CAS Information Engineering

## Leistungsnachweis für Modul DB & DWH

**Gruppenmitglieder**:

Andreas Fischer

Bernd Novotny

Tobias Schieferdecker

Inhaltsverzeichnis:

Leistungsnachweis für Modul DB & DWH 1

1. Übersetzung des gegeben ERMs in den uns bekannten Dialekt 2

2. Auslagern der relevanten Daten in csv-Dateien 5

3. Erstellung der staging-area 6

4. Reparieren der korrupten Daten 8

5. Konzipierung des Sternschemas für das DWH 10

6. Erstellung und Befüllen des DWH 12

7. Auswertungen auf Basis des DWH 12

8. Fazit und Schlusswort 12

# **Übersetzung des gegeben ERMs in den uns bekannten Dialekt**

Es soll die gegebene Datenbank “Northwind” genommen und gemäss einem neu anzufertigenden Schema migriert werden. Das Schema soll im ERM-Dialekt von H.-W. Buff ausgeführt werden.

## Ausgangsdaten

Zur Erzeugung der Datenbank liegen die beiden SQL-Skripte

* Northwind-Database.sql
* Northwind-Daten.sql

vor. Sie können in MySQL-Workbench geöffnet und ausgeführt werden.

Nach erfolgreichem Durchlauf liegt die Datenbank mit 20 Entitäten und einigen Testdaten eingelesen vor.

Das Schema der ERP-Datenbank “Northwind” (siehe Bild 1) liegt in der “Bachman's crow-foot notation” vor. Sie kann z.B. erzeugt werden durch das MySQL-Tool Workbench.

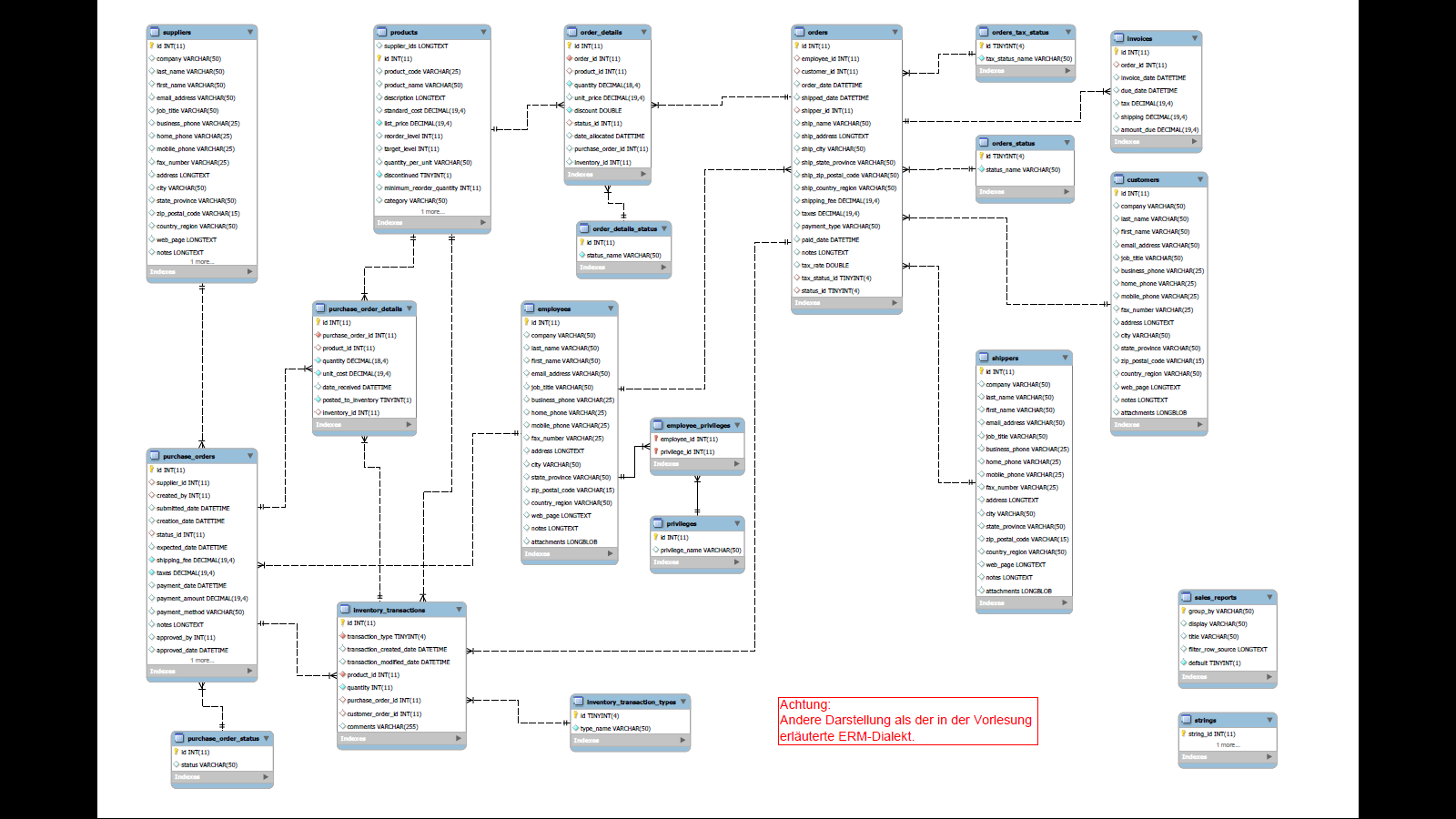


Bild : Schema der ursprünglichen ERP-Datenbank “Northwind”.

Die Relationen sind hier durch Verbindungen mit speziellen Symbolen dargestellt, die die Kardinalitäten angeben. Bild 2 listet diese auf.

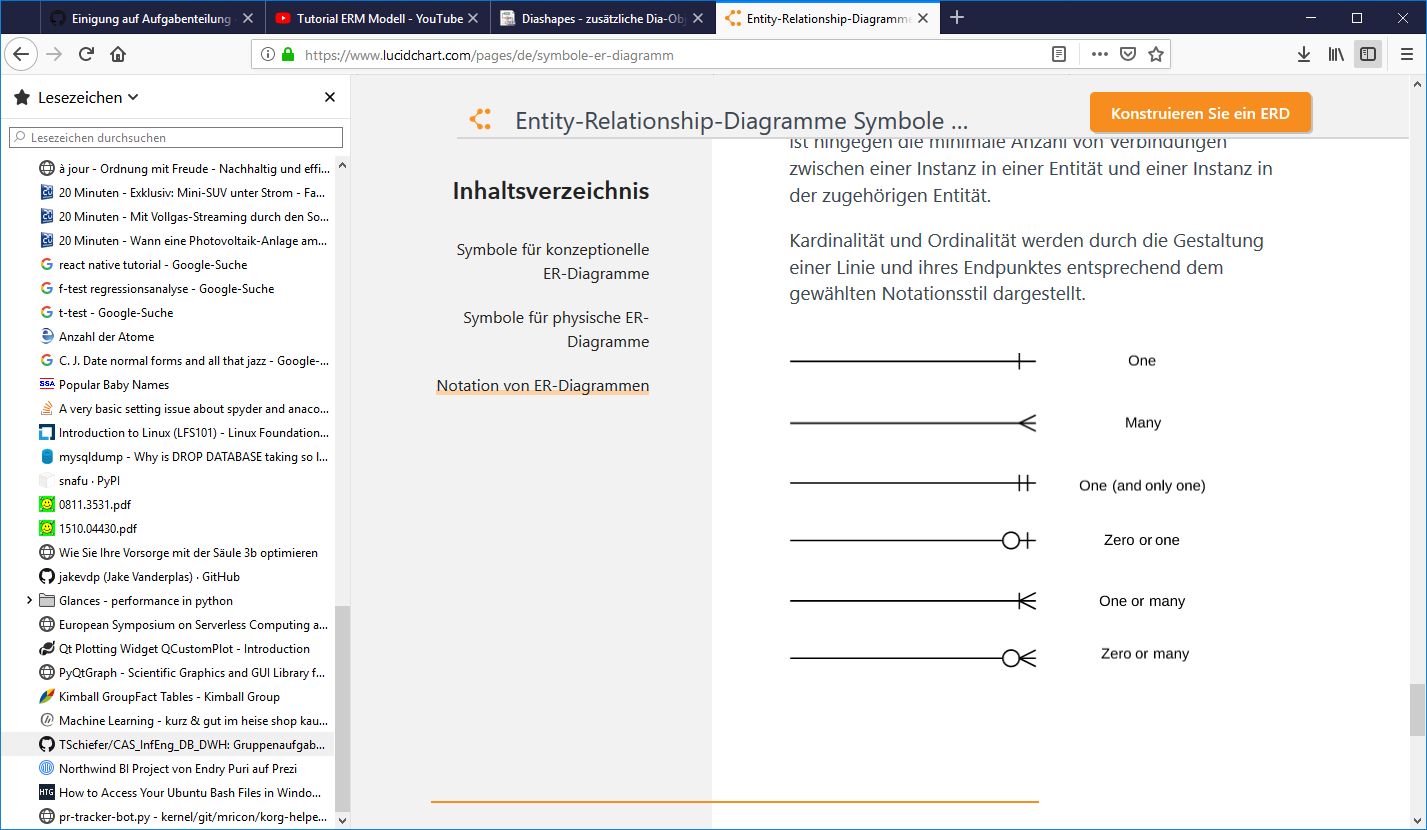


Bild : Bedeutung der verschiedenen Symbole in Bachmann's crow foot Notation

Um das neue ERM-Diagramm zu zeichnen wird das Tool Dia verwendet. Siehe dazu <https://de.wikipedia.org/wiki/Dia_(Software>) oder <http://dia-installer.de/index.html.de> .

## Entity-Relationship-Modell (ERM)

Bild 3 zeigt das neu erstellt ERM-Diagramm in Buff-Notation.

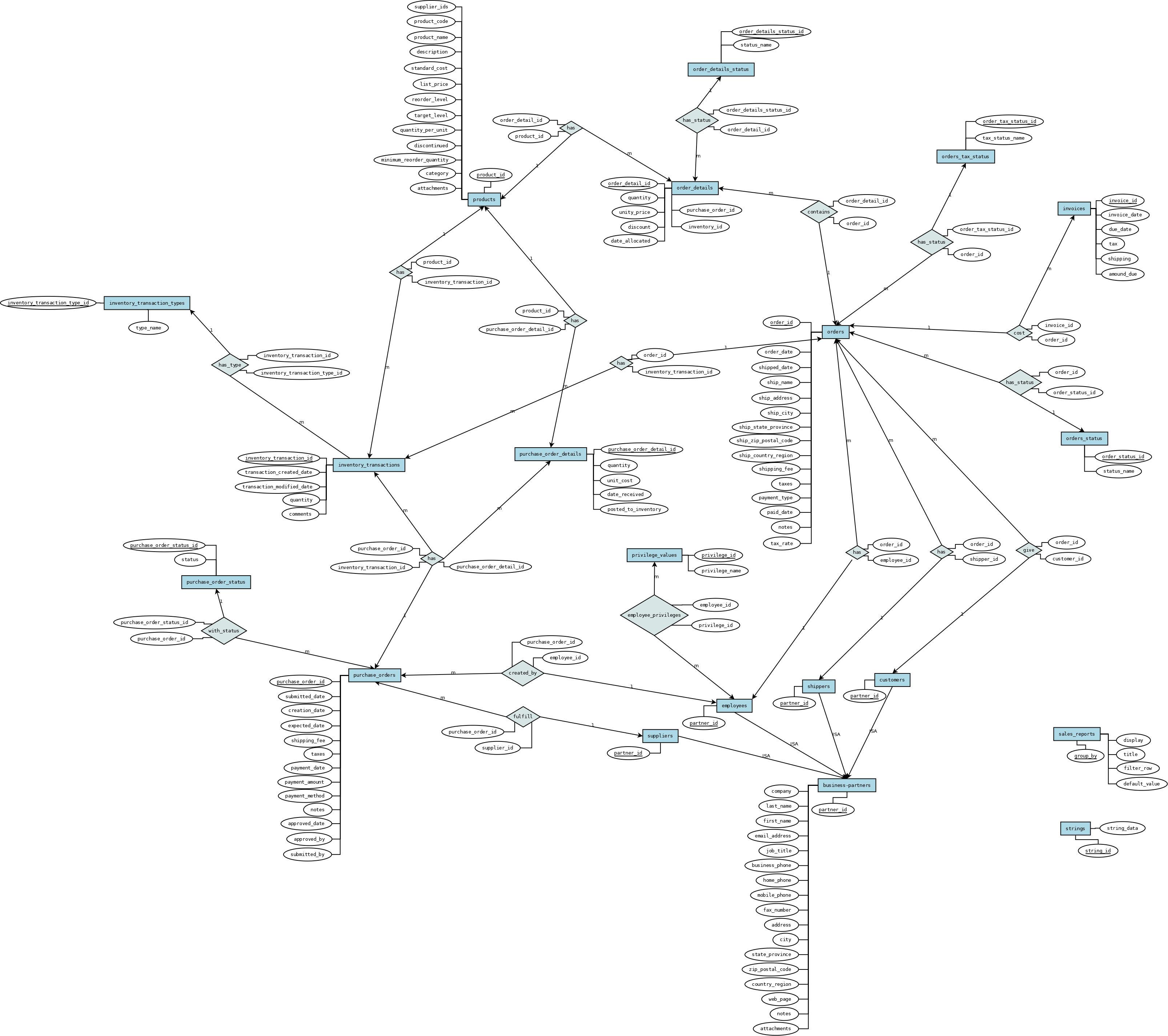


Bild : Übersetzung des Northwind-Schemas in den Dialekt von Buff.

Speziell wurde versucht ISA-Beziehungen einzubauen, da die Entitäten “suppliers”, “employees”, “shippers” und “customers” identische Attribute aufweisen. Dazu wurde neu die Entität “business-partners” eingefügt, vergleiche Bild 4.

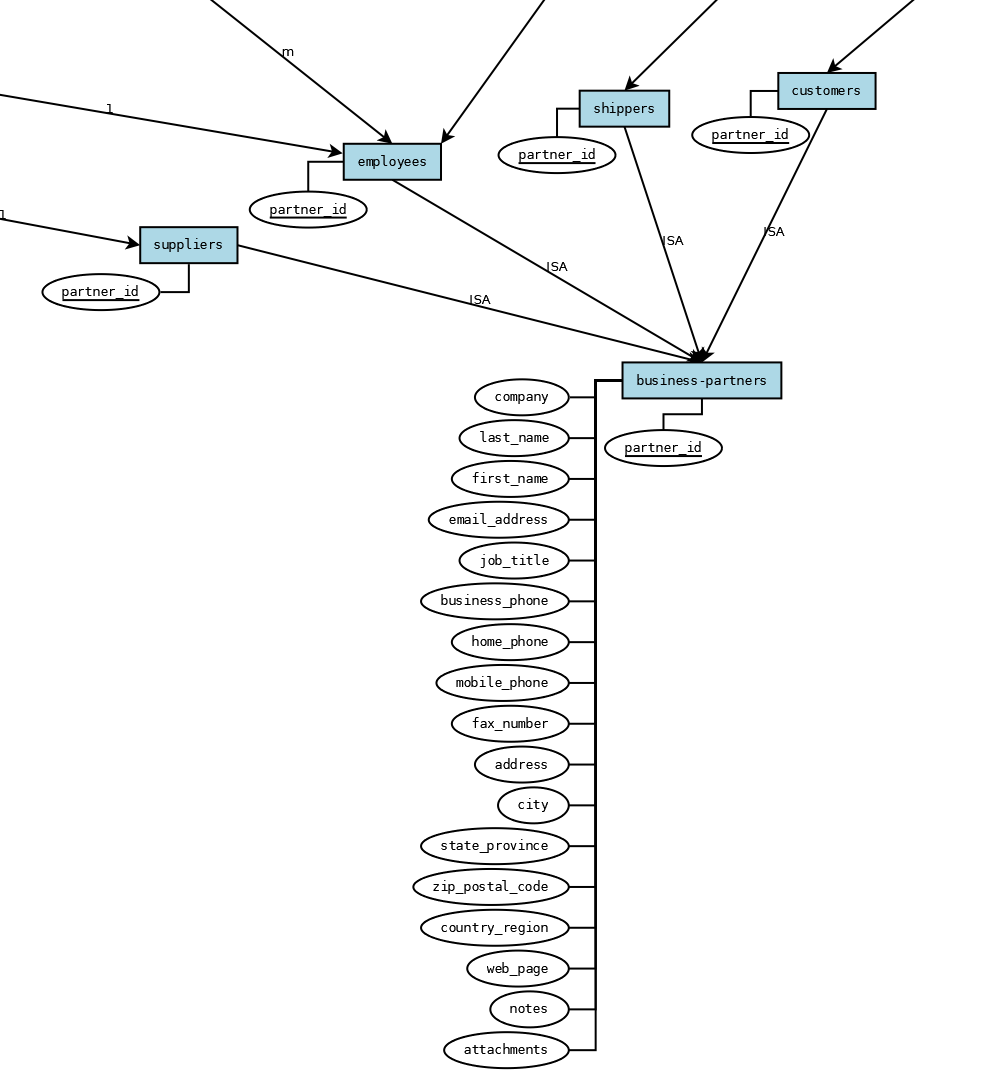


Bild : Die Entität "business-partners".

Beobachtungen:

* Das Diagramm ist grafisch anspruchsvoller und grösser als das ursprüngliche. Vor allem wenn viele Attribute vorhanden sind.
* Es sind mehr Elemente nötig, denen gute Namen gegeben werden müssen. Dies ist nicht immer einfach.
* Das gewählte Tool ist nicht ideal. Das Handling und die Darstellung könnten verbessert werden. Vor allem das Zeichnen der Relationen und deren Beschriftung sollte besser sein.
* Den Schlüsseln der Entitäten müssen eindeutige Bezeichnungen gegeben werden, da sie nicht nur in den Entitäten sondern auch in den Relationen verwendet werden.

## SQL Skripte

Zur Umsetzung des ERM-Diagramms in eine Datenbank werden SQL-Skripte benötigt. Als Basis werden die ursprünglichen Skripte genommen und angepasst.

Zur Erzeugung der Datenbank muss als erstes immer das Skript **Northwind2-Database.sql** ausgeführt werden. Es löscht eine schon vorhandene Datenbank und baut dann eine neue wieder auf.

Danach können die Test-Daten eingelesen werden durch das Skript **Northwind2-Data.sql.**

Beobachtungen:

* Die ursprünglichen Skripte wurden scheinbar aus der Datenbank im MySQL-Workbench exportiert. Darin sind die Bezeichner alle “gequoted” mit “backticks”. Dies wurde in den neuen Skripten nicht mehr beibehalten.
* Die Schlüssel wurden wie im Diagramm konsequent mit Entitätsnamen und angehängtem “\_id” bezeichnet.
* Die Reihenfolge beim Einlesen der Daten ist wichtig. Daten auf die später verwiesen wird, müssen schon in der Datenbank sein. Die Constraints werden sonst nicht erfüllt.
* In den ursprünglichen Daten sind einzelne Records vorhanden, die auf Entitäten verweisen sollten aber mit NULL gefüllt. Diese konnten erst eingelesen werden, als die NULL-Werte durch vorhandene Fremdschlüssel ersetzt wurden.
* Da die Entitäten “suppliers”, “employees”, “shippers” und “customers” nur noch aus Schlüsseln bestehen und eine ISA-Beziehung zu der Entität “business\_partners” haben, mussten die Schlüssel angepasst werden.

Die beiden Skripte laufen ohne Fehlermeldungen oder Warnungen durch und erzeugen bzw. füllen die Datenbank.

# **Auslagern der relevanten Daten in csv-Dateien**

Nachdem die Northwind-Datenbank mit der uns zur Verfügung gestellten Datei “Northwind-Database.sql” mittels MySQL-Workbench erzeugt und im Folgenden im selben Tool mittels der Datei “Northwind-Data.sql” befüllt wurde, haben wir uns der Aufgabe gewidmet, die für unsere späteren Auswertungen relevanten Daten in csv-Dateien auszulagern.



Bild 5 Der Knopf für den Export einer Datentabelle aus MySQL-Workbench ist rot umrandet.

Für diesen Schritt haben wir kein Skript verwendet. Wir haben vielmehr einfach die Funktionalität von MySQL-Workbench verwendet haben, auf Knopfdruck ein csv-File aus der gerade betrachteten Tabelle zu erzeugen (siehe Bild 5). Die Tabellen, die uns für spätere Auswertungen interessant schienen, waren:

* “customers”
* “products”
* “orders”
* “order\_details

Hierbei wurden an dieser Stelle keine Einschränkung der Daten vorgenommen, sondern das dem SQL-Befehl

“SELECT \* FROM ‘TABLE’ ;” (mit ‘TABLE’ gleich einer der vier von uns ausgesuchten Tabellen) entsprechenden Resultat in das csv-Format exportiert. Die entsprechenden csv-Files sind dem zip-File “skripte\_und\_daten\_CAS\_DB\_DWH.zip” beigefügt, das im Email-Anhang mitverschickt wurde.

# **Erstellung der staging-area**

Zunächst ein paar Worte zu unserem Umgang mit der staging-area: Zur effizienteren Arbeitsaufteilung haben wir die Aufgaben der Einrichtung der staging-area und der Erzeugung des multidimensionalen Modells getrennt. Aus Gründen der Einfachheit und der Möglichkeit, schneller mit der Arbeit zu beginnen, haben wir anfangs die Daten für die verschiedenen Transformationen (Reparatur der korrupten Daten, Befüllen des multidimensionalen Modells, ...) aus csv-Dateien eingelesen und teils auch wieder als csv-Dateien rausgeschrieben. Die staging-area bestand somit letztlich aus dem Filesystem.

Aus zeitlichen Gründen haben wir darauf verzichtet, dies noch so umzustellen, dass die staging-area in Form einer Datenbank implementiert würde, d.h. dass das Einlesen – bis auf den einen Schritt zur Befüllen der staging-area Datenbank – immer aus Datenbanktabellen vorgenommen würde.

Wir wollen aber gerne demonstrieren, wie theoretisch die Integration der Daten über die Datenbank ausgesehen hätte. Dazu haben wir die staging-area Datenbank erstellt und mit den notwendigen Tabellen befüllt. Nur eben die Integration in den Daten-Flow ist nicht gegeben.

Es folgt eine kurze Erläuterung, wie unsere staging-area-Datenbank erzeugt und befüllt wird.

1. Das Skript zur Erzeugung der leeren Datenbank ist in MySQL-Workbench auszuführen und lautet “create\_staging\_area\_schema.sql”. Bei der Ausführung des Skripts werden die verschiedenen Tabellen mitsamt ihren Variablen erzeugt, die Reihen bleiben dabei jedoch leer (siehe Bild 6). An dieser Stelle werden schon nicht mehr alle Variablen der ursprünglichen Tabellen übernommen, sondern nur noch jene für die späteren Abfragen notwendigen.

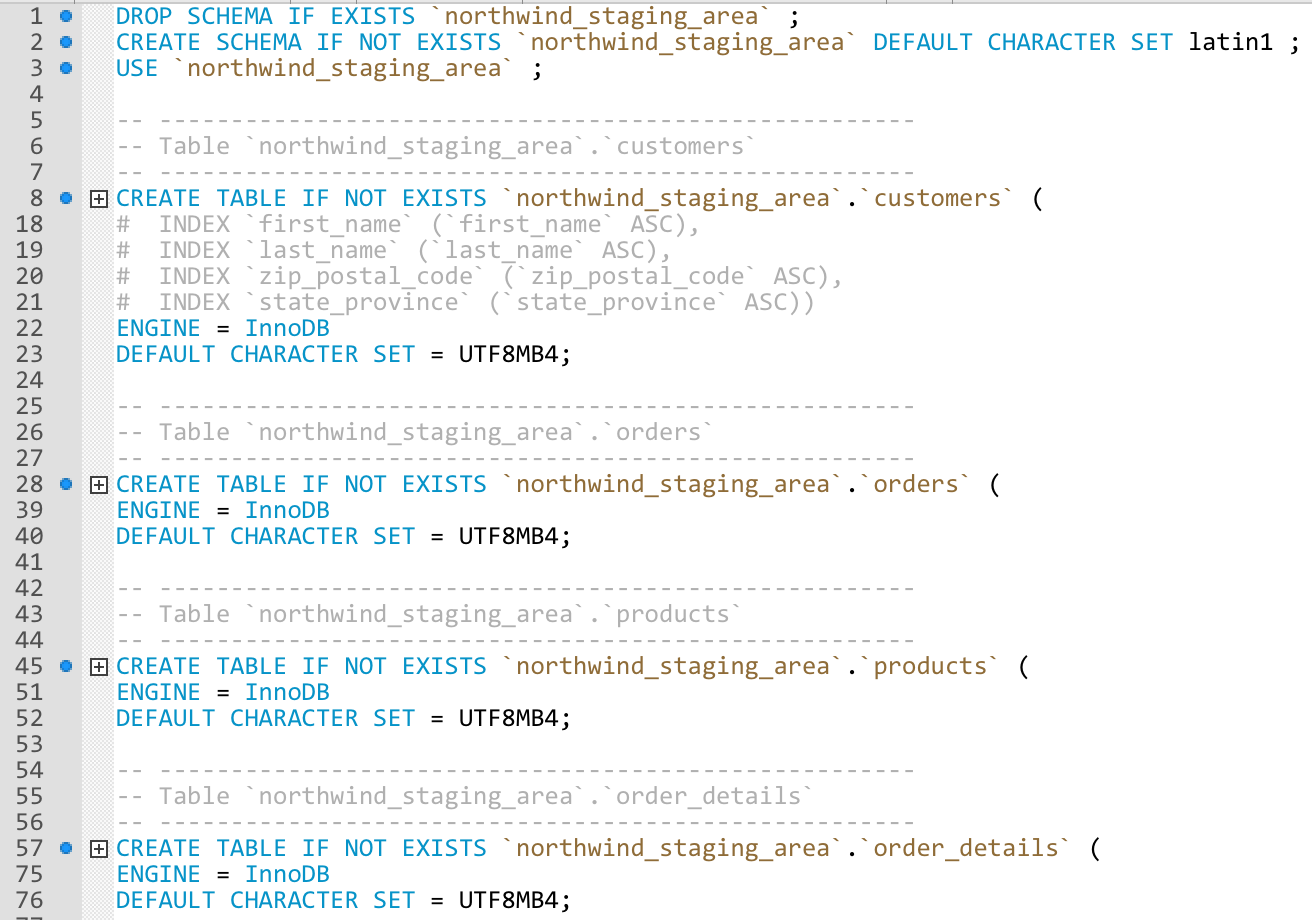


Bild 6: Skript "create\_staging\_area\_schema.sql". Für eine bessere Übersicht sind die Abschnitte für die Erzeugung der einzelnen Spalten zusammengeklappt.

1. Das Befüllen der Datenbank erfolgt mittels des Pentaho Data Integration (PDI) Spoon Skripts “add\_data\_to\_staging\_area.ktr” (siehe Bild 7). Jedes der csv-Files wird einzeln mit einem “CSV Input”-Schritt eingelesen und jeweils mit einem “Insert / Update”-Schritt in die Datenbank geschrieben. Es hat sich hierbei gezeigt, dass durch die Einschränkung durch die Verlinkung der Tabellen untereinander durch Schlüssel-Fremdschlüssel-Beziehungen, das gleichzeitige Befüllen der Tabellen eine Fehlermeldung provoziert. Um dies zu vermeiden, muss man zunächst die Verbindung zwischen “read orders” und “Insert orders into staging area” sowie diejenige zwischen “read order details” und “Insert order details into staging area” deaktivieren und die Spoon-Transformation laufen lassen. Nachdem dies erfolgreich geschehen ist, kann man die beiden deaktivierten Verbindungen wieder aktivieren und die Transformation erneut starten. Danach ist die staging-area Datenbank gefüllt.

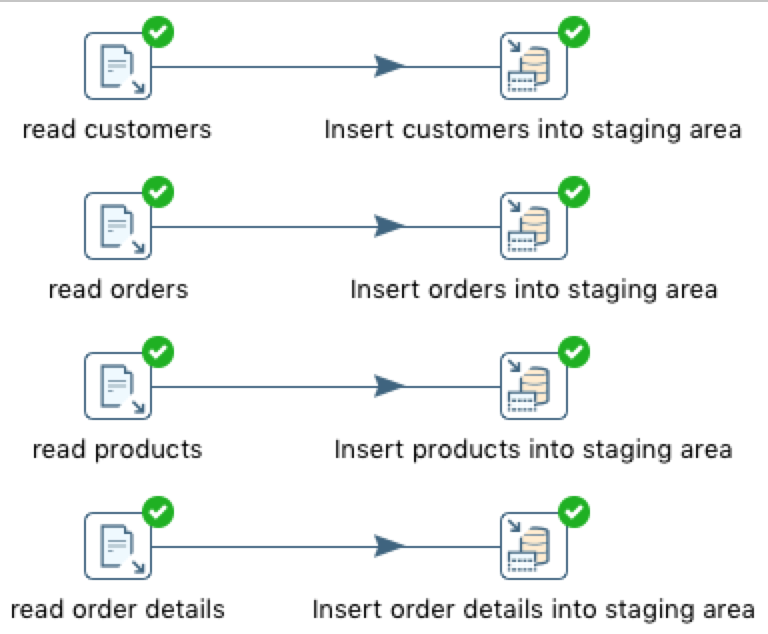


Bild 7: Transformation in Spoon zum Befüllen der staging-area-Datenbank nach erfolgreicher Ausführung.

# **Reparieren der korrupten Daten**

Um – wie in der Aufgabenstellung angegeben – herauszufinden, inwiefern sich die Einträge der identischen Produkte unterscheiden, wurde ein kleines Skript namens “study\_products\_table.R” in der Programmiersprache R verwendet. Dieses ist auch Teil des Lieferumfangs unseres zip-Files. Zu seiner Ausführung ist die Installation des Pakets “dplyr” nötig. Das Skript tut prinzipiell folgendes:

1. Einlesen der csv-Datei “products.csv”.
2. Es erzeugt einen logischen Vektor mit der Information, wann ein Produktname zum mindestens zweiten Mal auftritt (man sieht hier, dass nur die letzten 3 stellen im Vektor “TRUE” sind)
3. Da ein Datensatz von allen doppelten Einträgen (inklusive des ersten Auftretens eines später wiederholten Eintrags) erwünscht ist, wird ein filternder “semi-join()” des ursprünglichen Datensatzes gegen den Datensatz mit den doppelt vorkommenden Produktnamen gemacht. Der Key ist der Produktname. Hierbei entsteht der Datensatz mit den doppelten Einträgen. Dieser hat sechs Zeilen.
4. Mit dem Befehl “View()” wird der Datensatz visuell dargestellt (siehe Bild 8).
5. Es zeigt sich, dass drei verschiedene Produkte jeweils doppelt vorkommen.
6. Um sicher zu gehen, dass auch wirklich alle Einträge bis auf “id” doppelt sind, wird für die drei Produkte mit dem Befehl “identical()” jeweils genau das überprüft – mit Erfolg. Das heisst also, dass diese drei Zeilen bis auf den Eintrag “id” identisch sind, und die letzten drei Zeilen des ursprünglichen Datensatzes (bzw. allgemeiner, die doppelten) entfernt werden müssen.

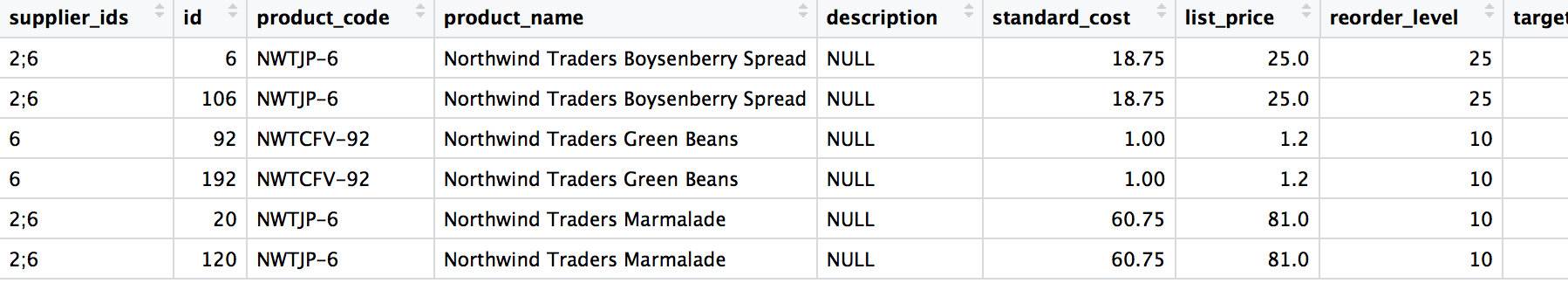


Bild : Ausschnitt (Reihen vollständig, Spalten unvollständig) aus Datensatz doppelter Einträge, extrahiert mittels "R" aus der Tabelle Produkte der Northwind-Datenbank.

Wie schon erwähnt, passiert das Reparieren der fehlerhaften Daten bei unserer Lösung ausserhalb der staging-area Datenbank. Die Daten werden von dem Spoon-Skript “staging\_area-v2.ktr” aus den csv-Files “products.csv” und “order\_details.csv” eingelesen und die reparierten – und zumindest im Falle der “order details”-Tabelle auch um weitere Variablen aus den anderen csv-Files ergänzten – Daten werden in die csv-Files “products\_corrected.csv” und “order\_details\_corrected.csv” geschrieben. Das Hinzufügen von Spalten zur Tabelle der Bestellungsdetails erfolgt im Hinblick auf eine Vereinfachung der Überführung der Daten in das Sternschema.

Die Transformationen zur Behebung der Datenprobleme in der ursprünglichen Northwind-Datenbank laufen folgendermassen ab (siehe auch Bild 9):

1. Produkt-Tabelle
   1. Einlesen der csv-Datei “products.csv”
   2. Sortierung der Zeilen der Tabelle nach Produktnamen mit einem “Sort rows”-Schritt. Dies ist notwendig, da der folgende Schritt nicht funktioniert, falls doppelte Reihen nicht nacheinander kommen.
   3. Entfernen der doppelten Einträge mit einem “Unique rows”-Schritt. Hierbei werden alle Variablen bis auf die Variable “id” verglichen.
   4. Sortierung der Zeilen der Tabelle nach “id”, so wie sie ursprünglich geordnet waren.
   5. Schreiben der Daten in das csv-File “products\_corrected.csv”
2. Bestelldetails-Tabelle
   1. Einlesen der csv-Datei “order\_details.csv”.
   2. Hinzufügen der Variablen “customer\_id”, “employee\_id” (diese Variable ist in der staging-area Datenbank nicht vorhanden. Sie wird aber auch für die Auswertungen nicht benötigt), “order\_date”, “payment\_type” aus der Tabelle “orders” mittels eines Schrittes vom Typ “Stream Value Lookup”.
   3. Hinzufügen der Variable “standard\_cost” aus der Tabelle “products”
   4. Hinzufügen der Variablen “city”, “state\_province” und “country\_region” aus der Tabelle “customer”
   5. Kreieren einer neuen Spalte “discount\_corr”, die den Betragswert der Variable “discount” enthält. Dies geschieht durch einen Schritt vom Typ “Calculator”.
   6. Mit einem Schritt “Select / Rename values” werden 15 Variablen ausgewählt und gleichzeitig die Variable “discount” durch “discount\_corr” ersetzt.
   7. Im letzten Schritt werden die Daten in das csv-File “order\_details\_corrected.csv” geschrieben.

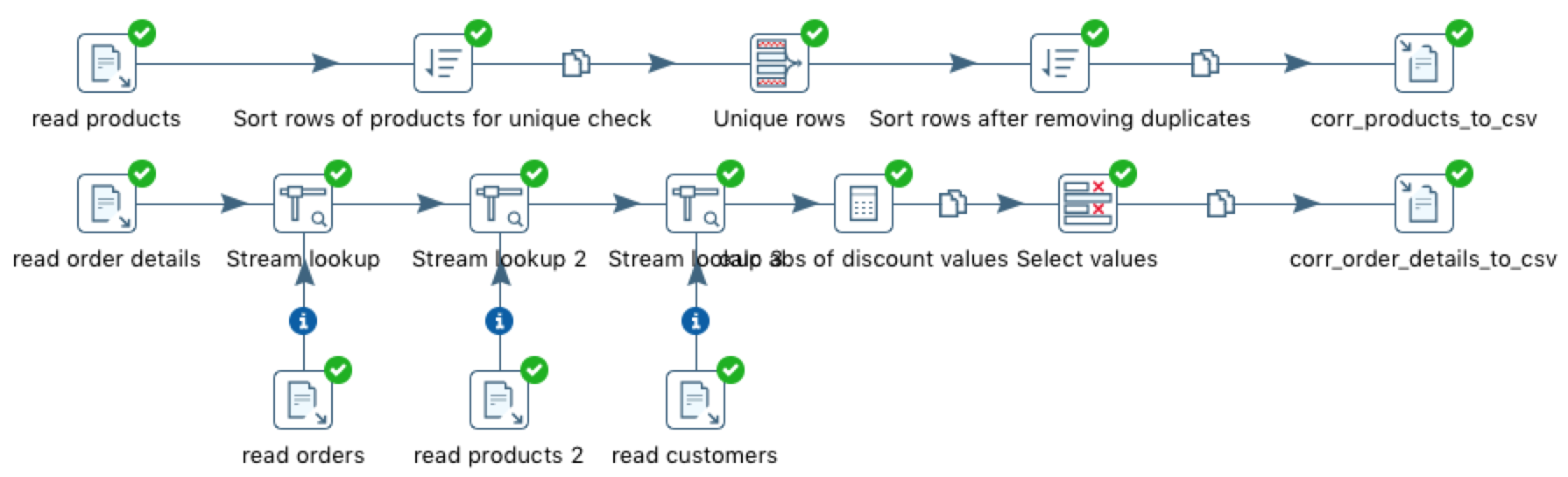


Bild 9: "staging\_area-v2.ktr" nach erfolgreichem Durchlauf. Hier werden fehlerhafte Daten korrigiert und im Fall der Bestelldetails auch weitere Spalten hinzugefügt. Am Ende werden die Daten in die zwei im Text erwähnten csv-Files geschrieben.

Wie in Abschnitt 3 erwähnt, sollte das Einlesen von sowohl der “products”- als auch der “order\_details”-Tabellen eigentlich aus der staging-area Datenbank durch einen “Table input”-Schritt geschehen. Das Schreiben der beiden Tabellen sollte eigentlich ebenso in die staging-area Datenbank geschehen, entweder mit einem “Dimension lookup / update”-Schritt oder mit einem “Table output”-Schritt.

# **Konzipierung des Sternschemas für das DWH**

Für das multidimensionale Modell haben wir ein einfaches Stern-Schema gewählt. Die Faktentabelle im Zentrum basiert auf der Relation “order\_details” aus dem Ausgangsschema. In “order\_details” finden sich die Einzelpositionen aller Bestellungen, über die sich geeignete Auswertungen durchführen lassen.

So sollen später zum Beispiel Umsätze pro Monat oder Gewinnzahlen für bestimmte Produkte ermittelt werden.

Bei den Dimensionen beschränken wir uns auf die Bereiche “Kunde”, “Produkt”, “Region”, “Bestellung” und “Zeit”. Zusätzlich wären Auswertungen über die Spediteure, die Lieferanten und die zuständigen Sachbearbeiter denkbar, was entsprechende weitere Dimensionen erfordern würde.

Bei der Dimension “Region” wäre eine Erweiterung auf das Schneeflocken-Schema denkbar gewesen, da es sich bei Ländern, Bundesstaaten und Städten um unterschiedliche Granularitäten räumlicher Angaben handelt. Allerdings erlaubt die geringe Datenmenge ohnehin keine fein granularen Auswertungen, so dass wir auf diese Unterteilung verzichtet haben.

Bild 10 zeigt das vollständige Schema.



Bild : ERM-Diagramm unseres Sternschema-Modells.

# **Erstellung und Befüllen des DWH**

Das Ziel-Schema wurde direkt durch SQL-Befehle in MySQL implementiert. Die notwendigen Erweiterungen, insbesondere für “Slowly Changing Dimensions” werden durch Pentaho vorgenommen. Hier die SQL-Befehle zum Erstellen des Data Warehouse:

DROP DATABASE IF EXISTS NORTHWIND\_DWH;

CREATE DATABASE NORTHWIND\_DWH;

USE NORTHWIND\_DWH;

CREATE TABLE kunde\_dim (

KUNDE\_ID\_DIM INT(11) NOT NULL AUTO\_INCREMENT,

CUSTOMER\_ID INT(11) NOT NULL,

COMPANY VARCHAR(50) NULL DEFAULT NULL,

FIRST\_NAME VARCHAR(50) NULL DEFAULT NULL,

LAST\_NAME VARCHAR(50) NULL DEFAULT NULL,

CONSTRAINT PK\_KUNDE\_DIM PRIMARY KEY (KUNDE\_ID\_DIM)

);

CREATE TABLE bestellung\_dim (

BESTELLUNG\_ID\_DIM INT(11) NOT NULL AUTO\_INCREMENT,

BESTELLUNG\_ID INT(11) NOT NULL,

PAYMENT\_TYPE VARCHAR(11) NULL DEFAULT NULL,

ORDER\_DATE DATE,

CONSTRAINT PK\_BESTELLUNG\_DIM PRIMARY KEY   
 (BESTELLUNG\_ID\_DIM)

);

CREATE TABLE produkt\_dim (

PRODUKT\_ID\_DIM INT(11) NOT NULL AUTO\_INCREMENT,

PRODUKT\_ID INT(11) NOT NULL,

PRODUKT\_CODE VARCHAR(25) NULL DEFAULT NULL,

PRODUKT\_NAME VARCHAR(50) NULL DEFAULT NULL,

STANDARD\_COST DECIMAL(19 , 4 ) NULL DEFAULT '0.0000',

CONSTRAINT PK\_PRODUKT\_DIM PRIMARY KEY (PRODUKT\_ID\_DIM)

);

CREATE TABLE zeit\_dim (

ZEIT\_ID\_DIM INT(11) NOT NULL AUTO\_INCREMENT,

ORDER\_DATE DATE,

ORDER\_DAY INTEGER,

ORDER\_MONTH INTEGER,

ORDER\_YEAR INTEGER,

CONSTRAINT PK\_ZEIT\_DIM PRIMARY KEY (ZEIT\_ID\_DIM)

);

CREATE TABLE region\_dim (

REGION\_ID\_DIM INT(11) NOT NULL AUTO\_INCREMENT,

CITY VARCHAR(25) NULL DEFAULT NULL,

STATE\_PROVINCE VARCHAR(2) NULL DEFAULT NULL,

COUNTRY\_REGION VARCHAR(15) NULL DEFAULT NULL,

CONSTRAINT PK\_REGION\_DIM PRIMARY KEY (REGION\_ID\_DIM)

);

CREATE TABLE verkauf\_fact (

KUNDE\_ID\_DIM INT(11) NOT NULL,

PRODUKT\_ID\_DIM INT(11) NOT NULL,

ZEIT\_ID\_DIM INT(11) NOT NULL,

REGION\_ID\_DIM INT(11) NOT NULL,

BESTELLUNG\_ID\_DIM INT(11) NOT NULL,

BRUTTO\_ERLOES INTEGER,

NETTO\_ERLOES INTEGER,

KOSTEN INTEGER,

GEWINN INTEGER

);

Das Befüllen des Schemas mit den Daten erfolgt über zwei Pentaho Transformationen:

1. “star-schema.ktr”: diese Transformation befüllt die Dimensionstabellen
2. “”facts-table.ktr”: durch diese Transformation wird die Fakten-Tabelle befüllt

**Befüllen der Dimensionstabellen**

Wir beginnen mit dem Befüllen der Produkte-Dimension. Diese besteht aus zweit Schritten (siehe Bild Bild 11):



Bild : Schritte zum Befüllen der Tabelle "produkt\_dim" der Datenbank "NORTHWIND\_DWH" aus dem Spoon-File "star-schema.ktr".

Der Schritt “Produkte korrigiert” liest die Informationen aus der Staging Area ein, d.h. das csv-File, das wir in den Schritten 2 bis 4 erstellt haben. Im konkreten Fall das File “products-corrected.csv”. Dazu wird in Pentaho das Element “Input -> CSV file input” gewählt und mit einem Doppelklick geöffnet. Der Schritt wird umbenannt, der Pfad auf das CSV-File selektiert und als Delimiter das Komma gesetzt (siehe Bild 12).

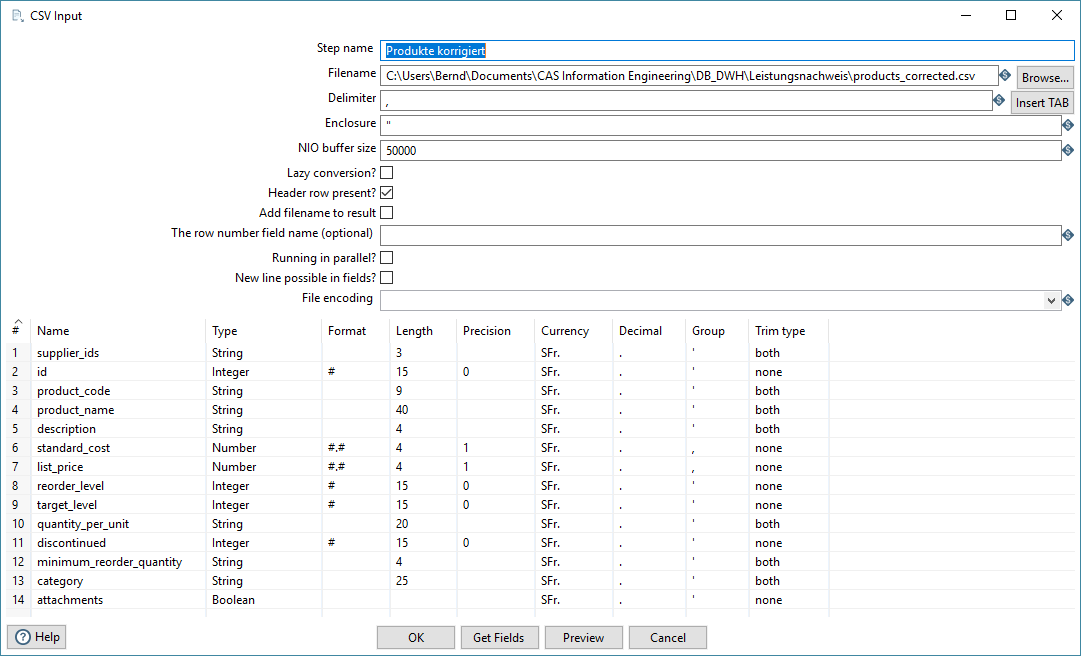


Bild : Spoon-Schritt "CSV Input" für Tabelle "produkt\_dim".

Nachdem mit “Get Fields” die Attribute aus dem CSV File identifiziert wurden, werden die Datentypen überprüft. Wo nötig werden Anpassungen vorgenommen. Vor allem bei String Feldern könnten in Zukunft längere Werte vorkommen, so dass sich unter Umständen eine Erhöhung der Feldlänge lohnt.

Ein Blick auf das CSV File zeigt zudem, dass sich durch den Korrekturschritt im CSV File Leerzeichen eingeschlichen haben (siehe Bild 13).

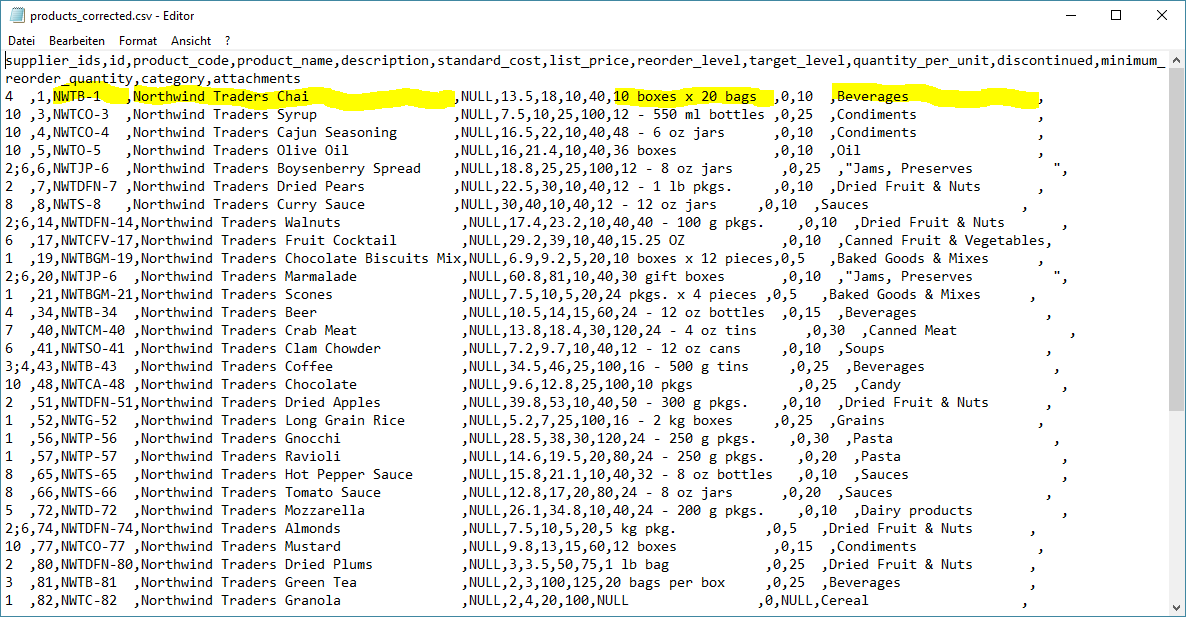
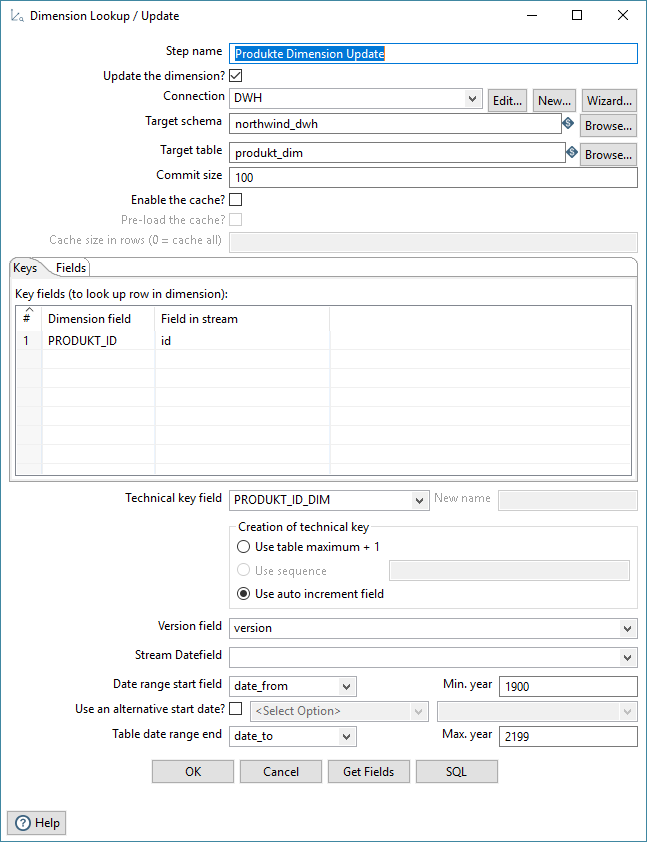


Bild : Durch den Korrekturschritt sind im csv-File Produkte-Tabelle Leerzeichen in einigen Spalten erzeugt worden.

Daher wird im Transformationsschritt beim Datentyp String unter “Trim type” die Option “both” gewählt, um die Leerzeichen zu entfernen. Die Leerzeichen scheinen zwar nur am Ende eingefügt zu sein, aber um auf sicher zu gehen werden Leerzeichen am Anfang und am Ende des Strings entfernt.

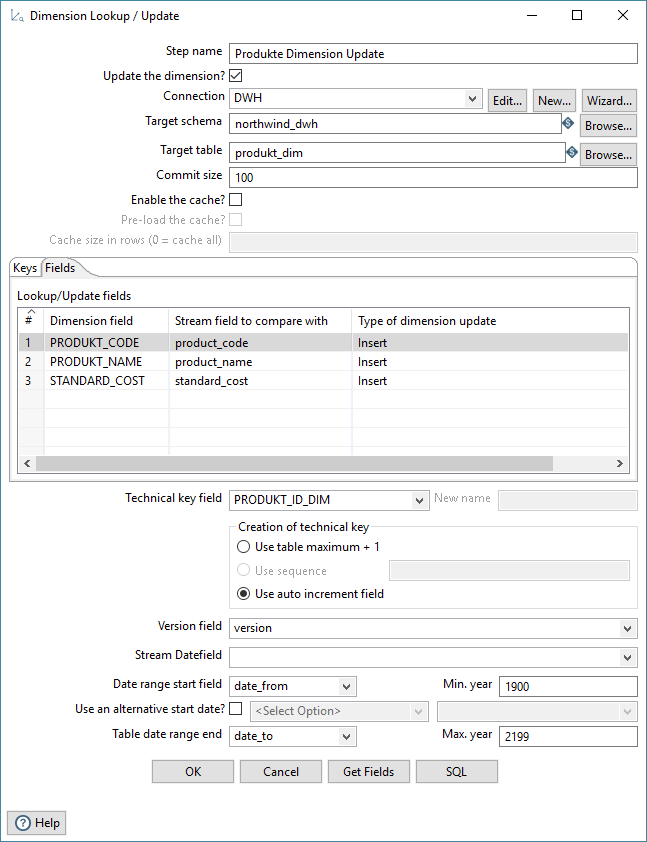
Mit “Preview” werden die Daten nochmal überprüft und dann mit “OK” der Schritt geschlossen.

Im zweiten Transformationsschritt werden aus dem Eingabe-Stream die für die Dimensionstabelle relevanten Attribute extrahiert und dann in die Datenbank geschrieben. Dafür wählen wir das Transformationselement “Data Warehouse -> Dimension lookup/update” und verbinden es mit dem vorherigen Schritt. Ein Doppelklick öffnet folgendes Fenster:



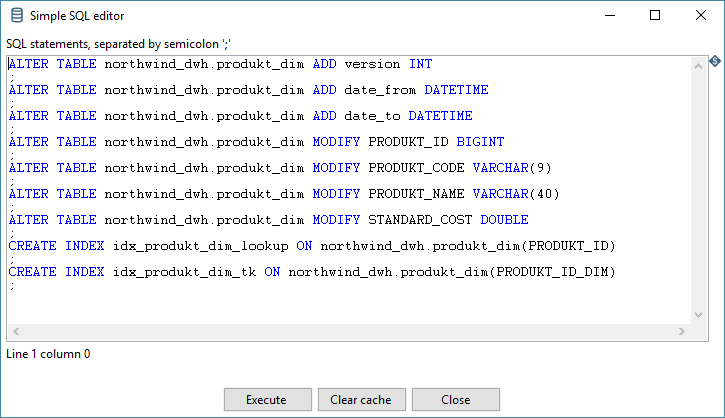
Wir lassen Pentaho den Surrogat-Schlüssel “PRODUKT\_ID\_DIM” befüllen und verwenden den Auto Increment der Datenbank.

Um die für uns interessanten Felder der Dimension zu wählen und “Slowly Changing Dimensions” vom Typ 2 zu implementieren, wechseln wir ins Tab “Fields”:

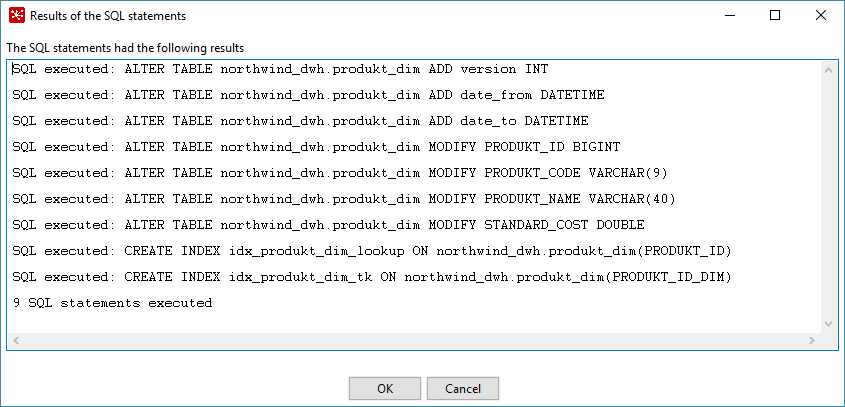


Wie gut zu erkennen ist, beschränken wir uns auf drei Felder aus dem Eingabe-Stream (product\_code, product\_name und standard\_cost) und mappen sie auf die Felder unseres Datenbankschemas. Für die Typ 2 SCD wird als “Type of dimension update” der Wert “Insert” gewählt.

Bevor die Transformation ausgeführt werden kann, muss dass generierte SQL ausgeführt werden. Sonst fehlen die notwendigen Schemaänderungen in der Datenbank. Dazu wird auf den Button “SQL” geklickt:



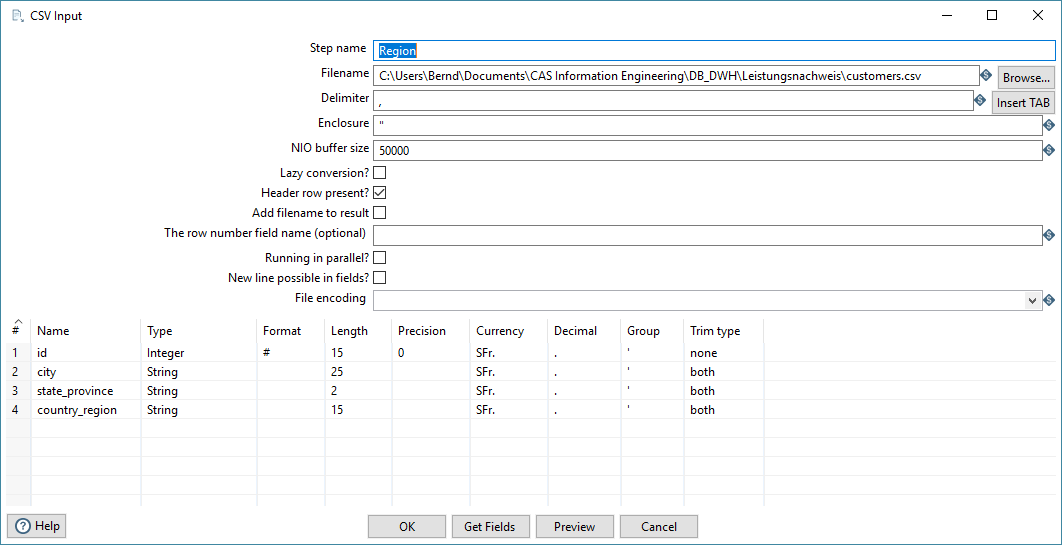
Ein Klick auf “Execute” führt die notwendigen Änderungen in der Datenbank durch. Pentaho bestätigt uns die erfolgreiche Ausführung:



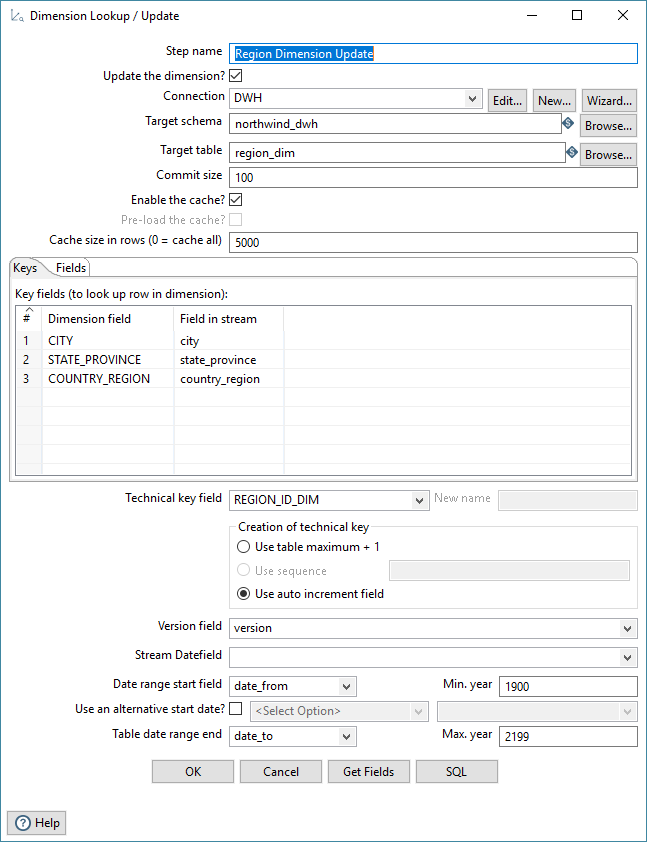
Danach können die Fenster durch “OK” bzw. “Close” wieder geschlossen werden.

In analoger Weise wird für die Dimensionen “Kunde” und “Bestellung” vorgegangen.

Bei “Region” ist zu beachten, dass dafür in der Ausgangs-DB keine eigene Relation vorliegt. Wir beziehen diese Daten aus der Relation “customers” über die Felder “city”, “state\_province” und “country\_region”.

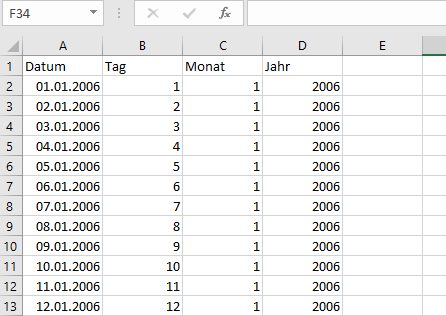


Wir dürfen für die Identifikation der Zeilen nicht den Schlüssel “id” dieser Relation heranziehen, da wir sonst doppelte Einträge in der Dimensionstabelle erhalten würden. Stattdessen wählen wir die Kombination der drei Attribute als Key für die Identifikation von Datensätzen in der customers-Relation.

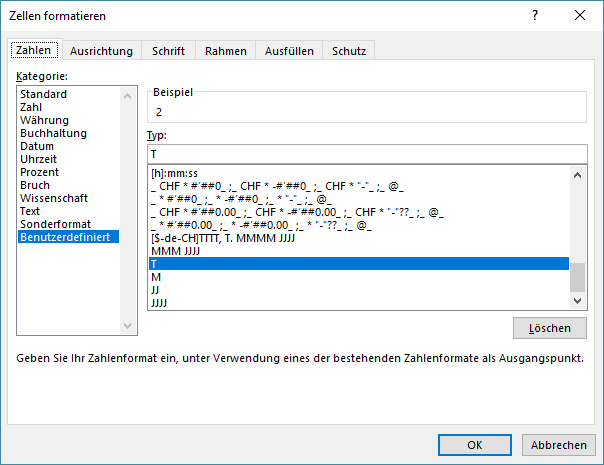


Zusätzliche Felder sind nicht zu wählen, da wir in der Dimensionstabelle nur an diesen Werten interessiert sind.

Für die Zeit-Dimension schauen wir zuerst in den Ausgangsdaten, welcher Zeitraum relevant ist. Offensichtlich handelt es sich um das Jahr 2006. Die Dimensionstabelle soll also mit den Tagen dieses Jahres gefüllt werden. Der einfachste Ansatz wäre gewesen, eine solche Tabelle mit ein paar Mausklicks in Excel zu erzeugen, als CSV zu speichern und dieses dann in gewohnter Weise einzulesen und in das DWH zu schreiben.



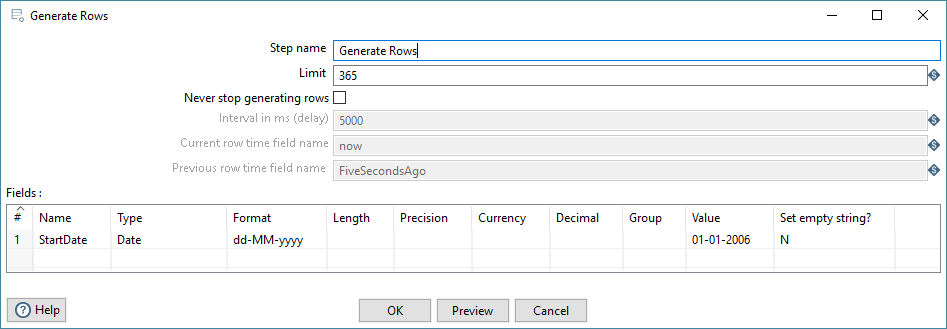
Die Spalte B lässt sich leicht duch eine benutzerdefinierte Formatierung füllen. Gleiches gilt dann für die Spalten C und D.



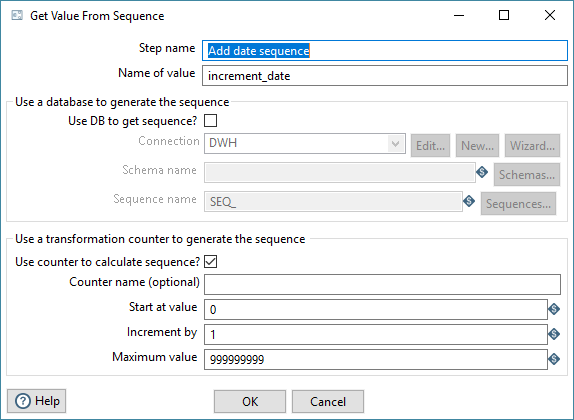
Statt des Ansatzes mit Excel haben wir einen Ansatz mit Pentaho Transformationsschritten gewählt:



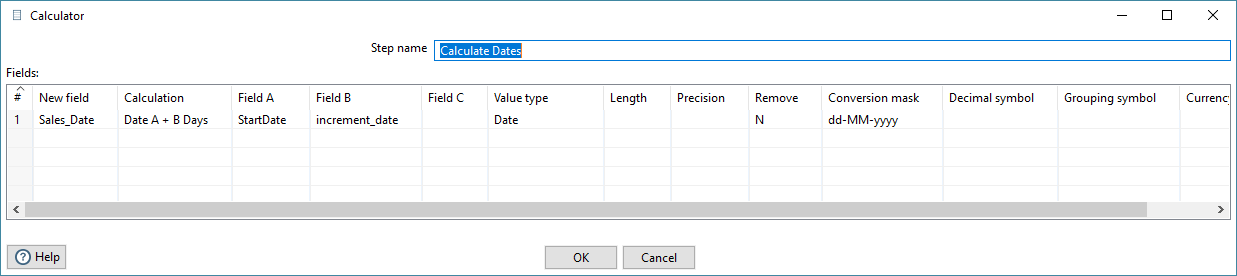
Im ersten Schritt werden mit dem Element “Generate Rows” 365 Zeilen erzeugt, für jeden Tag des Jahres eine, beginnend am 1 Januar 2006:



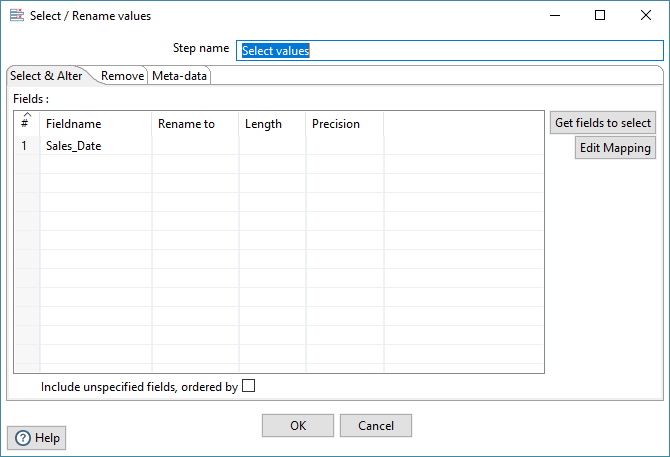
Nach diesem Schritt steht zunächst in jeder Zeile der 1. Januar 2006. Im zweiten Schritt wird ein Zähler generiert (“Add Sequence”), der für jedes Element, das er aus dem Eingangsstream erhält jeweils um 1 hochzählt. Wir erhalten also eine Sequenz von 0 bis 364



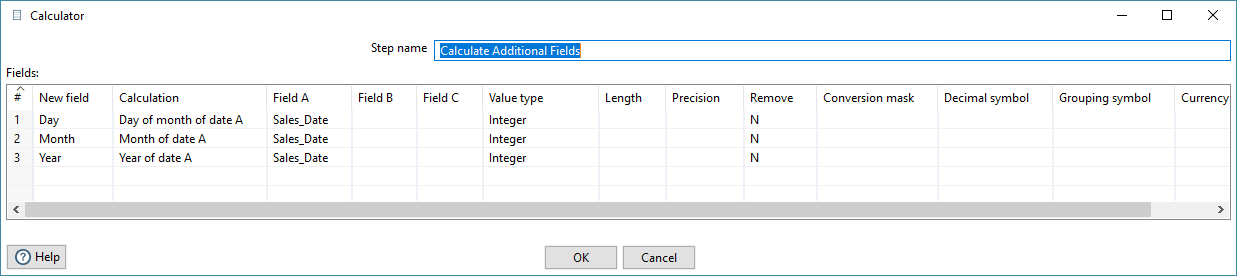
Im dritten Schritt wird dieser Zähler auf die Datumssequenz angewendet (“Calculator” Schritt). Wichtig ist es, bei “Conversion Mask” das gleiche Format wie im ersten Schritt zu definieren.



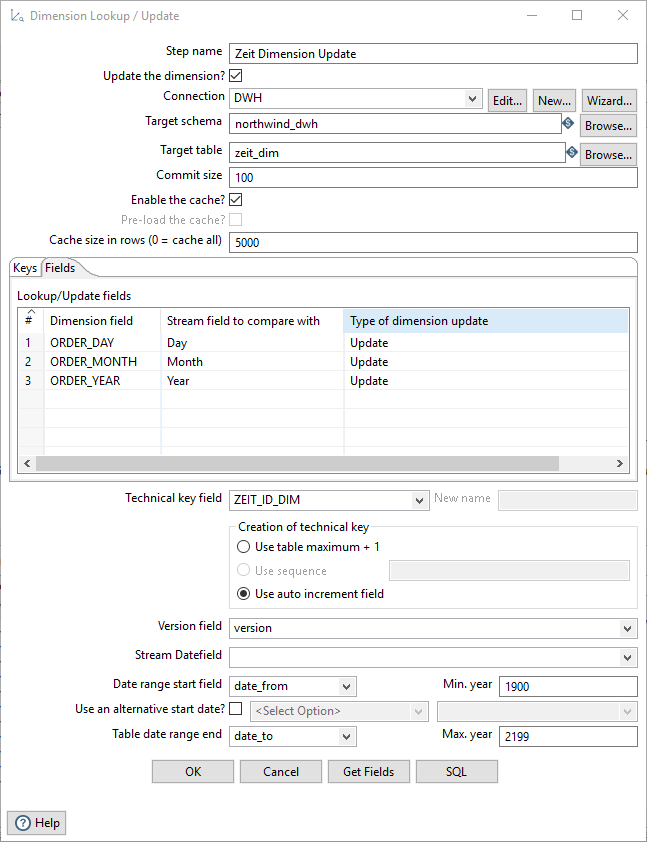
Nun wird der Datenstrom wird auf das Feld reduziert, um das es eigentlich geht: das Datumsfeld. Das geschieht mit Hilfe des “Select Values” Elements.

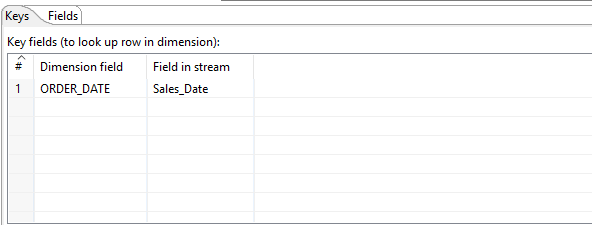


Für die Sequenz an Datumsfeldern (ein Element pro Tag im Jahr) können nun die anderen Felder der Zeit-Dimensionstabelle berechnet werden mithilfe eines “Calculator” Schritts:

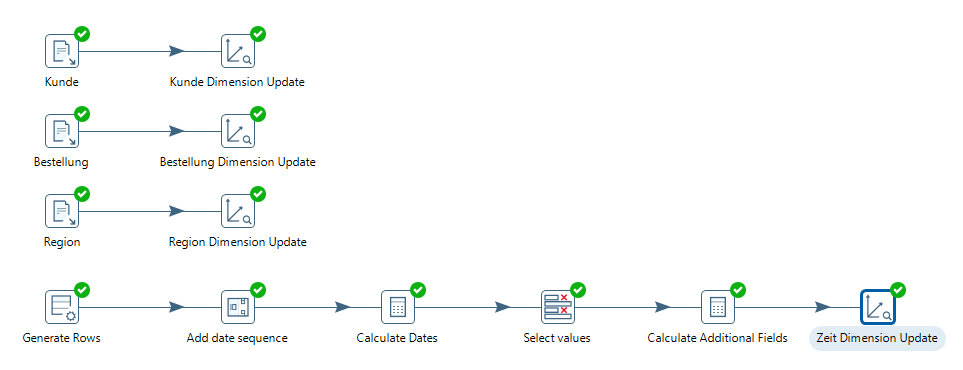


Als abschliessender Schritt wird die Dimensionstabelle wie zuvor gefüllt:

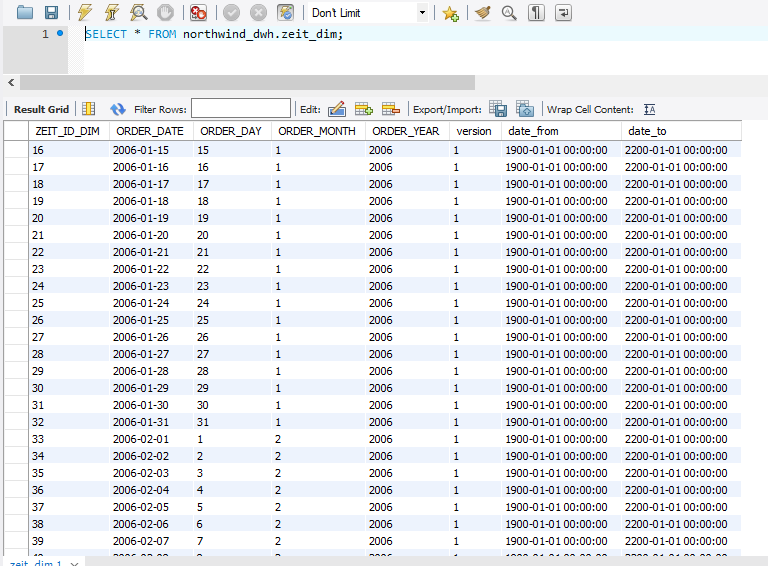




Nun kann die komplette Transformation ausgeführt werden:

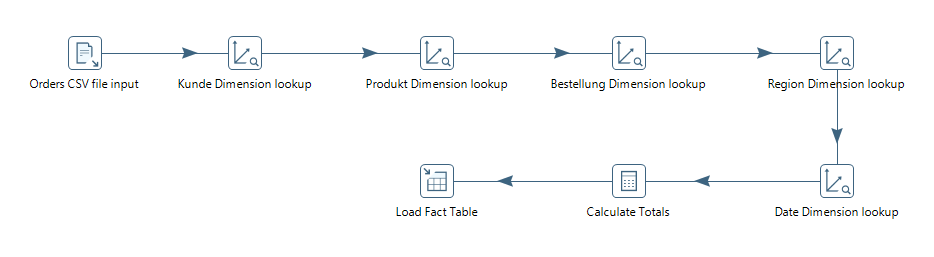


Ein exemplarischer Blick in die Datenbank auf die Dimensionstabellen am Beispiel der Zeit-Dimension:

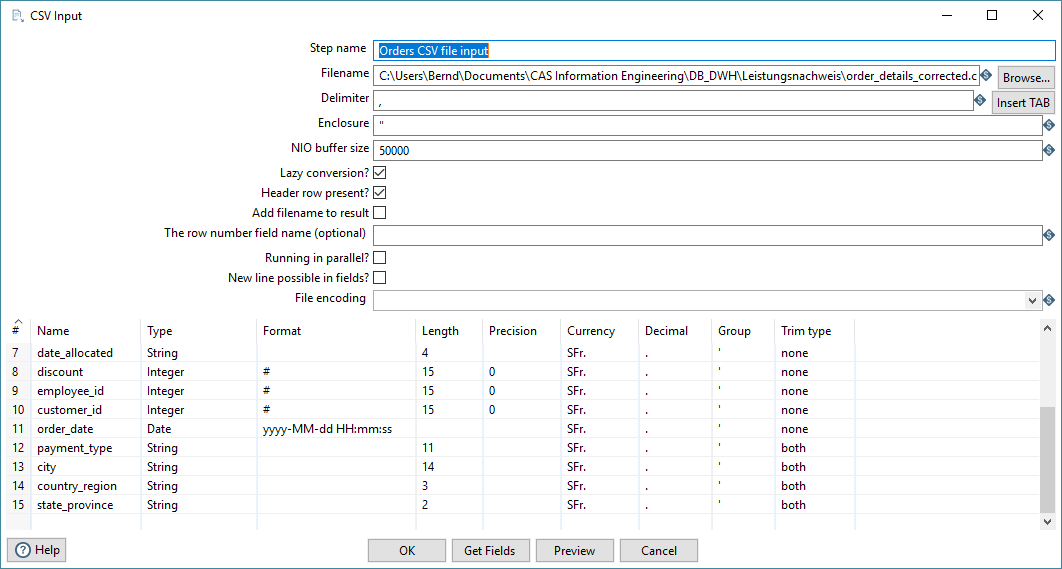


**Erzeugen der Fakten**

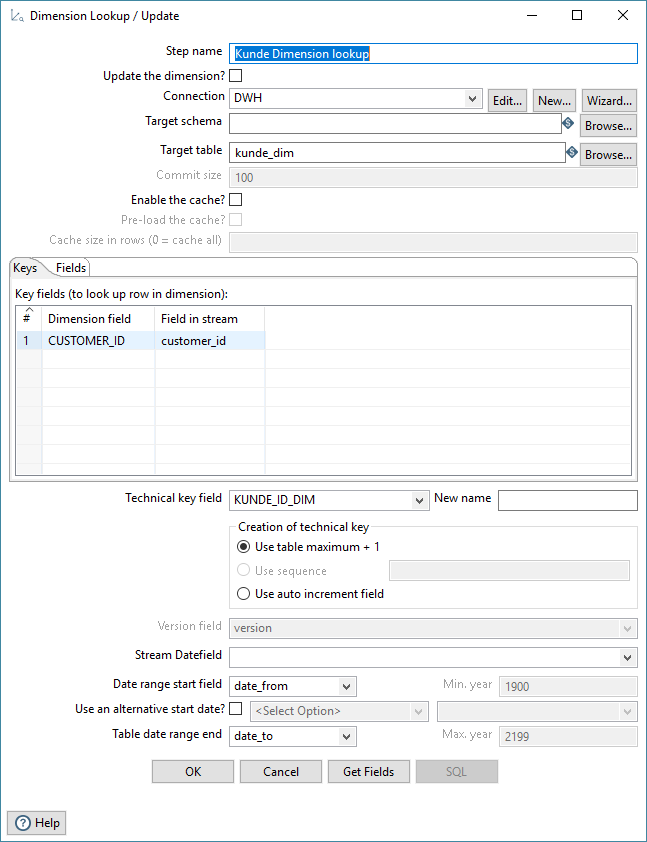
Wie eingangs erwähnt basiert die Faktentabelle auf “order\_details” aus der Ausgangs-DB. Diese Relation wird im ersten Schritt eingelesen.



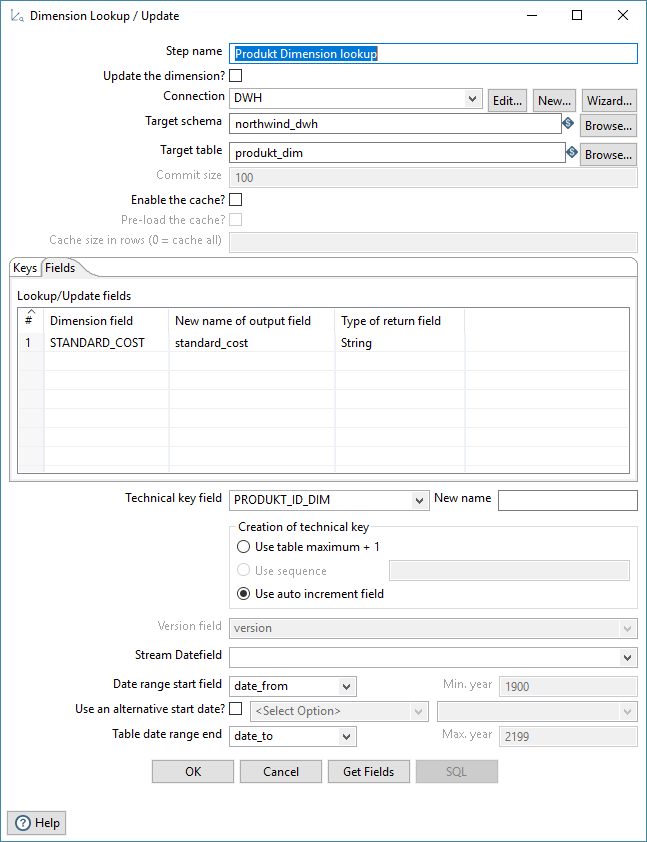
Auch hier schauen wir wieder, dass allfällige Leerzeichen am Anfang und Ende von String entfernt werden, damit das Matching sauber funktioniert.



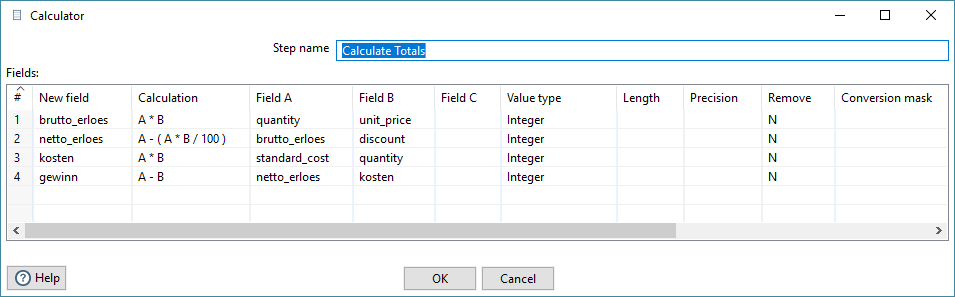
Danach werden im “Dimension lookup/update” Schritten für jeden Datensatz die Surrogatschlüssel aus den Dimensionstabellen gelesen.



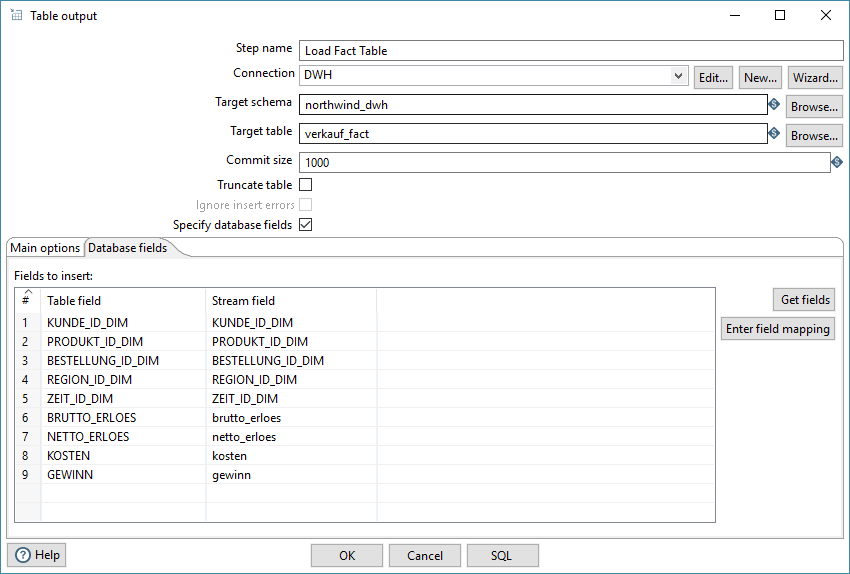
Für die abschliessende Kalkulation der für die Faktentabelle definierten Werte (Netto\_Erloes, Brutto\_Erloes, Gewinn und Kosten) wird aus den Produkten noch der Wert für die Standard-Kosten in den Datenstream eingefügt.



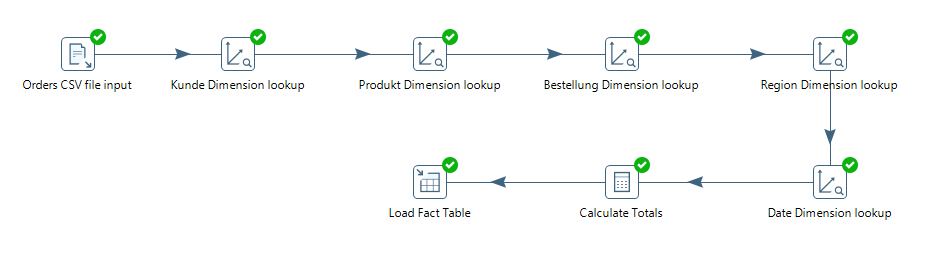
Diese Information wird dann im “Calculator” Step (vorletzter Schritt der Transformation) in den Berechnungen genutzt:



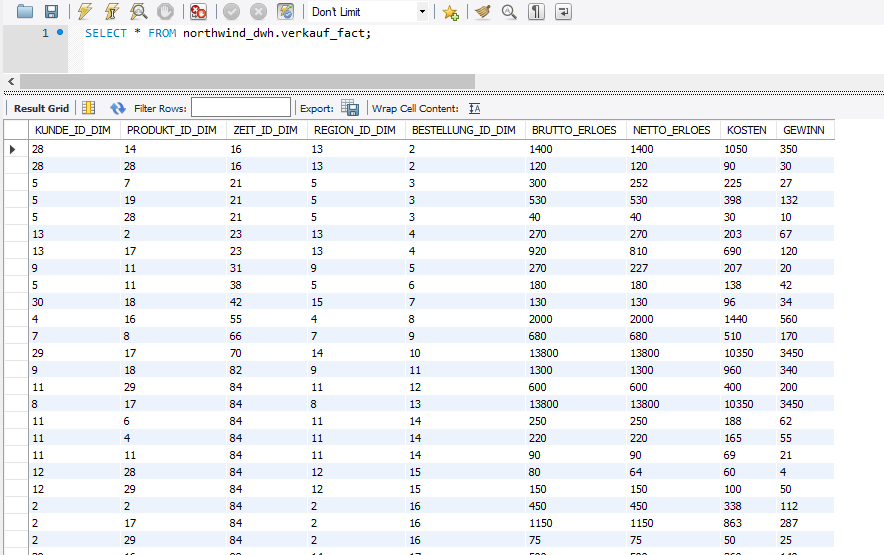
Im letzten Schritte werden dann alle Datensätze der Faktentabelle einschl. kalkulierter Attribute in das Data-Warehouse geschrieben:



Nach dem alle Schritte definiert sind, lassen wir die Transformation laufen.

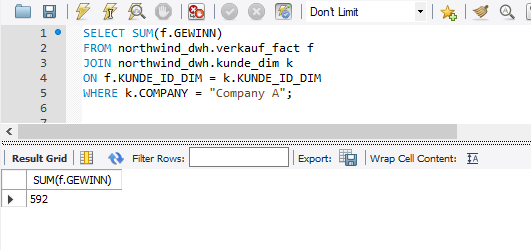


Ein Kontrollblick in die Datenbank bestätigt, dass die Faktentabelle geschrieben wurde:

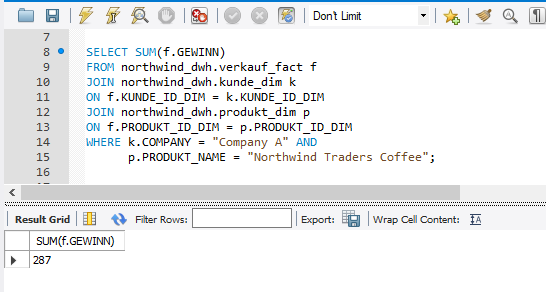


# **Auswertungen auf Basis des DWH**

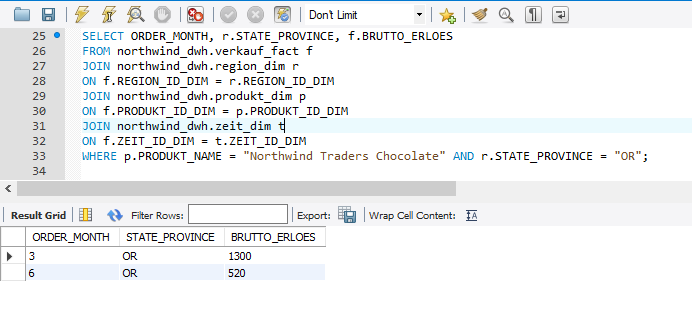
Wir beginnen mit einer sehr einfachen Auswertung, bei der wir den Gesamtgewinn berechnen, den wir mit unserem Kunden “Company A” erzielt haben:



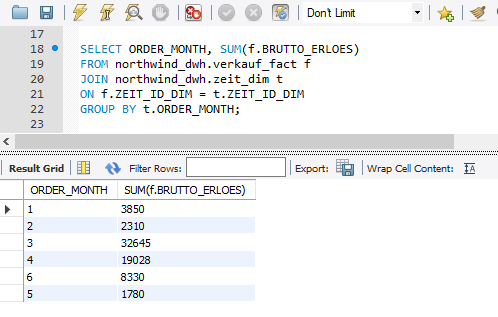
Als nächstes wollen wir uns anschauen, wieviel Gewinn wir mit diesem Kunden mit unserem Produkt “Northwind Traders Coffee” erzielt haben:



Beim Produkt “Northwind Traders Chocolate” interessiert uns nun vor allem, wie sich der Verkauf bei unseren Kunden in Oregon im Lauf des Jahres entwickelt hat.



Und last but not least wollen wir uns einen Überblick über die monatlichen Brutto-Erlöse verschaffen, d.h. die Erlöse vor Abzug aller Rabatte:



# **Fazit und Schlusswort**

Die Arbeit an diesem durchaus umfangreichen Objekt war sicher lehrreich auf verschiedenen Ebenen, zu denen die folgenden zählen:

* Besseres Kennenlernen der für uns neuen Tools “PDI” und “MySQL-Workbench”
* Bewusst werden, wie ERM-Schemas aufgebaut sind und Umgang mit unterschiedlichen ERM-Dialekten
* Umgang mit SQL-Syntax
* Effektive Zusammenarbeit in einer Gruppe (da lernt man nie aus)
* Arbeiten mit Versionskontrolle durch “git”.

Die Arbeit mit “git” hat gut funktioniert und den Austausch von Ideen und Dateien sicher erleichtert und es ermöglicht, den Überblick über die Änderungen im Filesystem zu behalten. Sicher wäre die Arbeit an diesem doch überschaubaren Projekt auch ohne git möglich gewesen, aber es hat – gefühlt – selbst auf diesem Niveau schon Vorteile.

Eine Sache, deren wir uns vorher nicht bewusst waren, ist, dass aufgrund der Definition von Fremdschlüssel-Constraints die Reihenfolge der Befüllung (und auch der Löschung) von Tabellen eine Rolle spielt.

Dadurch, dass das Projekt wirklich viele Aspekte des Moduls “DB & DWH” berührt hat, war es für uns sehr nützlich bei der Nachbereitung des Lehrmaterials.