tMysqlTableTransfer

Purpose

This component copies a defined set of rows from one table into another table. It generates an internal field mapping, by matching equal names in both tables.
By using two asynchronous threads, the component gains more performance, than the normal single threaded version, generated by a normal input->map->output compontents. It's main working scenario is copying data from one database to another.
This components transfers only values from columns which are exists in source table/query and target table. None existing columns in source and target will be ignored.

Properties

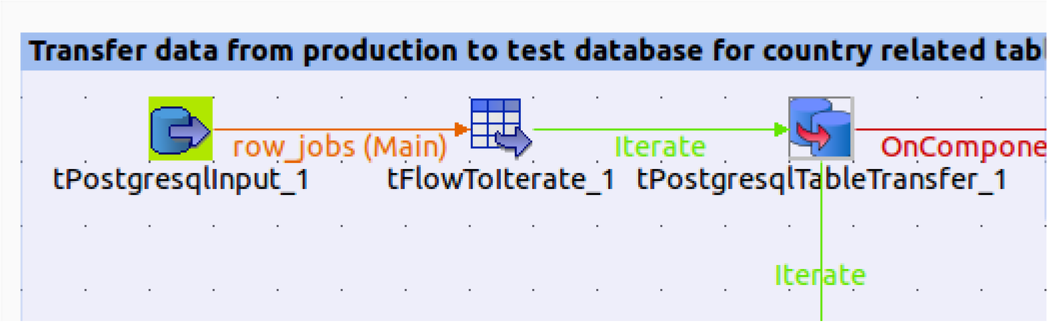
Component family		Database/Mysql Java only
Function		**tMysqlTableTransfer:** Transfer data from a query into a table rapidly
Purpose		**tMysqlTableTransfer** reads and writes data simultaneously and use automatically generated schemas and statements. It can in this version only insert data.
Basis settings	Use existing connections	If checked you can choose for source and target existing connections
	Source Connection Target Connection	source connection and target connection
	Source Host	Host of Mysql database
	Source Port	Port at which the Database is listining (Standard 3306)
	Source Database	Source database
	Source User	Username to connect to source database
	Source Password	Password for user to connect to source database
	Use self defined query	default=false: a source table name will be required
	Source table	visible if Use self defined query is not set. Name (without schema) of the table as

		source. The source table must have columns with fits to the column name+types of the target table. All fields of target table which does not find a matching field in source table will set to null.
	Source where clause	visible if Use self defined query is not set. Define the where condition to gather selected data.
	Source Query	visible if Use self defined query is set. Define the query to gather the source data. The query must return columns with fits to the column name+types of the target table. All fields of target table which does not find a matching field in source query will set to null.
	Target Host	Host of Mysql database
	Target Port	Port at which the Database is listining (usually 3306)
	Target Database	Mysql database
	Target User	Username to login into target database
	Target Password	Password to login into the target database
	Target Table	Name of the target table without schema.
	Truncate Target Table	Option to truncate the target table before import starts.
	Disable keys while importing	Option to disable indexes (only not unique indexes) to improve the performance. At the end, indexes will be enabled and that task can consume notable time.
	Log interval	Seconds between log output (which reflects read and write counter and the average transfer rate (rows/s)
	Die on error	default=true: Let the component stop working if errors occurs.
Advanced Settings	Source fetch size	default=10000. The amount of rows which will be loaded at once from the source database.
	Target batch size	default=10000. The amount of rows which will be processed at once in the target database.

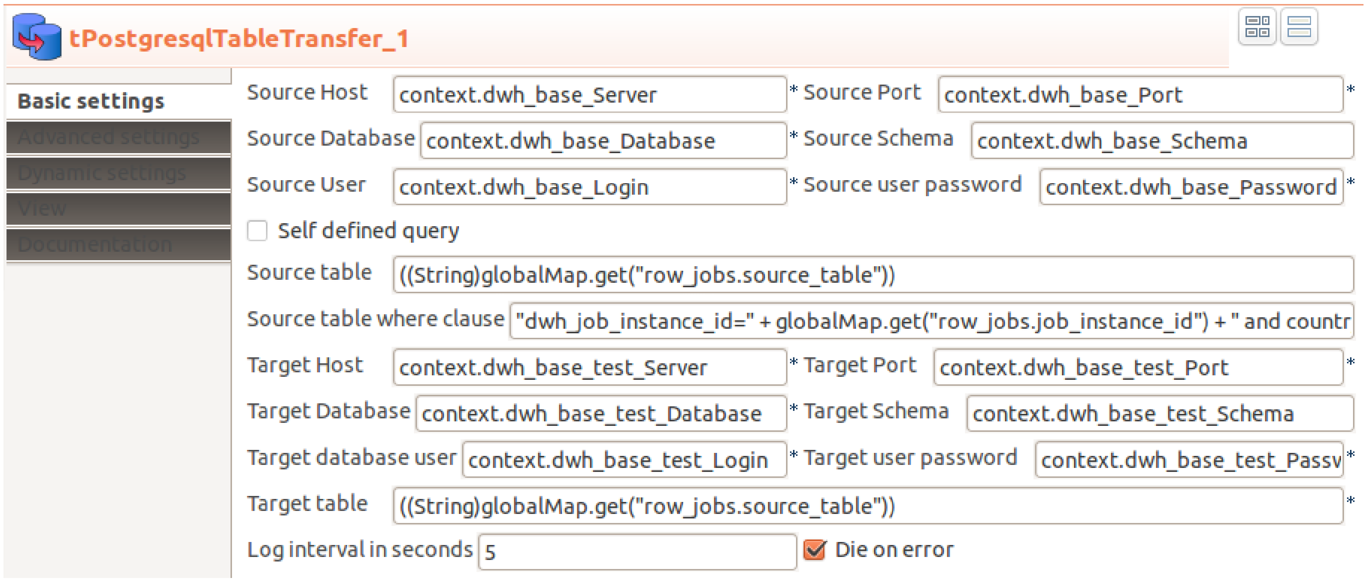
	Source Properties	Additional properties for the connection to source database. (key=value pairs separated with semicolon)
	Target Properties	Additional properties for the connection to the target database. (key=value pairs separated with semicolon)
	Synchron Transfer	Option to disable asynchronous transfer. This should be used only in seldom circumstances if performance will decrease with asynchronous transfer.
	Log out source query	default=true: writes the source query (regardless of self defined or internal generated) at log output.
	Log out target insert statement	default=false: writes the target insert prepared statement (before executing transfer).
	Single Instance	If true only a single instance of this component works within the sub job. Otherwise for every iteration a new instance will be created (not simultaneously!)

Scenario 1: Transfer a couple of tables

Filling a database test system using an amount of data from the productive system.
First we store a list of all tables which we want to transfer in an configuration table.
We iterate over these table names by reading the configuration table and start a transfer for each target.


Transfer data from production to test database for country related tabl
row_jobs (Main)
Iterate
OnCompone
tPostgresqlInput_1
tFlowToIterate_1
tPostgresqlTableTransfer_1
Iterate

The properties for tMysqlTableTransfer component:



Source Host	context.dwh_base_Server	* Source Port	context.dwh_base_Port *
Source Database	context.dwh_base_Database	* Source Schema	context.dwh_base_Schema
Source User	context.dwh_base_Login	* Source user password	context.dwh_base_Password *

□ Self defined query

Source table: ((String)globalMap.get("row_jobs.source_table"))

Source table where clause: "dwh_job_instance_id=" + globalMap.get("row_jobs.job_instance_id") + " and countr

Target Host	context.dwh_base_test_Server	* Target Port	context.dwh_base_test_Port *
Target Database	context.dwh_base_test_Database	* Target Schema	context.dwh_base_test_Schema
Target database user	context.dwh_base_test_Login	* Target user password	context.dwh_base_test_Passv *

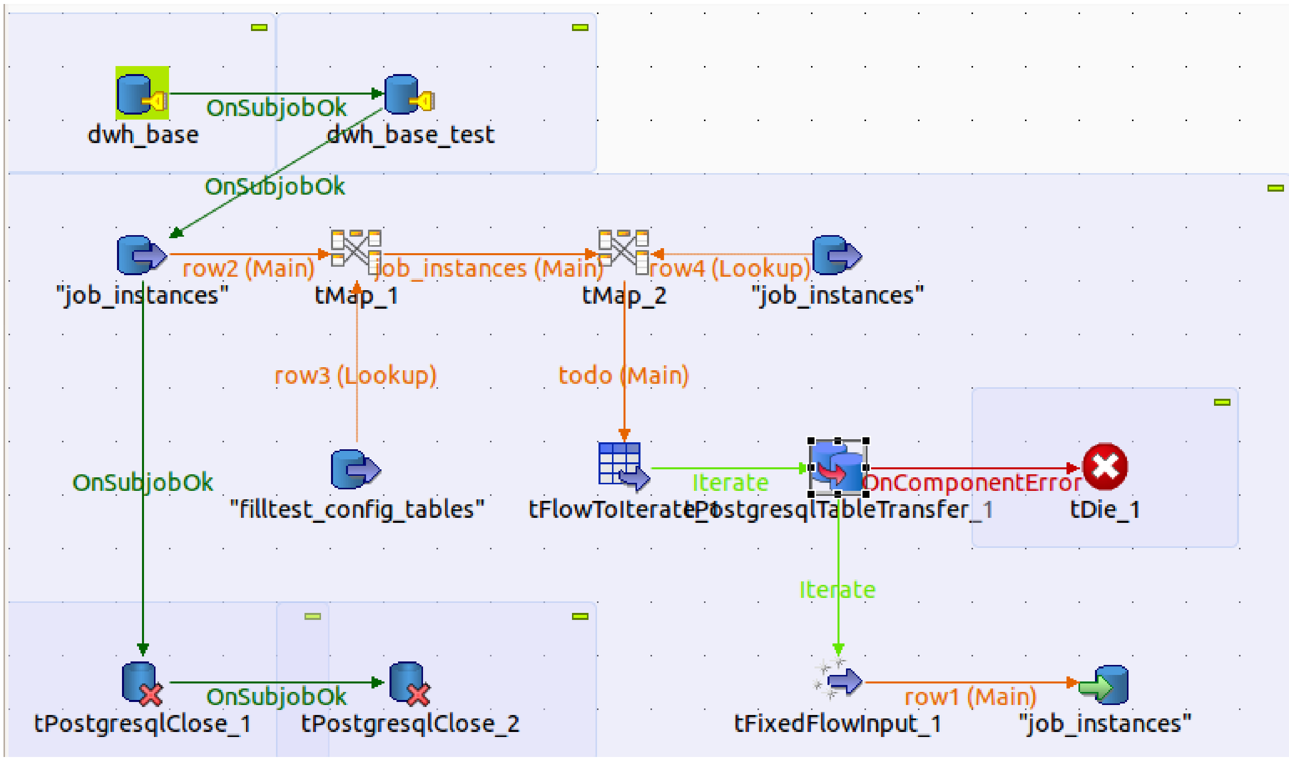
Target table: ((String)globalMap.get("row_jobs.source_table")) *

Log interval in seconds: 5 ☑ Die on error

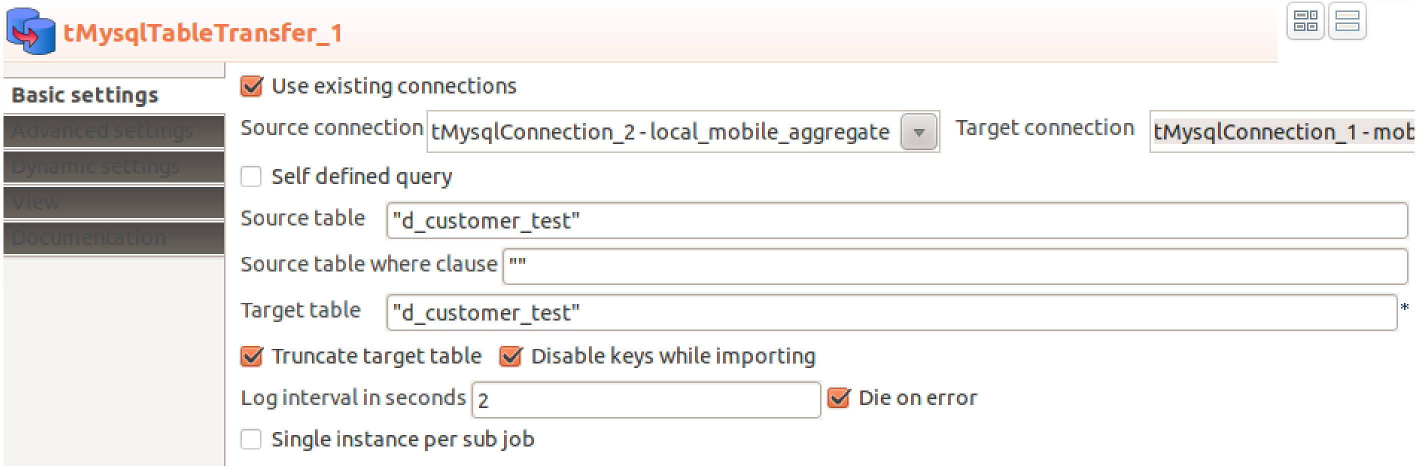
You can use context variables to set the properties.
This example shows of the component creates it own database connections.

Scenario 2: Transfer a couple of tables with using of existing connections

Here an example of using existing connections (using the derived component for PostgreSQL):



Appropriated properties for tMysqlTableTransfer:



☑ Use existing connections

Source connection: tMysqlConnection_2 - local_mobile_aggregate ▾ Target connection: tMysqlConnection_1 - mob

☐ Self defined query

Source table: "d_customer_test"

Source table where clause: ""

Target table: "d_customer_test" *

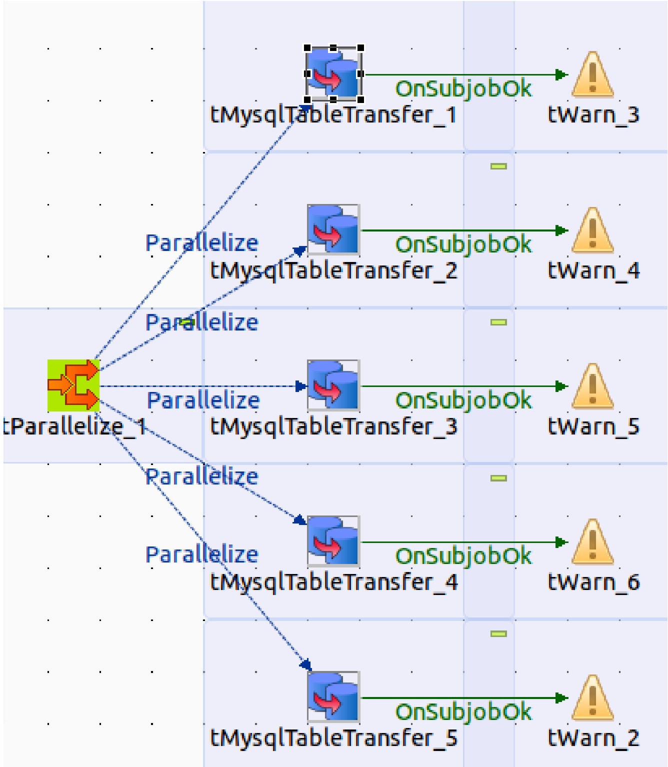
☑ Truncate target table ☑ Disable keys while importing

Log interval in seconds: 2 ☑ Die on error

☐ Single instance per sub job

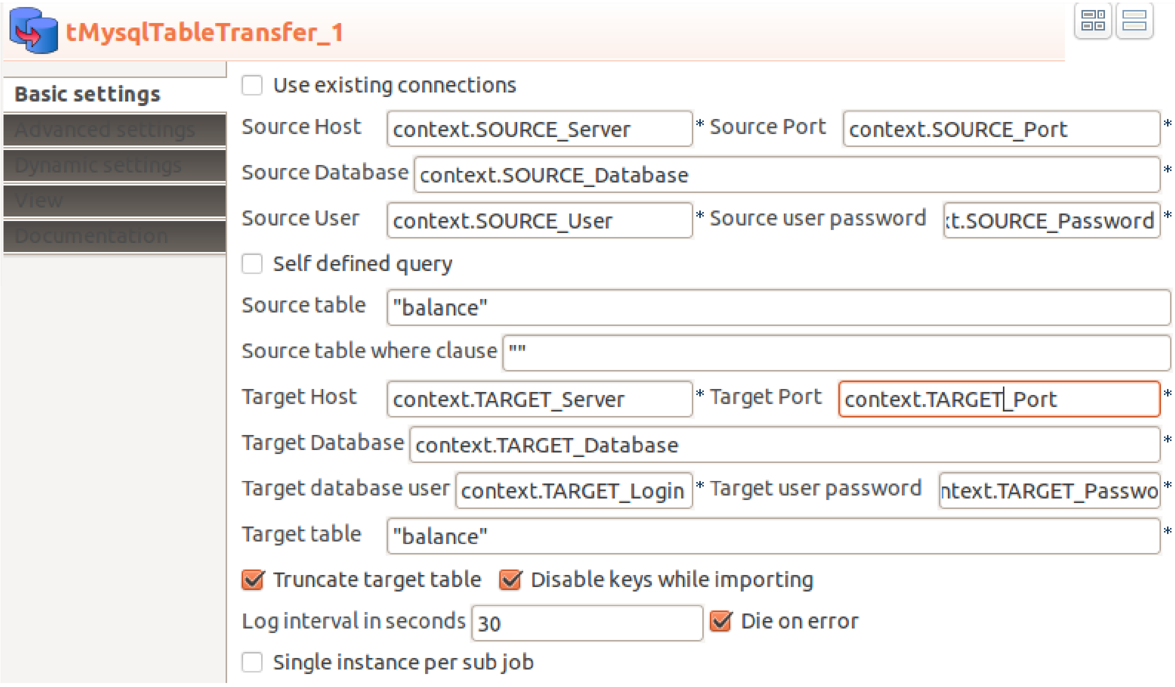
Single Instance property helps to save resources. An instance of tPostgresTableTransfer will be cached in the globalMap.

Scenario 3: Transfer a couple of tables in parallel running sub processes



Transfers can run parallel. After creating tMysqlTableTransfer_1 you can populate the others as copy and change the table names.

Here the appropriated settings



tMysqlTableTransfer_1

☐ Use existing connections

Source Host context.SOURCE_Server * Source Port context.SOURCE_Port *

Source Database context.SOURCE_Database *

Source User context.SOURCE_User * Source user password :t.SOURCE_Password *

☐ Self defined query

Source table "balance"

Source table where clause ""

Target Host context.TARGET_Server * Target Port context.TARGET_Port *

Target Database context.TARGET_Database *

Target database user context.TARGET_Login * Target user password ntext.TARGET_Passwo *

Target table "balance" *

☑ Truncate target table ☑ Disable keys while importing

Log interval in seconds 30 ☑ Die on error

☐ Single instance per sub job