# Leistungsnachweis DB & DWH

HS 2018

Patrik Lengacher und Raphael Monstein

## 1. Laden und studieren der bestehenden DB

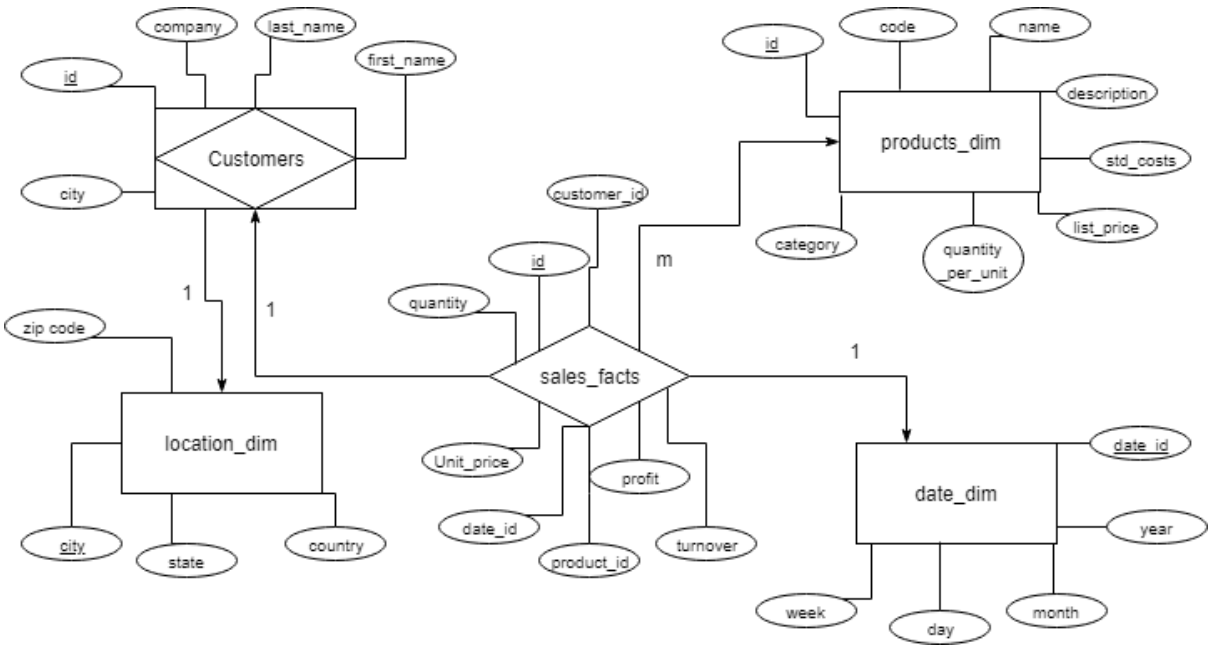
Die Datenbank der Aufgabestellung wurde geladen mit den SQL Skripten geladen und studiert.

## 2. Festlegen, welche Auswertungen mit dem DWH ermöglicht werden sollen

In einem ersten Schritt wurde festgelegt, welche Auswertungen mindestens gemacht werden sollen. Dabei wurden folgende Auswertungen definiert:

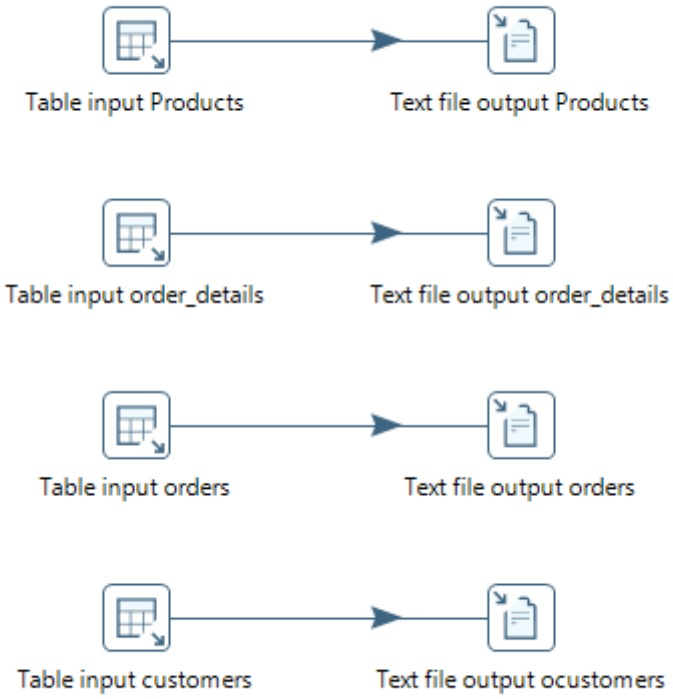
- Welche Produkte wurden am meisten gekauft?
- Welche Produkte hatten den höchsten Umsatz?
- Welche Produkte haben die höchste Marge?
- Woher stammen die Kunden mit dem höchsten Umsatz?

Um diese Auswertungen zu ermöglichen, wurde ein minimalistisches Schneeflockenschema ausgewählt. Damit die Fragestellung für die Fachanwender einfach ausgewertet werden können, werden die Daten nach dem folgenden ERM Diagram modeliert. Es werden neue Dimensionstabellen erstellt, welchen es erlauben Roll-Up und Drilldown Operationen durchzuführen.



## 3. Exportieren der DB in CSV Dateien

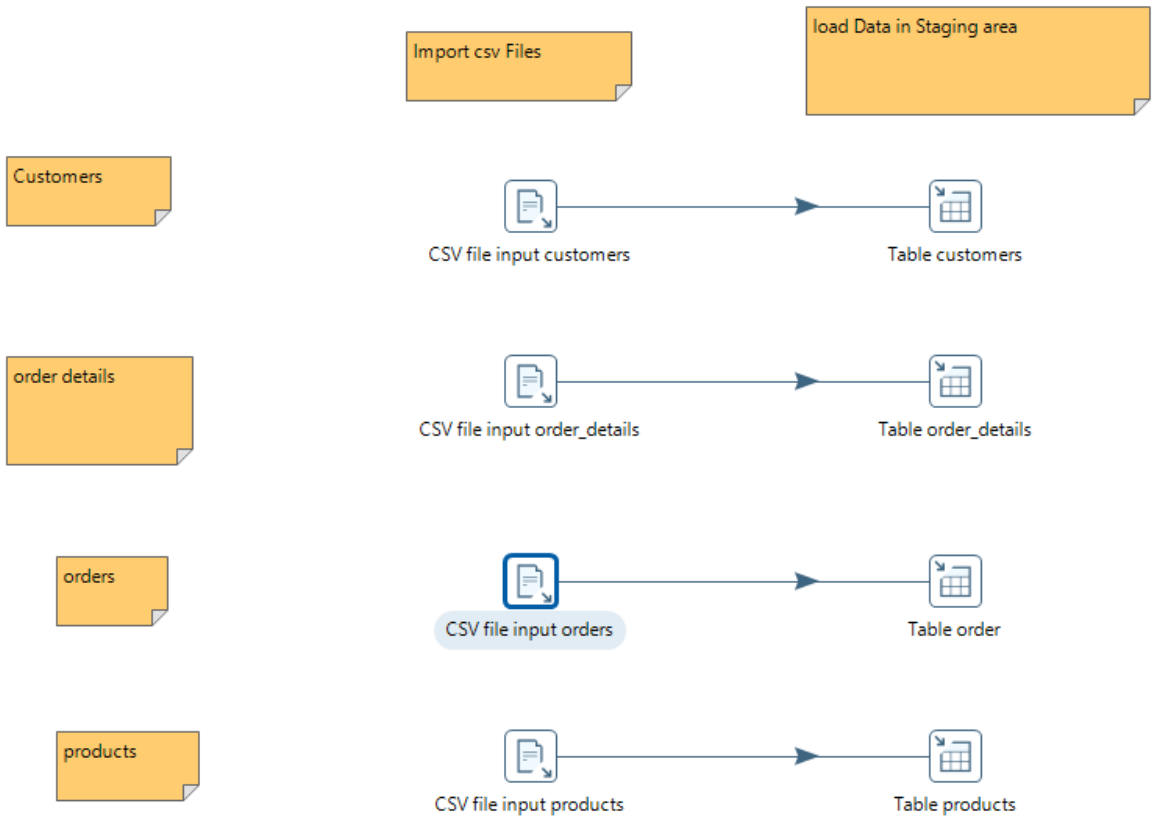
Die Daten in der Datenbank wurden in Pentaho Data Integration (PDI) geladen und von dort aus in CSV exportiert. Dieser Schritt ist nicht notwendig. Er ermöglicht es aber, das System auf verschiedenen Rechnern aufzusetzen. Durch das Erzeugen der Exporttabellen stellen wir sicher, dass alle die gleichen Inputtabellen und Formatierungen haben. Die Entwicklung der Transformation erfolgt straight-forward.  Ein Screenshot ist nachfolgend zu finden.




Dabei wurden nur die Tabellen der Datenbank in die CSV Dateien exportiert, die anschliessend auch benötigt werden.

## 4. Importieren der Daten in die Staging Area

Die Daten in den CSV Dateien wurden anschliessend eins zu eins in PDI in eine leere Datenbank namens *staging* importiert. Es wird versucht die Prozesse möglichst einfach und nachvollziehbar zu halten. So wurden in den PDI-Transformationen viele Notizen hinterleget und es wurde darauf geachtet, dass die Reihenfolge der Quell und Zieldaten möglichst eingehalten wird. Im ersten Import-Prozess in die Staging-Area werden noch keine Transformationen durchgeführt wie unten visualisiert wird.



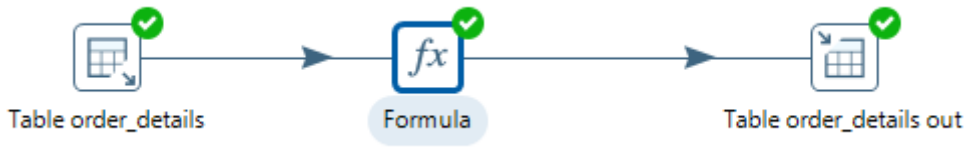

Da die importierten Daten noch, wie in der Aufgabestellung beschrieben, Qualitätsprobleme haben, mussten diese transformiert werden. Dies wurde in nächsten Schritt gemacht.

## 5. Beheben der Datenqualitätsprobleme

Die Datenqualitätsprobleme betrafen die Tabellen *order_details* und *products*.

### *Order_details*

In *order_details* wurden die alle Discounts als negativ erfasst. Das umwandeln in positive Werte kann relativ einfach in PDI gemacht werden. Dazu kann ein Formula-Block verwendet werden.




Die verwendete Operation im Formula-Block ist die *ABS* Funktion.

### *Products*

Die Tabelle *products* hatte das Problem, dass irrtümlicherweise gewisse Einträge doppelt erfasst wurden. Um herauszufinden, wie diese Einträge detektiert werden könne, wurden ein paar SQL Abfragen geschrieben. Als erstes wurde geschaut, ob *product_code* und *product_name* eindeutig sind.

```
SELECT COUNT(DISTINCT product_code), COUNT(DISTINCT id),
COUNT(DISTINCT product_name), COUNT(DISTINCT id)
FROM northwind.products;
```

| COUNT(DISTINCT product_code) | COUNT(DISTINCT id) | COUNT(DISTINCT product_name) | COUNT(DISTINCT id) |
|---|---|---|---|
| 43 | 47 | 45 | 47 |

Dies deutet darauf hin, dass es Duplikate in beiden Spalten gibt. Um einen besseren Eindruck zu bekommen, wurden die Duplikate angezeigt.

```
SELECT product_name, product_code, COUNT(product_code)
FROM northwind.products
GROUP BY product_code
HAVING COUNT(product_code) > 1;
```

| product_name | product_code | COUNT(product_code) |
|---|---|---|
| Northwind Traders Granola | NWTC-82 | 2 |
| Northwind Traders Corn | NWTCFV-93 | 2 |
| Northwind Traders Boysenberry Spread | NWTJP-6 | 2 |
| Northwind Traders Chicken Soup | NWTSO-99 | 2 |

```
SELECT product_name, product_code, COUNT(product_name)
FROM northwind.products
GROUP BY product_name
HAVING COUNT(product_name) > 1;
```

| product_name | product_code | COUNT(product_name) |
|---|---|---|
| Northwind Traders Corn | NWTCFV-93 | 2 |
| Northwind Traders Chicken Soup | NWTSO-99 | 2 |

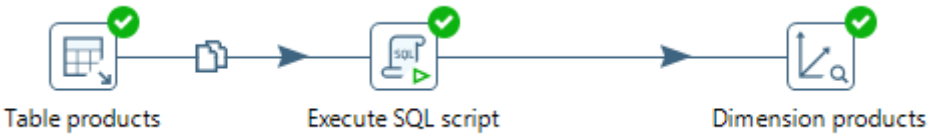
Weitere Abfragen haben ergeben, dass die Duplikate im *Product_code* unterschiedliche Beschreibungen und auch sonst unterschiedliche angaben haben, wie im nachfolgenden Screenshot dargestellt wird.

| supplier_ids | id | product_code | product_name | description | standard_cost | list_price | reorder_level | target_level |
|---|---|---|---|---|---|---|---|---|
| 1 | 21 | NWTBGM-21 | Northwind Traders Scones | NULL | 7.5000 | 10.0000 | 5 | 20 |
| 1 | 85 | NWTBGM-85 | Northwind Traders Brownie Mix | NULL | 9.0000 | 12.4900 | 10 | 20 |
| 1 | 86 | NWTBGM-86 | Northwind Traders Cake Mix | NULL | 10.5000 | 15.9900 | 10 | 20 |
| 1 | 82 | NWTC-82 | Northwind Traders Granola | NULL | 2.0000 | 4.0000 | 20 | 100 |
| 1 | 97 | NWTC-82 | Northwind Traders Hot Cereal | NULL | 3.0000 | 5.0000 | 50 | 200 |
| 10 | 48 | NWTCA-48 | Northwind Traders Chocolate | NULL | 9.5625 | 12.7500 | 25 | 100 |

Dabei wird ersichtlich, dass z.B. das Produkt NWTC-82, welches zwei Mal mit dem gleichen Produkt Code aufgeführt ist, wahrscheinlich kein Duplikat ist. Aus diesem Grund wurden nur die Tupel entfernt, welche den gleichen Produktenamen haben. Dies wurde mit der folgenden SQL Abfrage gemacht:

```
SELECT *
FROM northwind.products
WHERE id IN
(
    SELECT MIN(id)
    FROM northwind.products
    GROUP BY product_name
);
```

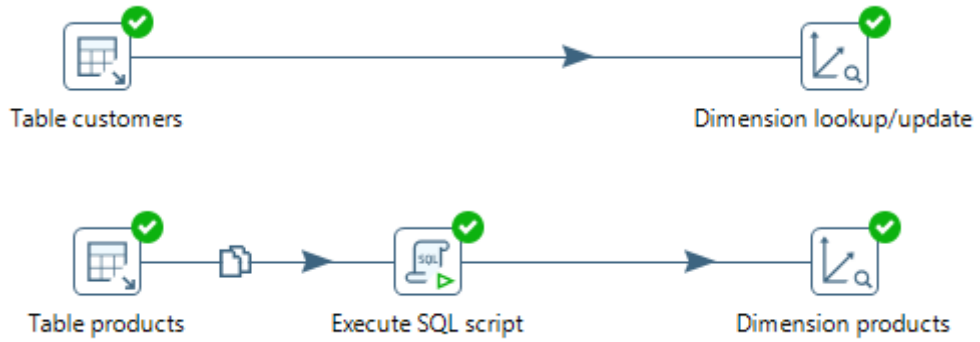
Nachdem die Abfrage in der MySQL Workbench getestet wurde, wurde die Transformation in PDI integriert.



Im Gegensatz zu der Transformation von *order_details* wurde die Transformation von Product explizit mit SQL gelöst um die Möglichkeit von generischen Transformationen mit PDI weiter zu erkunden. Diese Transformation ermöglicht es die ganze Mächtigkeit von SQL auszunutzen und nicht auf vorgefertigte Use-Cases von PDI zurückzugreifen. Dies biete auch die Möglichkeit durch Fachanwender entwickelte Abfragen oder Transformationen einfach in PDI zu integrieren. So sind auch alle Parameter, welche in der Transformation benutzt werden im SQL-Skript zu finden.

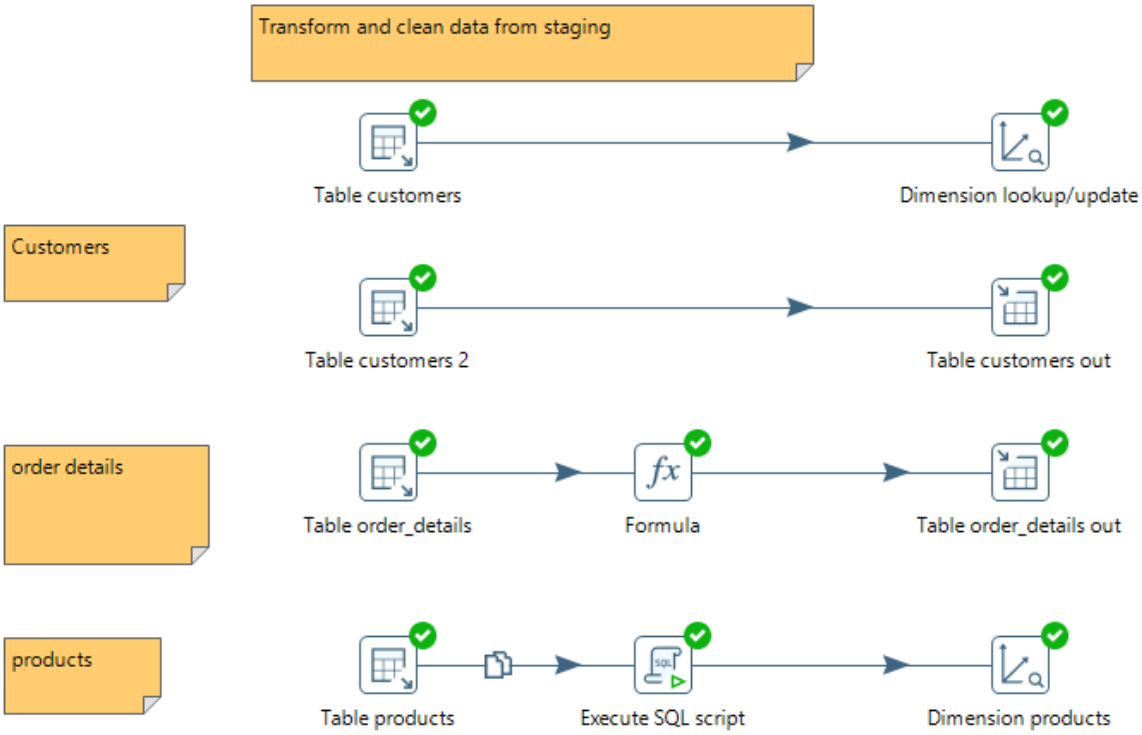
## 6. Erzeugen der Dimensionen

Wie in einer der Übungen gelernt, wurden die Dimensionen in PDI mit Dimension lookup/update Block erstellt.



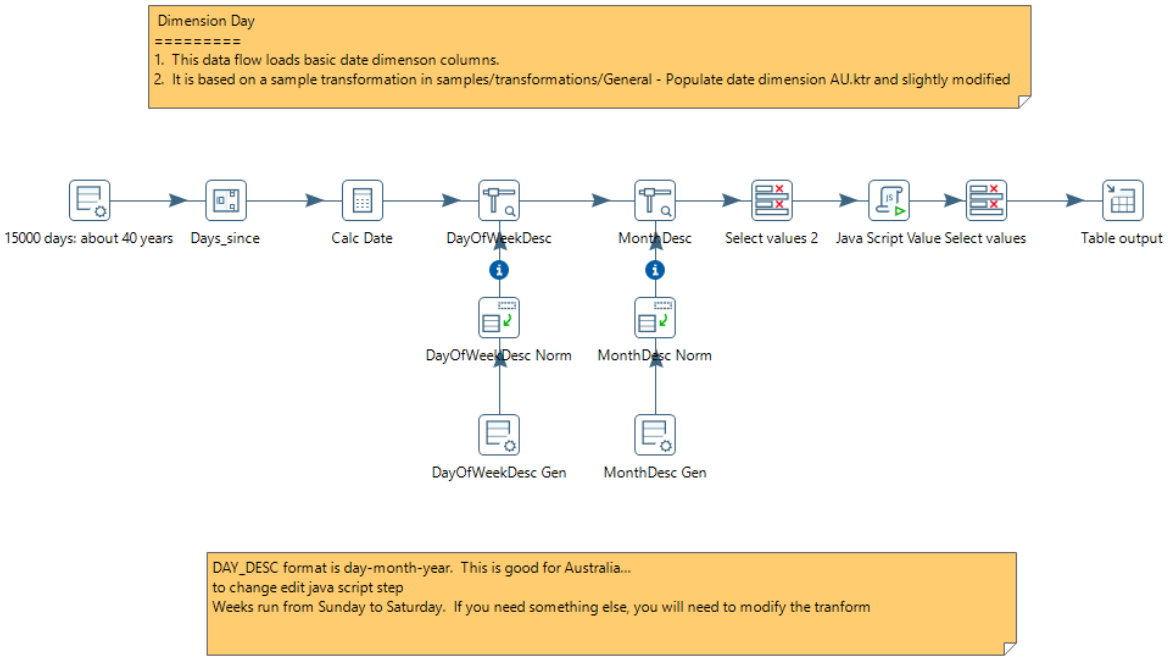

Dabei wurde der Einfachheit halber eine SCD Type 1 angewendet.

Eine Übersicht über alle Transformationen zur Behebung der Qualitätsprobleme und das Erzeugen der Dimensionen ist nachfolgend abgebildet. Die Tabelle Customers wird hier bewusst doppelt verwendet. Durch das im ERM-Diagramm definierte Schema entsteht die *location_dim*-Tabelle wie auch die *customers*-Tabelle aus der northwind.customers-Tabelle. Durch die konsistente Benutzung der Reihenfolge in den Transformationen und die Notizen bleibt auch diese Transformation sehr gut lesbar.


Transform and clean data from staging
Table customers
Dimension lookup/update
Customers
Table customers 2
Table customers out
order details
Table order_details
Formula
Table order_details out
products
Table products
Execute SQL script
Dimension products

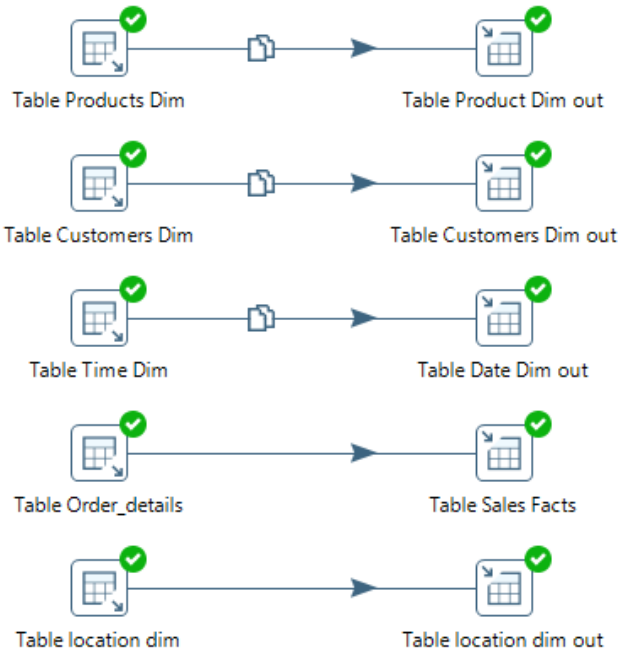
## 7. Erzeugen einer Datums-Dimension

Eine zusätzliche Dimension für das Datum wurde separat erzeugt. Die Idee ist, dass damit einfacher Abfragen zu gewissen Zeiträumen, z.B. nach Quartal, durchgeführt werden können. Diese Neuerstellte Zeit-Dimension lässt sich beim Ausbau dieses Data-Marts sehr einfach wiederverwenden sobald neue Tabellen mit einer Zeitlichen Abhängigkeit in das System eingefügt werden. Durch diese Modularität bleibt das System sehr flexibel und einfach einsetzbar. Der Block, welcher dazu verwendet wird, ist in den Beispielen von PDI enthalten und wurde leicht modifiziert. Der ursprüngliche Block kann im PDI Installationsverzeichnis unter *samples\transformations\ General - Populate date dimension AU.ktr* gefunden werden.



## 8. Erzeugen des Schemas im DWH

Alle bisher aufgezeigten Transformationen wurden in der Staging Area durchgeführt. In diesem Schritt werden die Daten in eine neue Datenbank mit dem Namen *DWH* gefüllt.



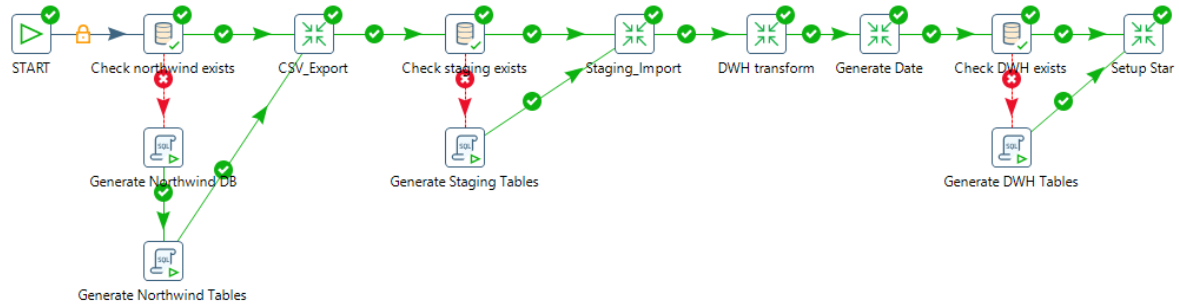

Bei diesem Schritt wurden nicht alle Attribute ins DWH kopiert, sondern nur noch diese, welche auch (potentiell) für spätere Abfragen benötigt werden. Besonders hervorzuheben ist, wie die Tabelle *sales_facts* erzeugt wird. Im Block *Table Order_details* wird folgende SQL Abfrage durchgeführt:

```
SELECT ord_d.id AS id,
       ord_d.quantity AS quantity,
       ord_d.unit_price AS unit_price,
       d.DATE_SK AS date_id,
       cust.id AS customer_id,
       prod.id AS product_id,
       ord_d.unit_price*ord_d.quantity AS turnover,
       (ord_d.unit_price*ord_d.quantity) -
(prod.standard_cost*ord_d.quantity) AS profit

FROM staging.order_details_clean AS ord_d,
     staging.date_dim AS d,
     staging.orders AS ord,
     staging.customers AS cust,
     staging.products_dim AS prod

WHERE ord_d.product_id = prod.id AND
      ord_d.order_id = ord.id AND
      ord.order_date = d.day_date AND
      ord.customer_id = cust.id;
```

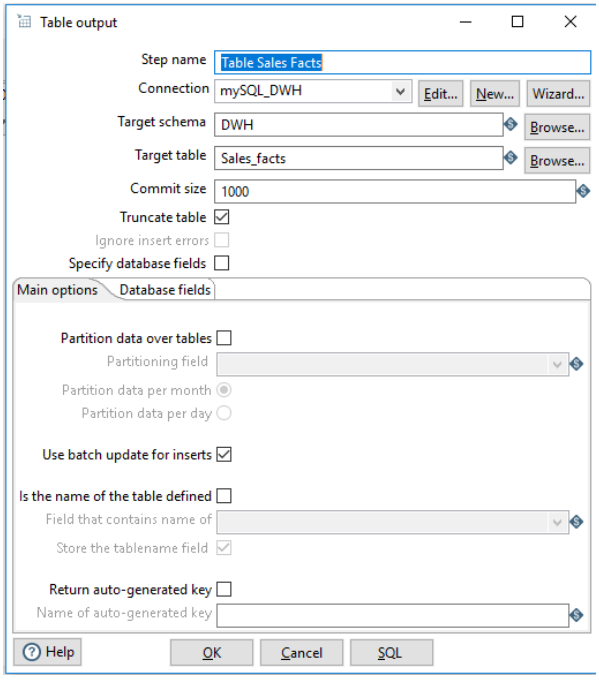
Um spätere Abfragen zu vereinfachen, werden die beiden Attribute *turnover* und *profit* erzeugt.

## 9. Erstellen eines Jobs

Damit die verschiedenen Transformationen nicht einzeln von Hand ausgeführt werden müssen, wurde ein Job erstellt.




Dieser Job führt die Transformationen in der richtigen Reihenfolge aus. Dabei wurde auch versucht, die Tabellen automatisch zu erzeugen, falls diese nicht vorhanden sind. Leider konnte dies nicht innerhalb von vernünftiger Zeit zum Laufen gebracht werden. Stattdessen müssen, falls die Datenbanken und Tabellen nicht existieren, die beiden SQL Skripts *staging.sql* und *DWH.sql* vorgängig laufen gelassen werden. Diese beide Skripts wurden mehr oder weniger mit Copy-Paste aus den PDI Befehlen erstellt, welche erzeugt werden, wenn eine Tabelle mit dem *Table output* erzeugt wird (beim drücken der Schaltfläche *SQL* in den Eigenschaften des Blocks).



Zu Beachten ist auch, dass bei jedem Erstellen der Tabellen die „Truncate Table“ Option aktiviert ist. Dadurch können die Daten des ERP-Systems nicht inkrementell geladen werden. So wird jedes Mal der komplette Datensatz ins DWH geladen. Dies ist eine sehr pragmatische Lösung welche einfach implementiert werden kann. Bei grossen ERP System jedoch kann ein solcher Workflow schnell zum Flaschenhals werden. Dies kann zur Folge haben, dass die Daten schlussendlich nicht mehr täglich sondern nur noch einmal wöchentlich während des Wochenendes geladen werden können.

## 10. Abfragen

Die oben definierten Abfragen können wurden alle direkt in SQL umgesetzt. Insbesondere die Zeit Dimension bietet eine sehr einfache Möglichkeit Roll-Up und Drilldown Operationen auf den Kennzahlen durchzuführen.

Die Resultate der Abfragen können mit der Datei *std_queries.sql* ausgeführt werden. Auf eine Auflistung der Resultate wird in diesem Bericht verzichtet.

Durch die neu geschaffene Struktur lassen sich die Daten auch in ein Excel-File exportieren und untersuchen. So bieten die Pivot-Tabellen eine gute Möglichkeit Fragestellungen einfach und schnell zu beantworten. Als Beispiel wurde der Turnover der verschiedenen Companies in den Staaten der USA veranschaulicht.

Umsatz der Companies nach Staaten

So erkennt man schnell, dass der höchste Umsatz 15432.5 aus dem Staat TN von Company BB erzielt wird.

| Summe von turnover | Spaltenbeschriftungen | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Zeilenbeschriftungen** | **CA** | **CO** | **FL** | **ID** | **IL** | **NV** | **NY** | **OR** | **TN** | **UT** | **WA** | **WI** | **(Leer)** | **Gesamtergebnis** |
| Company A | | | | | | | | | | | 2410.75 | | | 2410.75 |
| Company AA | | | | | | 1505 | | | | | | | | 1505 |
| Company BB | | | | | | | | | 15432.5 | | | | | 15432.5 |
| Company C | 2550 | | | | | | | | | | | | | 2550 |
| Company CC | | 2905.5 | | | | | | | | | | | | 2905.5 |
| Company D | | | | | | | 4569 | | | | | | | 4569 |
| Company F | | | | | | | | | | | | 8007.5 | | 8007.5 |
| Company G | | | | 13800 | | | | | | | | | | 13800 |
| Company H | | | | | | | | 4683 | | | | | | 4683 |
| Company I | | | | | | | | | | 3786.5 | | | | 3786.5 |
| Company J | | | | | 1412.5 | | | | | | | | | 1412.5 |
| Company K | | | 1019.5 | | | | | | | | | | | 1019.5 |
| Company L | | | | | | 1190 | | | | | | | | 1190 |
| Company Y | | | | | 860 | | | | | | | | | 860 |
| Company Z | | | 3625.25 | | | | | | | | | | | 3625.25 |
| (Leer) | | | | | | | | | | | | | | |
| **Gesamtergebnis** | **2550** | **2905.5** | **4644.75** | **13800** | **2272.5** | **2695** | **4569** | **4683** | **15432.5** | **3786.5** | **2410.75** | **8007.5** | | **67757** |

Welche Firmen haben den höchsten Umsatz?

Durch die Struktur des Dimensionalen Modelles lassen sich Bsuiness Intelligence Abfragen auch sehr einfach per SQL definieren wir in den untenstehenden Analysen darstellen. Die SQL Befehle sind einfach zu lesen und zu entwickeln. Diese Abfrage wurde mit dem folgenden SQL erstellt.

*SELECT cust.company AS Company,*

*SUM(sales.unit_price - prod.standard_cost) AS Margine*

*FROM dwh.sales_facts AS sales,*

*dwh.customers_dim AS cust,*

*dwh.product_dim AS prod*

*WHERE cust.id = sales.customer_id AND*

*prod.id = sales.product_id*

*GROUP BY cust.id*

*ORDER BY SUM(sales.unit_price - prod.standard_cost) DESC;*

| | Company | Margine |
|---|---|---|
| ▶ | Company D | 45.55 |
| | Company F | 40.625 |
| | Company BB | 30.0125 |
| | Company H | 29.674999999999994 |
| | Company A | 21.589999999999996 |
| | Company Z | 18.6 |
| | Company J | 18.04 |
| | Company CC | 16.4375 |
| | Company L | 16 |
| | Company I | 15.987499999999997 |
| | Company C | 14.912500000000001 |
| | Company G | 11.5 |
| | Company K | 11.49 |
| | Company Y | 8 |
| | Company AA | 4 |

In welchem Quartal hat northwind den höchsten Einnahmen erzielt?

Diese Abfrage wurde mit dem untenstehenden Code ausgeführt.

*-- in which quarter did we generate the biggest revenue*

*SELECT date.quarter_number AS Quarter,*

*date.YEAR_NUMBER AS Year,*

*SUM(sales.quantity*sales.unit_price) AS Revenue*

*FROM dwh.sales_facts AS sales,*

*dwh.date_dim AS date*

*WHERE date.date_sk = sales.date_id*

*GROUP BY date.year_quarter_name*

*ORDER BY SUM(sales.quantity*sales.unit_price) DESC;*

| | Quarter | Year | Revenue |
|---|---|---|---|
| ▶ | 1 | 2006 | 38686.75 |
| | 2 | 2006 | 29070.25 |

## 11. Fazit

Mit einem sauber hergeleiteten und korrekt erstellen dimensionale Modell lassen sich einfach geschäftsrelevante Fragestellungen beantworten und diese nach gängigen Gruppierungsmethoden zusammenfassen. Durch die verschiedenen Dimensionen, welche bei jedem Anwendungsfall anders sind, können mittel Roll-Up und Drilldown verschieden Sichten auf die Problemstellung einfach und reproduzierbar erstellt werden. Darüber hinaus können mit einem solchen Model SQL-Mächtige Anwender aber auch Excel-User verschiedenste Auswertungen machen und so dem Business einen Mehrwert generieren.