UNIVERSITATEA
„ALEXANDRU IOAN CUZA“
din IAŞI

FEAA
FACULTATEA DE ECONOMIE
ŞI ADMINISTRAREA AFACERILOR

Workshop strategy: ***T2 – ETL Tool***

Workshop theme: ***Data Integration using TALEND***

Team SDBIS03:

Procopciuc Irina

Ionescu Robert Florin

Chelaru Iustina (Laschi)

Table of contents

Table of figures

1. General information about Talend

Talend is an open-source software platform used for data integration and data management, being specialized in big data integration also. This tool provides features like cloud, big data, enterprise application integration, data quality, and master data management, having a unified repository to store metadata alongside different sources (Guru99, 2020). Talend provides us with multiple functions, but the one described in this paper will be the data integration feature. The main benefits of using Talend for data integration are (Guru99, 2020):

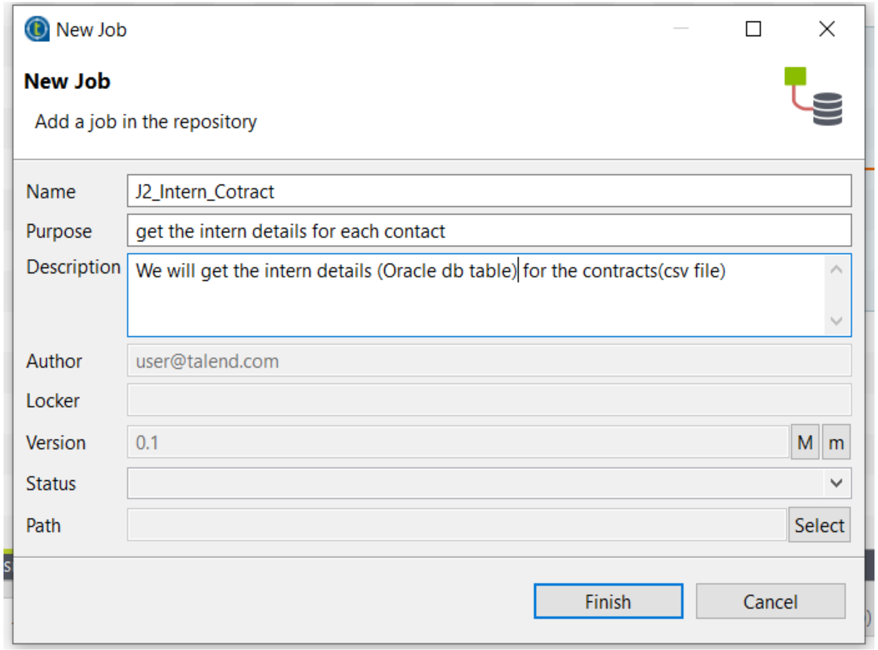
- *Agile integration* leads to a faster response to business requirements.
- *Team productivity*, making the application a very efficient tool to work with, having powerful versioning, analysis, and testing facilities.
- *Easy management* is visible through the advanced scheduling and monitoring features and also the real-time integration with dashboards.
- *Stay ahead in the competition*, the tool offers real-time updates of the used features.
- *Pay the lowest price of ownership* because Talend offers a subscription-based pricing model.

This ETL tool also offers Open Studio, an open-source for data integration and big data. Talend open studio helps with handling a huge amount of data with the help of big data components (Pedamkar, 2020).

2. Data integration example

For this workshop, we will use Talend Open Studio in order to create an example of data integration between two different sources, a CSV file, and an oracle table. The CSV file contains details about contracts (idcontract, startdate, enddate, salary, and bonuses). The table we will use from the Oracle database is the intern table which has the following columns: idintern, internfirstname, internlastname, faculty, position, and idcontract. The main purpose of this example is to find out which position has the highest salary and which one has the lowest and we can see that if we integrate the data from the sources mentioned above.

We will start by creating a new project in Talend Studio. After this operation is completed, it might take some time, we have to create a new job, specifying the name, purpose, and description. The name of our job will be J2_Intern_Contract, as shown in the figure below (Figure 1).



New Job

New Job

Add a job in the repository

Name: J2_Intern_Cotract

Purpose: get the intern details for each contact

Description: We will get the intern details (Oracle db table) for the contracts(csv file)

Author: user@talend.com

Locker:

Version: 0.1

Status:

Path:

Figure 1. Create new job

After creating the new job we can proceed with the other tasks. The created job can be seen in the left menu, under Job Designs from the repository section (Figure 2).

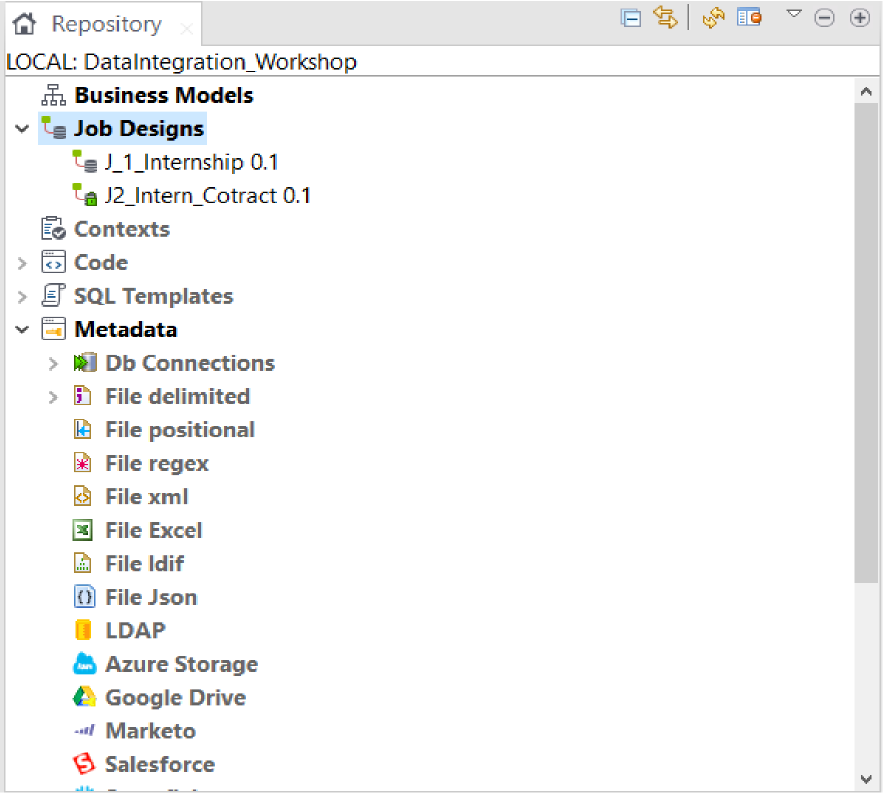
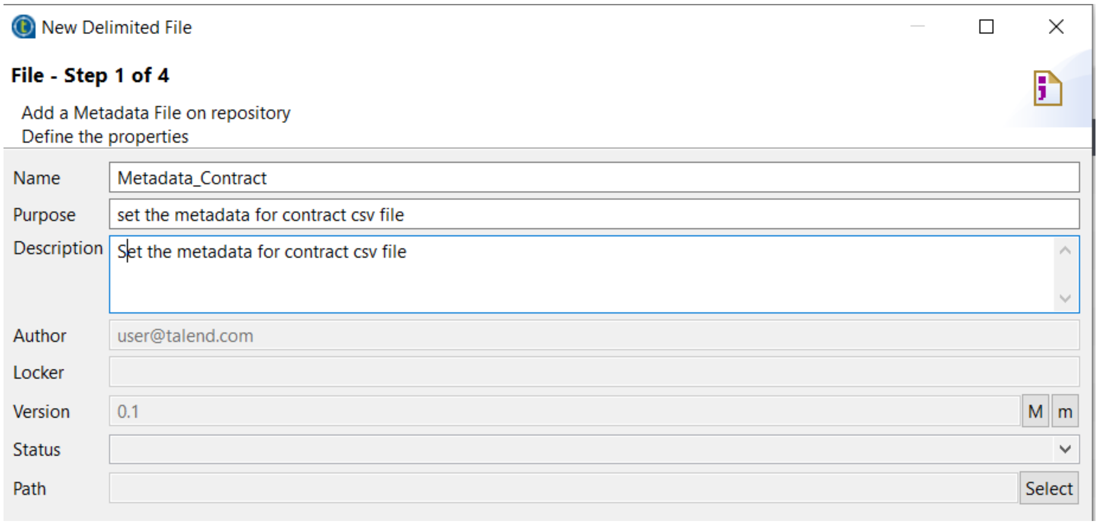


Figure 2. Repository section (left menu)

Besides the left section from Talend Studio, we also have a design view for each job. In this space, we can drag and drop any object from the palette, object that can be input file types, database objects, or output files. The first object we will use is a tFileInputDelimited, for our contract data. Before editing the component in order to give some details about where the file is located and the data format, we should create metadata for this type of file so that we can use it in other jobs also. The new metadata will be found in the left menu, under the Metadata tab. For this, we have to proceed with the completion of the four needed steps and at the end, we will have new metadata defined for delimited files that have the same structure and columns as contract file. The figures below will show all the needed steps to create new metadata.



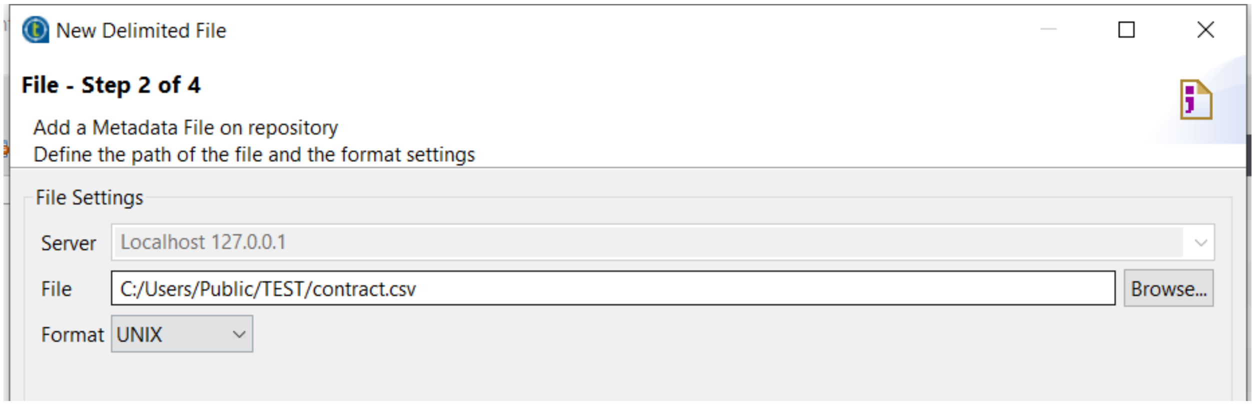
New Delimited File

File - Step 1 of 4

Add a Metadata File on repository
Define the properties

Name: Metadata_Contract
Purpose: set the metadata for contract csv file
Description: Set the metadata for contract csv file
Author: user@talend.com
Locker:
Version: 0.1
Status:
Path:

Figure 3. Create new metadata for delimited files (Step 1)



New Delimited File

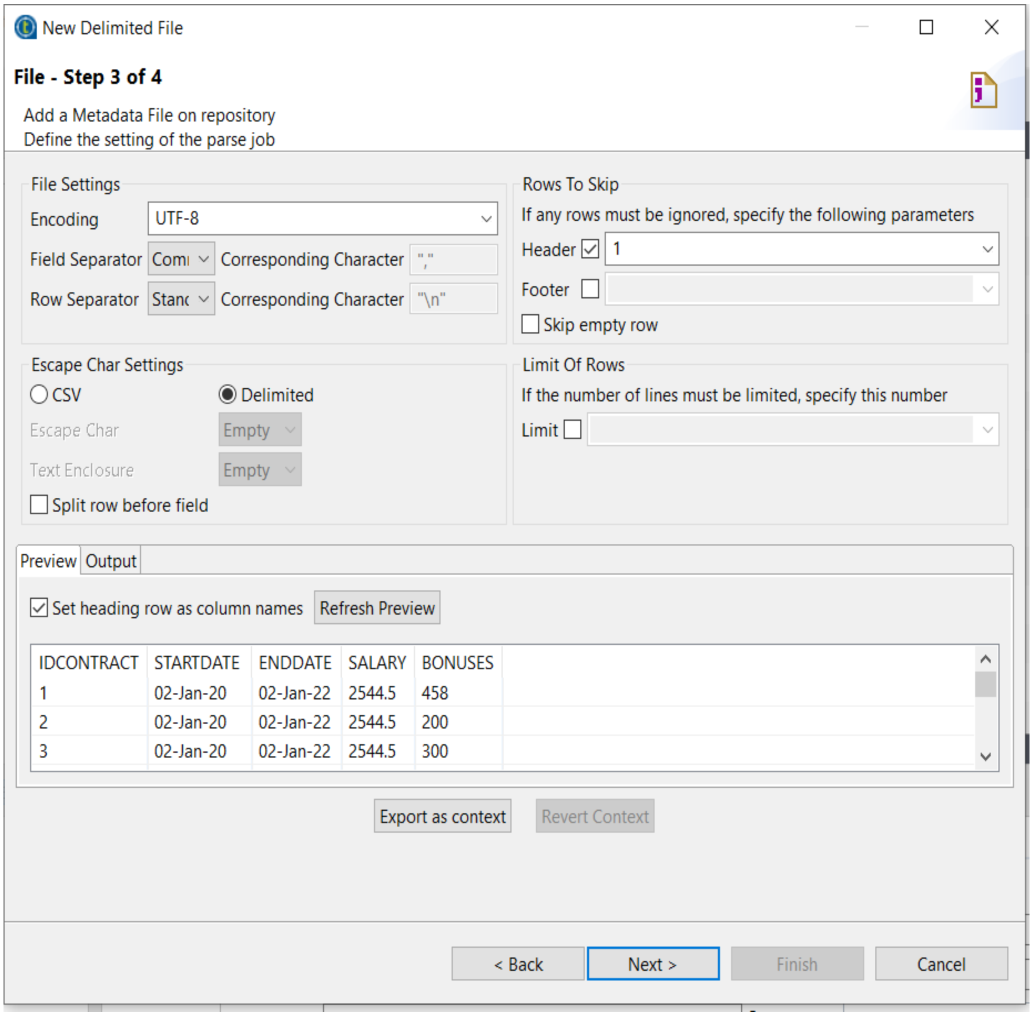
File - Step 2 of 4

Add a Metadata File on repository
Define the path of the file and the format settings

File Settings

Server: Localhost 127.0.0.1
File: C:/Users/Public/TEST/contract.csv
Format: UNIX

Figure 4. Create new metadata for delimited files (Step 2)



New Delimited File

File - Step 3 of 4

Add a Metadata File on repository
Define the setting of the parse job

File Settings

- Encoding: UTF-8
- Field Separator: Comı — Corresponding Character: ","
- Row Separator: Stanc — Corresponding Character: "\n"

Rows To Skip

If any rows must be ignored, specify the following parameters

- Header ☑ 1
- Footer □
- □ Skip empty row

Escape Char Settings

- ○ CSV ● Delimited
- Escape Char: Empty
- Text Enclosure: Empty
- □ Split row before field

Limit Of Rows

If the number of lines must be limited, specify this number

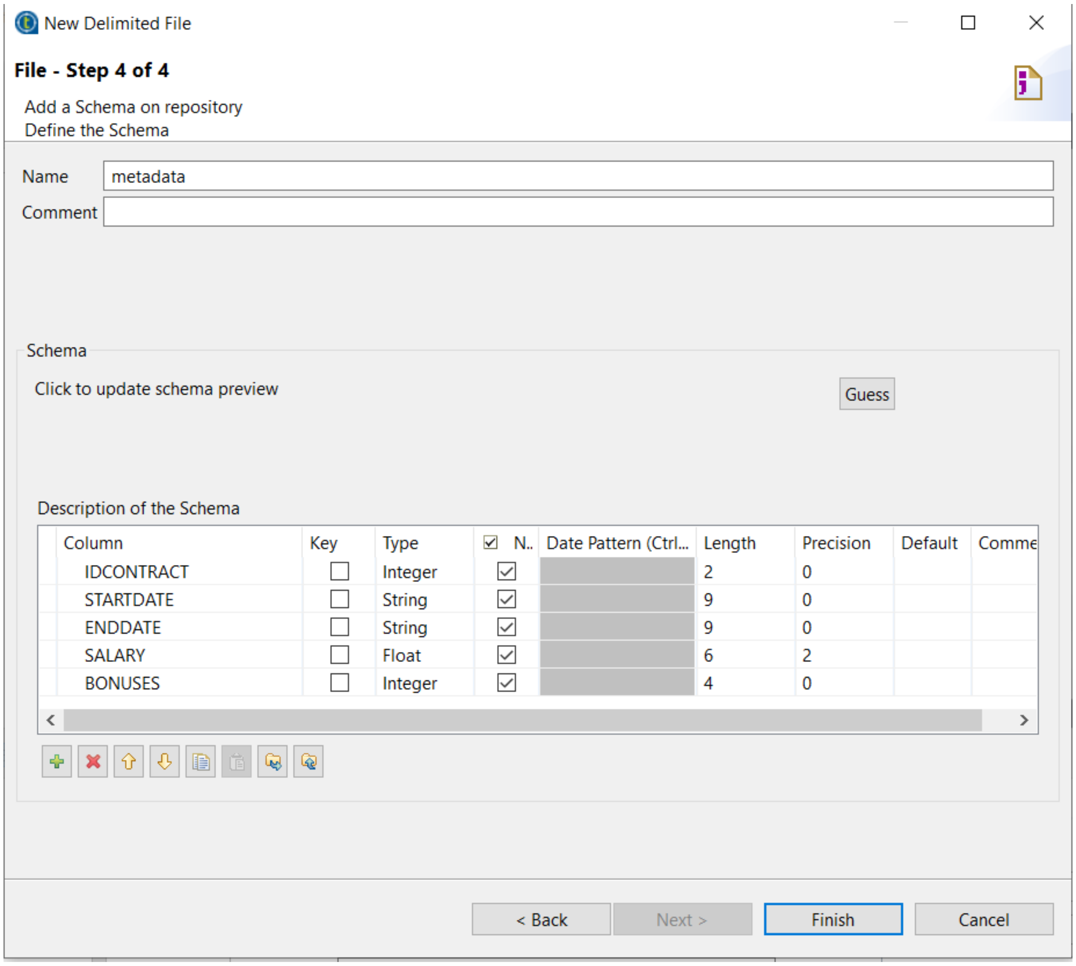
- Limit □

Preview | Output

☑ Set heading row as column names

IDCONTRACT	STARTDATE	ENDDATE	SALARY	BONUSES
1	02-Jan-20	02-Jan-22	2544.5	458
2	02-Jan-20	02-Jan-22	2544.5	200
3	02-Jan-20	02-Jan-22	2544.5	300

Figure 5. Create new metadata for delimited files (Step 3)



File - Step 4 of 4

Add a Schema on repository
Define the Schema

Name: metadata

Comment:

Schema

Click to update schema preview

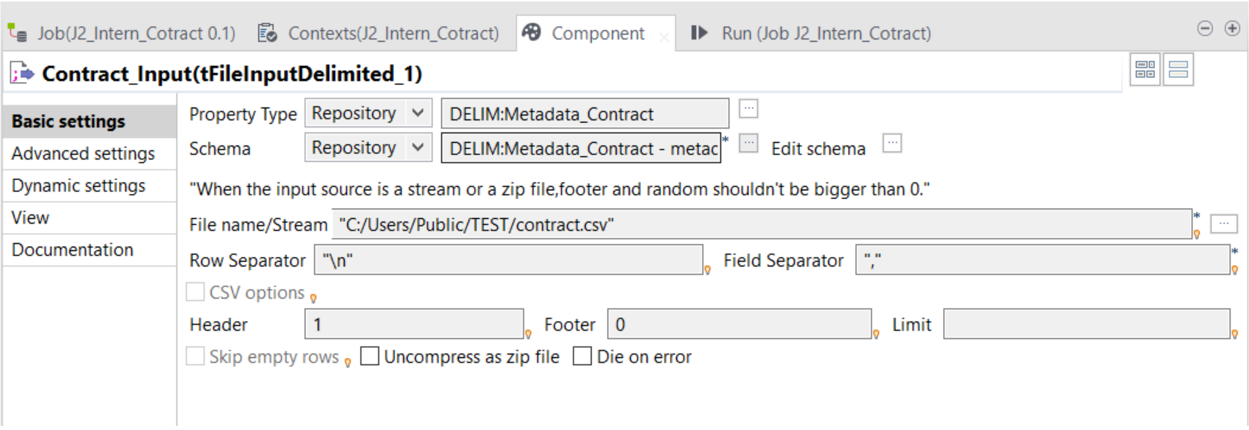
Description of the Schema

Column	Key	Type	☑ N..	Date Pattern (Ctrl...	Length	Precision	Default	Comme
IDCONTRACT	□	Integer	☑		2	0		
STARTDATE	□	String	☑		9	0		
ENDDATE	□	String	☑		9	0		
SALARY	□	Float	☑		6	2		
BONUSES	□	Integer	☑		4	0		

Figure 6. Create new metadata for delimited files (Step 4)

The first step to create a new metadata file delimited is to specify the name, purpose, and description, similar to the creating of a new job. In the second step, we will choose the path to the file we want to use for creating the new metadata. The third step is the most important. Here we have to specify the file separator if we have headers included in the selected file or any other characters that should be skipped. In the bottom section of the thirds' screen popup is the preview section, where we have a glimpse of the data from the file. The last step includes specifying the name of the schema to include on the repository and the description, which is basically the preview from earlier, but here we can edit the keys, data type, and other properties.

After pressing the finish button we are ready to go back to the component tab of our CSV file. Here we will have to choose the property type and since we created the previous file delimited, we will choose the repository option from the dropdown. The schema will be retrieved automatically since we created just one, but we can change it if we feel like it. The other fields from the figure above represent data that is retrieved from the specified schema and is in read-only format. If we wouldn't have chosen a schema, we would have to fill that manually each time we needed that type of configuration for a file.



Job(J2_Intern_Cotract 0.1)
Contexts(J2_Intern_Cotract)
Component
Run (Job J2_Intern_Cotract)

Contract_Input(tFileInputDelimited_1)

Basic settings
Advanced settings
Dynamic settings
View
Documentation

Property Type: Repository | DELIM:Metadata_Contract

Schema: Repository | DELIM:Metadata_Contract - metac | Edit schema

"When the input source is a stream or a zip file,footer and random shouldn't be bigger than 0."

File name/Stream: "C:/Users/Public/TEST/contract.csv"

Row Separator: "\n" Field Separator: ";"

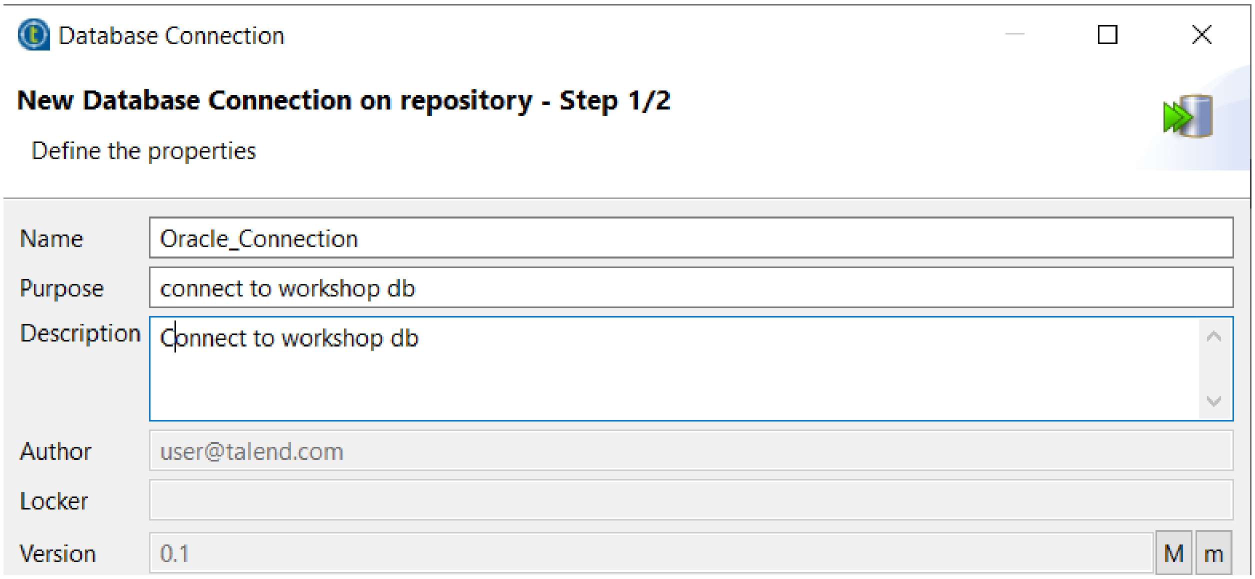
□ CSV options

Header: 1 Footer: 0 Limit:

□ Skip empty rows □ Uncompress as zip file □ Die on error

Figure 7. Component tab for tFileInputDelimited

Since we have our first input data source in place, it's time to configure the next one. In our case, it will be a tDBInput. If we double click the input we'll be able to edit some properties and define the specific database type. We will choose Oracle from the database dropdown and for property type, we will also use a database connection saved in the metadata tab, but we have to create it first. To create a new database connection we need to follow two steps. The first one includes, as we got used to it, specifying the connection name, purpose, and a description. And the second step includes the specification of the actual connection data. Here we will choose Oracle with SID and we will fill in the database version, the credentials, the server, and port, and also the side and schema name. It is useful to also test the connection, to be sure we didn't mismatch any of the fields. The whole process is captured in the two below figures.



Database Connection

New Database Connection on repository - Step 1/2

Define the properties

Name: Oracle_Connection

Purpose: connect to workshop db

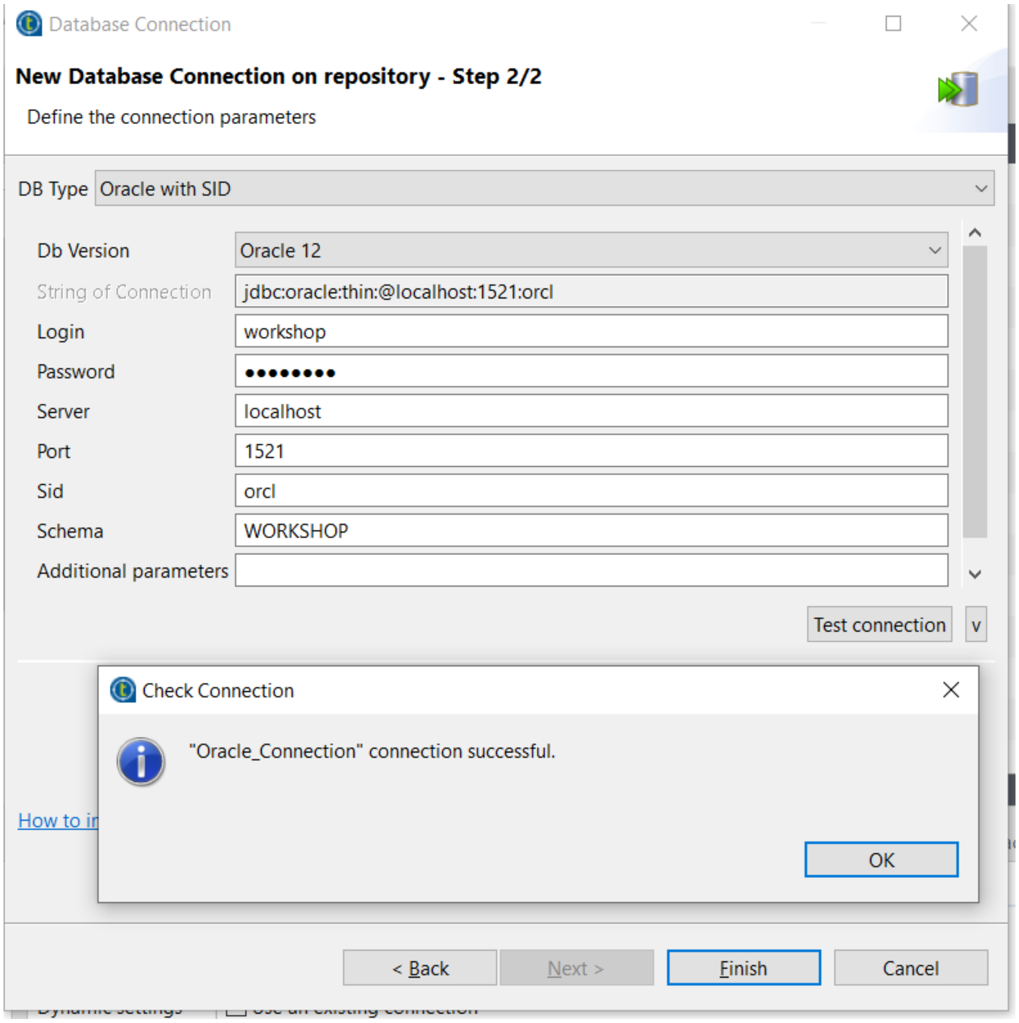
Description: Connect to workshop db

Author: user@talend.com

Locker:

Version: 0.1

Figure 8. Create new database connection (Step 1)



New Database Connection on repository - Step 2/2

Define the connection parameters

DB Type: Oracle with SID

Db Version: Oracle 12

String of Connection: jdbc:oracle:thin:@localhost:1521:orcl

Login: workshop

Password: ●●●●●●●●

Server: localhost

Port: 1521

Sid: orcl

Schema: WORKSHOP

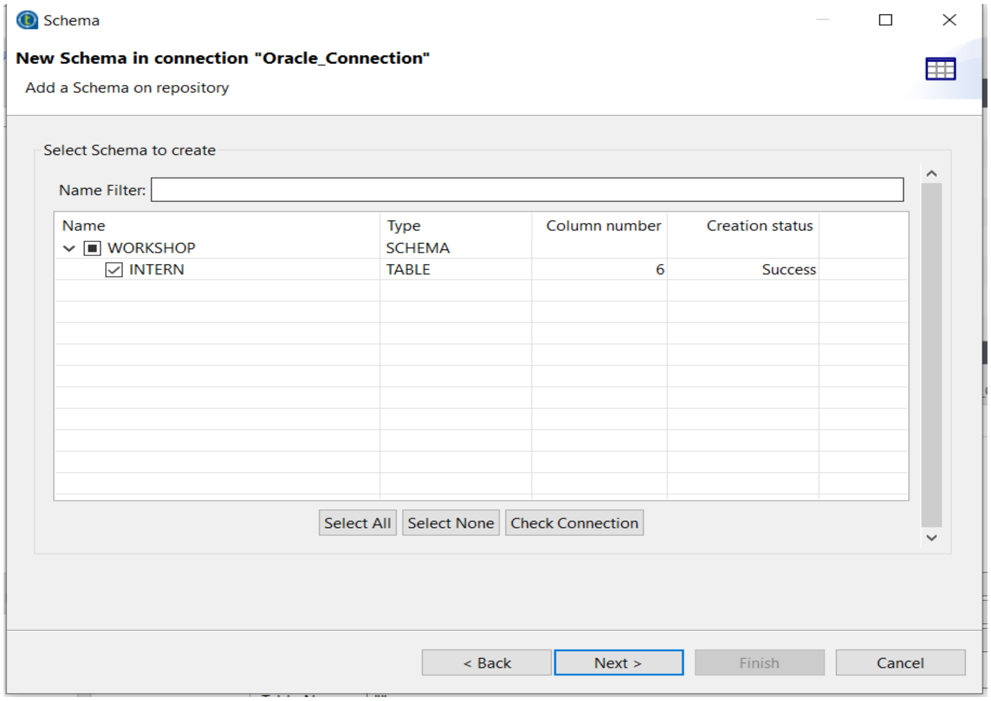
Additional parameters:

Check Connection

"Oracle_Connection" connection successful.

Figure 9. Create new database connection (Step 2)

Now that we have a connection established, we have to create a new schema for the intern table and also a query to retrieve the data from the database. The first step is the schema. Here, the first step is to filter the database object types, but we will leave all of them, so we go on next to the second step. On this popup, we have to select the intern table, as shown in the figure below.



New Schema in connection "Oracle_Connection"

Add a Schema on repository

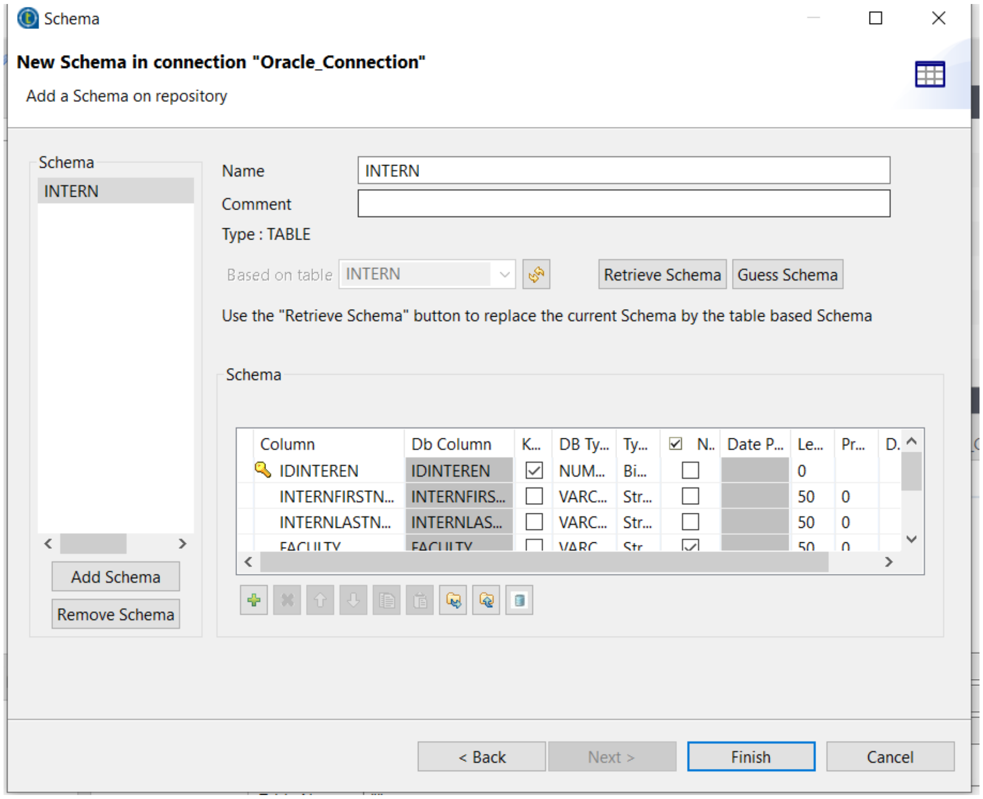
Select Schema to create

Name Filter:

Name	Type	Column number	Creation status
WORKSHOP	SCHEMA		
☑ INTERN	TABLE	6	Success

Figure 10. Create new database schema in DB connection (Step 1)

Afterward, we will get a preview of the schema and we can edit the key, data type, or other attributes of each column.



New Schema in connection "Oracle_Connection"

Add a Schema on repository

Schema: INTERN

Name: INTERN

Comment:

Type : TABLE

Based on table: INTERN

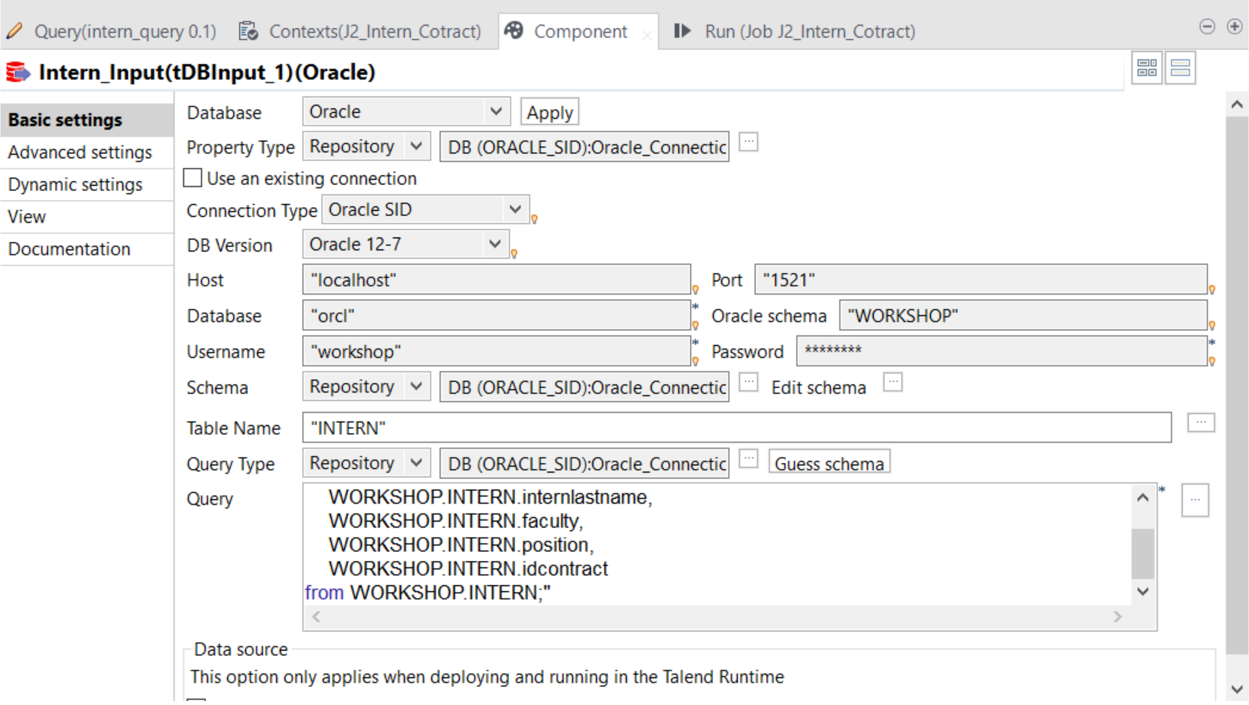
Use the "Retrieve Schema" button to replace the current Schema by the table based Schema

Schema

Column	Db Column	K...	DB Ty...	Ty...	☑ N..	Date P...	Le...	Pr...	D.
IDINTEREN	IDINTEREN	☑	NUM...	Bi...	□		0		
INTERNFIRSTN...	INTERNFIRS...	□	VARC...	Str...	□		50	0	
INTERNLASTN...	INTERNLAS...	□	VARC...	Str...	□		50	0	
FACULTY	FACULTY	□	VARC	Str	☑		50	0	

Figure 11. Create new database schema in DB connection (Step 2)

The next step is to come back to the oracle input from the design view and in the component tab, we can retrieve the details from the newly created database connection. The property type will be taken from the repository and that represents the actual database connection. The next fields until schema are read-only and will be retrieved from the metadata we created. Since we have a schema also for the oracle connection for the intern table, we will choose it in the schema field. The last thing we have to specify is the query, and we will also retrieve it from the repository.



Intern_Input(tDBInput_1)(Oracle)

Database: Oracle
Property Type: Repository | DB (ORACLE_SID):Oracle_Connectic
□ Use an existing connection
Connection Type: Oracle SID
DB Version: Oracle 12-7
Host: "localhost" Port: "1521"
Database: "orcl" Oracle schema: "WORKSHOP"
Username: "workshop" Password: ********
Schema: Repository | DB (ORACLE_SID):Oracle_Connectic
Table Name: "INTERN"
Query Type: Repository | DB (ORACLE_SID):Oracle_Connectic
Query:

```
    WORKSHOP.INTERN.internlastname,
    WORKSHOP.INTERN.faculty,
    WORKSHOP.INTERN.position,
    WORKSHOP.INTERN.idcontract
from WORKSHOP.INTERN;"
```

Data source
This option only applies when deploying and running in the Talend Runtime

Figure 12. Component tab for tDBInput

Now that we have both input files, we need a new component that can put these two together. For this operation, we will use tMap, a component that transforms and routes data from single and multiple sources to single or multiple destinations. After adding the tMap to the design view, we'll have to add new rows to connect all the elements. This thing can be done by right-clicking on the input component and then linking them to the tMap component like in the figure below.

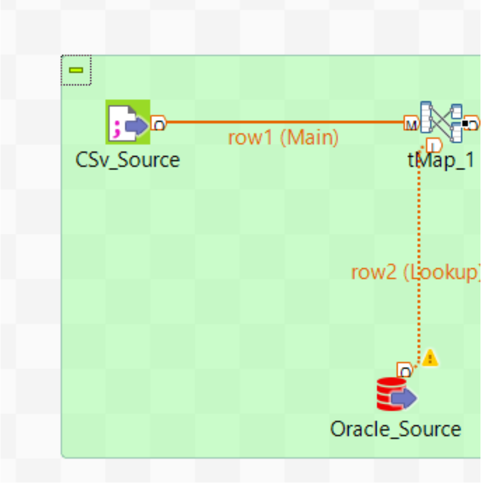
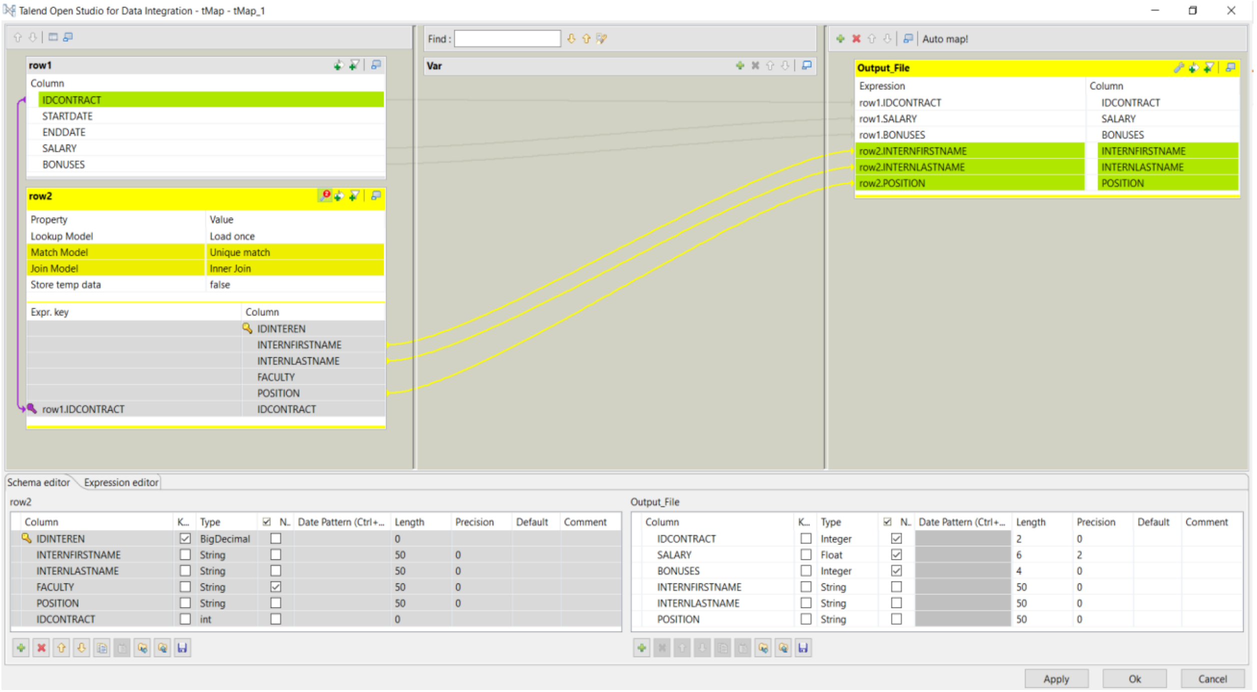


Figure 13. tMap on design view

A double click on the tMap component will open a new window where we can link the two inputs together and define the output file. It is important to specify for row2 that the matching model is a unique match and the join model is inner join, otherwise, the output file will contain too many records and it will not be representative for our analysis. We will drag and drop fields from the input files to the output file. In the figure below we see that our output file will contain idcontract, salary, bonuses, internfirstname, internlastname, and position. Also, we have to be very careful that the data types from the input are the same as the ones from the output so we won't get any error regarding this.



row1

Column
IDCONTRACT
STARTDATE
ENDDATE
SALARY
BONUSES

row2

Property	Value
Lookup Model	Load once
Match Model	Unique match
Join Model	Inner Join
Store temp data	false

Expr. key	Column
	IDINTEREN
	INTERNFIRSTNAME
	INTERNLASTNAME
	FACULTY
	POSITION
row1.IDCONTRACT	IDCONTRACT

Output_File

Expression	Column
row1.IDCONTRACT	IDCONTRACT
row1.SALARY	SALARY
row1.BONUSES	BONUSES
row2.INTERNFIRSTNAME	INTERNFIRSTNAME
row2.INTERNLASTNAME	INTERNLASTNAME
row2.POSITION	POSITION

row2

Column	K...	Type	☑ N..	Date Pattern (Ctrl+...	Length	Precision	Default	Comment
IDINTEREN	☑	BigDecimal	□		0			
INTERNFIRSTNAME	□	String	□		50	0		
INTERNLASTNAME	□	String	□		50	0		
FACULTY	□	String	☑		50	0		
POSITION	□	String	□		50	0		
IDCONTRACT	□	int	□		0			

Output_File

Column	K...	Type	☑ N..	Date Pattern (Ctrl+...	Length	Precision	Default	Comment
IDCONTRACT	□	Integer	☑		2	0		
SALARY	□	Float	☑		6	2		
BONUSES	□	Integer	☑		4	0		
INTERNFIRSTNAME	□	String	□		50	0		
INTERNLASTNAME	□	String	□		50	0		
POSITION	□	String	□		50	0		

Figure 14. tMap edit view

After filling in all the needed options, we can press the apply button and go back to the design view. Here, we need to add an output file and for this, we choose the tFileOutputExcel.

Then we link it with the tMap using the output table we designed in the figure above. The design view after running the job show looks something like this:

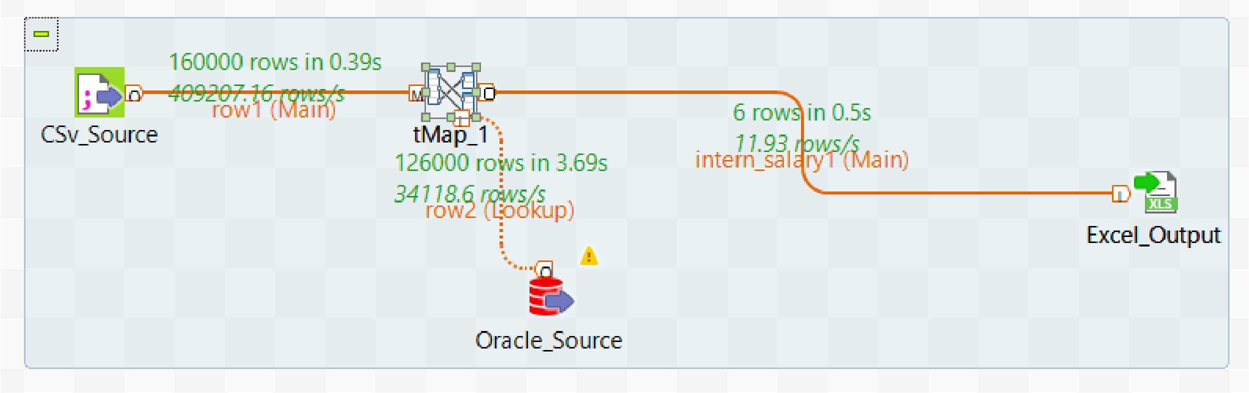
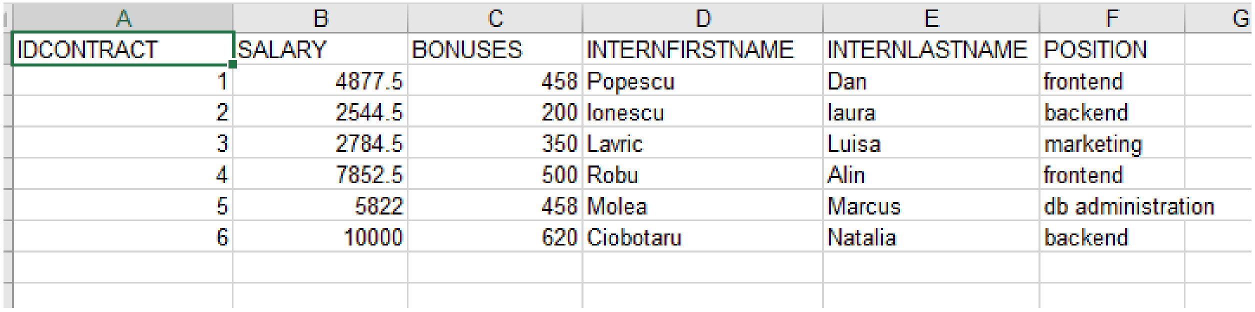


Figure 15. Final design view

Here we can see that our output file has 6 rows, the matches between the two input files, and it looks like in the figure below:



IDCONTRACT	SALARY	BONUSES	INTERNFIRSTNAME	INTERNLASTNAME	POSITION
1	4877.5	458	Popescu	Dan	frontend
2	2544.5	200	Ionescu	laura	backend
3	2784.5	350	Lavric	Luisa	marketing
4	7852.5	500	Robu	Alin	frontend
5	5822	458	Molea	Marcus	db administration
6	10000	620	Ciobotaru	Natalia	backend

Figure 16. Output file data

We can observe in this output file that for our example the position with the highest salary is the backend and the one with the lowest is marketing.

Bibliography

Guru99. (2020). *https://www.guru99.com/talend-tutorial.html*. Retrieved from Talend Tutorial for Beginners: What is Talend ETL Tool [Example].

Pedamkar, P. (2020). *https://www.educba.com/talend-data-integration/*. Retrieved from Talend data Integration.

Talend, https://www.talendforge.org/