

Talend

Introduction

Sreenivas_Ram



6.2



Talend Open Studio

Big Data

Talend Open Studio for Big Data combines big data technologies into a unified open source environment simplifying the loading, extraction, transformation and processing of large and diverse data sets. Users are also presented with a full palette of components for NoSQL connectivity, all under an open source Apache license.

[Learn More](#) →

[Download Now](#)

BPM

Bonita Open Solution from Talend's partner Bonitasoft, is a robust, open source, business process management (BPM) solution for modeling, creating and optimizing business processes.

[Learn More](#) →

[Download Now](#)

Data Integration

Talend Open Studio for Data Integration provides a set of data integration tools to access, transform and integrate data from any business system in real time or batch to meet both operational and analytical data integration needs.

[Learn More](#) →

[Download Now](#)

Data Quality

Talend Open Studio for Data Quality delivers end-to-end profiling and monitoring capabilities with the ability to identify anomalies, find relationships in data, and ensure data quality is fit-for-use within your enterprise application

[Learn More](#) →

[Download Now](#)

ESB

Talend ESB is a reliable and scalable [enterprise service bus](#) (ESB) for the creation, connection, mediation and management of services and their interactions.

[Learn More](#) →

[Download Now](#)

MDM

Talend Open Studio for MDM provides master data management, data stewardship, data integration and data quality in a single platform.

[Learn More](#) →

[Download Now](#)

What Talend Open Studio is?

- An open source graphical development environment
- It comes over 800 pre-built connectors
- It is quick and easy to connect
 - databases, transform files, load data, move, copy and rename files, and
 - connect individual components for complex integration processes.
- It is a code generator
- Is easy to use and reduces the time taken to develop integrations
- Integration jobs are created from components that are configured rather than coded and jobs can be run from within the development environment or executed as standalone scripts.

Use cases

- Some common use cases for Talend Open Studio for Data Integration include:
 - Data migration from one database to another
 - Regular file exchanges between systems
 - Data synchronization
 - ETL (Extract, Transform, and Load)

Tech Mahindra

History of Talend Open Studio

- Talend was founded in 2005 and is an open source software vendor providing solutions for data integration, data quality, master data management, enterprise service bus, and business process management.
- Talend's first product, Talend Open Studio for Data Integration, was launched in 2006, under the name Talend Open Studio, and has since been downloaded over 20 million times.
- Talend has continued to develop its product portfolio and has added complementary tools that provide a single platform for application, data, and process integration.
- The Talend Open Studio brand has since been adopted across the range of Talend's products.

Benefits of Talend Open Studio

- The Studio is open source, free to download and use, with access to the source code, allowing users to extend the product to their particular needs if required.
- The Studio is a great productivity-booster. It's easy to learn and quick to develop with. Even novice developers will be building complex integrations in no time.
- The Studio's pre-built components handle many common and not-so-common tasks. Developers can focus on the end-to-end process, rather than the low-level technical details.
- Talend has an active and open user community. Practical, problem-solving advice is e

Working with Talend Open Studio

Tech Mahindra

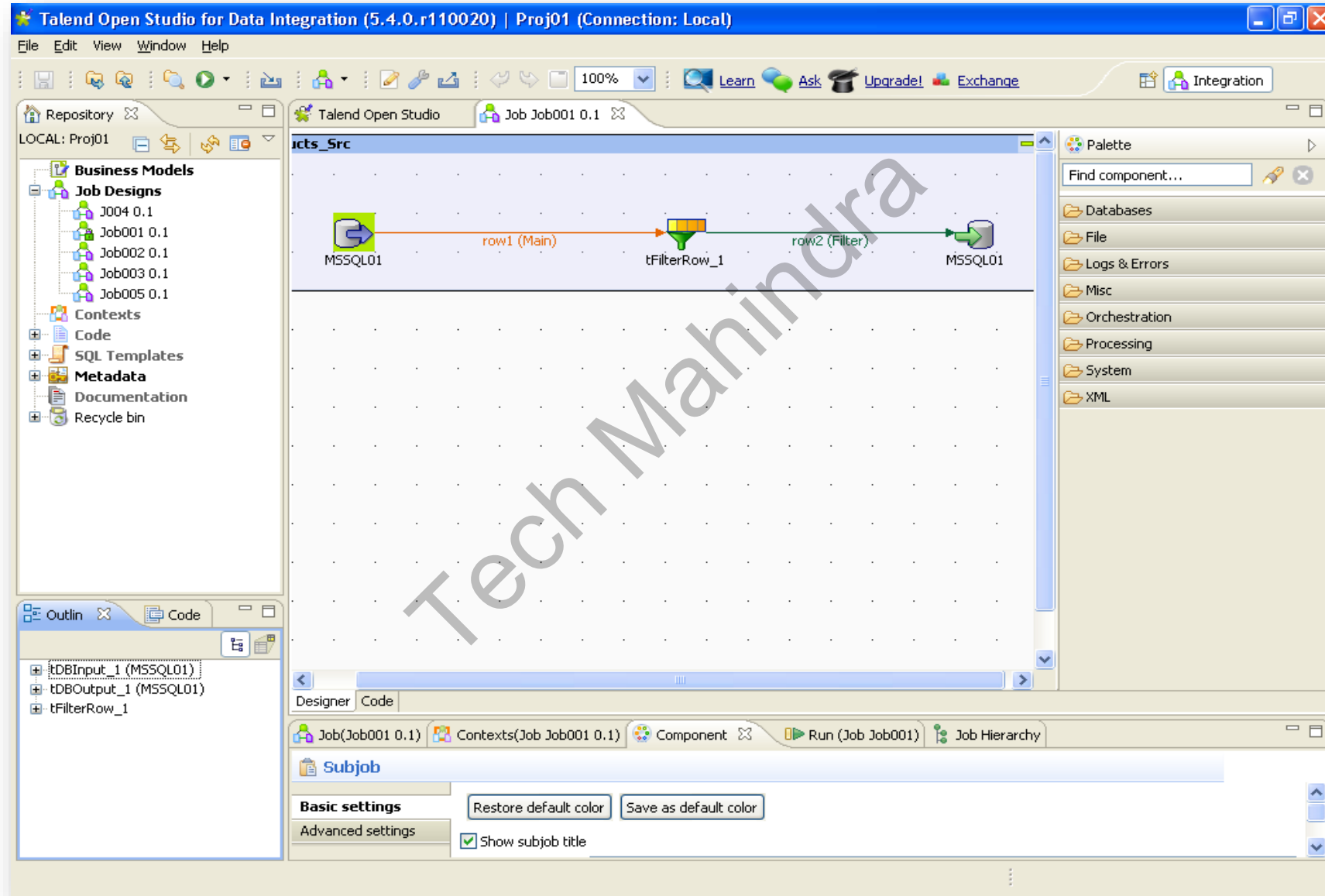
Studio definitions

- Few definitions:
 - A **repository** is the storage location Talend Studio uses to gather data related to all of the technical items that you use either to describe business models or to design Jobs
 - A **workspace** is a directory on your computer that contains one or more projects
 - A **project** is a logical grouping of one or more jobs
 - A **job** is a group of one or more components that, when executed, implement a data flow or integration process
 - A **component** is a preconfigured connector used to perform a specific data integration operation
 - An **item** is the fundamental technical unit in a project. Items are grouped, according to their types, as: Job Design, Business model, Context, Code, Metadata, etc. One item can include other items.

Starting the Studio

1. When you start the Studio for the first time, click on **Accept** license notification to proceed. We can now:
 - Import a demo project
 - Create a new project
 - Change some basic settings
2. We will start by amending some settings. Click on **Advanced**. You will see the following screen:
3. First change the workspace location. Click on the **Change** button and select an appropriate path, such as **C:\Talend\Workspace**.
4. Click on the **Restart** button now.
5. Import the demo project provided with the Studio. Select the default demo project, TALENDDDEMOSJAVA and click on Import.
6. Enter name, such as **DEMOPROJECT**, and click on **Finish**
7. Make sure the demo project is highlighted and click on the **Open** button.
8. Click on **Skip** for now. The Studio will start to load.
9. Once the Studio is open, it runs a **Generation Engine Initialization** process. We can choose to have this run in the background if you wish by clicking on the **Run in Background** button as shown in the following screenshot:
10. Click on the **Start now!** button

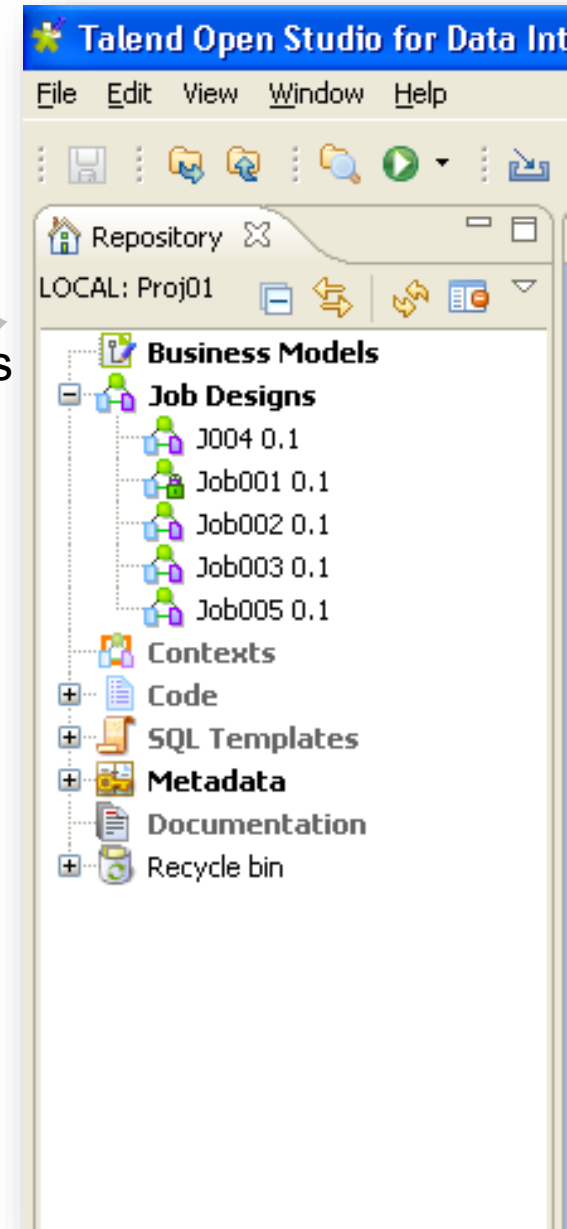
The environment



The environment

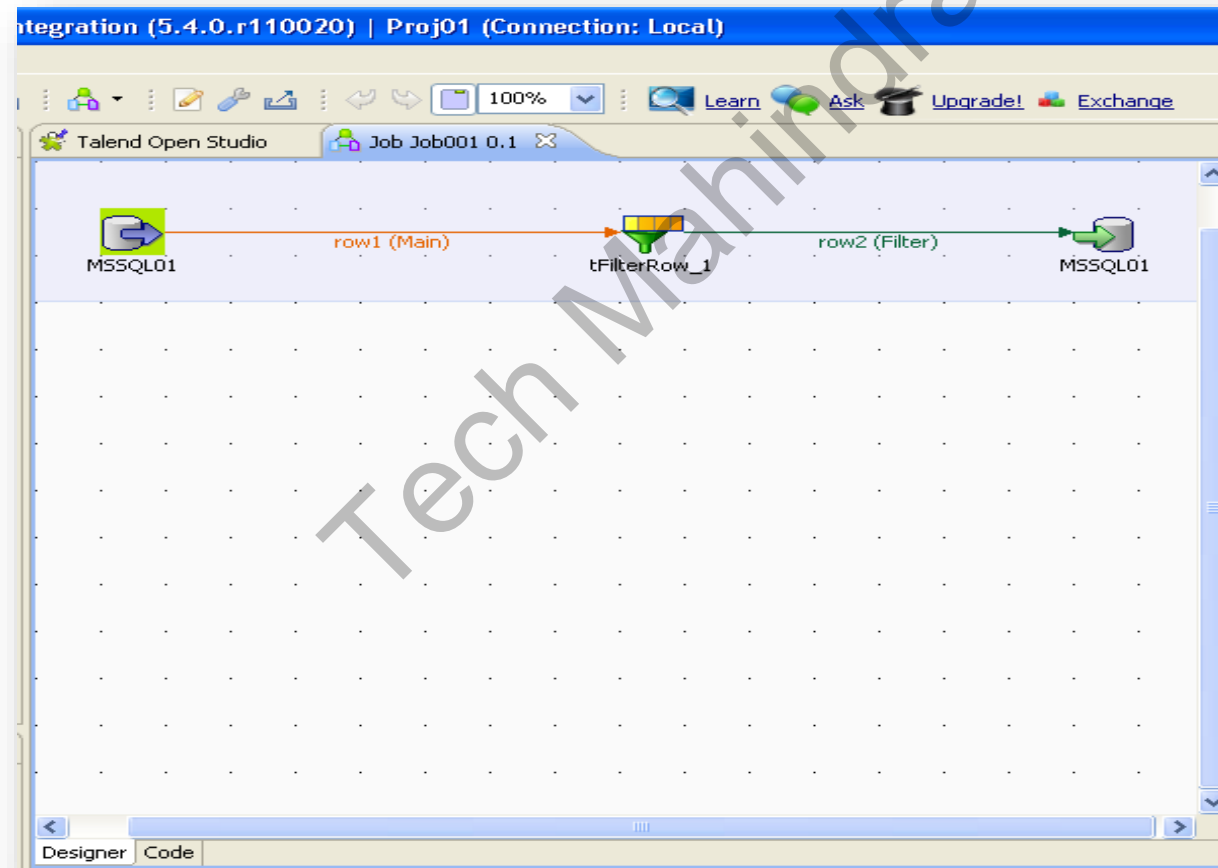
- **Repository Window**

- Job Designs, Business Models, Metadata,
- Database Connection details,
- FTP connection details and File schema definitions
- Reusable code snippets
- Contexts – global or job-specific variables



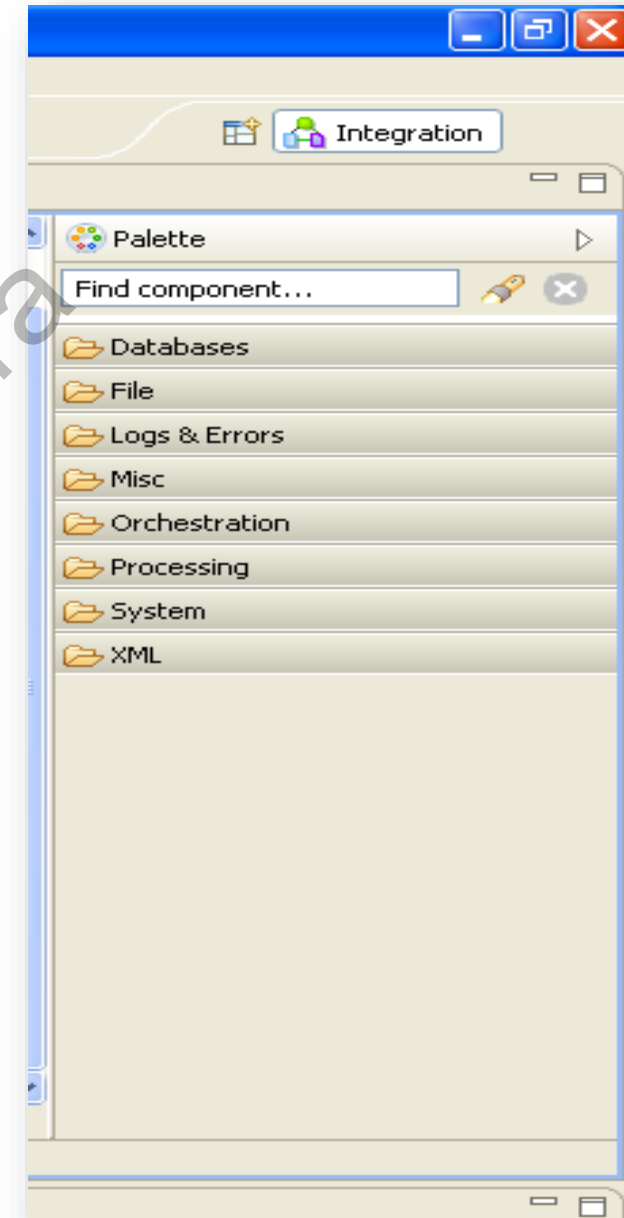
The environment

- **Design Workspace**
 - Developers space for designing jobs and configuring components



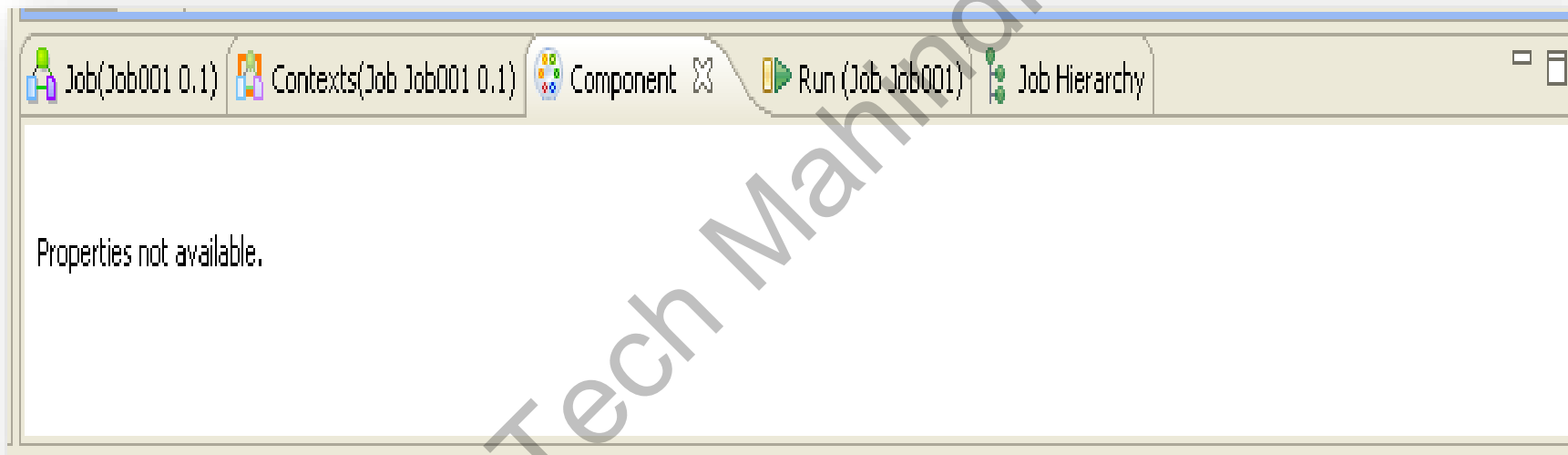
The environment

- **The Palette**
 - Collection of components used in data integration jobs



The environment

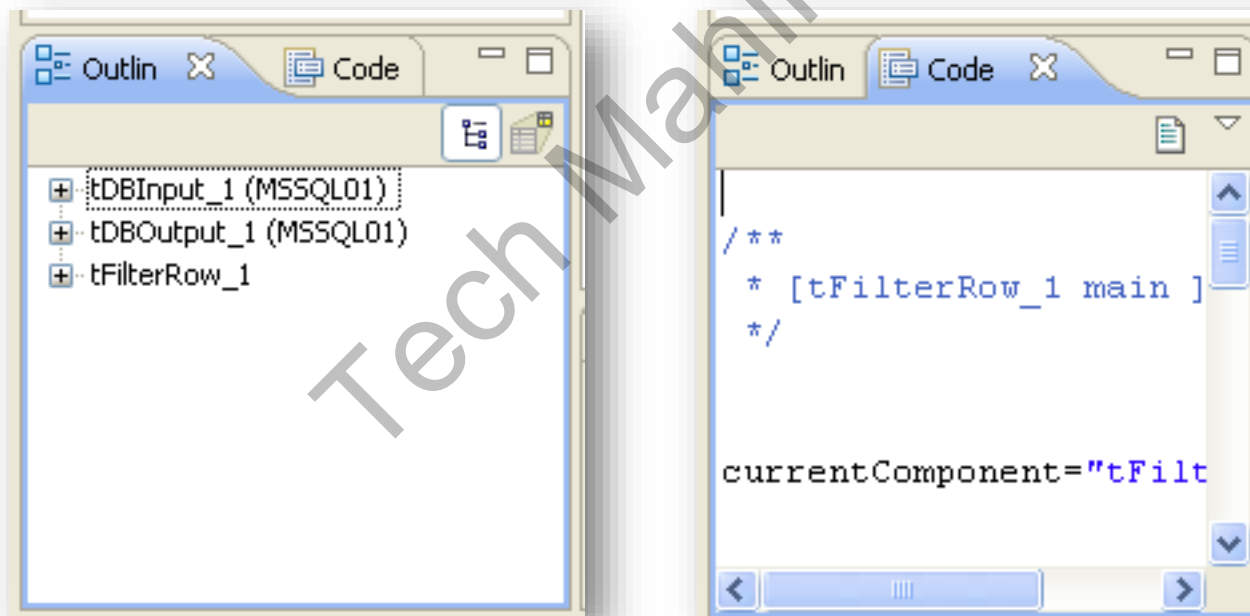
- **Configuration Tabs**
 - Display properties of jobs or specific components



The environment

- **Outline and Code panels**

- Lists the components that have been added to the design workspace
- Gives quick access to standard variables of each component
- Displays code associated with each component



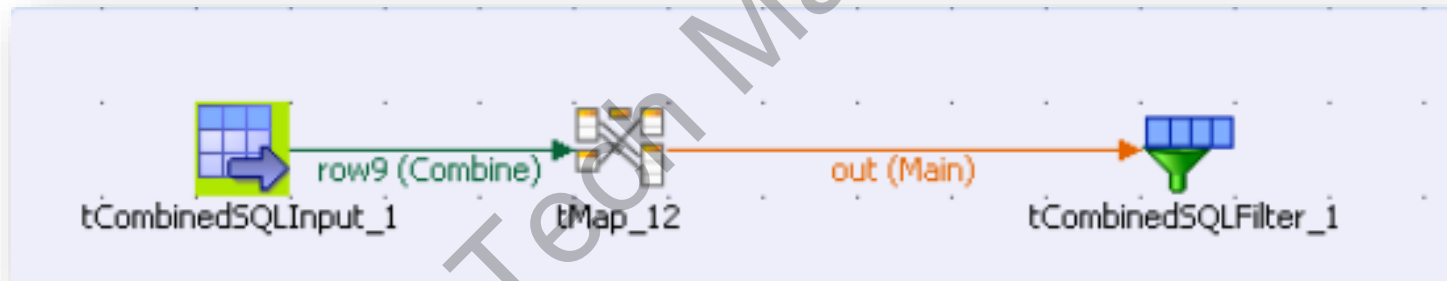
Talend

First Job – Welcome Msg

Sreenivas_Ram

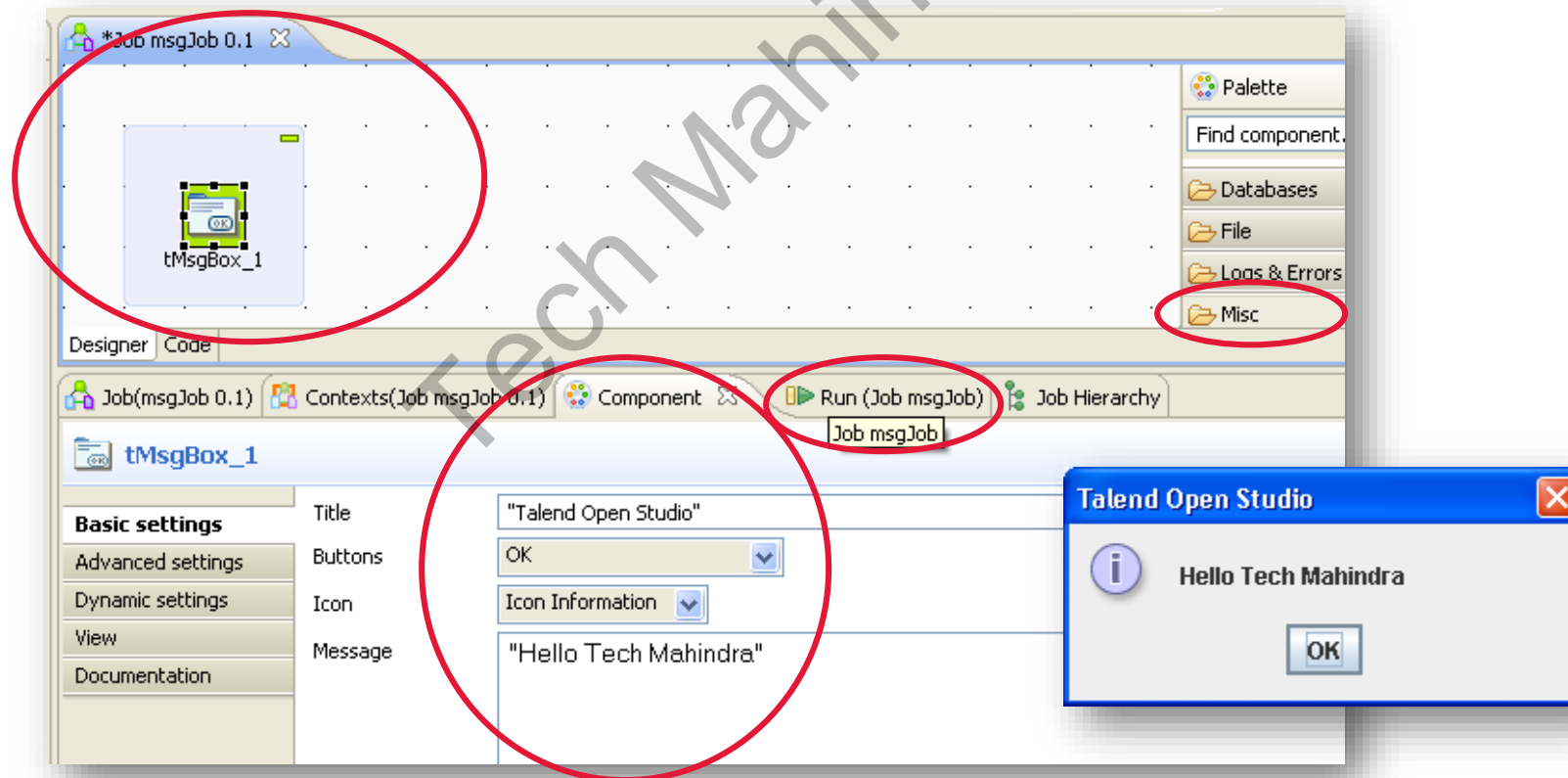
What is a Job design

- A Job Design is the runnable layer of a business model.
- It is a graphical design, of one or more components connected together, that allows you to set up and run dataflow management processes.
- A Job Design translates business needs into code, routines and programs, in other words it technically implements your data flow.



Creating a simple Job

- In Repository, right-click on **Job Designs** and select **Create Job**.
- Enter details of **New Job** and click **Finish**.
- Drag **tMsgBox** component from **Misc** folder of **Palette** window on to the job
- Click the tMsgBox1 component and click **Components** tab, set properties.
- Save the Job and go to Run tab and click **Run** button. [F6 will also run]



Demo

Create a simple job

Tech Mahindra

Talend

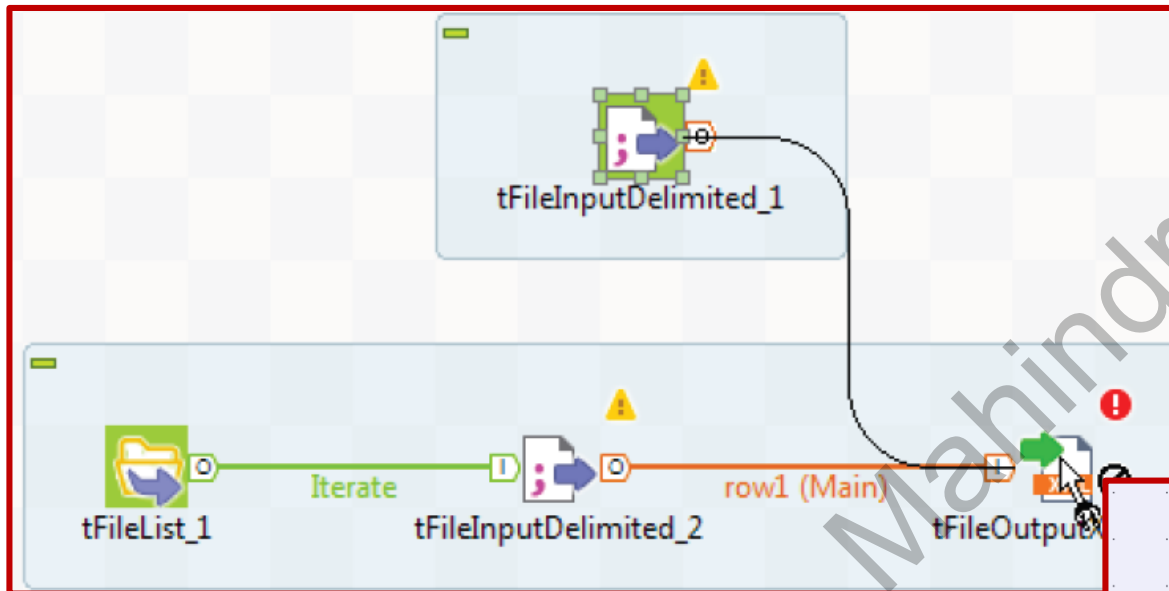
Connections

Sreenivas_Ram

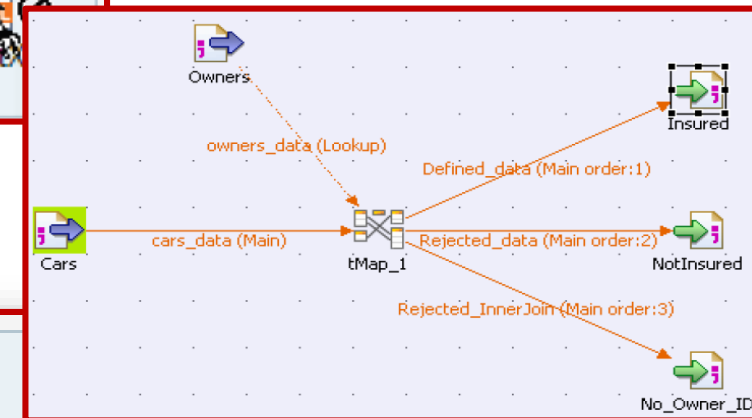
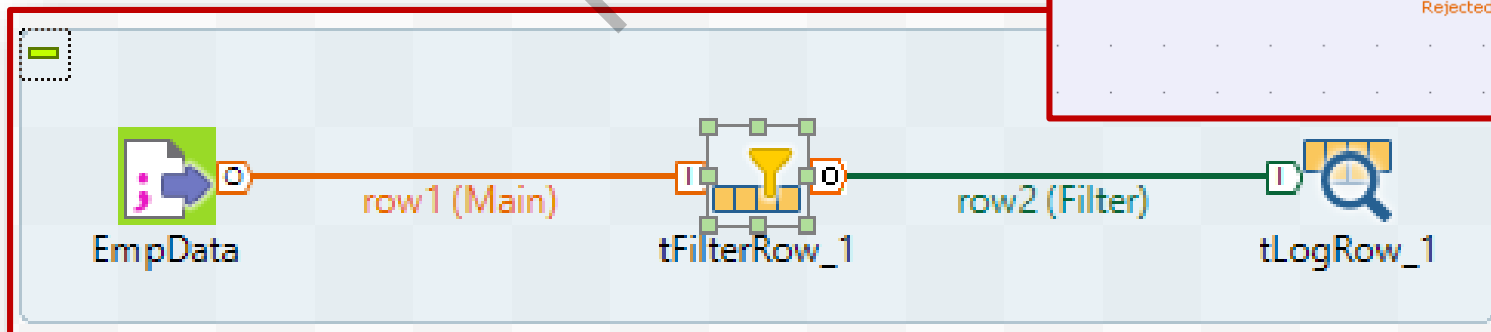
Connections

- A Job or a subjob is composed of a group of components logically linked to one another via connections.
- You need to use the connections to define how the components in use are coordinated.
- **Connection types**
 - **Row connection**
 - A **Row** connection handles the actual data.
 - Main, Lookup,
 - Filter, Rejects,
 - ErrorReject, Output,
 - Unique/Duplicates, Multiple Input/Output
 - **Iterate connection**
 - The Iterate connection can be used to loop on files contained in a directory, on rows contained in a file or on DB entries.
 - **Trigger connections**
 - Trigger connections define the processing sequence, i.e. no data is handled through these connections
 - **Link connection**
 - These links transfer table schema information to the ELT mapper component in order to be used in specific DB query statements.

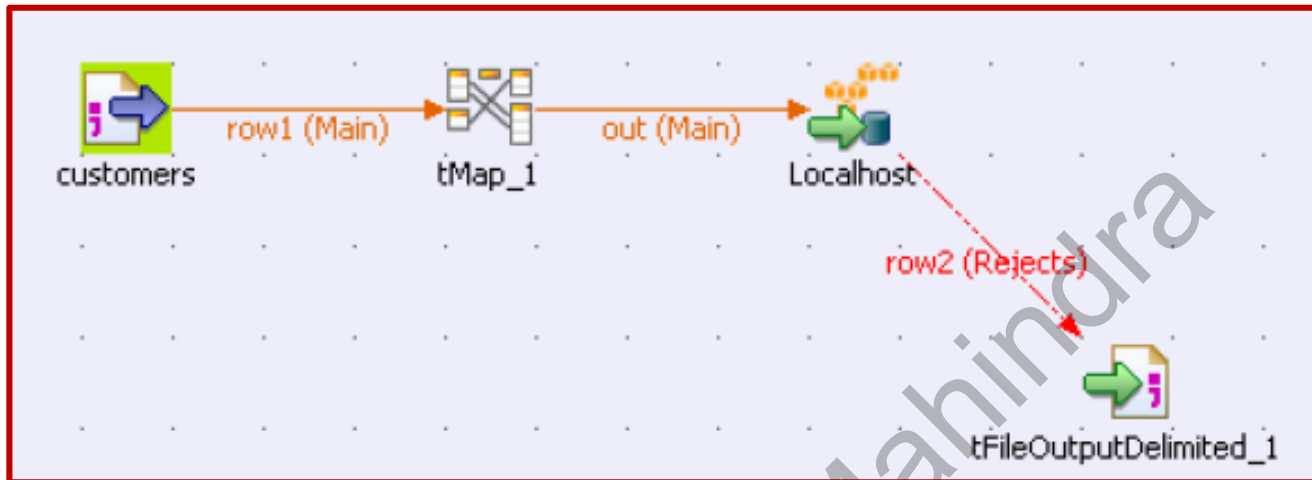
Row Connections



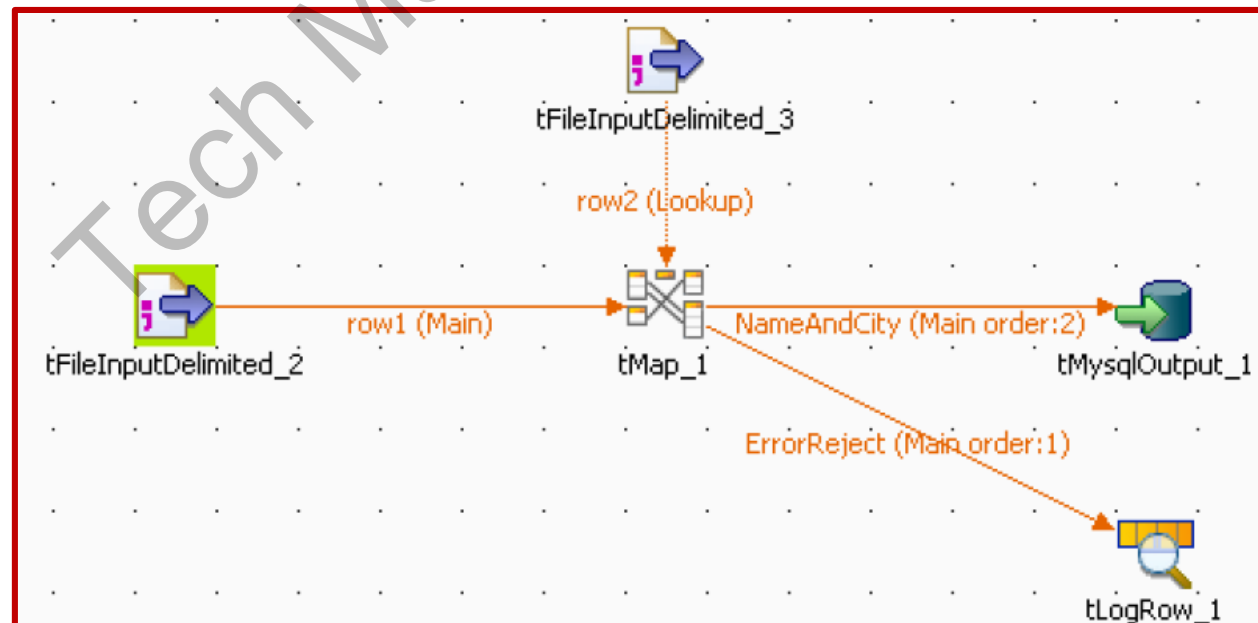
- Row Main
- Iterate
- Multiple Input / Output
- Filter



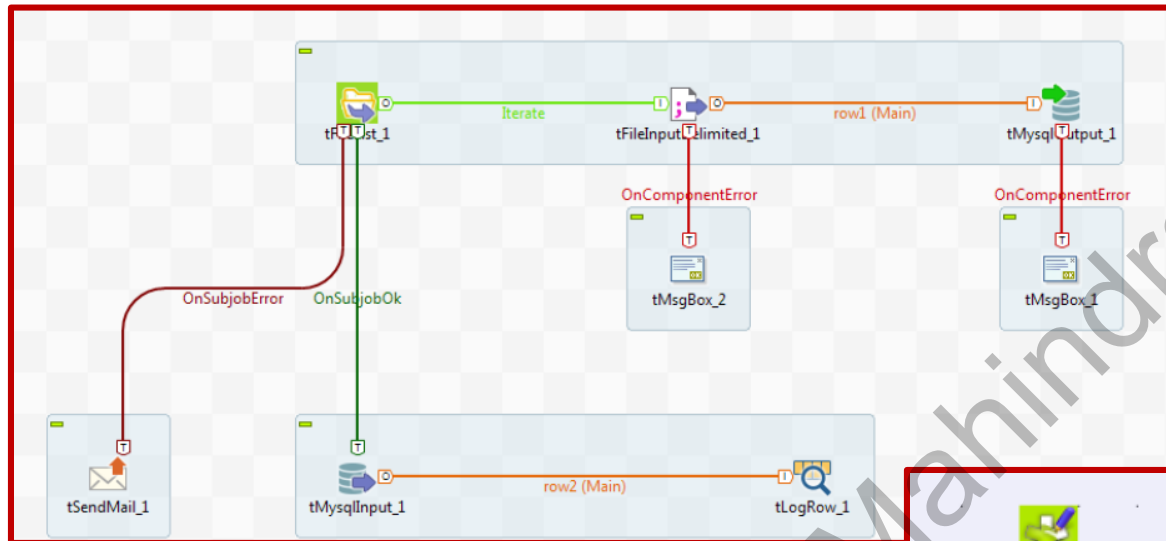
Lookup, Reject, Error Reject Connections



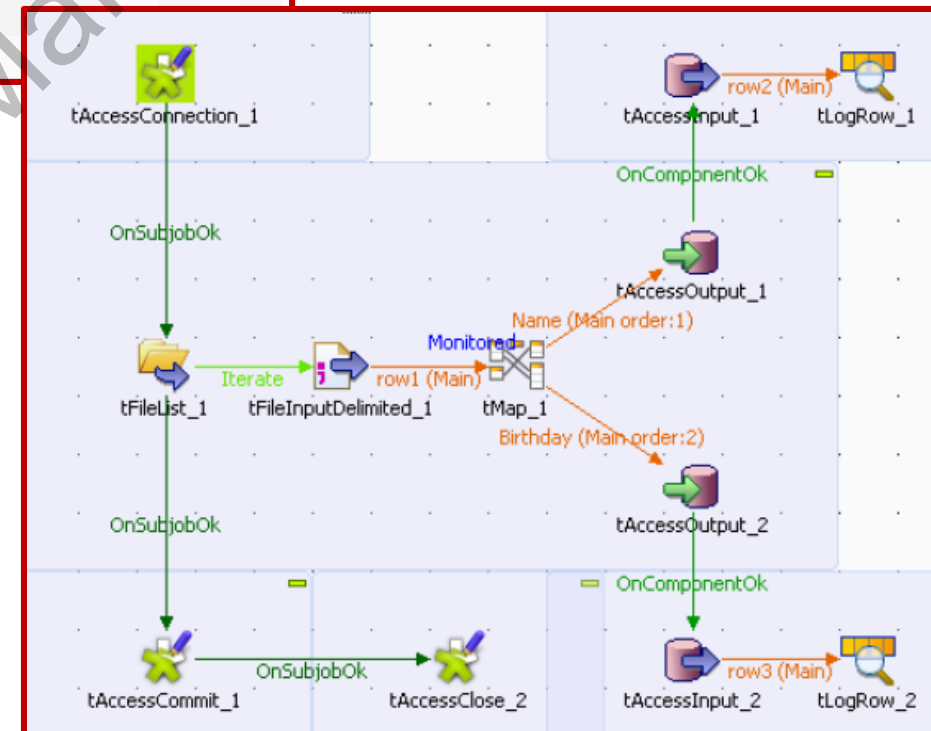
- **Reject**
- **Lookup**
- **Error Reject**



Trigger Connections



- OnSubjobOK
- OnSubjobError
- OnComponentOK
- OnComponentError



Talend

Filter Row

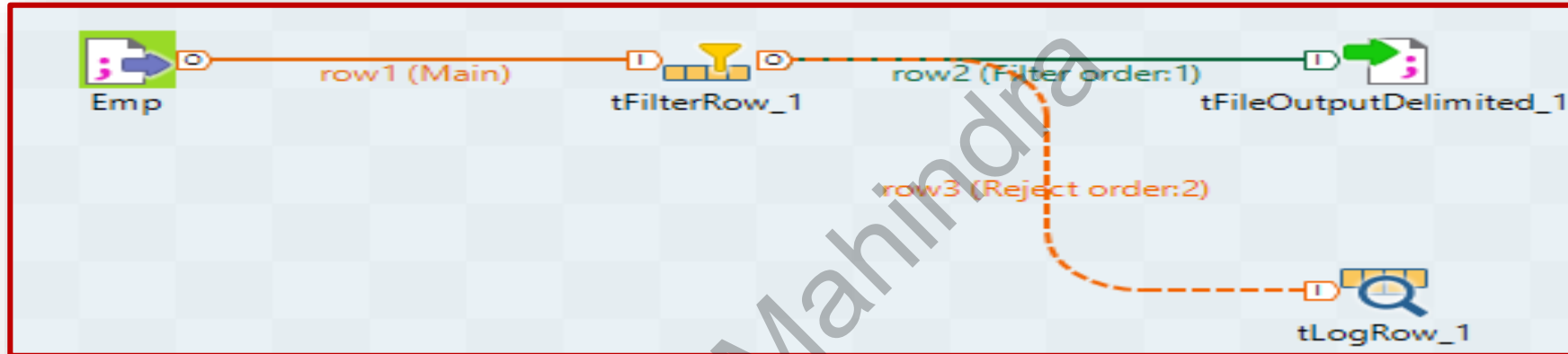
Sreenivas_Ram

Filter Row

- **tFilterRow** filters input rows by setting one or more conditions on the selected columns
- **Conditions**
 - **Input column:** Select the column of the schema the function is to be operated on
 - **Function:** Select the function on the list
 - **Operator:** Select the operator to bind the input column with the value
 - **Value:** Type in the filtered value, between quotes if needed

Filter Row

- Design the job as follows and set properties



Job(J003_FilterEx 0.1) Contexts(J003_FilterEx) Component x Run (Job J003_FilterEx)

tFilterRow_1

Basic settings

Schema: Built-In Edit schema Sync columns

Logical operator used to combine conditions: And*

Conditions

InputColumn	Function	Operator	Value
DEPTNO	Empty	Equals	20

Demo

Create a simple job

Tech Mahindra

Talend

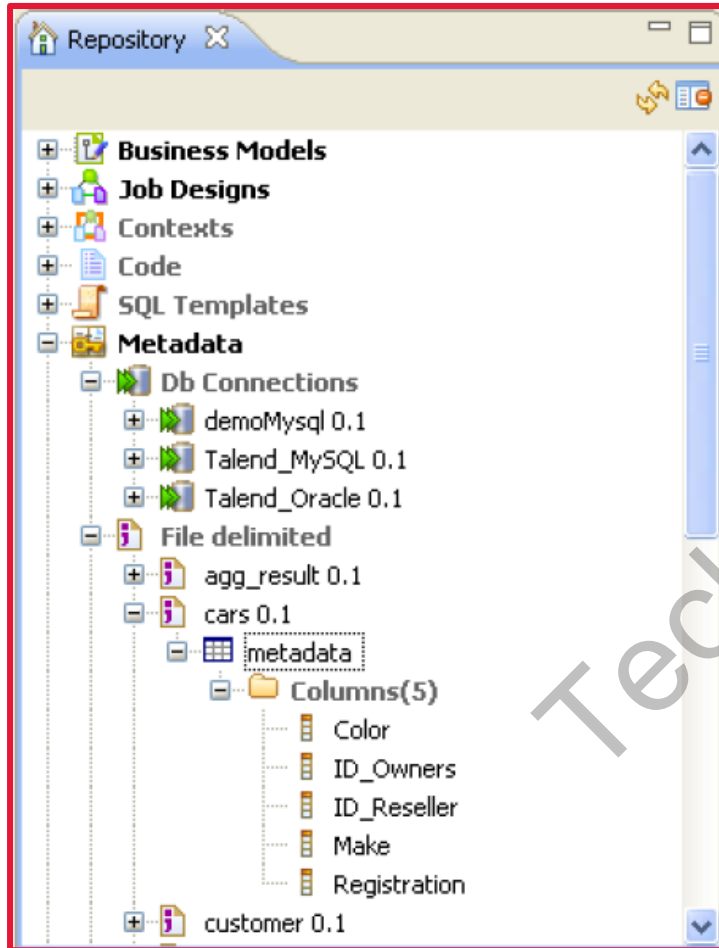
Managing Metadata

Sreenivas_Ram

Managing Metadata

- To create and manage various metadata items in the **Repository** that can be used in all the Job designs
- The **Metadata** folder in the **Repository** tree view stores reusable information on files, databases, and/or systems that need to be created in Jobs
- Various corresponding wizards help you store these pieces of information that can be used later to set the connection parameters of the relevant input or output components and the data description called "schemas" in a centralized manner in *Talend Studio*
- The procedures of different wizards slightly differ depending on the type of connection chosen

Managing Metadata



From *Talend Studio*, one can set up the following :

- a DB Schema
- a JDBC schema
- a SAS connection
- a file schema
- an LDAP schema
- a Salesforce schema
- a generic schema
- a MDM connection
- a FTP connection
- a WSDL schema
- Etc ...

Talend

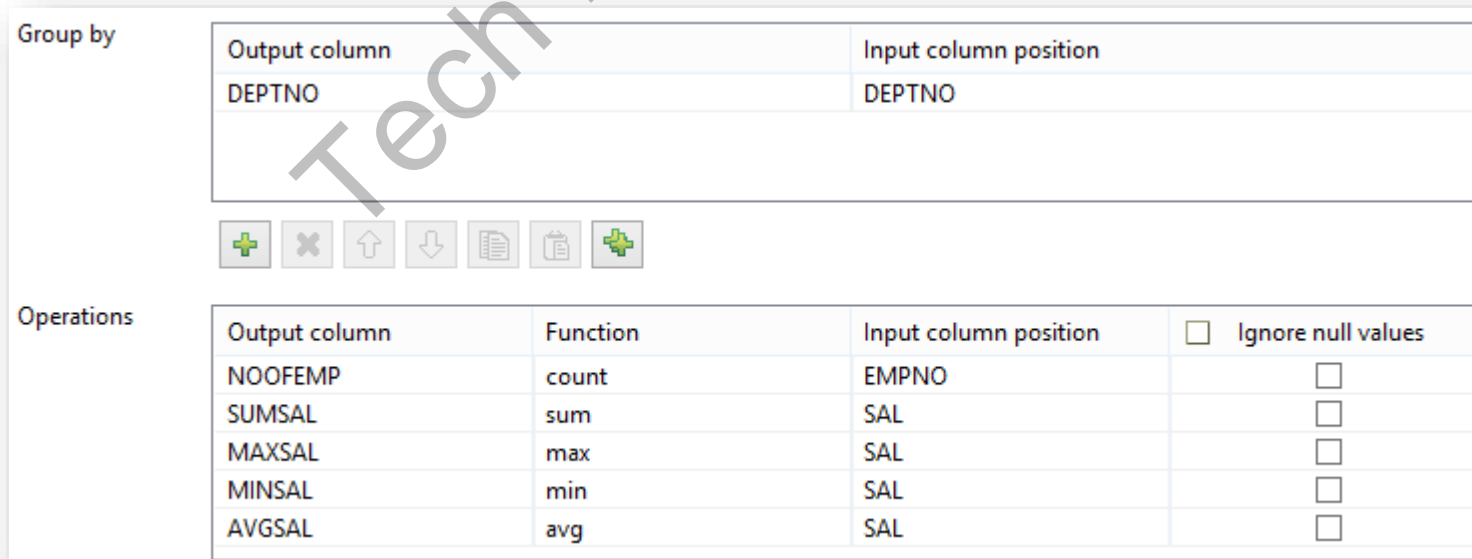
Aggregate Row

Aggregate Row

- **AggregateRow** receives a flow and aggregates it based on one or more columns. For each output line, are provided the aggregation key and the relevant result of set operations (min, max, sum...).
- **Group by**
 - Define the aggregation sets, the values of which will be used for calculations.
 - **Output Column:** Select the column label in the list offered based on the schema structure you defined. You can add as many output columns as you wish to make more precise aggregations.
 - **Input Column:** Match the input column label with your output columns, in case the output label of the aggregation set needs to be different.
- **Operations** Select the type of operation along with the value to use for the calculation and the output field

Aggregate Row

- Drag tFileInputDelimited, tAggregatorRow and tLogRow
- Set Properties of **tFileInputDelimited**
FileName, CSV Options, Header, Footer etc..
- Set Properties of **tAggregatorRow**
GroupBy : DeptNo
Operations :



The screenshot shows the configuration window for the tAggregatorRow component. It has two main sections: 'Group by' and 'Operations'.

Group by:

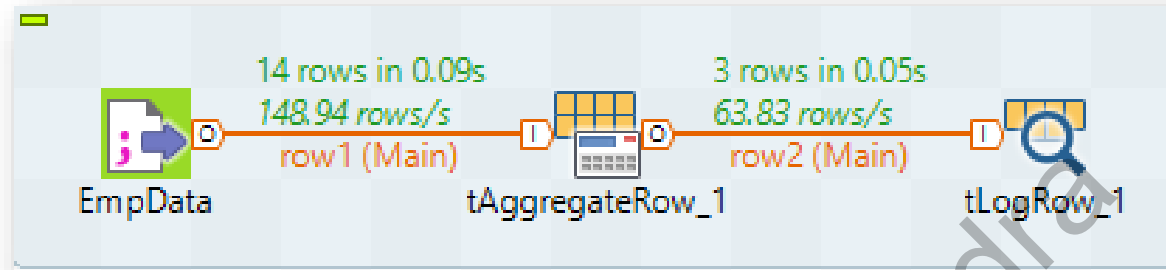
Output column	Input column position
DEPTNO	DEPTNO

Below the table are icons for adding (+), removing (x), moving up (↑), moving down (↓), and a refresh/clear icon.

Operations:

Output column	Function	Input column position	<input type="checkbox"/> Ignore null values
NOOFEMP	count	EMPNO	<input type="checkbox"/>
SUMSAL	sum	SAL	<input type="checkbox"/>
MAXSAL	max	SAL	<input type="checkbox"/>
MINSAL	min	SAL	<input type="checkbox"/>
AVGSAL	avg	SAL	<input type="checkbox"/>

Aggregate Row



Designer Code

Job(J004_AggregatorEx 0.1) Contexts(J004_AggregatorEx) Component Run (Job J004_AggregatorEx)

Job J004_AggregatorEx

Basic Run
Debug Run
Advanced settings
Target Exec
Memory Run

Execution

Run Kill Clear

Starting job J004_AggregatorEx at 16:40 09/08/2016.

[statistics] connecting to socket on port 3784
[statistics] connected

tLogRow_1					
DEPTNO	NOOFEMP	SUMSAL	MAXSAL	MINSAL	AVGSAL
20	5	10875	3000	800	2175.0
10	3	8750	5000	1300	2916.6667
30	6	9400	2850	950	1566.6666

[statistics] disconnected
Job J004_AggregatorEx ended at 16:40 09/08/2016. [exit code=0]

☐ Line limit 100 ☒ Wrap

Demo

Create a simple job

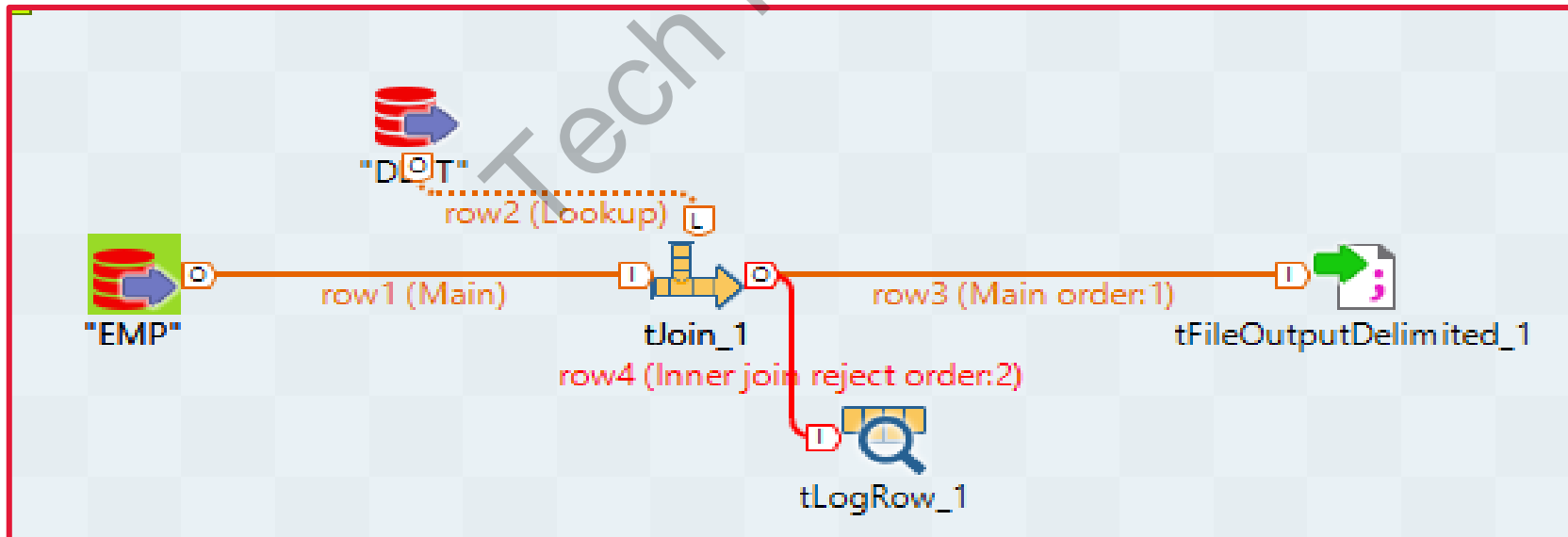
Tech Mahindra

Talend
Joiner

Tech Mahindra

Join

- Drag **Emp** and **Dept – OracleInput** Components, **tJoin**, **tFileOutputDelimited** and **tLogRow** components and connect as shown in the diagram.
- Set Properties of **tFileInputDelimited**
FileName, **CSV Options**, **Header**, **Footer** etc..



Demo

Create a simple job

Tech Mahindra

Talend

Global Variables and Contexts

Sreenivas_Ram

Global Variables and Contexts

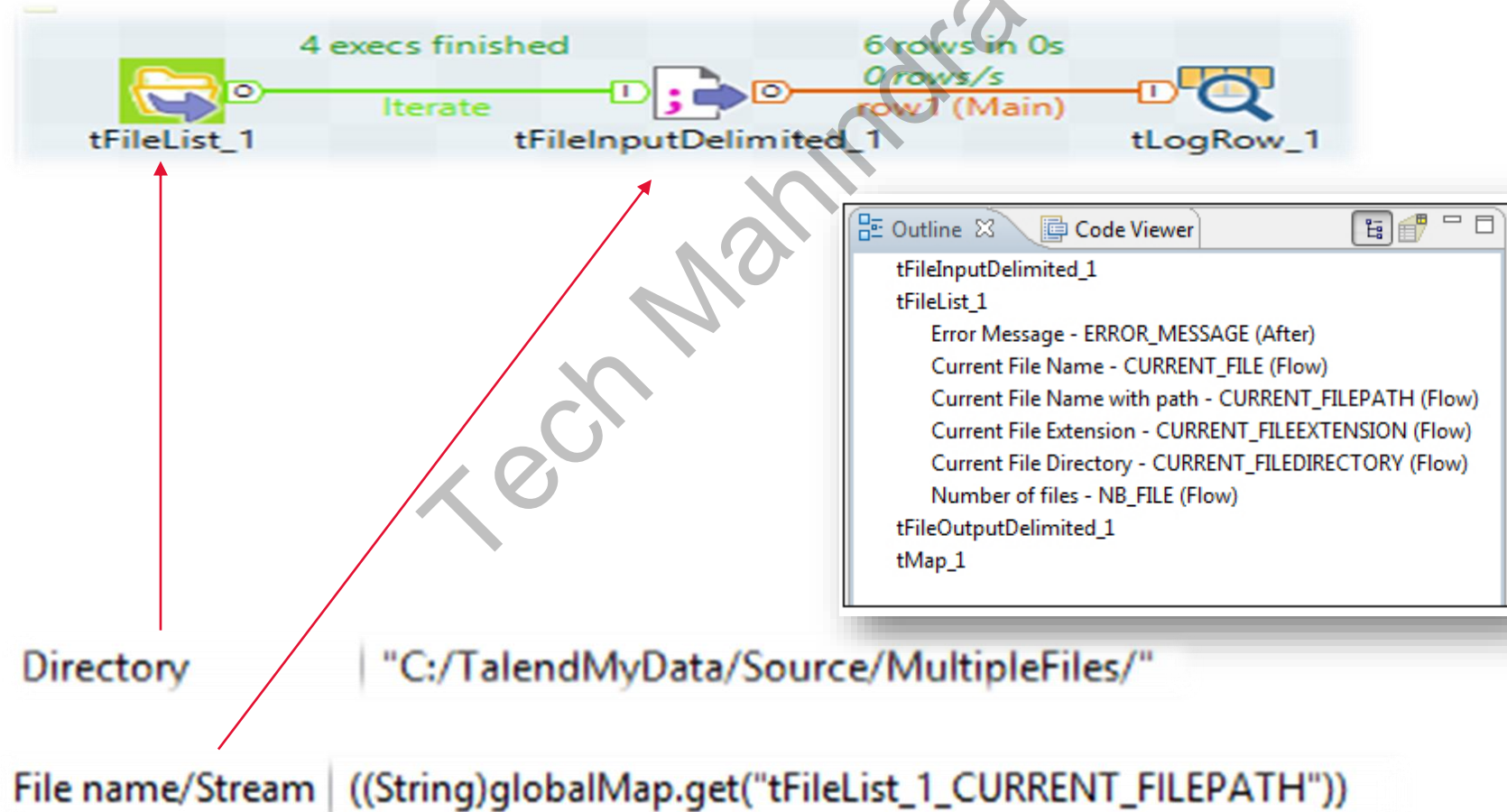
- Variables
 - Placeholder for values used during execution
 - Variable's values changes at different executions
 - values also change within a single execution
- Types of Variables:
 - Studio Global Variables
 - User defined global variables
 - Job Contexts

Studio Global Variables

Tech Mahindra

Studio Global Variables

- Available to all components, modules or functions within a Job
- **Ex:** Usage of Studio Global Variables -



User defined Global Variables

Tech Mahindra

User defined global variables

- Ad-hoc variables that can be configured in your jobs
- **Ex:**
 - Fixed" values that you need to access in your integration job—client = "Acme Corporation" or tax_rate = 20%, for example
 - Variables that you create from standard Java functions—day, month, year values, for example
 - Variables that you set based on the data accessed by the integration job
- **globalMap.put("varName", varValue)** function to sets the variable value
- **globalMap.get("varName")** function to gets the variable value
- **Ex:** Using **globalMap** function

```
System.out.println("myString=" + globalMap.get("myString"));
System.out.println("myInteger=" + globalMap.get("myInteger"));
```

User defined global variables

- Ex: Using **tSetGlobalVar** component

The screenshot shows the Job Designer interface for a job named "Job tSetGlobalVarExample 0.1". The workflow consists of two components: **tSetGlobalVar_1** and **tJava_1**. A green arrow labeled "OnSubjobOk" connects the output of **tSetGlobalVar_1** to the input of **tJava_1**. Below the workflow, the "Designer" tab is active, showing the configuration for **tSetGlobalVar_1**. The "Variables" section is expanded, displaying a table of defined global variables.

Key	Value
"myString"	"Hello World!"
"myInteger"	999

The screenshot shows the configuration for the **tJava_1** component. The "Code" tab is active, displaying the following Java code:

```
System.out.println(((String)globalMap.get("myString")));  
System.out.println(((String)globalMap.get("myInteger")));
```

JOB Contexts

Job contexts

- The variables we can create to execute jobs with different parameters for different environments or scenarios
- **Ex:** Connection info in 3-different environments – using context variables.
- Invoked at runtime
- Different ways of implementing
 - **embedded variables,**
 - **repository variables,** or
 - **external variables.**

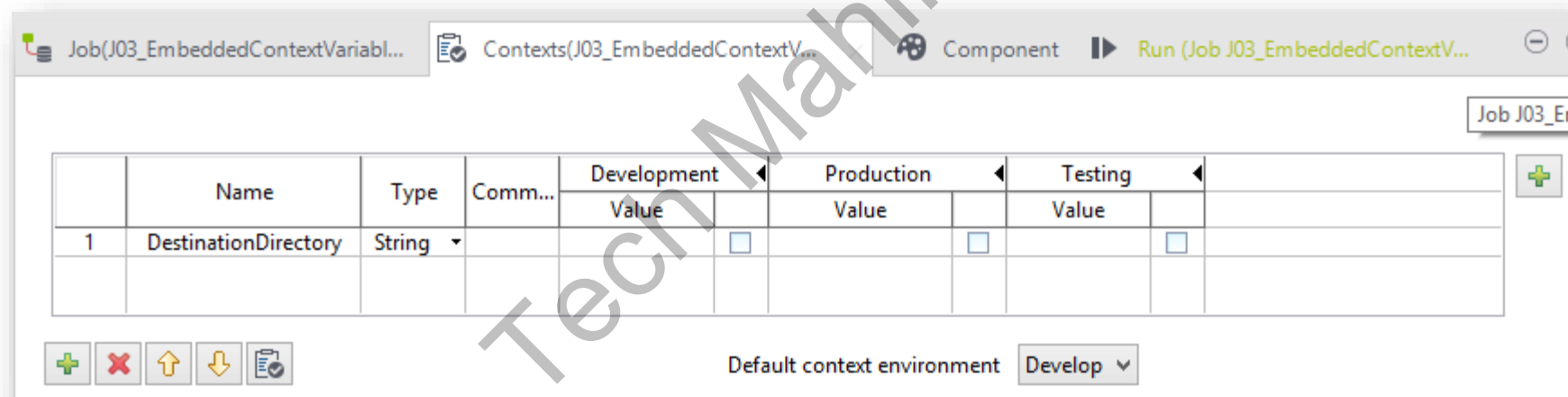
JOB Contexts

Embedded Context Variables

Tech Mahindra

Job Contexts : Embedded Variables

- Embedded in the job and are configured much like any other component parameters
- **Ex:** How to access different file locations with different context ("test" context or a "production" context)

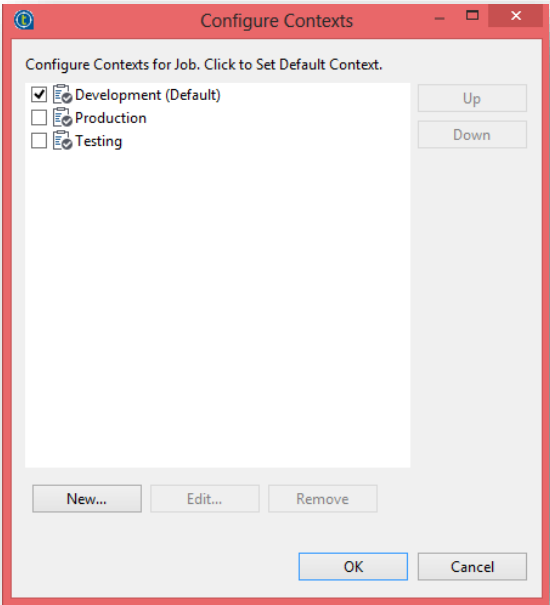


The screenshot shows the 'Job Contexts' configuration window. The title bar includes 'Job(J03_EmbeddedContextVariabl...', 'Contexts(J03_EmbeddedContextV...', 'Component', and 'Run (Job J03_EmbeddedContextV...'. The main area contains a table with the following structure:

	Name	Type	Comm...	Development	Production	Testing
				Value	Value	Value
1	DestinationDirectory	String		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there are icons for adding (+), deleting (X), moving up (↑), moving down (↓), and saving (floppy disk). At the bottom right, there is a label 'Default context environment' and a dropdown menu currently set to 'Develop'.

Job Contexts : Embedded Variables



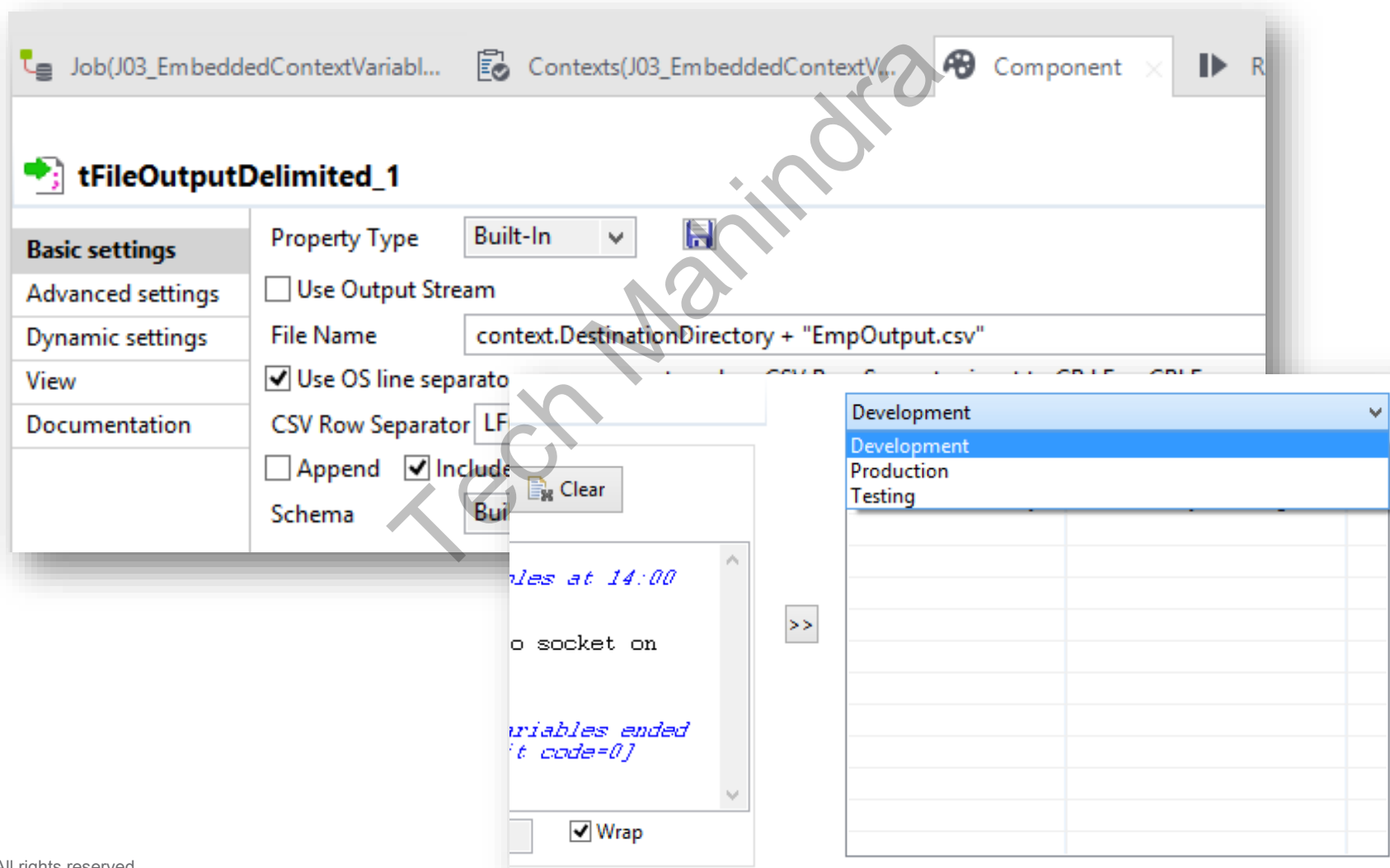
Create context variables using Context tab

- Setting multiple contexts [Development / Production / Testing] of the same variable (DestinationDirectory)

	Comment	Development		Production		Testing		
		Value		Value		Value		
1		"C:/TalendMyData/Target/Developi	<input type="checkbox"/>	"C:/TalendMyData/Target/Producti	<input type="checkbox"/>	"C:/TalendMyData/Target/Testing/	<input type="checkbox"/>	

Job Contexts : Embedded Variables

Using the context variable in the job

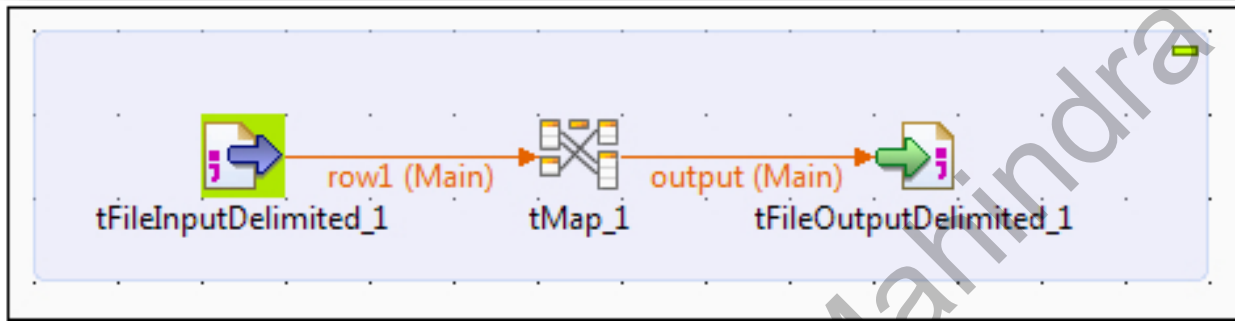


JOB Contexts

Repository Context Variables

Job Contexts : Repository context variables

- Contexts can be stored in the Studio repository, allowing them to be centrally maintained, available to other developers and reused on other jobs



Design the Job

**Create Context Group
and variables using
Repository → Contexts →
Create Context Group**

The image shows the 'Repository Explorer' on the left and a 'Context Creation Form' on the right.

Repository Explorer:

- Business Models
- Job Designs
 - J01_ContextVariables 0.1
 - J02_UserDefinedGlobalVar 0.1
 - J03_EmbeddedContextVariable
 - J04_RepositoryContextVariable
- Contexts
 - RepoContext_TargetFolder 0.1
- Code
- SQL Templates
- Metadata

Context Creation Form (Step 1 of 2):

Add any required information

Name	RepoContext_TargetFolder
Purpose	
Description	

Job Contexts : Repository context variables

Create / Edit a context group

Step 2 of 2
Define the contexts, variables and values

	Name	Type	Comment	Development Value	Production Value	Testing Value
1	DestinationFolderName	String		"C:/TalendMyDat	"C:/TalendMyD	"C:/TalendMyD

+ -

Default context environment Development

< Back Next >

Defining different Contexts using Contexts Tab

Select the Default Context

Configure Contexts

Configure Contexts for Job. Click to Set Default Context.

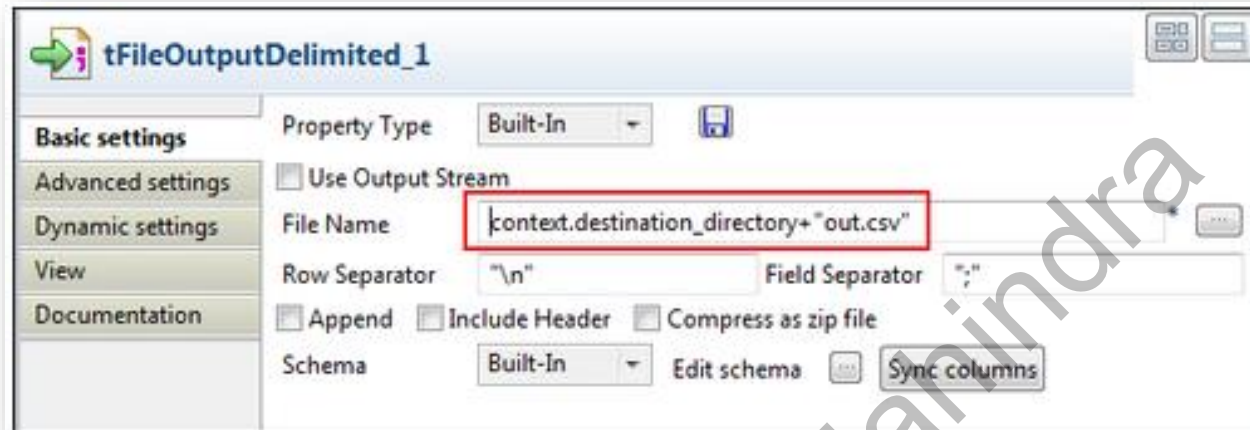
- ☒ Development (Default)
- ☐ Production
- ☐ Testing

Up Down

New... Edit... Remove

OK Cancel

Job Contexts : Repository context variables



Setting default
Context

Using the required
context at run-time

Production	
Name	Value
destination_di...	"C:/Talend/Workspace/Context/Produc...

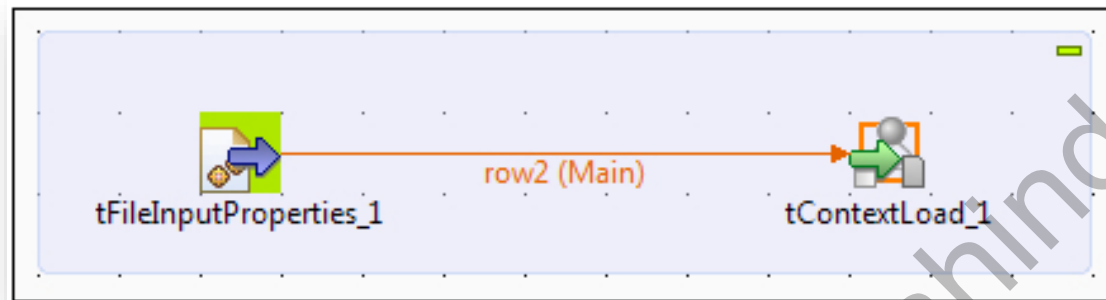
JOB Contexts

External Context Variables

Tech Mahindra

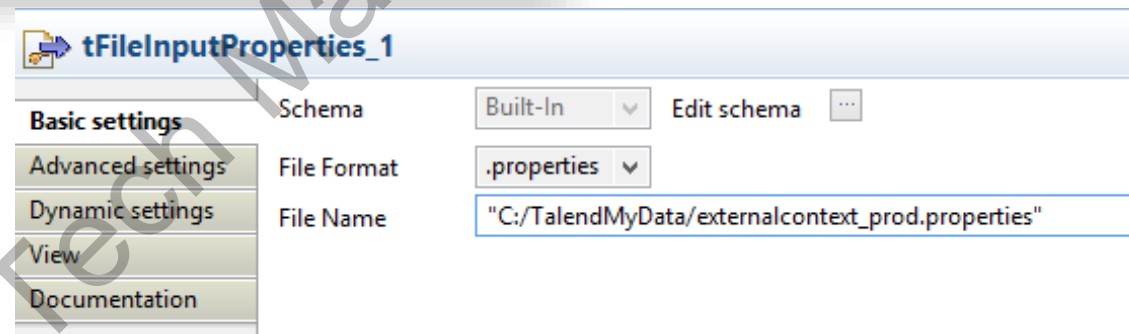
Job Contexts : External context variables

- External context variables are variables held in a file and loaded into the Studio job at runtime

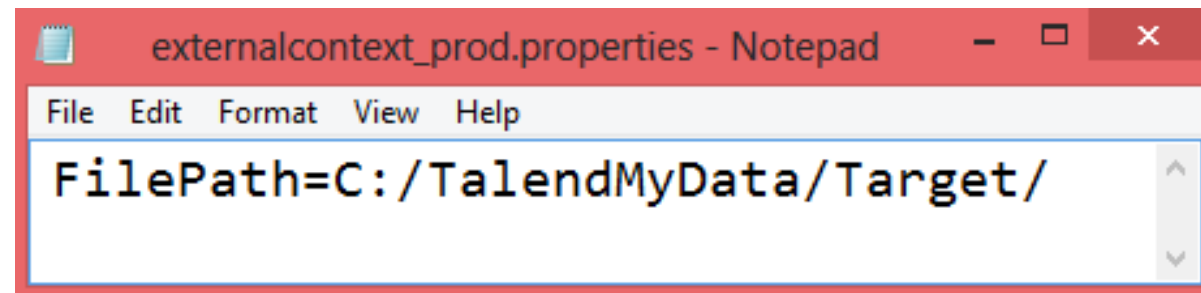


Create External Properties file and use it with File properties component

Create External Properties file and use it with File properties component



Sample properties file (External Context)

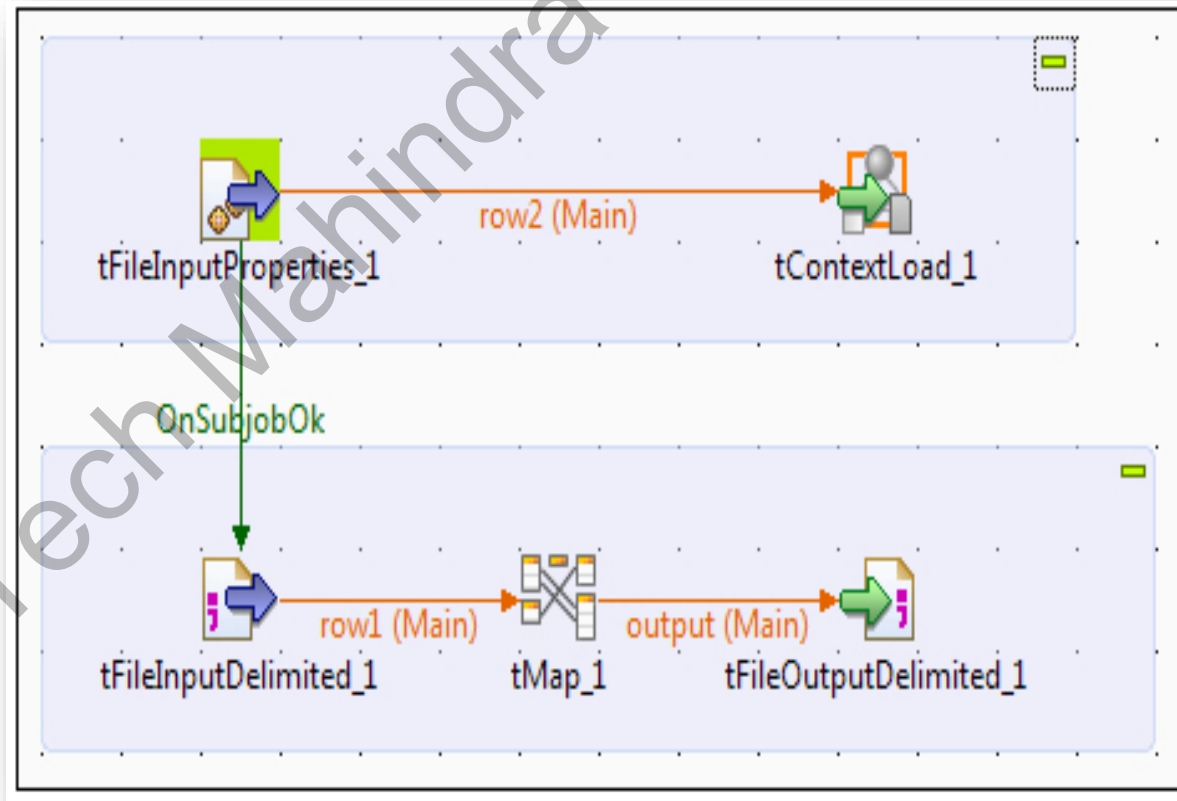


Job Contexts : External context variables

- External context variables are variables held in a file and loaded into the Studio job at runtime

Create context variable
and map to external
variable file path to the
context variable



Use it in file out



Job Contexts : External context variables

- Properties of tInputFileDelimited, tOutputFileDelimited

Basic settings	Property Type	Repository	DELIM:EmpMetaData	
Advanced settings	"When the input source is a stream or a zip file, footer and random shouldn't be big"			
Dynamic settings	File name/Stream	"C:/TalendMyData/Source/EMP.csv"		
View	CSV Row Separator	LF("\n")	Field Separator	","

 tFileOutputDelimited_1				
Basic settings	Property Type	Built-In		
Advanced settings	<input type="checkbox"/> Use Output Stream			
Dynamic settings	File Name	context.FilePath + "EmpExtrnContextOutput.csv"		
View	<input type="checkbox"/> Use OS line separator as row separator when CSV Row Separator is set to CR, LF			
Documentation	CSV Row Separator	LF("\n")	Field Separator	","

Talend

tMap - Router

Sreenivas_Ram

tMap – as router

- Drag EMP and DEPT tables from Oracle Metdata
- Drag **tMap**, 3 copies of **tFileOutputDelimited** and connect as shown in the pic below
- In Mapping editor of **tMap** :
 - Connect **DeptNo** from **EMP** to **DEPT** columns to join them
 - Let the Join condition be **InnerJoin**
 - Drag both **EMP** and **DEPT** columns on two 3-Output components
 - Select **Loop Inner Join Reject** option to **TRUE** and map related rows

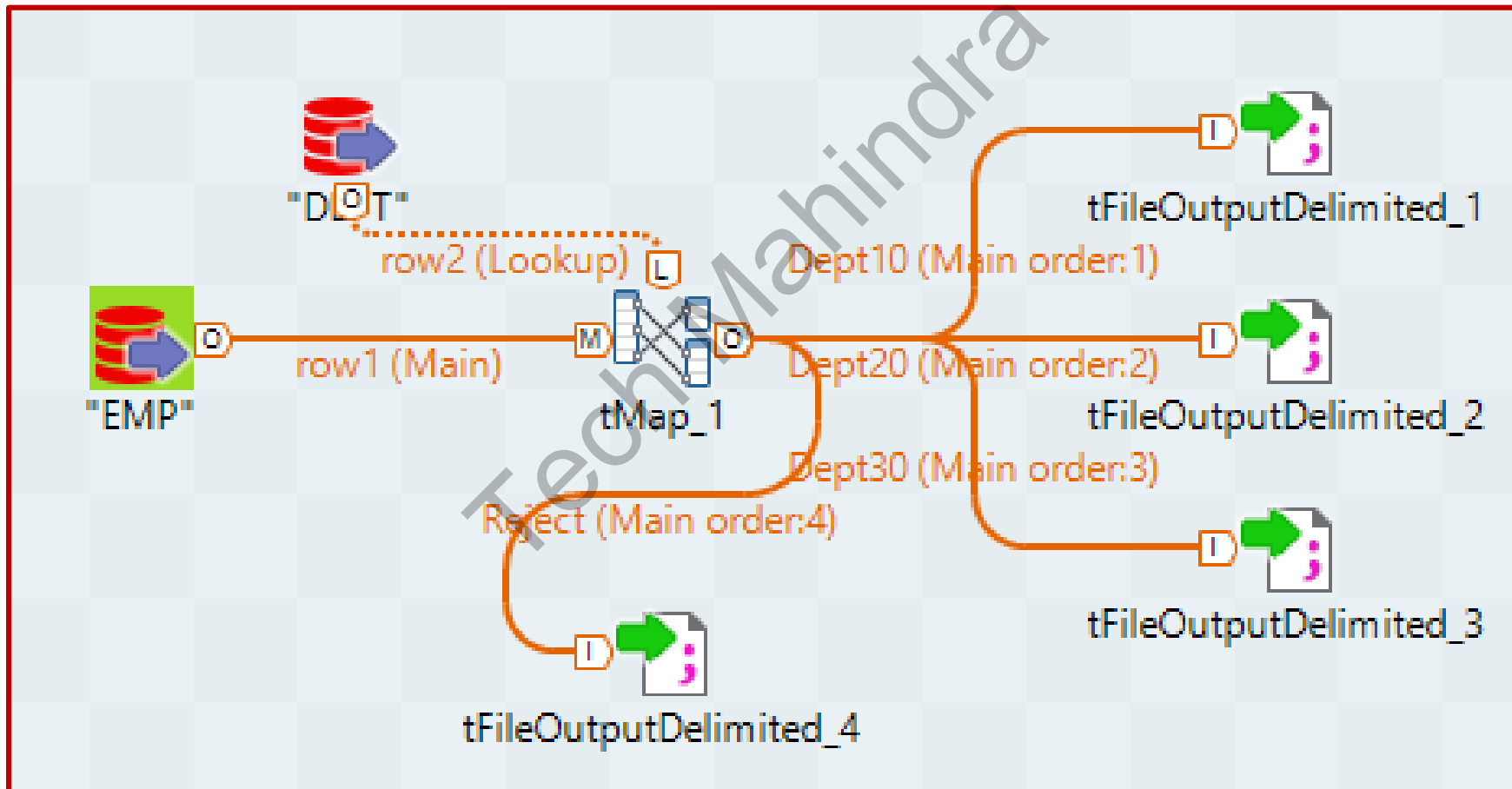
The screenshot displays the tMap Mapping Editor interface. On the left, the 'row1' component lists columns: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. Below it, the 'row2' component shows properties: Lookup M... (Load once), Match Mo... (First match), Join Model (Inner Join), and Store temp... (false). The central workspace shows a mapping diagram with orange arrows connecting the 'row1' columns to the 'row2' properties. On the right, the 'Dept10' component lists columns: Dept20, Dept30, and Reject. Below it, the 'Reject' component shows properties: Catch output reject (false), Catch lookup inner join rej... (true), and Schema Type (Built-In). The 'Catch lookup inner join rej...' property is highlighted in yellow. Below the properties, the 'Expression' table lists: row1.EMPNO, row1.ENAME, and row1.DEPTNO. The 'Column' table lists: EMPNO, ENAME, and DEPTNO.

Property	Value
Catch output reject	false
Catch lookup inner join rej...	true
Schema Type	Built-In

Expression	Column
row1.EMPNO	EMPNO
row1.ENAME	ENAME
row1.DEPTNO	DEPTNO

tMap — as router contd ...

- **Save** and **Run** the job



Talend
tReplace

Sreenivas_Ram

tReplace

- This Job searches and replaces various typos and defects in a csv file then operates a column filtering before producing a new csv file with the final output
- Drag **tFileInputDelimited**, **tReplace**, **tFilterColumns**, **tFileOutputDelimited** on to the job.

Street	FirstName	LastName	Amount
street	John	Kennedy	98.3
street	Richad	Nikson	78.23
street	Richard	Nikson	78.2
street	toto	Nikson	87.23
street	Richard	Nikson	87.23
street	George*t	bush	99.98

tReplace_1

Schema: Built-In Edit schema Sync columns

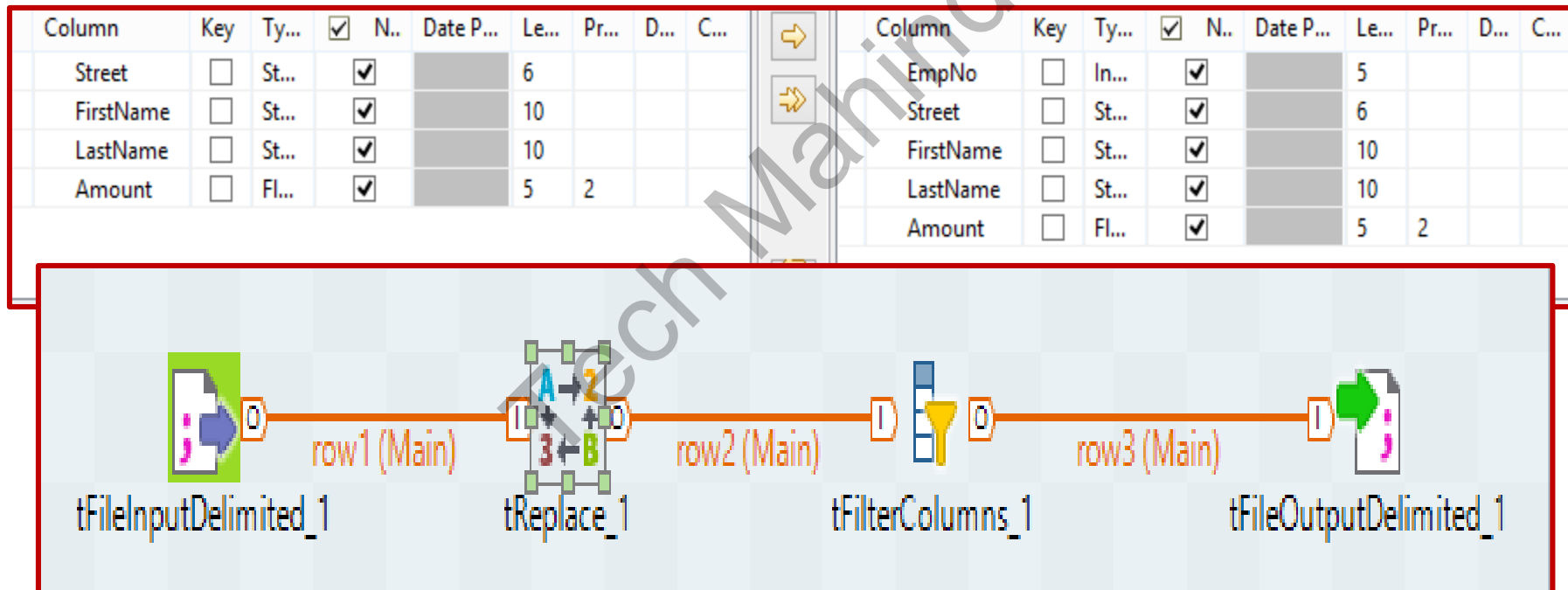
☒ Simple mode

Search/Replace:

InputColumn	Search	Replace with	<input checked="" type="checkbox"/> Whole w...	<input type="checkbox"/> Case Sen...	<input type="checkbox"/> Glob expr.
Street	"streat"	"Street"	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Street	"street"	"Street"	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FirstName	"*t"	""	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
LastName	"Nikson"	"Nikxon"	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FirstName	"toto"	"Toto"	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

tReplace ... contd.

- Add an additional column in **tFilterColumn** component
- **Save** and **Run** the job



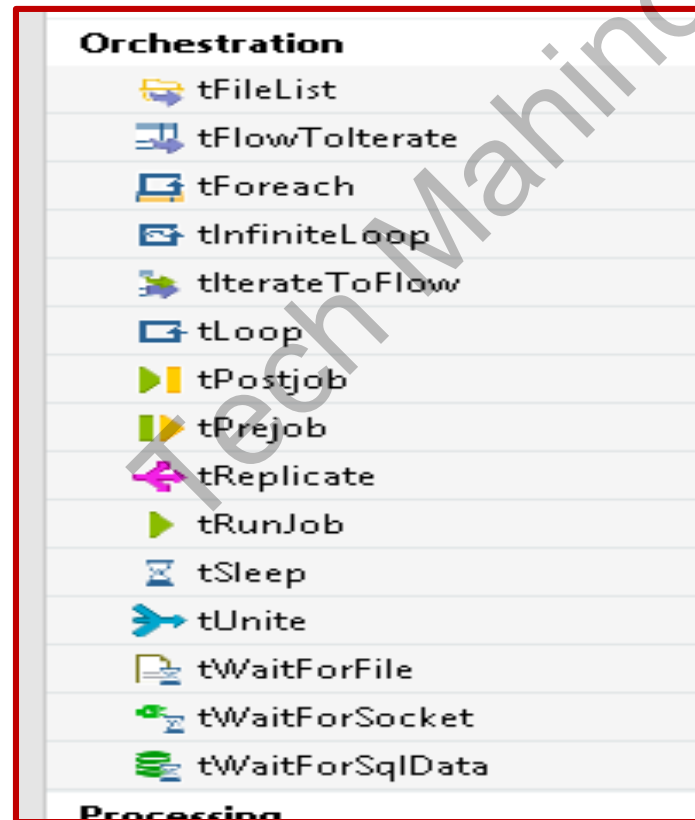
Talend

Execute Job in Loop

Sreenivas_Ram

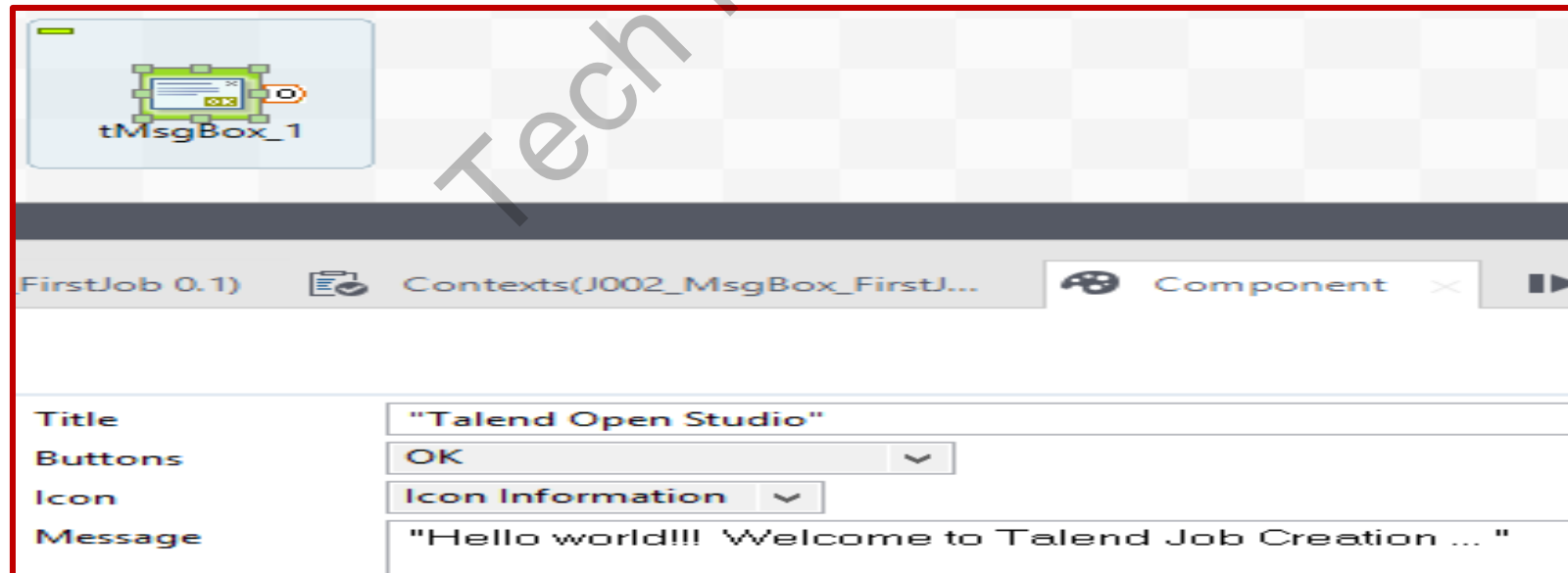
Orchestration components

- The Orchestration family groups together components that help you to **sequence** or **orchestrate** tasks or processing in your Jobs or subjobs and so on



Execution of a Job in Loop

- This scenario describes a Job composed of a parent Job and a child Job. The parent Job implements a loop which executes n times a child Job, with a pause between each execution



In the parent Job

- Drop a **tLoop**, a **tRunJob** and a **tSleep** component from the **Palette** to the design workspace.
- Connect the **tLoop** to the **tRunJob** using an **Iterate** connection
- Then connect the **tRunJob** to a **tSleep** component using a **Row** connection.
- **tLoop** Component properties :
 - From : 1
 - To : 2
 - Step : 1
- **tRunJob** component Properties :
 - Job : **J002_MsgBox_FirstJob**
- **tSleep** component properties
 - pause (in seconds) : 10

On the Child Job

- Drop the following components:
tMsgBox
- On **MsgBox** properties:
Message: "**Hello world!!! Welcome to Talend Job Creation ...** "

▪ **Run the job**

Talend

Tech Mahindra

Sreenivas_Ram

tPreJob
tPostJob

Tech Mahindra

tPreJob & tPostJob Components

- The **tPreJob** and **tPostJob** components are part of the Orchestration family of components
- These components help to make your Jobs flow with more elegance, and provide the ability to perform some clean-up, should your Job failure

Tech Mahindra

tPreJob

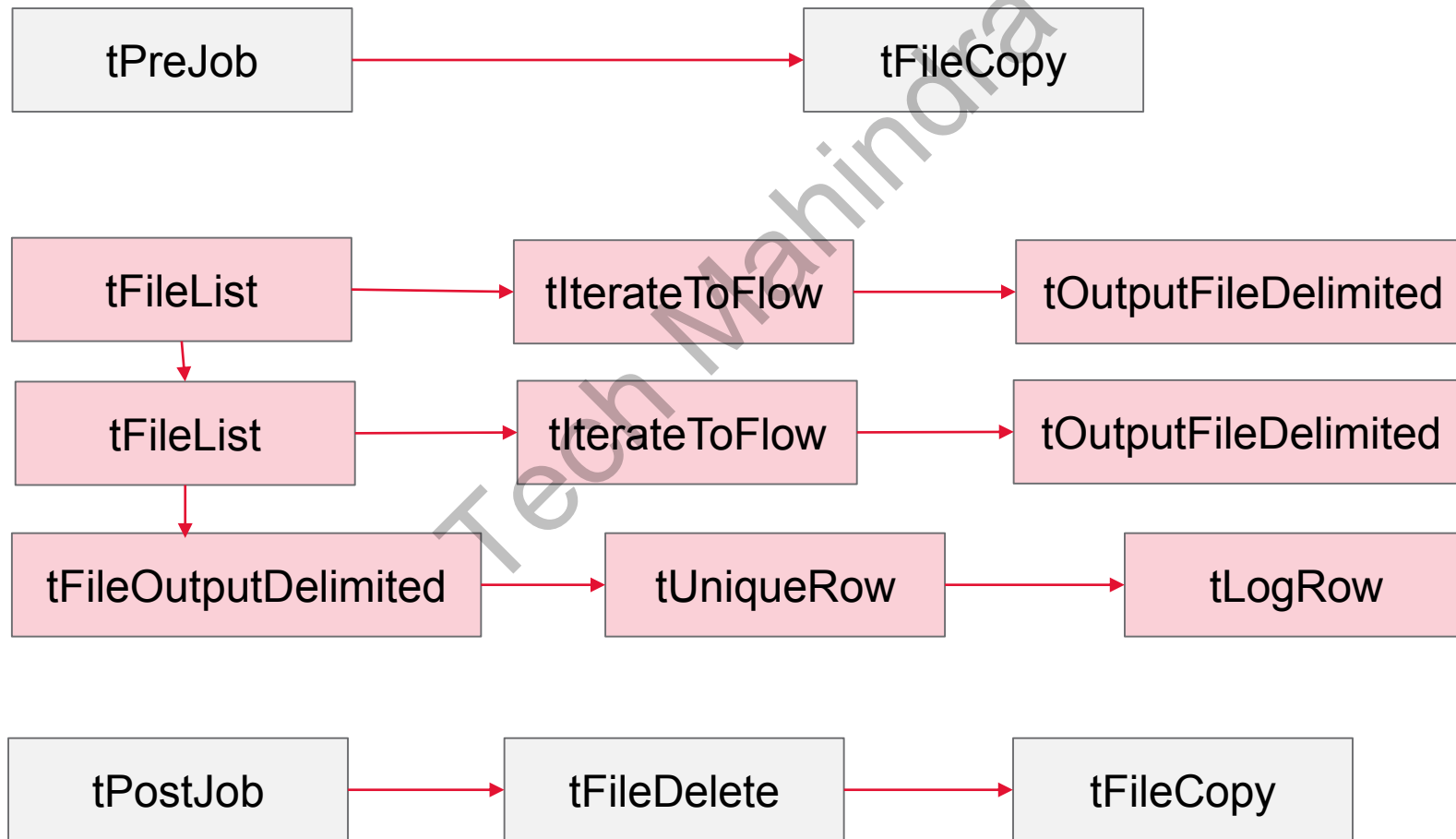
- The **tPreJob** component starts the first series of Subjobs that will be executed, when your Job starts
- Functionally, it performs no action that you could not achieve by connection your components in the correct order; however, it does provide a clear demarcation between your Job's initialisation sequence, and the main tasks that the Job is intended to perform
- The following tasks could be considered as pre-Job tasks: -
 - Load Context
 - Validate external environment
 - Establish database connections
 - Test connectivity to external services
 - Check input files exist
 - logging the start of your Job

tPostJob

- The **tPostJob** component starts the final series of Subjobs that will be executed, when your Job finishes; that is, after **tPreJob** has completed and any other Subjobs that you have defined
- A key aspect to the execution of this component is that it is always executed, even when an Exception is thrown by any previous processing
- This makes it ideal for controlling de-initialisation and clean-up.
 - The following tasks could be considered as post-Job tasks: -
 - Deletion of temporary files
 - Database disconnection
 - logging the finish of your Job

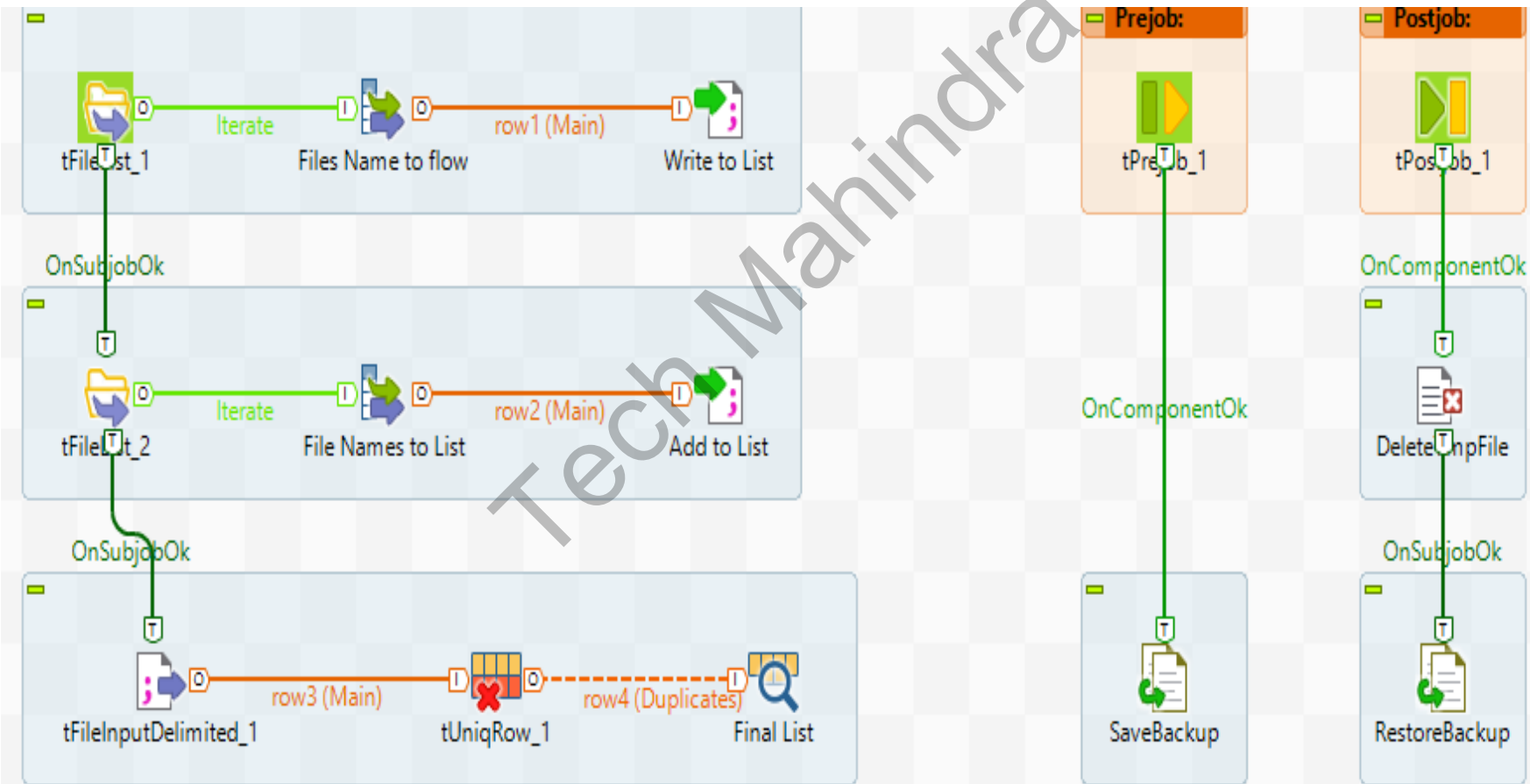
Job Design

- Design a Job with the following Components



Connectors

- These two components only support an *out* connector of **OnComponentOK**.



Configurations (Properties)

- **InPreJob > Save Backup (tFileCopy)**

SaveBackup (tFileDelimited)

FileName	"C:/TalendMyData/Source/MultipleFiles/EMP10.csv"
Destination Directory	"C:/TalendMyData"
Destination Name	"NewEmp10.csv"

- **InPostJob > Delete Temp File (tFileCopy)**

Delete Tmp File (tFileDelete)

FileName	"C:/TalendMyData/Source/MultipleFiles/EMP10.csv"
----------	--

Restore Backup (tFileCopy2)

FileName	"C:/TalendMyData/Source/MultipleFiles/NewEmp10.csv"
Destination Directory	"C:/TalendMyData/Source/MultipleFiles/"
Rename Destination Name	"Emp10.csv"
Remove Source File	Select
Replace existing File	Select

Configurations (Properties) ...contd.,

■ Main job

tFileList	
Directory	"C:/TalendMyData/Source/MultipleFiles/"
File Names to flow (tIterateToFlow)	
FileName	"((String)globalMap.get("tFileList_1_CURRENT_FILE"))"
tFileOutputDelimited (Write to List)	
FileName	"C:/TalendMyData/FileList1.csv"
tFileList	
Directory	""C:/TalendMyData/Source/MultipleFiles2/"
File Names to flow (tIterateToFlow)	
FileName	"((String)globalMap.get("tFileList_1_CURRENT_FILE"))"
tFileOutputDelimited (Write to List)	
FileName	"C:/TalendMyData/FileList1.csv"
Select	Append Option

Configurations (Properties) ...contd.,

- **Main job ... contd...**

tFileInputDelimited

FileName/stream	"C:/TalendMyData/FileList1.csv"
-----------------	---------------------------------

tUniqueRow

UniqueKey – Add FileName	Key attribute
-----------------------------	---------------

tLogRow

Select	Table View
--------	------------

Talend

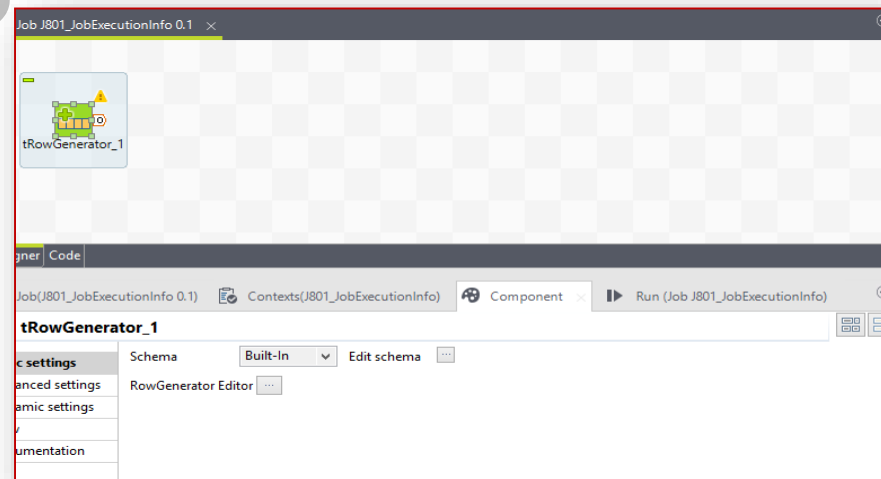
Information about Jobs Executions

Sreenivas_Ram

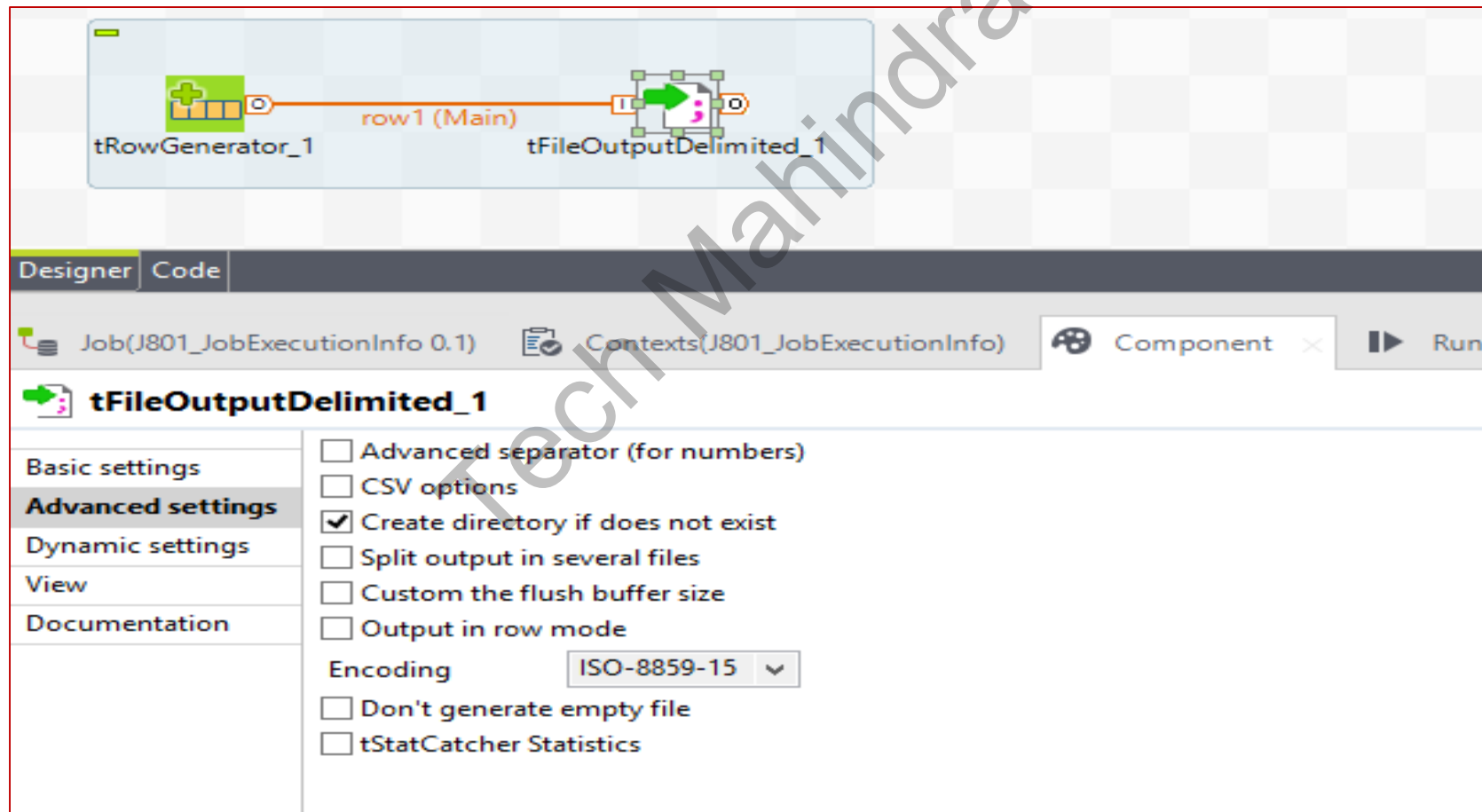
■ Job Execution Info

- Create a New job - Name it as **JobExecutionInfo**
- Drag a RowGenerator Component and create row structure as :
EmpNo Integer **Numeric.random(1101-1900)**
EName String **RowGenerator.getLastName()**
SAL Integer **Numeric.random(3000,99000)**

Schema		Functions	Environment variables	Preview
Column	Type	Functions	Environment variables	Preview
EMPNO	Integer	Numeric.random(int,int)	min value=> 1101 ; max value=> 1990 ;	
ENAME	String	TalendDataGenerator.getLastName()		
SAL	Integer	Numeric.random(int,int)	min value=> 3000 ; max value=> 99000 ;	



- Add **tFileOutputDelimited** to the Job
- Connect both RowGenerator and FileOutputDelimited
- Create an error – say : **uncheck Create Directory if not exists** in Advanced settings



- Run the Job by clicking – **F6**

The screenshot displays the Talend Studio interface during a job execution. At the top, a job diagram shows a flow from 'tRowGenerator_1' to 'tFileOutputDelimited_1'. The 'tFileOutputDelimited_1' component is highlighted with a red box and an 'Error!' icon. Below the diagram, the 'Designer' tab is active, and the 'Run' button is highlighted. The 'Job J801_JobExecutionInfo' window is open, showing the 'Execution' tab. The execution log contains the following text:

```
Starting job J801_JobExecutionInfo at 15:54 11/08/2016.  
[statistics] connecting to socket on port 3752  
[statistics] connected  
Exception in component tFileOutputDelimited_1  
java.io.FileNotFoundException:  
C:\TalendMyData\NewTarget\EmployeeData.csv (The system cannot find the  
path specified)  
    at java.io.FileOutputStream.open0(Native Method)  
    at java.io.FileOutputStream.open(Unknown Source)  
    at java.io.FileOutputStream.<init>(Unknown Source)  
    at java.io.FileOutputStream.<init>(Unknown Source)  
    at  
    prooj001.j801_jobexecutioninfo_0_1.J801_JobExecutionInfo.tRowGenerator_  
    Process(J801_JobExecutionInfo.java:559)  
    at
```

At the bottom of the execution window, there are checkboxes for 'Line limit' (set to 100) and 'Wrap' (checked).

- Add an ErrorMsg in Context and use it in ErrorMsg box

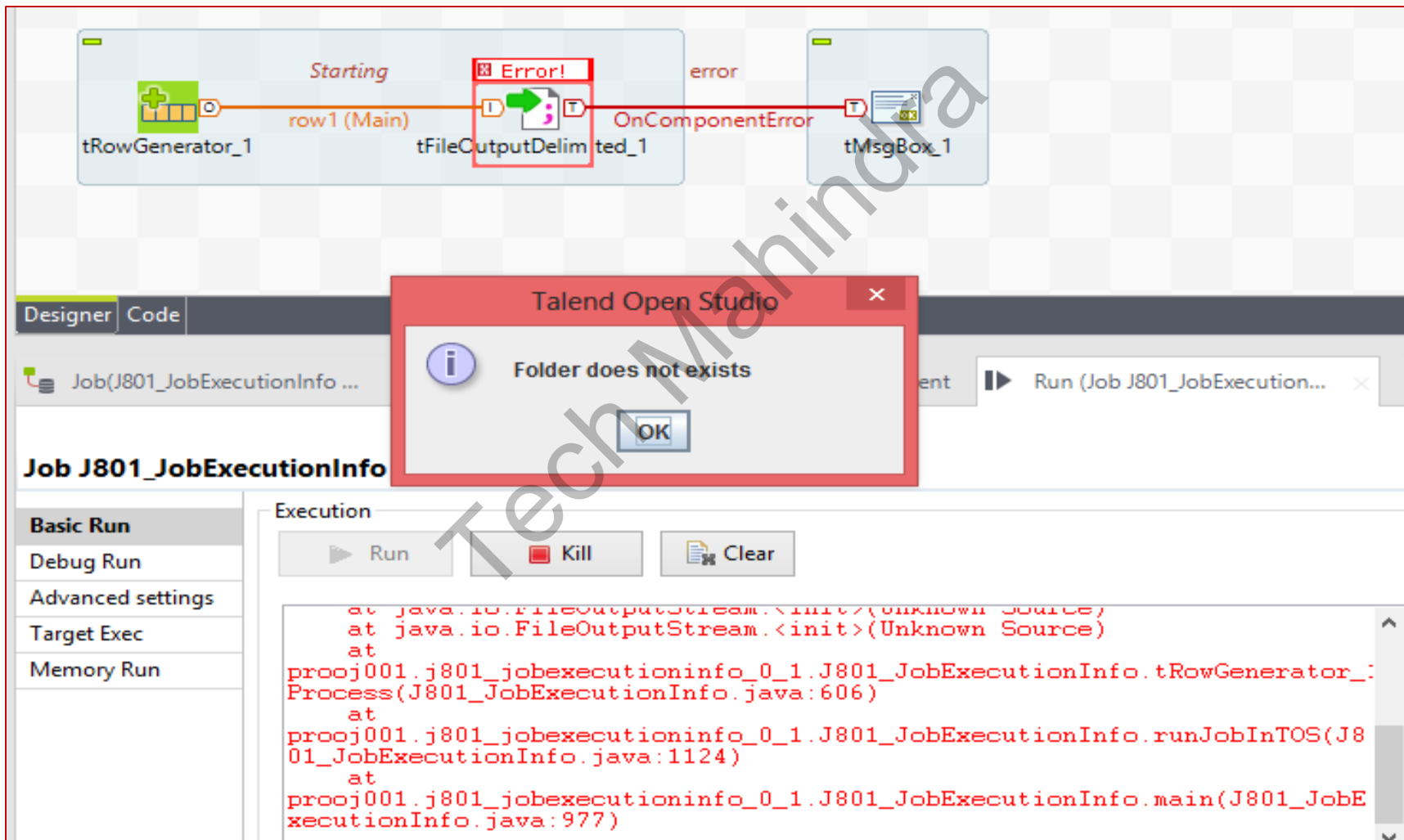
The screenshot displays a workflow designer interface. The top section shows a workflow diagram with the following components and connections:

- Starting** (Start node) connects to **tRowGenerator_1** (row1 (Main)).
- tRowGenerator_1** connects to **tFileOutputDelimited_1** (highlighted with a red box and an "Error!" icon).
- An **OnComponentError** event is connected to **tFileOutputDelimited_1** and **tMsgBox_1**.
- The connection from the event to **tMsgBox_1** is labeled **error**.

The bottom section shows the **Contexts** table for the job **Job(J801_JobExecutionInfo ...)**.

	Name	Type	Comment	Default Value
1	ErrorMsgGroup (from repository)			
2	ErrorMsg	String		"Folder does not exists "

- Add **tMsgBox** to the job
- Connect using **OnComponentError** to FileOutputDelimited



- Drag **tDie** component and connect in place of **tMsgBox**, then delete tMsgBox

The screenshot displays the SAP ABAP Designer interface. In the top workspace, a flow diagram is visible. It starts with a 'Starting' event, followed by a 'row1 (Main)' loop. Inside the loop, there is a 'tFileOutputDelimited_1' component. A red line labeled 'OnError' connects the 'tFileOutputDelimited_1' component to a 'tDie_1' component. The 'tDie_1' component is a red circle with a white 'X' inside. Below the workspace, there is a 'Designer' tab and a 'Code' tab. The 'Designer' tab is active, showing the 'tDie_1' component's properties. The properties are as follows:

tDie_1	
Basic settings	Die message: context.ErrorMessage
Advanced settings	Error code: 4
Dynamic settings	Priority: Error
View	
Documentation	

- Add **tWarn** and connect to **tRowGenerator** using **onSubjectOK**
- Add **tLogCatcher** and **tLogRow** as shown
- Run the Job – **F6**

The screenshot displays the Talend Studio interface. The top section shows the job design canvas with the following components and connections:

- tWarn_1** (Warning icon) connected to **tRowGenerator_1** via the **OnSubJobOk** event.
- tRowGenerator_1** (Data generator icon) outputs **row1 (Main)** to **tFileOutputDelimited_1** (File output icon).
- tFileOutputDelimited_1** has an **OnError** event connected to **tDie_1** (Fatal error icon).
- tLogCatcher_1** (Log catcher icon) is connected to **tLogRow_1** (Log row icon).

The bottom section shows the job execution details for **Job(J801_JobExecutionInfo 0.1)**. The **Execution** tab is active, displaying the following error message:

```
Exception in component tFileOutputDelimited_1
java.io.FileNotFoundException: C:\TalendMyData\NewTarget\EmployeeData.csv (The
at java.io.FileOutputStream.open0(Native Method)
at java.io.FileOutputStream.open(Unknown Source)
at java.io.FileOutputStream.<init>(Unknown Source)
```

- Add another **tMsgBox** and connect to **tLogRowGenerator** using **OnSubjobOK** – add “**Job is done !!!**” in the MsgBox text
- **Run the job – F6.**

The screenshot shows the Talend Studio interface. The top part displays the job design, including components like **tWtH_1**, **tLogCatcher_1**, **tLogRow_1**, **tRowGenerator_1**, **tFileOutputDelimited_1**, and **tDie_1**. The **tFileOutputDelimited_1** component is highlighted with a red box and labeled "Error!". The bottom part shows the job execution details for "Job J801_JobExecutionInfo". The "Execution" tab is active, displaying the error message: "Exception in component tFileOutputDelimited_1 java.io.FileNotFoundException: C:\TalendMyData\NewTarget\EmployeeData.csv (The system cannot find the path specified) at java.io.FileOutputStream.open0(Native Method)".

- Correct the error [Create the directory in **tFileOutputDelimited**]
- Run the Job. You will see the Msg – **Job is done !!!**

The screenshot displays the Talend Open Studio interface during a job execution. A central message box titled "Talend Open Studio" with an information icon and the text "Job is done !!!" is prominently shown. The background shows the job design canvas with several components: a warning icon (tWarn_1), a row generator (tRowGenerator_1) with performance metrics "100 rows in 0.05s" and "2127.66 rows/s", a file output component (tFileOutputDelimited_1), a log catcher (tLogCatcher_1) with metrics "1 rows in 0.12s" and "8 rows/s", and a log row component (tLogRow_1). The bottom panel shows the "Job J801_JobExecutionInfo" with a log table.

Job J801_JobExecutionInfo				
Execution				
2016-08-11 17:59:08	TKzDFd	TKzDFd	TKzDFd	PROOJ001 J801_JobExecutionInfo
tWarn	tWarn_1	Job Started ...	42	

Talend

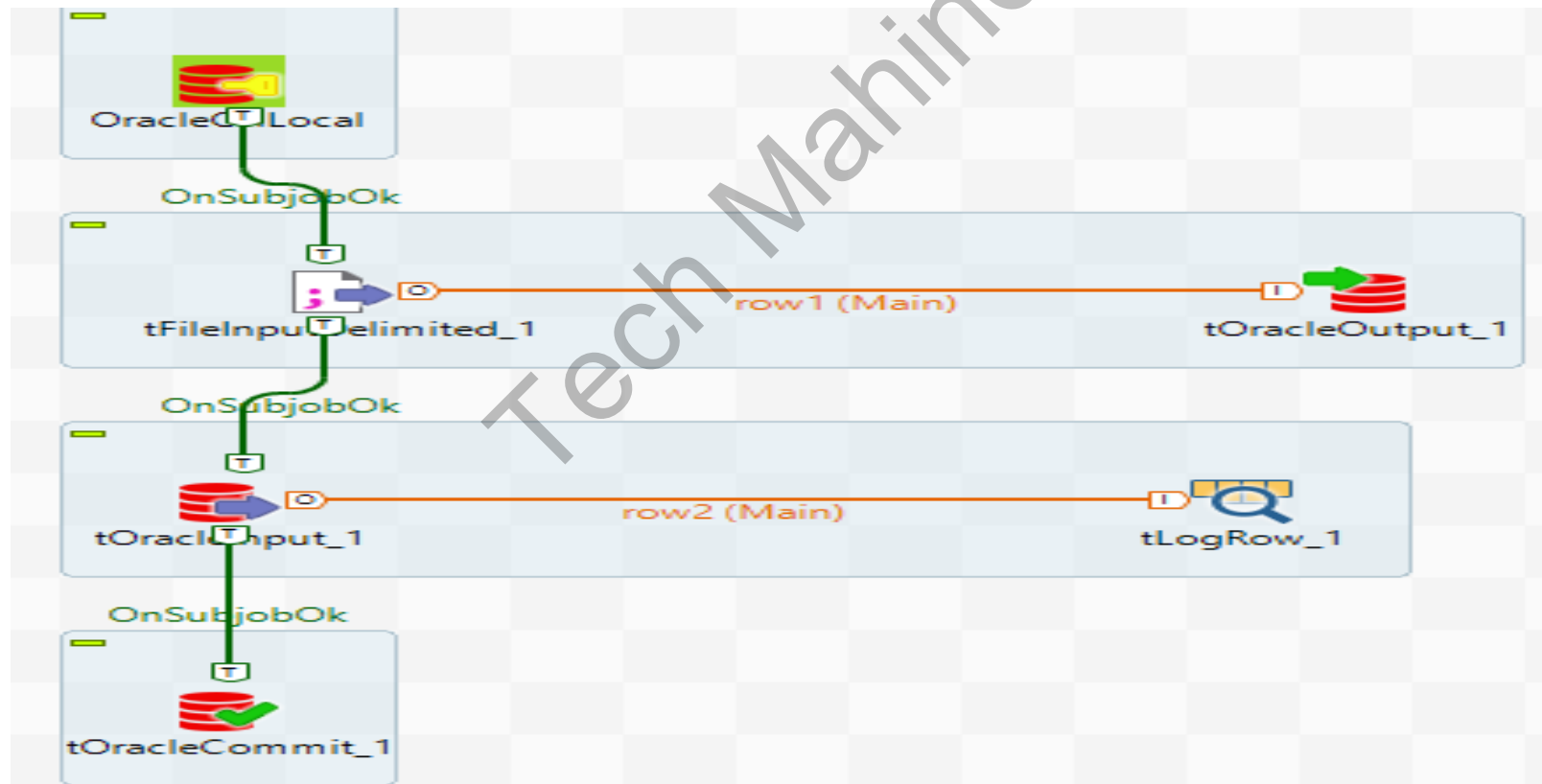
Oracle database

Sreenivas_Ram

Inserting data into a **Oracle** table and
extracting useful information from it

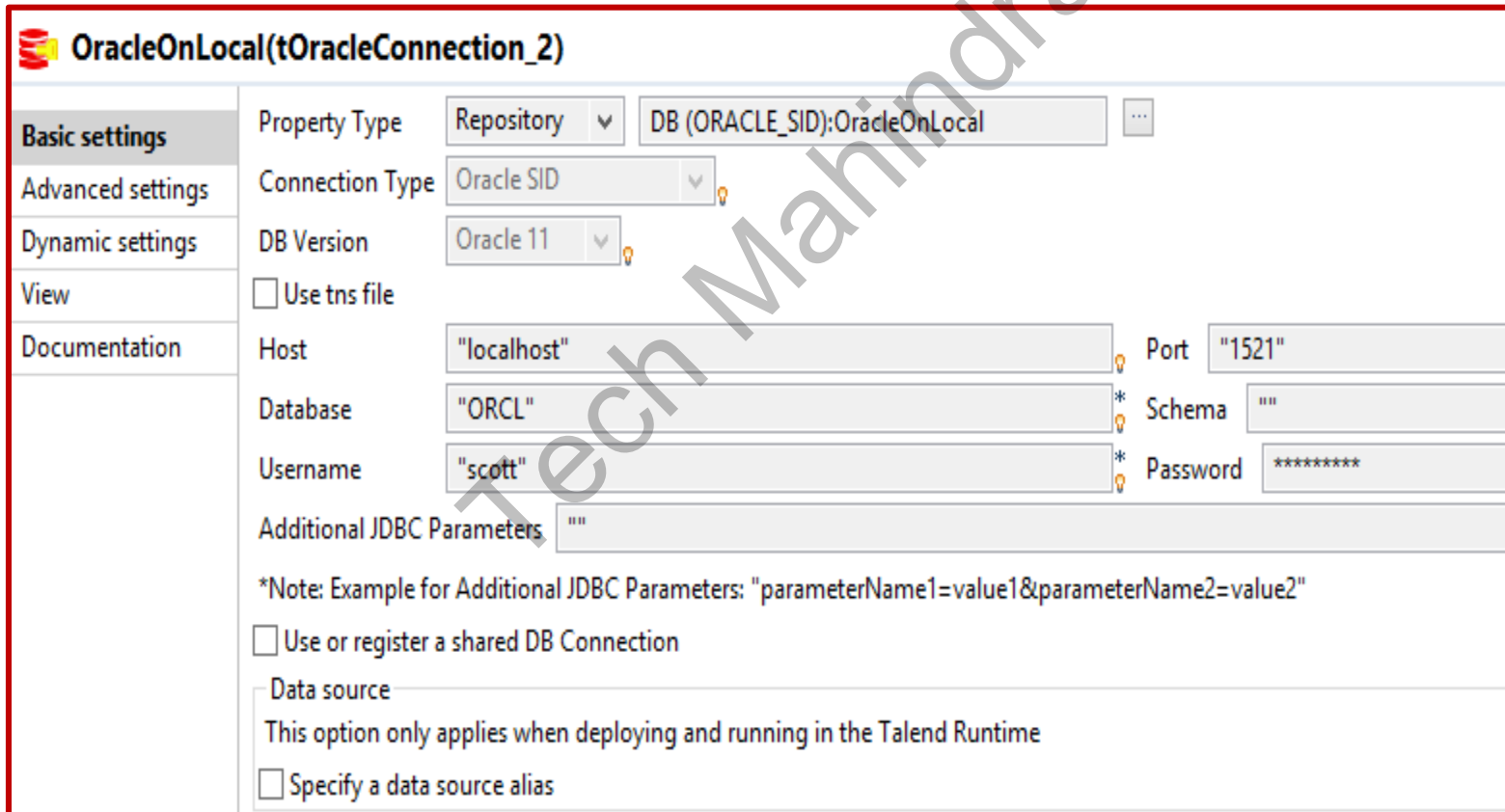
Writing and Reading data - Oracle

- Design the job with **tOracleConnection**, **tFileInputDelimited**, **tOracleOutput**, **tOracleInput**, **tLogRow** and **tOracleCommit** components as follows:



Configure the components

- **Oracle Connection** : or : Configure **Metadata** to set **Oracle Connection**



OracleOnLocal(tOracleConnection_2)

Basic settings

Property Type: Repository | DB (ORACLE_SID):OracleOnLocal

Connection Type: Oracle SID

DB Version: Oracle 11

☐ Use tns file

Host: "localhost" | Port: "1521"

Database: "ORCL" | Schema: ""

Username: "scott" | Password: "*****"

Additional JDBC Parameters: ""

*Note: Example for Additional JDBC Parameters: "parameterName1=value1¶meterName2=value2"

☐ Use or register a shared DB Connection

☐ Data source

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

☐ Specify a data source alias

Reading data – Writing to ORACLE table

- Sample data - **tFileInputDelimited**

tFileInputDelimited_1

Basic settings Property Type **Repository** **DELIM:Ora-EmpSample**

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

Dynamic settings File name/Stream **"C:/TalendMyData/Source/Ora-SampleEmp.csv"**

View CSV Row Separator **LF("\n")** Field Separator **" "**

Documentation ☒ **CSV options** Escape char **""** Text enclosure **""**

Header **1** Footer **0** Limit

Schema **Repository** **DELIM:Ora-EmpSample - metadata** Edit schema

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

id	name	wage
51	Harry	2300
40	Ronald	3796
17	Theodore	2174
21	James	1986
2	George	2591
89	Calvin	2362
84	Ulysses	3383
4	Lyndon	2264
17	Franklin	1780
86	Lyndon	3999

tOracleOutput_1

Basic settings ☒ **Use an existing connection** Component List **tOracleConnection_2 - OracleOnLocal**

Advanced settings Table **"WagesInfo"**

Dynamic settings Action on table **Create table if does not exist** Action on data **Insert**

View Warning : this component configuration will automatically generate a commit before data insert/update/delete

Documentation Schema **Built-In** Edit schema **Sync columns**

☐ Die on error

Read data - Oracle Input - write to Log

The image displays three Talend components used for reading data from an Oracle database and writing it to a log.

tOracleInput_1

- Basic settings:**
 - ☒ Use an existing connection
 - Component List: tOracleConnection_2 - OracleOnLocal *
 - Schema: Repository | DB (ORACLE_SID):OracleOnLocal - WAGESIN | Edit schema
 - Table Name: "WAGESINFO" | ...
 - Query Type: Built-In | Guess Query | Guess schema
 - Query:

```
"SELECT
WAGESINFO.\"ID\",
WAGESINFO.NAME,
WAGESINFO.WAGE
FROM WAGESINFO
WHERE WAGE >
(SELECT avg(WAGE)
FROM WAGESINFO)
ORDER BY \"ID\"
"
```

 | ...

tLogRow_1

- Basic settings:**
 - Schema: Built-In | Edit schema | Sync columns
 - Mode:
 - ☐ Basic
 - ☒ Table (print values in cells of a table)
 - ☐ Vertical (each row is a key/value list)

tOracleCommit_1

- Basic settings:**
 - Component List: tOracleConnection_2 - OracleOnLocal *
 - ☒ Close Connection

Content with log4j

Talend

ORACLE-SCD

Sreenivas_Ram

Types of SCD

TYPE – I

- Overwrites old with new data.
- No history is maintained
- No need to change anything

TYPE – II

- Multiple records created
- Data history is maintained
- Can be use like start-date, end-date, or versioning or maintaining a flag

TYPE – III

- Multiple columns are created
- Limited data history is maintained (one level)
- Creating column like – Previous-Col, Current-Col

TYPE – IV

- Separate history table is maintained

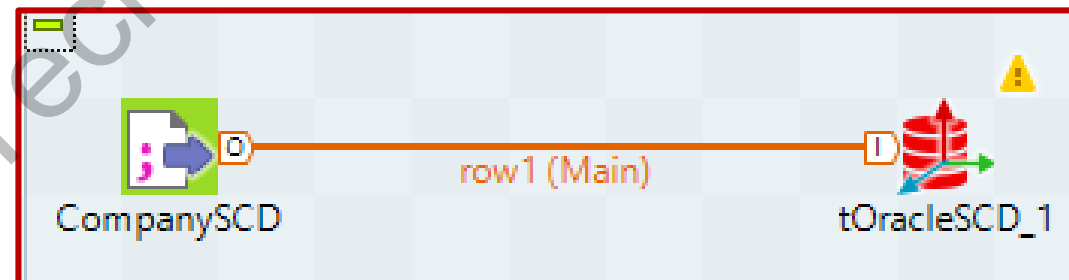
TYPE – VI

- Hybrid (1 + 2 + 3)

SCD-Type-I

- Change the data in the current record itself by overwriting
- Drag and drop CSV Data as source to CompanySCD
- Select InputFileDelimited details with **Compnay** Table in **ORCLESCD**
- In the SCD Editor – Map **CompanyID** as **SourceKey** and **CompanyName** as **type-1 field** changes

CompanyID	CompanyName
COM001	TCS
COM002	ONGC
COM003	Reliance
COM004	ITC
COM005	Coal India
COM006	Infosys
COM007	HDFC Ltd
COM008	SBI
COM009	Tata
COM010	Wipro



SCD Editor for Type-I Changes

filter

Unused

Source keys

CompanyID

Surrogate keys

name

CompanyKey

creation

Table max + 1

complement

Type 0 fields

Type 1 fields

CompanyName

Type 2 fields

Versioning

type	name	creation	
start	scd_start	Job start time	▼
end	scd_end	NULL	▼
<input type="checkbox"/> version	scd_version		
<input type="checkbox"/> active	scd_active		

Type 3 fields

current value	previous value

OK

Cancel

Changes made in Company Table

- Before and after execution

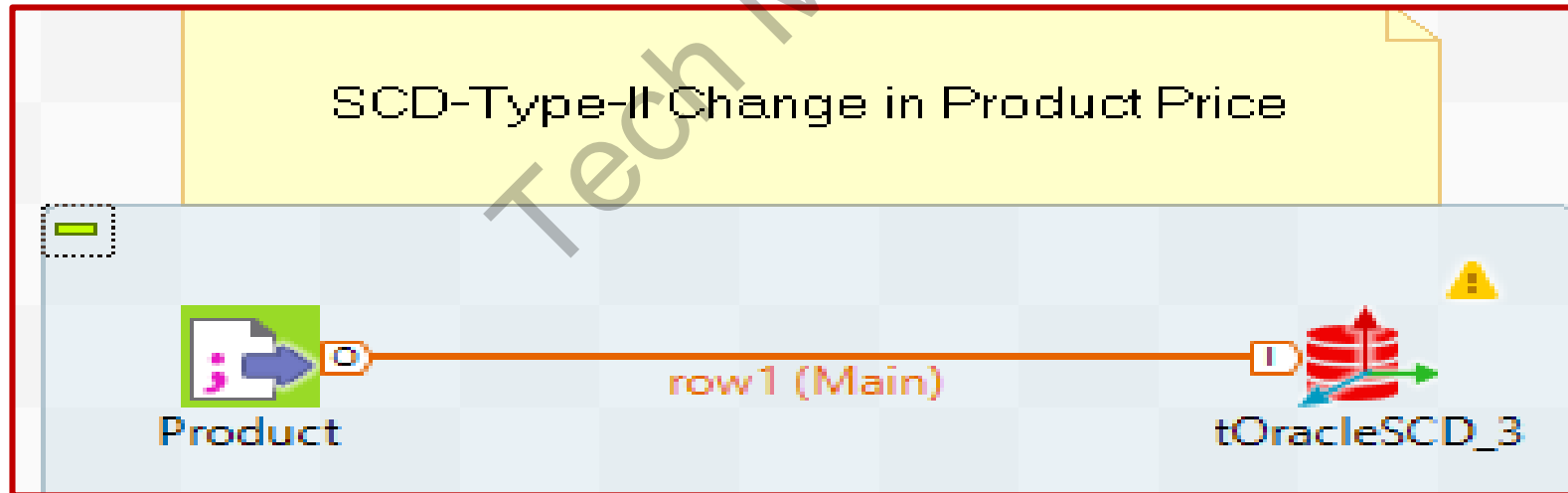
Compa...	CompanyID	COMPANYNAME	CompanyKey	CompanyID	COMPANYNAME
1	COM001	TCS	1	COM001	TCS
2	COM002	ONGC	2	COM002	ONGC
3	COM003	Reliance	3	COM003	Reliance
4	COM004	ITC	4	COM004	ITC Ltd
5	COM005	Coal India	5	COM005	Coal India
6	COM006	Infosys	6	COM006	Infosys
7	COM007	HDFC Ltd	7	COM007	HDFC Ltd
8	COM008	SBI	8	COM008	SBI
9	COM009	Tata	9	COM009	Tata Co Ltd
10	COM010	Wipro	10	COM010	Wipro

SCD – Type-II

Tech Mahindra

SCD-Type-II

- Multiple records created and the data history is maintained
- Drag and drop CSV Data as source - **Products**
- Select InputFileDelimited details with **DimProducts** Table in **ORACLESCD**
- In the SCD Editor – Map **ProductID** as **SourceKey** and ProductName, as type-0 field changes, **Price** as **Type-2 field changes** add **Productkey** as **Surrogate Key** Leave Start_Date and En_Date as it is.



SCD-Type-II ... contd...

- The original data of **DimProducts**

filter

Unused

End_Date

Start_Date

Source keys

ProductCode

Surrogate keys

nameProductKey

creationTable max + 1

complement

Type 0 fields

ProductName

Type 1 fields

Type 2 fields

Price

Versioning

type	name	creation	com
start	Start_Date	Job start time	▼
end	End_Date	NULL	▼
<input type="checkbox"/> version	scd_version		
<input type="checkbox"/> active	scd_active		

Type 3 fields

current value

previous value

OK

Cancel

ProductKey	ProductCode	ProductName	Price	Start_Date	End_Date
1	PROD001	LG Nexus 5	22000	2016-08-18	NULL
2	PROD002	Samsung Galaxy Note 2	15000	2016-08-18	NULL
3	PROD003	Nokia Lumia 1111	9999	2016-08-18	NULL
4	PROD004	Sony Xperia Z	20000	2016-08-18	NULL
5	PROD005	HTC One Mini	9999	2016-08-18	NULL
6	PROD006	Samsung Galaxy Core Duos	9999	2016-08-18	NULL
7	PROD007	Nokia Lumia 625	9999	2016-08-18	NULL
8	PROD008	Sony Xeria L	9888	2016-08-18	NULL
9	PROD009	Micromax Canvas	5555	2016-08-18	NULL
10	PROD010	HTC One V	8888	2016-08-18	NULL

Changes in Price (csv)

ProductCode	ProductName	Price	Start_Date	End_Date
PROD001	LG Nexus 5	20000	10/15/2013	NULL
PROD002	Samsung Galaxy	12000	10/15/2013	NULL
PROD003	Nokia Lumia 11	9999	10/15/2013	NULL
PROD004	Sony Xperia Z	20000	10/15/2013	NULL
PROD005	HTC One Mini	9000	10/15/2013	NULL
PROD006	Samsung Galaxy	9999	10/15/2013	NULL
PROD007	Nokia Lumia 62	9000	10/15/2013	NULL
PROD008	Sony Xeria L	8000	10/15/2013	NULL
PROD009	Micromax Canv	5555	10/15/2013	NULL
PROD010	HTC One V	8888	10/15/2013	NULL

SCD-Type-II ... contd...

- Changes reflected in **DimProducts** table [MSMSQL]

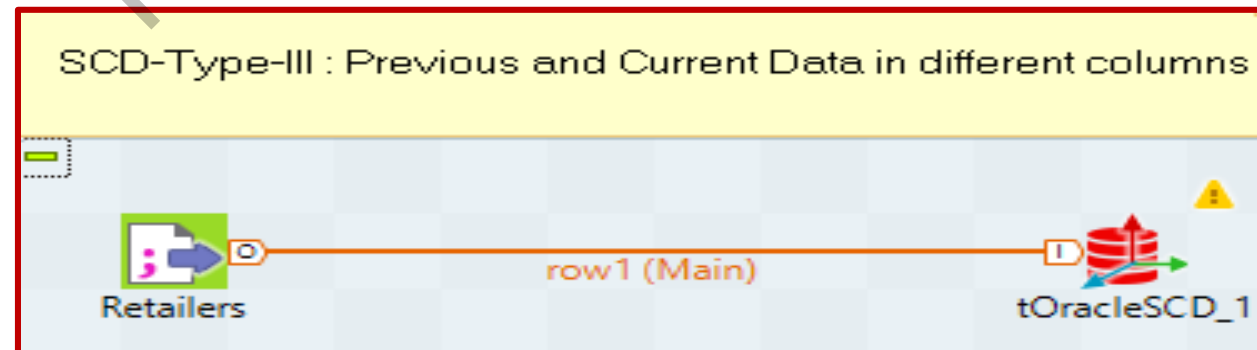
ProductKey	ProductCode	ProductName	Price	Start_Date	End_Date
1	PROD001	LG Nexus 5	22000	2016-08-18	2016-08-18
2	PROD002	Samsung Galaxy Note 2	15000	2016-08-18	2016-08-18
3	PROD003	Nokia Lumia 1111	9999	2016-08-18	NULL
4	PROD004	Sony Xperia Z	20000	2016-08-18	NULL
5	PROD005	HTC One Mini	9999	2016-08-18	2016-08-18
6	PROD006	Samsung Galaxy Core Duos	9999	2016-08-18	NULL
7	PROD007	Nokia Lumia 625	9999	2016-08-18	2016-08-18
8	PROD008	Sony Xeria L	9888	2016-08-18	2016-08-18
9	PROD009	Micromax Canvas	5555	2016-08-18	NULL
10	PROD010	HTC One V	8888	2016-08-18	NULL
11	PROD001	LG Nexus 5	20000	2016-08-18	NULL
12	PROD002	Samsung Galaxy Note 2	12000	2016-08-18	NULL
13	PROD005	HTC One Mini	9000	2016-08-18	NULL
14	PROD007	Nokia Lumia 625	9000	2016-08-18	NULL
15	PROD008	Sony Xeria L	8000	2016-08-18	NULL

SCD – Type-III

Tech Mahindra

SCD – Type-III

- To maintain history
 - Using a date (active date)
 - Versioning
 - IsActive (flag)
- Design the job as follows:
- Drag Retailers (csv) – **tFileInputDelimited** and **tORACLESCD**
- Make the following changes in SCD editor
- **RetailerID** – SourceKey
- **Company** and **RName** in Type-0 – No changes
- **RetailerKey** – Surrogate Key
- Add **Start_Date** and **End_Date** in Versioning in place of original columns



SCD – Type-III ... contd.

- SCD Designer

filter

Unused

RetailerID

Source keys

nameRetailerKey

creationTable max + 1

Surrogate keys

Country

RName

Type 0 fields

Type 1 fields

Type 2 fields

type

start

end

☐ version

☐ active

scd_start

scd_end

scd_version

scd_active

Job start time

NULL

complement

Type 3 fields

current value

previous value

State

Prev_State

OK

Cancel

SCD – Type-III ... contd.

- Original Source Data

RetailerID	Name	State	Country
RET0001	Reliance	Maharastra	India
RET0002	Pantaloons	Maharastra	India
RET0003	Provogue	Maharastra	India
RET0004	Shopper Stp	Maharastra	India
RET0005	ITC	West Bengal	India
RET0006	Trent	Maharastra	India
RET0007	McDonalds	Illinois	
RET0008	Aditya Birla	Maharast	
RET0009	Titan	Karnataka	
RET0010	Kewel Kiran	Maharast	

- Updated data

RetailerKey	RetailerID	RName	State	Country	Prev_State
1	RET0001	Reliance	Maharastra	India	NULL
2	RET0002	Pantaloons	Telangana	India	Maharastra
3	RET0003	Provogue	Maharastra	India	NULL
4	RET0004	Shopper Stp	Andhra Pradesh	India	Maharastra
5	RET0005	ITC	West Bengal	India	NULL
6	RET0006	Trent	Maharastra	India	NULL
7	RET0007	McDonalds	Illinois	USA	NULL
8	RET0008	Aditya Birla	Maharastra	India	NULL
9	RET0009	Titan	Kamataka	India	NULL
10	RET0010	Kewel Kiran	Maharastra	India	NULL

Talend

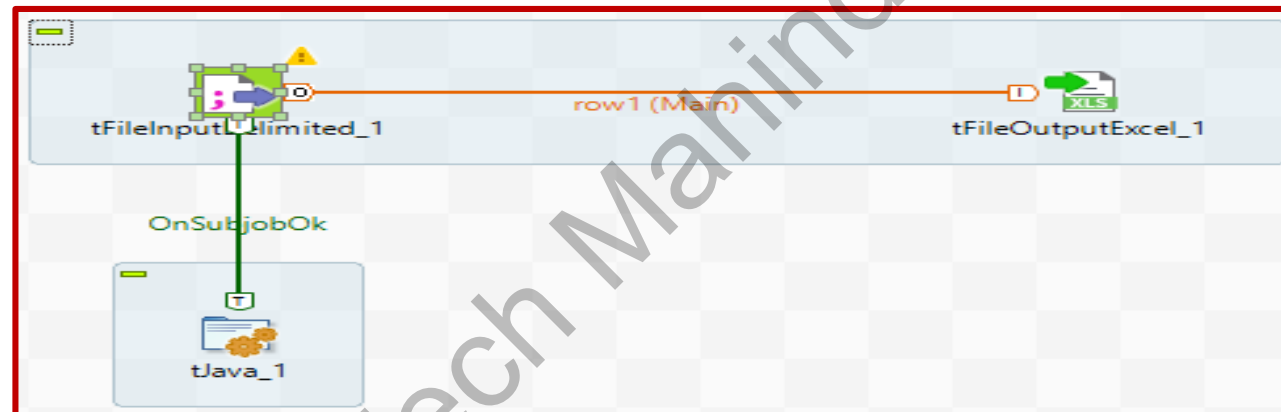
Printing out a **variable content**

using **tJava** component

Sreenivas_Ram

tJava Ex:

- The Job aims at printing out the number of lines being processed using a Java command and the global variable
- Design the job as follows: - using **tFileInputDelimited**, **tFileOutputExcel** and **tJava** components.



- Connect the **tFileInputDelimited** to the **tFileOutputExcel** using a **Row Main** connection. The content from a delimited txt file will be passed on through the connection to an xls-type of file without further transformation
- Then connect the **tFileInputDelimited** component to the **tJava** component using a **Trigger > On Subjob Ok** link

Properties of Components

- **tFileInputDelimited** : 1. In File name field, enter the path and file name
- 2. Asimple text file made of **two columns: Names** and their **Emails**
- Click the **Edit Schema** button, and set schema accordingly
- **tFileOutputExcel** :
- Enter File Name an define schema.

tFileOutputExcel_1

Property Type: Built-In

☐ Write excel2007 file format(xlsx)

☐ Use Output Stream

File Name: "C:/Output/Email_List.xls"

Sheet name: "Email"

☒ Include header

☐ Append existing file

☐ Is absolute Y pos.

Font: Default

☒ Define all columns auto size

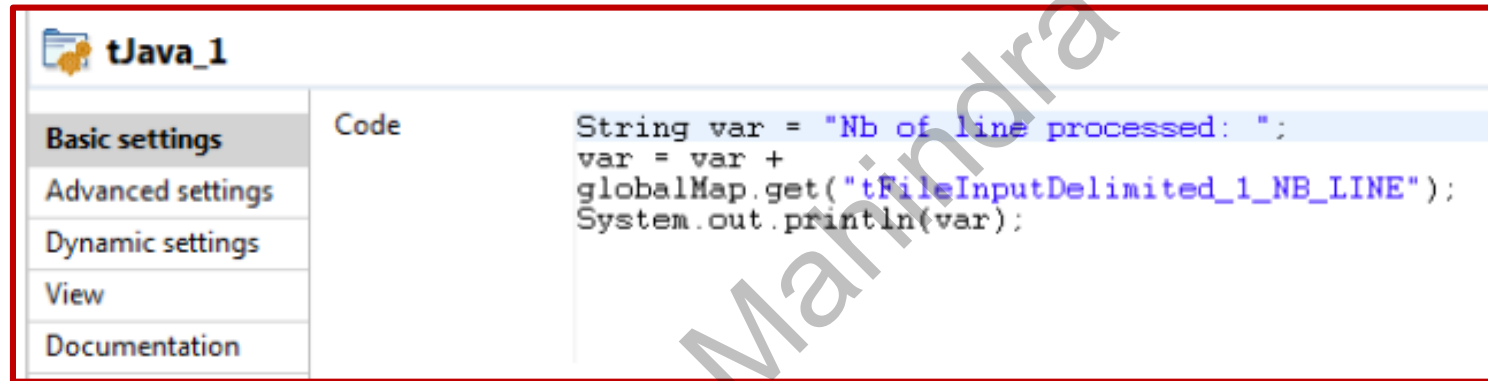
Define column auto size

Column	Auto size
Names	<input checked="" type="checkbox"/>
Emails	<input checked="" type="checkbox"/>

Schema: Built-In Edit schema Sync columns

Properties of Components ... contd ...

- In the code area :



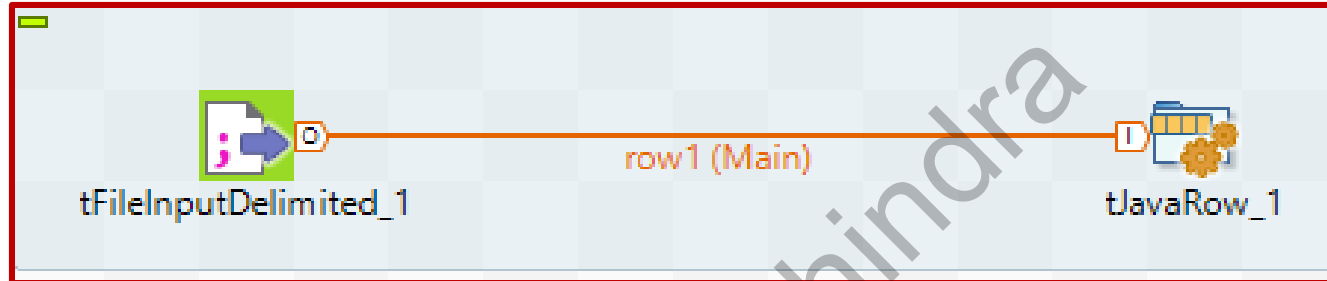
- Execute the job

Talend
Row-wise processing using
tJavaRow Component

Sreenivas_Ram

tJavaRow

- We can process each row of the flow using Java code



Properties - tFileInputDelimited

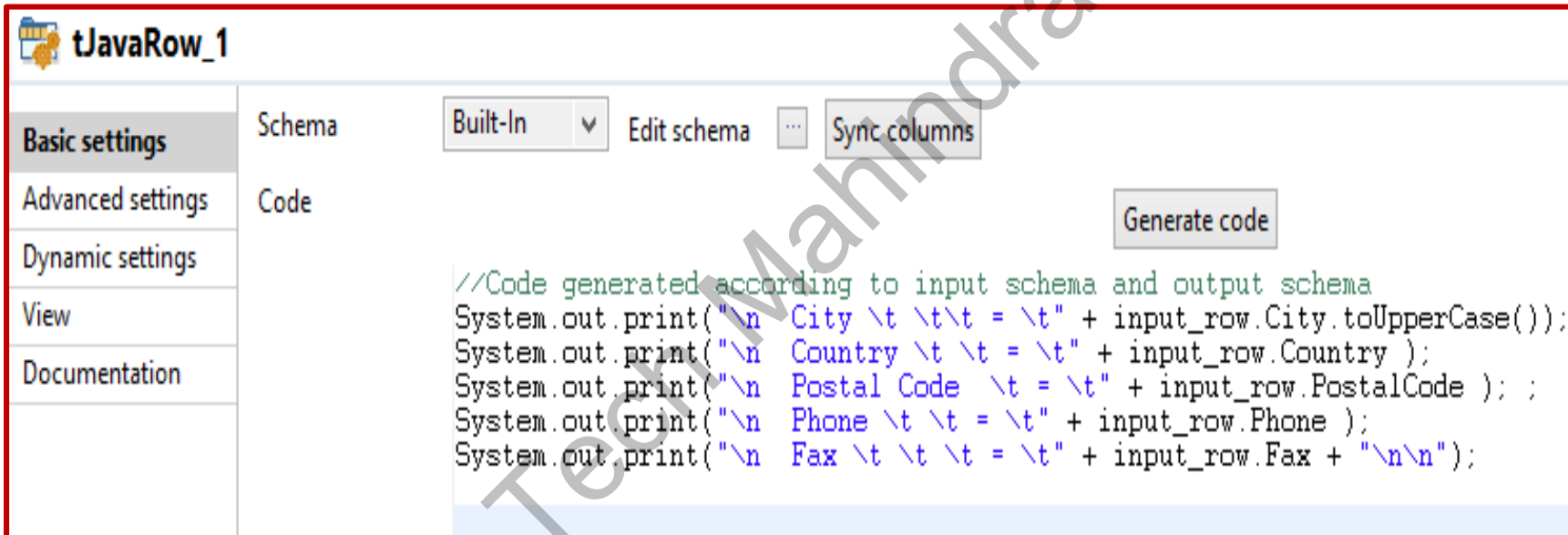
- We use column schema - as listed

The screenshot shows the 'Basic settings' tab of the 'tFileInputDelimited_1' properties dialog. The 'Property Type' is set to 'Repository' and 'DELIM:CityCountryList'. The 'File name/Stream' is 'C:/TalendMyData/Source/CityCountryList.csv'. The 'CSV Row Separator' is 'LF("\n")' and the 'Field Separator' is ','. The 'CSV options' are checked, with 'Escape char' set to '""' and 'Text enclosure' set to '""'. The 'Header' is '1' and the 'Footer' is '0'. The 'Schema' is 'Repository' and 'DELIM:CityCountryList - metadata'. There are checkboxes for 'Skip empty rows', 'Uncompress as zip file', and 'Die on error'.

City
Country
Fax
Phone
PostalCode

Properties

- **tJavaRow** : Enter the following **Java Code** - to read multiple lines of input and print the same



The screenshot shows the 'tJavaRow_1' configuration window. The left sidebar has the following tabs: 'Basic settings' (selected), 'Advanced settings', 'Dynamic settings', 'View', and 'Documentation'. The main area is divided into two sections: 'Schema' and 'Code'. The 'Schema' section has a dropdown menu set to 'Built-In', and buttons for 'Edit schema' and 'Sync columns'. The 'Code' section has a 'Generate code' button and a text area containing the following Java code:

```
//Code generated according to input schema and output schema
System.out.print("\n City \t \t\t = \t" + input_row.City.toUpperCase());
System.out.print("\n Country \t \t = \t" + input_row.Country );
System.out.print("\n Postal Code \t = \t" + input_row.PostalCode ); ;
System.out.print("\n Phone \t \t = \t" + input_row.Phone );
System.out.print("\n Fax \t \t \t = \t" + input_row.Fax + "\n\n");
```

- Execute the Job

Talend

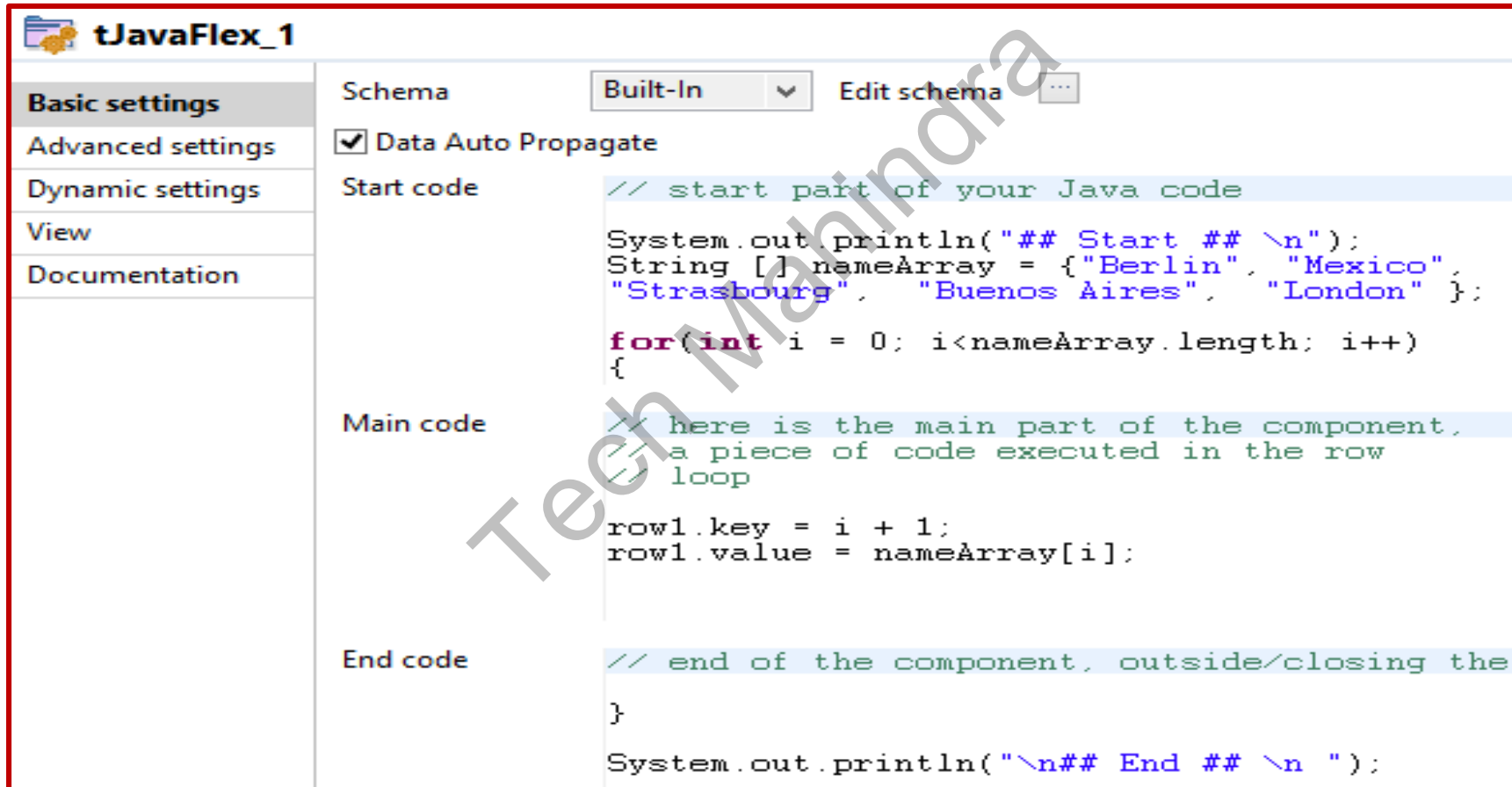
Multi-part Java Code processing using

tJavaFlex Component

Sreenivas_Ram

Code to process in three setps

- Sample code :



The screenshot shows the tJavaFlex_1 IDE interface. On the left is a sidebar with a tree view containing: Basic settings (selected), Advanced settings, Dynamic settings, View, and Documentation. The main area is divided into two panes. The top pane shows 'Schema' set to 'Built-In' with an 'Edit schema' button. Below it, 'Data Auto Propagate' is checked. The bottom pane is divided into three sections: 'Start code', 'Main code', and 'End code'. Each section contains Java code snippets. A large 'Tech Mahindra' watermark is visible diagonally across the code area.

```
// start part of your Java code

System.out.println("## Start ## \n");
String [] nameArray = {"Berlin", "Mexico",
"Strasbourg", "Buenos Aires", "London" };

for(int i = 0; i<nameArray.length; i++)
{

// here is the main part of the component,
// a piece of code executed in the row
// loop

row1.key = i + 1;
row1.value = nameArray[i];

// end of the component, outside/closing the
}

System.out.println("\n## End ## \n ");
```

Thank you

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.

Tech
Mahindra