**Introduction**

In the previous lessons, we used the Northwind database. Now we want to build a data warehouse (DWH) for the Northwind database. A data warehouse is simply a relational database. That is, it is a database where the data is stored in the form of tables. This data can be manipulated by means of the SQL (Structured Query Language) language. The difference with a 'regular' database is that the data warehouse is not specifically made for an application, but that it is a collection box of data from all kinds of different source databases. Because all relevant data is collected in the data warehouse, it can serve as a source for all reporting and analysis. Each report takes its data from the same database and 'tells' the same story. This makes it the 'single version of the truth '.

1. What are the potential benefits of building a DWH for the Northwind database?

If you need to create a DWH, you need to draw a star schema, according to the theory of Kimball.

A star schema is a model that is easy to understand and therefore easy to use for a subject matter expert.

A star schema consists of two types of tables:

* the fact table: is the table where quantities that make a process measurable are stored. KPIs are based on these quantities
* dimension tables: are the tables that contain the context that gives meaning to the facts.

The star schema takes its name from the fact that the fact table is central and the dimension tables are arranged around it in a kind of star structure.

**Draw a star schema**

1. Draw a star schema of the DWH to be developed

* Almost every DWH has a Date Dimension.
  + Give some examples that illustrate why a Date Dimension is useful.
  + How does a Date Dimension looks like?
  + What are columns in a Date Dimension
* Which other dimensions are you going to create?
* Are you going to put the field Phone or Fax in DimCustomer?   
  Explain why these aren't important in a DWH.
* Which dimension will be Slowly Changing Dimensions.
  + Which fields are you going to add to implement an SCD Type 2?   
    What is the meaning of an SCD?

1. What are the facts? How is the Fact table going to look like?

**Create the database NorthwindDWH + the dimenson tables + the fact table**

1. Write a .sql script to create the database NorthwindDWH, the 3 dimension tables and the fact table. Use the following code to start from.

IF NOT EXISTS (SELECT \* FROM sys.databases WHERE name = 'NorthwindDWH')

BEGIN

CREATE DATABASE NorthwindDWH;

END;

GO

USE [NorthwindDWH]

CREATE TABLE [dbo].[DimCustomer](

[CustomerKey] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

[CustomerID] [nchar](5) NULL,

[CompanyName] [nvarchar](40) NULL,

[City] [nvarchar](15) NULL,

[Region] [nvarchar](15) NULL,

[Country] [nvarchar](15) NULL,

[StartDate] [date] NULL,

[EndDate] [date] NULL

)

CREATE TABLE [dbo].[DimDate](

[DateKey] [int] NOT NULL PRIMARY KEY,

[FullDateAlternateKey] [date] NOT NULL,

[DayOfMonth] [varchar](2) NULL,

[EnglishDayNameOfWeek] [varchar](10) NOT NULL,

[DutchDayNameOfWeek] [varchar](10) NOT NULL,

[DayOfWeek] [char](1) NULL,

[DayOfWeekInMonth] [varchar](2) NULL,

[DayOfWeekInYear] [varchar](2) NULL,

[DayOfQuarter] [varchar](3) NULL,

[DayOfYear] [varchar](3) NULL,

[WeekOfMonth] [varchar](1) NULL,

[WeekOfQuarter] [varchar](2) NULL,

[WeekOfYear] [varchar](2) NULL,

[Month] [varchar](2) NULL,

[EnglishMonthName] [varchar](10) NOT NULL,

[DutchMonthName] [varchar](10) NOT NULL,

[MonthOfQuarter] [varchar](2) NULL,

[Quarter] [char](1) NULL,

[QuarterName] [varchar](9) NULL,

[Year] [char](4) NULL,

[MonthYear] [char](10) NULL,

[MMYYYY] [char](6) NULL

)

…

1. Add the Foreign Key constraints
   * ProductKey in FactSales is a foreign key to ProductKey in DimProduct
   * CustomerKey in FactSales is a foreign key to CustomerKey in DimCustomer
   * OrderDateKey in FactSales is a foreign key to DateKey in DimDate

**DimDate**

Almost every data warehouse has a date dimension, because time plays an important role in our lives.

An important goal of Business Intelligence is to substantiate feelings that arise from experience with hard facts. It follows from this that it is important to be able to compare matters such as turnover against periods. In this way, periods can be compared and growth and contraction calculated. There is almost no company that can do without this kind of comparison. Therefore, almost every data warehouse has a date dimension. Forecasting is also an important process in many organizations. That is actually trying to look ahead in time. With trend analyzes and a bit of machine learning, we can make increasingly better predictions with good data. Again, time plays an important role here.

If you include a column in the fact table containing the date of when that fact occurred, ie the order date if it concerns an order fact table, you are not there yet. With that date you can calculate what you need: the year, the quarter, the month, the name of the day and so on, on which a certain order was placed. However, in analyses, you often want to calculate the turnover for each month, each quarter, … or you want to find out on which day of the week you sell most. So instead of recalculating the name of the day or the name of the month over and over again, you can do this one time and store the result in a date dimension. So the date dimension contains descriptive information. You should think of the name of the day, the week number in which that day falls, the month of that day, both as a number and by name (so month number is 1 and month name is January), whether the day is a working day, and so on.

In some data warehouses, it must be possible to analyze the facts down to the second. This means, by the same principle, that you create a record for every possible second. If you want to store ten years of history, you get over 315 million records. It is probably advisable to create a separate date and time dimension. These contain over 3,600 and over 86,000 records respectively.

The key of a date dimension preferably looks like the example this: 20230611

This is an integer and is also stored as such in the date dimension. It doesn't take much imagination to recognize the date June 11, 2023 in this number. The advantage of a key like this is that integers are more efficient to store and provide better performance than dates. By first including the year and then the month and day, the correct sorting order is maintained.

The above is not true for all database products. In some products, using a real date instead of the smart key is better.

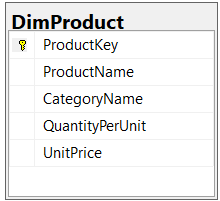
Scripts to generate date dimensions are available for download from many places.

1. Download from Chamilo the file Documenten > Scripts > Fill\_DimDate.sql. In this file you find the code of the stored procedure FillDimDate
2. Make sure you understand this script. This script is retrieved from <https://www.codeproject.com/Articles/647950/Create-and-Populate-Date-Dimension-for-Data-Wareho> and will be used to fill DimDate
3. Write a stored procedure CheckAndUpdateDimDate
   * Calculate the minimum date in OrderDate.
   * Calculate the minimum date in DimDate.
   * Calculate the maximum date in DimDate.
   * If DimDate is empty or (the minimum date in OrderDate < minimum date in DimDate) or (today > maximum date in DimDate)
     + Delete everything from DimDate
     + Fill DimDate with the appropriate dates (from januari, 1 of the minimum year in OrderDate until December, 31 of the current year).   
       Use the stored procedure FillDimDate.
4. Execute the stored procedure CheckAndUpdateDimDate.
5. Check if NorthwindDWH > DimDate contains the correct data

Afbeelding met tekst, schermopname, nummer

Automatisch gegenereerde beschrijving

**Fill DimProduct 🡪 Fill dimension using a view**

1. We need to fill DimProduct in NorthwindDWH using the data from Northwind.
   * If a product disappeared out of Northwind, nothing should be done.
   * If the properties of a product change, the changes should be tranferred to NorthwindDWH
   * If a new product was inserted into Northwind, the new product should be inserted into NorthwindDWH
2. First create a view in the **OLTP database Northwind.** We will use this view to fill **DimProduct**

CREATE VIEW VWProduct AS

SELECT p.ProductID ProductKey, p.ProductName, c.CategoryName, p.QuantityPerUnit, p.UnitPrice

FROM products p JOIN categories c ON p.CategoryID = c.CategoryID;

1. Create a stored procedure CheckAndUpdateDimProduct that contains 1 MERGE – statement
   * If a product disappeared out of Northwind, nothing should be done.
   * If the properties of a product change, the changes should be tranferred to NorthwindDWH
   * If a new product was inserted into Northwind, insert the new product also into NorthwindDWH

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

-- Fill DimProduct

-- Add products to DimProduct that are already in the OLTP database (Products)

-- but that aren't yet in the DWH (DimProduct)

-- It's possible that there are products in DimProduct that aren't in Products any more,

-- but that is no problem.

CREATE OR ALTER PROCEDURE CheckAndUpdateDimProduct

AS

BEGIN

MERGE NorthwindDWH.dbo.DimProduct as t -- t = target

USING Northwind.dbo.VWProduct as s -- s = source

ON (t.ProductKey = s.ProductKey)

-- Which rows are in source and have different values for ProductName, CategoryName,

-- QuantityPerUnit or UnitPrice?

-- Update those rows in target with the values coming from source

WHEN MATCHED AND -- to be completed: there are some changes to productname, …

THEN UPDATE SET -- to be completed: the new values should be transferred to DimProduct

-- Which rows are in Product and not in DimProduct?

-- Insert those rows from source into target

WHEN NOT MATCHED BY target --> rows to insert

THEN INSERT VALUES -- to be completed

~~-- Which rows are in DimProduct and not any more in Product?~~

~~-- Do nothing~~

~~WHEN NOT MATCHED BY source~~

END

1. Execute the stored procedure CheckAndUpdateDimProduct.
2. Check if NorthwindDWH > DimProduct contains the correct data

Afbeelding met tekst, Lettertype, nummer, schermopname

Automatisch gegenereerde beschrijving

1. Make some changes to Product in Northwind

UPDATE Products

SET UnitPrice = 30

WHERE ProductID = 1

INSERT INTO Products(ProductName, CategoryID, UnitPrice)

VALUES('Tiramisu', 2, 3.0)

1. Execute the stored procedure CheckAndUpdateDimProduct.
2. Check if the changes are visible in DimProduct

Afbeelding met tekst, Lettertype, lijn, nummer

Automatisch gegenereerde beschrijving

1. Make again some changes to Product in Northwind (to restore the original price of the product Chai. The product Tiramisu stays in the DWH!)

UPDATE Products

SET UnitPrice = 18.0

WHERE ProductID = 1

DELETE FROM Products

WHERE ProductName = 'Tiramisu' and UnitPrice = 3.0

1. Execute the stored procedure CheckAndUpdateDimProduct.
2. Check if the changes are visible in DimProduct

Afbeelding met tekst, Lettertype, nummer, schermopname

Automatisch gegenereerde beschrijving

…

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

**Fill DimCustomer 🡪 Slowly changing dimension**

1. DimCustomer is a Slowly Changing Dimension Type 2. The term 'Slowly' mainly refers to the fact that some data changes only occasionally. These are ad hoc changes. For example, customers who move. That happens occasionally, but certainly not every day, let alone several times a day.

If you later calculate the turnover per customer country, after a customer has moved, and you have overwritten the old value for country with the new value, then all orders coming from this customer are registered with his new country.

Slowly Changing Dimension type 2 is intended for this kind of situation. With an SCD type 2 you create a completely new record with every change. All columns that did not change, are adopted unchanged.   
So, there are 'current' records and 'closed' records. 'Current' records are those records with 'Null' as EndDate. 'Closed' records are those records with an actual date as EndDate. In that case, the StartDate and EndDate denote the time span for which those values for region, country, … were true for this specific customer.  
Note that it is still clear that it concerns the same customer, as the customerid remains unchanged.

1. We need to fill DimCustomer in NorthwindDWH using the data from Northwind.
   * If a customer disappeared out of Northwind, nothing should be done.
   * If a new customer was inserted into Northwind, the new customer should be inserted into NorthwindDWH. The Startdate is today.
   * If the column values of a customer have changed in Northwind
     + The Enddate in DimCustomer should get the date of yesterday
     + A new record is inserted into DimCustomer with the new values and the Startdate is today
2. Perhaps, it seems as a good idea to use a MERGE statement to implement the filling and updating of DimCustomer, something that looks like the following

MERGE NorthwindDWH.dbo.DimCustomer as t -- t = target

USING Northwind.dbo.Customers as s -- s = source

ON (t.CustomerID = s.CustomerID)

-- Which rows are in source and have different values for CompanyName, City, Region or Country?

-- Set EndDate to yesterday

-- Insert a new record with the new values

WHEN MATCHED AND t.EndDate IS NULL AND (s.CompanyName <> t.CompanyName OR s.City <> t.city OR s.Region <> t.region OR s.country <> t.country)

Not allowed

THEN UPDATE SET t.EndDate = @yesterday

THEN INSERT (CustomerID, CompanyName, City, Country, StartDate) VALUES (s.CustomerID, s.CompanyName, s.City, s.Country, GETDATE());

-- Which rows are in source and not in target?

-- Insert those rows from source into target

WHEN NOT MATCHED BY target

THEN INSERT (CustomerID, CompanyName, City, Country, StartDate) VALUES (s.CustomerID, s.CompanyName, s.City, s.Country, GETDATE());

-- Which rows are in target and not in source?

-- Do nothing

~~WHEN NOT MATCHED BY source~~

1. Unfortunately this can’t be solved using the MERGE statement because the INSERT command is not allowed in combination with 'WHEN MATCHED'. So we’ll need to find another way
2. We will use the following stored procedure to fill and update DimCustomer

-- Fill DimCustomer -- Slowly Changing Dimension

CREATE OR ALTER PROCEDURE CheckAndUpdateDimCustomer

AS

BEGIN

-- Create a temporary table #CustomerTempTable to contain the values that have changed

SELECT dc.CustomerKey, c.customerID, c.companyName, c.city, c.region, c.country

INTO #CustomerTempTable

FROM NorthwindDWH.dbo.DimCustomer dc JOIN Northwind.dbo.Customers c

ON dc.CustomerID = c.CustomerID

WHERE dc.EndDate IS NULL AND (dc.CompanyName <> c.CompanyName OR dc.City <> c.city OR dc.Region <> c.region OR dc.country <> c.country)

-- Change the EndDate of those records which contain old values to yesterday

DECLARE @yesterday DATETIME = (SELECT DATEADD(day, -1, GETDATE()))

UPDATE NorthwindDWH.dbo.DimCustomer

SET EndDate = @yesterday

WHERE customerKey IN (SELECT CustomerKey FROM #CustomerTempTable)

-- Insert these records (from the temporary table) into NorthwindDWH.dbo.DimCustomer.

-- Use Today as StartDate.

INSERT INTO NorthwindDWH.dbo.DimCustomer(CustomerID, CompanyName, City, Region, Country, StartDate)

SELECT CustomerID, CompanyName, City, Region, Country, GETDATE()

FROM #CustomerTempTable

DROP TABLE #CustomerTempTable

-- Insert the entirely new customers from Northwind.dbo.Customers into DimCustomer.

-- This will also make sure that DimCustomer get’s filled for the first time (if it was still empty)

-- If DimCustomer was empty => Startdate is the minimum date of Orders for the newly inserted records

DECLARE @StartDate DATETIME = (SELECT MIN(OrderDate) FROM Northwind.dbo.Orders)

-- If DimCustomer already contains values => Startdate = today for the newly inserted records

IF EXISTS (SELECT \* FROM DimCustomer)

SET @StartDate = GETDATE()

INSERT INTO DimCustomer(CustomerID, CompanyName, City, Region, Country, StartDate)

SELECT CustomerID, CompanyName, City, Region, Country, @StartDate

FROM Northwind.dbo.Customers

WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM NorthwindDWH.dbo.DimCustomer)

END

1. To understand why this stored procedure is ‘doing the job’, take a look at the following steps.
2. DimCustomer is empty at the beginning.
3. Execute the stored procedure CheckAndUpdateDimCustomer.
4. Check if NorthwindDWH > DimCustomer contains the correct data. Notice that the value of the StartDate is 2016-07-04, which is the minimum order date in Northwind = the start date of the database.

Afbeelding met tekst, schermopname, nummer, Lettertype

Automatisch gegenereerde beschrijving

1. It’s the second part of CheckAndUpdateDimCustomer that is responsible for filling DimCustomer the first time. It adds all the customers that aren’t in DimCustomer yet (CustomerID isn’t in DimCustomer yet) to DimCustomer

-- Insert the entirely new customers from Northwind.dbo.Customers into DimCustomer.

-- This will also make sure that DimCustomer get’s filled for the first time (if it was still empty)

-- If DimCustomer was empty => Startdate is the minimum date of Orders for the newly inserted records

DECLARE @StartDate DATETIME = (SELECT MIN(OrderDate) FROM Northwind.dbo.Orders)

-- If DimCustomer already contains values => Startdate = today for the newly inserted records

IF EXISTS (SELECT \* FROM DimCustomer)

SET @StartDate = GETDATE()

INSERT INTO DimCustomer(CustomerID, CompanyName, City, Region, Country, StartDate)

SELECT CustomerID, CompanyName, City, Region, Country, @StartDate

FROM Northwind.dbo.Customers

WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM NorthwindDWH.dbo.DimCustomer)

1. Insert a new record in Customers in Northwind

INSERT INTO Customers(CustomerID, CompanyName, City, PostalCode, Country)

VALUES ('DELDI','Delicious Dining', 'Aalst', '9300', 'Belgium')

1. Execute the stored procedure CheckAndUpdateDimCustomer.
2. Check if the changes are visible in DimCustomer
3. Notice that the StartDate now is today.
4. Again, make some changes to Customer in Northwind

UPDATE Customers SET postalCode = 'B-9000', city = 'Ghent' WHERE CustomerID = 'MAISD';

UPDATE customers SET city = 'Oxford' WHERE CustomerID = 'AROUT';

1. Execute the stored procedure CheckAndUpdateDimCustomer.
2. Check if the changes are visible in DimCustomer

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

1. It’s the first part of CheckAndUpdateDimCustomer that is responsible for updating already existing customers in DimCustomer to their new values.
   * The temporary table contains the new values for old (already existing) records in DimCustomer (including CustomerKey!). Notice that we added dc.EndDate IS NULL to the WHERE – clause. This is because we only want to compare the values of Northwind.dbo.Customers to the most recent values in DimCustomer.
   * We use this temporary table (and more specifically the CustomerKey’s ) to update the EndDate of those changed records in DimCustomer to yesterday
   * We use this temporary table to insert the new values for the already existing records into DimCustomer. The StartDate is today. There is no EndDate, meaning this is the most recent version of the data.
2. Make some changes to Customer in Northwind

UPDATE Customers SET city = 'Paris' WHERE CustomerID = 'BONAP';

UPDATE customers SET city = 'Berlin' WHERE CustomerID = 'PICCO';

INSERT INTO Customers(CustomerID, CompanyName, City, PostalCode, Country)

VALUES ('CHOCO','Chocolat Corner', 'Brugge', '8000', 'Belgium')

1. Execute the stored procedure CheckAndUpdateDimCustomer.
2. Check if the changes are visible in DimCustomer

**Fill FactSales 🡪 fill fact table**

1. We will use the following SQL statement to fill the table FactSales repeatedly

INSERT INTO NorthwindDWH.dbo.FactSales(OrderLine, ProductKey, CustomerKey, OrderDateKey, OrderUnitPrice, OrderQuantity, OrderDiscount)

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

AND

/\* only add new lines + make sure it runs from an empty FactSales table \*/

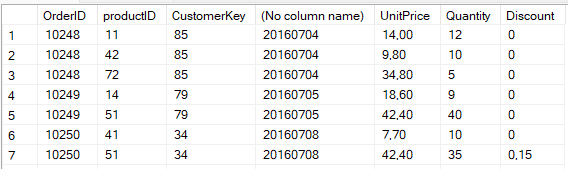
o.OrderID > (SELECT ISNULL(MAX(OrderLine),0) from NorthwindDWH.dbo.FactSales)

1. To understand why this SQL statement is 'doing the job', take a look at the following steps
   * Execute the following SQL statement in the DWH

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join DimCustomer dc on o.CustomerID = dc.CustomerID



* + In case there is 1 CustomerKey for a CustomerID, there is no problem.

However, if there are multiple keys for 1 CustomerID; e.g. voor MAISD, the result looks like this

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

o.CustomerID = 'MAISD'

Afbeelding met tekst, nummer, menu, schermopname

Automatisch gegenereerde beschrijving

* + This is no good. Each order is joined with both CustomerKey’s (50 and 92) for Maison Dewey.
  + To solve this problem, we have to know which of both CustomerKeys was valid at the time of the order. We add a WHERE – clause to the previous SELECT – statement

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

* + Let’s check for the orders of Maison Dewey

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

AND

o.CustomerID = 'MAISD'

Afbeelding met tekst, schermopname, nummer, menu

Automatisch gegenereerde beschrijving

* + This looks good for the orders of Maison Dewey in the past. But what if there is a recent order of Maison Dewey? To make sure, the SQL command is correct, we add a new order for Maison Dewey, and the order concerns 1 product, i.e. the product with productID = 55

INSERT INTO Orders(CustomerID, EmployeeID, OrderDate)

VALUES ('MAISD', 5, GETDATE())

DECLARE @most\_recent\_orderID INT = (select @@IDENTITY)

INSERT INTO OrderDetails(OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES(@most\_recent\_orderID, 55, 24, 10, 0)

* + Let’s check again for the orders of Maison Dewey

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

AND

o.CustomerID = 'MAISD'

Afbeelding met tekst, nummer, schermopname, menu

Automatisch gegenereerde beschrijving

* + This looks OK
  + We need to insert all of the records of the SELECT statement into the fact table of the DWH. So we add the INSERT – statement

INSERT INTO NorthwindDWH.dbo.FactSales(OrderLine, ProductKey, CustomerKey, OrderDateKey, OrderUnitPrice, OrderQuantity, OrderDiscount)

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

* + At first sight, this looks OK: the fact table is being filled.
  + But what happens if we add a new order and 2 new orderlines for Maison Dewey in Northwind?

INSERT INTO Orders(CustomerID, EmployeeID, OrderDate)

VALUES ('MAISD', 5, GETDATE())

DECLARE @most\_recent\_orderID INT = (select @@IDENTITY)

INSERT INTO OrderDetails(OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES(@most\_recent\_orderID, 68, 15, 10, 0)

INSERT INTO OrderDetails(OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES(@most\_recent\_orderID, 28, 45.6, 10, 0)

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

* + We need to copy the new orderlines from Northwind to the fact table of NorthwindDWH using the previous SQL command

INSERT INTO NorthwindDWH.dbo.FactSales(OrderLine, ProductKey, CustomerKey, OrderDateKey, OrderUnitPrice, OrderQuantity, OrderDiscount)

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

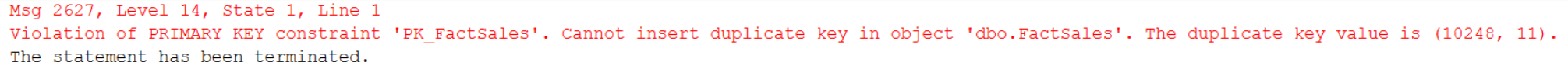
join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

* + We get an error message



* + The error message stems from the fact that we try to copy all the orderlines again into the fact table. This is wrong, we only need to copy the new orderlines to the fact table.
  + Let’s change the SQL statement once again

INSERT INTO NorthwindDWH.dbo.FactSales(OrderLine, ProductKey, CustomerKey, OrderDateKey, OrderUnitPrice, OrderQuantity, OrderDiscount)

SELECT od.OrderID, od.productID, dc.CustomerKey, CAST(FORMAT(o.OrderDate,'yyyyMMdd') as int), od.UnitPrice, od.Quantity, od.Discount

from Northwind.dbo.OrderDetails od JOIN Northwind.dbo.Orders o ON od.OrderID = o.OrderID

join NorthwindDWH.dbo.DimCustomer dc on o.CustomerID = dc.CustomerID

WHERE

/\* Slowly Changing Dimension dimCustomer \*/

o.OrderDate >= dc.startDate and (dc.EndDate is null or o.orderdate <= dc.EndDate)

AND

/\* only add new lines + make sure it runs if FactSales is empty \*/

o.OrderID > (SELECT ISNULL(MAX(OrderLine),0) from NorthwindDWH.dbo.FactSales)

* + This seems to give the correct result

Afbeelding met tekst, Lettertype, lijn, nummer

Automatisch gegenereerde beschrijving

* + Finally, let’s wrap the entire SQL statement into a stored procedure UpdateFactSales
  + To repeatedly update the DWH, we have to execute the 4 stored procedures

EXEC CheckAndUpdateDimDate

EXEC CheckAndUpdateDimProduct

EXEC CheckAndUpdateDimCustomer

EXEC UpdateFactSales

* + We can wrap the stored procedures into 1 stored procedure

CREATE OR ALTER PROCEDURE UpdateDWH

AS

BEGIN

EXEC CheckAndUpdateDimDate

EXEC CheckAndUpdateDimProduct

EXEC CheckAndUpdateDimCustomer

EXEC UpdateFactSales

END

* + Final check

-- Changes made to Northwind

UPDATE Products

SET UnitPrice = 25.0

WHERE ProductID = 1

INSERT INTO Products(ProductName, CategoryID, UnitPrice)

VALUES('Chocomousse', 2, 3.0)

DECLARE @most\_recent\_productID INT = (select @@IDENTITY)

UPDATE customers SET city = 'Glasgow' WHERE CustomerID = 'AROUT';

INSERT INTO Customers(CustomerID, CompanyName, City, PostalCode, Country)

VALUES ('FANFO','Fantastic Food', 'Kortrijk', '8500', 'Belgium')

INSERT INTO Orders(CustomerID, EmployeeID, OrderDate)

VALUES ('AROUT', 5, GETDATE())

DECLARE @most\_recent\_orderID INT = (select @@IDENTITY)

INSERT INTO OrderDetails(OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES(@most\_recent\_orderID, @most\_recent\_productID, 4.5, 10, 0)

INSERT INTO OrderDetails(OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES(@most\_recent\_orderID, 28, 47, 10, 0)

-- Executing the stored procedure UpdateDWH in NorthwindDWH

EXEC UpdateDWH

-- Checking the results in NorthwindDWH

-- DimCustomer

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

-- DimProduct

Afbeelding met tekst, nummer, Lettertype, schermopname

Automatisch gegenereerde beschrijving

…



-- FactSales

