

EXPLORAÇÃO DE DADOS / DATA MINING

(Part I) Laboratory Assignment 03: Brief introduction on using Pentaho Data Integration/Kettle

This document presents useful tips for implementing laboratory assignment 3.

Setup

- The accounts created in Microsoft SQL Server require changing the password on first use. This prevents the access to the database using third-party applications. So, the first login must be performed using an application such as Microsoft SQL Server Management Studio. If you do not have any Microsoft SQL Server Client application, you can ask a colleague to perform the first login and change the password or send an email to jose.moreira@ua.pt.
- To connect from PDI to SQL Server you also need a JDBC driver. Go to <https://www.microsoft.com/en-us/download/details.aspx?id=11774> and download the archive *sqljdbc_6.0.8112.200_enu.tar.gz*. Unzip archive and copy *sqljdbc_6.0\enu\jre8\sqljdbc42.jar* into *data-integration/lib* in the Pentaho base directory. For more information, please refer to <http://bigdatafan.blogspot.com/2016/05/connect-pentaho-pdi-spoon-to-sql-server.html>. You may also need to copy *jtds-1.3.1.jar* (<https://sourceforge.net/projects/jtds/files/jtds/1.3.1/>) into the same folder.
- Use the database *northwindJDBC* instead of *Northwind*. This will be your source database. The username and password are *northwindJDBC*. To create a connection to *northwindJDBC* go to File → New → Database connection and enter the following information:

Database Connection

General

Advanced

Options

Pooling

Clustering

Connection name:
Source - NorthwindJDBC - MS SQL Server

Connection type:
MS SQL Server
MS SQL Server (Native)
MariaDB
MaxDB (SAP DB)
MonetDB
MySQL
Native Mondrian
Neoview
Nettezza
OpenERP Server
Oracle
Oracle RDB

Access:
Native (JDBC)
ODBC
JNDI

Settings

Host Name:
deti-sql-aulas.ua.pt

Database Name:
northwindJDBC

Instance Name:

Port Number:
1433

Username:
moreira

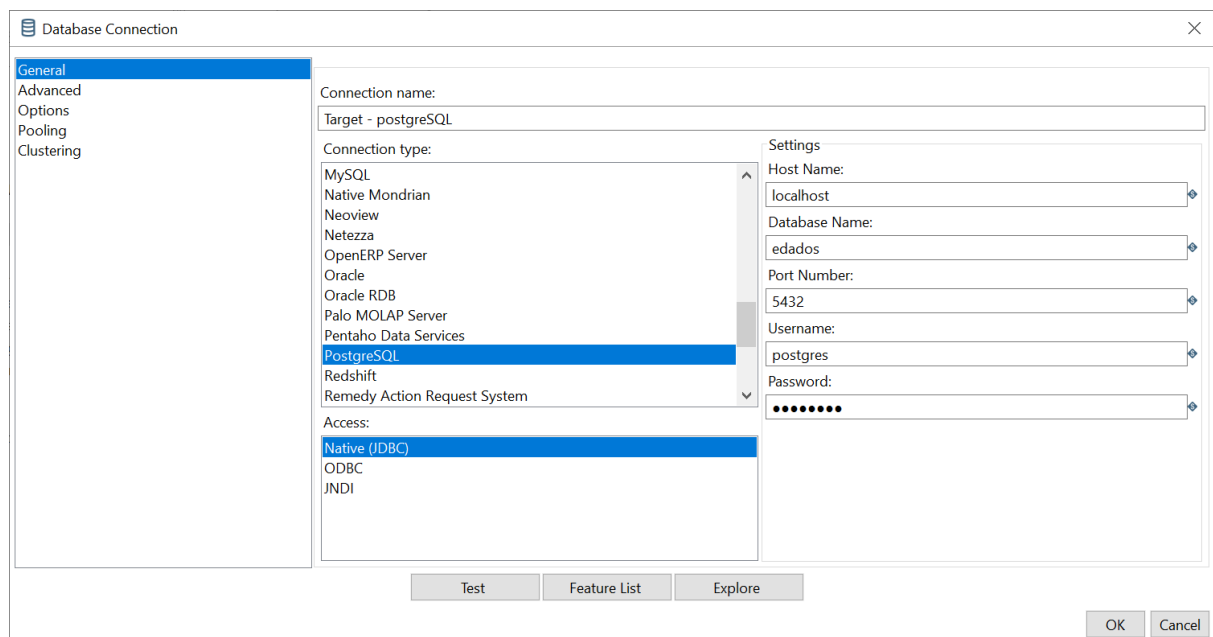
Password:
••••

☐ Use .. to Separate Schema and Table

Test Feature List Explore

OK Cancel

- Create a connection to PostgreSQL as follows. This will be your target database.



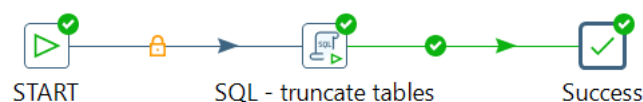
- Validate your transformations step by step, i.e., run a transformation, select the pane *Preview Data* in the *Execution Results* pane and click on each entry (operator) to see the intermediate results.

Quick start

1. Highlights on working with Kettle / PDI and PostgreSQL.

- Share your database connections (right-click on the connection name) to make them visible to all *Jobs* and *Transformations*.
- All table names and the attributes names in PostgreSQL must be in capital letters and surrounded by quotation marks (“). This is because PostgreSQL is case sensitive. The same does not hold for Microsoft SQL Server.
- The basic concepts are *Jobs* (workflow-like models for coordinating resources, execution, and dependencies of ETL activities) and *Transformations* (data flows):
<https://help.pentaho.com/Documentation/7.0/OL0/OY0/030/010>
- All steps in a transformation are started and run in parallel so the initialization sequence is not predictable.

2. Create a new Job to clear all tables in your *Data Mart* (the target database in PostgreSQL)

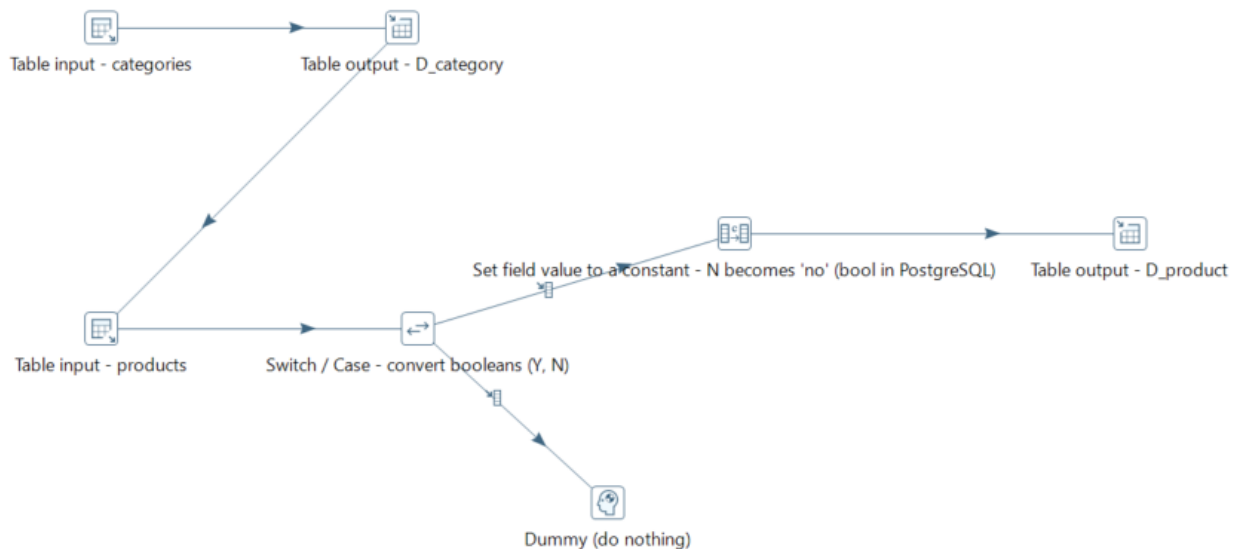


2.1. Go to *Design View* and add a *Start* entry (no need to select any automatic scheduling option).

2.2. Add an *SQL* entry and write the SQL statements to delete all data from your tables (TRUNCATE TABLE xyz CASCADE).

2.3. Add a *Success* entry to complete your job.

3. Create a new transformation to load the data into the dimension *Product* (tables *D_category* and *D_product*) using the data from the table *Categories* and *Products* in the Northwind database.



- 3.1. Go to *Design View* and add a *Table Input* entry to select the columns you need data from the table *Categories* in the Northwind database.

Table input

Step name: Table input - categories

Connection: Source - NorthwindJDBC

SQL:

```
SELECT
  CategoryID
, CategoryName
FROM Categories
```

Line 1 Column 0

Enable lazy conversion: ☐

Replace variables in script?: ☐

Insert data from step:

Execute for each row?: ☐

Limit size: 0

Buttons: Help, OK, Preview, Cancel

- 3.2. Add a *Table output* entry to create a mapping between the fields of the source table (*Categories*) and the target table (*D_category*). Do not forget to add quotation marks to “D_category” as well as to the names of attributes in the *Table Field* column (PostgreSQL database).

Table output

Step name: Table output - D_category

Connection: Target - postgresSQL [Edit... New... Wizard...]

Target schema: [Browse...]

Target table: "D_CATEGORY" [Browse...]

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	"ID_CATEGORY"	CategoryID
2	"CATEGORY_NAME"	CategoryName

[Get fields] [Enter field mapping]

[Help] [OK] [Cancel] [SQL]

- 3.3. Add a *Table Input* entry to select the columns you need data from the table *Products* in the Northwind database.
- 3.4. PostgreSQL does not accept Boolean values specified as 'Y' and 'No', but it accepts 'yes' and 'no', among other options. So, it is necessary to convert the values of *Discontinued* from the table *Products* into something recognized by PostgreSQL. Several alternatives exist. This example uses a *switch* entry and a *Set field to a constant* entry and migrates the products that are not discontinued.

Switch / case

Step name: Switch / Case - convert booleans (Y, N)

Field name to switch: Discontinued

Use string contains comparison: ☐

Case value data type: Boolean

Case value conversion mask:

Case value decimal symbol:

Case value grouping symbol:

Case values

#	Value	Target step
1	Y	Dummy (do nothing)
2	N	Set field value to a constant - N becomes 'no' (bool in PostgreSQL)

Default target step:

[Help] [OK] [Cancel]

Set field value to a constant

Step name: Set field value to a constant - N becomes 'no' (bool in PostgreSQL)

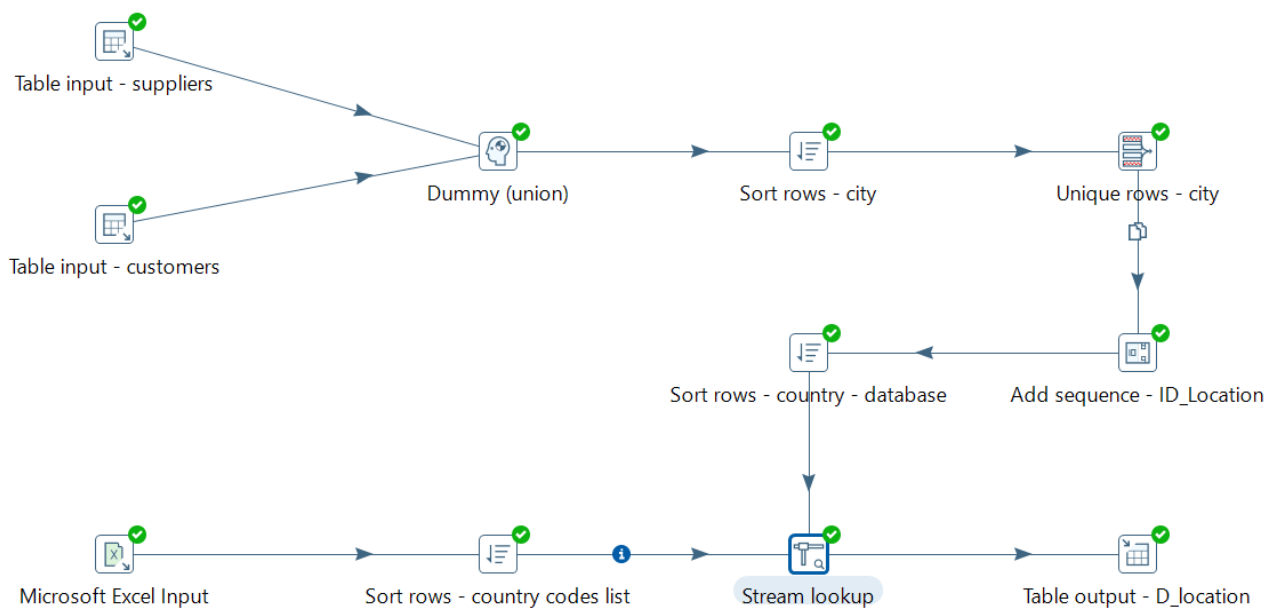
Use variable in constant: ☐

#	Field	Replace by value	Conversion mask (Date)	Set empty string?
1	Discontinued	'no'		N

Help OK Get Fields Cancel

3.5. Add a *Table output* entry to create a mapping between the fields of the data stream and the target table (*D_category*).

4. Create a new transformation to load the data into the dimension *Location*.



- This dimension must have one record for each distinct *City* in the tables *Suppliers* and *Customers*.
- The union of the values in both tables is performed by the *Dummy* entry and the tuples must be ordered before removing the duplicates using the *Unique rows* operator.
- As there is no attribute key for this dimension, it is necessary to use the entry *Add sequence* to create an auto number that will be assigned to the *ID_location*.
- The Microsoft Excel Input refers to the file *countryCodesList.xls*, which holds a table with the ISO country codes. These are the codes needed to fill the attribute *Country_code* in *D_location*.
- The rows of both data streams must be in the same order to perform a *Stream lookup* to create a data stream with the attributes *City*, *Region*, *PostalCode* and *Country*, plus the corresponding ISO-3 country codes.

Stream Value Lookup

Step name: Stream lookup

Lookup step: Sort rows - country codes list

The key(s) to look up the value(s):

#	Field	LookupField
1	Country	CountryName

Specify the fields to retrieve :

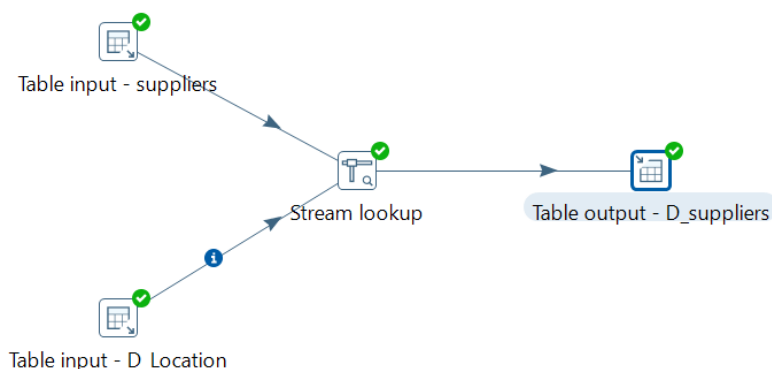
#	Field	New name	Default	Type
1	ISO3			String

☒ Preserve memory (costs CPU)
☐ Key and value are exactly one integer
☐ Use sorted list (i.s.o. hashtable)

- The attributes *state_name*, *region_name* e *continent_name* are not null, but they are empty in the source databases. You may change the definition of the corresponding attributes in the target database to allow null values, or you may use a *switch / case* entry and a *Set field value to a constant* to substitute null values by default values at your choice.

- The table *D_suppliers* (dimension Suppliers) has a foreign key to *D_location*. As the values of *ID_location* are generated automatically, it is not possible to create a simple mapping of a column in the source database to the foreign key column in the target database. So, it is necessary to add a *Lookup* entry to compare the values of the attribute *City* (table *Suppliers*) from the source database with the values of the attribute *City_name* in the target database (table *D_location*) and retrieve the corresponding keys (*ID_location*).

However, using a *Database Lookup* entry with a table from a PostgreSQL is not error free. An alternative is to use a *Table input* entry to retrieve the data from the table *D_location* (create an alias for the column names, if necessary) followed by a *Stream lookup* entry.



The same holds for the foreign key in the table *D_customers* (dimension Customers).

- The source to load the data into the fact table (*F_sales*) is the table *Orders* (source database). This is the table that holds the measures *Quantity*, *Unit price*, *Discount* and *Freight*, but this is only a part of the data that must be loaded into *F_sales*. So, the transformation that loads the data into the fact table must use several *Lookup* or *Join* entries to retrieve, step by step, all required data.

Documentation

Pentaho Data Integration (Kettle):

<https://wiki.pentaho.com/display/EAI/Latest+Pentaho+Data+Integration+%28aka+Kettle%29+Documentation>

Quick start: <https://intellipaat.com/blog/tutorial/pentaho-tutorial/getting-started-with-transformations/>