

TALEND Interview questions and Answers

1.(<http://www.deepinopensource.com/talend-interview-questions/>)

1. Talend – Merge multiple files into single file with sorting operation.
2. Loading Fact Table Using Talend
3. ROWNUM Analytical Function in Talend
4. SCD-2 Implementations in Talend
5. Deployment strategies in Talend
6. Custom Header Footer in Talend
7. Data Masking Using Talend
8. How to use Shared DB Connection in Talend
9. Load all rows from source to target except last 5
10. Late Arriving Dimension Using Talend
11. Date Dimension Using Talend
12. Dynamic Column Ordering Of Source File Using Talend
13. Incremental Load Using Talend
14. Getting Files From FTP Server
15. Initializing Context At Run Time Using Popup
16. User Define Function In Talend
17. Calling DB Sequence From Talend

2.(<http://www.talendutorials.com/talend-interview-questions>)

1. Difference between tAggregatedRow and tAggregateSortedRow in Talend
2. How to resume job execution from same location if job get failed in Talend
3. How to execute more than one sub jobs parallel in Talend
4. How to iterate filename and directories in Talend
5. What is the difference between OnSubjobOK and OnComponentOK in Talend
6. How can you pass a value form parent job to child job in Talend
7. How to call stored procedure and function in Talend Job
8. How to export job and execute outside from Talend Studio
9. How to pass value from outside in Talend
10. Can I define schema of database or tables at run time
11. What is tReplicate in Talend
12. What is tUnite in Talend Open Studio
13. How to optimize talend job to stop outOfMemory runtime error
14. How to optimize Talend Performance
15. How to execute multipule SQL statements with one component in Talend
16. What is tSystem component in Talend

17. Can I execute multiple commands at one time with a tSystem component
18. What is difference between tMap and tFilterrow in Talend

3.(<http://www.cram.com/flashcards/talend-interview-questions-5197224>)

1. What is the difference between the ETL and ELT components of Talend Open Studio?
2. How does one deploy Talend projects?
3. What are the elements of a Talend project?
4. What is the most current version of Talend Open Studio?
5. How do you implement versioning for Talend jobs?
6. What is the tMap component?
7. What is the difference between the tMap and tJoin components?
8. Which *component* is used to sort data?

4.(<http://msureshreddy.blogspot.in/2013/08/talend-interview-questions.html>)

1. What is Talend ?
2. What is difference between ETL and ELT components of Talend ?
3. How to deploy talendprojects ?
4. What are types of available version of Talend ?
5. How to implement versioning for talendjobs ?
6. What is tMapcomponent ?
7. What is difference between tMap and tJoincomponents ?
8. Which component is used to sort that data ?
9. How to perform aggregate operations/functions on data in talend ?
10. What types of joins are supported by tMapcomponent ?
11. How to schedule a talendjob ?
12. How to runs talend job as web service ?
13. How to Integrate SVN with Talend ?
14. How to run talend jobs on Remote server ?
15. How to pass data from parent job to child jobs through trunjobcomponent ?
16. How to load context variables dynamically from file/database ?
17. How to run talend jobs in Parallel ?
18. What is Context variables ?
19. How to export a talendjob ?

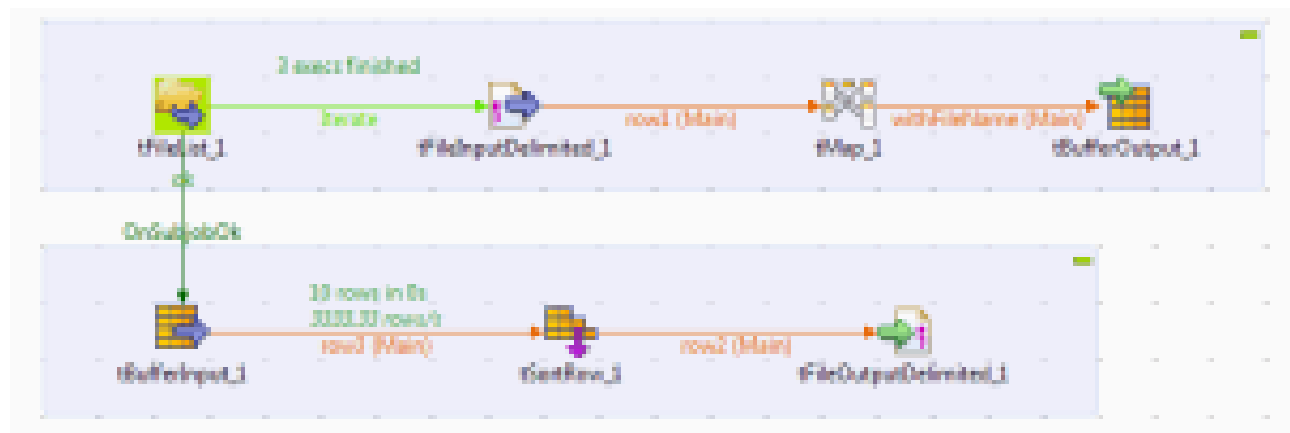
5.(Talend best practices)

1. Talend workspace path should not contain any spaces.

2. Never forget to perform Null Handling.
 3. Create Repository Metadata for DB connections and retrieve database table schema for DB tables.
 4. Use Repository Schema for Files/DB and DB connections.
 5. Create Database connection using t<Vendor>Connection component and use this connection in the Job. Do not make new connection with every component.
 6. Always close the connection to database using t<Vendor>Close component.
 7. Create a Repository Document corresponding to every Talend job including revision history.
 8. Provide Sub Job title for every sub job to describe the sub job purpose/objective.
 9. Avoid Hard Coding in Talend Job component. Instead use Talend context variables.
 10. Create Context Groups in Repository
 11. Use Talend.properties file to provide the values to context variables using tContextLoad.
 12. Create Variables in tMap and use the variables to assign the values to target fields.
 13. Create user routines/functions for common transformation and validation.
 14. Develop Talend job iteratively.
 15. Always Exit Talend open studio before shutting down the PC.
 16. Always rename Main Flows in Talend Job to meaningful names.
 17. Always design Talend jobs by keeping performance in mind.
- Talend Job Design - Performance Optimization Tips
1. Remove Unnecessary fields/columns ASAP using tFilterColumns component.
 2. Remove Unnecessary data/records ASAP using tFilterRows component
 3. Use Select Query to retrieve data from database
 4. Use Database Bulk components
 5. Store on Disk Option
 6. Allocating more memory to the Jobs
 7. Parallelism
 8. Use Talend ELT Components when required
 9. Use SAX parser over Dom4J whenever required
 10. Index Database Table columns
 11. Split Talend Job to smaller Subjobs

1. Talend – Merge multiple files into single file with sorting operation.

Scenario: Merging multiple input sources into a single target along with the file names with a additional column and sorting operations on all files with in the flow
The snapshot below shows the overall mapping.



As a source we have taken tFileList which will pull all our source files from the specified directory, then tFileInput Component will read all files one by one. In source file path you have to specify the global variable which holds the address of the current file in process using–

File name/Stream `((String)globalMap.get("tFileList_1_CURRENT_FILEPATH"))`

`((String)globalMap.get("tFileList_1_CURRENT_FILEPATH"))`

Later **tMap** is used to assign one extra column which holds the name of the file and again name of the file is to be retrieve using global variable in our case it is

withFileName	
Expression	Column
row1.first	first
row1.last	last
<code>((String)globalMap.get("tFil...</code>	newColumn

`((String)globalMap.get("tFileList_1_CURRENT_FILE"))`

After completion of first job we fire aonSubJobOk trigger to do our rest work.
now,

tBufferInput will hold all the source file data at once and later tBufferInput will pull all the buffered data produced by tBufferOutput.

Finally, tSort Component will sort all the three source file data using some column as a key and tFileout-component will produce the final output.

that's all, now you can create a job and simply run it—>>>

2.Loading Fact Table Using Talend

In this post we will discuss how to load a fact table in a data warehouse using your dimension table and the data staged in a staging table. We will just show you what is the procedure of loading and further complexities depends upon your business requirements.

In our example we took THREE dimension tables.

1. DIM_BOOK
2. DIM_CUSTOMER
3. DIM_TIME

All of the dimension tables are SCD-type implemented other than TIME DIMENSION.

STEPS:—>>>

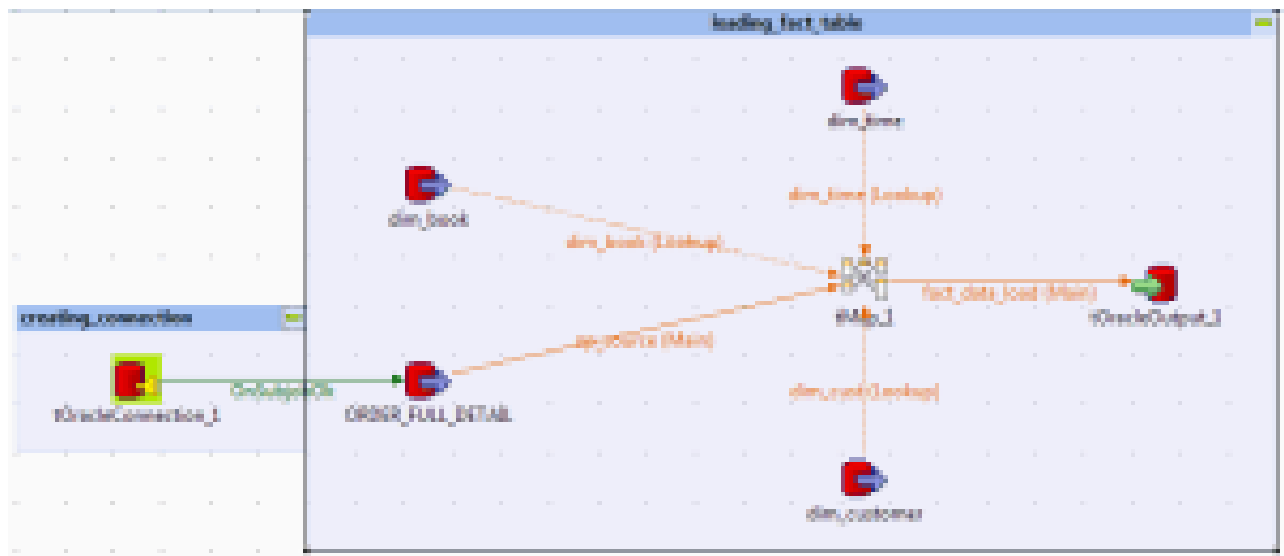
1) create an connection first as i am using oracle as a database that's why i used tOracleConnection.

2) After Successful connection run the further part.

From Dimension tables you have to fetch the data using Query Editor and place a condition where END_DATE is null, if you have implemented SCD type-2 This will bring all those records who have currently validated state.

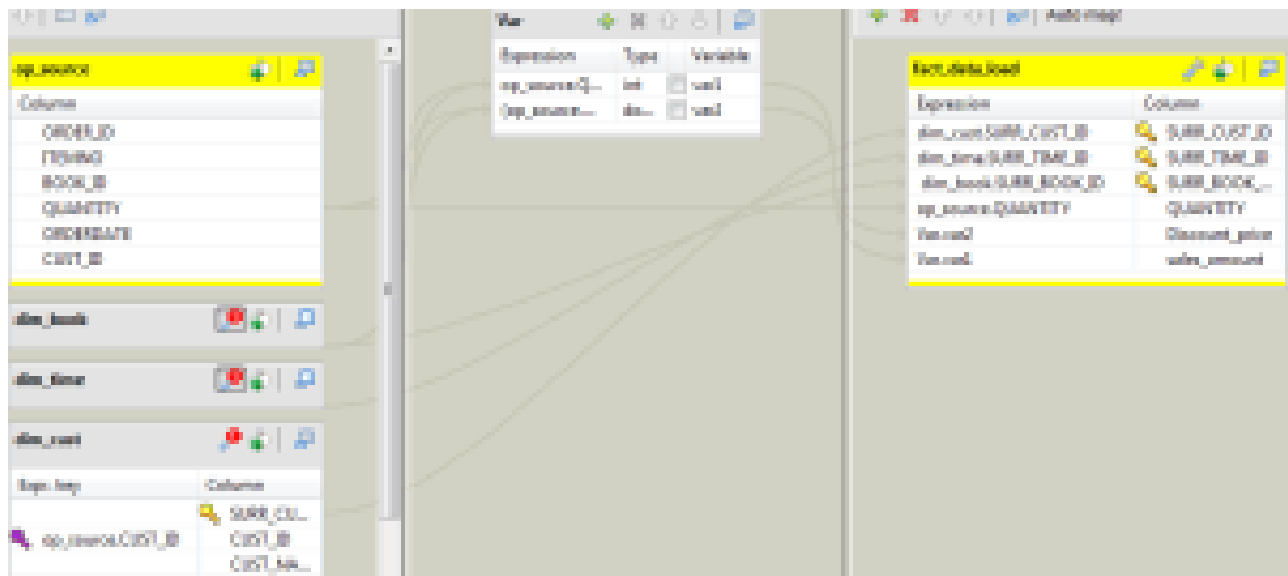
Now Fetch the OLTP data stored currently in Staging Table, and put all connections in tMap.

REMEMBER—your staging table that has to be loaded must be first linked then only it will work fine and all other dimensions connected will work as a LOOKUP.



Now in tMap join all the dimensions with your source data using the keys and fetch down the SURROGATE_KEY and put all those skeys in the fact table. In my join condition i have used inner join as a join method.

In tmap component i have use some calculations to find out what are the percentage in discount and total value for the order . It may be anything depend upon your requirements.



note:- Just keep in mind data type conversion you have to keep in mind other wise it'll give you trouble a lot in my case i just converted my data types in staging itself.

3.ROWNUM Analytical Function in Talend

Analytical funtions are quite useful in SQL and help us avoid extra coding.We encountered a similar scenario where in we had to implement a function similar to the ROWNUM() OVER (partition by clause) analytical functionality in Talend.

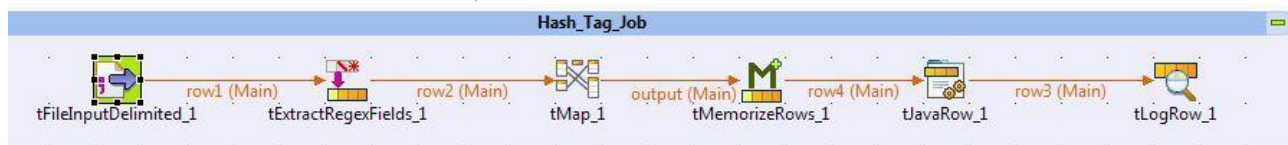
We ll give an overview of the Talend Job we created in order to attain the functionality:

Prerequisites:

Create a context variable and Initialize it to 0.

Scenario: Implement the ROW_NUMBER() over (partition by) analytical functionality in Talend

The image below gives the overview of the job, the components to focus on would be the tMemorizeRows and the tJavaRow component.



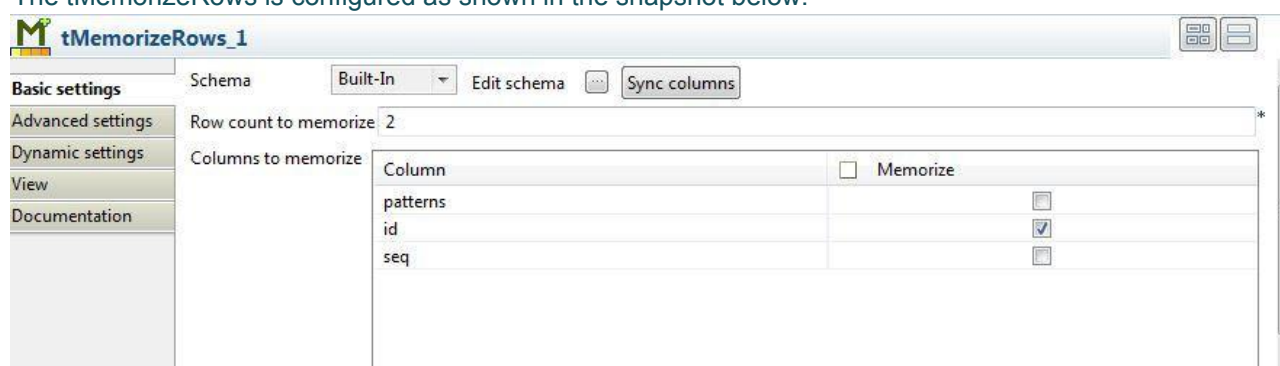
The input file is as shown in the snapshot below:

```
ID,NAME
1,Ankit
1,kansal
2,sarthak
2,singhal
1,Vaibhav
1,utkarsh
2,Arun
3,abc
3,def
```

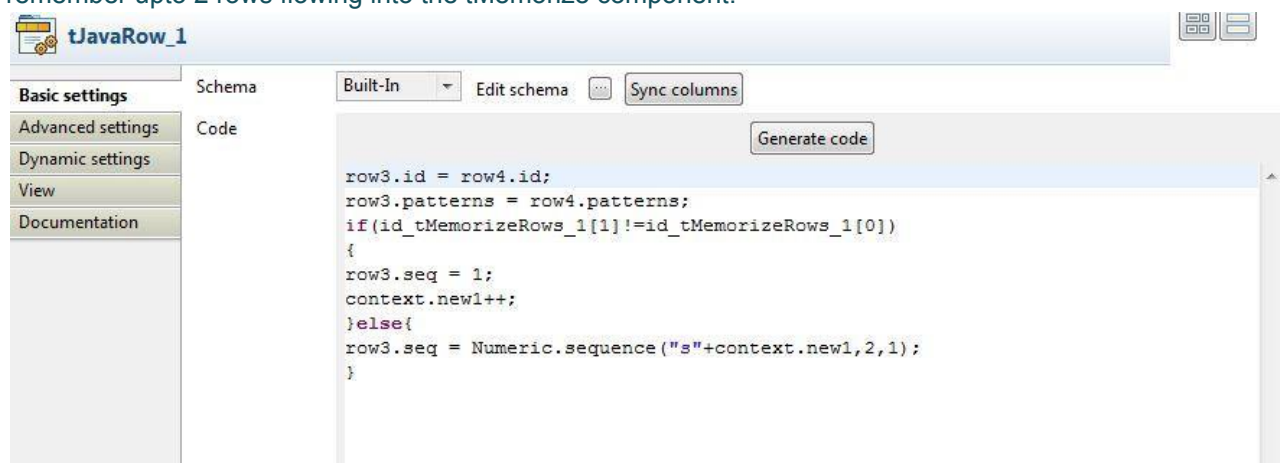
We need output to look something like this:

```
ID,NAME,ROWNUM
1,Ankit,1
1,kansal,2
1,Vaibhav,3
1,utkarsh,4
2,Sarthak,1
2,singhal,2
2,Arun,3
3,abc,1
3,def,2
```

The tMemorizeRows is configured as shown in the snapshot below:



This configuration specifies that we MEMORIZE the ID column , and the row count specifies that we remember upto 2 rows flowing into the tMemorize component.

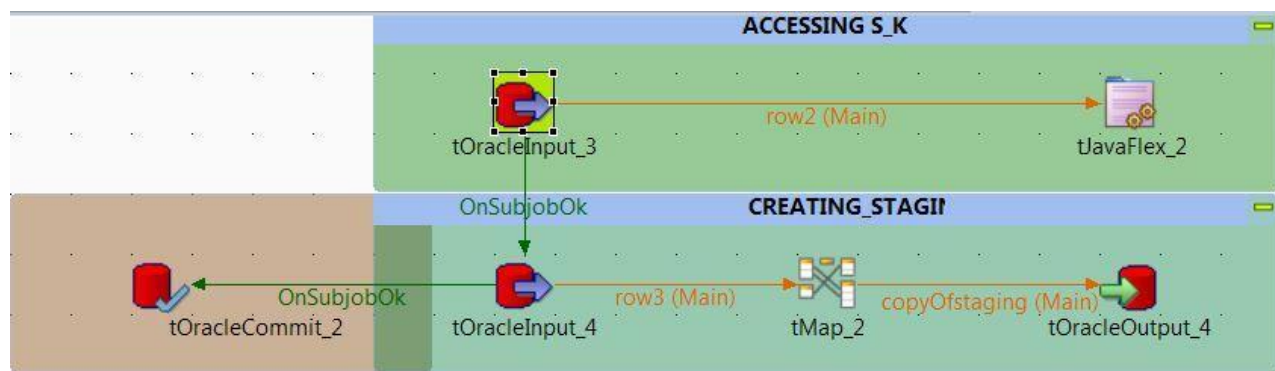


The most important part of the job ,is configuring the tJavaRow component ,wherein we specify the logic for implementing this analytical function.

The java code compares the previous and current row using the index for tMemorizeRows as shown in the snapshot..

4. SCD-2 Implementations in Talend

CREATING STAGING TABLE



1) From tOracleInput_3 bring the max surrogate key present in your dimension table using query
“SELECT max(user_profile_sur_key) FROM ANKIT_KANSAL.user_profile”

USER PROFILE IS OUR DIMENSION TABKE

2) Create a context variable snumber of integer type and assign a default value 0 and in the next step assign the max surrogate key to the variable using tJavaFlex component.

Variables	Values as tree	Values as table	
Name	Source	Type	Script code
snumber	built-in	int Integer	context.snumber

tJavaFlex_2

Basic settings
Advanced settings
Dynamic settings
View
Documentation

Main code

```

if(row2.user_profile_sur_key==null)
{
context.snumber=0;
}
else
{
context.snumber=row2.user_profile_sur_key;
}

```

4) tOracleInput4 takes all your data from the source oracle system KEEP ALL THE SOURCE AS A VARCHAR2 FORMAT ONLY.

5) using tMap component calculate your date and also you can find out your GOOD and BAD records depending upon the business requiremetns.

AS FOLLOWS:-

```

row3.TWEET_CREATED_AT .substring(8,10).concat("-").
concat(row3.TWEET_CREATED_AT
.substring(4,7)).concat("-").
concat(row3.TWEET_CREATED_AT .substring(27, 31))

```

Var	Valu
row3.TWEET_ID	null
row3.TWEET_C...	null
row3.TEXT	null
row3.SOURCE	null
row3.RETWEET...	null


```
(context.snumber=context.snumber+1).toString()
```

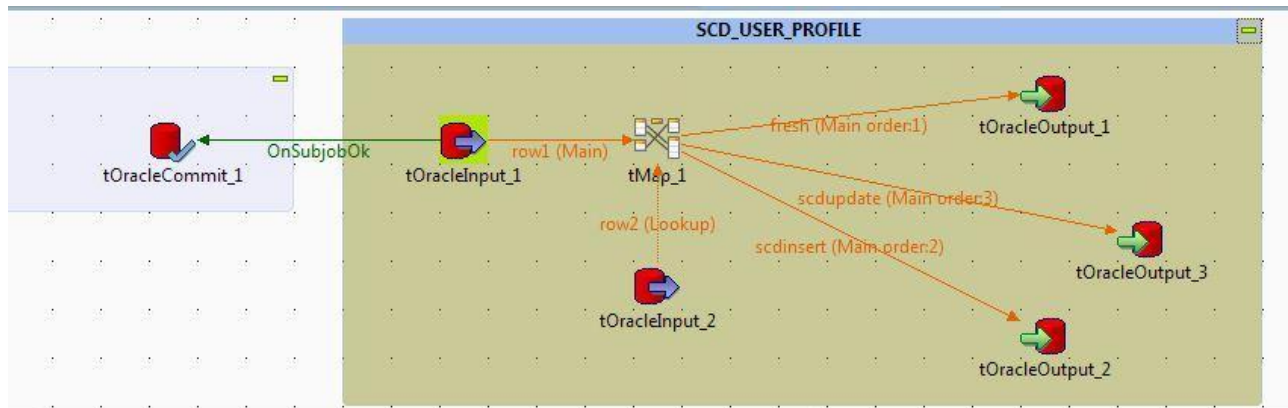
In the value part perform these operations

```
(Mathematical.NUM((row3.TWEET_ID==null)? "~":row3.TWEET_ID)) ==1)? ((row3.TEXT==null)? "FAULT TEXT": "") :  
((row3.TEXT==null)? "FAULT TEXT & TET_ID": "FAULTY  
TWEET_ID")
```

checking tweet_id is numeric or not

6) Finally load the data using running the job.

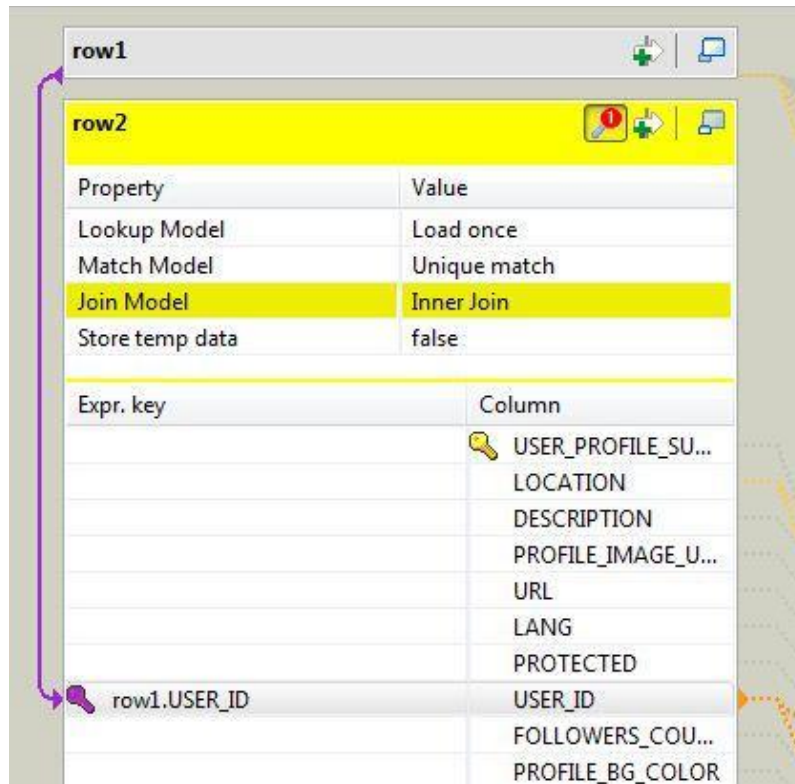
NOW SCD TYPE 2 IMPLEMENTATION



In the above figure

tOracleInput_1 contains the staging data.

tOracleInput_2 contains the data from the dimension table thus lookup from the dimension table must be performed to check whether the record exists or not.



creating the relationship using equi join

*If both data structure does not matches because staging may consist the data in string format for that column then you can achieve this join by converting the data type with in tmap only
 Integer.parseInt(row1.USER_ID) at the join level only.

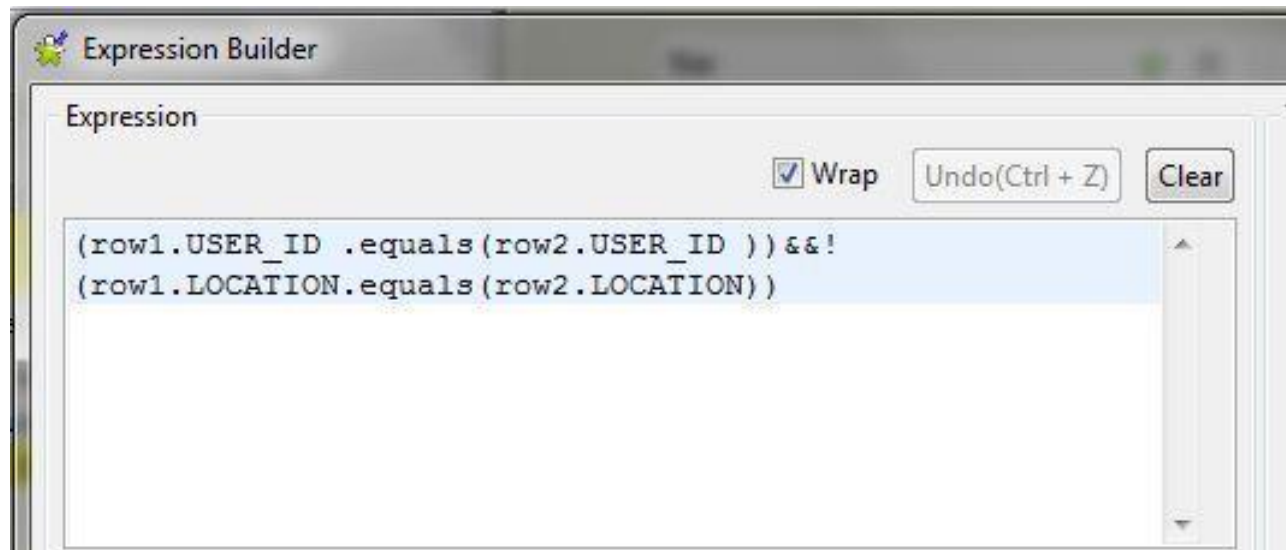
CATCHING FRESH INSERTS:-

At the right hand side you can achieve this functionality by enabling catch lookup inner join reject to true.

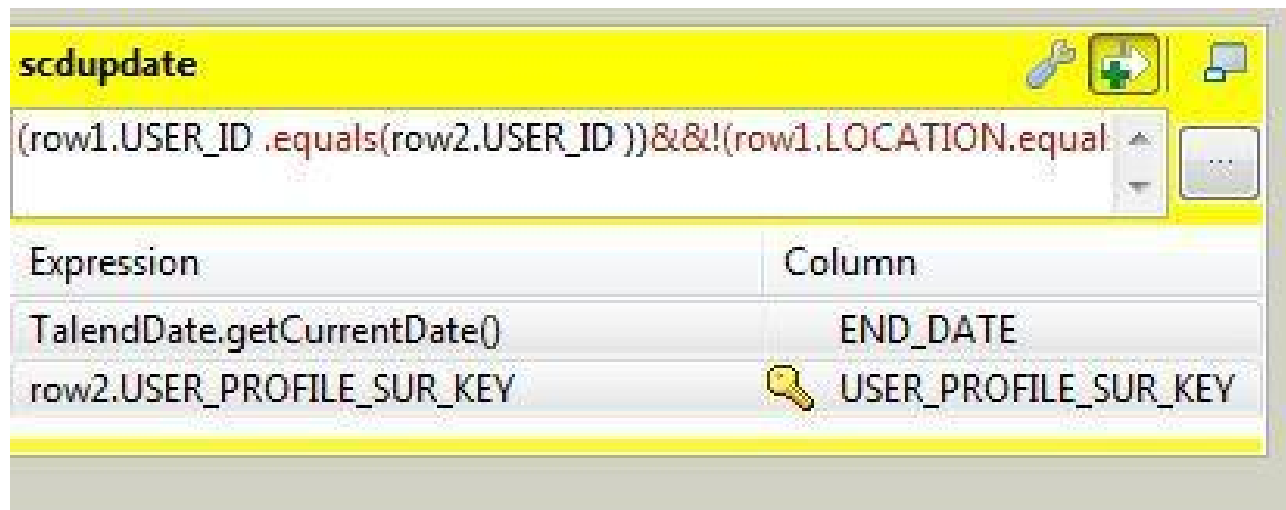
fresh	
Property	Value
Catch output reject	false
Catch lookup inner join reject	true
Schema Type	Built-In
Expression	Column
Integer.valueOf(row1.SKEY)	USER_PROFILE_SUR...
row1.LOCATION	LOCATION
row1.DESCRPTION	DESCRIPTION
row1.PROFILE_IMAGE_URL	PROFILE_IMAGE_URL

IF A ROW ALREADY EXISTS THEN INSERTING A NEW RECORD WITH A FRESH VALUE AND UPDATING THE OLD RECORD WITH END DATE

scdinsert	
(row1.USER_ID .equals(row2.USER_ID))&&!(row1.LOCATION.eq	
Expression	Column
Integer.valueOf(row1.SKEY)	USER_PROFILE_SUR...
row1.LOCATION	LOCATION
row2.DESCRPTION	DESCRIPTION
row2.PROFILE_IMAGE_URL	PROFILE_IMAGE_URL
row2.URL	URL
row2.LANG	LANG



TO UPDATE THE EXISTING RECORD WITH ASSIGNING THE END DATE WITH THE CURRENT DATE



5. Deployment strategies in Talend

Talend provides the flexibility to provide multiple deployment strategies namely:

1) Deployment using script/batch files. 2) Deployment using a Web Service.

Deployment using script/batch files

Deployment Strategies Talend involves generation of **.bat** or **.sh** files that can be executed on their respective OS. When we export a job from Talend, a set of files will be generated which includes class and jar files specific to that job, a **.bat** and **.sh** files are also generated which can be executed for a particular job. The job executed through these files run in silent-mode. One of the main advantages of this strategy is that the client or user is not exposed to the complexity of the job all he needs is java installed on his system and simply run the batch file.

Procedure to export a job: 1) Right –Click on the job that needs to be exported from the Repository Panel.

2) Click on the Export Job option.

3) A dialog box appears, specify the path to which you want to export the job.

4) Choose Export type as Autonomous job and click on okay.

5) The job will be exported and be saved as a ZIP file on the specified path.

Deployment using a Web Service(Executing jobs REMOTELY!!)

This method involves remote execution of the job via a web service. This method is independent of the OS used and can be executed via a web browser. As we know Talend is based on java, the web deployment feature is used extensively.

A WAR file is generated for the specified job, which is used to deploy the job on the application server.

Procedure to export a job: 1) **Right –Click on the job that needs to be exported from the Repository Panel.**

2) **Click on the Export Job option.**

3) **A dialog box appears , specify the path to which you want to export the job.**

4) **Choose Export type as WAR file and click on okay.**

5) **The job will be exported and be saved as a WAR file on the specified path.**

6) **Run Application server ,like Apache Tomcat and add the WAR file to the Tomcat directory as shown in the screenshot below.**

manager | None specified | Tomcat Manager Application | true | 1 | Expire sessions with idle > 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload: persjankitkansal/Desktop/File_Input_0.1.war | Browse...

Deploy

```
<ns1:item>
  --<ns1:item xsi:type="ns1:ArrayOf_xsd_string">
    --<ns1:item xsi:type="xsd:string">
      1,3,"Hirvonen, Mrs. Alexander (Helga E Lindqvist)",female,22,1,1,3101298,32,2875,_S
    --<ns1:item>
      0,3,"Svensson, Mr. Johan Cervia",male,14,0,0,7538,9,225,_S
  --</ns1:item>
--</ns1:item>
```

Deployment StrategiesTalend

OR

you can directly go the webapps folder of your Tomcat Server and simply paste the WAR files over there.

7)Once deployed we can run the job using the following link on the web browser.

http://localhost:8989/File_Input_0.1/services/File_Input?method=runJob

or

[localhost:8989/File_Input_0.1/services/File_Input?runJob\(\)](http://localhost:8989/File_Input_0.1/services/File_Input?runJob())

whereFile_Input refers to the job name. Localhost refers to the server method refers to the execution method.8989 is the application server port no.

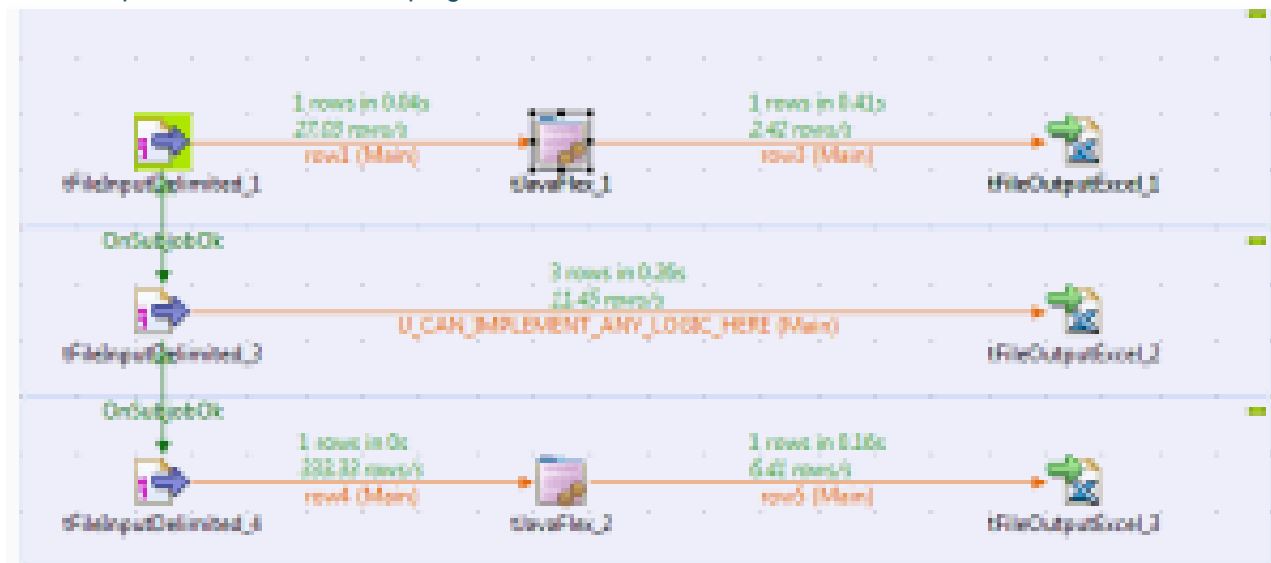
8)On successful execution the following screen is seen.



Deployment StrategiesTalend

6.Custom Header Footer in Talend

Scenario:- Adding custom header and footer using Talend where in Header includes spaces which are generally not allowed and as well as footer contains no of records to be written on to the file. below snapshot shows the overall progress



tFileInputDelimited1_1 is used to take input source file, just select **limit=1** while fetching records and in, tJavaFlex_1 write your own header which you want to populate on the destination file in the main body code section.

let's say in my case i have used

row3.first="First Name";

row3.last="Last Name";

Please be sure that in your tFileOutPutExcel_1 you have to uncheck include header option. This completes your first job.

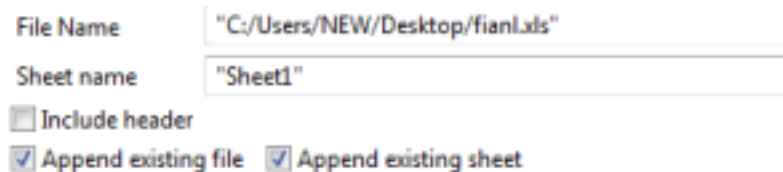
now,

In your second job use your input source file and process the data as per your requirements and further after processing put all your data to the same output file as defined earlier kindly uncheck include header option and check **append existing file** and **append existing sheet** option.

last,

In your third job while fetching data use limit = 1 for records and only select one column in your schema definition and further intJavaFlex component use the following code to print the number of rows at the header.

```
row5.first=((Integer)globalMap.get("tFileInputDelimited_3_NB_LINE")).toString()+" "+"rows";
```



The screenshot shows a configuration window with the following fields and options:

- File Name:** "C:/Users/NEW/Desktop/fian.xls"
- Sheet name:** "Sheet1"
- ☐ Include header
- ☒ Append existing file
- ☒ Append existing sheet

for last two output_files

*file names in all the three jobs must be same with different select options.

7.Data Masking Using Talend

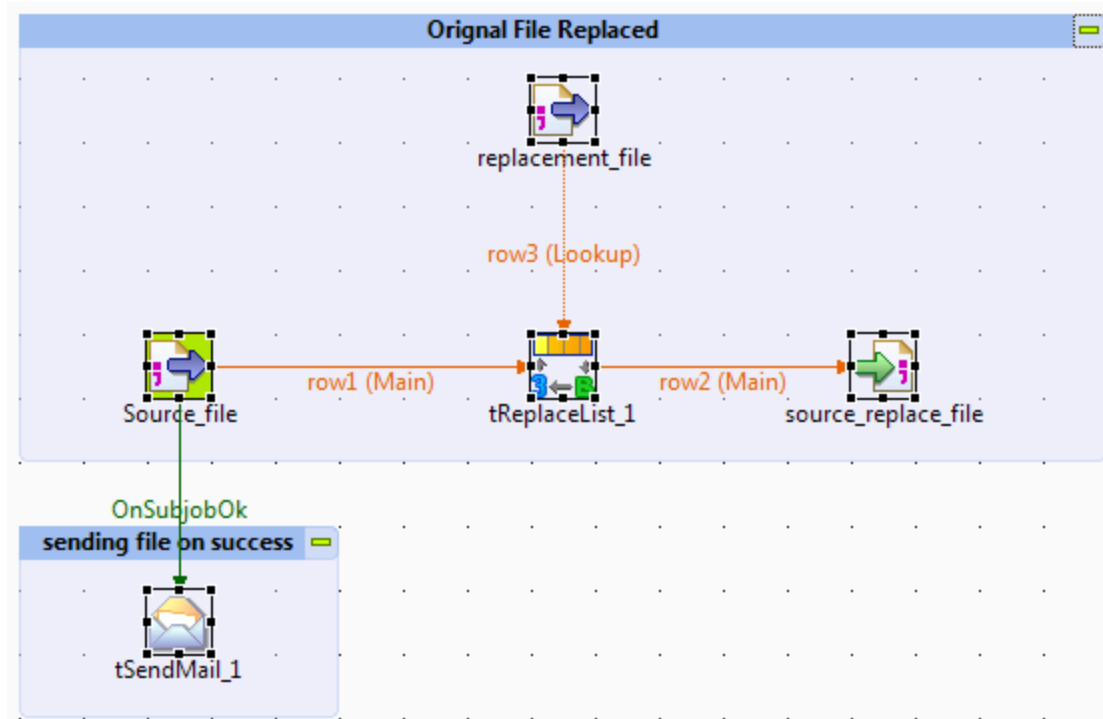
Data Masking-->>Data Masking is a process of encrypting a data field to protect data that is classified as personal identifiable data, personal sensitive data or commercially sensitive data, however the data must remain usable for the purposes of undertaking valid test cycles. It must also look real and appear consistent. production systems generally consists of intensively sensitive data and you can not take chances to make it available for others easily without some changes and for better performance of applications it's also required to have similar kind of data for testing purposes.

If we are talking to current social networking boom if any of user related information become available without any security measures than it may lead to various disastrous results. So, to overcome these problems some encrypting strategies are required which will fulfill the needs.

Now,

Implementing Data Masking Using Talend

Overall Job Descriptive Image:-



JOB DESCRIPTION-->>

A source– file is taken which contains all valid data but which you do not want to give directly after known risks involved.

Replacement– File contains the data which is to be used to replace against the source matching data. Take a tReplaceList component from the palette, and its properties define as:-

*Scheme:

Lookup search column:

Lookup replacement column:

*Column options:

Column	<input type="checkbox"/> Replace	<input type="checkbox"/> Case insensitive
row1.txt	<input type="checkbox"/>	<input type="checkbox"/>
row1.replace	<input type="checkbox"/>	<input type="checkbox"/>

Lookup search column is the column which contains the same data present in your source file column and corresponding Lookup replacement column contains the data which is used to map/replace with the matches found against the source column.

In column option part you have to check the box on which you want to search performed and simultaneously replacement should be done.

e.g.

Your source column contains:-

Text(*Col_name*)

"Hello this is a text which is to be replaced"

replacement file contains

Text|Replacement

Hello|HO

text|msg

is|tis

replaced|wow
which|haha

output generated

HO this tis a msghaha tis to be wow

and finally using tSendMail component the transformed file is sent to the Destination with all the security measures applied.

8.How to use Shared DB Connection in Talend

Today we will discuss how to use or register a shared DB connection, across your jobs. In real time scenarios it's not recommended to create connections again and again so instead of creating new connections you can use your old registered connections through out your sub jobs.

For Demonstration purpose we have created three jobs in which we are using a single connection through out our all jobs.

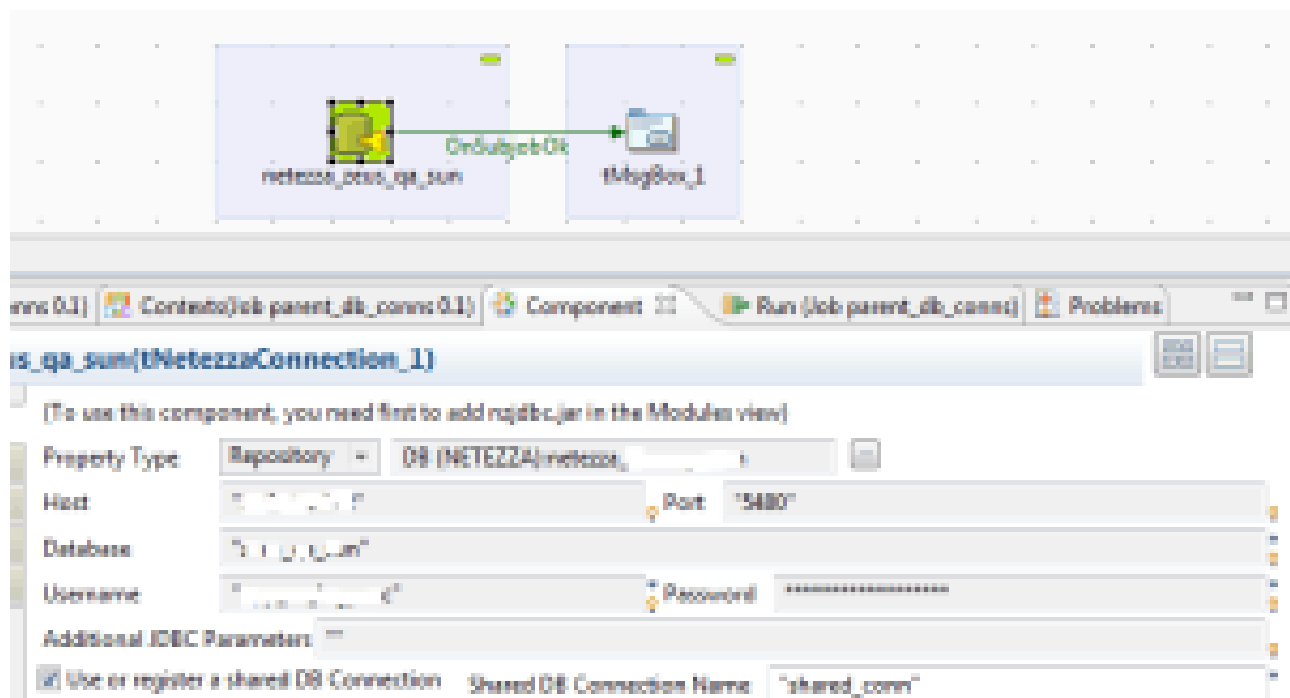
JOBS DESC:-

1. shared_conn_demo--> This job holds our two child jobs (parent,child)
2. parent_db_conns-->The primary job is the one in which connection(shared) is registered for our sub jobs.
3. child_job_conn-->This is the third job which will be using the connection created in parent job.

STEPS

1) parent_db_conns:- Create a connection as you create normally and then at the last component property check **use or register a shared DB connection** and put a name you want to give for shared connection.

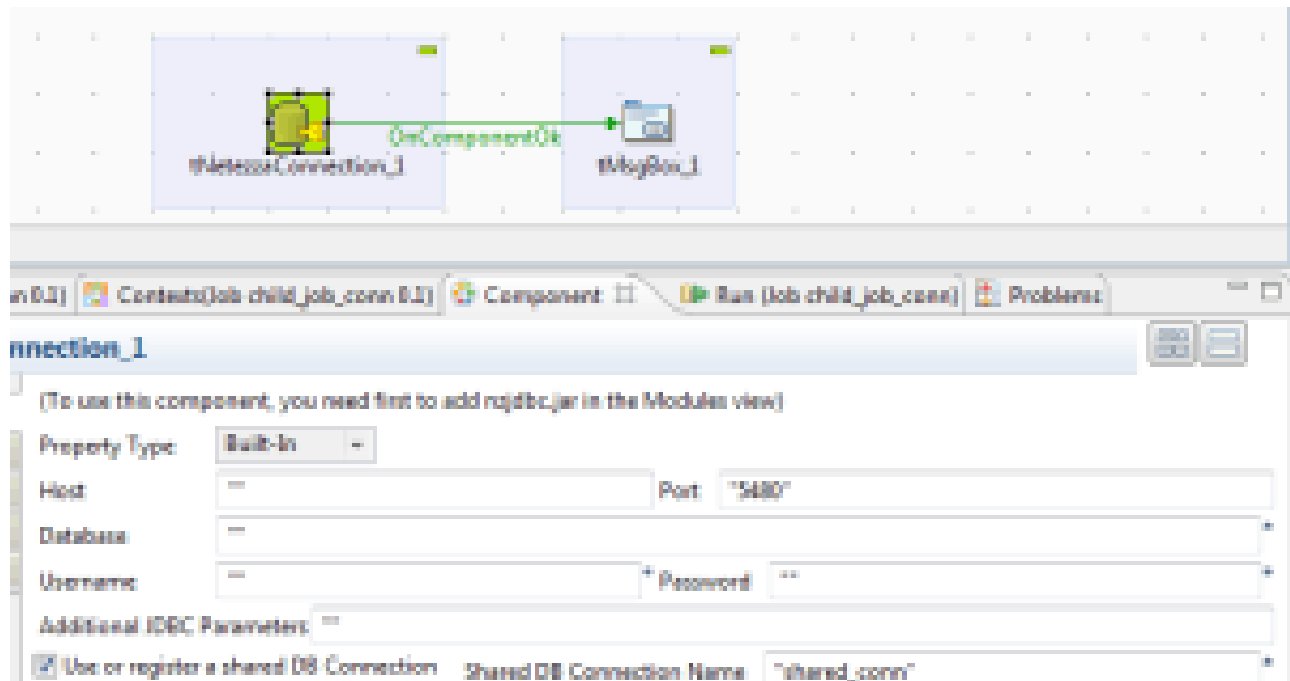
NOTE- The name must be in ""



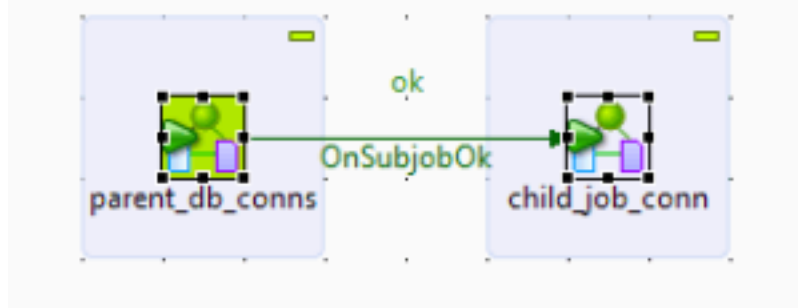
We cannot share our own db_conn values
just fill all the enteries required to create a connection here

2) child_job_conn:- Again check the shared DB component tab and write "shared_conn" or name which you had given in the property box to utilize the connection.

and leave all the field blanks.



2) **shared_conn_demo**:-Now you just have to connect both the jobs and then run them.
Give the trigger as **OnSubjobOk**



That's it now you can use or share your connections through out your all jobs and save connections.

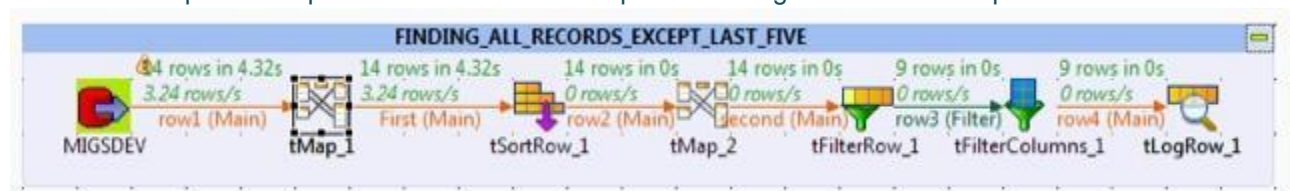
9.Load all rows from source to target except last 5

SCENARIO

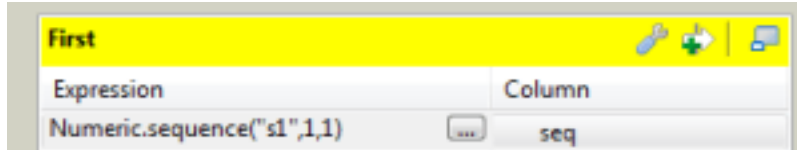
Today we will discuss a situation in which we have to dump all source records to target except the last 5 records.

SOLUTION

We need a couple of components to achieve this requirement as given below in snapshot.



SEQUENCE CREATION



By using above class and method you can generate a sequence in Talend.

LOGIC

Pull all the rows from source and further in tmap create a sequence using **Numeric.sequence** method further sort all the rows using tSortRow in descending order key based upon that sequence generated in tMap, now create another sequence using tmap. Resulting your last rows from source that are currently at top because of sort method will assigned a sequence from 1 and so on.

Finally, use a tFilter and restrict all rows that you want to limit in my case its 5

Just use filter and give condition as

InputColumn	Function	Operator	Value
seq	Empty	Greater than	5



The number you select in tFilter the process will leave those rows to reach to the target.

10.Late Arriving Dimension Using Talend

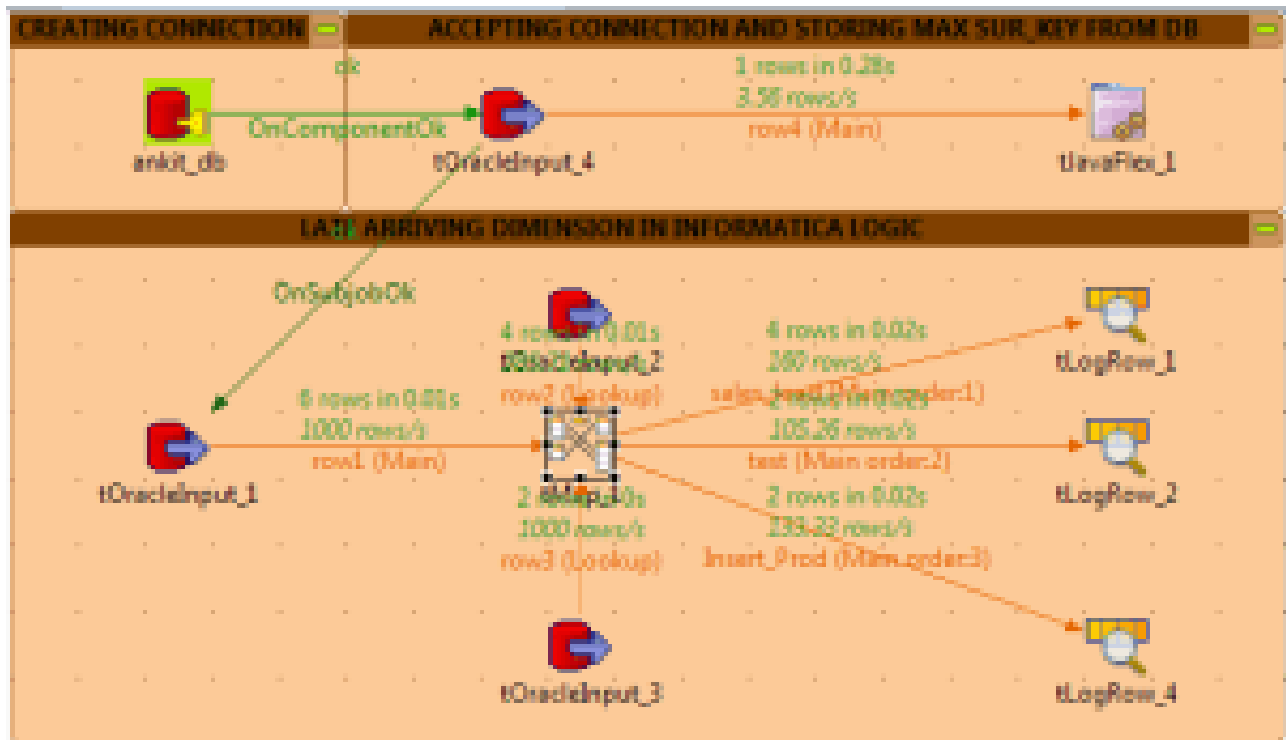
In one of our previous posts we had already covered the concept and implementation of [late arriving dimension using informatica](#).

So, directly we will show you how to achieve this functionality using Talend DI tool.

COMPONENTS REQUIRED

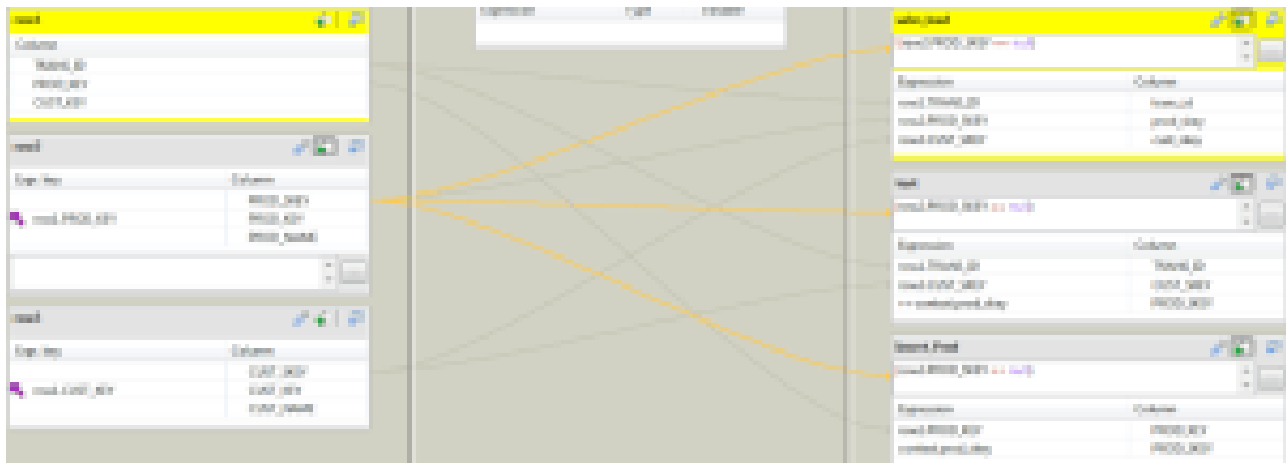
- tOracleInput
- tOracleOutput
- tLogRow
- tJavaFlex
- tMap

Given below is the complete job description:-



JOB DESCRIPTION:-

1. Create a first job that will bring the max surrogate key (using sequel query) from the production dimension table and save in one of context variable using **tJavaFlex** Component. In main code section of tJavaFlex write **context.prod_skey=row4.prod_skey;**
2. Create a second job that runs after successful completion of first job and here bring the tOracleInput_1 component that will bring all the source data that resides into your staging area.
3. Pull two more oracle input sources that will act as a lookup one for prod_dim table and other one is for cust_dim table.
4. tMap Job Desc as described in the below figure



Use Left Outer Joiner Model at the left hand side in both the lookup i.e. for customer and product. THIS JOB IS CREATED TO IMPLEMENT LATE ARRIVING DIMENSION CONCEPT FOR PROD_DIM ONLY.

5. The Top right corner output in tMap is used to populate your normal fact table but make sure to check whether your **prod_dimskey must not be null**.

6. The second insert is used again to update your fact table but with the record that contains no surrogate key as late arriving dimension concept here you just increment your context variable and populate using **++context.prod_skey**.

11.Date Dimension Using Talend



Now,

first column will return you data serially from the current date and second column will give you first date for every quarter.

Most of the implementation is completed using existing components.

12.Dynamic Column Ordering Of Source File Using Talend

Dynamic Column Ordering

FileInputDelimited_1

Dataflow_1

FileOutputExcel_1

First, of all create some context variables which will help you to build up the complete job as given in the screen shot.

first variable count is used for some process which i will tell you later during the post, and produce/create a number of context variables as the number of columns in your source file, i have three columns for demo so i have created three variables as var1,var2,var3.

Remember to bring the data from first row only do not skip the first row.

Column	Key	Type	PK	FK	Date/Fk	Len	Pos	DL	CC
RollID	<input type="checkbox"/>	Int	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
RollID	<input type="checkbox"/>	Int	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
RollID	<input type="checkbox"/>	Int	<input type="checkbox"/>	<input checked="" type="checkbox"/>					

Column	Key	Type	PK	FK	Date/Fk	Len	Pos	DL	CC
moving	<input type="checkbox"/>	Int	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
dropbox	<input type="checkbox"/>	Int	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
valley	<input type="checkbox"/>	Int	<input type="checkbox"/>	<input checked="" type="checkbox"/>					

*****CODE BEGINS*****

```
context.count=context.count+1;
```

```
{
    System.out.println("in first if");
}
```

```
if(input_row.field1.equals("ename"))
```

```
context.var1=1;
```



```

    }
    if(input_row.field2.equals("ename"))
    {
        context.var2=1;
        System.out.println("ename var2");
    }
    if(input_row.field3.equals("ename"))
    {
        context.var3=1;
    }
    if(input_row.field1.equals("deptno"))
    {
        context.var1=2;
    }
    if(input_row.field2.equals("deptno"))
    {
        context.var2=2;
    }
    if(input_row.field3.equals("deptno"))
    {
        context.var3=2;
    }
    if(input_row.field1.equals("salary"))
    {
        context.var1=3;
    }
    if(input_row.field2.equals("salary"))
    {
        context.var2=3;
    }
    if(input_row.field3.equals("salary"))
    {
        context.var3=3;
    }
}
//at this stage your mapping variables already know which column contains which data so use all the
mapping variables and forward the data to the defined schema columns.
if(context.var1.equals(1)){ output_row.ename = input_row.field1;}if(context.var2.equals(1)){
output_row.ename = input_row.field1;}if(context.var3.equals(1)){ output_row.ename =
input_row.field1;}if(context.var1.equals(2)){ output_row.deptno =
input_row.field2;}if(context.var2.equals(2)){ output_row.deptno =
input_row.field2;}if(context.var3.equals(2)){ output_row.deptno =
input_row.field2;}if(context.var1.equals(3)){ output_row.salary =
input_row.field3;}if(context.var2.equals(3)){ output_row.salary =
input_row.field3;}if(context.var3.equals(3)){ output_row.salary = input_row.field3;}
*Take all your source data as a string data type and later on using tConvertTyoe change the schema data
type def.

```

*****CODE ENDS*****

So, this is one of the way you can handle dynamic column ordering of your source file

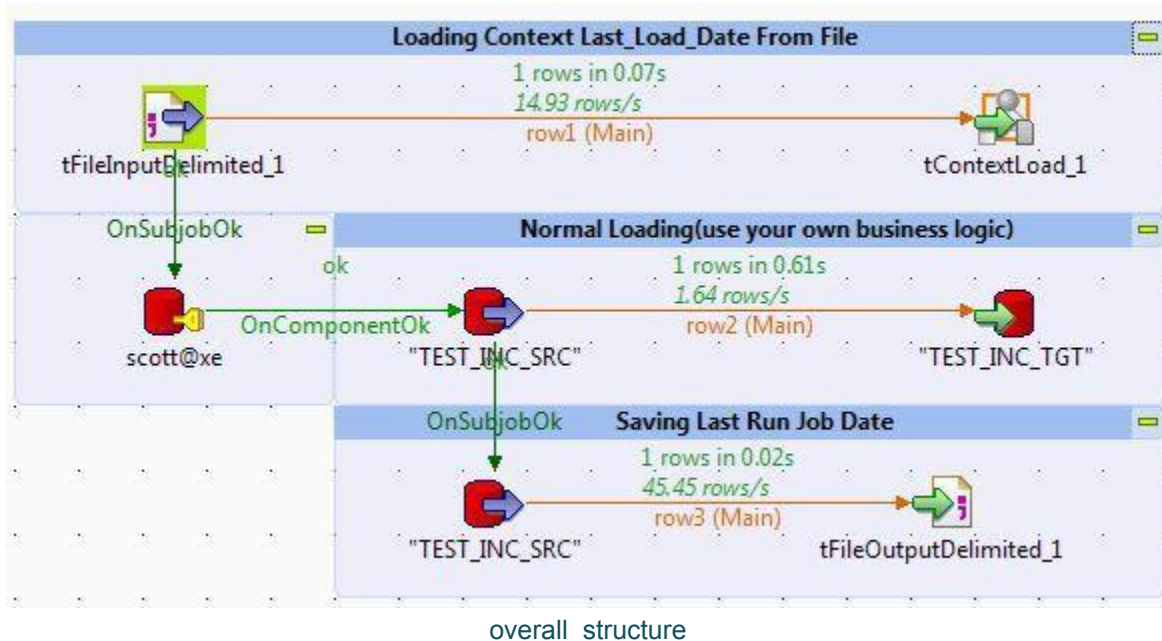
13.Incremental Load Using Talend

Incremental Load is way of loading your target tables in data warehouse such that new or updated data from your source system will only affect your target system.

It's pretty similar with your novel reading, let's say first day you read some pages then on other day you will start reading from the point you left and so on.

So, today we will discuss one of the implementation technique for incremental loading using Talend Open Studio.

Overall Job Desc:-



Execution Steps:

1. Create two context variables one will hold the directory structure of your parameter file(file will store your last run date info) specify some default value which you can further change using property file while deployment and the other variable is used within the job to hold the last run status. Load the last run context variable using tContextLoad.
2. Create a sql query which will hit your source database and pull down data as per your needs. In my case query is with condition `tdate<sysdate and tdate>"+context.Last_Load_Date+"`. As in most of the cases reporting is done till the previous day data, tdate is a column which holds the record's updated/insert date status so we are pulling data as per the given condition.
Total Data Fetched = Data Greater than Last_Time_Run + Data Less than current_date.
3. Simple update your parameter file that is holding your previous run date with `current_date-1`.
"select 'Last_Load_Date;||to_char(sysdate-1,'dd-mon-yyyy') as prev_day from dual"

14. Getting Files From FTP Server

Today, we came up with a scenario which appears in front of most of the developers. Getting Files From FTP Server Using Talend. The provided solution is achieved using Talend Open Studio as a tool.

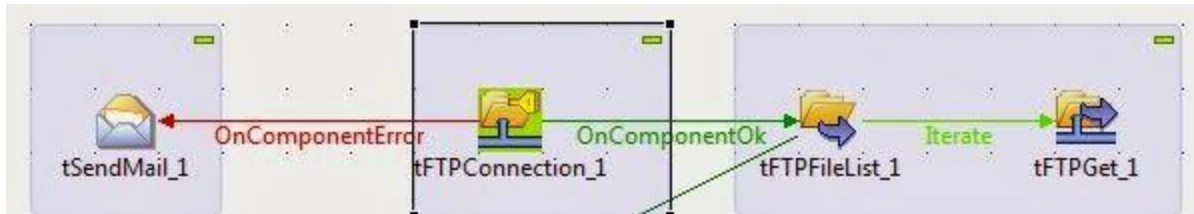
In the given scenario we assume that we don't know the exact names of the files to be downloaded from the remote location and we just have a knowledge of the extensions type and the path of the incoming files.

SOLUTION:-

Take Three Components as

1. tFTPConnection
2. tFTPFileList
3. tFTPGet

Note:- If you know the name of the file then you just need tFTPConnection and tFTPGet to fulfill the requirement just you need to enter the name of the file in tFTPGet component.



Getting Files From FTP Server Using Talend

DESC:-

tFTPConnection creates a connection to your FTP server, you must provide all the necessary entries to create a connection to the FTP server.

For our demo purposes we have use tSendMail component to capture refuse connection exception and send a mail to the service team if the connection can't be made to the server.

Then, in **tFileList** component provide the remote directory address in double quotes "/home/—" and in file list using file mask just specify all the types of files you want to retrieve from the remote server.

1. *.csv
2. *.tsv
3. *.gze.t.c

in different lines by pressing add button.

Connect your **tFTPList** to the **tFTPGet** component using Iterate link. This will provide file names list to the **tFTPGet** component one at a time.

Finally, In **tFTPGet** component fill all the details required such as the local directory path to which you want to copy the files and carefully provide the same remote directory path as provided earlier in the **tFTPList** and finally in the Files option of the **tFTPGet** component write the following **CODE**.

`((String)globalMap.get("tFTPFileList_1_CURRENT_FILE"))`

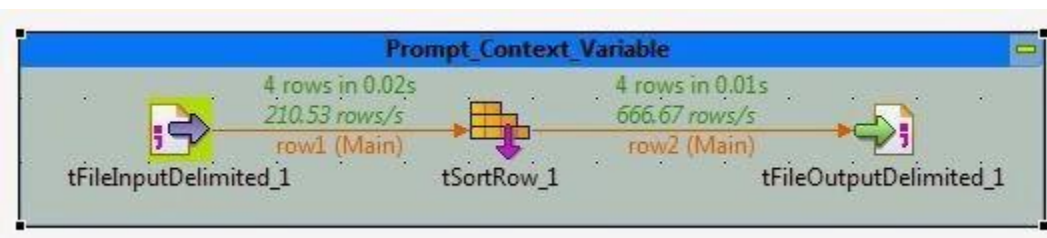
The given job will successfully retrieve all the files which will satisfy your extension condition specified earlier in the **tFileList** component.

15.Initializing Context At Run Time Using Popup

we will discussing a basic functionality for Talend Open Studio, which can be very much useful and can be implemented wide range of scenarios.

Initializing Context At Run Time Using Popup window.

Overall Job Info : -



Initializing Context At Run Time Using Popup

PROBLEM: -

It happens to most of the times that your business logic remains the same but your parameters such as file name, db name, passwords e.t.c. changes frequently, and to make the things workout either you have to make changes in the **Default.properties** or you have to open the Talend Studio and make explicit changes to run the job.

SOLUTION :-

Talend provides a simple solution by providing a **context prompt option** to your variables thus every time you run your job a popup-window will appear and there u just need to simply put your values.

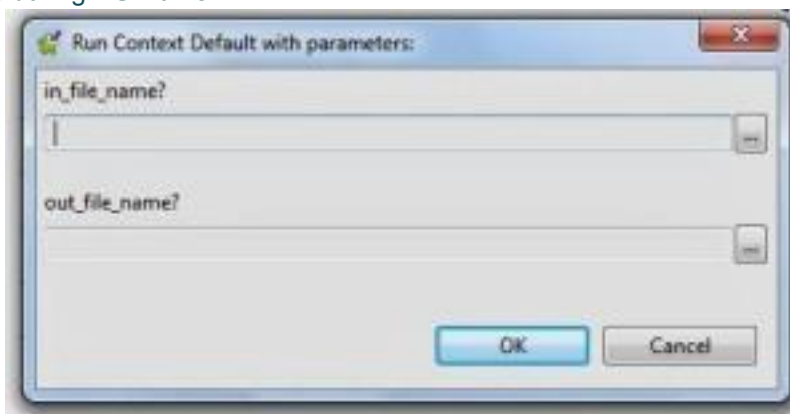
Variables	Values as tree	Values as table	
Variable	Context		Prompt
in_file_name			
	Default	<input checked="" type="checkbox"/>	in_file_name?
out_file_name			
	Default	<input checked="" type="checkbox"/>	out_file_name?

Initializing Context At Run Time Using Popup

Just click on the prompt tab in front of the variable and check the text box for those variable who you want to assign values dynamically..

RUN JOB : -

This is how it looks during **RUN** time.



Initializing Context At Run Time Using Popup

This way you can overcome most of the basic problems occurs during your development and execution....

16.User Define Function In Talend

Talend list of unique features also includes writing your own set of functions, which provides you great flexibility in your work , and one of its feature that is widely used is “User Define Function In Talend“

STEPS:-

- 1) In left panel of the environment right click the routines tab and **create routine** by providing suitable names such as **user_sum**.
(you can write any routine name as per your requirement but it should make sense for better understandability)
- 2) Create a desired function to achieve your desired functionality like **f_sum**.

```

public class user_sum {

    /**
     * helloExample: not return value, only print "hello" + message.
     * {example} helloExample("world") # hello world !.
     */
    public static int f_sum(int first_number,int second_number) {
        return first_number+second_number;
    }
}

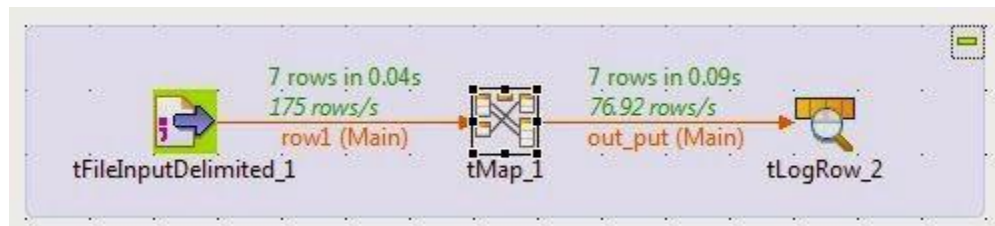
```

User Define Function InTalend

(you can create any function as per your requirement)

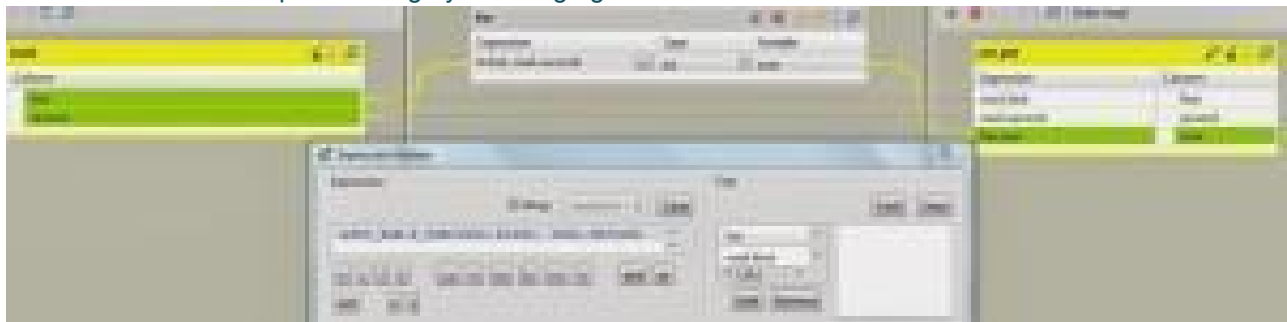
Note : – Use of static keyword while creating function helps in calling a that function without creating a object for that class. If you do not specify static keyword while creating a function then it becomes mandatory for you to call that function by first creating an object for that class.

3)Call to a function



User Define Function InTalend

In this job our function **f_sum** takes two numbers and returns the sum for those two numbers. A call is made to function in tMap describing by the image given below.



User Define Function InTalend

however you can use this function at any place inside the job where talend permits to write your own code or to use methods.

CALLING A FUNCTION : – user_sum.f_sum(row1.first, row1.second)

You can also create/import libraries from outside which contains Classes and functions and after importing those libraries into your job you can use features from those...

So, this how you can create and use “**User Define Function InTalend**” and use them in different situations.

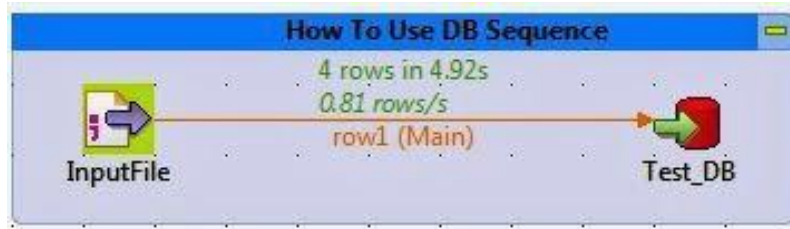
17.Calling DB Sequence From Talend

Today we will discuss how call db sequence from Talend while loading data. For demo purposes we have taken Oracle as our Database.**STEPS:**

1) Create a sequence in Database using : -create sequence test_seq start with 1 increment by 1;

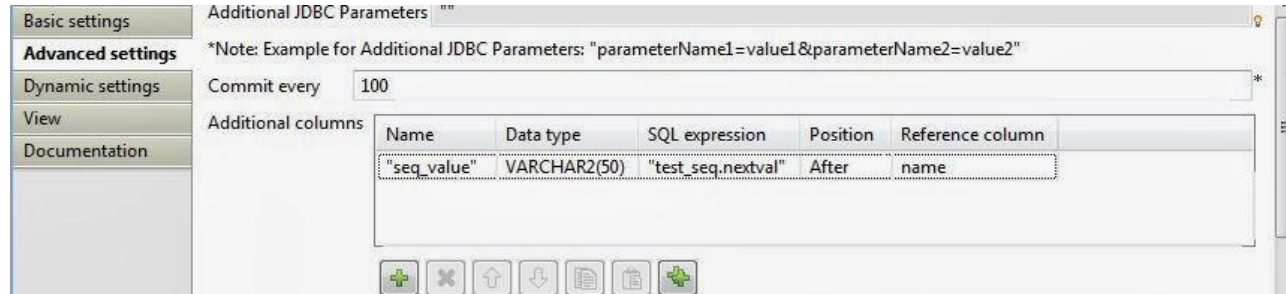
The above statement will create a sequence in Oracle Database. **select test_seq.nextval from dual;**

This statement is very important as it will initialize your sequence. **2)Take a source file/Database and connect it to your oracle output component with all schema defined at Target Level.**



calldb sequence from Talend

3) Go to **Advance Setting Tab** of oracle output **component** and change the settings as defined in the screenshot.



calldb sequence from Talend

sequence_name.nextval is the command which you have to write in SQL Expression. so that for every row this expression will call your Sequence present at Database Level.

(source: <http://www.talendutorials.com/talend-interview-questions>)

1. Difference between tAggregatedRow and tAggregateSortedRow in Talend

tAggregateRow is most useful component in Talend open studio. It works on SQL Group by features. It receives a flow and aggregates it based on one or more columns. It is widely used to perform operation on common SQL aggregate functions like min, max sum etc. After aggregation, tAggregateRow provide output for each line.

Before jumping to TalendtAggregateRow component in Talend, let me explain you aggregate function in SQL.

Aggregate function always returns a single result based on groups of rows. It generally used with SELECT statement in SQL queries. For example if we want to get average of the maximum salary of all department in given table, we can use this SQL query

```
SELECT AVG (MAX (salary)) as AVG_SALARY FROM employees GROUP BY department_id;
```

AVG_SALARY

18929.33

Talend Open Studio component (tAggregatedRow) performs same calculation and return same expected result. Some common aggregate function commonly used in Talend are min, max, avg, sum, first, last, list, count etc.

Talend provide two components for above data aggregation.

- **tAggregateRow**
- **tAggregateSortedRow**

The differences between tAggregateRow and tAggregateSortedRow are
tAggregateRow

- It can't sort the value based on any key field, simply return aggregated value

- All standard SQL aggregate function applied in this component

- Sorting of data either ascending or descending could be achieved by tSortRow

tAggregateSortedRow

- This component sort the data based on key field along with aggregation

- It also gives facility to provide "Input rows count" where we can pass number that represent how many row we want to aggregate.

Below screenshot explain how to use tAggregateRow in Talend

Here is the list of records available in my input file. Based on the first column "Grappe", I need to aggregate data from this list and try to find

- List of all "Name of BTS" for a particular group.
- Total number of "degree of BTS" in a particular group.

I have used three components (tFileInputDelimited, tAggregateRow&tLogRow) to check how to aggregate data in Talend, combined all three components via Main flow.

You can double click on first component (inputfile in above screen) and set its property and schema. Please click on Edit Schema to add schema of input file.

Now you need to set properties for tAggregateRow, simply double click and set it as per above screenshot.

You need to select "schema" as "Built – In"

For "Group by" you need to provide column name by which you want to grouping.

Here I have applied two aggregate function (count, list) in input file and generated same output column.

Your job is finally ready, after clicking on run button in tool bar, you have above output.

In output screen you can see list of column values and their count, but if you closely have a look into output your data is not sorted.

If you want your result in sorted way, you can use `tSortRow` in above code, or you can also use `tAggregateSortedRow`

Now let me create another sub job with `tAggregateSortedRow` and combine in main job. This component will show you the differences between `tAggregateRow` & `tAggregateSortedRow`.

Follow the instruction according to above screenshot. Double click and locate your input file and add schema by clicking on Edit schema button. Combine all three components via Main link.

Add `tAggregatedSortedRow` from component pane and set its property similar to `tAggregateRow`.

For "Group by" you need to add column from input file based on that you need for grouping.

For "Input row count" you need to set number of columns needs to be sorted. You can use your total number of row value here.

After setting properties for tAggregateSortedRow you need to make it sub job of main above tAggregateRow job, so that the sub job only executed after completion of main job.

Now everything is complete. You need to execute this complete job and compare both the the output.

You will found that the second sub job has sorted value based on "Group by" field for tAggregateSortedRow.

2. How to resume job execution from same location if job get failed in Talend

3. How to execute more than one sub jobs parallel in Talend

Before you jump to tutorials, you must know that Talend is completely developed using Java language. So each and every beauties of java can be directly used inside Talend Open Studio.

Sometime in Talend we need to execute multiple sub jobs in parallel. It means that while executing one job, Talend will start another sub job without interrupting the previous running job. This is something what known as parallel execution in Talend and same concept known as multithreading in Java language.

Before jump to main article let me briefly explain what is multithreading in Java.

Multithreading is a process of executing more than one sub-process simultaneously. It means two different code blocks (known as thread) from your single Java program will share same memory location but execute independently to each other.

As it share common memory location so switching between different Java code block takes lesser time. As well as we save memory for other resource. The idea behind of multithreading is to provide parallel execution of program

Life cycle of a thread (Sub block)

According to Sun Microsystems, thread can have 4 states. But for better understanding I am explaining you in 5 states.

1. **New:** This is the case where you just created instance of thread class but it's not started.
2. **Runnable:** This is the state where a thread or sub program is executing in Java Virtual Machine (JVM), but the thread scheduler has not selected it to be the running state. You can simply say that it is ready to run.
3. **Running:** This is the state where thread scheduler has selected it for running state.
4. **Blocked:** This is the state where thread is still alive but currently not eligible to run.
5. **Time waiting:** This is the state where a thread is waiting to execute another thread in specified time.
6. **Terminated:** A thread that has exited from memory.

The above theory I just explained you only for your knowledge that how Talend work in multithreading environment and how it handle multiple sub jobs in their execution.

But you should not worry about above technical details. You can simply follow screenshot to enable multithreading or parallel execution of sub jobs in Talend Open Studio.

Note: Make sure that your all sub jobs should be independent to each other, so that it will not affect to other part of your job.

I simply used three components tRowGenerator, tLogRow, tJavaRow and connected via main link.

tRowGenerator – basically used to generate random row for testing purpose. I said it to generate 5 dummy rows with three columns each row for my first job.

Double click on tRowGenerator and click on (+) button to add column name in schema and you can also assign its value under "function". You can set parameters for function below the schema tab. Have a look into the screenshot

Once you define your columns name and its length and other parameters, make sure that you have entered value for "Number of Rows for RowGenerator". This is the number which will decide to generate numbers of row. For above example, I am generating only 5 rows. Click on Ok to complete it.

Map it with tLogRow to display all generated record on the screen. After that connect it with tJavaRow to display custom message. I simply wrote *System.out.println("Executing 1st Job");*
Here is the screenshot.

Repeat same sub job block for three times to make three independent sub jobs.

Make sure that for every tJavaRow you should write different message

For example I have written below 3 different messages for tJavaRow

tJavaRow_1 *System.out.println("Executing 1st Job");*

tJavaRow_3 *System.out.println("Executing 2nd Job");*

tJavaRow_4 *System.out.println("Executing 3rd Job");*

Finally my one main job with three different sub-jobs is ready to execute.

Now click execute this job and you will see the result, its execute in sequential (first job then second job then third job).

Now let me enable the parallel execution for all three sub jobs.

Go to **right bottom corner** of Talend Open Studio and **click on Parallel job execution** button

Once you clicked on this button it open Parallel Job Execution window as below

Click on **"Extra"** and check **Multithread execution**

Finally threading is applied in your job. This exercise will execute all three jobs independently based on java multithreading concept.

Now save your job and click run to execute it.

Here is your output after execution

If you not able to see **Parallel job execution** button at the bottom right corner in your Talend Open Studio, go to **Window** in menu bar then click on **show view** then click on **Talend** then select **Job**. After doing this exercise you will be able to show button on bottom right corner.

4. How to iterate filename and directories in Talend

File operation is very important in data integration. In this tutorial I am going to explain how to get list of files and folders name from a directory or a sub directory.

tFileList component in Talend Open studio is used for listing of all files and directories. You can iterate it and get list of all files, directory from current folder as well as sub directory.

As usual, I am going to read configuration setting from an xml file and store into a global variable. I shall use it to connect to the desired folder.

Here is content written in my config file "Config.xml" and store in my local drive.

```
<ServerConfiguration>
  <!-- Input file configuration -->

  <arg name="InputFilePath">C:/Config_Files/input_files</arg>
  <arg name="LogPath">C:/logs</arg>

</ServerConfiguration>
```

I have connected tPrejob, tFileInputXML, tSetGlobalVar component according to below screenshot.

tPrejob : It's always a good practice to start our job from tPrejob. Once tPrejob component is ok I have connected to tFileInputXML.

tFileInputXML : It is an input component and used to map xml file. **Double click** on **tFileInputXML** to set its properties setting

Note: you can **click** on "**Edit schema**" to add columns, available in your config file and map them using xPath query.

tSetGlobalVar : It is used to store global values for your next component. **Double click** on **tSetGlobalVar** and provide details.

Here you just created two global variables. It will widely accessible in your entire Talend application by pre-defined Talend method **globalMap.get()**

You can access these global variables like this

```
(String)globalMap.get("InputPath");  
(String)globalMap.get("LogPath");
```

So you just finished your configuration part of your job. Now let me jump to file iteration part and list me all files & folder name.

File Iteration

We need three components for file iteration.

1. tFileList
2. tIterateToFlow
3. tLogRow

tFileList: It contain list of all files and directories. It iterate on a set of files from a particular directory. This component comes with lots of useful properties. You can set it as per your needs

tFileList basic settings

Directory (String)globalMap.get("InputPath")

File Type	<ol style="list-style-type: none"> 1. Both (List all files and all folders) 2. Directories (List only for folders) 3. Files (List only files)
Include subdirectories	Check if you want to iterate in child directory.
Files	You can decide which type of file you want to iterate. You can add more file extension by simply clicking on (+) button
Order by	You can decide which file iterate first.
Order action	Either file iterates will asc or desc.
For file operation, Talend gives some predefined constant for tFileList	

Assume that you are using tFileList_1 your component

tFileList_1_CURRENT_FILE : Get current file name
tFileList_1_CURRENT_FILEPATH : Get current filename with complete path
tFileList_1_CURRENT_FILEEXTENSION : Get extension of current file.
tFileList_1_CURRENT_FILEDIRECTORY: Get current file directory.
tFileList_1_NB_FILE : Get number of total files.

tIterateToFlow : It transform an iterate link to a flow input.

When you iterate a directory using tFileList, there could be a chance to found more than one files and more inner directories. So we need looping to get all lists.

tIterateToFlow component convert each iteration into an input flow, so that input flow can be stored some place to get it list.

In this example tIterateToFlow will receive each iteration and convert into a flow and we can write this flow into a file or store into a log.

Double click on **tIterateToFlow** then click on **Edit schema** to add column name.

You can use ((String)globalMap.get("tFileList_1_CURRENT_FILE")) to list of all filename based on your selection for tFileList.

Now you have successfully designed your job, this job will return all csv file, as we have used .csv for a mask for tFileList component above.

Here is your complete job screenshot

Once you finish your above job with complete setting, click run button to execute this job. I am sure you will return list of csv files in log screen

You can also download this example at end of article. If you feel any issue to run this job, don't hesitate to contact me, I am sure I shall try to help you.

5.What is the difference between OnSubjobOK and OnComponentOK in Talend

6. How can you pass a value form parent job to child job in Talend

7.How to call stored procedure and function in Talend Job

8.How to export job and execute outside from Talend Studio

9.How to pass value from outside in Talend

10.Can I define schema of database or tables at run time

11.What is tReplicate in Talend

12.What is tUnite in Talend Open Studio

13.How to optimize talend job to stop outOfMemory runtime error

14.How to optimize Talend Performance

15.How to execute multipule SQL statements with one component in Talend

16.What is tSystem component in Talend

17.Can I execute multiple commands at one time with a

tSystem component

Talend is an award winning open source ETL (Extract, Transform, Load) tool that provides data integration, data management, enterprise application integration services for small projects to enterprise-wide implementations. We from Talend Tutorials provide Talend Training Courses for individual & corporate professionals. We offer free class room Talend training in Delhi. You can attend our weekend Talend ETL training course. We also offer free Talend training materials download, and you can watch Talend video tutorials, download Talend sample jobs and read how to work with Talend Open Studio

Talend is an award winning open source ETL (Extract, Transform, Load) tool that provides data integration, data management, enterprise application integration services for small projects to enterprise-wide implementations. We from Talend Tutorials provide Talend Training Courses for individual & corporate professionals. We offer free class room Talend training in Delhi. You can attend our weekend Talend ETL training course. We also offer free Talend training materials download, and you can watch Talend video tutorials, download Talend sample jobs and read how to work with Talend Open Studio

Talend is an award winning open source ETL (Extract, Transform, Load) tool that provides data integration, data management, enterprise application integration services for small projects to enterprise-wide implementations. We from Talend Tutorials provide Talend Training Courses for individual & corporate professionals. We offer free class room Talend training in Delhi. You can attend our weekend Talend ETL training course. We also offer free Talend training materials download, and you can watch Talend video tutorials, download Talend sample jobs and read how to work with Talend Open Studio

18. What is difference between tMap and tFilterrow in Talend

3. (<http://www.cram.com/flashcards/talend-interview-questions-5197224>)

What is the difference between the ETL and ELT components of Talend Open Studio?

How does one deploy Talend projects?
What are the elements of a Talend project?
What is the most current version of Talend Open Studio?
How do you implement versioning for Talend jobs?
What is the tMap component?
What is the difference between the tMap and tJoin components?
Which *component* is used to sort data?

4. (<http://www.vikramtakkar.com/search/label/Talend>)

1. Talend **workspace** path should not contain any spaces.

Workspace paths to avoid:

c:\Open Project\Talend Open Studio\workspace
d:\My Projects\Talend\workspace

Recommended Workspace Path:

c:\Talend\workspace
d:\OpenProject\Talend\workspace
c:\MyProject\repository

It is always recommended to not to have any space between the Talend workspace path. We tend to encounter few issues if we have these spaces in path. Hence we should always avoid these spaces.

2. Never forget to perform **Null Handling**.

Example #1 - Bad

```
if(myString.length() > 0)
    System.out.println(myString.toUpperCase());
```

Example #2 - Good

```
if(!Relational.ISNULL(myString) && myString.length() > 0)
    System.out.println(myString.toUpperCase());
```

Always perform NULL handling for the field which is going to be used in any kind of expression. Otherwise Talend Job will throw NullPointerException.

3. Create **Repository Metadata** for **DB connections** and **retrieve database table schema for DB tables**.

It allows you to quickly retrieve the schema of database tables and help rapid development. If you will try to create schema for database table one by one it will take long time. [Click here for more details on creating DB connections and retrieving schema.](#)

4. Use **Repository Schema** for **Files/DB and DB connections**.

It allows you to change the schema at one place, without having to change the schema in every job. Also, you don't need to open every job to find out if the changed schema is part of the Job or not. Changing at one place in Repository will allow you to change in every job. Click on links below to know How to create Repository metadata:

[Creating Metadata for Delimited files.](#)

[Creating Metadata for XML files.](#)

[Creating Metadata for Excel files.](#)

5. Create Database connection

using **t<Vendor>Connection** component and use this connection in the Job. Do not make new connection with every component.

As most of the database have maximum connection limit. In your Talend Job if you are using multiple database components then it may fail because of maximum allowed connection issue. [Click here to know, How to share database connection.](#)

6. Always close the connection to database using **t<Vendor>Close** component.



7. Create a **Repository Document** corresponding to every Talend job including revision history.

This will allow you to track the changes done on any Talend Job.

Sample documentation below:

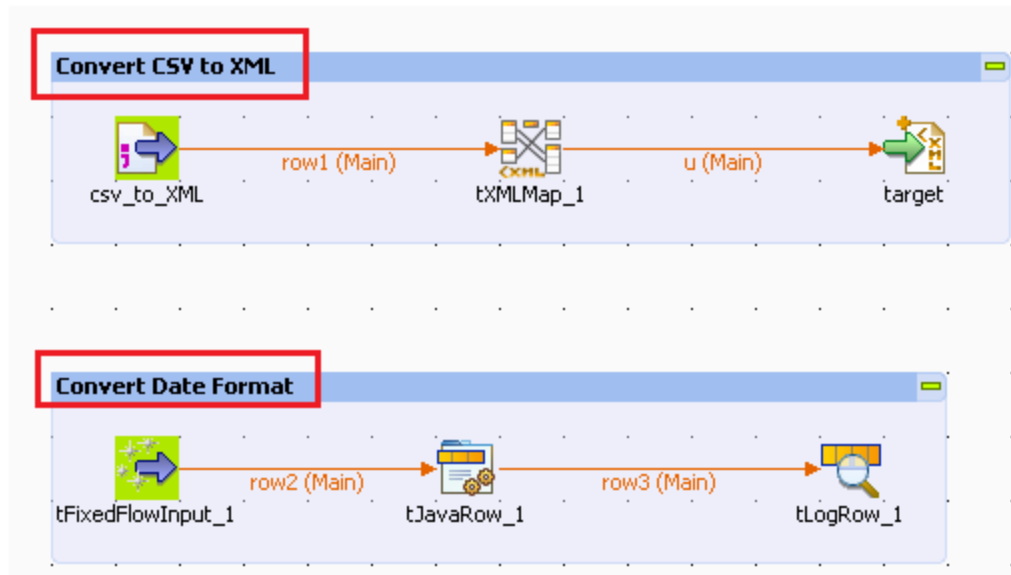
Talend Job Description:

In this paragraph, write down the high level description and functionality of Talend Job

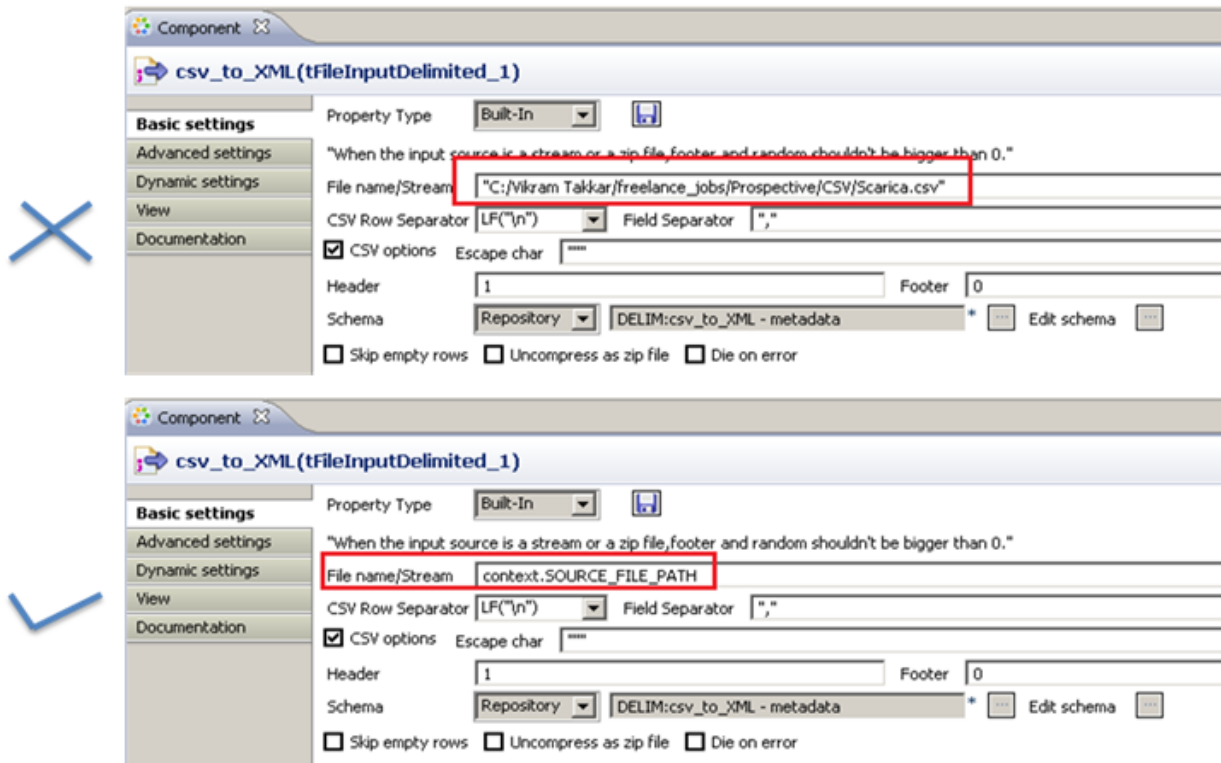
Revision History

1.0	04-10-2014	Initial Development
1.1	07-10-2014	Modification to Source and Target repository Schema
1.2	09-10-2014	Modification to transformation logic.

8. Provide **Sub Job title** for every sub job to describe the sub job purpose/objective.



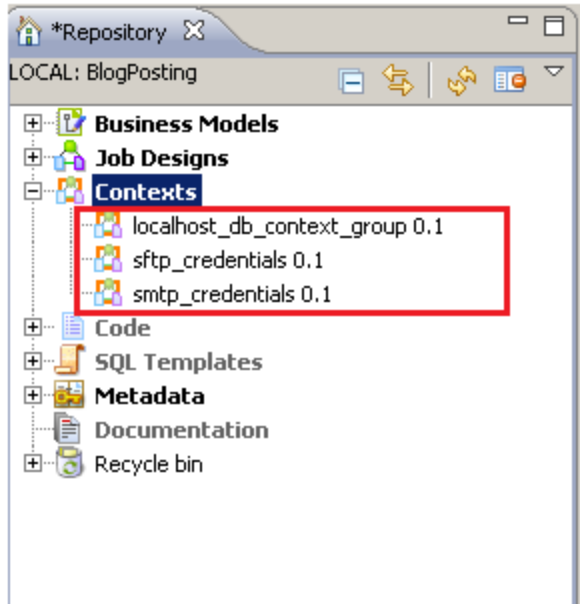
9. Avoid **Hard Coding** in Talend Job component. Instead use Talend context variables.



10. Create *Context Groups* in Repository

Context Group will allow you to use the same context variables in any number of jobs without having to create again and assign value again to them. Imagine your Project requires 20 context variables and there are 10 jobs that require those context variables. Without context groups it will be very difficult to create those context variables again and again in every job.

You can create different context groups for different functionality of variables. For example, you can have different context group for database parameters , SMTP params and SFTP params etc.



Click on links below to know more about context variables and context groups:

1. [Understand Context Variables Part 1](#) (Context Variables, Context groups)
2. [Understand Context Variables Part 2](#) (Define context variables in Repository, which can be made available to multiple jobs)
3. [Understand Context Variables Part 3](#) (Populate the values of context variables from file. tContextLoad)
4. [How to Pass Context Variables to Child Jobs.](#)
5. [How to Pass context Variables/ Parameters through command line.](#)

11. Use *Talend.properties* file to provide the values to context variables using *tContextLoad*.

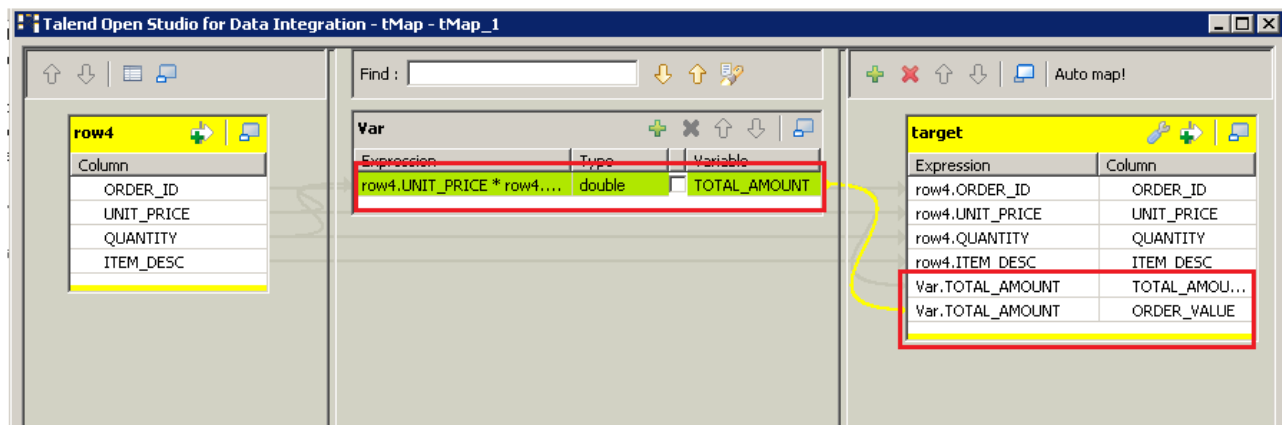
Always provide the value of context variables either through database table or through Talend.properties file. Below is sample of Talend.properties file.

```
SOURCE_DIR_PATH=C:/[redacted]/source
TEMP_DIR_PATH=C:/[redacted]/temp
LOG_DIR_PATH=C:/[redacted]/log
DB_HOST=[redacted]
DB_USERNAME=talenduser
DB_PASSWORD=[redacted]
DB_PORT=1433
DB_NAME=[redacted]
DB_SCHEMA=dbo
DB_ADD_PARAMS=[redacted]
```

[Click hereto understand How to Populate the values of context variables from fileusingtContextLoad component.](#)

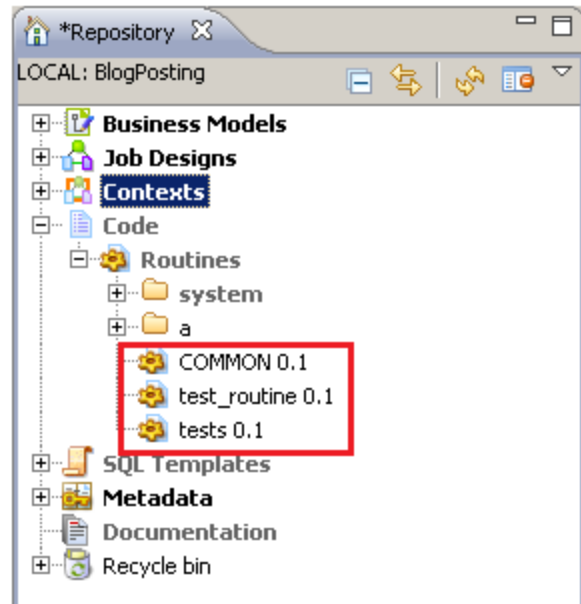
12. Create *Variables* in tMap and use the variables to assign the values to target fields.

For multiple use of single expression or for using the same mapping for multiple target fields, it is always good to create a variable in tMap and assign the value of that variable in target fields. It will allow to only evaluating the expression once for multiple number of times.



13. Create user *routines/functions* for common transformation and validation.

Always create routines/functions for all common transformations and validation rules which can be used in multiple Talend jobs.



[Click hereto know, How to create user routines and functions.](#)

14. Develop Talend job *iteratively*.

Divide the Talend Job to multiple sub jobs for easy maintainability. First create a subjob and then test it and then move to next sub job.

15. Always Exit **Talend open studio** before shutting down the PC.

Talend workspace may get corrupted sometimes, if you shutdown your machine before exiting Talend

Open Studio. So always exit Talend before shutting down PC.

16. Always rename *Main Flows* in Talend Job to meaningful names.

Thanks to [+BalázsGunics](#) for this point. It is always good to rename the main flows in Talend Job to more meaning full names so that when you refer the fields in tMap component or using tFlowIterate it will be easy to refer and understand which data is coming from which flow.

17. Always design Talend jobs by keeping performance in mind.

Talend Job Design - Performance Optimization Tips

I am going to share few of the Performance tuning tips that I follows while designing Talend Job. Let me know your comments on the same and also let me know, if there are any other performance optimization methods you follows and are helpful.

Here we go..

1.Remove Unnecessary fields/columns ASAP using tFilterColumns component.

It is very important to remove the data from the Job flow which is not required as soon as possible. e.g. we have a huge lookup file having more than 20 fields but we only need two fields (Key, Value) while performing the lookup operation. Now if we do not filter the columns before join then the whole file will be read into memory for performing lookup hence occupying unnecessary space. However, if we filter fields and only keep two required columns then the memory occupied by lookup data is much less i.e. in this example 10 times less.

2. Remove Unnecessary data/records ASAP using tFilterRows component.

Similarly, It is necessary to remove the data from the job flow which is not required in the Job. Having less data in your job flow will always allow your Talend Job to perform better.

3. Use Select Query to retrieve data from database - When retrieving data from database, it is recommended to use the select query in the **t<DB>Input** component e.g. tMySQLInput, tOracleInputetc to select only required data. In the select query itself you can provide the required fields to fetch and also provide the where condition and filter only required data. This will allow only required data to be fetched in the job flow rather than complete table unload.

4. Use Database Bulk components - While loading huge datasets to database from Talend Job, it is recommended to use Bulk components provided by Talend for almost all databases. For more details and demonstration of performance optimization using Bulk components click [here](#).

5. Store on Disk Option - There can be several possible reasons for a low performance of Job. Most common reason may include:

- Running a Job which contains a number of buffer components such as tSortRow, tFilterRow, tMap, tAggregateRow, tHashOutput for example
- Running a Job which processes a very large amount of data.

In Jobs that contain buffer components such as tSortRow as well as tMap, you can change the basic configuration to store temporary data on disk rather than in memory.

For example, tMap, select the option Store on disk for lookup data to be stored on a defined path. This will allow not to take the whole data into memory which will keep the memory available for operations and temp data will be fetched from disk.

6. Allocating more memory to the Jobs- If you cannot optimize the Job design, you can at least allocate more memory to the Job. Allocating more memory to job will allow the job to perform better.

-**Xms** signifies the initial heap size of the Job.

-**Xmx** signified the maximum size to which heap can grow. (maximum memory allocated to Job)

7. Parallelism -Most of the time we need to run few jobs/sub jobs in parallel to maximize the performance and reduce overall job execution time. However, Talend doesn't automatically execute the subjobs in Parallel. E.g. If we have a Job which loads two different tables from two different files and there is no dependency between both loads then Talend will not automatically execute the Jobs in parallel. Talend will execute one of the sub job(randomly) and when one is finished then it start execution of the second subjob. You can achieve the parallelization in following two ways:

- Using the **tParallelize** component of Talend. (only available in Talend Integration Suite)
- Running SubJobs in Parallel by using the **Multithreaded** Executions. This option is also available in Talend Open Studio. However, this option is disabled by default. You can enable this option from Job view. Visit the article "[Parallel Execution Sub Jobs in Talend Open Studio](#)" for more details and demonstration of Parallel execution of Sub Jobs in Talend Open Studio.

8. Use Talend ELT Components when required- ETL components are very handy and helps to optimize performance of the job when we need to perform transformation on data within a single database. There are couple of scenarios where we can use ELT components e.g. performing a join between the data in different table in same database. Benefit of using ELT component is that It will not unload the data from database tables into Job flow for performing the transformations. However, it will Talend will automatically create Insert/Select statements which will directly run on DB server. So if the database tables are indexed properly and data is huge then ELT method can provide to be much better option in terms of performance of the Job. For more details on ELT components [click here](#).

9. Use SAX parser over Dom4J whenever required - When parsing Huge XML files try using the SAX parser in the Generation mode in the Advanced Settings of tFileInputXML component. However SAX parser comes with few downsides e.g. we can only basic XPATH expression and can not use expressions like Last , array selection of data [] etc. But if your requirement is getting accomplished using SAX parser, you must prefer it over Dom4J.

Visit the article "[Handling Huge XML files in Talend](#)", for demonstration of performance optimization of SAX parser.

Visit the article "[Difference between Dom4J and SAX parser in Talend](#)", for detailed difference between Dom4J and SAX parser.

10. Index Database Table columns - When updating the data in a Table through Talend Job, it is recommended to index the database table columns on the same fields which is defined as Key in the Talend Database output component. Having the index defined on the key will allow the job to run much faster as compared to non indexed keys.

11. Split Talend Job to smaller Subjobs- Whenever possible, one should split the complex Talend job to smaller Subjobs. Talend operates pipe line parallelism i.e. after processing few records it passes to downstream components even if the previous component has finished processing all records. Hence if we will design a JOB having complex number of operations in single subjob then the performance of the job will reduce. It is advisable to bread the complex Talend job to smaller Subjobs and then control the flow of Job using Triggers in Talend.

Q1). What is the full name of Talend?

Ans: Talend Open Studio

2).What is Talend Open Studio?

Ans: Talend Open Studio for Data Integration is an open source data integration product developed by Talend and designed to combine, convert and update data in various locations across a business.

3).When was Talend Open Studio come into existence/launched?

Ans: launched in October 2006

4).Talend Open Studio written in which computer language?

Ans: Java

5). What is the most current version of Talend Open Studio?

Ans: Talend Open Studio 5.6.0

6). What is the difference between the ETL and ELT?

Ans: ETL: Extract, Transform, and Load (ETL) is a process that involves extracting data from outside sources, transforming it to fit operational needs (sometimes using staging tables), then loading it into the end target database or data warehouse. This approach is reasonable as long as many different databases are involved in your data warehouse landscape. In this scenario you have to transport data from one place to another anyway, so it's a legitimate way to do the transformation work in a separate specialized engine.

ELT:

Extract, Load, Transform (ELT) is a process where data is extracted, then loaded into a staging table in the database, transforming it where it sits in the database and then loading it into the target database or data warehouse.

7). What is the use of tLoqateAddressRow component in Talend?

Ans: This component is use for correct mailing addresses associated with customer data to ensure a single customer view and better delivery for their customer mailings.

8). Can I change the background color of the Job designer?

Ans: Yes. Change the background color of the Job designer by clicking Preferences on the Window menu, followed by Talend, Appearance, Designer, and then Colors.

9). Can you define a schema at run time?

Ans: No, schemas must be defined during design, not run time.

10). Can you define a variable that is accessible from multiple Jobs?

Ans: Yes, you can declare a static variable in a routine, and add the setter/getter methods for this variable in the routine. The variable is then accessible from different Jobs.

11). Can you save my personal settings in the DQ Portal?

Ans: No, you can't.

12). Can you edit generated code directly?

Ans: This is not possible, you cannot directly edit the code generated for a Talend Job.

13). If you want to include your own Java code in a Job, use one of these methods:

Ans:

- Use a tJava, tJavaRow, or tJavaFlex component.
- Create a routine by right-clicking Routines under Code in the Repository and then clicking Create routine

14). Can you use ASCII or Binary Transfer mode in SFTP?

Ans: No. Secure(or SSH) File Transfer Protocol (SFTP) is not FTP. It was defined as an extension to SSH and assumes an underlying secure channel. There is no relationship between FTP and SFTP, so concepts such as "transfer mode" or "current remote directory" that exist in FTP do not exist in SFTP.

For the same reason, there is no transfer option when you select 'SFTP Support' on a tFTPxxx component.

15). Which component is used to sort data?

Ans: tSortRow,tExternalSortRow

16). What is the default pattern of a Date column in Talend?

Ans: By default, the date pattern for a column of type Date in a schema is "dd-MM-yyyy".

17). What is a component?

Ans: Basically a component is a functional piece that performs a single operation. For example, tMySQLInput extracts data from a MySQL table, tFilterRow filters data based on a condition.

Physically, a component is a set of files stored within a folder named after the component name. All native components are located in:

<Talend Studio installation dir>/plugins/org.talend.designer.components.localprovider _/components/ directory.

Each component is a sub-folder under this directory, the folder name is the component name.

Graphically, a component is an icon that you can drag and drop from the Palette to the workspace.

Technically, a component is a snippet of generated Java code that is part of a Job which is a Java class. A Job is made of one or more components or connectors. The job name will be the class name and each component in a job will be translated to a snippet of generated Java code. The Java code will be compiled automatically when you save the job.

18). What is the difference between "Insert or Update" and "Update or Insert"?

Ans: •Insert or Update: First tries to insert a record, but if a record with a matching primary key already exists, instead updates that record.

•Update or Insert: First tries to update a record with a matching primary key, but if none already exists, instead inserts the record.

From a results point of view, there are no differences between the two, nor are there significant performance differences. In general, choose the action that matches what you expect to be more common: Insert or Update if you think there are more inserts than updates, Update or Insert if you think there are more updates than inserts.

19). What is the difference between Built-In and Repository?

Ans: Built-in: all information is stored locally in the Job. You can enter and edit all information manually.

Repository: all information is stored in the repository.

You can import read-only information into the Job from the repository. If you want to modify the information, you must take one of the following actions:

•Convert the information from Repository to Built-in and then edit the built-in information.

•Modify the information in the Repository. Once you have made the changes, you are prompted to update the changes into the Job.

20). Built-In vs Repository, Which is better?

Ans: It depends on the way you use the information is used. Use Built-In for information that you only use once or very rarely. Use Repository for information that you want to use repeatedly in multiple components or Jobs, such as a database connection.

21). What is the difference between OnSubjobOK and OnComponentOK?

Ans: OnSubjobOK and OnComponentOK are trigger links, which can link to another subjob.

The main difference between OnSubjobOK and OnComponentOK lies in the execution order of the linked subjob. With OnSubjobOK, the linked subjob starts only when the previous subjob completely finishes. With OnComponentOK, the linked subjob starts when the previous component finishes.

22). How can you normalize delimited data in Talend Open Studio?

Ans: By using the tNormalize component

23). What is SVN?

Ans: --

24). What is tMap?

Ans: tMap is an advanced component, which integrates itself as plugin to Talend Studio. tMap transforms and routes data from single or multiple sources to single or multiple destinations. It allows you to define the tMap routing and transformation properties.

25). What types of joins are supported by the tMap component?

Ans: Inner, outer, unique, first, and all joins

26). What is tDenormalizeSortedRow?

Ans: tDenormalizeSortedRow combines in a group all input sorted rows. Distinct values of the denormalized sorted row are joined with item separators. tDenormalizeSortedRow helps synthesizing sorted input flow to save memory.

27). Which Talend component is used for data transform using built in .NET classes?

Ans: tDotNETRow helps you facilitate data transform by utilizing custom or built-in .NET classes.

28). What is tJoin?

Ans: tJoin joins two tables by doing an exact match on several columns. It compares columns from the main flow with reference columns from the lookup flow and outputs the main flow data and/or the rejected data.

29). What do you understand by MDM in Talend?

Ans: Master data management, through which an organization builds and manages a single, consistent, accurate view of key enterprise data, has demonstrated substantial business value including improvements to operational efficiency, marketing effectiveness, strategic planning, and regulatory compliance. To date, however, MDM has been the privilege of a relatively small number of large, resource-rich organizations. Thwarted by the prohibitive costs of proprietary MDM software and the great difficulty of building and maintaining an in-house MDM solution, most organizations have had to forego MDM despite its clear value.

30). What's new in v5.6?

Ans: This technical note highlights the important new features and capabilities of version 5.6 of Talend's comprehensive suite of Platform, Enterprise and Open Studio solutions.

31). With version 5.6, Talend:

Ans:

- Extends its big data leadership position enabling firms to move beyond batch processing and into real-time big data by providing technical previews for the Apache Spark, Apache Spark Streaming and Apache Storm frameworks.
- Enhances its support for the Internet of Things (IoT) by introducing support for key IoT protocols (MQTT, AMQP) to gather and collect information from machines, sensors, or other devices.

- Improves Big Data performance: MapReduce executes on average 24% faster in v5.6 than in v5.5, and 53% faster than in v5.4.2, while Big Data profiling performance is typically 20 times faster in v5.6 compared to v5.5.
- Enables faster updates to MDM data models and provides deeper control of data lineage, more visibility and control.
- Offers further enterprise application connectivity and support by continuing to add to its extensive list of over 800 connectors and components with enhanced support for enterprise applications such as SAP BAPI and Tables, Oracle 12 GoldenGate CDC, Microsoft HDInsight, Marketo and Salesforce.com.