4 Abfragen am Beispiel des Aggregation Frameworks

Anhand der Datensätze lassen sich anschließend Abfragen zu Analysezwecken durchführen. Neben einfachen Abfragen nach einzelnen schlüssel-werten Paaren oder Bereichen, ermöglicht MongoDB auch komplexere Abfragen mittels Map-Reduce oder dem Aggregation Framework.

```
> db.posts.aggregate([
...     { $match: { post_status: "publish" } },
...     { $match: { "post_meta.country": "US" } },
...     { $group: { _id: "aggregate", total_likes: { $sum: "$post_meta.likes" } } }
... ])
{ "_id" : "aggregate", "total_likes" : 66 }
>
```

Abbildung 5: Auslesen der Likes von veröffentlichten Beiträgen aus den USA anhand eines Beispieldatensatzes

In Abbildung 5 wird eine Abfrage anhand des Aggregation Frameworks durchgeführt. Zunächst wird die Collection „posts“ ausgewählt. Mittels der „$match“ Operatoren innerhalb der Aggregation Abfrage werden Datensätze anhand bestimmter Schlüssel-Wert-Paare ermittelt. In diesem Fall sollen alle Beiträge mit dem Status „publish“ und dem Standort „US“ ausgelesen werden. Die „$group“-Anweisung gibt ein Dokument mit der ID „_id“ zurück und summiert in diesem Dokument die Werte „likes“ aus dem Embedded-Dokument „post_meta“, welches die beiden „$match“ Bedingungen erfüllt. Dieses Dokument könnte nun in Verbindung mit anderen Dokumenten für weitere Abfragen dienen. Dieses Praxisbeispiel zeigt, welche Hürden bei der Migration von Datensätzen einer relationalen Datenbank hin zu einer NoSQL Datenbanken entstehen. Ein besonderes Augenmerk muss hier auf die Struktur der Daten gelegt werden. Anhand dieses Vorgehens lassen sich kostengünstige, skalierbare Möglichkeiten für die Verwaltung von riesigen Datenmengen schaffen auf Basis von NoSQL-Datenbanken schaffen.

Literaturverzeichnis

[Roeb11] Kubacki W. Mark: Storing and Managing Big Data - NoSQL, Hadoop and More: High-impact Strategies - What You Need to Know. Tebbo, 2011.

[Meie16] Meier Andreas; Kaufmann Michael: SQL- & NoSQL-Datenbanken. Springer Vieweg, Hidelberg, 2016.

[Gaur13] Vaish Gaurav: Getting Started with Nosql. Packt Publishing, Birmingham, 2013.

[Coro16] Coronel Carlos; Morris Steven: Database Systems: Design, Implementation, & Management. Cengage Learning, Boston, 2016.

[Brad17] Bradshaw Shannon; Chodorow Kristina: Mongodb: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, 2017.

Kontakt

Prof. Dr. Thomas Barton, Marco Graf
Hochschule Worms
Erenburgerstr. 19, 67549 Worms
barton@hs-worms.de, graf@hs-worms.de

Aufwärtskompatibilität von Modellierungstechniken für Data Warehouses anhand von 3NF, Dimensional Modeling und Data Vault

Katja Kubitta, Harald Ritz

Zusammenfassung

Data Warehouses sind immer stärker gefordert auf neue Anforderungen in kürzester Zeit zu reagieren. Eine gewisse Flexibilität wird erreicht, indem Data Warehouses in mehreren Schichten aufgebaut werden. Dabei ist die zentrale Komponente das Core Data Warehouse (Core DW). Aufbauend auf diesem integrierten Datenbestand werden bspw. Data Marts zu Auswertungszwecken aufgesetzt.
Dieser Artikel untersucht die Aufwärtskompatibilität der drei bekanntesten Modellierungstechniken für Core Data Warehouses: 3NF, Dimensional Modeling und Data Vault. Mit Aufwärtskompatibilität sind dabei die Auswirkungen von Änderung im Core DW hinsichtlich nachgelagerter Data Marts gemeint. Es wird untersucht, ob sich eine Modellierungstechnik bzgl. Aufwärtskompatibilität besonders von den anderen Modellierungstechniken abhebt.
Zur Untersuchung der Aufwärtskompatibilität werden umfangreiche Szenarien von Änderungen an einem Core DW ermittelt. Diese Änderungen werden anhand der jeweiligen Modellierungstechnik durchgeführt und hinsichtlich der Auswirkungen auf bspw. nachgelagerte Data Marts bewertet. Trotz der verschiedenen Vorgehensweisen in den drei Modellierungstechniken zeigen sich hinsichtlich Aufwärtskompatibilität keine wesentlichen Unterschiede zwischen den Modellierungstechniken. Daraus folgt, dass Lösungen hinsichtlich weiterer Flexibilität eher nicht in der Aufwärtskompatibilität, sondern verstärkt in der Entkopplung von Core DW und Data Marts zu suchen sind, indem die Änderungen im Core DW vor Data Marts gekapselt werden.

1 Einleitung

Zur optimalen Unterstützung verschiedener Fachbereiche oder Prozesse eines Unternehmens wird auf spezialisierte Software zurückgegriffen. Daher liegen die Daten eines solchen Unternehmens in unterschiedlichen Speicherformaten vor. Entscheider benötigen zur Entscheidungsunterstützung allerdings konsistente und integrierte Daten übergreifend aus mehreren Softwaresystemen. Dafür wird eine zentrale Datenbasis benötigt, in der die Daten losgelöst von den unterschiedlichen operativen Systemen rein für vielfältige analytische Zwecke verwendet werden. Eine solche Datenbasis, die durch Integration und Konsolidierung von Daten aus unterschiedlichen operativen Systemen entsteht, wird Data Warehouse genannt (s. [GoRP09], S. 5).
Da Unternehmen heutzutage auf schnell wechselnde Anforderungen reagieren müssen, ist ein Data Warehouse so zu konzipieren, dass es die Flexibilität besitzt, die neuen Anforderungen schnell umzusetzen, damit das Unternehmen wettbewerbsfähig bleibt (s. [TrZi16], S. 1). Ein Konzept, diesen Anforderungen gerecht zu werden, ist der Aufbau eines Data Warehouse in mehreren Schichten. Die zentrale Komponente ist dabei ein Core DW, das den zur Auswertung benötigten Datenbestand eines Unternehmens integriert und historisiert

speichert (s. [Hahn14], S. 16). Zur besseren Auswertung von Daten werden aufbauend auf dem Core DW weitere Schichten oder Data Marts mit Teildaten aufgesetzt.
Die wechselnden Anforderungen erfordern auch Änderungen im Core DW. Diese Änderungen können sich auf den Datenfluss vom Core DW zu nachgelagerten Schichten oder Data Marts auswirken.
Für die Modellierung eines Core DW existieren unterschiedliche Modellierungstechniken (s. [ArRi12], S.23-32). Der vorliegende Artikel gibt einen Überblick über gängige Modellierungstechniken für Core Data Warehouses und analysiert die Auswirkungen von Änderungen innerhalb dieser Modellierungstechniken auf nachgelagerte Komponenten. Dabei wird darauf geachtet, ein möglichst breites Spektrum an Anwendungsfällen von Änderungen zu betrachten.
Im zweiten Kapitel dieses Artikels werden grundlegend die unterschiedlichen Modellierungstechniken vorgestellt und einige Begriffe erläutert. Das dritte Kapitel analysiert und evaluiert die Anwendungsfälle und deren Änderungen in der Modellierung im Core DW sowie die Auswirkungen dieser Änderungen auf nachgelagerte Komponenten. Im abschließenden vierten Kapitel werden die Ergebnisse zusammengefasst sowie ein Ausblick auf mögliche weiterführende Untersuchungen und Methoden zum Kapseln von Änderungen vor abhängigen Komponenten gegeben.

2 Grundbegriffe und Modellierungstechniken

2.1 Aufwärtskompatibilität

Der Begriff *Aufwärtskompatibilität* kommt ursprünglich aus dem Softwarebereich. Der Begriff beschreibt, dass Funktionen, die in einer älteren Softwareversion nutzbar sind, auch in einer neueren Softwareversion weiterhin verwendbar bleiben (s. [LaSi01])
Im Zusammenhang mit dieser Arbeit ist unter dem Begriff *Aufwärtskompatibilität* folgendes zu verstehen. Vorausgesetzt wird, dass ein Data Warehouse aus mehreren Komponenten besteht. Der zentrale Bestandteil dabei ist ein Core DW. Ausgehend davon werden nachgelagerte Komponenten, wie bspw. Data Marts, mit den Daten des Core DW versorgt. Durch neue Anforderungen oder auch Änderungen wird das Core DW fortlaufend angepasst. Die Aufwärtskompatibilität ist dabei, dass die nachgelagerten Komponenten auch bei einer neuen Version des Core DW über die gleichen Schnittstellen unverändert die Daten erhalten. Damit ist gemeint, dass die nachgelagerten Komponenten keiner Anpassung bedürfen, obwohl sich das Core DW ändert.

2.2 Änderungsszenario

Ein Änderungsszenario ist eine neue Anforderung von Endbenutzern, die aktuell mit den Daten im Core DW nicht umgesetzt werden kann. Dabei gibt es viele verschiedene Szenarien, die ggf. nur Änderungen oder Erweiterungen an der Beladung des Core DW zur Folge haben, und andere Szenarien, die ggf. eine strukturelle Änderung des Core DW erfordern.
Darunter fallen sowohl die Anforderungen, neue Informationen ins Core DW zu bringen, als auch die Anpassung und Änderung schon vorhandener Informationen, wie bspw. eine Berechnungsformel. Zusätzlich gibt es technische Anforderungen, die Änderungen am Core DW zur Folge haben. Dazu zählen bspw. Namenskonventionen für Tabellen und Spalten.

Dafür wurden mittels Experteninterviews Änderungsszenarien an einem Core DW ermittelt und kategorisiert. Es wurde darauf geachtet, ein breites Spektrum an Änderungsszenarien zu ermitteln, um eine möglichst umfassende Untersuchung zu erhalten.

2.3 Normalisierte Modellierungstechnik nach Bill Inmon

Inmon empfiehlt für das Core DW eine Modellierung, die nah an der dritten Normalform (s. [Geis14], S. 186-191) liegt, und für nachgelagerte Data Marts die dimensionale Modellierung nach Ralph Kimball (s. [InSN08], S.18ff und [Hahn14], S.13-15).

2.4 Dimensionale Modellierungstechnik nach Ralph Kimball

Kimball untergliedert seine Modellierungstechnik hauptsächlich in die zwei Bestandteile *Faktentabellen* und *Dimensionstabellen*, die im Folgenden erläutert werden.

2.4.1 Faktentabellen

In den Faktentabellen stehen die Daten, die zur Messung und Untersuchung eines Geschäftsprozesses relevant sind. So werden in einer Faktentabelle bspw. die Transaktionen von Verkäufen gespeichert, damit sind die Verkaufspositionen (Produkte mit jeweiligem Preis und Anzahl) gemeint. Faktentabellen bestehen zum einen aus Kennzahlen und zum anderen aus Fremdschlüsseln auf die Dimensionstabellen. Der Primärschlüssel der Faktentabelle besteht aus den Fremdschlüsseln auf die Dimensionstabellen oder aus einer Teilmenge von ihnen (s. [Beck06]).
Mit Kennzahlen werden die zugrunde liegenden Geschäftsprozesse quantifiziert und messbar. Daher liegen diese Kennzahlen in der Regel in numerischer Form vor. Jegliche beschreibenden Attribute, die nicht in direkter Abhängigkeit des einzelnen Faktums stehen, sind in Dimensionstabellen auszulagern (s. [KiRo13], S. 12).
Nahezu alle Faktentabellen haben eine oder mehrere Referenzen auf eine Dimension, die die Zeit beinhaltet. Die Inhalte einer Faktentabelle weisen einen Zeitbezug auf, nämlich die Information, wann das jeweilige Faktum stattgefunden hat (s. [Kimb04]).
Es ist wichtig, darauf zu achten, die Daten in der Faktentabelle möglichst auf atomarer Ebene und nicht aggregiert zu erfassen. Im Beispiel der Verkäufe bedeutet das, dass jede einzelne Verkaufsposition in der Faktentabelle gespeichert wird. Außerdem sind alle Daten in der Faktentabelle mit der gleichen Granularität zu speichern (s. [KiRo13], S. 11). Für jeden notwendigen Detailgrad sollte eine eigene Faktentabelle verwendet werden, um spätere Komplikationen zu vermeiden.
Die Faktentabellen sind die grundlegenden Tabellen für die nachfolgenden Analysen und Abfragen. Daher sind sie auch mit Abstand die größten Tabellen und enthalten meist mehrere Milliarden Datensätze. Über das gesamte Data Warehouse gesehen, machen die Faktentabellen ca. 90% des gesamten Speicherplatzes aus, während die Dimensionstabellen nur ca. 10% des Speicherplatzes benötigen (s. [KiRo13], S. 12).

2.4.2 Dimensionstabellen

Die Dimensionstabellen enthalten vor allem textliche Informationen zu den Objekten, mit denen die Fakten in Beziehung stehen. Im Beispiel des Verkaufs sind dies das Produkt und der Kunde, an den das Produkt verkauft wird. Diese beiden Objekte stehen mit Einträgen aus der Faktentabelle in Beziehung. Während die Anzahl an Attributen bei den Faktentabellen oft recht klein ist, können Dimensionstabellen eine große Menge an Attributen aufweisen, die vorwiegend Texte beinhalten (s. [KiRo13], S. 13).

Dimensionstabellen haben als Primärschlüssel eine einzige Spalte, bspw. eine Surrogate ID (SID), die in den Faktentabellen referenziert wird. Die SID dient zur eindeutigen Identifizierung eines Datensatzes in der Dimension (s. [KiRo13], S. 46).
Neben der SID wird in den Dimensionen auch der fachliche Schlüssel gespeichert. Die Inhalte der Dimensionstabellen werden nicht zu Analysezwecken, sondern zur Gruppierung der Daten aus der jeweiligen Faktentabelle verwendet. Ein Beispiel dafür ist die Kundengruppe, die in der Dimensionstabelle Kunde enthalten ist.
Wesentlich ist die Entscheidung, ob die Dimensionen denormalisiert (Star-Schema) oder normalisiert (Snowflake-Schema) modelliert werden.

2.4.3 Star- und Snowflake-Schema

Im Star-Schema hat die Faktentabelle die zentrale Position. Die Dimensionstabellen werden um die Faktentabelle herum angeordnet. Der Umriss der Modellierung erinnert dabei an einen Stern, woher das Star-Schema seinen Namen hat. Die Besonderheit dieser Modellierung liegt in der Denormalisierung der Dimensionstabellen. Zu jeder Dimension gibt es nur eine Dimensionstabelle, in der alle Informationen abgelegt werden. So werden auch Hierarchien in den Dimensionstabellen abgebildet, weshalb eine gewisse Redundanz in den Daten der jeweiligen Dimensionstabelle entsteht (s. [BaGü13], S. 243-246).
Bei der normalisierten Form von Dimensionstabellen, auch Snowflake-Schema genannt, werden redundante Daten in eigene Tabellen ausgelagert und anschließend referenziert (s. [KiRo13], S. 15).

2.5 Data Vault nach Dan Linstedt

Als wichtigen Unterschied zur normalisierten und dimensionalen Modellierungstechnik werden Schlüssel, deskriptive Informationen und Beziehungen bei Data Vault in drei getrennten Typen von Tabellen abgelegt.
Diese Modellierungstechnik konzentriert sich auf die Modellierung des Core DW. Für Data Marts empfiehlt auch Dan Linstedt die dimensionale Modellierung (s. [Hult12], S. 45).

2.5.1 Hub

Ein Hub enthält mehrere gleichartige Objekte der Wirklichkeit, die für das jeweilige Unternehmen von Bedeutung sind. Beispiele dafür sind Kunde, Konto oder Produkt. Hubs enthalten keine beschreibenden Informationen oder Fremdschlüssel.
In Hubs werden die Business Keys (fachliche Schlüssel) der jeweiligen Objekte gespeichert. Beispiele hierfür sind Kundenummern, Kontonummern oder Produktnummern. Dabei können diese Business Keys auch aus mehreren Attributen bestehen. Ist dies der Fall, müssen alle diese Attribute im Hub abgebildet werden (s. [Hult12], S. 84-87).
Weiterhin enthält ein Hub standardmäßig folgende Felder:

- SID: Besteht der Business Key aus mehreren Attributen, so wird eine SID als Primärschlüssel verwendet.
- Load Date: Ein Zeitstempel, der angibt, wann der Business Key zum ersten Mal geladen wurde.
- Record Source: Gibt das Quellsystem an, aus dem der Eintrag stammt.

2.5.2 Link

Zur Speicherung von Beziehungen zwischen zwei oder mehreren Hubs oder Links werden Links verwendet. Genau wie ein Hub enthält auch ein Link keine beschreibenden Informationen. In einen Link werden die SIDs der beteiligten Hubs bzw. Links übernommen, wodurch

die Verbindung zwischen den Hubs bzw. Links hergestellt wird. Diese Vorgehensweise, Verbindungen zu speichern, entspricht einer n:m-Beziehung. Jede Beziehung wird nach Data Vault gemäß einer n:m-Beziehung in einer Link-Tabelle gespeichert (s. [Hult12], S. 94-104).
Zusätzlich zu den Fremdschlüsseln auf die SIDs der beteiligten Hubs und Links enthält ein Link die gleichen Felder, analog zum Hub.

2.5.3 Satellit

In Satelliten werden die deskriptiven (beschreibenden) Informationen gespeichert und historisiert. Satelliten gehören immer zu genau einem Hub oder Link. Sowohl Hubs als auch Links können beliebig viele Satelliten haben.
Der Primärschlüssel eines Satelliten besteht aus der SID des zugehörigen Hubs oder Links und einem Zeitstempel bzw. Ladedatum mit der Angabe, wann die Informationen in das Data Warehouse geladen wurden (s. [Hult12], S. 114). Dabei sind Satelliten die einzigen Komponenten, die ein zeitliches Attribut als Teil ihres Primärschlüssels verwenden.
Um die Gültigkeit eines Eintrags abzubilden, wird das Ladedatum und optional ein Enddatum verwendet. Jedes Mal, wenn eine Änderung an beschreibenden Informationen erfolgt, wie bspw. eine Änderung des Kundennamens, wird eine neue Zeile mit den geänderten Daten in den Satelliten geladen (s. [Hult12], S. 115).
Wie viele Satelliten ein Hub oder Link erhält, hängt von der Entscheidung ab, ob bzw. wie die deskriptiven Daten gruppiert werden. Eine mögliche und oft verwendete Gruppierung ist die nach der Änderungsrate, mit der sich die einzelnen Informationen ändern (s. [Lins11], S. 78-84).

3 Untersuchung und Bewertung

In diesem Kapitel werden die zuvor vorgestellten Modellierungstechniken hinsichtlich ihrer Aufwärtskompatibilität untersucht und bewertet. Grundlage der Untersuchung sind zum einen die ermittelten Änderungsszenarien und zum anderen die gewählten Kriterien zur Bewertung der Änderungsauswirkungen im Core DW sowie in Bezug auf die nachgelagerten Komponenten. Damit die Modellierungstechniken vergleichbar miteinander untersucht werden können, wurde vor der Untersuchung ein einheitliches Grundbeispiel erstellt. An diesem Grundbeispiel wurden anschließend die Änderungen aus den Szenarien jeweils durchgeführt und anhand der folgenden Kriterien untersucht.

3.1 Bewertungskriterien

Die Bewertungskriterien bilden die Grundlage für das Beurteilen der Auswirkungen, die die jeweilige Änderung auf die unterschiedlichen Modellierungstechniken hat. Die Kriterien sind so gewählt, dass eine umfangreiche Untersuchung der Änderungsauswirkungen stattfindet. Bei allen Kriterien werden Aufwand und Komplexität betrachtet. Ein hoher Aufwand kann dabei sowohl aus einer hohen Anzahl an Anpassungen als auch aus einer einzelnen großen Anpassung resultieren. Unter Komplexität wird in diesem Artikel die Schwierigkeit von Änderungen verstanden. Je höher die Komplexität einer Änderung ist, desto höher ist die Fehleranfälligkeit ihrer jeweiligen Realisierung und entsprechend der nötige Testaufwand.

3.1.1 *Umfang Core-Strukturänderung*

Dieses Kriterium drückt aus, wie stark sich die Struktur der Daten im Core DW ändern muss, um die Änderung eines Szenarios abbilden zu können. Zu Strukturänderungen zählen bspw. Ergänzungen oder Änderungen von Tabellen, Attributen oder Fremdschlüsselbeziehungen. Je größer die für ein Szenario nötigen Strukturänderungen sind, desto größer ist der zu erwartende Einfluss, den das Szenario auf nachgelagerte Komponenten hat.

3.1.2 *Umfang Core-Datenänderung*

Dieses Kriterium bewertet, ob oder in welchem Umfang die im Core DW vorhandenen Daten für das Szenario geändert werden müssen. Je nach Szenario können bspw. Datenkorrekturen, Datenergänzungen oder Datentransfers nötig werden. Datenänderungen im Core DW haben meist Auswirkungen auf nachgelagerte Komponenten.

3.1.3 *Umfang Core-Beladungsänderung*

Mit diesem Kriterium wird beschrieben, wie umfangreich die ETL-Prozesse zur Beladung des Core DW angepasst werden müssen, um die Änderung des jeweiligen Szenarios abbilden zu können. Erstellungen von neuen ETL-Prozessen sowie Änderungen oder Beendigungen bestehender ETL-Prozesse können je nach Szenario erforderlich werden.
Bei manchen Szenarien hängt die Anzahl der anzupassenden ETL-Prozesse von Faktoren ab, die sich nicht aus dem Szenario selbst ergeben, so z.B. von der Anzahl der vorhandenen Fremdschlüsselbeziehungen zu betroffenen Tabellen. In solchen Fällen spiegelt dieses Kriterium den Mindestumfang an Anpassungen wider, und ein hochgestelltes Sternchen (*) hinter der Bewertung gibt an, dass der tatsächliche Umfang aufgrund äußerer Faktoren höher sein kann.

3.1.4 *Aufwärtskompatibilität*

Szenarien, die keine Auswirkungen auf nachgelagerte Komponenten haben, sind aufwärtskompatibel. Die Auswirkungen von Szenarien, die die Aufwärtskompatibilität nicht erfüllen, können unter Umständen vor nachgelagerten Komponenten verborgen werden, indem die ETL-Prozesse, die die nachgelagerten Komponenten beladen, so angepasst werden, dass sie die Auswirkungen kapseln. Der Umfang der zur Kapselung nötigen Anpassungen an den ETL-Prozessen wird durch dieses Kriterium bewertet.
Die Bewertung bezieht sich dabei stets auf genau eine nachgelagerte Komponente. Für jede weitere nachgelagerte Komponente muss die Anpassung erneut durchgeführt werden, wobei davon ausgegangen wird, dass der Umfang der Anpassungen zwischen verschiedenen nachgelagerten Komponenten jeweils ähnlich ist.
Kann eine Änderung vor einer nachgelagerten Komponente auch durch Anpassung eines ETL-Prozesses nicht gekapselt werden, so müssen die nachgelagerten Komponenten angepasst werden. Dieser Fall wird in der Bewertung gesondert berücksichtigt.

3.2 *Bewertungsskala*

Alle vier Bewertungskriterien nutzen eine gemeinsame Bewertungsskala. Dadurch sind die Bewertungen der Kriterien untereinander vergleichbar.
Insgesamt hat sich zur Bewertung eine gemeinsame Skala als zweckmäßig erwiesen, die Aufwand und Komplexität zusammenfasst. Dabei werden Aufwand und Komplexität jeweils in drei Stufen unterteilt (niedrig, mittel, hoch), wobei sich die Komplexität nur bei hohen Aufwänden in der Bewertung widerspiegelt. Auf diese Weise werden komplizierte Änderungen höher gewichtet als aufwendige aber einfache Änderungen. Dadurch werden bei komplizierten Änderungen die höhere Fehleranfälligkeit und der höhere Testaufwand berücksichtigt.

Die Skala verfügt zusätzlich über zwei Extremstufen, die die beiden Sonderfälle abbilden, dass eine Änderung einerseits unnötig oder andererseits unmöglich ist (Stufen 0 und 6).

Bewertung	Beschreibung
0	unnötig / direkt gegeben
1	niedriger Aufwand, niedrige Komplexität
2	mittlerer Aufwand, niedrige Komplexität
3	hoher Aufwand, niedrige Komplexität
4	hoher Aufwand, mittlere Komplexität
5	hoher Aufwand, hohe Komplexität
6	unmöglich

Tab. 1.: Bewertungsskala

3.3 *Änderungsszenarien*

In der folgenden Tabelle werden die untersuchten Änderungsszenarien aufgelistet. Die Szenarien sind mit einer Abkürzung versehen, um sie in der Gesamtübersicht (Tab. 3) wiederzufinden.

Abkürzung	Szenario
NA	Neues Attribut
NE	Neuer Entitytyp
VAn	Veränderung eines Attributnamens
VEn	Veränderung eines Entitytypnamens
AzE	Attribut wird zu neuem Entitytyp
Kn:m	Kardinalität 1:n wird n:m
EG	Erhöhung der Granularität
VG	Verringerung der Granularität
FSE	Fachlichen Schlüssel erweitern
FSR	Fachlichen Schlüssel reduzieren
NQE	Neue Quelle zu bestehendem Entitytyp
DA	Datentyp eines Attributs ändern
ÄH	Änderung der Historisierung
DV:AS	Data Vault: Aufsplittung von Satelliten
BA	Berechnung eines Attributs ändern
ÄF	Änderung im Filter

Tab. 2.: Änderungsszenarien

Einige dieser Änderungsszenarien sind aus ihrem Namen heraus verständlich, die übrigen Szenarien werden im Folgenden kurz erläutert.
Unter dem Szenario *Attribut wird zu neuem Entitytyp* wird verstanden, dass ein Attribut in eine eigene Tabelle ausgelagert und über einen Fremdschlüssel mit der ursprünglichen Tabelle verbunden wird.
Bei der *Erhöhung der Granularität* wird der Fall untersucht, dass der Detailgrad der Daten im Core DW abnimmt. Damit ist bspw. gemeint, dass Daten bisher auf Tagesbasis ins Core DW geladen wurden und nach der Änderung kumuliert einmal pro Monat ins Core DW geladen werden. Die *Verringerung der Granularität* untersucht das Gegenteil.
Das Szenario *Fachlichen Schlüssel erweitern* untersucht den Fall, dass der bisherige fachliche Schlüssel nicht ausreicht, um ein Objekt in einem Unternehmen eindeutig zu identifizieren. Ein Beispiel dafür ist, dass ein Kunde nur durch die Kundennummer in Verbindung mit einer Mandantennummer eindeutig identifiziert werden kann, während der fachliche Schlüs-

sel bisher nur aus der Kundennummer bestand. Das Szenario *Fachlichen Schlüssel reduzieren* untersucht das Gegenteil.
Im Szenario *Neue Quelle zu bestehendem Entitytyp* wird untersucht, was sich ändert, wenn Daten aus einer weiteren Quelle in eine bestehende Tabelle geladen werden.
Beim Szenario *Änderung der Historisierung* wird untersucht, welche Auswirkungen es hat, wenn ein Attribut in der Historisierung von SCD Typ 1 auf SCD Typ 2 (Slowly Changing Dimensions nach Kimball, s. [KiRo13], S. 53-56) geändert wird. Dieses Szenario kann für Data Vault nicht untersucht werden, da die Historisierung in der Modellierung vorgeben ist. Als Ersatz wurde das Szenario *Data Vault: Aufsplittung von Satelliten* untersucht. Dabei werden die Attribute eines Satelliten in zwei oder mehrere Satelliten aufgeteilt.

3.4 Untersuchungsergebnisse

Die folgende Tabelle zeigt die ermittelten Bewertungen der untersuchten Szenarien. Die Abkürzungen stehen für die Szenarien aus Tabelle 2.

	Inmon				Kimball				Data Vault			
	Struktur	Daten	Beladen	Aufw.-komp.	Struktur	Daten	Beladen	Aufw.-komp.	Struktur	Daten	Beladen	Aufw.-komp.
NA	1	0	1	0	1	0	1	0	1	0	1	0
NE	1	0	1	0	1	1	1	0	2*	0	2*	0
VAn	1	0	1	1	1	0	1	1	1	0	1	1
VEn	1	0	1*	1	1	0	1*	1	1	0	2*	1
AzE	1	2	1	1	1	2	1	1	2	3	2	2
Kn:m	1	1	1	6	2	2	5	6	0	0	1	6
EG	1	0	4	6	1	0	4	6	1	0	4	6
VG	2	0	2	4	1	0	1	4	3	0	3	4
FSE	1	1	1*	6	0	5	1*	6	1	0	1	6
FSR	0	5	1*	6	0	5	1*	6	2*	3*	2*	6
NQE	0	0	1	1	0	0	1	1	0	0	2*	1
DA	1	0	1	6	1	0	1	6	1	0	1	6
ÄH	0	0	1	1	0	0	1	1	-	-	-	-
DV:AS	-	-	-	-	-	-	-	-	2*	5	1*	1
BA	0	0	1	1	0	0	1	1	0	0	1	1
ÄF	0	0	1	1	0	0	1	1	0	0	1	1

Tab. 3.: Gesamtübersicht aller Bewertungen

4 Fazit und Ausblick

Dieser Artikel beschäftigt sich mit der Frage, ob eine der gängigen Modellierungstechniken für Core DWs (Inmon, Kimball oder Data Vault) sich hinsichtlich der Aufwärtskompatibilität gegenüber nachgelagerten Komponenten besonders eignet. Denn Änderungen am Core DW sind nötig, um neuen Anforderungen gerecht zu werden. Dabei sollen bestehende nachgelagerte Komponenten, die nicht durch die Änderung beeinflusst werden sollen, nach Möglichkeit weiterhin ihre Daten ohne Anpassung erhalten können. Hat eine Änderung im

Core DW keine Auswirkungen auf den Datenfluss zu nachgelagerten Komponenten, so ist diese aufwärtskompatibel.
Ist eine Änderung nicht aufwärtskompatibel, so können ihre Auswirkungen unter Umständen vor nachgelagerte Komponenten verborgen (gekapselt) werden. Der Umfang einer solchen Anpassung der ETL-Prozesse ist ein Maß für die Kapselbarkeit einer Änderung und wurde im Rahmen dieses Artikels ebenfalls untersucht und bewertet. Demgegenüber sind manche Änderungen weder aufwärtskompatibel noch kapselbar. Bei solchen Änderungen muss ggf. jede nachgelagerte Komponente separat bzgl. der Änderung angepasst werden.
Aus den Bewertungen geht hervor, dass bei den untersuchten Szenarien sowohl die Aufwärtskompatibilität als auch die Kapselbarkeit von der Modellierungstechnik des Core DW unabhängig sind. Lediglich der Kapselungsumfang variiert bei Änderungen, die nicht aufwärtskompatibel aber kapselbar sind, zwischen den verschiedenen Modellierungstechniken. Insgesamt ist damit von den untersuchten Modellierungstechniken keine als besonders geeignet bzgl. Aufwärtskompatibilität hervorzuheben. Lediglich bei Erweiterungen im Core DW haben sich alle Modellierungstechniken als vollständig aufwärtskompatibel erwiesen, weil die betreffenden Daten bisher nicht an die nachgelagerten Komponenten weitergegeben wurden.
Aus dieser Untersuchung folgt, dass die Aufwärtskompatibilität kein Kriterium bei der Wahl der Modellierungstechnik für ein Core DW ist. Denn für die Modellierung eines Core DW sind die spezifischen Eigenschaften der Modellierungstechniken, wie etwa die starke Ausrichtung auf Geschäftsprozesse bei der dimensionalen Modellierung nach Kimball, wesentlich relevantere Kriterien als die geringfügigen Unterschiede bzgl. des Kapselungsumfangs von Änderungen.
Weiterhin lässt das Ergebnis der Untersuchung die Vermutung zu, dass die Kapselbarkeit von Szenarien unabhängig von den Modellierungstechniken ist. Aus der Untersuchung geht hervor, dass bei einigen Szenarien die Kapselbarkeit daran scheitert, dass Aggregationen in nachgelagerten Komponenten nach der Änderung zu fehlerhaften Daten führen können. Daher ist eine der offenen Fragen, ob sich eine Modellierungstechnik finden lässt, bei der auch nach der Änderung solche Aggregationen in nachgelagerten Komponenten korrekte Ergebnisse liefern. Gegebenenfalls lässt sich aber auch zeigen, dass bestimmte Änderungen grundsätzlich bei keiner Modellierungstechnik gekapselt werden können.
Ein anderer Ansatz zur Reduktion des Kapselungsaufwands ist, das Core DW und die nachgelagerten Komponenten durch eine Zwischenschicht zu entkoppeln. Dabei müssen die Änderungen aus dem Core DW nur in der Zwischenschicht gekapselt werden und der Datenfluss Richtung Data Marts bleibt unverändert. Eine Idee, diese Kapselschicht aufzubauen, liegt in der Nutzung von Datenbanksichten. Ferner ergibt sich die Frage, ob weitere Möglichkeiten für den Aufbau einer solchen Kapselschicht existieren. Gegebenenfalls lassen sich Kapselungen bestimmter Änderungen automatisieren, sodass der Aufwand für die Kapselung weiter reduziert werden kann.

Literaturverzeichnis

[ArRi12] Arnold, Christoph und Ritz, Harald: Eignung unterschiedlicher Faktenmodellierungen in Data-Warehouse-Systemen, in: Barton, Thomas u.a. (Hrsg.): Herausforderungen an die Wirtschaftsinformatik: Management und IT, News & Media, Berlin 2012, S.23-40.

[Beck06] Becker, Bob: Design Tip #81 Fact Table Surrogate Key, 2006, URL: http://www.kimballgroup.com/2006/07/design-tip-81-fact-table-surrogate-key/. Abruf am 2017-05-02.

[BaGü13] Bauer, Andreas und Günzel, Holger: Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung. dpunkt.verlag, Heidelberg, 2013.

[Geis14] Geisler, Frank: Datenbanken: Grundlagen und Design. mitp-Verlag, Heidelberg, 2014.

[GoRP09] Golfarelli, Matteo; Rizzi, Stefano und Pagliarani, Claudio: Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill, New York, 2009.

[Hahn14] Hahne, Michael: Modellierung von Business-Intelligence-Systemen: Leitfaden für erfolgreiche Projekte auf Basis flexibler Data-Warehouse-Architekturen. Edition TDWI. dpunkt.verlag, Heidelberg, 2014.

[Hult12] Hultgren, Hans: Modeling the Agile Data Warehouse with Data Vault. New Hamilton, Denver and Stockholm and NYC and Sydney, 2012.

[InSN08] Inmon, William H.; Strauss, Derek und Neushloss, Genia: DW 2.0 – The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann, Amsterdam, 2008

[Kimb04] Kimball, Ralph: Design Tip #51: Latest Thinking On Time Dimension Tables, 2004, URL: http://www.kimballgroup.com/2004/02/design-tip-51-latest-thinking-on-time-dimension-tables/. Abruf am 2017-05-02

[KiRo13] Kimball, Ralph und Ross, Margy: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley, Indianapolis, 2013.

[LaSi01] Lackes, Richard und Siepermann, Markus: Definition Aufwärtskompatibilität. 2001, URL: http://wirtschaftslexikon.gabler.de/Definition/aufwaertskompatibilitaet.html. Abruf am 2017-05-02.

[Lins11] Linstedt, Dan: Super Charge your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. Saint Albans, Vermont, 2011.

[TrZi16] Trahasch, Stephan und Zimmer, Michael (Hrsg.): Agile Business Intelligence: Theorie und Praxis. dpunkt.verlag, Heidelberg, 2016.

Kontakt

Prof. Dr. Harald Ritz, Katja Kubitta (B. Sc. Wirtsch.-Inform.)
Technische Hochschule Mittelhessen (THM)
Campus Gießen, Fachbereich MNI
Wiesenstraße 14, 35390 Gießen
T +49 641 309-2431, harald.ritz@mni.thm.de

Modellgetriebene Entwicklung – Vor- und Nachteile in der Test- und Einführungsphase

Melanie Vanderpuye, Marco Richter, Michael Guckert

Zusammenfassung

Auf der AKWI Tagung 2015 stellten wir in dem Konferenzbeitrag "Modellgetriebene Entwicklung einer Eclipse RAP-Anwendung unter Verwendung des Eclipse Modeling Frameworks" das Projekt IsyPlus vor, in dem ein neues Campus-Management System für das duale Angebot StudiumPlus der Technischen Hochschule Mittelhessen entwickelt werden sollte. Die Software ist nun fertiggestellt und wurde nahezu vollständig modellgetrieben entwickelt. Neben der CRUD Funktionalität wurden insbesondere auch die GUI sowie das Berechtigungsmodell komplett aus entsprechenden Modellen generiert. Neben dem mit BIRT-Elementen umgesetzten Berichtswesen wurden einzelne Prozessabläufen gezielt konventionell implementiert. Diese Abläufe integrieren die generierten Komponenten und bilden eine Klammer um die Einzelfunktionen. In der ca. dreijährigen Entwicklungszeit hat sich der modellgetriebene Ansatz absolut bewährt. Im Laufe des nach einem evolutionären Ansatz durchgeführten Projekts wurde das Klassenmodell wiederholt überarbeitet. Kompliziertere Zusammenhänge machten ein wiederholtes Refactoring notwendig. Diese Modelländerungen konnten durch den im Projekt entwickelten Generator immer wieder mit vergleichsweise geringem Aufwand im lauffähigen Code berücksichtigt werden, so dass stets ein frühes Testen der gesamten Software möglich blieb. Der Artikel vergleicht die Funktionalität und den Aufwand der Entwicklung mit den Erfahrungen aus der Entwicklung der abgelösten Software und diskutiert Schwierigkeiten, die aus dem allgemeinen Ansatz entstanden sind (z.B. Performance) und deren Lösung durch ergänzende konventionelle Programmierung. Darüber hinaus werden generelle Schlüsse für ein Vorgehen in einem derartigen Projekt gezogen.

1 Entscheidung für eine Neuentwicklung – die Vorphase des Projekts

StudiumPlus ist das duale und berufsbegleitende Studienkonzept der Technischen Hochschule Mittehessen (THM). (Vgl. [StudP17]) Die speziellen Anforderungen des dualen Studiums an ein Campus Management System – insbesondere die Abbildung der Zusammenarbeit mit den Partnerunternehmen – werden in der angebotenen Standardsoftware nur unzureichend oder gar nicht erfüllt. Daher fiel im Jahr 2003 die Entscheidung für die hausinterne Entwicklung von IsyPlus.
IsyPlus ist ein Informationssystem zur Verwaltung von personenbezogene Hochschuldaten zum dualen Studium. Zusätzlich unterstützt die Anwendung die administrativ tätigen Mitarbeiter bei der Durchführung verschiedener Arbeitsprozesse, die im Rahmen des Studienbetriebs erforderlich sind. Da die Software nicht mehr den Anforderungen gerecht wurde und Prozesse zunehmend nicht mehr vollständig unterstützt wurden, musste das System ersetzt werden. Aus diesem Grund fiel 2014 der Startschuss für eine Neuentwicklung des Systems (Vgl. [RGVM15], S. 215ff.) mit der Entscheidung dabei ganz neue Wege zu gehen.