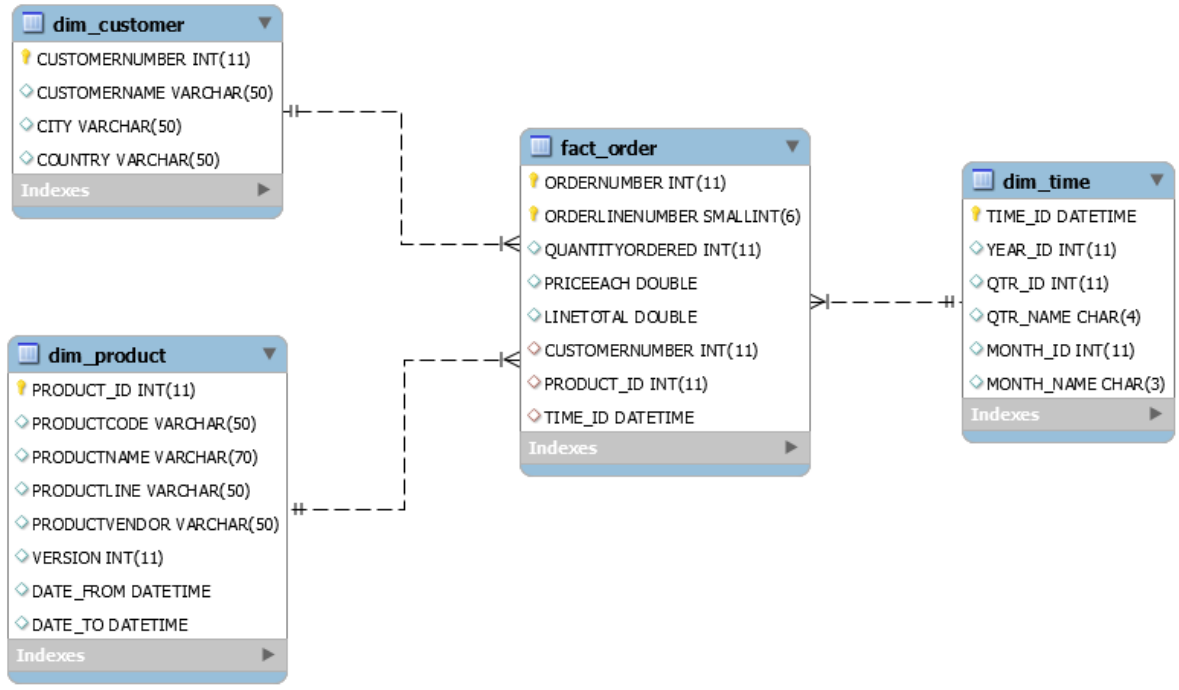
In this lab, we are going to create a data warehouse from the steelwheels database. The data warehouse will have a star schema with one fact table and three dimension tables, as shown in the following figure:



The data warehouse will be created as a separate database (called **steelwheels_dw**) in MySQL. For convenience, the SQL instructions needed to create the data warehouse tables are already provided in the script **steelwheels_dw.sql**.

Creating the data warehouse

1. Download the file **steelwheels_dw.sql**.

2. Take a moment to inspect the contents of the **steelwheels_dw.sql** script.
   - Locate the CREATE DATABASE statement.
   - Locate all CREATE TABLE statements.
   - Check the columns and data types for each table.
   - Check the primary and foreign keys for each table.

3. Open a terminal and navigate to the folder where the **steelwheels_dw.sql** script is located.

4. Execute the following command to login to the local MySQL server: **mysql -u aid -p**
   Password: **aid**

5. On the MySQL prompt, execute the following command to create the database:
   source steelwheels_dw.sql

6. Execute the following command to show the existing databases:
   show databases;

7. Check that you have both the **steelwheels** database and the **steelwheels_dw** data warehouse.

8. Leave this terminal open. It will be useful in the next steps.

Creating a connection to the data warehouse

9. Open a new terminal and navigate to the folder: **~/Pentaho/data-integration**

10. Start Pentaho Data Integration (PDI) with: **./spoon.sh**

11. In the **File** menu, select **New > Transformation**.

12. In the left pane, switch from the **Design** to the **View** tab, and expand **Transformations > Transformation 1 > Database connections**.

13. Right-click **Database connections** and select **New**.

14. In the **Database Connection** dialog, specify the following:
    - Connection Name: **steelwheels_dw**
    - Connection Type: **MySQL**
    - Access: **Native (JDBC)**
    - Host Name: **localhost**
    - Database Name: **steelwheels_dw**
    - Port Number: **3306**
    - User Name: **aid**
    - Password: **aid**

15. Press **Test** to test the database connection. A new dialog should say that the connection is OK.

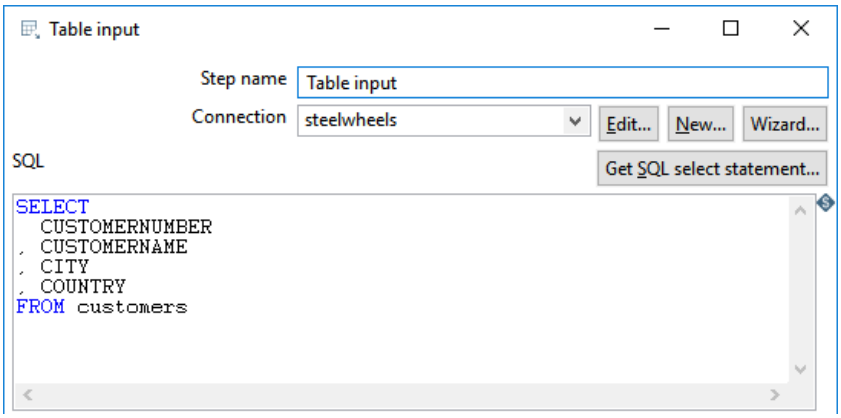
16. Close the **Database Connection** dialog with **OK**.

17. In the **View** tab, right-click the **steelwheels_dw** database connection and select **Share**. This will make the database connection available to other transformations.

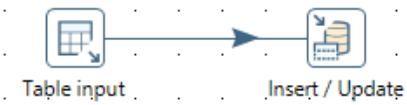
*Note: In the previous lab, you have already created a connection to the **steelwheels** database. The connection to **steelwheels_dw** is very similar; the only changes are in the **Connection Name** and in the **Database Name**. In this lab, we will use both connections: **steelwheels** and **steelwheels_dw**.*

Creating the customer dimension

1. Add a **Table Input** to the transformation and configure it as follows:
    - In **Connection** select **steelwheels**
    - Press the **Get SQL select statement** button and select the **customers** table
    - **Do you want to include the field-names in the SQL?** Answer: **Yes**
    - Then remove every field except **CUSTOMERNUMBER**, **CUSTOMERNAME**, **CITY**, **COUNTRY**. (*See the following figure.*)



2. Add an **Insert/Update** step to the transformation, and create a hop from the previous step.



3. Configure the **Insert/Update** step as follows:
   - In **Connection** select **steelwheels_dw**
   - In **Target table** click **Browse** and select the **dim_customer** table
   - The key that will be used to check if a customer already exists in the dimension table is **CUSTOMERNUMBER**. Therefore, configure **The key(s) to look up the value(s)** as follows:

The key(s) to look up the value(s):

#	Table field	Comparator	Stream field1	Stream field2
1	CUSTOMERNUMBER	=	CUSTOMERNUMBER	

   - The dimension table stores the following fields for each customer: **CUSTOMERNUMBER**, **CUSTOMERNAME**, **CITY**, **COUNTRY**. These fields will have to be inserted or updated in the table. Therefore, configure **Update fields** as follows:

Update fields:

#	Table field	Stream field	Update
1	CUSTOMERNUMBER	CUSTOMERNUMBER	Y
2	CUSTOMERNAME	CUSTOMERNAME	Y
3	CITY	CITY	Y
4	COUNTRY	COUNTRY	Y

4. Save the transformation as **/home/aid/Downloads/dim_customer.ktr** (if you are on the VM).

5. Run the transformation.

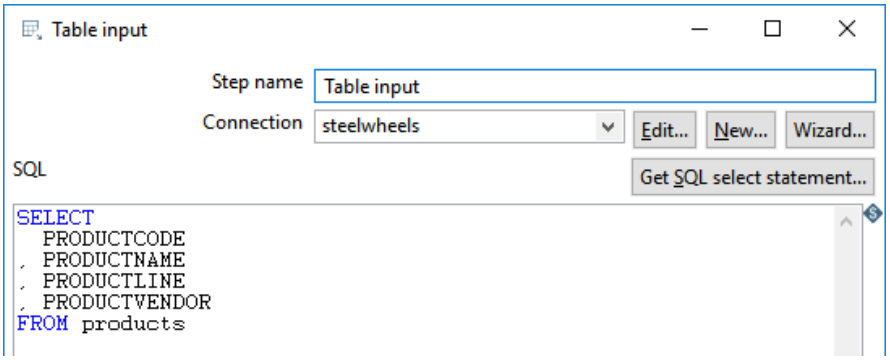
6. Using the command line, check that the data has been loaded into the **dim_customer** table in the **steelwheels_dw** data warehouse.
   (*See the following figure.*)

```
Command Prompt - mysql  -u aid -p
mysql> select * from dim_customer;
+----------------+----------------------------------+------------------+-------------+
| CUSTOMERNUMBER | CUSTOMERNAME                     | CITY             | COUNTRY     |
+----------------+----------------------------------+------------------+-------------+
|             97 | Madison Inc                      | ST  AUGUSTINE    | USA         |
|             98 | Johnson Inc                      | ST Cloud         | USA         |
|             99 | Tarallo Inc                      | Sanford          | USA         |
|            100 | Audio Video 'R' Us               | Orlando          | USA         |
|            103 | Atelier graphique                | Nantes           | France      |
|            112 | Signal Gift Stores               | Las Vegas        | USA         |
|            114 | Australian Collectors, Co.       | Melbourne        | Australia   |
|            119 | La Rochelle Gifts                | Nantes           | France      |
|            121 | Baane Mini Imports               | Stavern          | Norway      |
|            124 | Mini Gifts Distributors Ltd.     | San Rafael       | USA         |
|            125 | Havel & Zbyszek Co               | Warszawa         | Poland      |
|            128 | Blauer See Auto, Co.             | Frankfurt        | Germany     |
|            129 | Mini Wheels Co.                  | San Francisco    | USA         |
|            131 | Land of Toys Inc.                | NYC              | USA         |
|            141 | Euro+ Shopping Channel           | Madrid           | Spain       |
|            144 | Volvo Model Replicas, Co         | Luleå            | Sweden      |
|            145 | Danish Wholesale Imports         | Kobenhavn        | Denmark     |
|            146 | Saveley & Henriot, Co.           | Lyon             | France      |
|            148 | Dragon Souveniers, Ltd.          | Singapore        | Singapore   |
|            151 | Muscle Machine Inc               | NYC              | USA         |
|            157 | Diecast Classics Inc.            | Allentown        | USA         |
|            161 | Technics Stores Inc.             | Burlingame       | USA         |
|            166 | Handji Gifts& Co                 | Singapore        | Singapore   |
|            167 | Herkku Gifts                     | Bergen           | Norway      |
|            168 | American Souvenirs Inc           | New Haven        | USA         |
|            169 | Porto Imports Co.                | Lisboa           | Portugal    |
```

Creating the product dimension

7. Create a new transformation in PDI (Spoon).

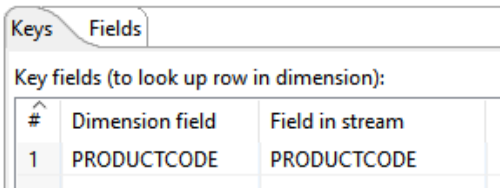
8. Add a **Table Input** to the transformation and configure it as follows:
   - In **Connection** select **steelwheels**
   - Press the **Get SQL select statement** button and select the **products** table
   - **Do you want to include the field-names in the SQL?** Answer **Yes**
   - Then remove every field except **PRODUCTCODE**, **PRODUCTNAME**, **PRODUCTLINE**, **PRODUCTVENDOR**.



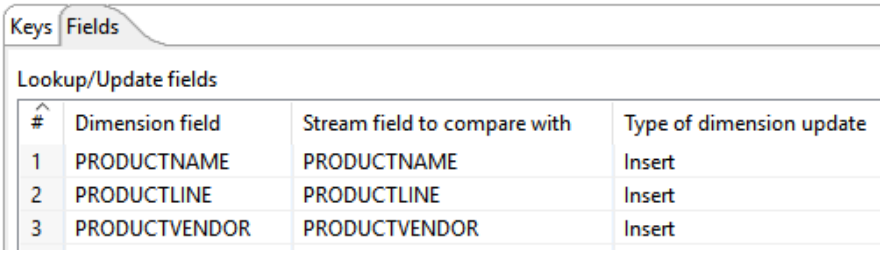
9. Add a **Dimension lookup/update**, and create a hop from the previous step.

10. Configure the **Dimension lookup/update** as follows:
    - In **Connection** select **steelwheels_dw**

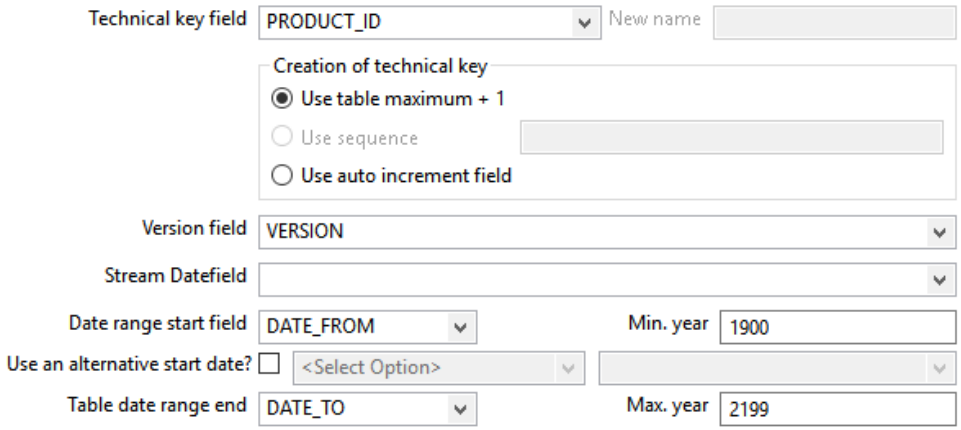
- In **Target table** click **Browse** and select the **dim_product** table
- The key that will be used to check if a product already exists in the dimension table is **PRODUCTCODE**. Therefore, configure **Keys** as follows:



- In addition, the dimension table stores the following fields for each product: **PRODUCTNAME**, **PRODUCTLINE**, **PRODUCTVENDOR**. Therefore, configure **Fields** as follows:



- Finally, configure the **Technical key field**, the **Version field**, the **Date range start field**, and the **Table date range end** as follows:



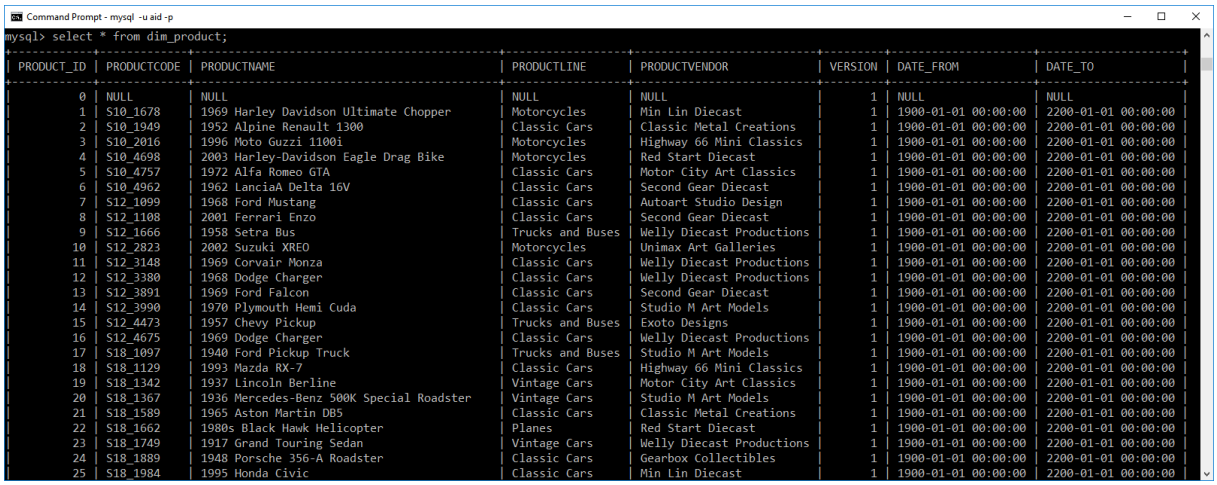
*Note: The **dim_product** dimension table will not use **PRODUCTCODE** as key. Instead, it will use **PRODUCT_ID** (an integer) as technical/surrogate key.*

*Note: **dim_product** is a **slowly-changing dimension**, meaning that there may be multiple versions of the same product, if the information about the product changes over time.*

11. Save the transformation as **/home/aid/Downloads/dim_product.ktr** (if you are on the VM).

12. Run the transformation.

13. Check that the data has been loaded into the **dim_product** table in the data warehouse. (*See the following figure.*)



PRODUCT_ID	PRODUCTCODE	PRODUCTNAME	PRODUCTLINE	PRODUCTVENDOR	VERSION	DATE_FROM	DATE_TO
0	NULL	NULL	NULL	NULL	1	NULL	NULL
1	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	Min Lin Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	Classic Metal Creations	1	1900-01-01 00:00:00	2200-01-01 00:00:00
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	Highway 66 Mini Classics	1	1900-01-01 00:00:00	2200-01-01 00:00:00
4	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	Red Start Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
5	S10_4757	1972 Alfa Romeo GTA	Classic Cars	Motor City Art Classics	1	1900-01-01 00:00:00	2200-01-01 00:00:00
6	S10_4962	1962 LanciaA Delta 16V	Classic Cars	Second Gear Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
7	S12_1099	1968 Ford Mustang	Classic Cars	Autoart Studio Design	1	1900-01-01 00:00:00	2200-01-01 00:00:00
8	S12_1108	2001 Ferrari Enzo	Classic Cars	Second Gear Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
9	S12_1666	1958 Setra Bus	Trucks and Buses	Welly Diecast Productions	1	1900-01-01 00:00:00	2200-01-01 00:00:00
10	S12_2823	2002 Suzuki XREO	Motorcycles	Unimax Art Galleries	1	1900-01-01 00:00:00	2200-01-01 00:00:00
11	S12_3148	1969 Corvair Monza	Classic Cars	Welly Diecast Productions	1	1900-01-01 00:00:00	2200-01-01 00:00:00
12	S12_3380	1968 Dodge Charger	Classic Cars	Welly Diecast Productions	1	1900-01-01 00:00:00	2200-01-01 00:00:00
13	S12_3891	1969 Ford Falcon	Classic Cars	Second Gear Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
14	S12_3990	1970 Plymouth Hemi Cuda	Classic Cars	Studio M Art Models	1	1900-01-01 00:00:00	2200-01-01 00:00:00
15	S12_4473	1957 Chevy Pickup	Trucks and Buses	Exoto Designs	1	1900-01-01 00:00:00	2200-01-01 00:00:00
16	S12_4675	1969 Dodge Charger	Classic Cars	Welly Diecast Productions	1	1900-01-01 00:00:00	2200-01-01 00:00:00
17	S18_1097	1940 Ford Pickup Truck	Trucks and Buses	Studio M Art Models	1	1900-01-01 00:00:00	2200-01-01 00:00:00
18	S18_1129	1993 Mazda RX-7	Classic Cars	Highway 66 Mini Classics	1	1900-01-01 00:00:00	2200-01-01 00:00:00
19	S18_1342	1937 Lincoln Berline	Vintage Cars	Motor City Art Classics	1	1900-01-01 00:00:00	2200-01-01 00:00:00
20	S18_1367	1936 Mercedes-Benz 500K Special Roadster	Vintage Cars	Studio M Art Models	1	1900-01-01 00:00:00	2200-01-01 00:00:00
21	S18_1589	1965 Aston Martin DB5	Classic Cars	Classic Metal Creations	1	1900-01-01 00:00:00	2200-01-01 00:00:00
22	S18_1662	1980s Black Hawk Helicopter	Planes	Red Start Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00
23	S18_1749	1917 Grand Touring Sedan	Vintage Cars	Welly Diecast Productions	1	1900-01-01 00:00:00	2200-01-01 00:00:00
24	S18_1889	1948 Porsche 356-A Roadster	Classic Cars	Gearbox Collectibles	1	1900-01-01 00:00:00	2200-01-01 00:00:00
25	S18_1984	1995 Honda Civic	Classic Cars	Min Lin Diecast	1	1900-01-01 00:00:00	2200-01-01 00:00:00

Testing the slowly-changing dimension

14. In the MySQL prompt, change to the **steelwheels** database with the command:
    use steelwheels

15. Search for cars with the following query:
    select PRODUCTCODE, PRODUCTNAME, PRODUCTLINE
    from products
    where PRODUCTLINE **like** '%Cars%' **order by** PRODUCTNAME;

16. You will see a list of **Vintage Cars** and **Classic Cars**. The first Classic Car that appears in this list is a **Porsche Roadster from 1948** (**S18_1889**). We will change this product to a Vintage Car.

17. Execute the following query:
    update products **set** PRODUCTLINE='Vintage Cars' **where** PRODUCTCODE='S18_1889';

18. Switch back to PDI (Spoon), and run the **dim_product** transformation again.

19. Now go back to the MySQL prompt, and change to **steelwheels_dw**:
    use steelwheels_dw

20. Execute the following query:
    select * **from** dim_product **where** PRODUCTCODE='S18_1889';

21. You will see that there are now two versions of the same product (same **PRODUCTCODE**, but different **PRODUCT_ID**). In version 1, the Porsche Roadster is listed as a Classic Car, and in version 2 it is listed as a Vintage Car.

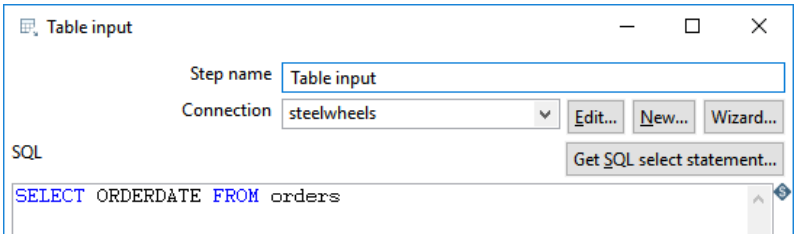
22. Check the **DATE_FROM** and **DATE_TO** fields of the two versions. When did the change from version 1 to version 2 occur? At the present time, which version is valid?

Creating the time dimension

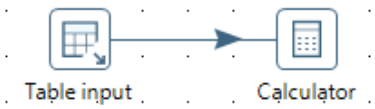
23. Create a new transformation in PDI.

24. Add a **Table Input** to the transformation and configure it as follows:

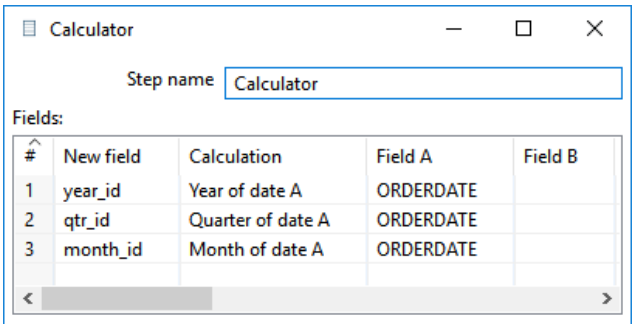
- In **Connection** select **steelwheels**
- Press the **Get SQL select statement** button and select the **orders** table
- **Do you want to include the field-names in the SQL?** Answer **Yes**
- Then remove every field except **ORDERDATE**.



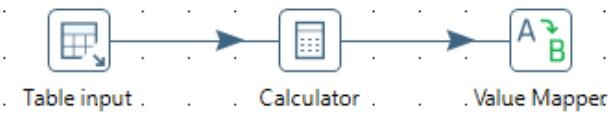
25. Add a **Calculator** step to the transformation, and create a hop from the previous step.



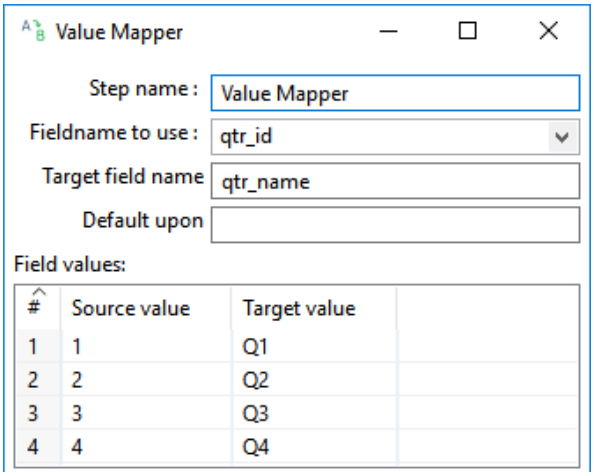
26. Configure the **Calculator** as follows:



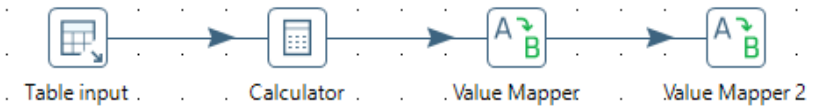
27. Add a **Value Mapper** to the transformation, and create a hop from the previous step.



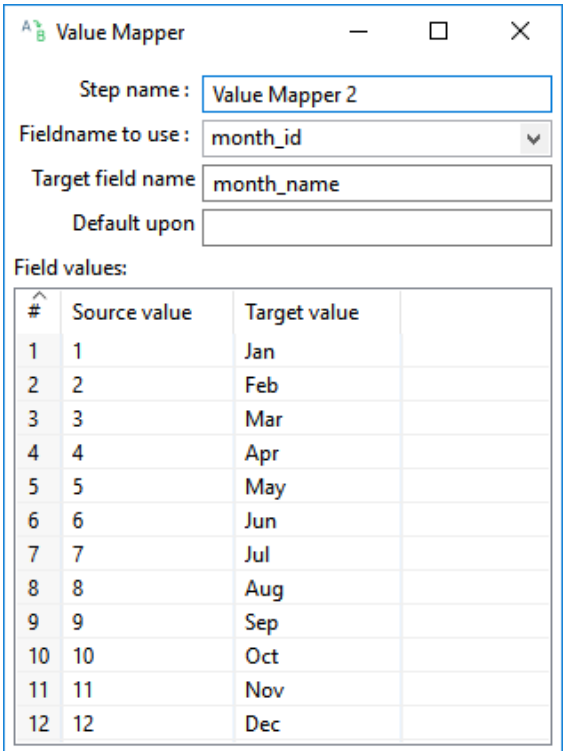
28. Configure the **Value Mapper** as follows:



29. Add a second **Value Mapper** to the transformation and connect it to the previous step.

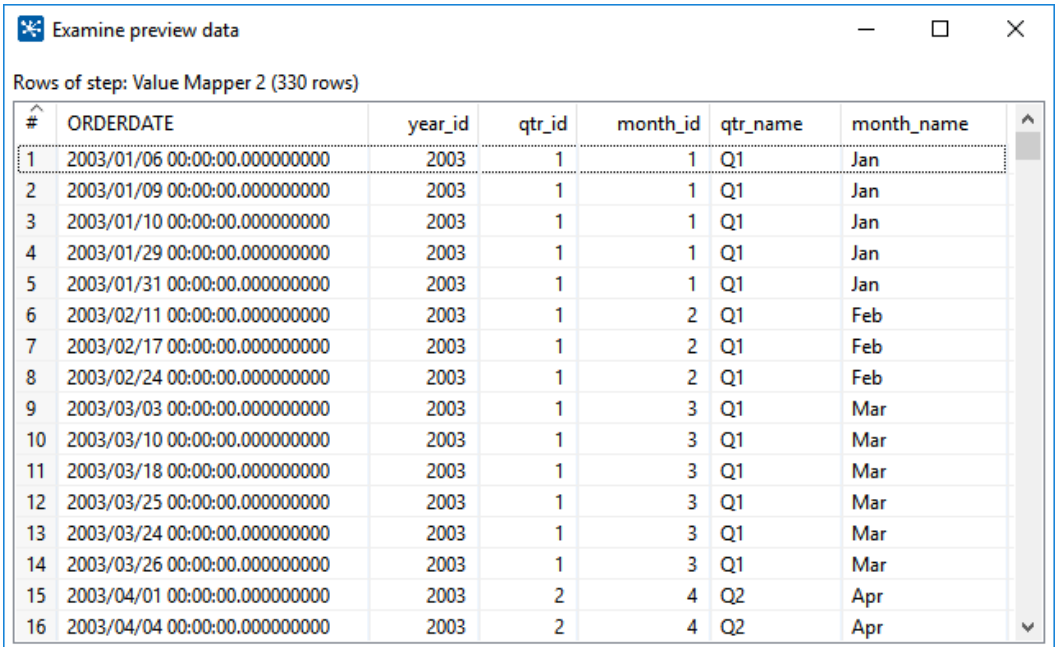


30. Configure **Value Mapper 2** as follows:



#	Source value	Target value
1	1	Jan
2	2	Feb
3	3	Mar
4	4	Apr
5	5	May
6	6	Jun
7	7	Jul
8	8	Aug
9	9	Sep
10	10	Oct
11	11	Nov
12	12	Dec

31. Do a **Preview** of **Value Mapper 2** and check that you have the following results:



#	ORDERDATE	year_id	qtr_id	month_id	qtr_name	month_name
1	2003/01/06 00:00:00.000000000	2003	1	1	Q1	Jan
2	2003/01/09 00:00:00.000000000	2003	1	1	Q1	Jan
3	2003/01/10 00:00:00.000000000	2003	1	1	Q1	Jan
4	2003/01/29 00:00:00.000000000	2003	1	1	Q1	Jan
5	2003/01/31 00:00:00.000000000	2003	1	1	Q1	Jan
6	2003/02/11 00:00:00.000000000	2003	1	2	Q1	Feb
7	2003/02/17 00:00:00.000000000	2003	1	2	Q1	Feb
8	2003/02/24 00:00:00.000000000	2003	1	2	Q1	Feb
9	2003/03/03 00:00:00.000000000	2003	1	3	Q1	Mar
10	2003/03/10 00:00:00.000000000	2003	1	3	Q1	Mar
11	2003/03/18 00:00:00.000000000	2003	1	3	Q1	Mar
12	2003/03/25 00:00:00.000000000	2003	1	3	Q1	Mar
13	2003/03/24 00:00:00.000000000	2003	1	3	Q1	Mar
14	2003/03/26 00:00:00.000000000	2003	1	3	Q1	Mar
15	2003/04/01 00:00:00.000000000	2003	2	4	Q2	Apr
16	2003/04/04 00:00:00.000000000	2003	2	4	Q2	Apr

32. Add an **Insert/Update** step to the transformation, and create a hop from the previous step.

33. Configure the **Insert/Update** step as follows:
    - In **Connection** select **steelwheels_dw**
    - In **Target table** click **Browse** and select the **dim_time** table
    - The key that will be used to check if a time already exists in the dimension table is **ORDERDATE**. Therefore, configure **The key(s) to look up the value(s)** as follows:

The key(s) to look up the value(s):

#	Table field	Comparator	Stream field1	Stream field2
1	TIME_ID	=	ORDERDATE	

*Note: The **TIME_ID** in the dimension table is mapped to the **ORDERDATE** stream field. This means that a **TIME_ID** in the data warehouse corresponds to an **ORDERDATE** in the original database.*

    - The dimension table stores the following fields for each time: **TIME_ID, YEAR_ID, QTR_ID, QTR_NAME, MONTH_ID, MONTH_NAME**. These fields will have to be inserted or updated in the table. Therefore, configure **Update fields** as follows:

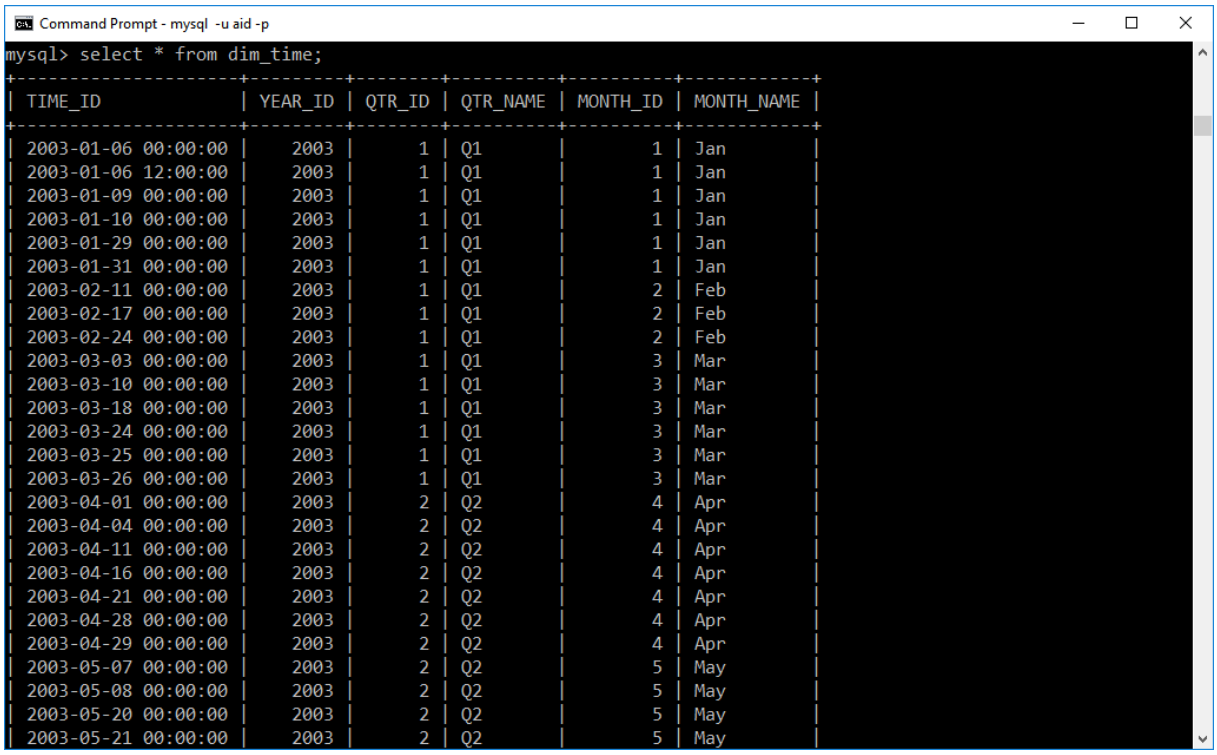
Update fields:

#	Table field	Stream field	Update
1	TIME_ID	ORDERDATE	Y
2	YEAR_ID	year_id	Y
3	QTR_ID	qtr_id	Y
4	QTR_NAME	qrt_name	Y
5	MONTH_ID	month_id	Y
6	MONTH_NAME	month_name	Y

34. Save the transformation as **/home/aid/Downloads/dim_time.ktr** (if you are on the VM).

35. Run the transformation.

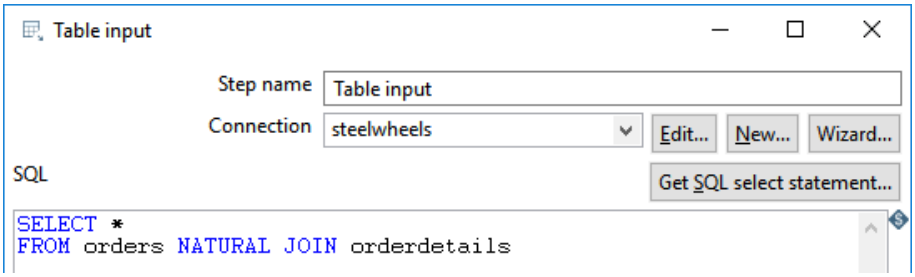
36. Check that the data has been loaded from the **orders** table in the **steelwheels** database to the **dim_time** table in the data warehouse.
    (*See the following figure.*)



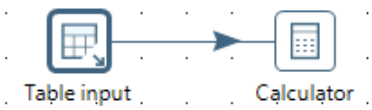
Creating the fact table

37. Create a new transformation in PDI.

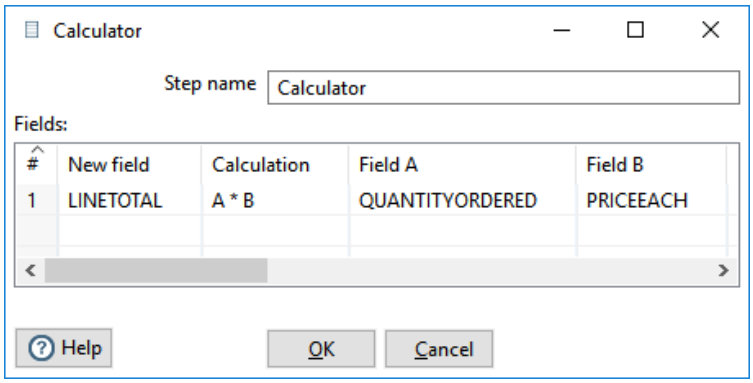
38. Add a **Table Input** to the transformation and configure it as follows:
    - In **Connection** select **steelwheels**
    - In **SQL** write the following query:
      SELECT * FROM orders **NATURAL JOIN** orderdetails



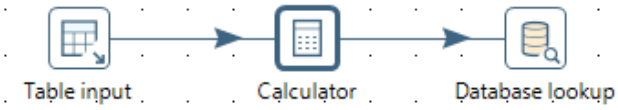
39. Add a **Calculator** step to the transformation, and create a hop from the previous step.



40. Configure the **Calculator** as follows:



41. Add a **Database lookup** step to the transformation, and create a hop from the previous step.



*Note: The purpose of this **Database lookup** is to get the **PRODUCT_ID** (the technical/surrogate key) for the **PRODUCTCODE** that comes from the **steelwheels** database.*

42. Configure the **Database lookup** as follows:
    - In **Connection** select **steelwheels_dw**
    - In **Lookup table** click **Browse** and select the **dim_product** table
    - The key that will be used to lookup the product in the dimension table is **PRODUCTCODE**. However, since a product may have multiple versions, we want to retrieve the version that was valid at the time when the order was placed. Therefore, configure **The key(s) to look up the value(s)** as follows:

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	PRODUCTCODE	=	PRODUCTCODE	
2	DATE_FROM	<=	ORDERDATE	
3	DATE_TO	>	ORDERDATE	

    - The information that we want to retrieve from the dimension table is the **PRODUCT_ID** (the technical/surrogate key). Therefore, configure the **Values to return from the lookup table** as follows:

Values to return from the lookup table :

#	Field	New name	Default	Type
1	PRODUCT_ID			Integer

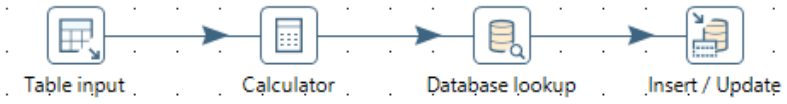
43. Do a **Preview** of the **Database lookup** in order to check that the **PRODUCT_ID** is being retrieved. (*See the following figure.*)

Examine preview data

Rows of step: Database lookup (1000 rows)

CUSTOMERNUMBER	ORDERLINENUMBER	PRODUCTCODE	QUANTITYORDERED	PRICEEACH	LINETOTAL	PRODUCT_ID
363	1	S24_3969	49	34	1666.0	80
363	2	S18_2248	50	68	3400.0	27
363	3	S18_1749	30	172	5160.0	23
363	4	S18_4409	22	87	1914.0	50
128	1	S18_2795	26	145	3770.0	33
128	2	S24_2022	46	54	2484.0	64
128	3	S24_1937	45	31	1395.0	61
128	4	S18_2325	25	151	3775.0	29
181	1	S18_1367	41	50	2050.0	20
181	2	S18_1342	39	123	4797.0	19
121	1	S24_2300	36	102	3672.0	65
121	2	S18_2432	22	54	1188.0	30
121	3	S32_1268	31	104	3224.0	85
121	4	S10_4962	42	129	5418.0	6
121	5	S18_4600	36	117	4212.0	52

44. Add an **Insert/Update** step to the transformation, and create a hop from the previous step.



45. Configure the **Insert/Update** step as follows:
    - In **Connection** select **steelwheels_dw**
    - In **Target table** click **Browse** and select the **fact_order** table
    - The key that will be used to check if a fact already exists in the fact table is **ORDERNUMBER** and **ORDERLINENUMBER**. Therefore, configure **The key(s) to look up the value(s)** as follows:

The key(s) to look up the value(s):

#	Table field	Comparator	Stream field1	Stream field2
1	ORDERNUMBER	=	ORDERNUMBER	
2	ORDERLINENUMBER	=	ORDERLINENUMBER	

    - The fact table stores the following fields for each fact: **ORDERNUMBER, ORDERLINENUMBER, QUANTITYORDERED, PRICEEACH, LINETOTAL, CUSTOMERNUMBER, PRODUCT_ID, TIME_ID**. These fields will have to be inserted or updated in the table. Therefore, configure **Update fields** as follows:

Update fields:

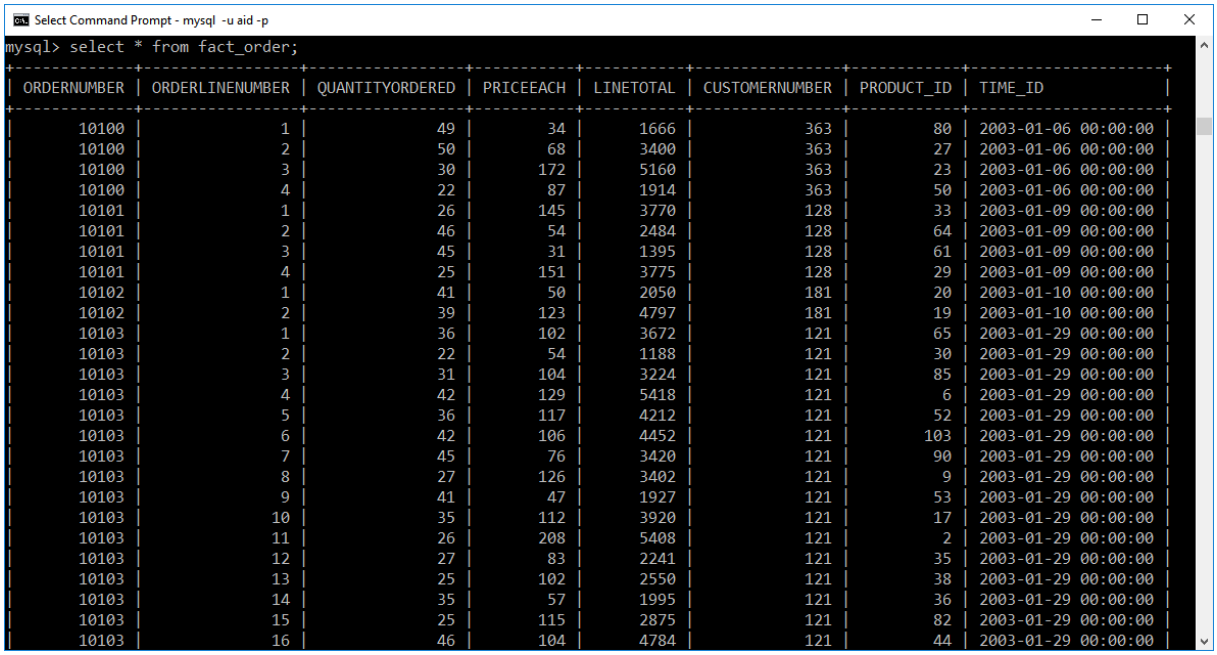
#	Table field	Stream field	Update
1	ORDERNUMBER	ORDERNUMBER	Y
2	ORDERLINENUMBER	ORDERLINENUMBER	Y
3	QUANTITYORDERED	QUANTITYORDERED	Y
4	PRICEEACH	PRICEEACH	Y
5	LINETOTAL	LINETOTAL	Y
6	CUSTOMERNUMBER	CUSTOMERNUMBER	Y
7	PRODUCT_ID	PRODUCT_ID	Y
8	TIME_ID	ORDERDATE	Y

*Note: The **TIME_ID** table field is mapped to the **ORDERDATE** stream field.*

46. Save the transformation as **/home/aid/Downloads/fact_order.ktr** (if you are on the VM).

47. Run the transformation.

48. Check that the data has been loaded into the **fact_order** table in the data warehouse.



ORDERNUMBER	ORDERLINENUMBER	QUANTITYORDERED	PRICEEACH	LINETOTAL	CUSTOMERNUMBER	PRODUCT_ID	TIME_ID
10100	1	49	34	1666	363	80	2003-01-06 00:00:00
10100	2	50	68	3400	363	27	2003-01-06 00:00:00
10100	3	30	172	5160	363	23	2003-01-06 00:00:00
10100	4	22	87	1914	363	50	2003-01-06 00:00:00
10101	1	26	145	3770	128	33	2003-01-09 00:00:00
10101	2	46	54	2484	128	64	2003-01-09 00:00:00
10101	3	45	31	1395	128	61	2003-01-09 00:00:00
10101	4	25	151	3775	128	29	2003-01-09 00:00:00
10102	1	41	50	2050	181	20	2003-01-10 00:00:00
10102	2	39	123	4797	181	19	2003-01-10 00:00:00
10103	1	36	102	3672	121	65	2003-01-29 00:00:00
10103	2	22	54	1188	121	30	2003-01-29 00:00:00
10103	3	31	104	3224	121	85	2003-01-29 00:00:00
10103	4	42	129	5418	121	6	2003-01-29 00:00:00
10103	5	36	117	4212	121	52	2003-01-29 00:00:00
10103	6	42	106	4452	121	103	2003-01-29 00:00:00
10103	7	45	76	3420	121	90	2003-01-29 00:00:00
10103	8	27	126	3402	121	9	2003-01-29 00:00:00
10103	9	41	47	1927	121	53	2003-01-29 00:00:00
10103	10	35	112	3920	121	17	2003-01-29 00:00:00
10103	11	26	208	5408	121	2	2003-01-29 00:00:00
10103	12	27	83	2241	121	35	2003-01-29 00:00:00
10103	13	25	102	2550	121	38	2003-01-29 00:00:00
10103	14	35	57	1995	121	36	2003-01-29 00:00:00
10103	15	25	115	2875	121	82	2003-01-29 00:00:00
10103	16	46	104	4784	121	44	2003-01-29 00:00:00

At this point, you have successfully loaded the data into the data warehouse. However, this requires running several transformations. We will now create a job to automate this ETL process.

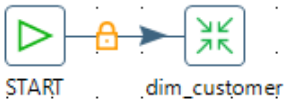
Creating a job

49. Create a new job in PDI (Spoon).

50. In the **Design** tab, expand **General**, and drag a **START** step to the canvas.

51. Add a **Transformation** step, and create a hop from the previous step.

52. Configure the **Transformation** as follows:
    - In **Entry Name** write **dim_customer**
    - In **Transformation**, write **/home/aid/Downloads/dim_customer.ktr** (if you are on the VM)



53. Add a new **Transformation** step, and create a hop from the previous step.

54. Configure the **Transformation** as follows:

- In **Entry Name** write **dim_product**
- In **Transformation**, write **/home/aid/Downloads/dim_product.ktr** (if you are on the VM)

55. Add a new **Transformation** step, and create a hop from the previous step.

56. Configure the **Transformation** as follows:
    - In **Entry Name** write **dim_time**
    - In **Transformation**, write **/home/aid/Downloads/dim_time.ktr** (if you are on the VM)

57. Add a new **Transformation** step, and create a hop from the previous step.

58. Configure the **Transformation** as follows:
    - In **Entry Name** write **fact_order**
    - In **Transformation**, write **/home/aid/Downloads/fact_order.ktr** (if you are on the VM)

59. Save the job as **/home/aid/Downloads/load_dw.kjb** (if you are on the VM)

60. Run the job.
    Note: The job runs a sequence of transformations. Each transformation runs upon successful completion of the previous one. You can run this job whenever you need to reload or update the data warehouse.

61. Take a screenshot of PDI showing that the complete job has run successfully.