# Training Basic ELT using Talend

The goal of the training is to learn how to build a simple data mart for the DELL STORE using Talend. You will learn the basics of ETL and how to use Talend.

Reference Material:

* Excel Sheet “dellstore\_sourcetotarget\_map.xlsx”

## Task 0: Prepare Data Source and access it

<https://hub.docker.com/r/aa8y/postgres-dataset/>

```

docker pull aa8y/postgres-dataset:dellstore

docker run -d --name pg-ds -p 5432:5432 aa8y/postgres-dataset:dellstore

```

Step 1: Use Command Line Access to check basic postgres commands

docker exec -it pg-ds psql -d dellstore

Use the commands “\dt” and “\d customers” to browse the data source.  
Find the command to quit the commandline using \?

Step 2: Access the database you can use dbeaver: (e.g. choco install dbeaver)

Hint: create a connection using dbeaver with the following parameters: Username: docker, Password: docker Database: dellstore

Step 3: Stop and restart the container, to check that you can interrupt your work

docker stop pg-ds

docker start pg-ds

## Task 1: Download Talend

Download Talend Openstudio for DataIntegration [Open Source Version]

Understand the basics of Talend by doing the Tutorial and the “Getting Startet” steps. (see PDF) https://info.talend.com/rs/talend/images/CB\_EN\_DI\_Cookbook\_DataIntegration.pdf

In case of questions please look at: <https://help.talend.com/>

## Task 2: Database Setup

Create new Database “dwh” in the postgresql docker image

Create Database schema (according to Excel File: Overview of schemata).

Don’t create the tables yet.

## Task 3: Big Picture

Draw a picture, how the dwh schema will look like. Include possible links and foreign keys between the tables.

## Task 4: Staging

Create staging tables (1:1 copy plus metadata columns) for the tables customer, product, categories.

Try to create the tables in a way, that is as fast as possible.

HINT for using SQL: Can you create the tables, with the almost the same SQL for each table?

Build a Job “stage <tablename>” for each table, which copies the source data to the staging tables using a TRUNCATE INSERT strategy.

Build a central job called “staging”, which includes the child jobs. For now populate the LOAD\_ID variable with the value 1.

Test the jobs, until all tables in staging have the same number of columns as in the source database

## Task 5: Simple Dimension

Build first Dimension (Produkt Dimension)

1. Create table prod\_dim with the columns mentioned in the dellstore\_sourcetotarget\_map in the integration schema as well as in the dwh schema (Tipp: Use the draft SQL from column H to make your job more easy)
2. Build a Job “int <tablename” to load the table in the integration schema using a TRUNCATE INSERT method according to the dellstore\_sourcetotarget\_map . Populate the PROD\_ID using a database sequence.
3. Create a central job “integration” and include the “int <tablename>” job in it.
4. Build a Job “load <tablename>” which compares all data in the table in the int schema with the data in the dwh schema and copies the data to the new environment using an INSERT/UPDATE method.
5. Create a central job “load” and include the “load <tablename>” job in it.
6. Test the new jobs by running it multiple times. The number of rows in all product tables should stay the same

## Task 6: Time dimension

Build a constant dimension

1. Create table time\_dim with the columns mentioned in the dellstore\_sourcetotarget\_map in the integration schema as well as in the dwh schema
2. Check, which dates are in the source tables (check orders.orderdate and orderliness.orderdate). Take the minimum date minus 5 years as start and the maximum date plus 10 years as end date for the next step.
3. Build a Job “int <tablename” to load the table in the integration schema using a TRUNCATE INSERT method, based on date values from a date generator.
4. Include the “int <tablename>” job in the central job “integration”.
5. Build a Job “load <tablename>” which compares all data in the table in the int schema with the data in the dwh schema and copies the data to the new environment using an INSERT/UPDATE method.
6. Include the “load <tablename>” job in the central job “load”.
7. Test the new jobs by running it multiple times. The number of rows in all time tables should stay the same

## Task 7: Load Control

1. Create a new central job “daily run”, which includes the central job staging, integration and load. Add dependencies between the three jobs, so that they run in the correct order.
2. Create the DWHLOAD table in the dwh schema (only).
3. Add a new job “dwhload\_control”, which reads the maximum LOAD\_ID from the DWHLOAD table. If no row is stored in the table the value 0 should be returned. Increase the ID by one and store it in the parameter “LOAD\_ID”. Create a load date LOAD\_DATE with the value 01.01.2018 and store it also in a parameter. Also write a new line to the DWHLOAD table containing the LOAD\_DATE and the updated LOAD\_ID.
4. Update all jobs to use the parameter to populate the LOAD\_ID.
5. Test the central job, test that each execution of the job generates a new LOAD\_ID and a new line in the DWHLOAD Table

## Task 8: Customer Dimension

1. Create table cust\_dim with the columns mentioned in the dellstore\_sourcetotarget\_map in the integration schema as well as in the dwh schema
2. Build a Job “int <tablename” to load the table in the integration schema using a TRUNCATE INSERT method according to the dellstore\_sourcetotarget\_map . Populate the CUST\_ID using a database sequence. Populate VALID\_FROM with the date 01.01.1900 and VALID\_TO with the date 31.12.9999. Populate CUST\_HID with the value from CUST\_ID.
3. Create a central job “integration” and include the “int <tablename>” job in it.
4. Build a Job “load <tablename>” which compares the dwh and the int data and uses SCD2 historization to load the data to the DWH. Use the following rules:

* A new version should be created, when one of the core content fields has changed, i.e. NAME, city, state, zip, country, region, age, income,gender)
* The only the current version has to be updated in case of a change of the address line or the contact information. (fields: address, email, phone)
* Every new version will have an unique CUST\_HID
* The valid date from of a new version should be the LOAD\_DATE
* The previous current version should be LOAD\_DATE -1 (e.g. 31.12.2017 in case of a load date of 01.01.2018)
* The most current version should have CUST\_CURRENT = ‘Y’. All other versions should have CUST\_CURRENT=’N’

1. Include the “load <tablename>” job in the central job “load”.
2. Run the job multiple times (after the first run, the table in the dwh shouldn’t change)
3. Set the LOAD\_DATE in the “dwhload\_control” to ’01.02.2018’ and update three customer records in the dellstore source database:
   1. Update name and phone of one customer
   2. Update phone and address of a second customer
   3. Update country and state of a third customer
4. Execute the full job, to see what happens for the individual customers
5. Repeated execution of the job should not change the database

## Task 9: Sales Fakt

1. Add the tables orders and orderliness to staging.
2. Create table sales\_fact with the columns mentioned in the dellstore\_sourcetotarget\_map in the integration schema as well as in the dwh schema
3. Build a Job “int <tablename” to load the table in the integration schema using a TRUNCATE INSERT method, as described in the source to target map.
4. Include the “int <tablename>” job in the central job “integration”, be careful to set the dependencies correctly.
5. Test the job in int until it works correctly
6. Build a Job “load <tablename>” which copies the data in the int schema to the dwh schema using an INSERT method.
7. Include the “load <tablename>” job in the central job “load”.
8. Run job “load <tablename” only one time. Then add a filter to the “int <tablename>” job, which processes only records, which are newer than the load date.
9. Test the new jobs by running it multiple times. The number of rows in all time tables should stay the same
10. Set the LOAD\_DATE in the “dwhload\_control” to ’01.02.2018’ and create new orders for the customer with the changed name from task 8. (be careful to use orderid and orderlineid which are max(value)+1´. Use a orderdate of 13.01.2018.

**INSERT** **INTO** public.orders

(orderid, orderdate, customerid, netamount, tax, totalamount)

**VALUES**((<max value>, '2018-01-14', id of changed customer, 313.24, 25.84, 339.08);

**INSERT** **INTO** public.orderlines

(orderlineid, orderid, prod\_id, quantity, orderdate)

**VALUES**(<max value>, see orders, 3353, 3, '2018-01-14');

1. Execute the full job and see how many rows are in int and how many rows are added to dwh.
2. Repeated execution of the job should not change the database

## Task 10: Customer Statistics

1. Create table cust\_stats with the columns mentioned in the dellstore\_sourcetotarget\_map in the dwh schema
2. Create a new job (to be included into the load job), which builds an aggregation for all customers in the cust\_stats table. Use the aggregations described in the sourcetotarget map based on the tables, which are already loaded to the dwh schema. Use a truncate\_insert method for loading the table.

## Task 10: Cleaning up

* Organize the jobs using folders
* Use context variables to replace the hardcoded database names
* Add a test and a production environment (including new docker images for
* Create new docker images for the production environment.