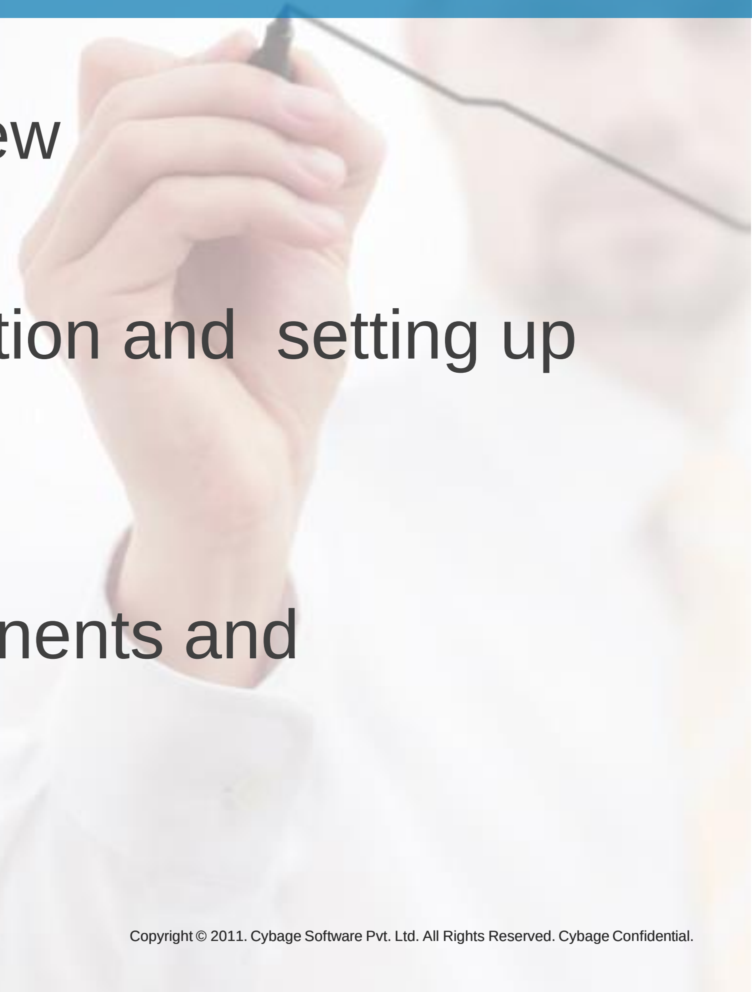
www.cybage.com

Talend Data Integration

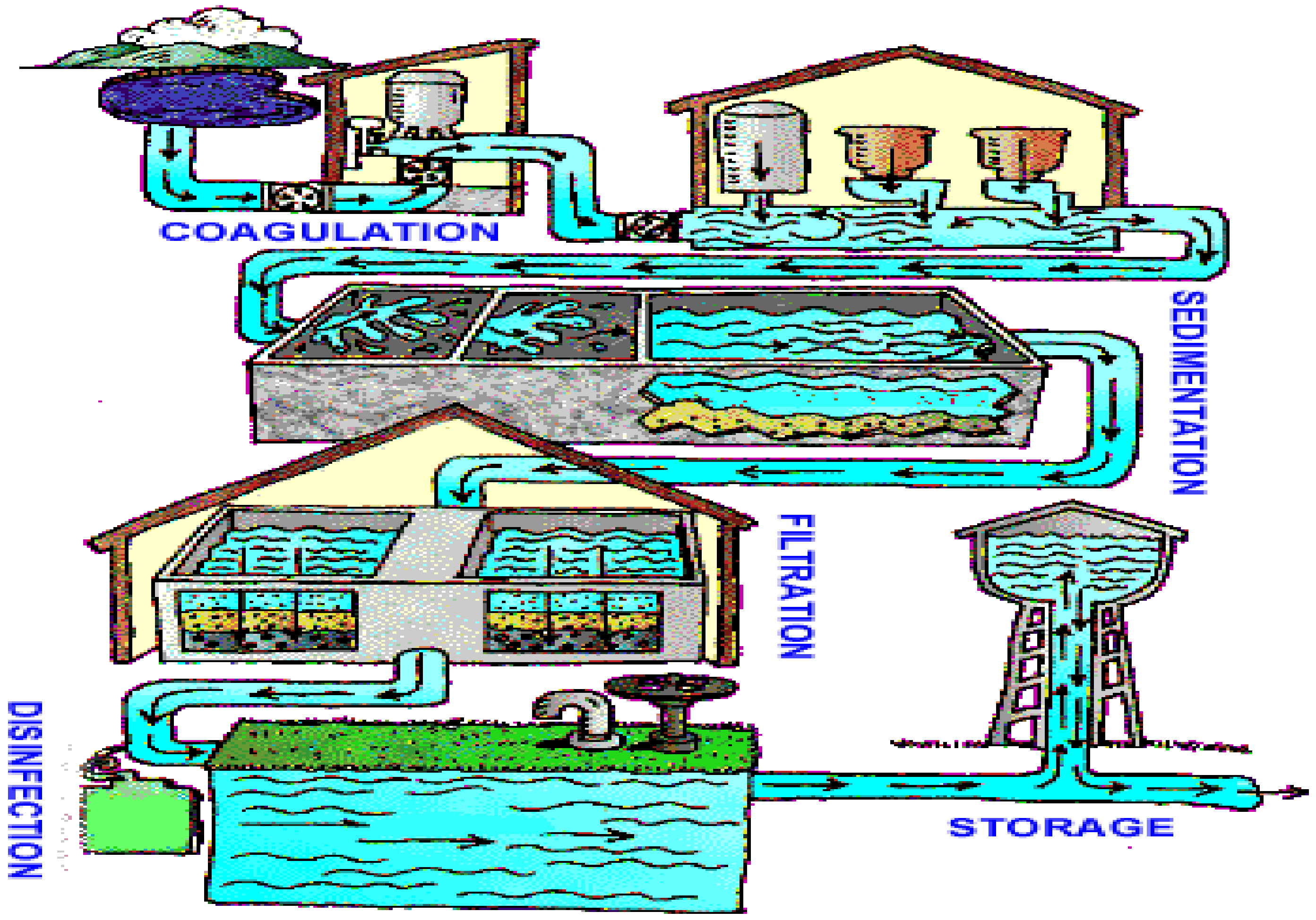
Presented by : Prashant Tikone

Day 1

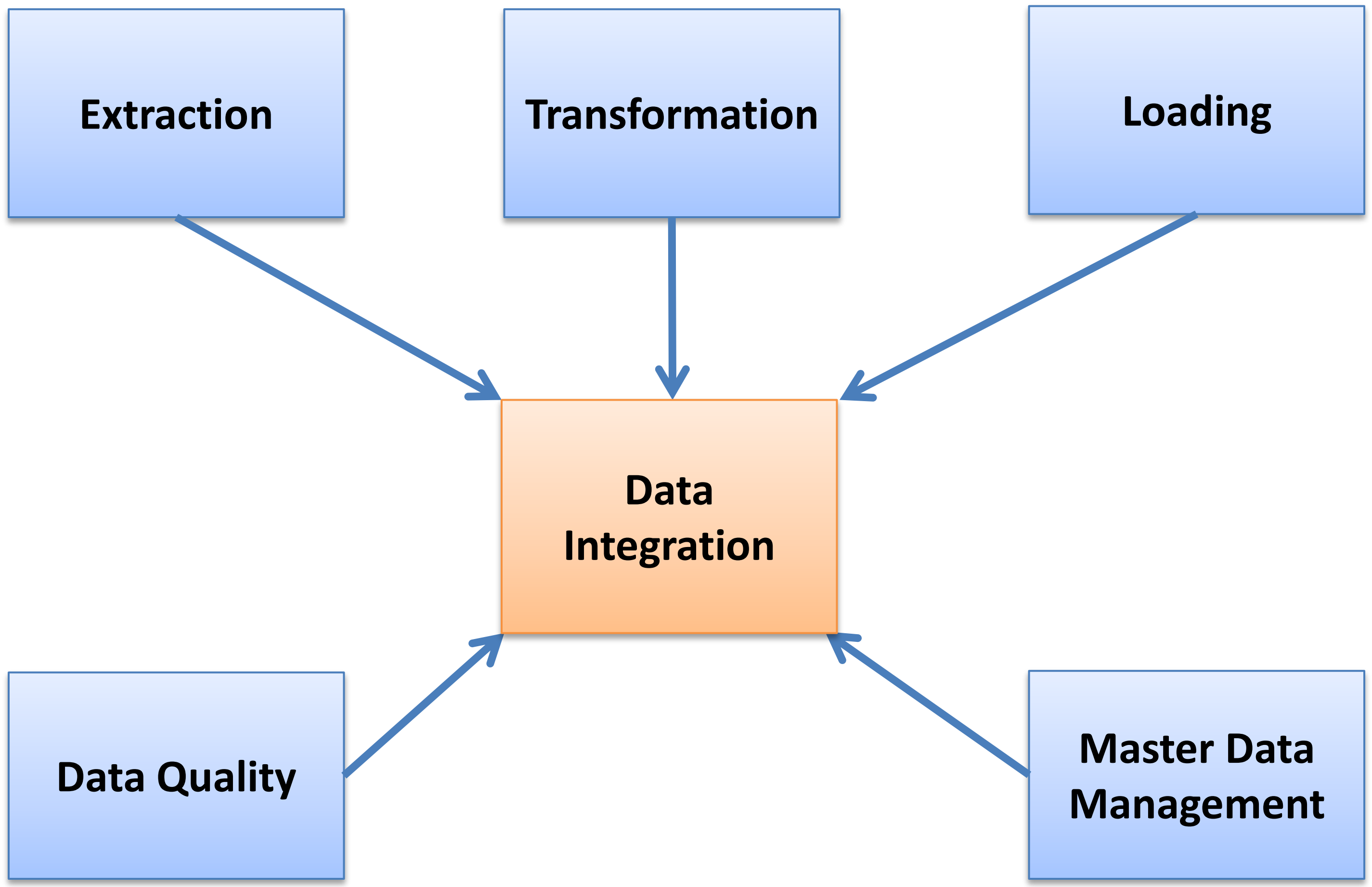


- Data Integration Overview
- Talend Platform
- User Interface – Installation and setting up
- Business Modeling
- Designing Talend Job
- Data Integration Components and Connections
- Demos

Data Integration – Filter /Cleanse/Store



Data Integration



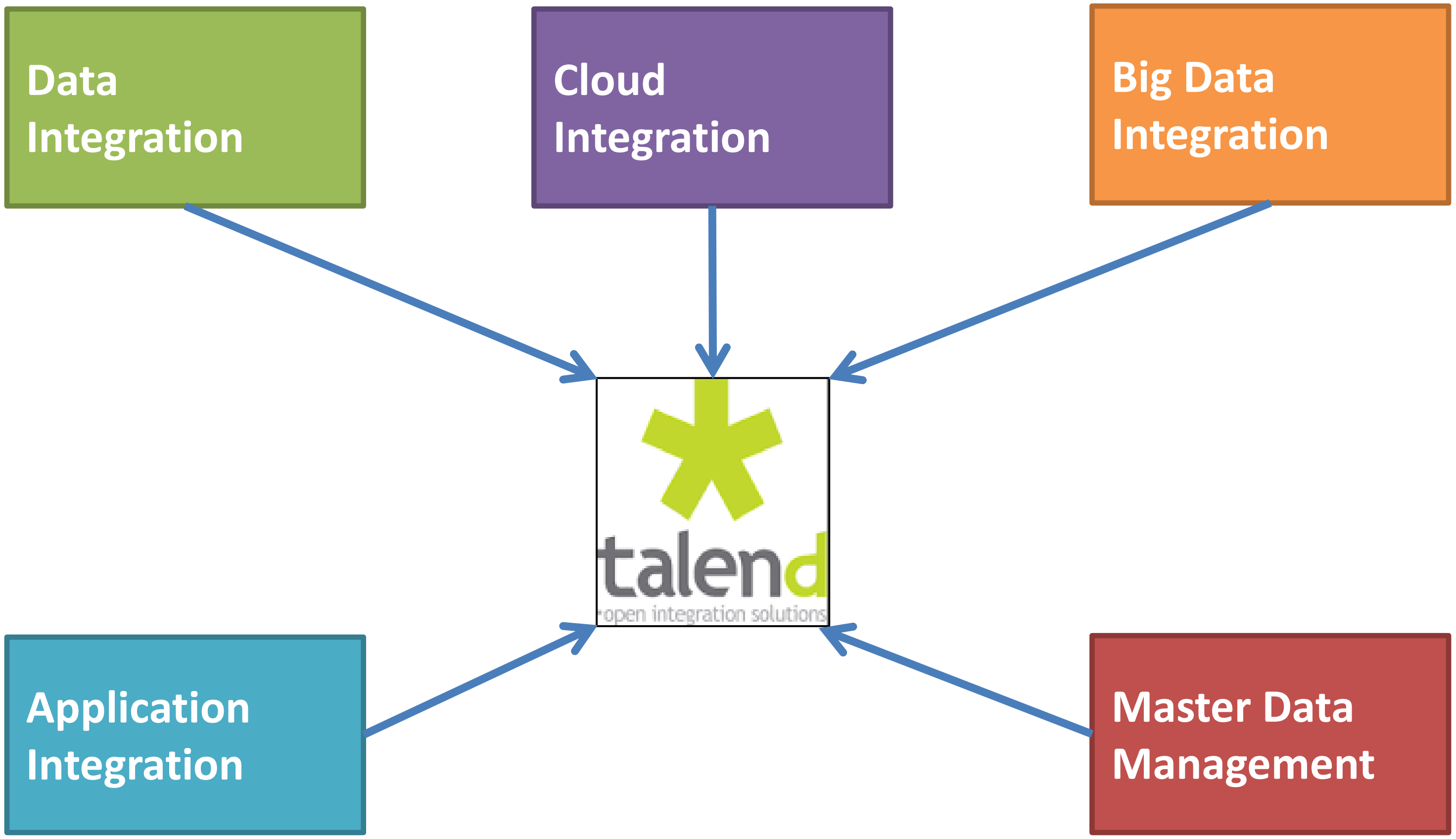
Data Integration

- Combining data residing in different sources and providing users with a **unified view** of these data.

- Create **Sharable data** collection to solve commercial or scientific problems.

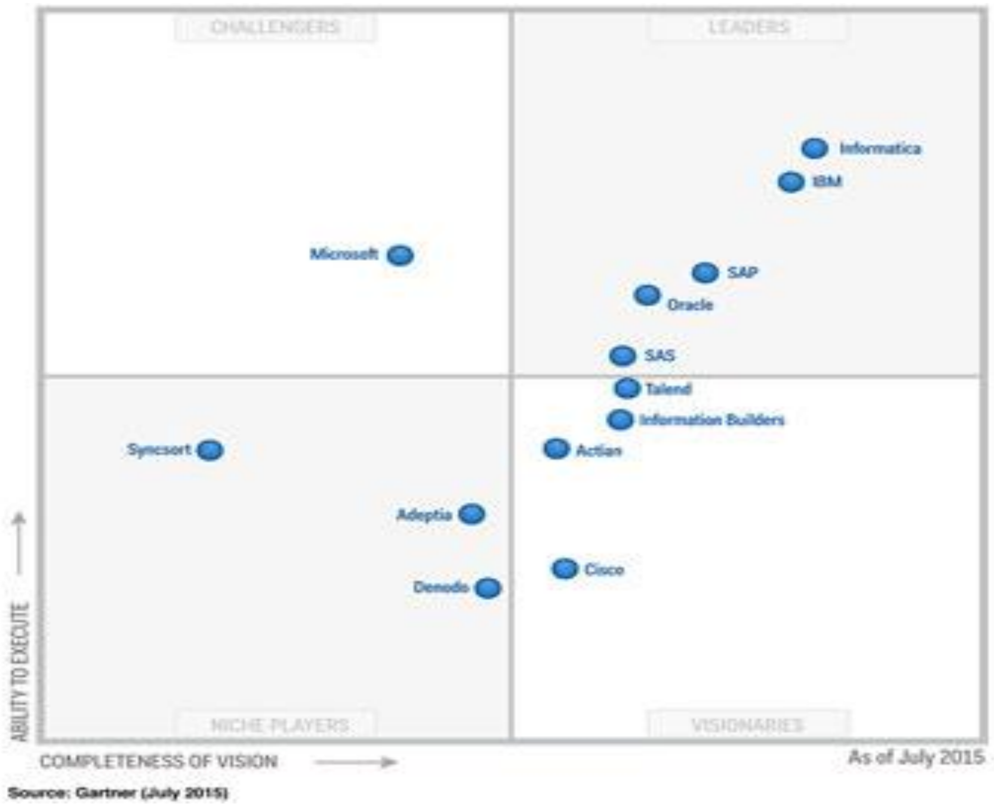
- Process of collecting of disparate data sets for **meta analysis**.

Talend Data Integration Platform



Talend Data Integration - Comparison

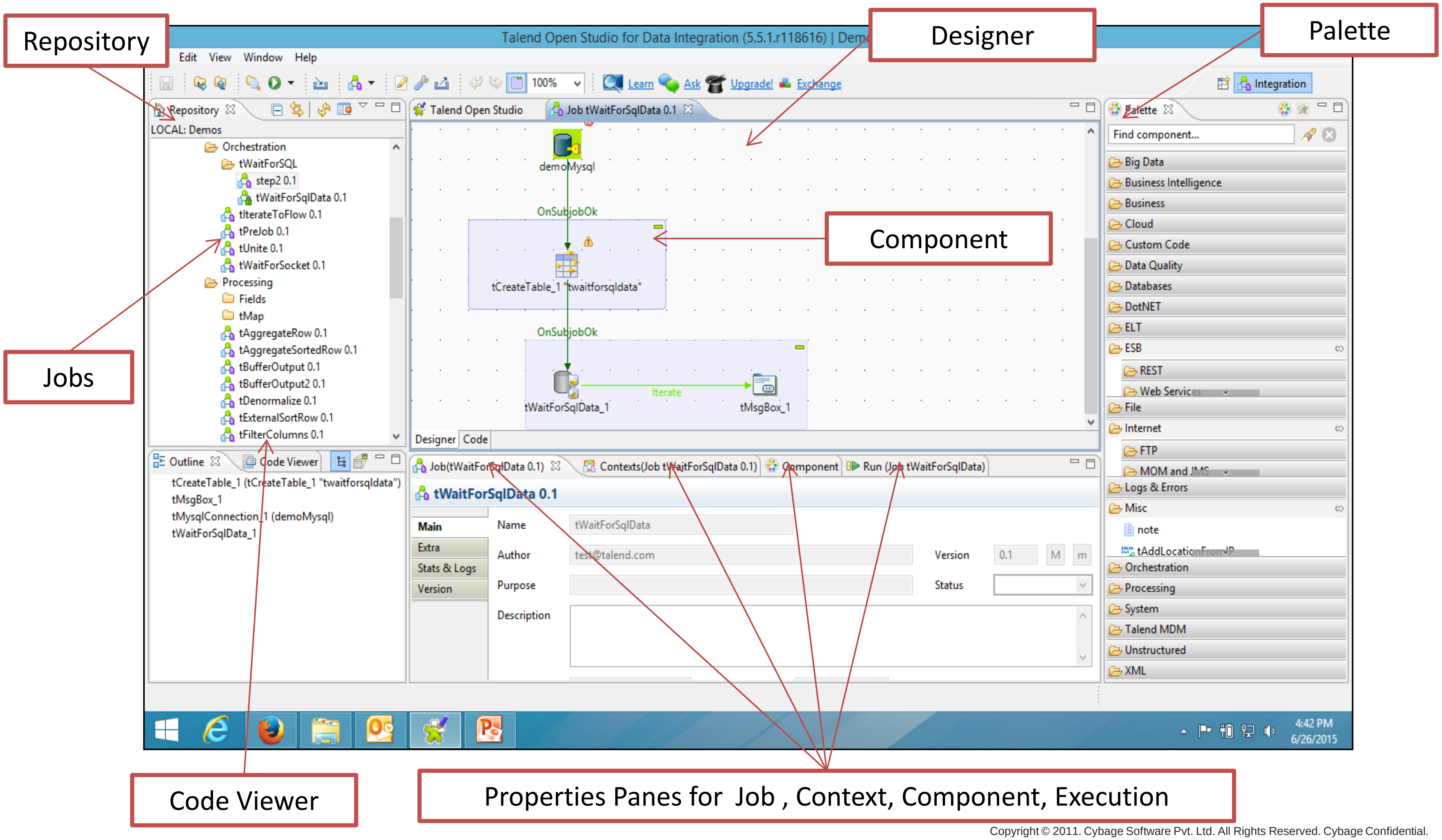
Gartner, Magic Quadrant for Data Integration Tools, 2015



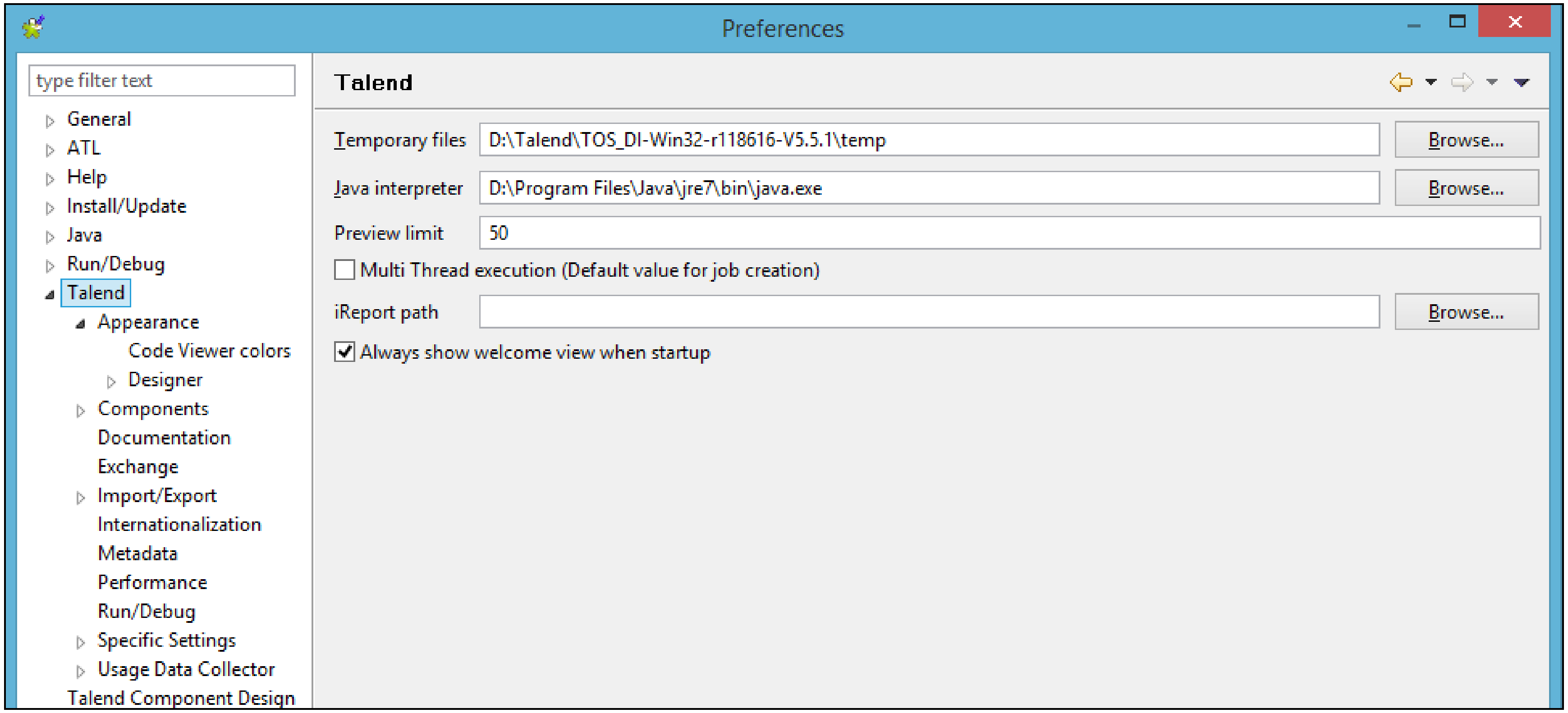
Data Integration Offerings

	Open Studio for Data Integration	Enterprise Data Integration	Platform for Data Management
License	Apache	Subscription	Subscription
Platform Capabilities	800+ Components & Connectors	+ Modeling, Testing, Sharing & Debugging	+ Repository Manager & Visual Mapping
Collaborate & Manage	----	Manage Administration, Deployment, & Automate Tasks	+ High availability, load balancing, and failover
Data Quality	----	---	Cleansing, Profiling, Stewardship
Support	TalendForge Community, Help Center access	+ Guaranteed Response Times, Web & Email Support	+ Phone Support

Talend Open Studio - GUI



Setting Preferences



Preferences

type filter text

- General
- ATL
- Help
- Install/Update
- Java
- Run/Debug
- Talend
  - Appearance
    - Code Viewer colors
    - Designer
  - Components
  - Documentation
  - Exchange
  - Import/Export
  - Internationalization
  - Metadata
  - Performance
  - Run/Debug
  - Specific Settings
  - Usage Data Collector
- Talend Component Design

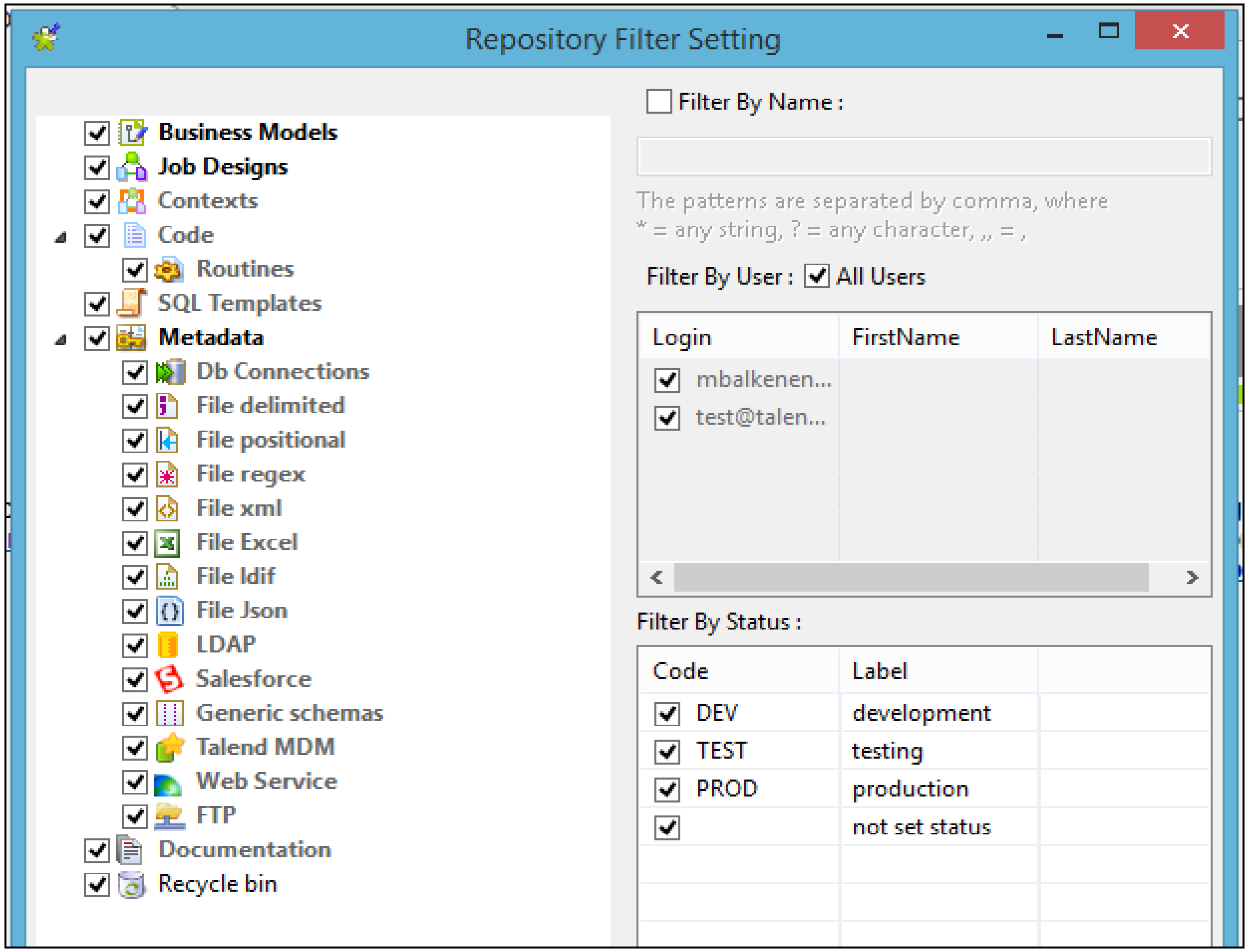
Talend

Temporary files	D:\Talend\TOS_DI-Win32-r118616-V5.5.1\temp	Browse...
Java interpreter	D:\Program Files\Java\jre7\bin\java.exe	Browse...
Preview limit	50	
iReport path		Browse...

☐ Multi Thread execution (Default value for job creation)

☑ Always show welcome view when startup

Setting up Repository


Repository Filter Setting

- [✓] Business Models
- [✓] Job Designs
- [✓] Contexts
- [✓] Code
  - [✓] Routines
- [✓] SQL Templates
- [✓] Metadata
  - [✓] Db Connections
  - [✓] File delimited
  - [✓] File positional
  - [✓] File regex
  - [✓] File xml
  - [✓] File Excel
  - [✓] File ldif
  - [✓] File Json
  - [✓] LDAP
  - [✓] Salesforce
  - [✓] Generic schemas
  - [✓] Talend MDM
  - [✓] Web Service
  - [✓] FTP
- [✓] Documentation
- [✓] Recycle bin

[ ] Filter By Name :

The patterns are separated by comma, where
* = any string, ? = any character, ,, = ,

Filter By User : [✓] All Users

Login	FirstName	LastName
✓ mbalkenen...		
✓ test@talen...		

Filter By Status :

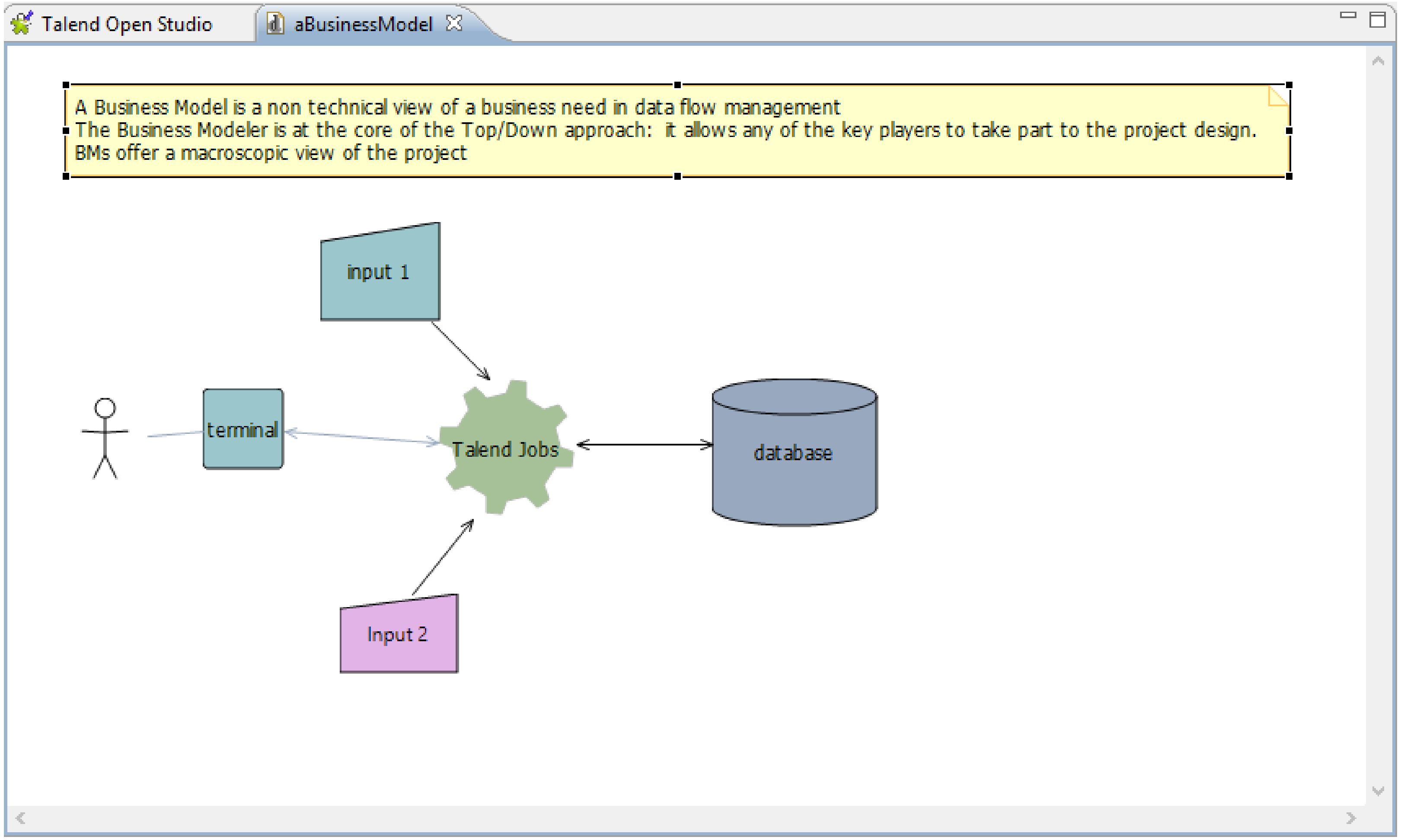
Code	Label
✓ DEV	development
✓ TEST	testing
✓ PROD	production
✓	not set status

Business Modeling

TOS offers Business Modeling objects to be used for Data Integration perspective.

- Draw business needs
- Create and assign numerous repository items to your model objects
- Define the business model properties of your model objects.

Business Modeling



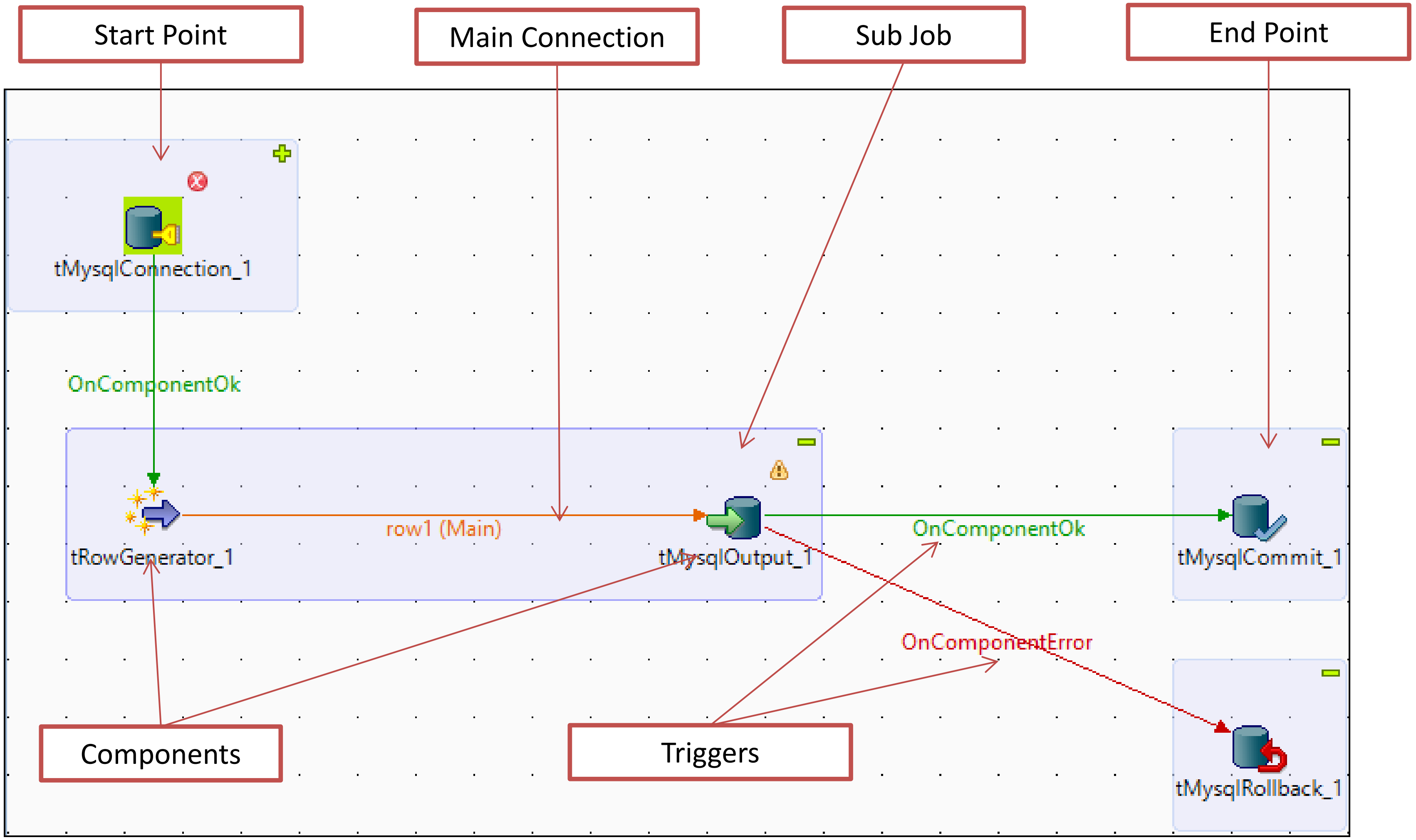
A Business Model is a non technical view of a business need in data flow management
The Business Modeler is at the core of the Top/Down approach: it allows any of the key players to take part to the project design.
BMs offer a macroscopic view of the project

Talend Job

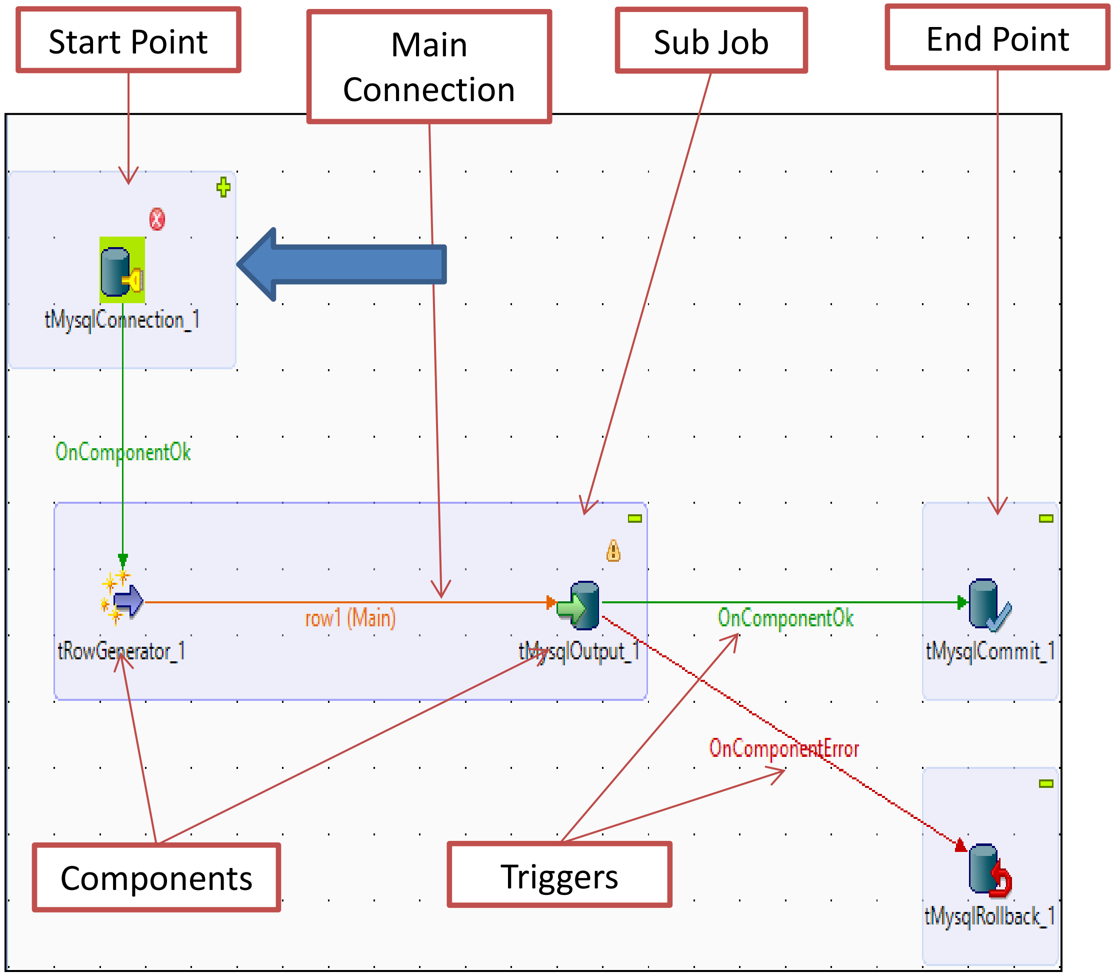
It is the heart of Talend Studio, the **Jobs** is meant to -

- Store all the metadata you need for graphically describing the jobs
- Assembly of components, connectors, parameters, colors and presentation stuff.

Talend Job Overview



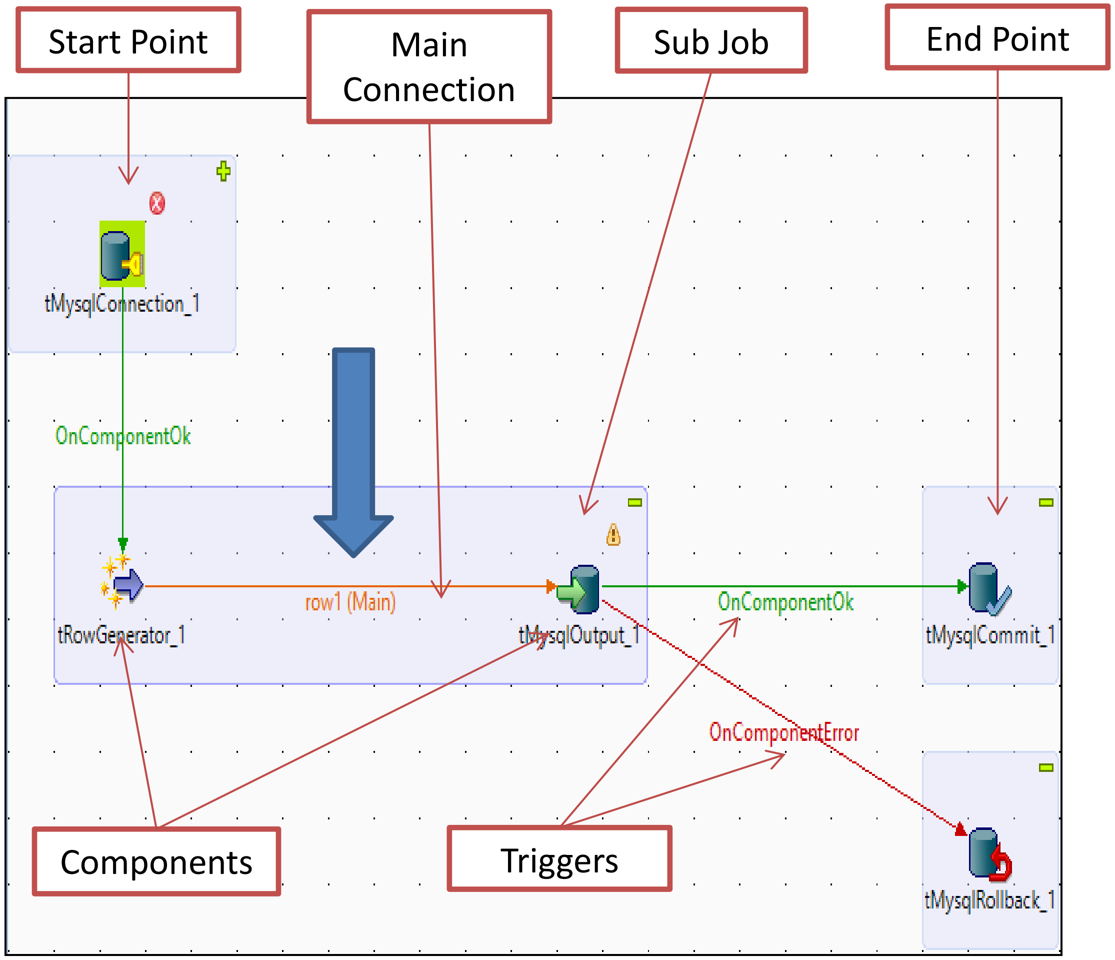
Talend Job



Start Point – The starting point component of a sub job is the one with a **green** background

- Job could have multiple start points
- The execution order could be unpredictable for jobs with multiple start points.

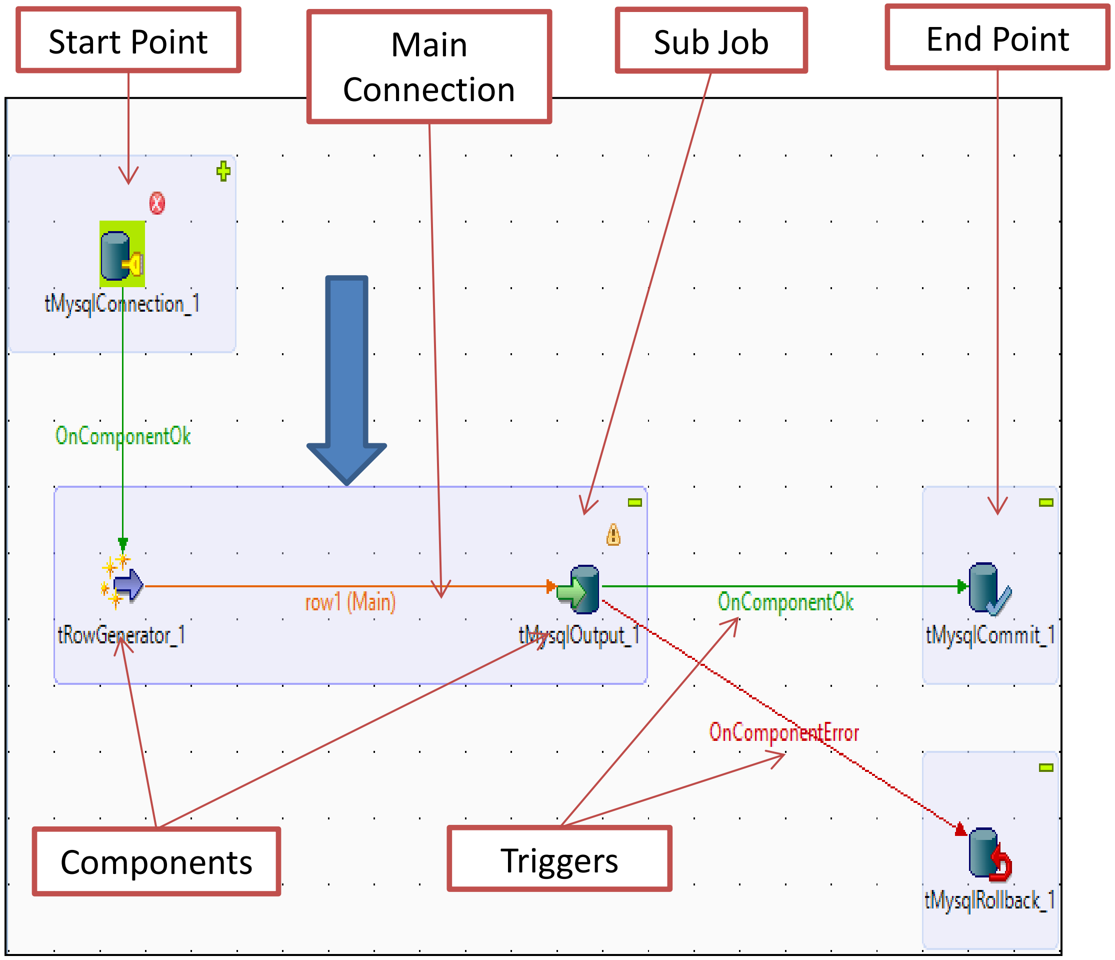
Talend Job



Main Connection The Main connections dictate the data flow.

- They move data between components
- Data is measured as per row / tuple basis.

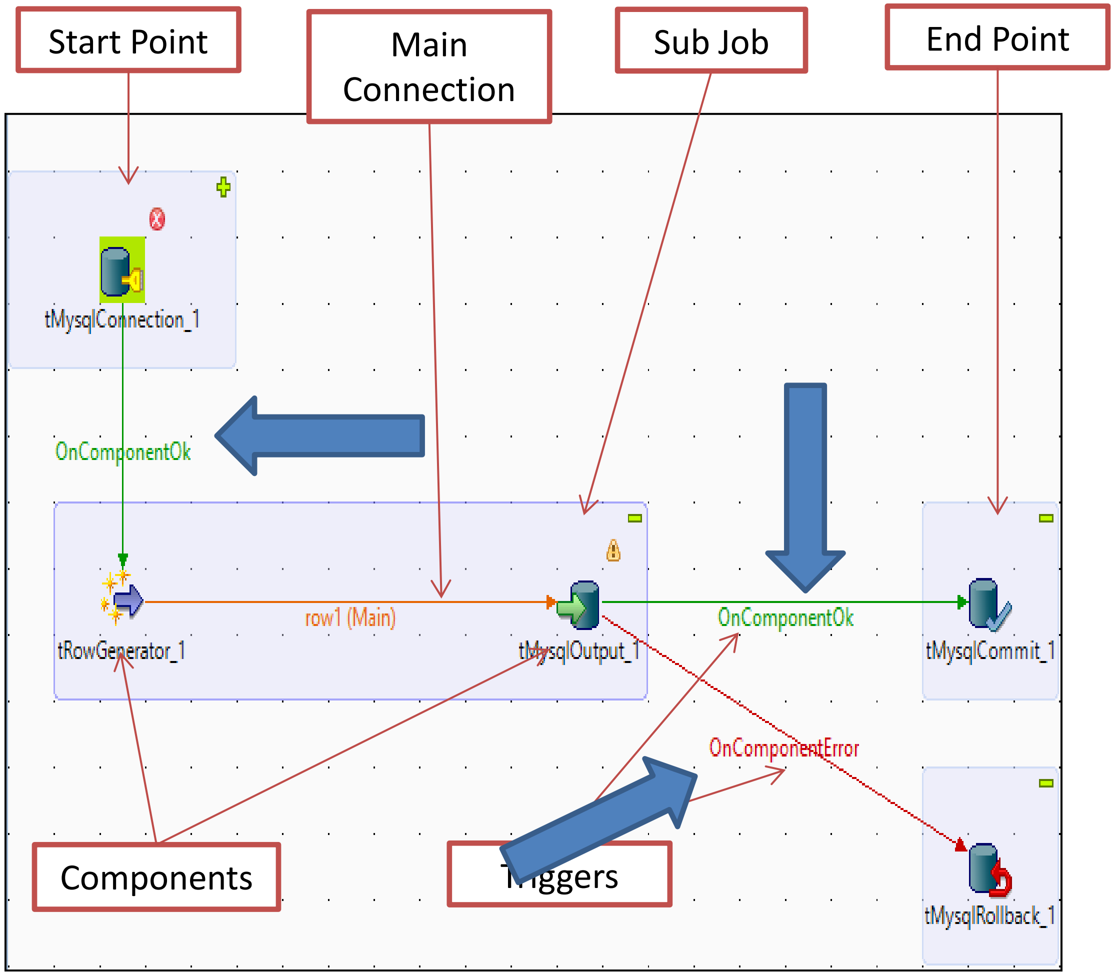
Talend Job



SubJob - A set of connected components all enclosed by a light-blue background.

You can have as many subjobs you need in a given job

Talend Job

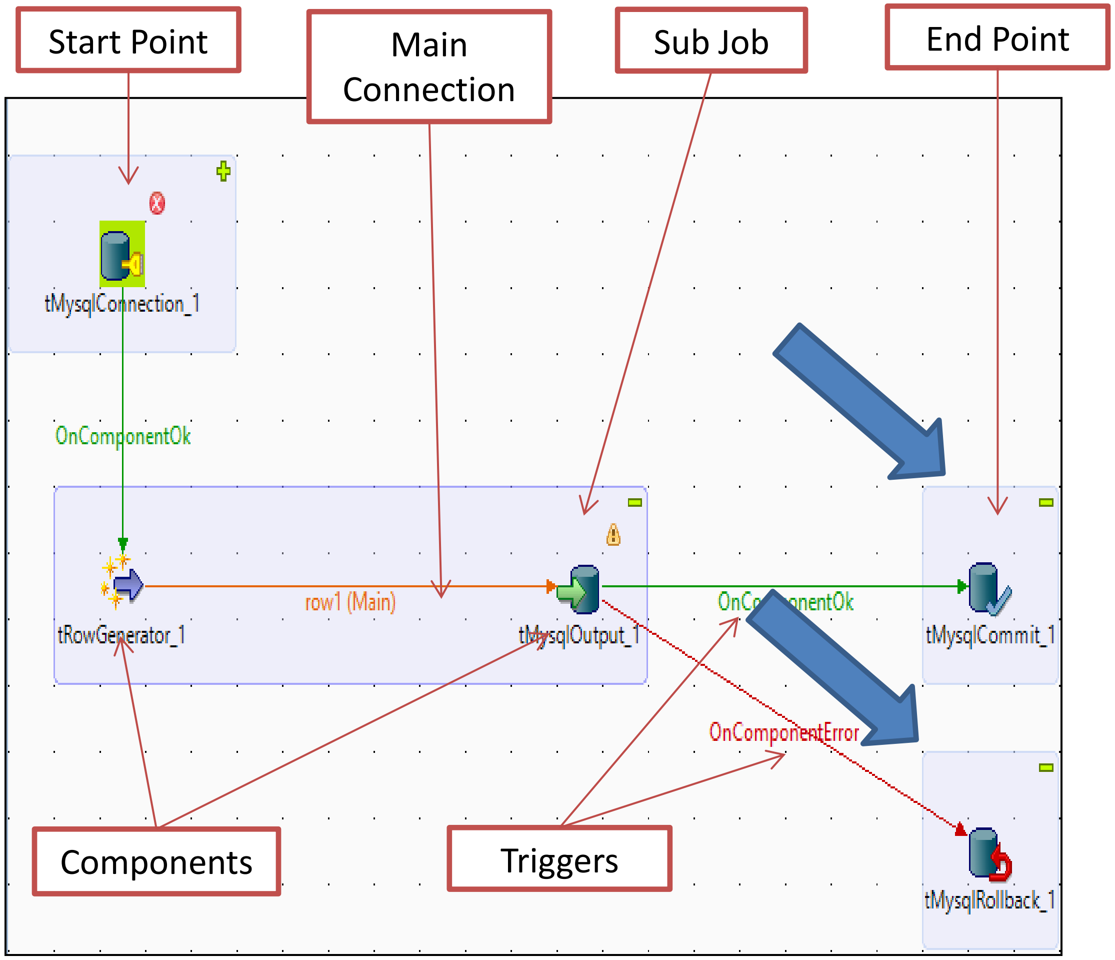


Triggers – They work as signaling mechanism between components.

There are two basic types.

- **Sub Job Triggers**
- **Component Triggers**
- Go/No-Go signals for the execution of one or more subjobs.
- Used for connecting subjobs.

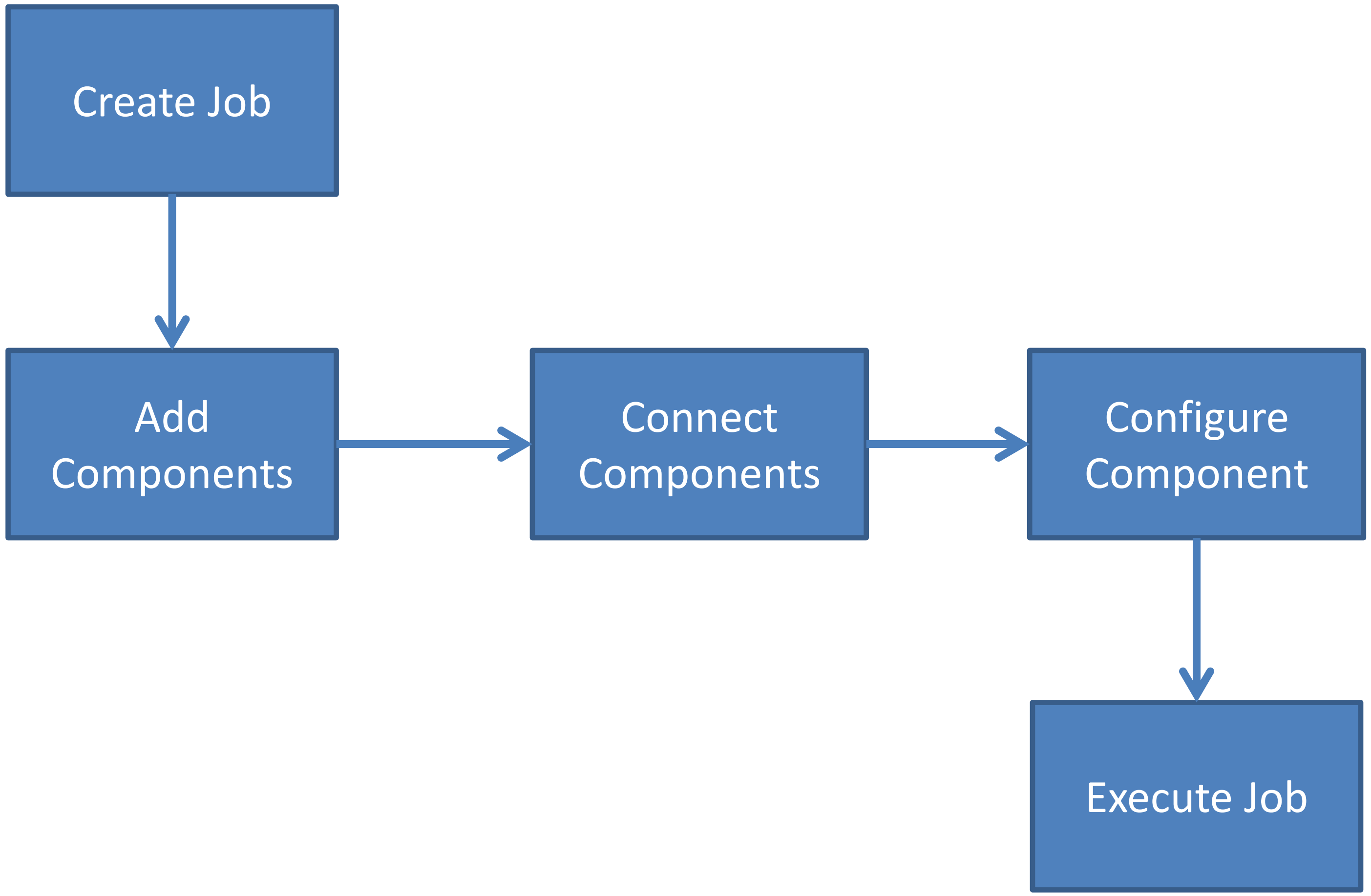
Talend Job



Endpoints – The component that has no outgoing connection forms an end point.

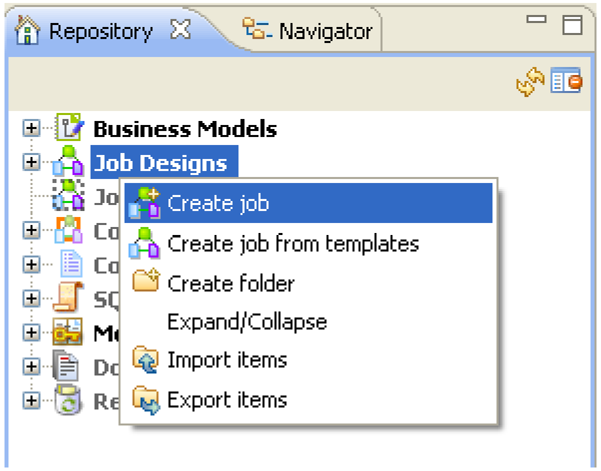
Job can have as many endpoints as needed.

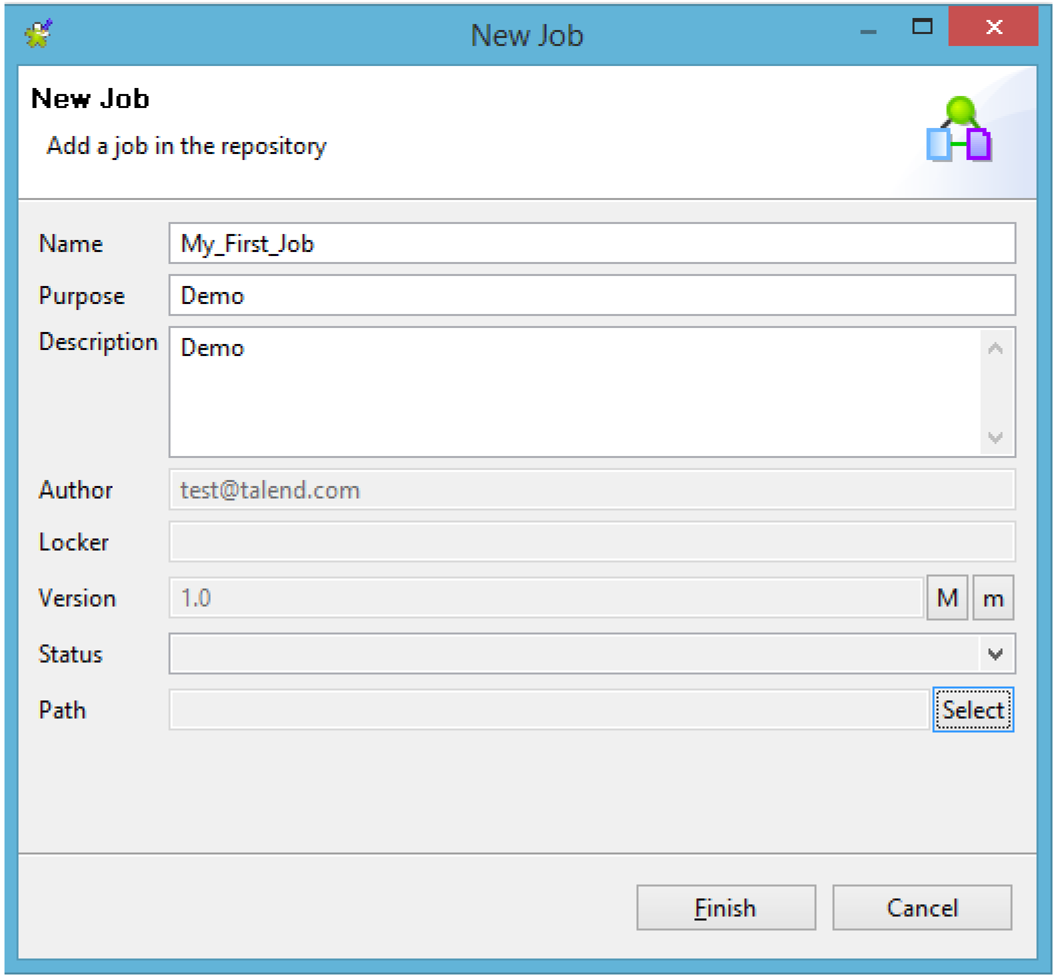
Talend Job Work Flow



Cybage®

Create Job



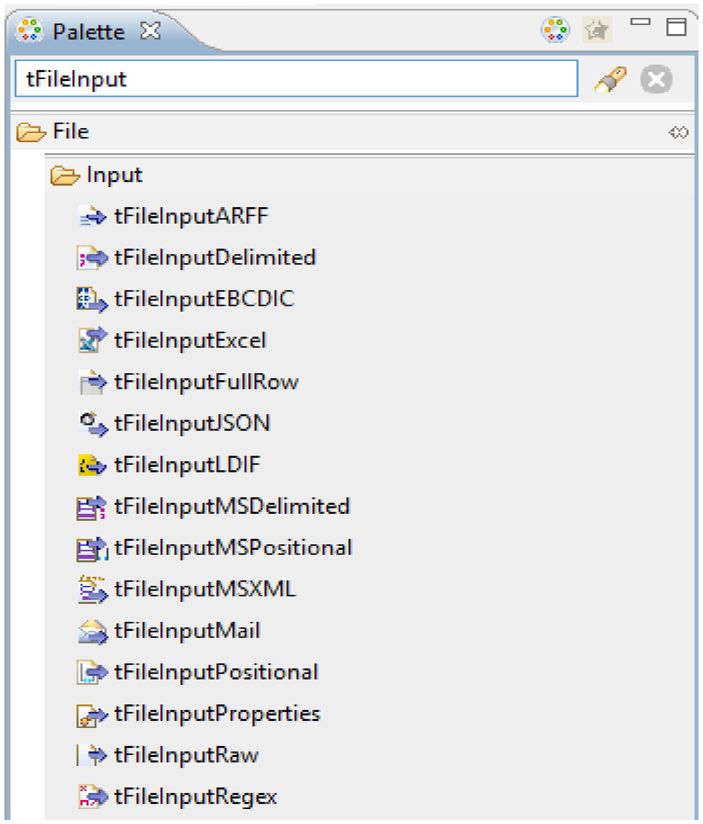

New Job

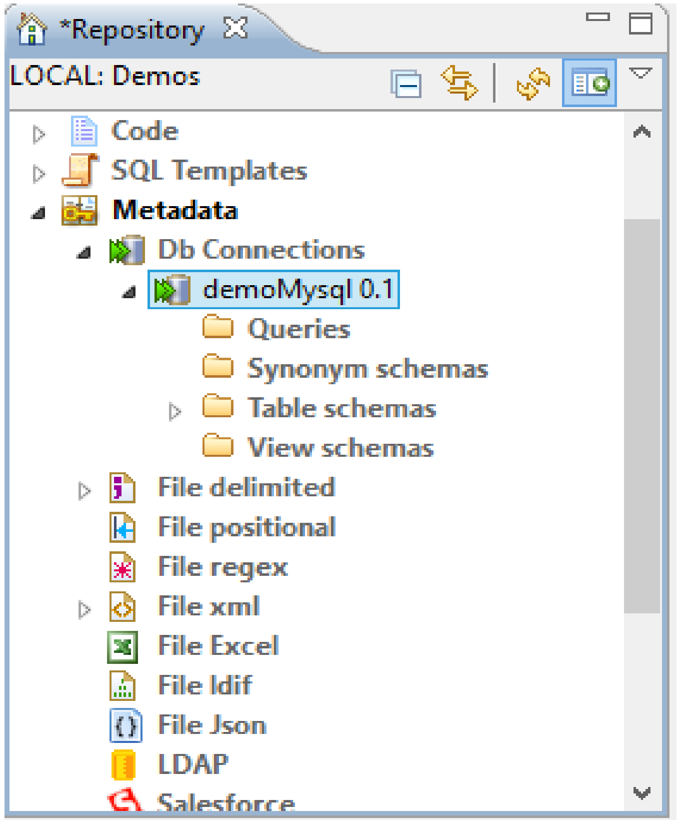
New Job

Add a job in the repository

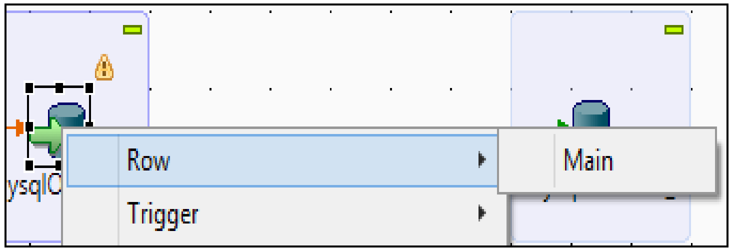
Name	My_First_Job
Purpose	Demo
Description	Demo
Author	test@talend.com
Locker	
Version	1.0
Status	
Path	

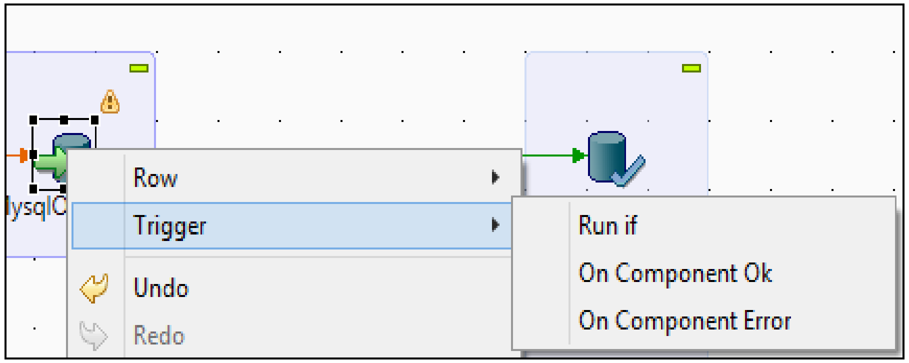
Add Components

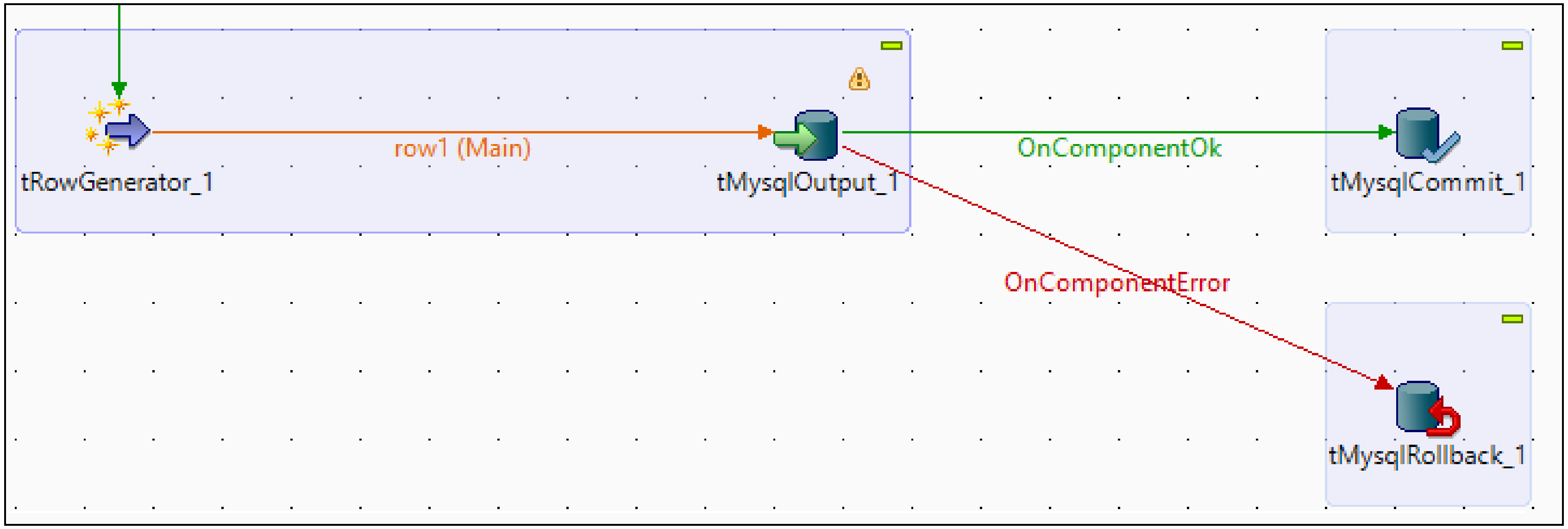




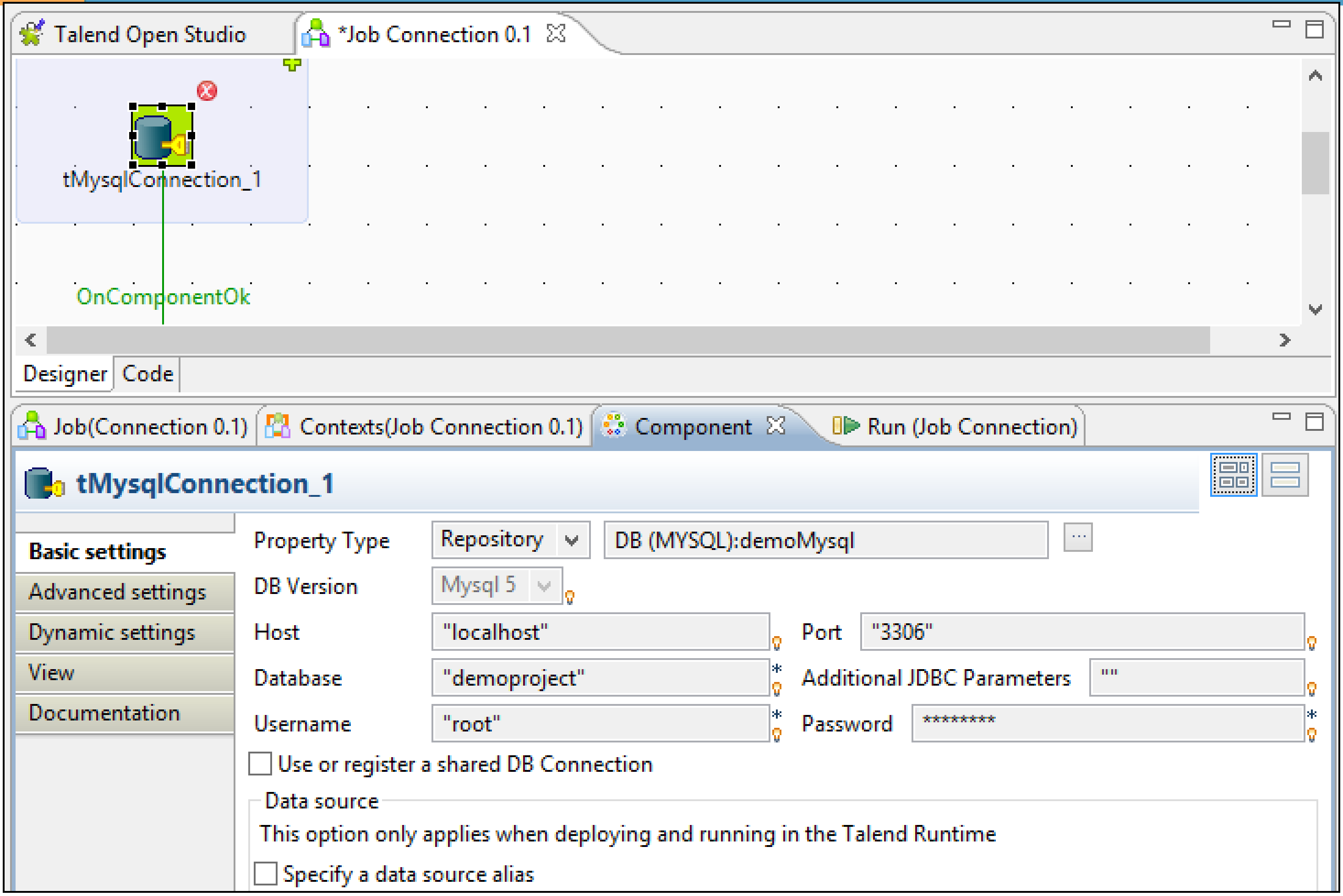
Connect Components







Configure Components



tMysqlConnection_1

Property Type: Repository — DB (MYSQL):demoMysql

DB Version: Mysql 5

Host: "localhost" Port: "3306"

Database: "demoproject" Additional JDBC Parameters: ""

Username: "root" Password: ********

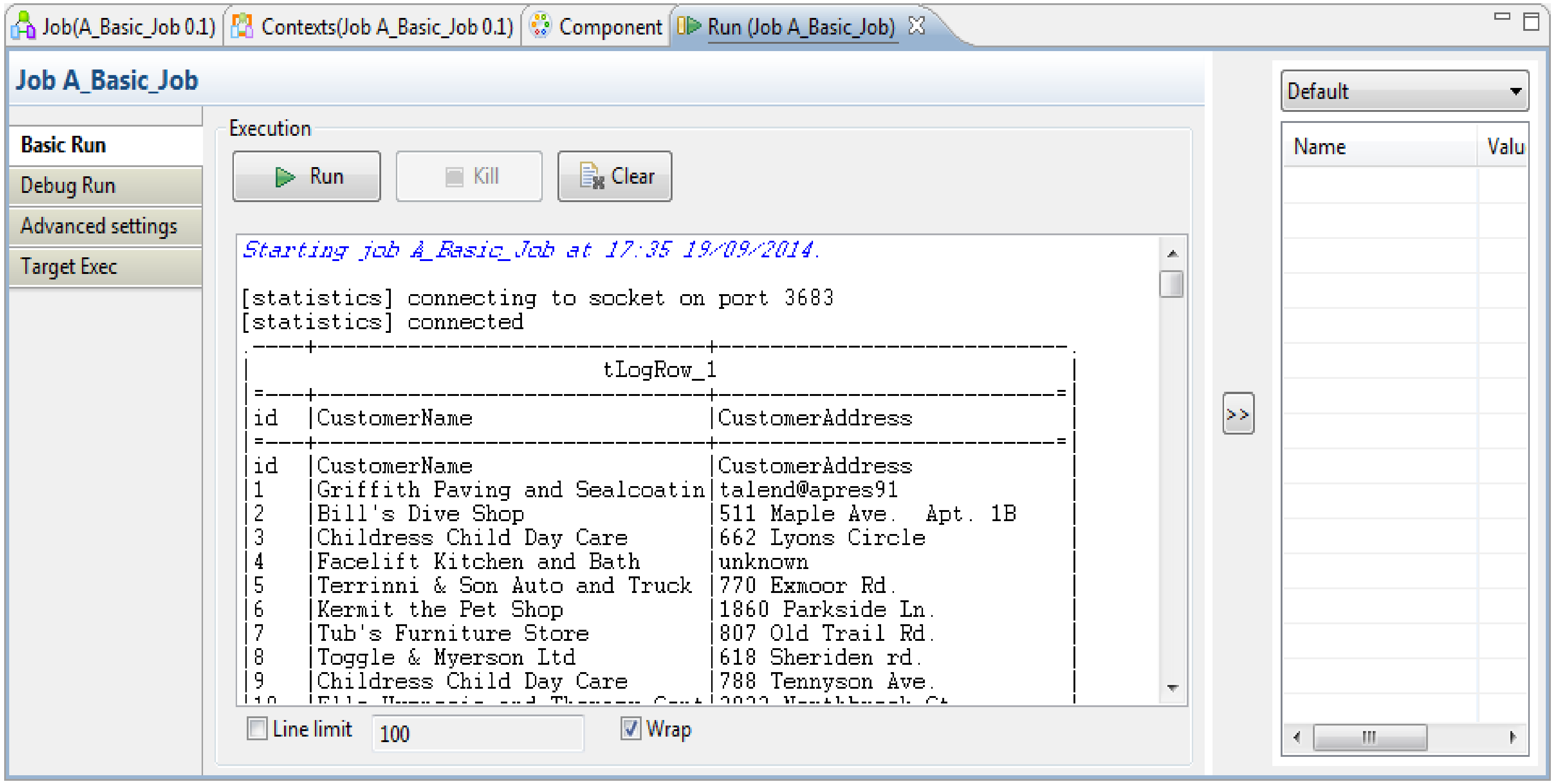
☐ Use or register a shared DB Connection

Data source

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

☐ Specify a data source alias

Execute Job



```
Starting job A_Basic_Job at 17:35 19/09/2014.

[statistics] connecting to socket on port 3683
[statistics] connected
.----+-------------------------------+--------------------------.
|                          tLogRow_1                            |
|=---+-------------------------------+-------------------------=|
|id  |CustomerName                   |CustomerAddress           |
|=---+-------------------------------+-------------------------=|
|id  |CustomerName                   |CustomerAddress           |
|1   |Griffith Paving and Sealcoatin|talend@apres91            |
|2   |Bill's Dive Shop               |511 Maple Ave. Apt. 1B    |
|3   |Childress Child Day Care       |662 Lyons Circle          |
|4   |Facelift Kitchen and Bath      |unknown                   |
|5   |Terrinni & Son Auto and Truck  |770 Exmoor Rd.            |
|6   |Kermit the Pet Shop            |1860 Parkside Ln.         |
|7   |Tub's Furniture Store          |807 Old Trail Rd.         |
|8   |Toggle & Myerson Ltd           |618 Sheriden rd.          |
|9   |Childress Child Day Care       |788 Tennyson Ave.         |
```

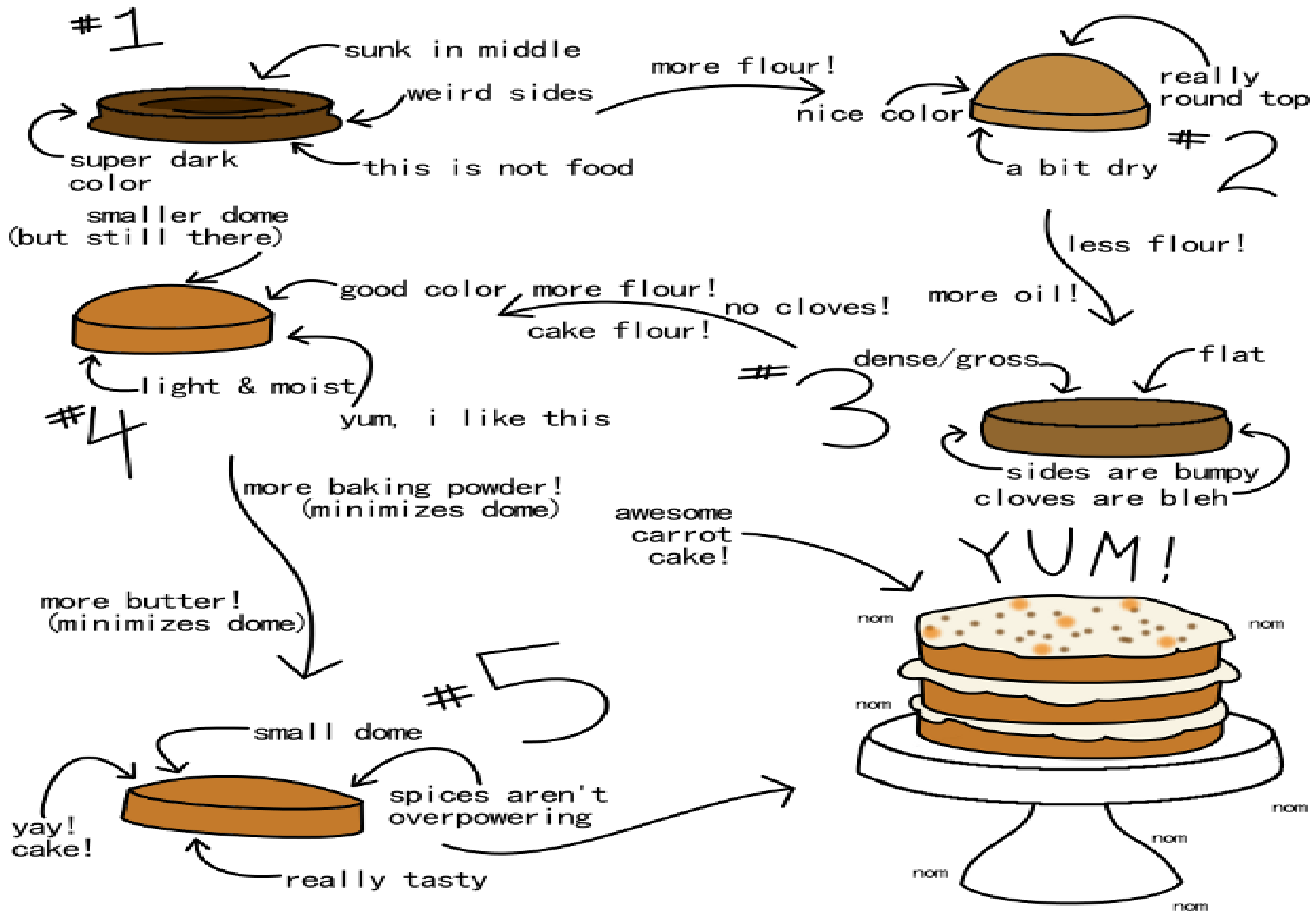
Talend Job Summary

- Data integration actions using a **library of technical components.**
- Change the default setting of components or create new components or family of components to match your exact needs.
- Set connections and relationships between components in order to define the **sequence and the nature of actions.**
- Access code at any time to edit or document the components in the designed Job.
- create and add items to the repository for reuse and sharing purposes

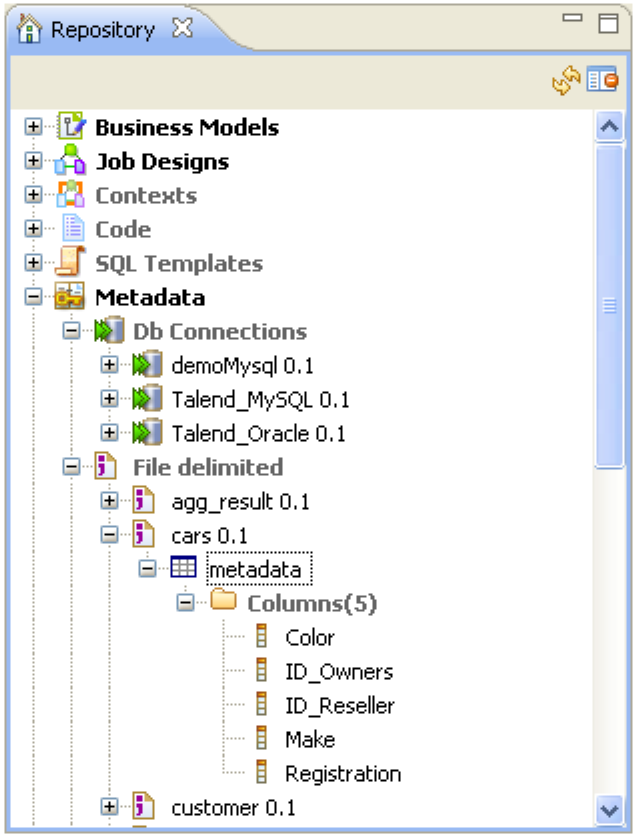
Day 2

- Overview of Talend Metadata Repository
- Understanding Context & Variables.
- Export and Import .
- Managing Job Execution (Debugging Talend Jobs)
- Talend - Mapping Data (tMap Component)
- Datawarehouse Concepts
- Demos

Data Integration – Prepare/Assemble/Deliver



Metadata Repository



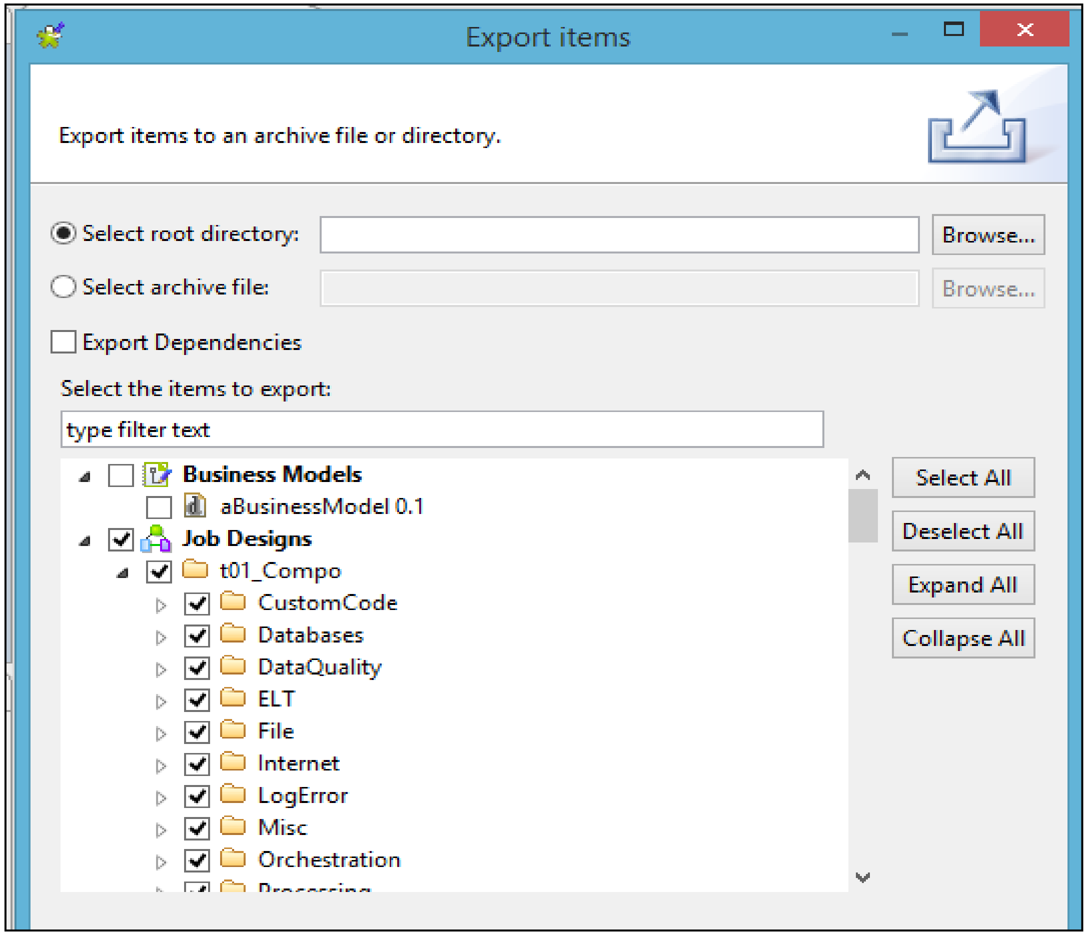
- The Metadata folder in the Repository tree view stores reusable information on files, databases, and/or systems that you need to create your Jobs.

- 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.

Export Items

- You can export multiple items from the repository onto a directory or an archive file.
- Possibility to export metadata information such as DB connection or Documentation along with your Job or your Business Model.

Export Items


Export items

Export items to an archive file or directory.

[● Select root directory:]

○ Select archive file:

☐ Export Dependencies

Select the items to export:

type filter text

- ☐ Business Models
  - ☐ aBusinessModel 0.1
- ☑ Job Designs
  - ☑ t01_Compo
    - ☑ CustomCode
    - ☑ Databases
    - ☑ DataQuality
    - ☑ ELT
    - ☑ File
    - ☑ Internet
    - ☑ LogError
    - ☑ Misc
    - ☑ Orchestration

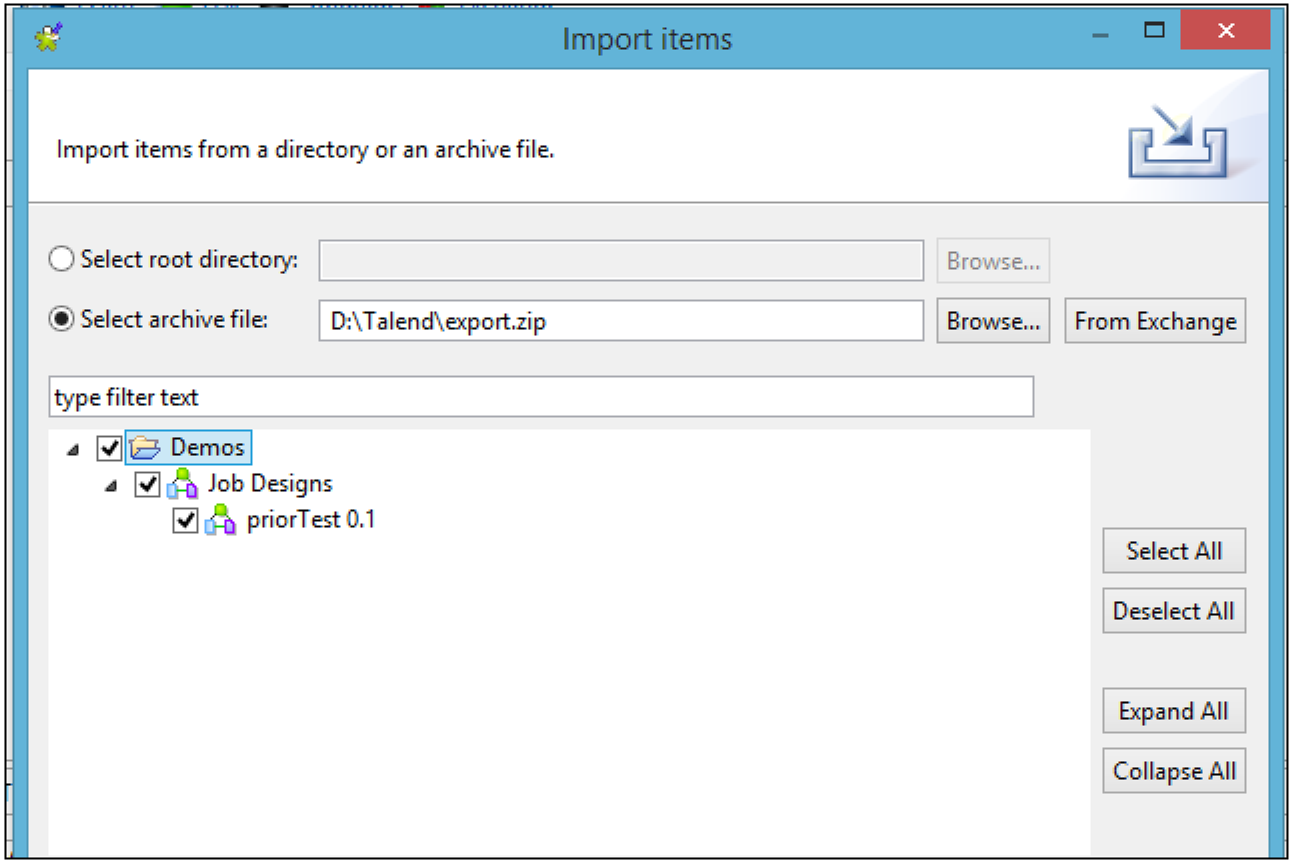
Import Items

You can import items from previous versions of *Talend Studio* or from a different project of your current version.

The items you can import are multiple:

- Business Models
- Jobs Designs
- Routines
- Documentation
- Metadata

Import Items



Import items

Import items from a directory or an archive file.

Select root directory: [ ] Browse...

[● Select archive file:] D:\Talend\export.zip Browse... From Exchange

type filter text

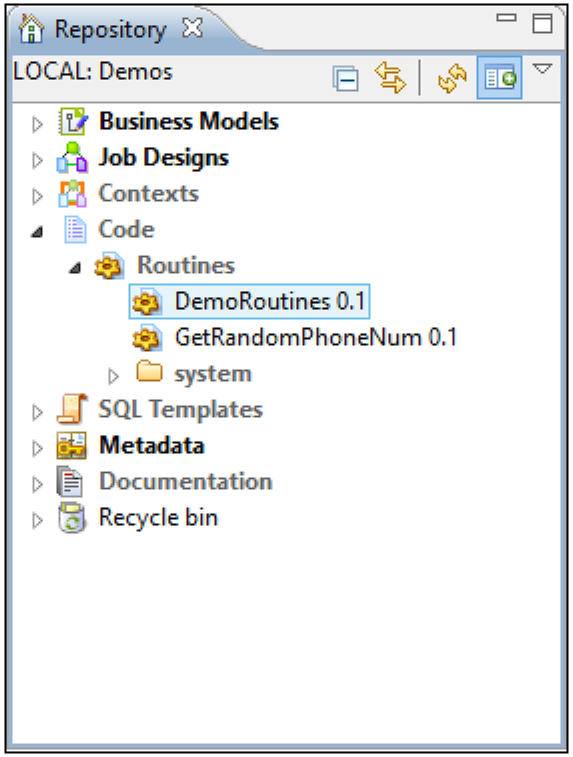
- ☑ Demos
  - ☑ Job Designs
    - ☑ priorTest 0.1

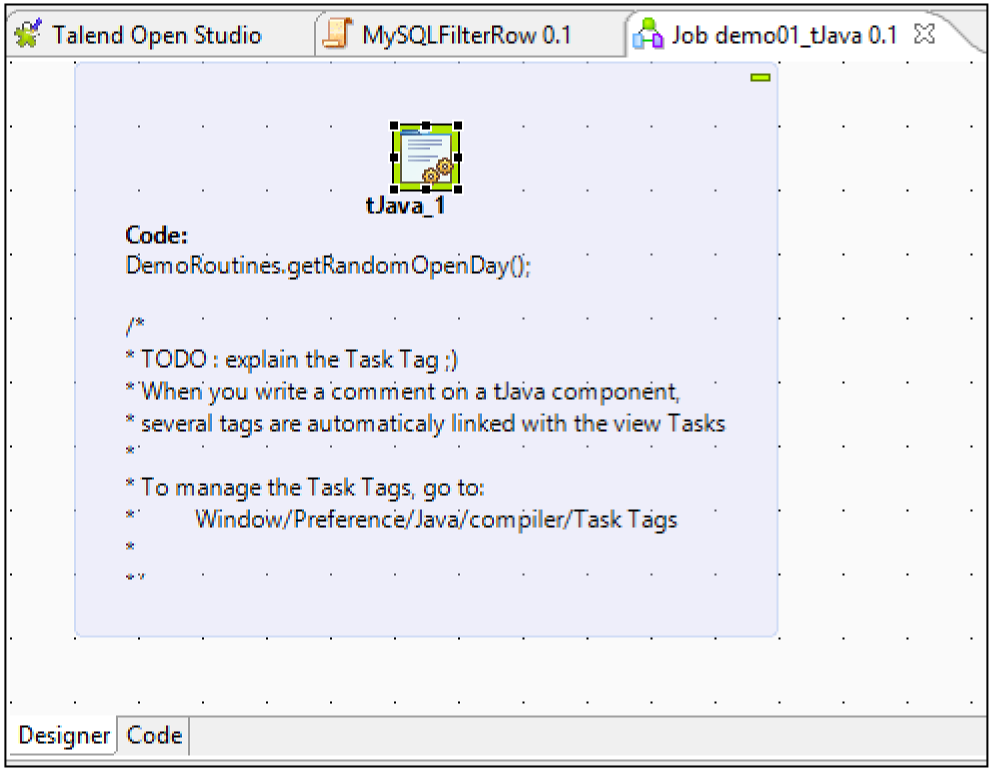
Routines

Routine - Complex Java functions, optimize for data processing and improve Job capacities.

- **System routines**: a number of system routines are provided. They are classed according to the type of data which they process: numerical, string, date etc.
- **User routines**: these are routines which you have created or adapted from existing routines.

Routines

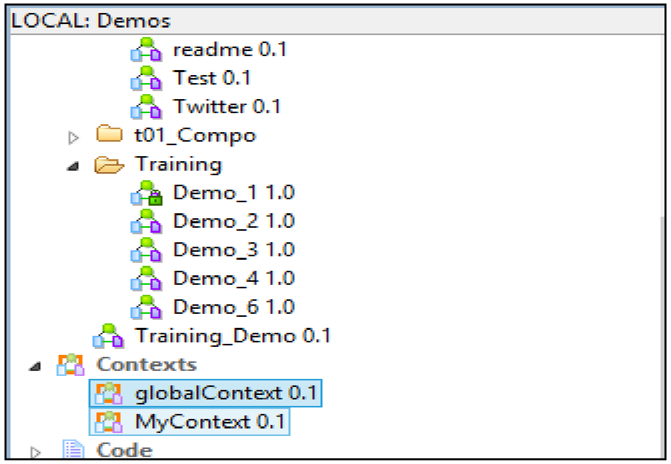




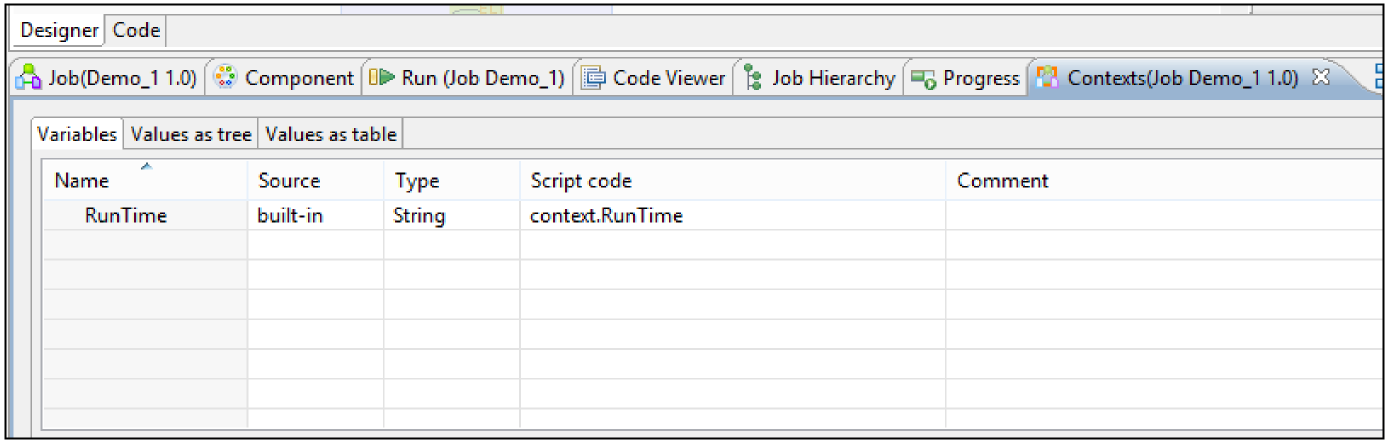
```
Code:
DemoRoutines.getRandomOpenDay();

/*
* TODO : explain the Task Tag ;)
* When you write a comment on a tJava component,
* several tags are automaticaly linked with the view Tasks
*
* To manage the Task Tags, go to:
*        Window/Preference/Java/compiler/Task Tags
*
**/
```

Contexts and variables


LOCAL: Demos
- readme 0.1
- Test 0.1
- Twitter 0.1
- t01_Compo
- Training
  - Demo_1 1.0
  - Demo_2 1.0
  - Demo_3 1.0
  - Demo_4 1.0
  - Demo_6 1.0
- Training_Demo 0.1
- Contexts
  - globalContext 0.1
  - MyContext 0.1
- Code

- Create context data sets on a one-shot basis from the **context tab of a Job**
- **Centralize** the context data sets in the Contexts node of the Repository tree view in order to reuse them in different Jobs.



Name	Source	Type	Script code	Comment
RunTime	built-in	String	context.RunTime	

Contexts and variables

Variables –

It represent values which change throughout the execution of a program.

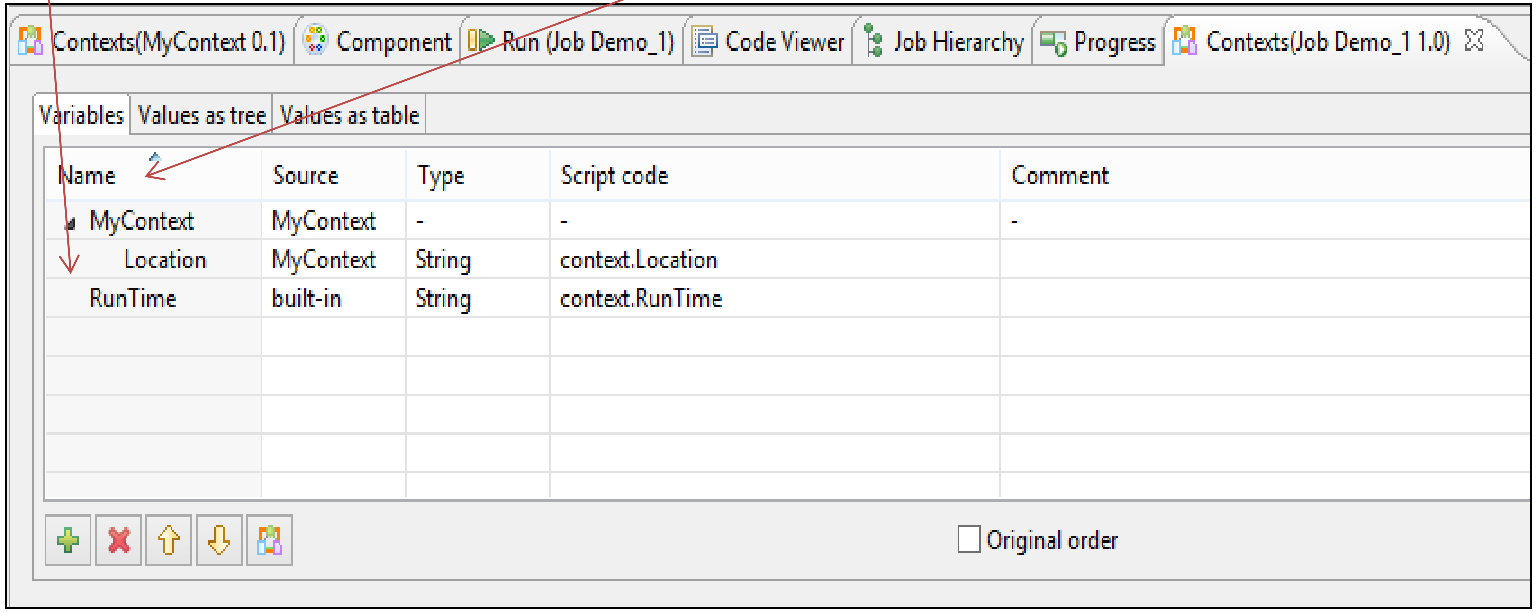
Global variable is a system variable which can be accessed by any module or function. It retains its value after the function or program using it has completed execution.

Context variable is a variable which is defined by the user for a particular context.

Contexts and variables

Adding Job Specific Context

Drag from Global Context Tree



Name	Source	Type	Script code	Comment
MyContext	MyContext	-	-	-
Location	MyContext	String	context.Location	
RunTime	built-in	String	context.RunTime	

Managing Jobs

Activating/Deactivating a component or a subjob

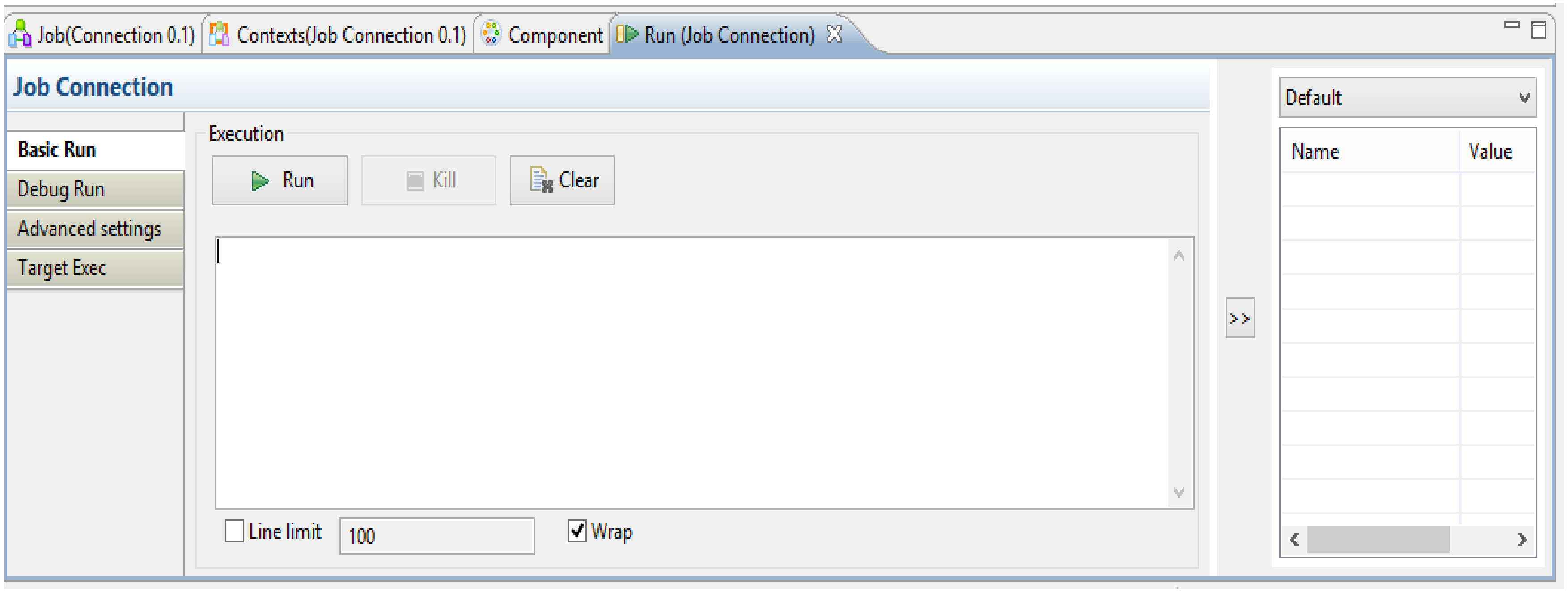
- You can activate or deactivate a subjob directly connected to the selected component.
- You can also activate or deactivate a single component as well as all the subjobs linked to a Start component.

Handling Execution

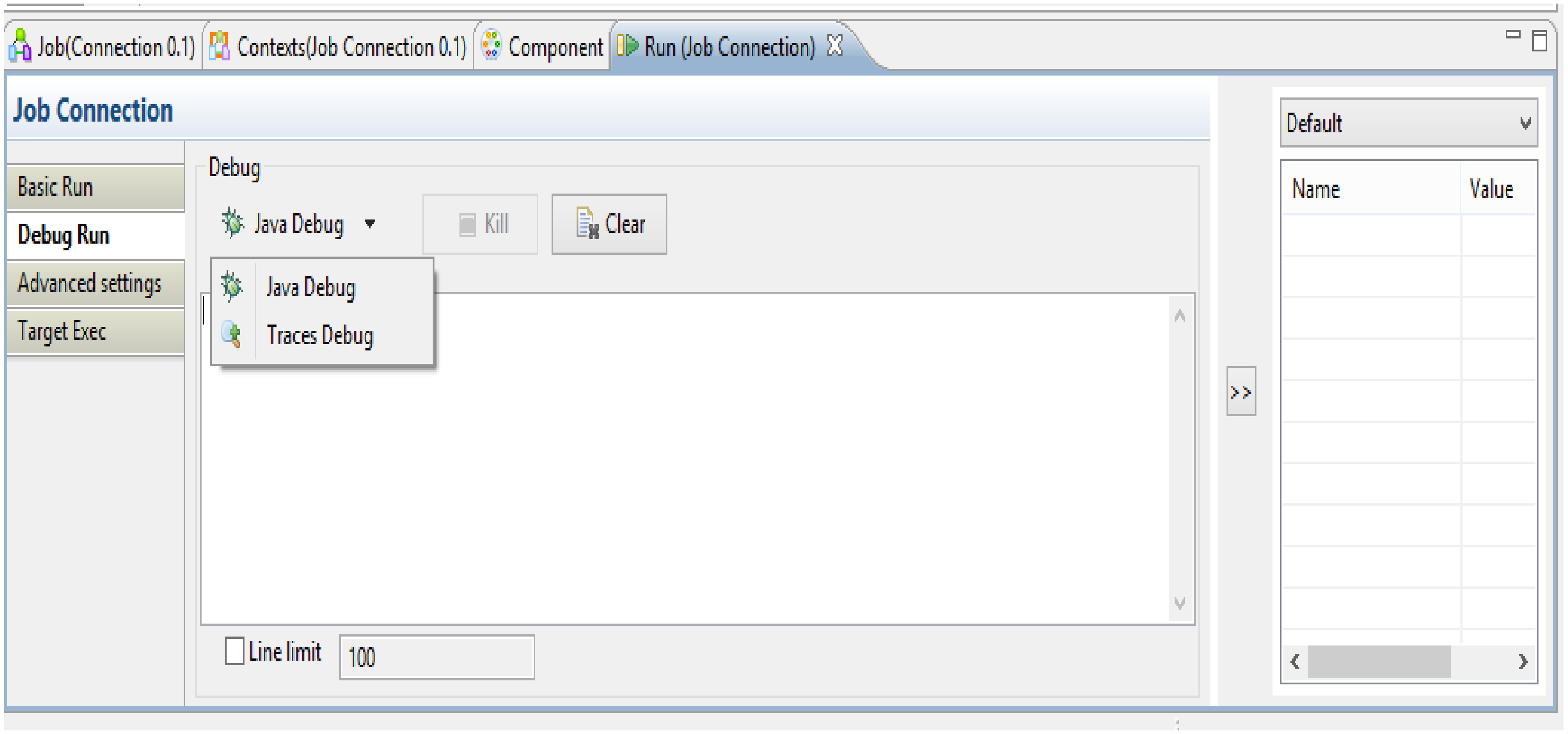
You can execute a Job in several ways.

- Normal mode
- Java Debug mode
- Traces Debug mode
- Set advanced execution settings

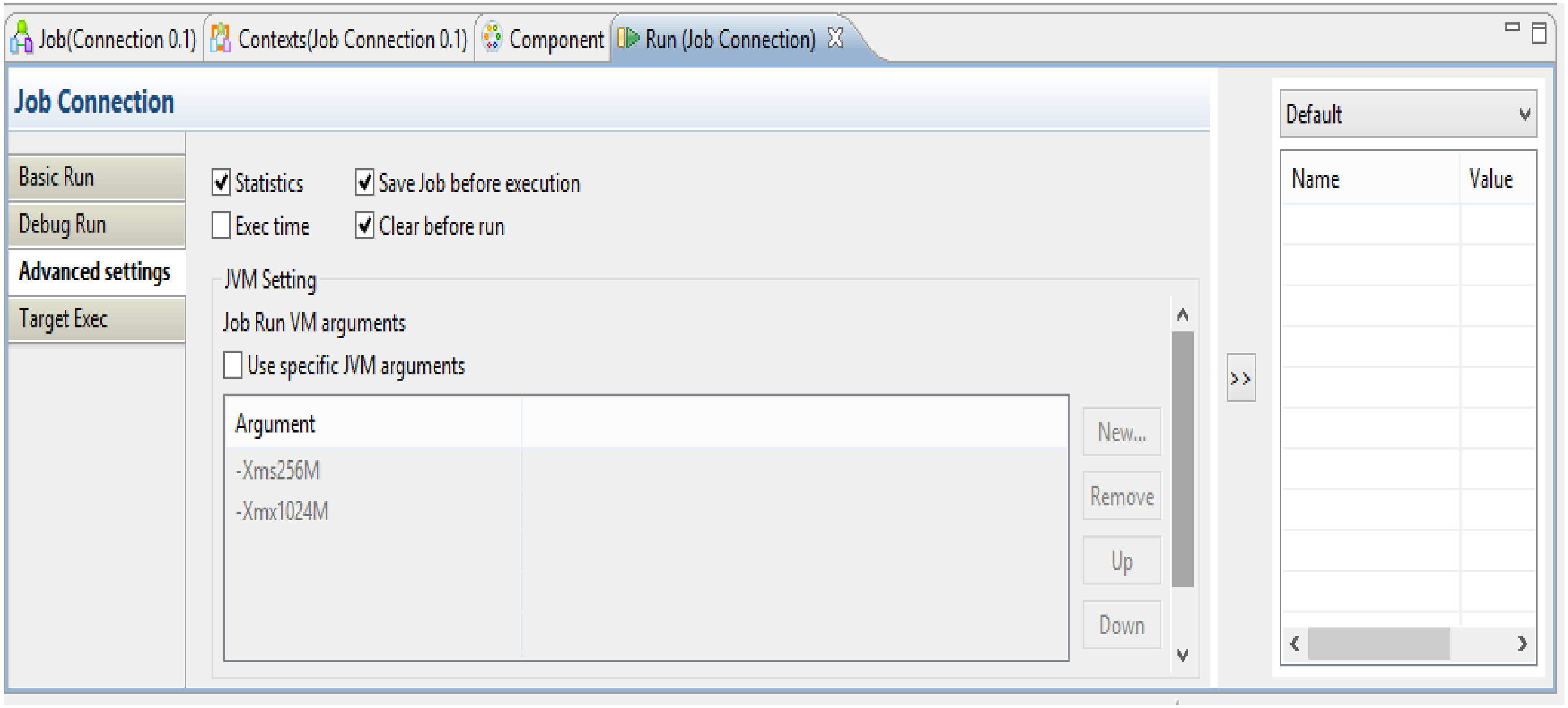
Normal mode


Job(Connection 0.1)
Contexts(Job Connection 0.1)
Component
Run (Job Connection)
Job Connection
Basic Run
Debug Run
Advanced settings
Target Exec
Execution
Run
Kill
Clear
Line limit
100
Wrap
>>
Default
Name
Value

Debug mode


Job(Connection 0.1)
Contexts(Job Connection 0.1)
Component
Run (Job Connection)
Job Connection
Basic Run
Debug Run
Advanced settings
Target Exec
Debug
Java Debug
Kill
Clear
Java Debug
Traces Debug
Line limit
100
Default
Name
Value

Set advanced execution settings


Job(Connection 0.1) | Contexts(Job Connection 0.1) | Component | Run (Job Connection)

Job Connection

Basic Run
Debug Run
Advanced settings
Target Exec

[✓] Statistics [✓] Save Job before execution
[ ] Exec time [✓] Clear before run

JVM Setting

Job Run VM arguments

[ ] Use specific JVM arguments

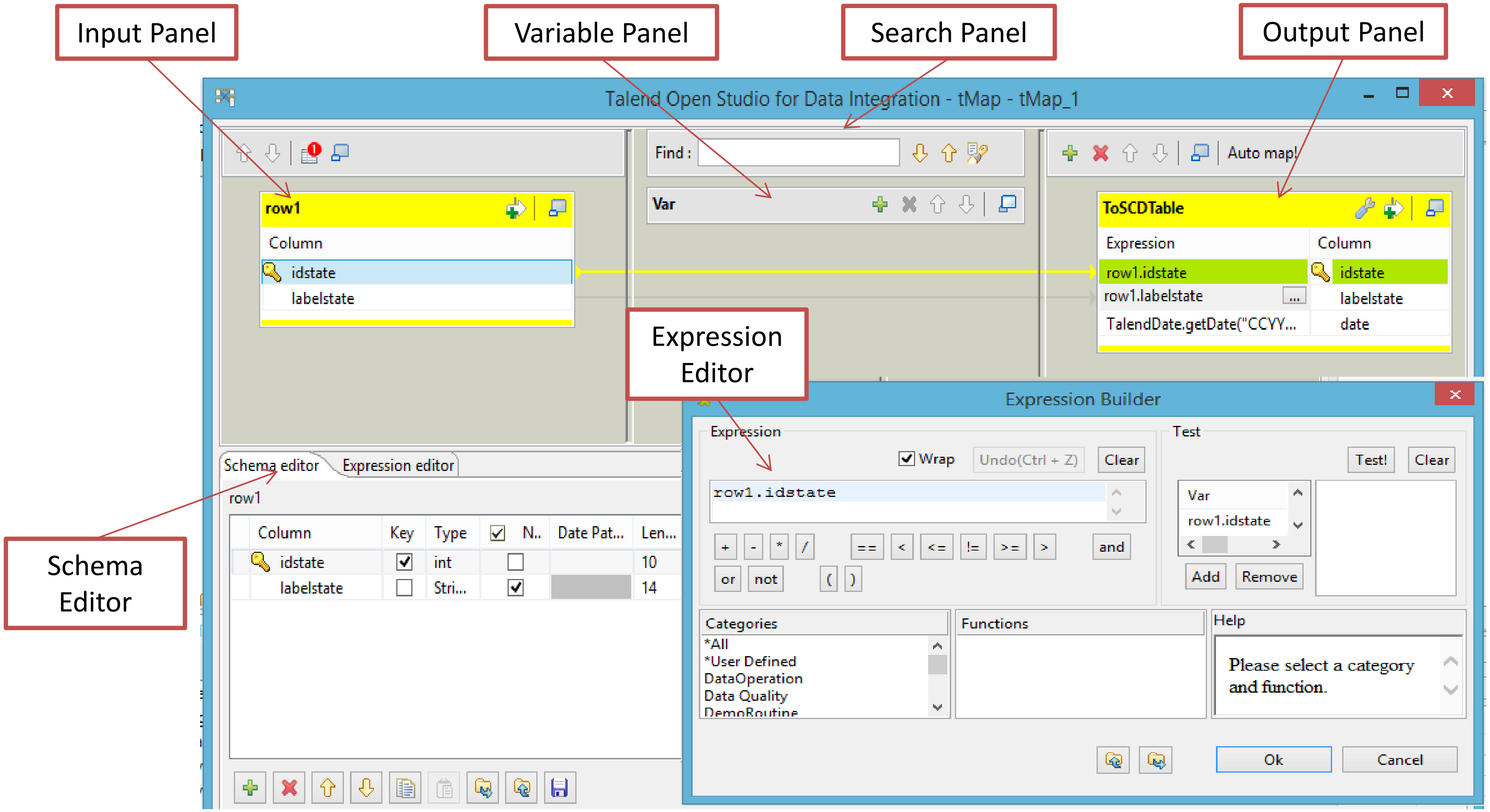
Argument
-Xms256M
-Xmx1024M

Mapping Data

- Mapping components are advanced components which require more detailed explanation

- The **Map Editor** is an "all-in-one" tool allowing you to define all parameters needed to map, transform and route your data flows via a convenient graphical interface.

Mapping Data



row1

Column	Key	Type	✓ N..	Date Pat...	Len...
idstate	✓	int	☐		10
labelstate	☐	Stri...	✓		14

Mapping Data

Input panel - It offers a graphical representation of all (main and lookup) incoming data flows. The data are gathered in various columns of input tables.

Variable panel - The central panel in the **Map Editor**. It allows the centralization of redundant information through the mapping to variable and allows you to carry out transformations.

Search panel - Above the **Variable panel**. It allow you to search in the editor for columns or expressions that contain the text you enter in the **Find** field.

Mapping Data

Output panel - Top right panel on the editor. It allows mapping data and fields from Input tables and Variables to the appropriate Output rows.

Schema editor - tab offers a schema view of all columns of input and output tables in selection in their respective panel.

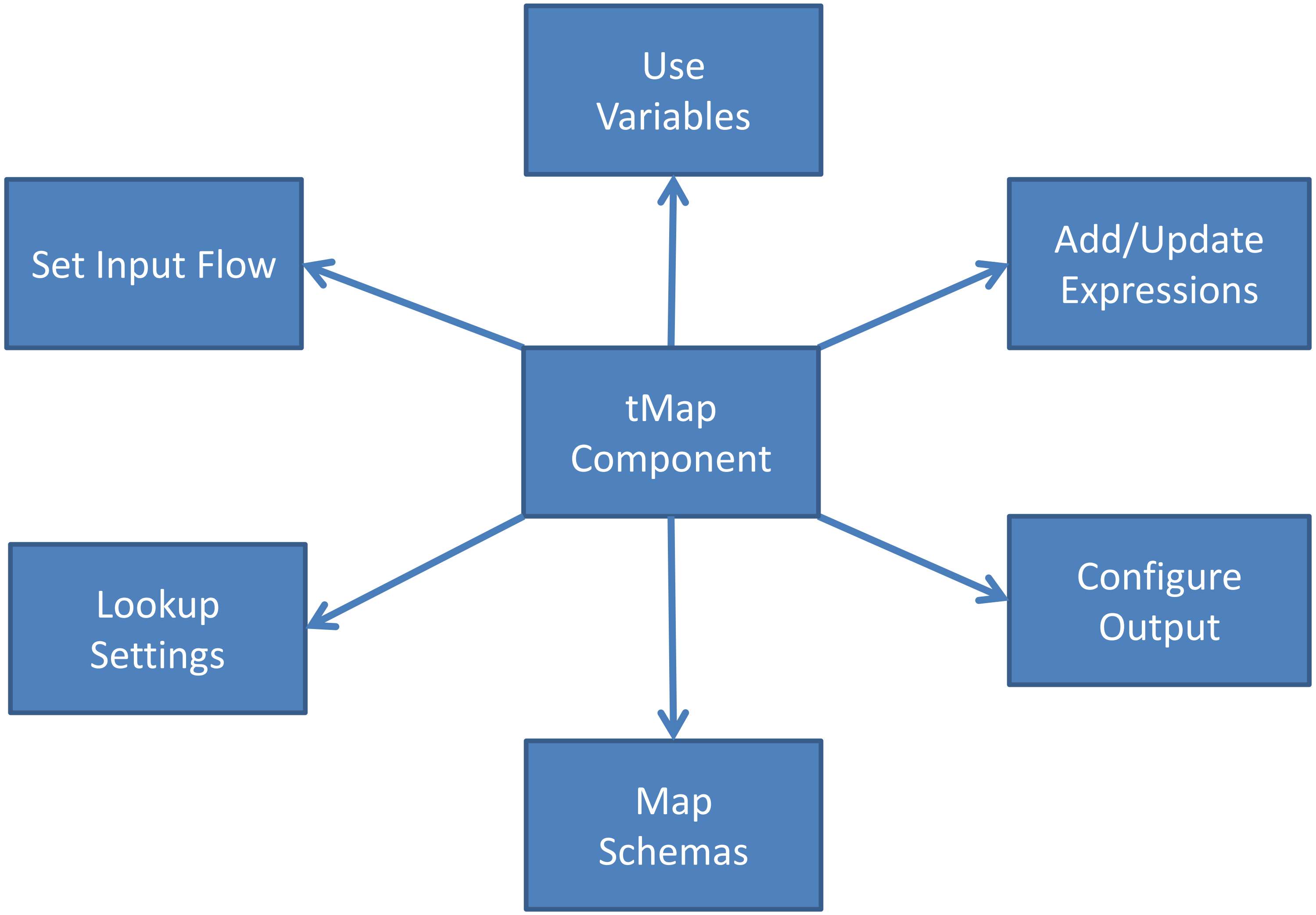
Expression editor - is the edition tool for all expression keys of Input/output data, variable expressions or filtering conditions.

Map Operations

tMap allows the following types of operations:

- data multiplexing and demultiplexing,
- data transformation on any type of fields,
- fields concatenation and interchange,
- field filtering using constraints,
- data rejecting.

Map - Summary


Use Variables
Set Input Flow
Add/Update Expressions
tMap Component
Lookup Settings
Configure Output
Map Schemas

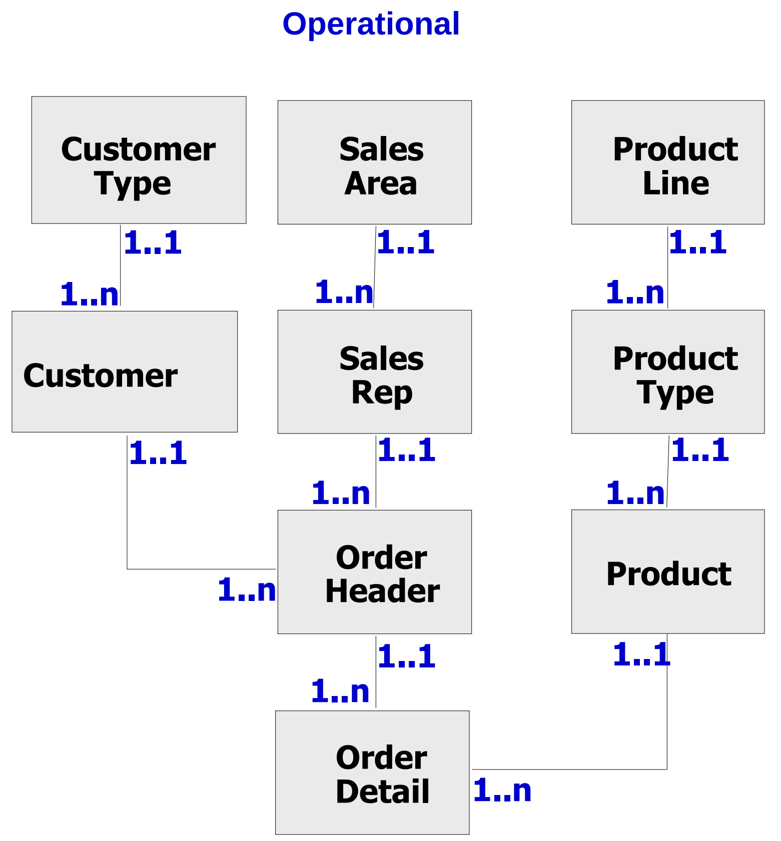
Building Dimensional Model

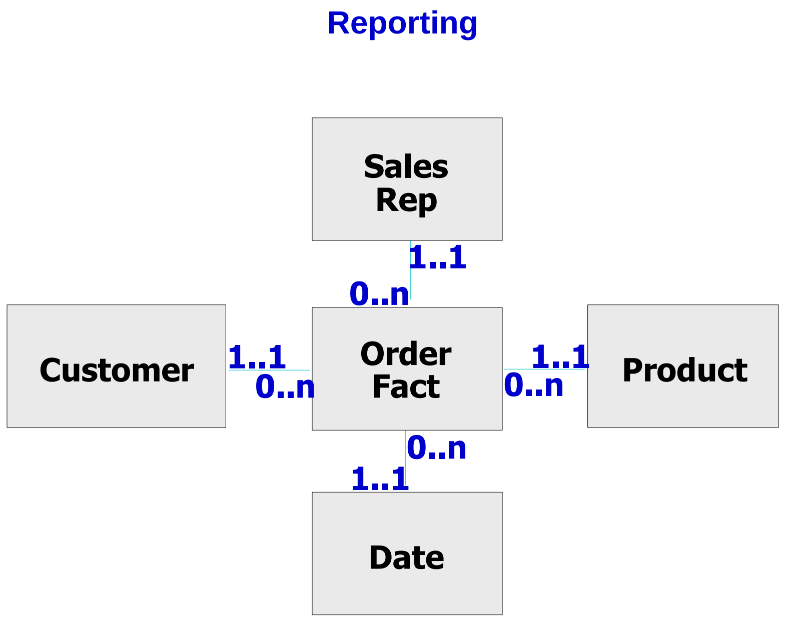
The four key decisions made during the design of a dimensional model

- Identify the Source Data for business process.
- Define the grain of data .
- Identify the dimensions.
- Identify the facts.

Operational vs Reporting Databases

- Relational databases are typically either:



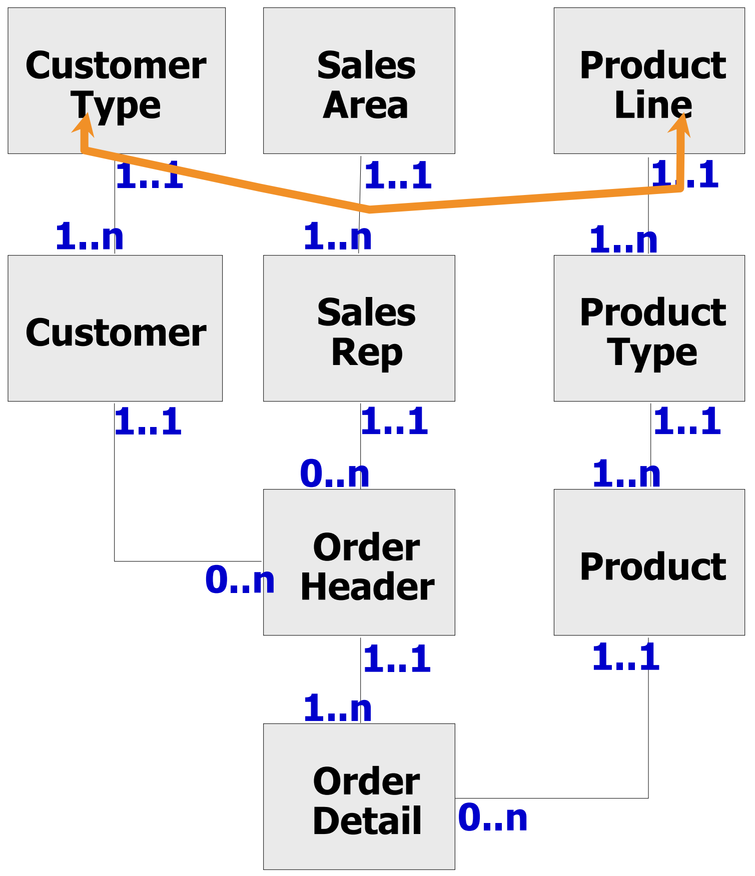


Features of an Operational Database

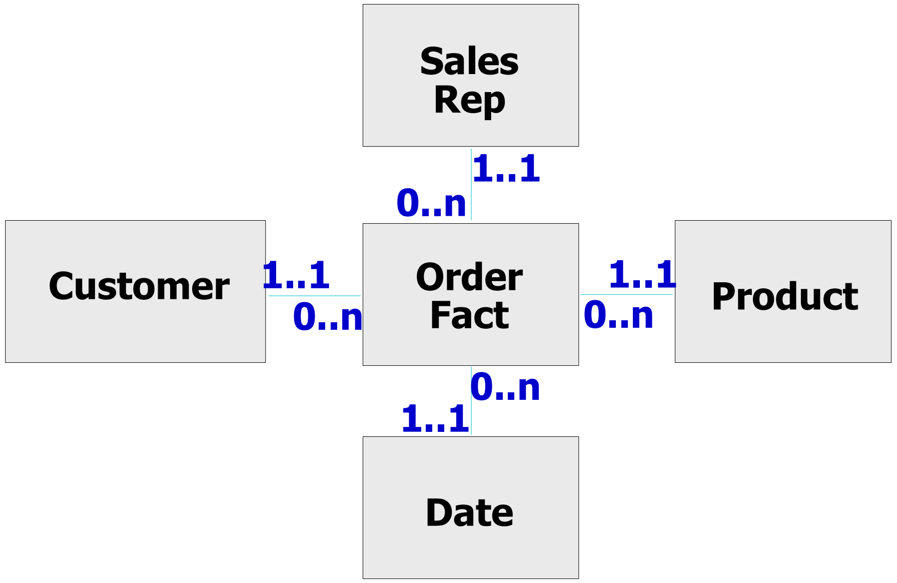
- Operational databases:
  - are designed to maximize accuracy and minimize redundancy
  - are optimized for writing/updating data rather than reading data
  - often result in monolithic designs with multiple joins
  - Large queries can perform slowly.

Identify Issues with Operational Databases

- "Show all customer types that bought from a product line."
- The query must check data in seven tables before returning a result set.



Reporting Databases (Star Schema Design)

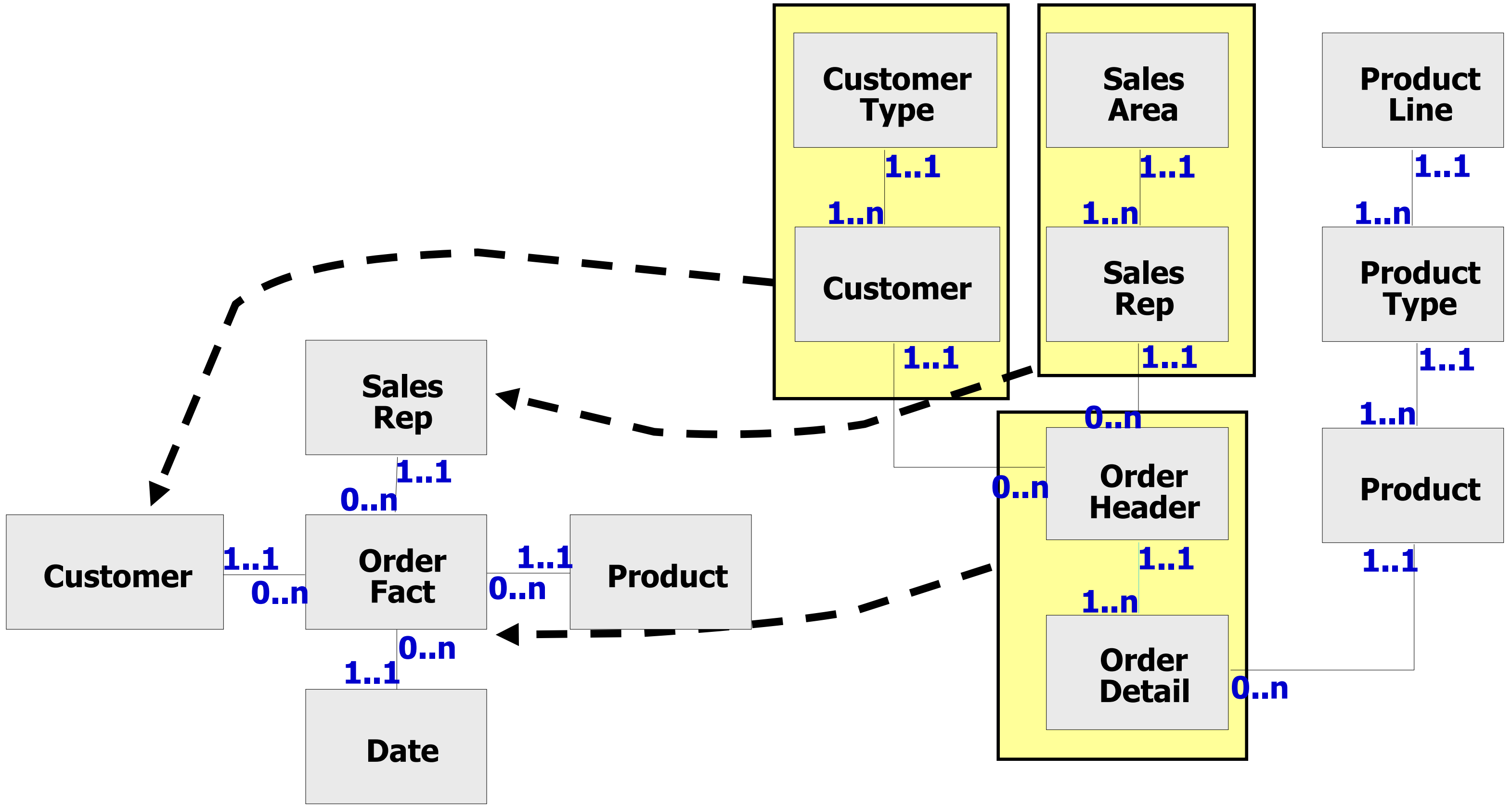


- Transcriptional data is stored in a fact table
- Reference data is stored in separate dimension tables

- **same information, but five tables instead of nine**

Create a Star Schema

- Collapse the relationships to form dimensions (perspectives).



Examine Operational Data

- Data is normalized

Product Line Table

PL#	PL_Desc
a	Classic Tents
b	Moose Boots

2 rows

Product Type Table

PL#	PT#	PT_Desc
a	1	Pup Tents
a	2	Family Tents
b	11	Child Boots
b	12	Adult Boots

4 rows

Product Table

PT#	Prod#	Prod_Desc
1	101	Green
1	102	Black
2	201	Yellow
2	203	Brown
11	1101	Blue
12	1102	Blue

6 rows

Before collapsing into a star schema dimension

Examine Reporting Data

- Data is de-normalized

Product Dimension Table

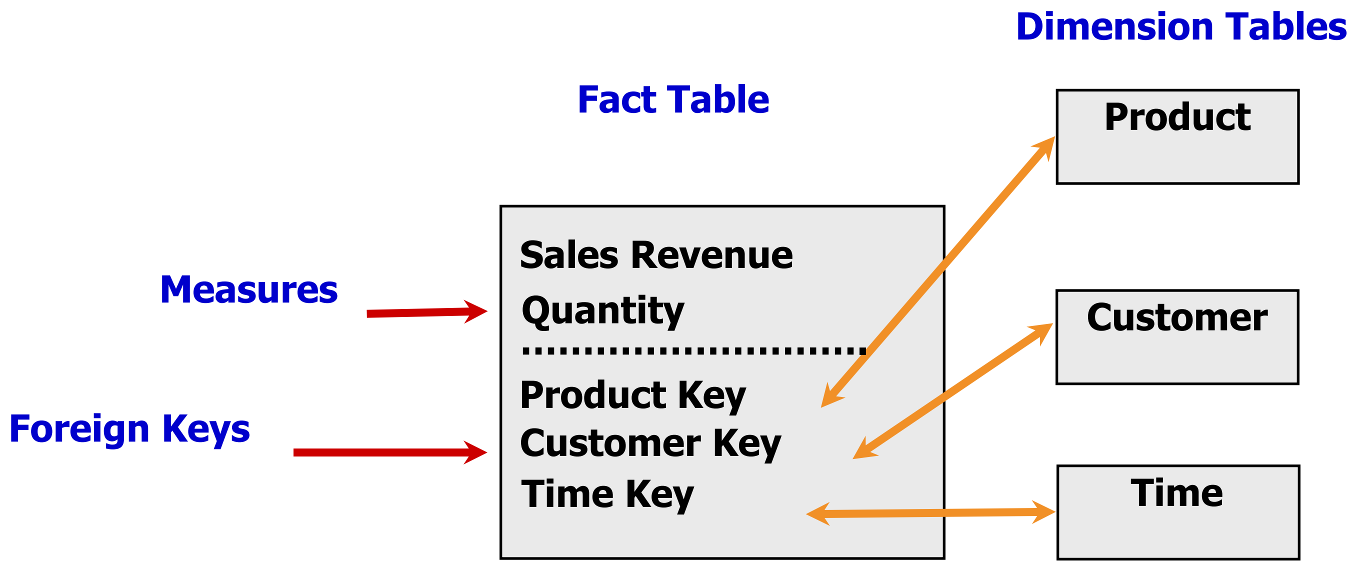
PL#	PL_Desc	PT#	PT_Desc	Prod#	Prod_Desc
A	Classic Tents	1	Pup Tents	101	Green
A	Classic Tents	1	Pup Tents	102	Black
A	Classic Tents	2	Family Tents	201	Yellow
A	Classic Tents	2	Family Tents	203	Brown
B	Moose Boots	11	Child Boots	1101	Blue
B	Moose Boots	12	Adult Boots	1102	Blue

6 rows

After collapsing into a star schema dimension

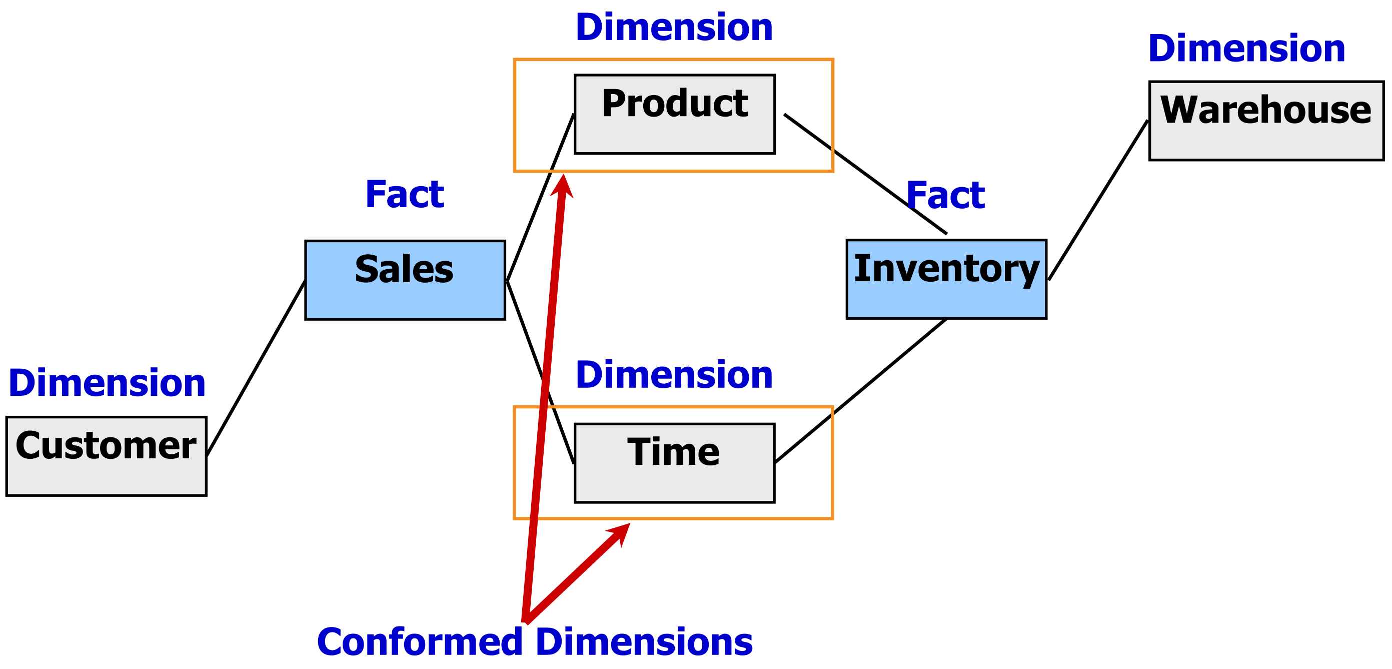
Fact Tables

- Fact tables contain the (usually additive) numbers by which a company measures itself:
  - Standard Selling Price - not additive
  - Sale Amount - additive



Dimension Tables

- Dimension tables provide descriptive information.
- Dimension tables may be “conformed” so that they are applicable to multiple fact tables.



Dimension Types

Slowly Changing Dimension –
Type 1: Overwrite

Before:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA

After:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	IL

Dimension Types

Slowly Changing Dimension –

Type 2: Add new row

Before:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA

After:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date
123	ABC	Acme Supply Co	CA	01-Jan-2000	21-Dec-2004
124	ABC	Acme Supply Co	IL	22-Dec-2004	

Dimension Types

Slowly Changing Dimension –

•Type 3: Add new attribute

Before:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State
123	ABC	Acme Supply Co	CA

After:

Supplier_Key	Supplier_Code	Supplier_Name	Original_Supplier_State	Effective_Date	Current_Supplier_State
123	ABC	Acme Supply Co	CA	22-Dec-2004	IL

Fact Types

Factless fact tables

Most Fact Tables are used to capture numerical results, but it is possible that the event merely records a set of dimensional entities coming together at a moment in time.

Such Fact table will have foreign keys from all related dimension tables without having any particular fact entry.

Example, an event of a student attending a class on a given day may not have a recorded numeric fact

Fact Types

Aggregate fact tables

- *Aggregate fact tables* are simple numeric rollups of atomic fact table data.
- Achieve improved query performance.
- Materialized views can serve as aggregate facts
- BI tools can choose appropriate (aggregated or atomic) aggregate level at query time.

Question?

Thank You