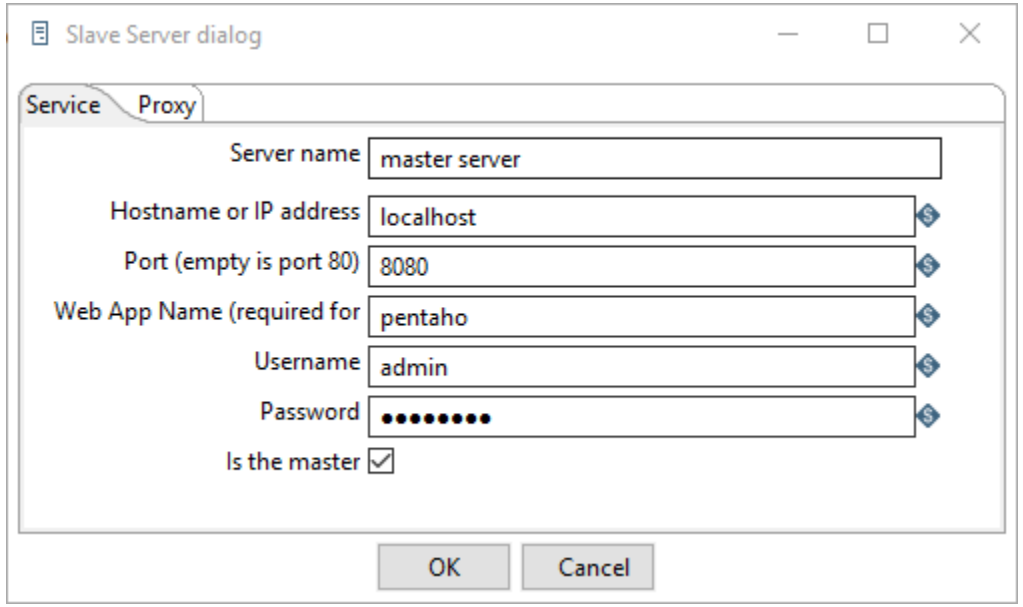
- Carte is a lightweight HTTP server that accepts commands from remote clients.
- These commands control the deployment, management, and monitoring of Jobs and Transformations.
- You can configure the carte server with an XML file (examples found in \pwd directory)

```
<slave_config>
<!-- on a master server, the slaveserver node contains information about this Carte instance -->
  <slaveserver>
    <name>master</name>
    <hostname>yourhostname</hostname>
    <port>8080</port>
    <username>cluster</username>
    <password>cluster</password>
    <master>Y</master>
  </slaveserver>
</slave_config>
```

GD5-2-1: Master DI Carte Server

- Open GD2-1-1_hello_world transformation in the Repository.
- On the View tab, select: Slave > New





- Right mouse click and select: Monitor
- Ensure that you have connected to the server.
- Access the server: http://localhost:8080/pentaho/status

Username: admin
Password: password

Configure a Slave Node

- Then start a Carte ‘listener’ on our server: port 8000

  ./carte.sh [xml config file]

  ..\carte.bat [xml config file]

  Or

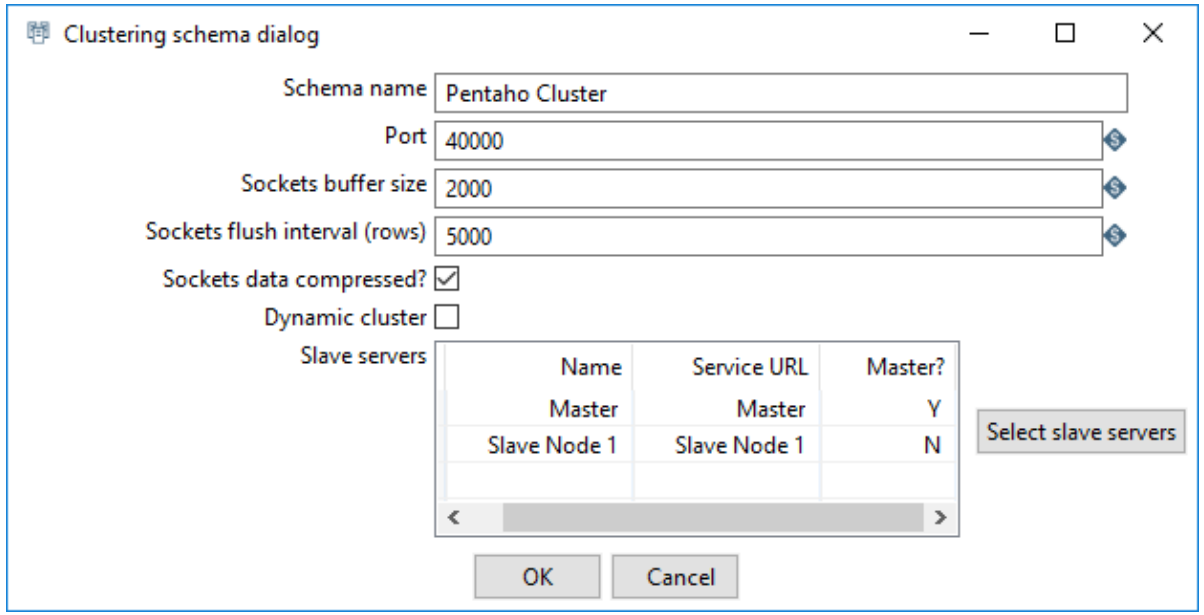
  [pentaho]\design-tools\data-integration\carte.bat localhost 8000

C:\WINDOWS\system32\cmd.exe - carte.bat localhost 8000

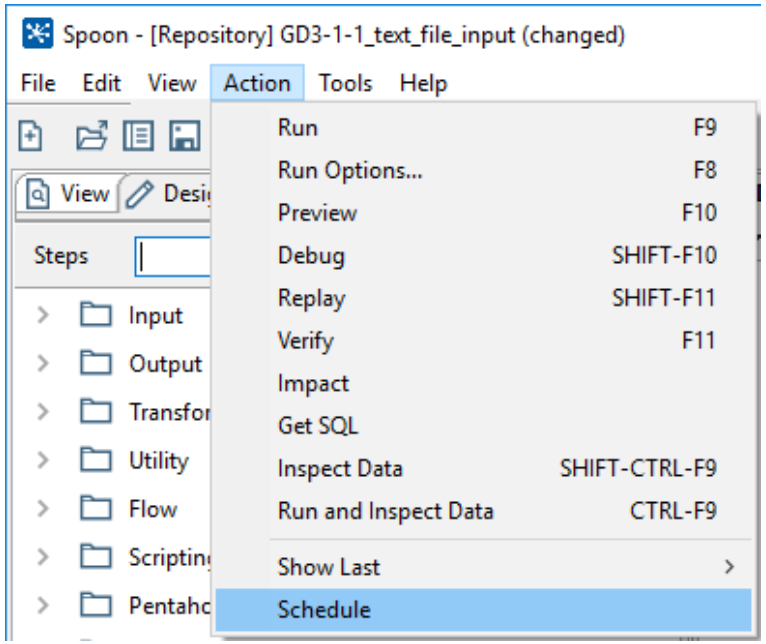
```
Not found in 'user.home' (C:\Users\joreilly) directory: C:\Users\joreilly\esapi\validation.properties
Loading validation.properties via file I/O failed.
Attempting to load validation.properties via the classpath.
validation.properties could not be loaded by any means. fail. Exception was: java.lang.IllegalArgumentException: Failed
to load ESAPI.properties as a classloader resource.
SecurityConfiguration for Logger.LogServerIP not either "true" or "false" in ESAPI.properties. Using default: true
2016/11/29 12:01:39 - Carte - Created listener for webserver @ address : localhost:8000
2016/11/29 12:12:40 - RepositoriesMeta - Reading repositories XML file: C:\Users\joreilly\.kettle\repositories.xml
2016/11/29 12:12:40 - PurRepositoryConnector - Creating security provider
2016/11/29 12:12:40 - PurRepositoryConnector - Creating repository web service
2016/11/29 12:12:40 - PurRepositoryConnector - Creating repository sync web service
2016/11/29 12:12:41 - PurRepositoryConnector - Repository web service created
2016/11/29 12:12:41 - PurRepositoryConnector - Creating unified repository to web service adapter
2016/11/29 12:12:41 - PurRepositoryConnector - Synchronizing repository web service
2016/11/29 12:12:41 - PurRepositoryConnector - Security provider created
2016/11/29 12:12:42 - PurRepositoryConnector - Registering security provider
2016/11/29 12:12:42 - PurRepositoryConnector - Repository services registered
2016/11/29 12:12:42 - PurRepository - Creating repository meta store interface
2016/11/29 12:12:42 - PurRepository - Connected to the enterprise repository
2016/11/29 12:12:43 - GD2-1-1_hello_world - Dispatching started for transformation [GD2-1-1_hello_world]
2016/11/29 12:12:43 - Hello World.0 - Finished processing (I=0, O=0, R=0, W=10, U=0, E=0)
```

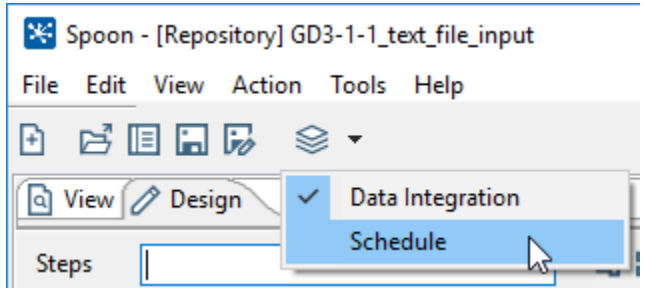
Cluster Schema

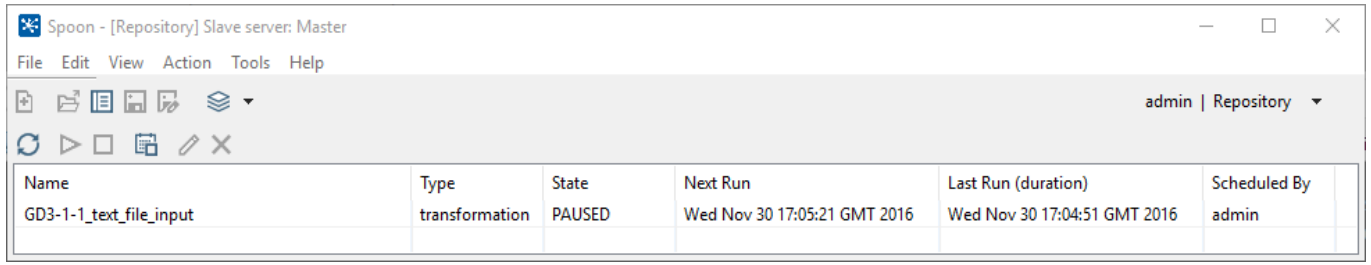
- Add the Slave Node
- Define the Cluster Schema:

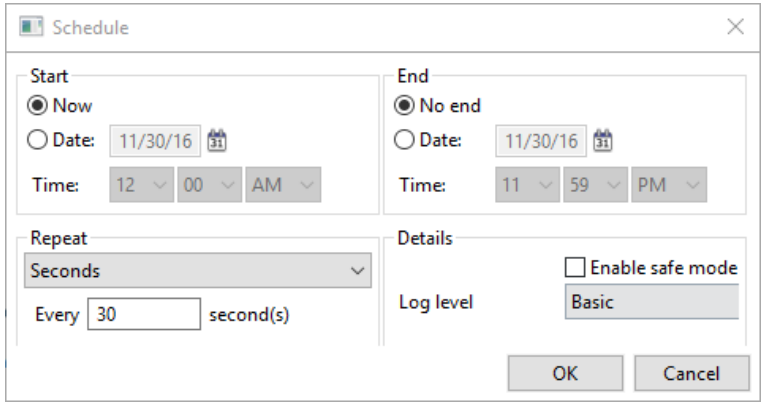


GD5-2-2: Monitor & Schedule

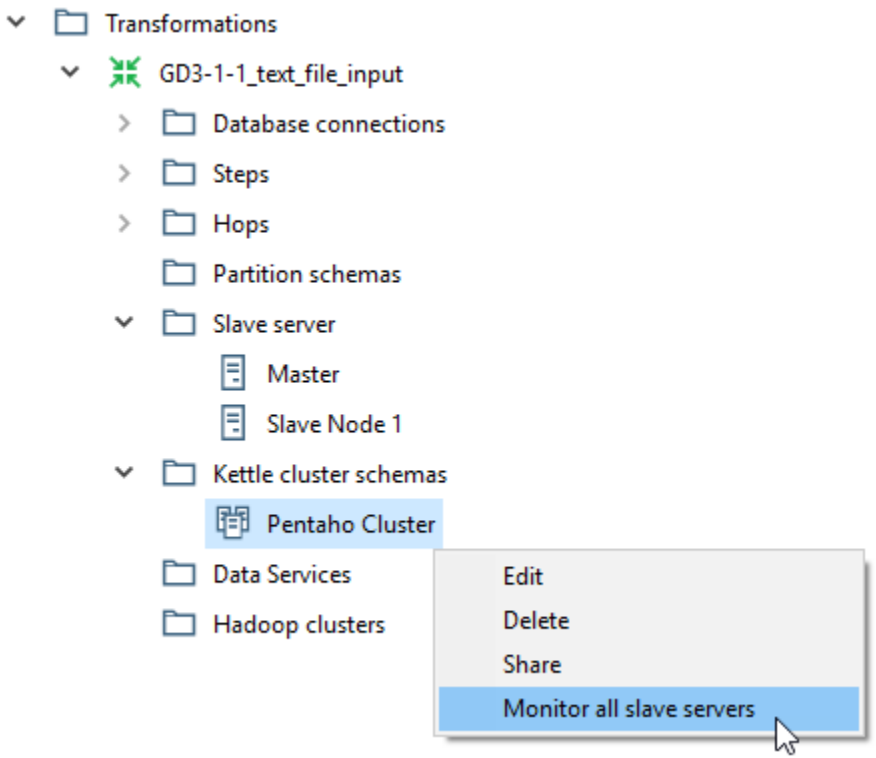








GD5-2-2: Monitor & Schedule



Master Node:

http://localhost:8080/pentaho/kettle/status

- Username: admin
- Password: password

Logging

Pentaho Data Integration

Topics

- Logging
- Log Entries
- Logging Architecture
- Database Logging
  - Guided Demo 6-4-1: Logging

Logging

- PDI features a logging framework that is used to provide feedback during transformation and job runs
- Logging is useful for monitoring progress during execution, but is also useful for debugging purposes
- Two types of logging can be used

  Log entries
  - Traditional logging
  - File-based approach
  - Verbose

  Database logging
  - Summarized results
  - RDBMS-based approach
  - Concise and structured
- Both types of logging can be used at the same time

Logging

- Most PDI components output logging information in the form of lines of text
- For example, when a step finishes, a line is generated to indicate this event

  2015/10/29 15:52:00 - Step name.0 -

  Finished processing (I=0, O=0, R=0, W=25, U=0, E=0)

- You can recognize three main parts in the log lines
  - The date and time
  - The name of step followed by a period and the step copy number
  - The logging text
- When you execute a transformation or a job, you can choose the logging level
  - Log level determines log verbosity and information logged
- Use the Write to log step/job entry to writing custom messages to the logfile

pentaho

Logging Levels

Logging levels are additive

- All the entries from previous level + selected level

Levels

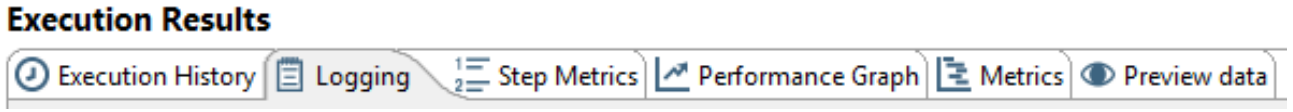
- Nothing – no log lines at all
- Error – shows error message if there is an error
- Minimal – informs you only at job or transformation level
- Basic (default) – basic summary information on individual steps
- Detailed – provides additional step information
  - Database steps provide info on database connection and executed SQL statements
- Debug – logs nearly all info, useful for debugging
- Row level – prints all available logging info
  - Prints row values as they pass through steps
  - eg. Finding the row that causes a transformation to fail

pentaho

Log File Locations

Local execution via Spoon

- Logging tab in the execution results pane



- Spoon_xxx.log in the temp files folder
  `\Users\Username\AppData\Local\temp`

Execution via Kitchen & Pan

- Logfile parameter defines location
- Note: Kitchen will only log information pertaining to the root job

Execution via the DI Server & Carte

- Configure logging via the logging settings tab of the job entry

Using the logging database

- Searching for problems in log files is not always convenient
- Database logging provides the option to write structured logging information to database tables for reporting and monitoring
- Logging tables dictate logging level (tables for Transformations)
  - Transformation: L_TRANS
  - Step: L_STEP
  - Metrics: L_METRICS
  - Performance: L_PERF
  - Logging channels: L_CHAN
- Configured via logging tab of the transformation settings
- Several database logging settings can be defined globally by system variables (kettle.properties)
  - DB, schema & table

Logging variables in kettle.properties

- The list of variables and descriptions can be viewed using the menu options – **Edit > Edit the Kettle.properties file**:

#	Variable name	Value	Description
1	KETTLE_CHANNEL_LOG_DB		The log channel log database default for all transformations and jobs
2	KETTLE_CHANNEL_LOG_SCHEMA		The log channel log schema default for all transformations and jobs
3	KETTLE_CHANNEL_LOG_TABLE		The log channel log table default for all transformations and jobs
4	KETTLE_EMPTY_STRING_DIFFERS_FROM_NULL	N	NULL vs Empty String. If this setting is set to Y, an empty string and null are different. Otherwise they are not.
5	KETTLE_JOBENTRY_LOG_DB		The job entry log database default for all jobs
6	KETTLE_JOBENTRY_LOG_SCHEMA		The job entry log schema default for all jobs
7	KETTLE_JOBENTRY_LOG_TABLE		The job entry log table default for all jobs
8	KETTLE_JOB_LOG_DB		The job log database connection default for all jobs
9	KETTLE_JOB_LOG_SCHEMA		The job logging schema default for all jobs
10	KETTLE_JOB_LOG_TABLE		The job logging table default for all jobs
11	KETTLE_LOG_SIZE_LIMIT	0	The log size limit for all transformations and jobs that don't have the "log size limit" property set in their respec...
12	KETTLE_MAX_LOG_SIZE_IN_LINES	0	The maximum number of log lines that are kept internally by Kettle. Set to 0 to keep all rows (default)
13	KETTLE_MAX_LOG_TIMEOUT_IN_MINUTES	0	The maximum age (in minutes) of a log line while being kept internally by Kettle. Set to 0 to keep all rows indef...
14	KETTLE_PLUGIN_CLASSES		A comma delimited list of classes to scan for plugin annotations
15	KETTLE_SHARED_OBJECTS		The location of the shared object file (xml) for transformations and jobs
16	KETTLE_STEP_LOG_DB		The steps log database default for all transformations
17	KETTLE_STEP_LOG_SCHEMA		The steps log schema default for all transformations
18	KETTLE_STEP_LOG_TABLE		The steps log table default for all transformations
19	KETTLE_STEP_PERFORMANCE_SNAPSHOT_LIMIT	0	The maximum number of step performance snapshots to keep in memory. Set to 0 to keep all snapshots indefi...
20	KETTLE_TRANS_LOG_DB		The transformation log database connection default for all transformations.
21	KETTLE_TRANS_LOG_SCHEMA		The transformation logging schema default for all transformations
22	KETTLE_TRANS_LOG_TABLE		The transformation logging table default for all transformations
23	KETTLE_TRANS_PERFORMANCE_LOG_DB		The transformation performance log schema default for all transformations
24	KETTLE_TRANS_PERFORMANCE_LOG_SCHEMA		The transformation performance log database connection default for all transformations
25	KETTLE_TRANS_PERFORMANCE_LOG_TABLE		The transformation performance log table default for all transformations

GD5-3-1: Set up logging to a database

- Setting up transformation database logging
  - Transformation, Step and Metrics tables
  - Performance and Logging modifies Metrics table
- Review the configuration
  - Logging tables
  - Fields to log
    - ID_BATCH: unique id per execution
    - Step name for LINES_xxx (transformation logging)
    - Date columns
  - Enable step performance monitoring checkbox (monitoring tab)
- Review logging tables
- Review the execution history tab
- Review the performance graph tab (execution results)

pentaho

GD5-3-1: Set up logging to a database

- Create a connection / database: L_ETL
- Create 5 tables – click on SQL button
  - Transformation: L_TRANS
  - Step: L_STEP
  - Metrics: L_METRICS
  - Performance: L_PERF
  - Logging channels: L_CHAN

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

Transformation log table | Step log table | Step performance log table | Logging channel log table | Metrics log table

#	Batch ID	Status	Read	Written	Updated	Input	Output	Rejected	Errors	Date range start	Date range end	Log date	Dependency date	Start time
1	11	end	0	0	0	0	0	0	0	2016/04/28 12:30:44	2016/04/28 12:33:11	2016/04/28 12:33:11	2016/04/28 12:33:11	2016/04/28 12:33:11
2	10	end	0	0	0	0	0	0	0	2016/04/28 12:16:26	2016/04/28 12:30:44	2016/04/28 12:30:45	2016/04/28 12:30:44	2016/04/28 12:30:44
3	9	end	0	0	0	0	0	0	0	2016/04/28 11:53:25	2016/04/28 12:16:26	2016/04/28 12:16:26	2016/04/28 12:16:26	2016/04/28 12:16:26
4	8	end	0	0	0	0	0	0	0	2016/04/28 11:38:27	2016/04/28 11:53:25	2016/04/28 11:53:25	2016/04/28 11:53:25	2016/04/28 11:53:25
5	7	end	0	0	0	0	0	0	0	2016/04/28 11:30:06	2016/04/28 11:38:27	2016/04/28 11:38:27	2016/04/28 11:38:27	2016/04/28 11:38:27

GD5-3-1: Set up logging to a database

- Use the connection / database in L_ETL
- Create 4 tables – click on SQL button
  - Job: L_JOB
  - Job Entry: L_JOB_ENT
  - Logging Channel: L_LOG_CHAN
  - Checkpoints: L_CHECK

Execution Results

History | Logging | Job metrics | Metrics

Job log table | Job entry log table | Logging channel log table | Checkpoints log table

#	Batch ID	Status	Read	Written	Updated	Input	Output	Rejected	Errors	Date range start	Date range end	Log date	Dependency date	Start time
1	2	end	0	0	0	0	0	0	0	2016/04/28 12:30:44	2016/04/28 12:33:11	2016/04/28 12:33:11	2016/04/28 12:33:11	2016/04/28 12:33:11
2	1	end	0	0	0	0	0	0	0	1899/12/31 23:00:00	2016/04/28 12:30:44	2016/04/28 12:30:45	2016/04/28 12:30:44	2016/04/28 12:30:44
3	0	end	0	0	0	0	0	0	1	1899/12/31 23:00:00	2016/04/28 12:29:31	2016/04/28 12:29:31	2016/04/28 12:29:31	2016/04/28 12:29:31

pentaho

Course Review

- Module 1 - Introduction
  - Architecture & Components
  - Navigating Spoon
- Module 2 - Concepts & Terminology
  - Transformations
  - Jobs
- Module 3 - Datasources
  - Files
  - Databases
- Module 4 - Data Enrichment
  - Merge streams
  - Joins

pentaho

Course Review

- Module 5 – Enterprise Solution
  - Jobs
  - Scalability – Clusters
  - Monitoring & Scheduling
  - Logging