Agile BI in der Praxis

Data Warehouse Automation

DWH Automation – darüber wird zurzeit viel gesprochen und geschrieben. Aber wie sieht die praktische Anwendung aus? In diesem Artikel führe ich Sie Schritt für Schritt durch die Erstellung eines kleinen DWHs und erläutere dabei, welche Aufgaben die DWH-Automatisierungslösung dabei übernimmt.

Natürlich ist es auch bei der DWH Automation so, dass es ein breites Spektrum von Lösungsansätzen gibt. Dabei lassen sich zwei wesentliche Vorgehensweisen unterscheiden [Eck15]:

- Model-Driven: Bei diesem Ansatz versucht man, die Anforderungen zuerst in einem konzeptionellen und oder logischen Datenmodell zu erfassen, bevor man es danach implementiert. Die Umsetzung in physikalische Modelle und Code wird dabei getrieben durch die logischen Modelle. Im Fehlerfall (zum Beispiel aufgrund unerwarteter Eigenheiten in den Quelldaten) muss erst das Modell angepasst werden und die Umsetzung erneut generiert werden.
- (Meta) Data-Driven: Bei diesem Ansatz arbeitet man so bald als möglich direkt mit verfügbaren Quelldaten und implementiert das DWH auf dem physikalischen Layer Schritt für Schritt. Dabei stellt man es sofort fest, wenn es zu Fehlern kommt, zum Beispiel aufgrund unerwarteter Eigenheiten in den Quelldaten. Darauf basierend kann man die Umsetzung anpassen.

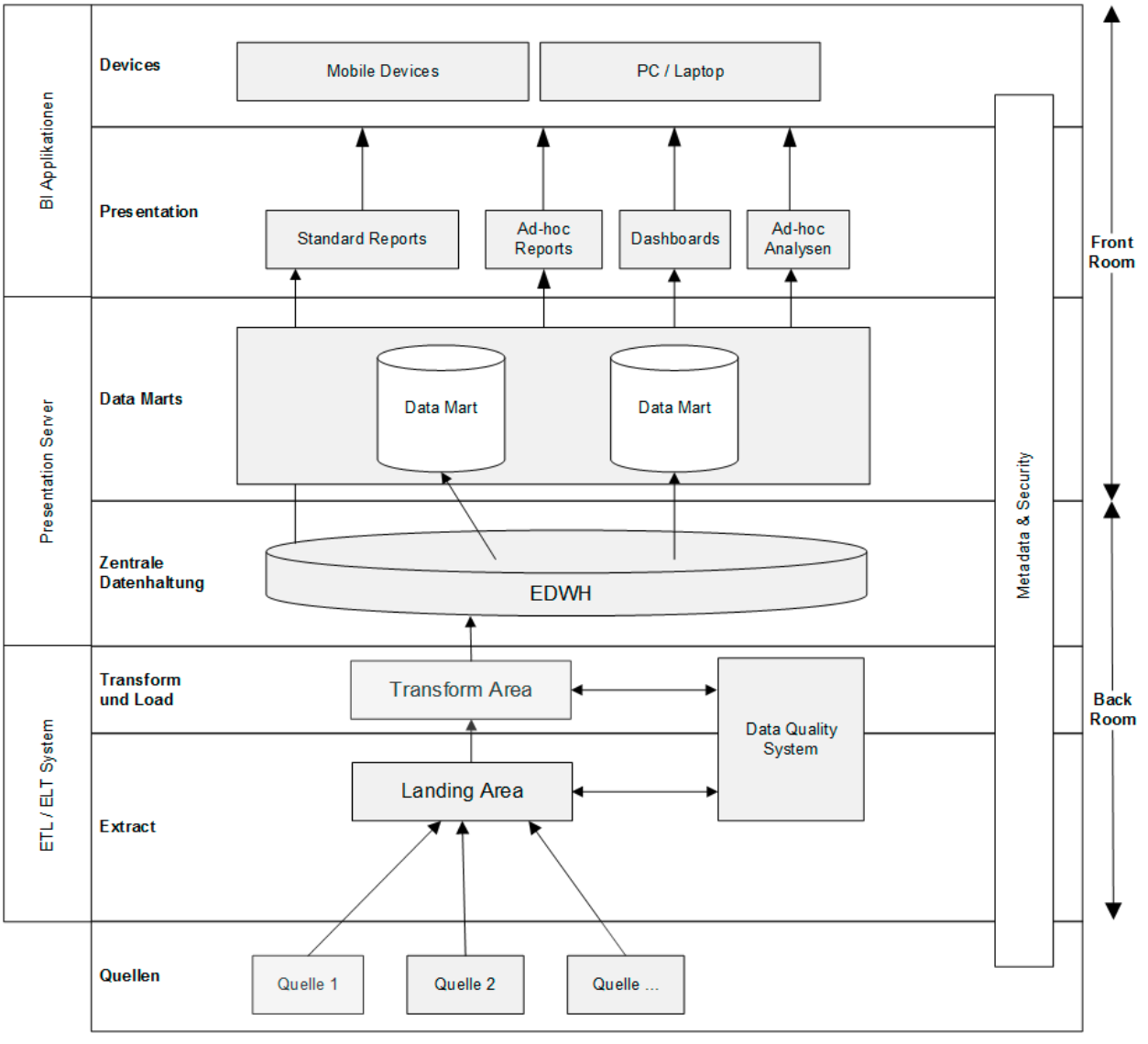


Abb. 1: Skizze einer typischen DWH-Schichtenarchitektur

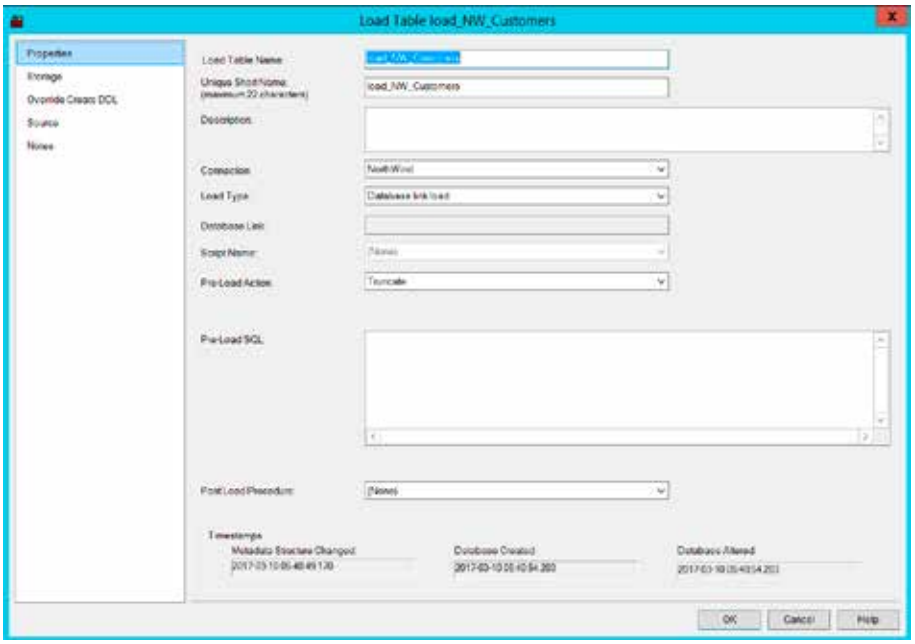


Abb. 2: Konfigurationsdialog der „Customer"-Load-Tabelle

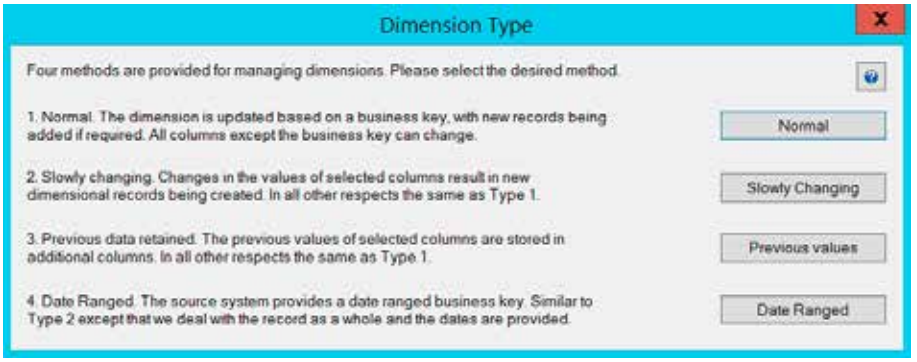


Abb. 3: Auswahlmöglichkeiten an Design-Patterns zur Historisierung einer Dimensionstabelle

In diesem Artikel fokussiere ich auf den Data-Driven-Ansatz und zeige ein Beispiel aus meiner Arbeit mit dem Werkzeug WhereScape RED. Ausgangslage ist dabei die folgende Architekturskizze, wobei als Quellsystem die Beispieldatenbank Northwind zum Einsatz kommt (Abbildung 1).

Vom Extract Layer bis zum EDWH

Zuerst einmal müssen die Daten aus den Quellen extrahiert werden. WhereScape setzt dabei auf einen ELT-Ansatz, das heißt, die Daten werden zuerst einmal in die Zieldatenbank des DWHs geladen. In WhereScape legt man dafür eine „Load Table" an (vgl. Abbildung 2).

Für dieses einfache Beispiel nutze ich auf der Ebene EDWH eine Star-Schema-Modellierung. Daher baue ich als Nächstes die Kundendimension.

Per Drag & Drop ziehe ich die Felder aus der soeben angelegten Load-Tabelle in den Bereich der Dimensionen. Dabei ist die zentrale Frage, welche Art von Historisierung ich mir wünsche (vgl. Abbildung 3).

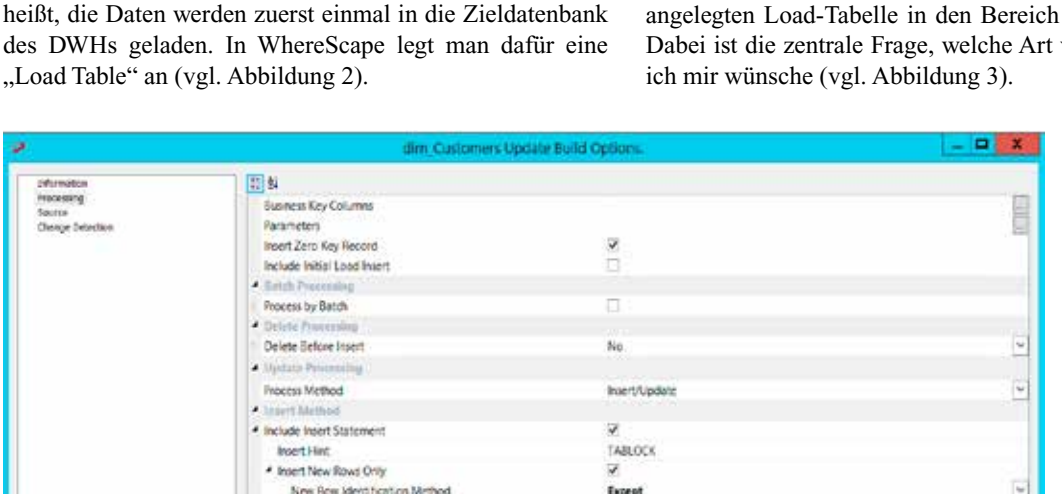


Abb. 4: Konfigurationsdialog der „Customer"-Dimensionstabelle

```
-- Update expiring rows
UPDATE [TABLEOWNER].[dim_Customers] WITH ( TABLOCK )
SET     dss_end_date = @v_current_date -0.00000005
      , dss_current_flag = 'N'
      , dss_update_time = @v_current_datetime
FROM
(
  SELECT load_NW_Customers.customerid  customerid
       , load_NW_Customers.companyname  companyname
       , load_NW_Customers.contactname  contactname
       , load_NW_Customers.contacttitle  contacttitle
       , load_NW_Customers.address  address
       , load_NW_Customers.city  city
       , load_NW_Customers.region  region
       , load_NW_Customers.postalcode  postalcode
       , load_NW_Customers.country  country
       , load_NW_Customers.phone  phone
       , load_NW_Customers.fax  fax
  FROM [TABLEOWNER].[load_NW_Customers] load_NW_Customers
  EXCEPT
  SELECT dim_Customers.customerid  customerid
    , dim_Customers.companyname  companyname
    , dim_Customers.contactname  contactname
    , dim_Customers.contacttitle  contacttitle
    , dim_Customers.address  address
    , dim_Customers.city  city
    , dim_Customers.region  region
    , dim_Customers.postalcode  postalcode
    , dim_Customers.country  country
    , dim_Customers.phone  phone
    , dim_Customers.fax  fax
  FROM [TABLEOWNER].[dim_Customers]
  WHERE dim_Customers.dss_current_flag = 'Y'
) AS changes
WHERE dim_Customers.customerid = changes.customerid
AND   dim_Customers.dss_current_flag = 'Y'
AND   (
        dim_Customers.companyname <> changes.companyname
      OR
        dim_Customers.contactname <> changes.contactname
      OR
        dim_Customers.contacttitle <> changes.contacttitle
      OR
        dim_Customers.address <> changes.address
      OR
```

Abb. 5: Auszug aus dem SQL-Code zur Beladung der „Customer"-Dimensionstabelle

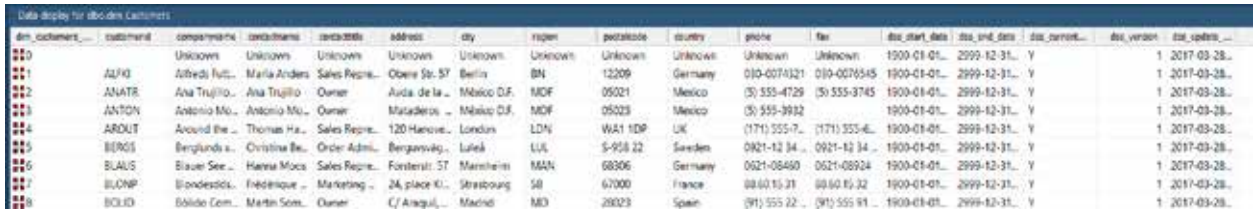
In diesem Fall hätte ich gerne eine Slowly-Changing-Dimension. In der Folge muss ich nun noch einige weitere Angaben machen. Unter anderem muss ich spezifizieren, welches meine Business-Key-Spalte ist. Zudem habe ich zahlreiche Möglichkeiten, die anschließende Code-Generierung zu steuern (Abbildung 4).

Im Hintergrund resultiert aus diesen Angaben die Erstellung der eigentlichen Tabelle und des SQL-Codes zu ihrer Beladung (Abbildung 5).

Abbildung 6 zeigt die Datenvorschau der neu angelegten Dimension. Ganz links sieht man: Für Dimensionen wurde automatisch ein Surrogate-Key-Feld angelegt. Aufgrund der Wahl „Slowly Changing" gibt es zwei Spalten zur Gültigkeitsdauer eines Datensatzes mit Start- und Enddatum sowie die Spalte mit dem Current-Flag. Zudem wurden bereits die gebräuchlichsten Indexe angelegt, und dies alles ohne mein eigenes Zutun (siehe Abbildung 7).

Als Nächstes möchte ich nun die Dimensionsdaten mit den Faktendaten zusammenbringen. Dazu lege ich eine Stage-Tabelle an und ziehe die benötigen Felder erneut per Drag & Drop zusammen (siehe Abbildung 8).

Im Falle der Dimensionen wird nur das Surrogate-Key-Feld hinzugefügt. Bezüglich der Fakten stammen die Daten aus zwei Quelltabellen, nämlich Orders und Order_Details.

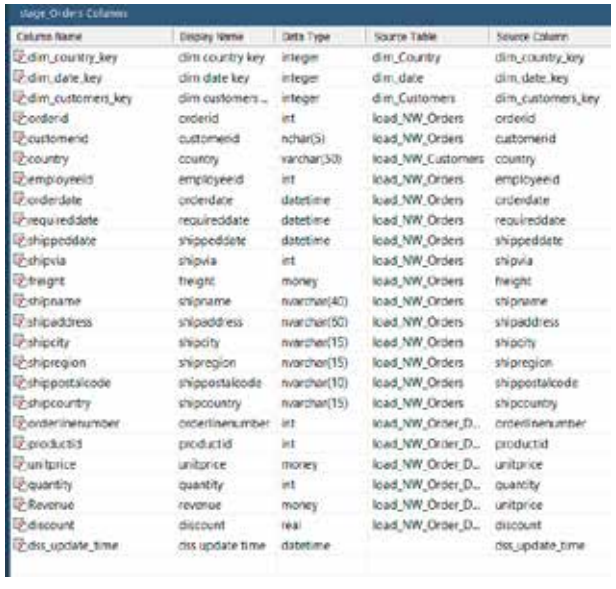


dim_customers_...	customerid	companyname	contactname	contacttitle	address	city	region	postalcode	country	phone	fax	dss_start_date	dss_end_date	dss_current...	dss_version	dss_update_...
0		Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
1	ALFKI	Alfreds Futt...	Maria Anders	Sales Repre...	Obere Str. 57	Berlin	BN	12209	Germany	030-0074321	030-0076545	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
2	ANATR	Ana Trujillo...	Ana Trujillo	Owner	Avda. de la ...	México D.F.	MDF	05021	Mexico	(5) 555-4729	(5) 555-3745	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
3	ANTON	Antonio Mo...	Antonio Mo...	Owner	Mataderos ...	México D.F.	MDF	05023	Mexico	(5) 555-3932		1900-01-01...	2999-12-31...	Y	1	2017-03-28...
4	AROUT	Around the ...	Thomas Ha...	Sales Repre...	120 Hanove...	London	LDN	WA1 1DP	UK	(171) 555-7...	(171) 555-6...	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
5	BERGS	Berglunds s...	Christina Be...	Order Admi...	Berguvsväg...	Luleå	LUL	S-958 22	Sweden	0921-12 34 ...	0921-12 34 ...	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
6	BLAUS	Blauer See ...	Hanna Moos	Sales Repre...	Forsterstr. 57	Mannheim	MAN	68306	Germany	0621-08460	0621-08924	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
7	BLONP	Blondesdcs...	Frédérique ...	Marketing ...	24, place Kl...	Strasbourg	SB	67000	France	88.60.15.31	88.60.15.32	1900-01-01...	2999-12-31...	Y	1	2017-03-28...
8	BOLID	Bólido Com...	Martín Som...	Owner	C/ Araquil, ...	Madrid	MD	28023	Spain	(91) 555 22 ...	(91) 555 91 ...	1900-01-01...	2999-12-31...	Y	1	2017-03-28...

Abb. 6: Tabellenstruktur der Dimensionstabelle mit Beispieldaten

Index Name	Table	Column	Type	Key Type	Rebuild F...	Unique	Foreign	Pri Imp	Active	Par...	Par...	Columns
dim_Customers_idx_0	dim_Customers	1	Clustered	Artificial	Never	Y	N	N	Y	0		dim_customers_key
dim_Customers_idx_A	dim_Customers	3	Composite ...	Business	Never	Y	N	N	Y	0		customerid, dss_current_flag, dss_version
dim_Customers_idx_SK	dim_Customers	13	Composite ...	Business	Never	Y	N	N	Y	0		customerid, companyname, contactname, contacttitle, address, city, region, postalcode, country, phone, fax, dss_current_flag, dss_...

Abb. 7: Übersicht der automatisch erstellten Indices für die „Customer"-Dimensionstabelle



Column Name	Display Name	Data Type	Source Table	Source Column
dim_country_key	dim country key	integer	dim_Country	dim_country_key
dim_date_key	dim date key	integer	dim_date	dim_date_key
dim_customers_key	dim customers ...	integer	dim_Customers	dim_customers_key
orderid	orderid	int	load_NW_Orders	orderid
customerid	customerid	nchar(5)	load_NW_Orders	customerid
country	country	varchar(30)	load_NW_Customers	country
employeeid	employeeid	int	load_NW_Orders	employeeid
orderdate	orderdate	datetime	load_NW_Orders	orderdate
requireddate	requireddate	datetime	load_NW_Orders	requireddate
shippeddate	shippeddate	datetime	load_NW_Orders	shippeddate
shipvia	shipvia	int	load_NW_Orders	shipvia
freight	freight	money	load_NW_Orders	freight
shipname	shipname	nvarchar(40)	load_NW_Orders	shipname
shipaddress	shipaddress	nvarchar(60)	load_NW_Orders	shipaddress
shipcity	shipcity	nvarchar(15)	load_NW_Orders	shipcity
shipregion	shipregion	nvarchar(15)	load_NW_Orders	shipregion
shippostalcode	shippostalcode	nvarchar(10)	load_NW_Orders	shippostalcode
shipcountry	shipcountry	nvarchar(15)	load_NW_Orders	shipcountry
orderlinenumber	orderlinenumber	int	load_NW_Order_D...	orderlinenumber
productid	productid	int	load_NW_Order_D...	productid
unitprice	unitprice	money	load_NW_Order_D...	unitprice
quantity	quantity	int	load_NW_Order_D...	quantity
Revenue	revenue	money	load_NW_Order_D...	unitprice
discount	discount	real	load_NW_Order_D...	discount
dss_update_time	dss update time	datetime		dss_update_time

Abb. 8: Spaltenübersicht der Stage-Tabelle

Im nächsten Schritt muss ich wieder einige Entscheidungen treffen, wie die Stage-Tabelle geladen werden soll (Abbildung 9).

Wie man bei dieser Auswahl sieht, setzt auch ein DWH-Automatisierungs-Tool das Wissen voraus, wie man ein DWH aus methodisch-konzeptioneller Sicht baut. Für mein aktuelles Fallbeispiel wähle ich Set.

Neben anderen Angaben muss ich nun spezifizieren, wie der Surrogate Key Lookup zu erfolgen hat, indem ich den Business Key aus den Faktendaten spezifiziere (siehe Abbildung 10).

Zu guter Letzt erstelle ich auf Basis der eben angelegten Stage-Tabelle noch die Faktentabelle (siehe Abbildung 11).

Vom Data Mart zum Frontend und weiter in die Produktionsumgebung

Nun kommen wir auf die Ebene Data Mart. Für dieses Fallbeispiel möchte ich dafür einen OLAP-Cube in Microsoft SQL Server Analysis Services erstellen. Dazu ziehe ich ein-

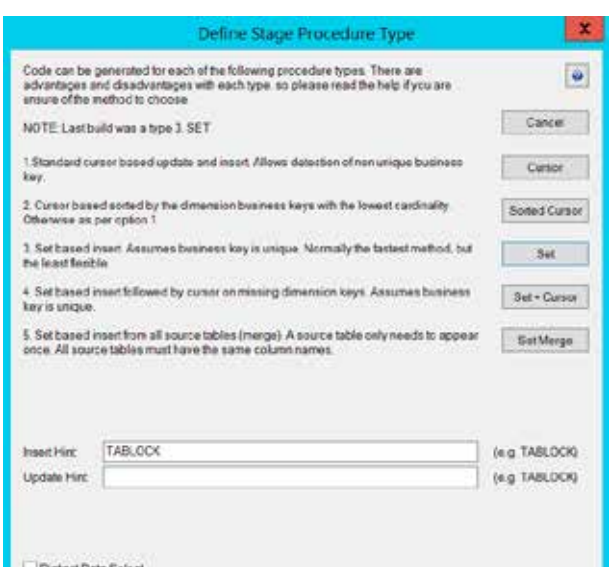


Abb. 9: Auswahl des Design-Patterns zur Beladung der Stage-Tabelle

Abb. 10: Zuweisung der Business Keys zwischen Stage-Tabelle und „Customer"-Dimensionstabelle

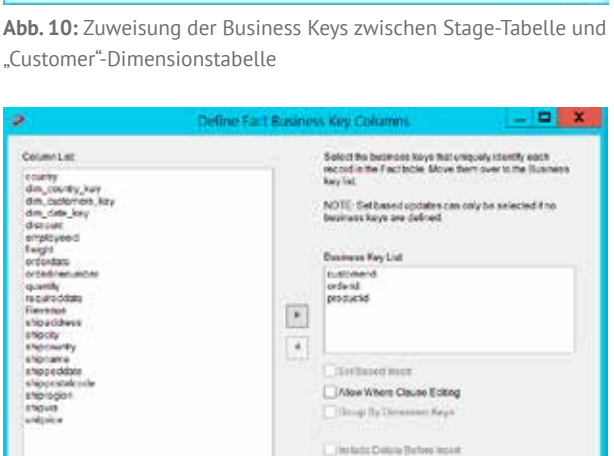


Abb. 11: Konfigurationsdialog der Faktentabelle

fach die Faktentabelle in den Bereich für OLAP-Cubes und definiere, welche Kennzahlen ich benötige (Abbildung 12).

Anschließend wird der Cube erstellt und einmal alles durchgeladen, um einen ersten einfachen Report zu erstellen (Abbildung 13).

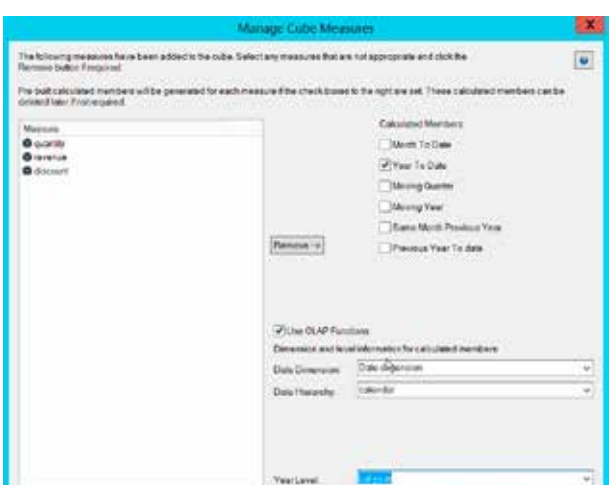


Abb. 12: Konfigurationsdialog des OLAP-Cubes

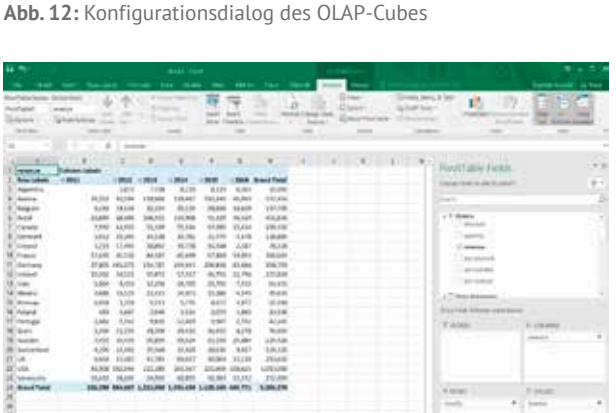


Abb. 13: Auswertung der Zahlen aus dem OLAP-Cube in Excel

Bevor wir die Lösung in die Produktionsumgebung überführen, möchte ich noch schnell dokumentieren. Mit zwei, drei Mausklicks erstelle ich eine Dokumentation auf Basis der verfügbaren Metadaten. Dazu gehören neben Textinformationen auch zahlreiche Diagramme, zum Beispiel hinsichtlich der Impact-Analyse (Abbildung 14).

Als letzter Schritt in diesem Fallbeispiel soll das eben erstellte DWH in die Produktionsumgebung überführt werden. Auch hier übernimmt die Automatisierungslösung grundsätzlich das benötigte Scripting. Als Entwickler steuere ich lediglich die dazu notwendigen Kontrollinformationen bei (Abbildung 15). Der Rest erledigt sich von alleine (Abbildung 16).

Zusammenfassung

Wenn man bei der praktischen Anwendung einer DWH-Automatisierungslösung einem Data-Driven-Ansatz folgt, sind die Quelldaten der Ausgangspunkt. Auf dieser Basis wird das DWH mittels der Spezifikation von Steuerungsinformationen entwickelt. Dabei ist das Verständnis von grundlegenden DWH-Design-Patterns (wie zum Beispiel Slowly Changing Dimensions) zentral, da die DWH-Automatisierungslösung darauf aufbaut und referenziert.

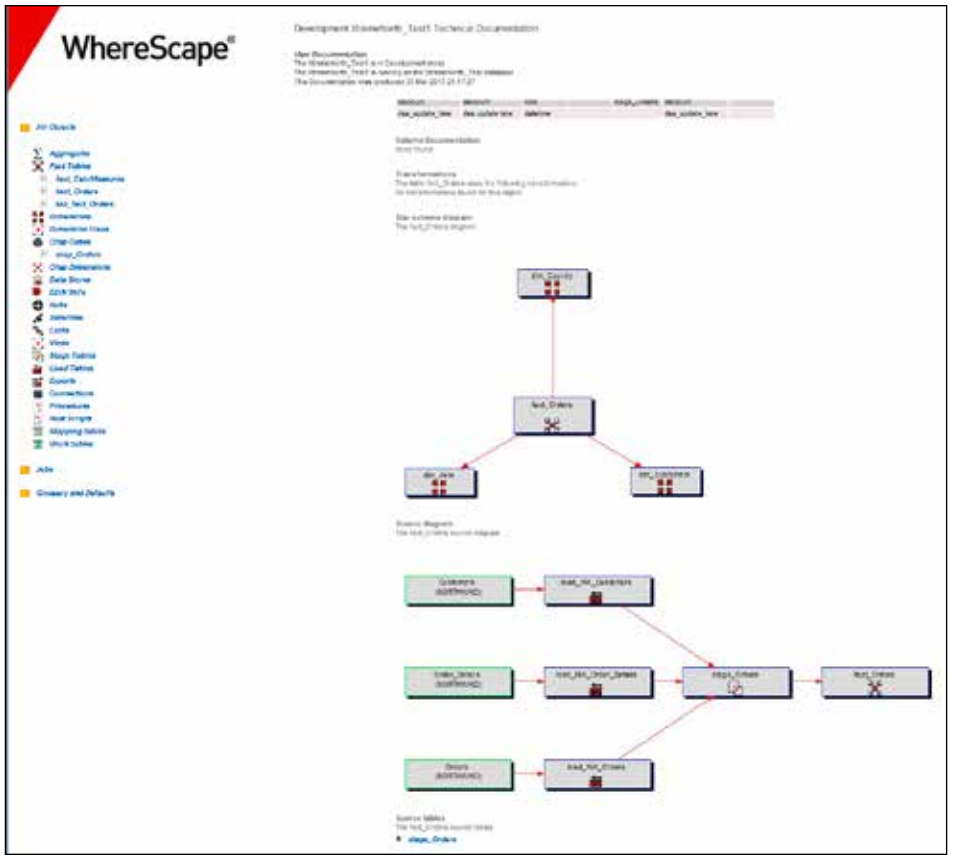


Abb. 14: Auszug aus der automatisch generierten Dokumentation

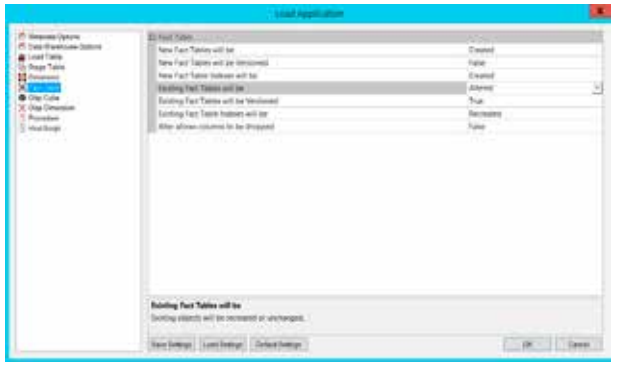


Abb. 15: Konfigurationsdialog der Load-Application

Abb. 16: Auszug aus dem Log-Dialog und Erfolgsmeldung nach dem Einspielen der Load-Application

Die Automatisierung übernimmt einen großen Teil der Fleißarbeit, sodass die bisher darin investierte Zeit auf stärker wertschöpfende Tätigkeiten verwendet werden kann. Gleichzeitig wird durch die Automatisierung die Durchlaufzeit, die nötig ist, um eine Anforderung von den Quelldaten bis zum Frontend zu entwickeln, signifikant reduziert.

[Quellen]

[Eck15] Eckerson, W.: Data Warehouse Automation Tools – Product Categories and Positioning, Wayne Eckerson, 2015, https://www.wherescape.com/media/1791/eckerson-group-dw-automation-tools-report.pdf (abgerufen am 29.3.2017)

Raphael Branger studierte Wirtschaftsinformatik und hält einen M.A. in Information Management. Heute arbeitet er – mit über 14 Jahren Praxiserfahrung – als Chief Knowledge Officer und Senior Solution Architect bei der IT-Logix AG im Business-Intelligence- und Data-Warehousing-Umfeld. Sein aktueller Fokus liegt im Bereich der BI-spezifischen Anforderungsanalyse sowie der Adaption agiler Methoden im Umfeld von BI. Neben seiner täglichen Projektarbeit ist Raphael Branger ein regelmäßiger Referent an nationalen und internationalen BI-Konferenzen. Sein aktuelles Thema auf der TDWI 2017 in München: „Übung macht den Meister: Anforderungen an ein Dashboard praktisch erheben".
E-Mail: rbranger@it-logix.ch