Stephan Trahasch • Michael Zimmer (Hrsg.)

Agile Business Intelligence

Theorie und Praxis

tdwi EUROPE

dpunkt.verlag

Inhaltsverzeichnis

4 Modellierung agiler BI-Systeme

Michael Hahne

Die heutige Geschäftswelt ist zunehmend durch eine starke Dynamik sowie sich schneller ändernde Anforderungen und Rahmenbedingungen gekennzeichnet. Das Business muss schneller auf die neuen Marktgegebenheiten reagieren und sich auf die gestiegene Volatilität einstellen. Zu Recht wird daher mehr Agilität gefordert, denn gerade für die dispositiven Systeme ergeben sich daraus Anforderungen an deren Flexibilität, um neue Anwendungen und Anpassungen bestehender analytischer Applikationen zeitgerecht zur Verfügung zu haben. Die geforderte Dynamik tangiert dabei insbesondere die Business-Intelligence-Architektur und -Modellierung, die im Kontext agiler Entwicklung zu gestalten und zu bewerten sind, sowie Aspekte der Automatisierung von Test und Entwicklung mit dem Ziel der schnelleren und qualitätsstabilen iterativen Softwareauslieferung. Das Kapitel beschreibt die im Kontext der Agilität relevanten Facetten der BI-Architektur, behandelt die gerade für das Core Data Warehouse wichtigen Ansätze der Gestaltung und geht auf die Best Practices dazu ein. Gegenstand der Darstellung sind u. a. Domänenkonzepte, 3NF- und Data-Vault-Modellierung sowie Aspekte der Historisierung und der Anwendung von Geschäftsregeln (Business Rules).

4.1 Business-Intelligence-Architektur

Heutige Data-Warehouse-Architekturen sind in mehrfacher Hinsicht mit neuen Anforderungen konfrontiert, die eine klare Architektur sowie adäquate Methoden des Entwurfs fordern. Neben den klassischen Erwartungen an die Kostenstrukturen für den Aufbau und den Betrieb von BI-Systemen sind zunehmend gerade die geschäftlichen Anforderungen, sogenannte *Business Requirements*, die aufgrund der stetig zunehmenden Dynamik des Geschäftslebens einer starken Volatilität unterworfen sind, wesentliche Treiber für neue BI-Projekte (vgl. [Zimmer et al. 2012]; [Trahasch et al. 2014]). Gefordert sind Architekturen und Konzepte, die ein schnelles Reagieren auf neue

Herausforderungen ermöglichen und somit die Time-to-Market von BI-Applikationen drastisch senken. Mehrschichtige Data-Warehouse-Architekturen, die als logische Klammer auch agile Methoden unterstützen und fördern, sind eine konzeptionelle Antwort hierauf.[1]

Bekannte Architekturvarianten unterscheiden sich deutlich hinsichtlich der Anzahl der Komponenten, der gesamten Komplexität, des Aufwands für Entwicklung und Betrieb sowie der Performance und Skalierbarkeit. Aber auch die Fähigkeit, effizient und agil mit neuen Anforderungen umzugehen und den Wandel zu unterstützen, ist ein wesentliches Erfolgskriterium für verschiedene Ansätze. Die BI-Architektur darf die Bemühungen um Agilität nicht ausbremsen bzw. behindern. Zugleich muss aber auch ein effizienter Betrieb gewährleistet sein. Diese beiden Facetten von BI-Lösungen stehen zunächst einmal in einem Zielkonflikt. Somit sind also Prozesse, Modellierungsmethoden und Architekturen gefordert, die die Anforderungen nach dynamischer Entwicklung einerseits und effizientem Betrieb andererseits in Einklang bringen und damit eine Balance zwischen *Build* und *Run* herstellen können (siehe [Göhl & Hahne 2011]).

Im Kontext von Business Intelligence stellt sich die Frage, wie sich Projekte durch die Einführung agiler Methoden ändern. Eine Adaption agiler Ansätze für die Entwicklung von BI-Lösungen betrifft die Aspekte der Architektur, der Organisation und der Methodik. Gerade die Anwendung agiler Methoden muss durch eine Organisationsform flankiert sein, die diese effektiv unterstützt.

4.1.1 Schichtenmodell der BI-Architektur

In einer Hub-and-Spoke-Architektur[2] dient das Core Data Warehouse (CDWH)[3] als Hub und erfüllt die Aufgaben der Integration, Qualitätssicherung und Datenverteilung an die Data Marts als Spokes, die einen hohen Grad an Anwendungsorientierung und vordefinierten betriebswirtschaftlichen Anreicherungen und Aggregationen aufweisen [Kemper et al. 2010]. Vorteile dieser Architektur liegen in der redundanzfreien Logik und in der zentralen Datenintegration und Aufbereitung.

Diese Architektur erfordert in der Konsequenz unterschiedliche Modellierungsansätze, um differenzierte Anforderungen auf den einzelnen Ebenen berücksichtigen und realisieren zu können. Eine zentrale Rolle für die Gewährleistung der notwendigen Flexibilität, um auf neue geschäftliche Anforderungen schnell reagieren zu können, spielt dabei die Ausgestaltung des

zentralen unternehmensweiten und mehrschichtigen Data Warehouse, eines *Enterprise Data Warehouse* (EDW), das Mechanismen zur Historisierung implementiert und die vollständige Historie abbildet. In Verbindung mit neuen Konzepten wie Data Lakes (vgl. Kap. 8 oder 14) können mehrschichtige EDW-Architekturen so erweitert werden, dass neue Anforderungen in sogenannten *Data Marts on Demand* schnell erfüllt werden können.

In einer BI-Architektur können die in Abbildung 4–1 dargestellten Schichten differenziert werden. Im Acquisition Layer erfolgt die Aufnahme der Rohdaten aus den Quellsystemen. Ziel der Transformations- und Harmonisierungsprozesse ist die Ablage im Integration Layer, in dem die aufbereiteten Daten in ihrer vollständigen Historie vorliegen. Die Aggregation hinsichtlich der betriebswirtschaftlichen Anforderungen erfolgt auf dem Weg in den Reporting Layer [Hahne 2014].

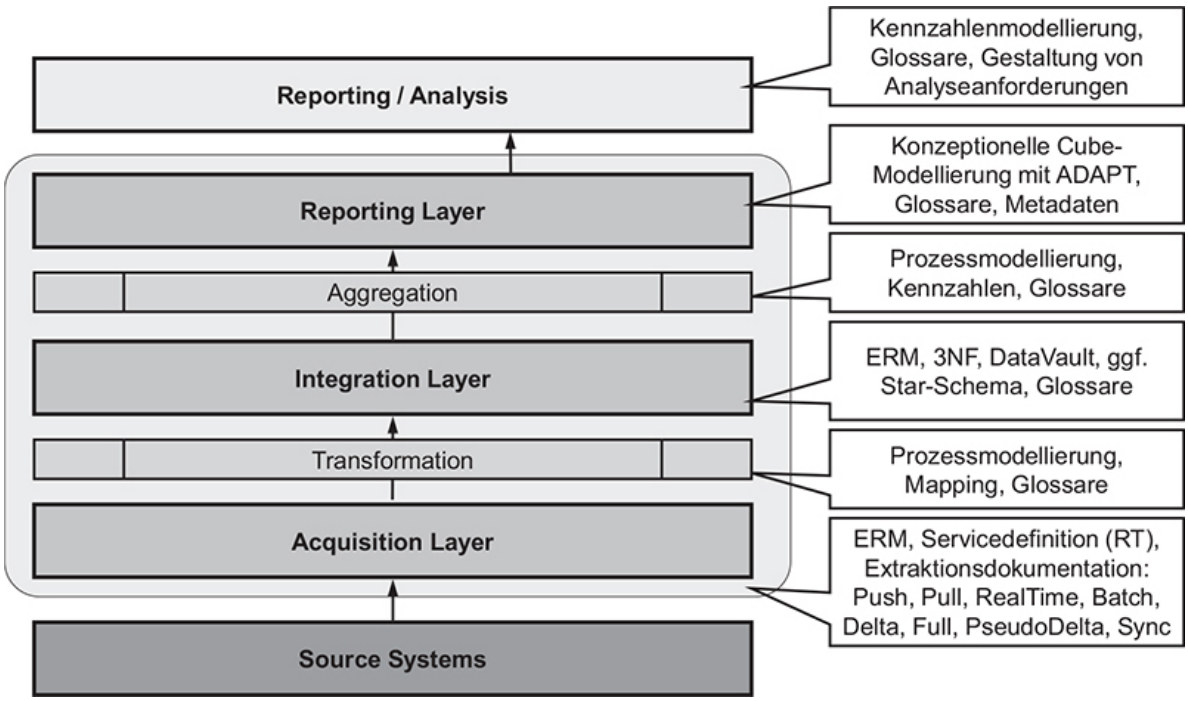


Abb. 4–1 *Gestaltung und Dokumentation in BI-Architekturen*

Zusätzlich können die Daten auch in vertikaler Richtung als Domänen gruppiert werden. Domänenkonzepte haben sich in aktuellen BI-Architekturen etabliert, denn durch diese ist es möglich, Prozesse der Datenintegration zu entzerren und insbesondere unterschiedliche Service Level [Hahne 2015] und Ladezyklizitäten zu implementieren. Ein wesentlicher Vorteil des Aufbaus mehrschichtiger Architekturen und eines EDW liegt in der dadurch entstandenen Flexibilität, mit der auf neue geschäftliche Anforderungen an BI-Anwendungen reagiert werden kann. Auf Basis der im EDW vorgehaltenen Daten können im Reporting Layer

neue Data Marts flexibel aufgebaut werden, um neue Anforderungen umzusetzen. Sind andere Data Marts mit der Zeit obsolet, können diese auch problemlos gelöscht werden. Die komplette Historie der Daten findet sich bereits im EDW, sodass keinerlei Datenverluste damit verbunden sind.

Systeme, die nach dem Grundprinzip des mehrschichtigen Aufbaus gestaltet sind, haben zum einen den Vorteil, dass sie bei geeigneter Ausgestaltung den Aspekt der Flexibilität voll berücksichtigen können und damit das, was als »anticipating the unknown« beschrieben wird, leichter ermöglichen. Neuen geschäftlichen Anforderungen kann dadurch schnell und flexibel begegnet werden. Zum anderen kommt aber noch ein weiterer Vorteil hinzu, der erst bei Betrachtung des Betriebs eines Data Warehouse und der damit zusammenhängenden Veränderungsprozesse deutlich wird. Die Komplexität im Betrieb eines Data Warehouse steigt überproportional mit der Anzahl genutzter Applikationen und den abzubildenden Datensträngen. Eine saubere Architektur reduziert den durch neue Applikationen oder Datenbereiche hinzukommenden Aufwand nicht nur bei der Erstellung neuer Applikationen oder Wartung bestehender Anwendungen, sondern auch bei deren Betrieb.

Die Verlagerung von Businesslogik möglichst nah zum Endbenutzer hin, eingebettet in eine adäquate Architektur, hat sich in vielen Projekten mittlerweile bewährt. Gerade wegen der großen Volatilität der geschäftlichen Anforderungen ist es sinnvoll, diese Regeln erst sehr spät im Prozess der Datenveredelung anzusiedeln, damit die durch die Volatilität implizierten Änderungen nicht bereits das Core Data Warehouse betreffen.

Typischerweise sind jedoch die Abhängigkeiten zwischen einzelnen Regeln der Geschäftslogik nicht immer so eindeutig einzelnen Domänen zuordenbar. Ableitungen und Businesslogik finden im Data Warehouse ihren Eingang in Modelle, die in Abbildung 4–2 mit Derivation-Area-(DA-)Modell beschrieben sind.

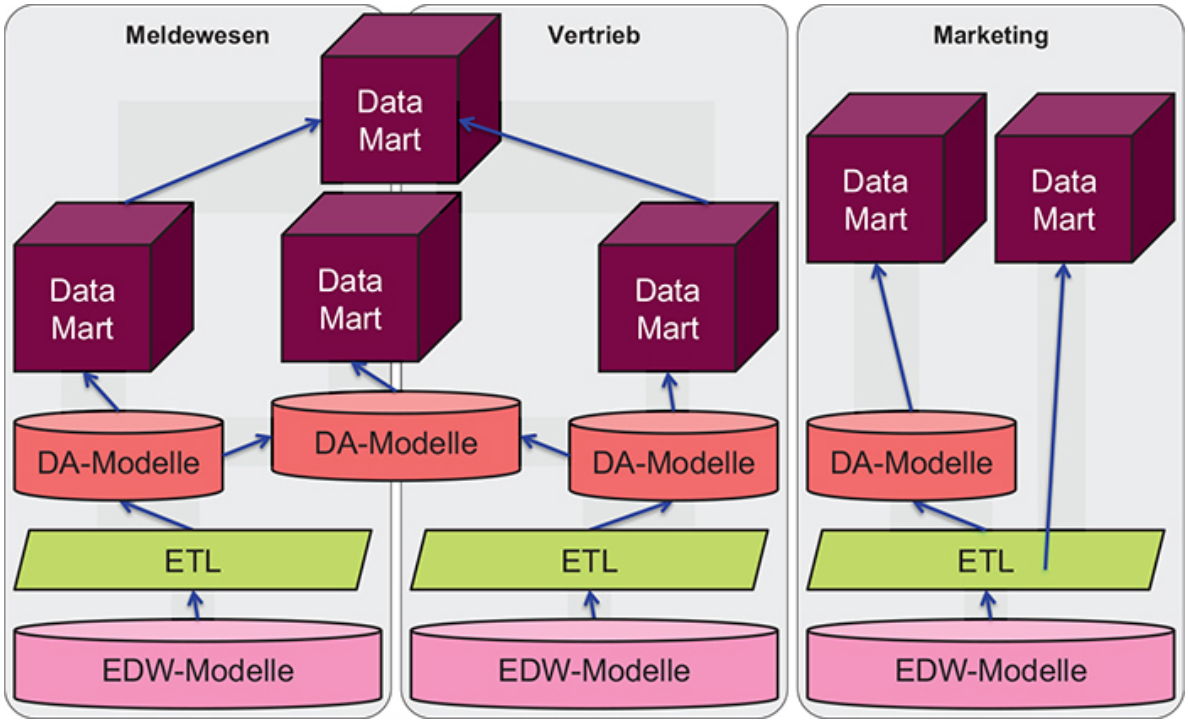


Abb. 4–2 *Geschäftsregeln und Ableitungen in BI-Architekturen*

Darüber hinaus ist es auch möglich, dass Ableitungsmodelle zwar innerhalb einer Domäne berechnet werden, diese jedoch auch in anderen Domänen Verwendung finden. Dies kann entweder direkt erfolgen oder, wie in den Domänen Meldewesen und Vertrieb in Abbildung 4–2, auch durch die Zusammenführung über kaskadierende Ableitungsmodelle [Hahne 2015].

Die aufgeführten Modelle zur Implementierung der Geschäftsregeln müssen dabei nicht notwendigerweise persistent ausgeprägt sein. Aktuelle Architektur- und Modellierungsansätze wie die Data-Vault-Methode sehen hier explizit eine Virtualisierung vor (vgl. dazu [Linstedt 2011], [Hultgren 2012]). Nicht allein durch diese, aber doch wesentlich von diesen gefördert, ermöglichen die Ansätze der Virtualisierung auch eine bessere Unterstützung von Self-Service-BI. Und gerade in diesen Architekturansätzen ist dabei die Nutzung eines Domänenkonzepts eher förderlich als hinderlich.

4.1.2 Modellierung des Core Data Warehouse

Um die steigende Komplexität von BI-Systemen und BI-Anwendungen noch beherrschen zu können, wird seitens des Betriebs solcher Lösungen eine starke Standardisierung forciert (vgl. [Zimmer et al. 2012], [Zimmer 2015]). Zugleich wird eine stärkere Flexibilität zur Berücksichtigung volatiler Geschäftsanforderungen notwendig. Dieser Zielkonflikt kann als ein Abwägen

der Aspekte *Build* und *Run* beschrieben werden. Die Antwort auf diese Herausforderung ist eine geeignete BI-Architektur. Eine zentrale Komponente ist hierin das Core Data Warehouse, oft auch als *Single Point of Trust* oder *Single Point of Truth (SPOT)* bezeichnet. Die Wahl eines Modellierungsparadigmas und die Vorgehensweise zur Gestaltung des Core Data Warehouse beeinflusst im Wesentlichen die Eigenschaften eines Data Warehouse.

Die möglichen Ansätze unterschiedlicher Herangehensweisen an die Gestaltung des Core Data Warehouse folgen im Regelfall entweder dem Star-Schema-Ansatz oder basieren auf einer Form der Modellierung in einer Normalform, wobei die klassische dritte Normalform durchaus häufig anzutreffen ist. Damit bieten sich mehrere Optionen zur Gestaltung eines Core Data Warehouse, die im spezifischen Einzelfall abzuwägen sind. Die Modellierungsoptionen für ein Core Data Warehouse und die damit verbundenen Architekturvarianten können anhand der folgenden Kriterien beurteilt werden:

- Deckt das Core Data Warehouse bereits die Aufgaben einer Staging Area mit ab?
- Sind die wesentlichen Aufgaben eines Core Data Warehouse vollständig abgedeckt?
- Sind mit dem Core Data Warehouse auch die Anforderungen des Propagation Layer und des Reporting Layer bereits berücksichtigt?
- Ist die Methodik für einen effizienten Umgang mit Change-Prozessen geeignet?[4]
- Fördert die Methode den Ansatz des Test Driven Design, also einer Entwicklungsmethodik, die zunächst das erwartete Ergebnis definiert und durch Tests deren Erreichung im Entwicklungsprozess misst?
- Lassen sich die Modelle gut für die Automatisierung von Tests benutzen?
- Inwiefern sind die Modelle geeignet, durch Automatisierungsmechanismen aus Artefakten generiert zu werden?

4.2 Star-Schema-Modellierung im Core Data Warehouse

Im Verständnis der Kimball-Architektur eines Data Warehouse ist die dimensionale Modellierung die Grundlage für den Aufbau eines Core Data Warehouse, das auch Dimensional Data Warehouse oder Data-Mart-Busarchitektur genannt wird.[5]

4.2.1 Granulare Star-Schemata im Core Data Warehouse

Für eine zentrale Architekturkomponente, die als *Single Point of Truth* fungiert, ist es wichtig, keine unnötigen Aggregationsschritte vorwegzunehmen und dadurch die weitere Verwendung der harmonisierten Daten einzuschränken. Somit ist auch ein dimensional aufgebautes Core Data Warehouse granular bzw. belegorientiert aufzubauen. Dies ist stets eine Frage des Abwägens, Best Practice ist jedoch, allgemein im Core Data Warehouse die granularste Ebene abzubilden.

Übertragen auf ein Star-Schema bedeutet dies, dass die Modelle atomar sein sollten (siehe [Kimball et al. 2008]) und damit zumindest eine Belegdimension sowie ggf. eine weitere Dimension für die Positionen der Belege haben müssen. Diese können nach den bereits ausgeführten Regeln der Gestaltung als degenerierte Dimension ausgeprägt sein. Dies hat sich in der Praxis bewährt (siehe [Kimball & Ross 2002]).[6]

Die sich im Rahmen einer normalisierten Modellierung ergebenden abhängigen Attribute eines Belegs sind im Star-Schema jedoch nicht als Attribut einer solchen Dimension aufgebaut, sondern finden vielmehr Eingang in die Definition von weiteren Dimensionen.

Die Verknüpfung einer Dimension zur Faktentabelle stellt immer eine *As-posted-* Sicht der granularen Ebene der Dimension dar. In der Konsequenz muss bei der Bewirtschaftung der Faktentabelle für jeden Satz eine Dimensionsausprägung für alle Dimensionen zugeordnet werden [Hahne 2010].

Ändert sich im Zeitablauf in dem Staging-Modell ein Satz, z. B. indem der Warenlieferant geändert wird, muss der Satz im Star-Schema korrigiert werden. Es gibt also Situationen, in denen Änderungen der Daten im Staging-Bereich auch zu Änderungen im Star-Schema führen, die nicht alleine durch Hinzufügen neuer Sätze abzubilden sind.

Um die granular strukturierten Star-Schema-Modelle im Core Data Warehouse weniger anfällig gegenüber Datenänderungen auszugestalten, ist es möglich, Aspekte der Historisierung von Fakten in Form einer Ladezeitdimension zu

berücksichtigen. Dies erhöht jedoch die Komplexität der Bewirtschaftung und der Abfrage solcher Modelle.

4.2.2 Bewertung dimensionaler Modelle im Core Data Warehouse

In einem Star-Schema sind die Daten bereits umfassenden Transformationen und verschiedenen Schritten der Datenaufbereitung unterzogen worden, sodass die Anforderungen an einen Staging-Bereich nicht mit abgedeckt werden können. Vielmehr ist es notwendig, dass der Staging-Bereich mit einer persistenten Extraktionshistorie ausgestattet ist, damit die erneute Extraktion aus den Quellsystemen vermeidbar ist.

Die Aufgaben eines Core-Data-Warehouse-Modells mit den Funktionen der Sammlung, Harmonisierung und Distribution im Sinne eines SPOT sind durch derartige Modelle erfüllbar, insofern ist der Ansatz der Star-Schema-Modellierung für ein Core Data Warehouse für dessen Kernaufgaben als geeignet anzusehen.

Ein wichtiger Aspekt in der Beurteilung der Eignung eines Modellierungsparadigmas für ein Core Data Warehouse ist der Umgang mit Änderungen, insbesondere aufgrund geänderter geschäftlicher Anforderungen oder Quellstrukturen. Die Änderungen fachlicher Anforderungen sind oftmals durch eine geänderte Businesslogik abzudecken. Im Star-Schema bedeutet dies bereits eine Modelländerung, da die Businesslogik Bestandteil der Transformationsschritte zum dimensionalen Modell ist. Wegen der bereits umfassend implementierten Businesslogik ist auch eine automatisierte Generierung der Modelle als schwierig zu bewerten. Tests können zwar automatisiert durchgeführt werden, ändern sich jedoch mit jeder umgesetzten Änderung, und eine automatisierte Erstellung von Testfällen ist dabei auch als sehr schwierig anzusehen.

Aber auch die Änderung von Quellstrukturen führt im Regelfall zu umfassenden Änderungen des Star-Schema-Modells, da z. B. neue Attribute hinzukommen oder auch neue Tabellen im Staging-Modell entstehen. Auch für die Prozesse zur Datenintegration zwischen dem Staging-Modell und dem Core-Data-Warehouse-Modell hat dies weitreichende Änderungen zur Konsequenz. Daher impliziert eine Änderung der Quellmodelle auch immer weitreichende Änderungen im Core Data Warehouse und dessen Bewirtschaftung. Es entsteht somit ein großer Aufwand für Entwicklung und Test, da alle bereits produktiv

gesetzten ETL-Prozesse potenziell betroffen sind und daher recht viele Dinge getestet werden müssen.

Da in einem Star-Schema ein Großteil der Businesslogik bereits berücksichtigt ist, ergibt sich auf der anderen Seite ein geringer Anpassungsaufwand für die Prozesse der Datenintegration in Richtung Reporting Layer. Insbesondere die Data Marts sind in allen Architekturvarianten im Regelfall dimensional modelliert und oftmals in Form des Star-Schemas implementiert. Für die Gestaltung der dimensionalen Modelle gibt es neben den Methoden der Modellierung wie z. B. ADAPT auch Vorgehensweisen, die auf agilen Ansätzen fußen und etwa in Form von *Story Telling* oder *Visual Thinking* kreative Wege der Abstimmung mit dem Nutzer einschlagen [Corr 2013].

Die Aspekte der unitemporalen Historisierung sind durch die Ansätze der Zeitstempelung und allgemein durch die Verfahren der Implementierung von Slowly Changing Dimensions (SCD) abgedeckt [Hahne 2012]. Bitemporale Historisierung ist nicht Kernbestandteil des Star-Schema-Ansatzes und bedarf besonderer Modellierungsvarianten.[7]

4.3 3NF-Modelle im Core Data Warehouse

Bei der Betrachtung von Aspekten der Normalisierung hat sich gezeigt, dass die Normalisierung im Star-Schema und die daraus resultierenden sogenannten Snowflake-Schemata selten sinnvoll sind. Dies ist jedoch komplett anders im Fall der Gestaltung eines Core Data Warehouse.

4.3.1 Core-Data-Warehouse-Modellierung in 3NF

Die Modellierung des Core Data Warehouse nah an einer dritten Normalform entsprechend der Lehre zur Datenmodellierung impliziert eine Ausrichtung auf die Kerninformationsobjekte und führt damit zu einem fachlichen Modell mit den wesentlichen Objekten und deren Beziehungen zueinander. Es fördert somit das Verständnis für die Zusammenhänge zwischen den Geschäftsobjekten.

Für das Beispielmodell mit Einkaufsbelegen und deren Positionen, noch vor den Änderungen, ergibt sich im ersten Ansatz das Modell aus Abbildung 4–3 mit seinen zwei Kernentitäten Einkaufsbeleg und Einkaufsbelegposition für die Kopf- und Positionsdaten.

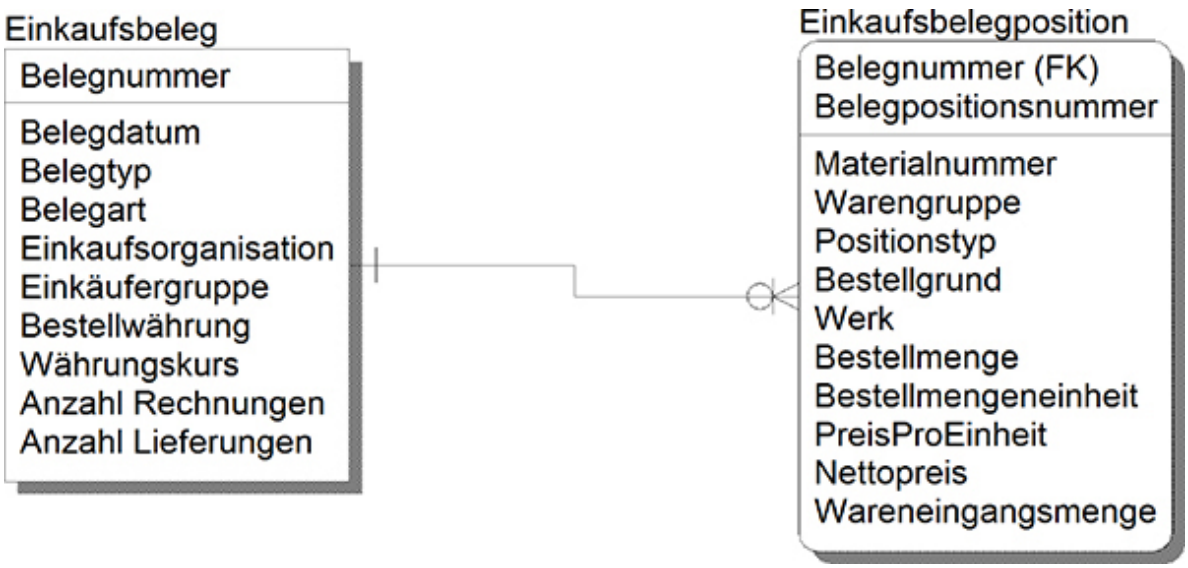


Abb. 4–3 *3NF-Beziehung zwischen Beleg und Position*

Die Attribute sind nicht weiter aus diesen beiden Entitäten herausgebrochen. Eine weitere daran anknüpfende Entität in dem Modell könnte dann z. B. der Artikel sein.

4.3.2 Historisierungsaspekte von 3NF-Modellen

Auf Basis eines 3NF-Modells können die meisten Anforderungen an ein Core Data Warehouse sehr gut abgebildet werden. Jedoch führt die Berücksichtigung von Aspekten der Historisierung zu Modellen, die sehr komplex und schlecht zu lesen sind, da dies insbesondere Many-to-many-Beziehungen an Stellen impliziert, wo fachlich lediglich eine klassische 1:n-Beziehung existiert.

Es hat sich eine Modellierungsvariante durchgesetzt, die diese negativen Aspekte ausgleicht. Dabei ist ein wesentlicher Aspekt in der Trennung der fachlichen Beziehung von den Aspekten der Historisierung zu sehen. Abbildung 4–4 zeigt diesen unter dem Stichwort *Ankermodellierung* bekannten Ansatz für das Beispiel der Einkaufsbelege.[8]

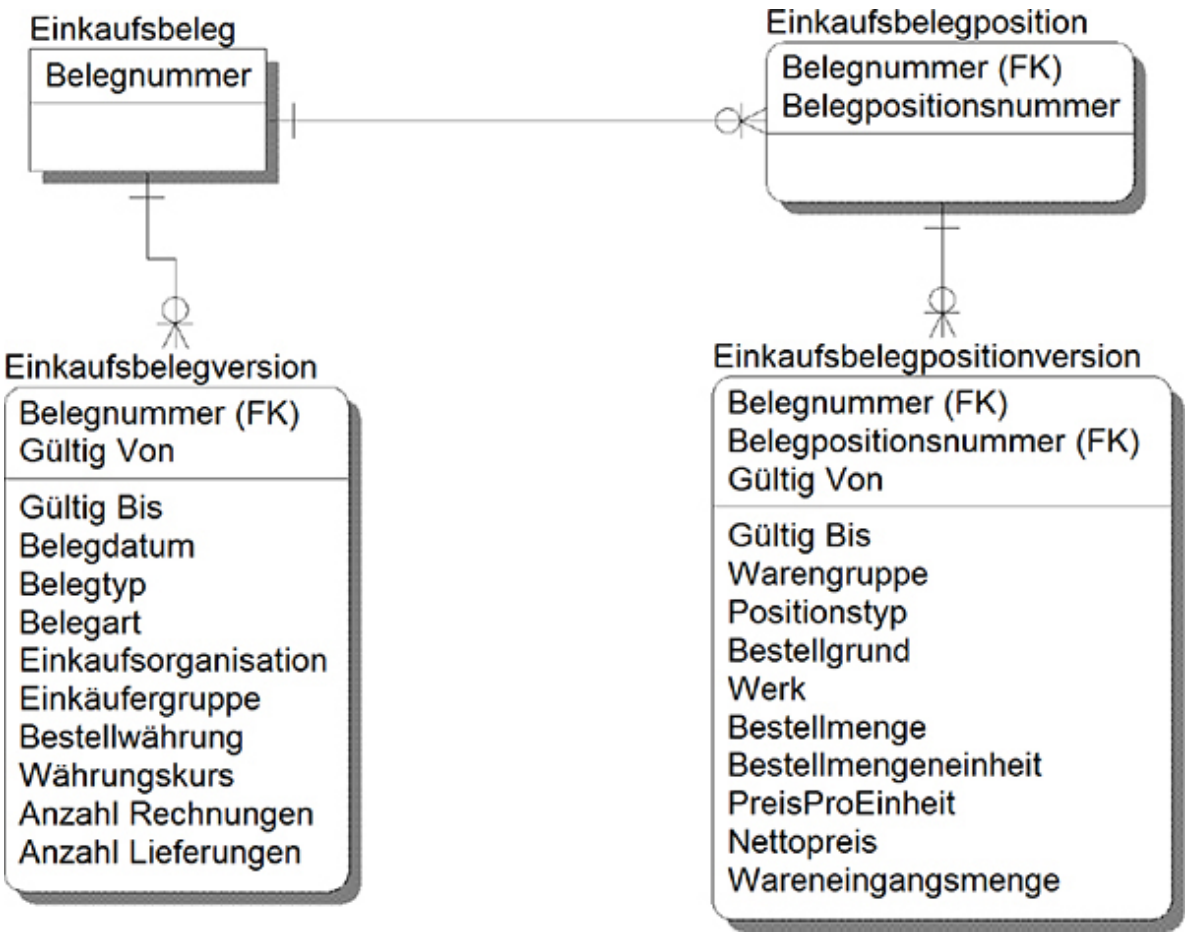


Abb. 4–4 *Abbildung von Versionen zur Historisierung im 3NF-Modell*

Die zugrunde liegende fachliche Beziehung zwischen Beleg und Belegposition ist vom Typ der 1:n-Kardinalität direkt ersichtlich und besteht lediglich aus den beiden Entitäten ohne weitere Attribute. Diese Kernentitäten heißen auch *Anker*.

An die Anker angehängt sind die Entitäten für die Versionen, also die einzelnen zeitlich abgegrenzten Attributausprägungen. Für die Versionen des Einkaufsbelegs ergibt sich dann als Schlüsselattribut neben der Belegnummer selbst in der Versions-entität auch noch ein Zeitstempel, in diesem Fall das *Gültig Von*-Datum. Somit sind verschiedene Ausprägungen der Attribute eines Einkaufsbelegs über die zeitliche Gültigkeit abgegrenzt. Für die Positionen des Einkaufsbelegs ergeben sich in der Version die Schlüsselattribute *Belegnummer, Belegpositionsnummer* und *Gültig Von*.

Zwischen dem Anker und seinen Versionen existiert immer eine 1:-n-Beziehung. Diese hätte bei einer Auflösung hin zur tragenden Entität, im Beispiel Einkaufsbeleg und Einkaufsbelegposition, die Konsequenz einer sich daraus ergebenden Many-to-many-Beziehung zwischen den fachlichen Entitäten. Dies ist somit eine der wichtigen Eigenschaften des Ansatzes der Ankermodellierung,

denn hierdurch bleibt die fachliche Beziehung mit ihrer Kardinalität erhalten und verliert nicht aufgrund der Historisierung ihre Aussagekraft.

An die Ankertabellen können ohne Probleme weitere Tabellen neben der eigentlichen Versionstabelle angehängt werden. Insbesondere ist es in diesen Modellen möglich, Attribute in verschiedene Versionstabellen zu gruppieren und auf diese Art z. B. originäre Attribute, die unverändert von der Quelle übernommen wurden, sowie abgeleitete Attribute, die als Bestandteil der ETL-Prozesse zur Bewirtschaftung des Core Data Warehouse berechnet bzw. bestimmt werden, voneinander zu trennen.

In einigen Branchen hat sich mittlerweile die bitemporale Historisierung durchgesetzt, der zufolge die Zeiten der fachlichen Gültigkeit von der technischen Gültigkeit zu differenzieren sind. Dies findet sich insbesondere in den Bereichen, in denen das operative System eine fachliche Gültigkeit pflegt, die dort auch rückwirkend oder in die Zukunft hin verändert werden kann wie etwa bei Verträgen eines Versicherungsunternehmens.

Die Core-Data-Warehouse-Modelle erhalten dann neben der zeitlichen Zuordnung der fachlichen Gültigkeit noch die technische Gültigkeit aus der Perspektive der Kenntniserlangung im Data Warehouse.

Es hat sich dabei bewährt, im Fall der Anforderung an eine bitemporale Historisierung mit einem Standard-Pattern der Historisierung alle betroffenen Entitäten im Core Data Warehouse gleich zu behandeln. Die Abfragen sind durch die notwendigen temporalen Joins komplexer, daher ist die Reporting-Ebene oftmals nur in Form einer festen zeitlichen technischen Perspektive abgebildet.[9]

Für die Reporting-Ebene mit den Data Marts spielt die Verknüpfung verschiedener bitemporaler, unitemporaler oder gar nicht historisierter Modellkomponenten eine große Rolle. Im Sinne der Data-Marts-on-Demand sind diese unterschiedlichen Perspektiven schnell und einfach aus einem sauber modellierten Core Data Warehouse heraus generierbar und dies ist damit ein wesentlicher Grund für die ermöglichte Agilität zum Benutzer hin. Kurze Iterationen und schnelle Auslieferung neuer Inhalte im Sinne der agilen Entwicklung werden hierdurch aktiv unterstützt und gefördert. Neben die technische Komplexität der Verknüpfung und Bildung konsistenter granularer Historien tritt dann auch noch die Herausforderung, den Benutzer nicht mit den Optionen zu überfordern und das Risiko potenziell falscher Abfragen zu reduzieren.

4.3.3 Domänenkonzepte im 3NF-Modell

Nicht alle Applikationen eines Data Warehouse haben die gleichen Anforderungen hinsichtlich Flexibilität, Verfügbarkeit, Latenz und Performance. Auch die Anforderungen an die Datenqualität können sehr unterschiedlich ausfallen. Von daher ist es nicht immer angemessen, alle Applikationen nach den gleichen Regeln zu behandeln und auch alle durch die gleichen Stufen und Komponenten des Data Warehouse in der gleichen Form durchlaufen zu lassen, da dies für einige Bereiche zu strenge Anforderungen und für andere ggf. zu geringe Anforderungen berücksichtigt. Hier bietet sich daher die Aufteilung in Domänen für die einzelnen Gruppen der Service Level an. Durch die Gruppierung der Anforderungen an Flexibilität, Verfügbarkeit, Latenz und Performance erfolgt eine gleichförmige Behandlung von Daten mit gleichen Anforderungen.

Die Aufteilung in Domänen impliziert nicht notwendigerweise eine vollständige Disjunktheit. Vielmehr ist es durchaus möglich, dass es zu Überlappungen kommt. Diese können beispielsweise gemeinsam genutzte Modellelemente im EDW bzw. Core Data Warehouse sein.

Durch den Einsatz eines Domänenkonzepts zur Berücksichtigung verschiedener Service Level ergibt sich somit eine Reduktion der Abhängigkeiten und insbesondere die Möglichkeit, explizit für jede Applikation oder sogar bis auf Ebene einzelner Objekte sauber zwischen den Anforderungen zu differenzieren. Dies betrifft dabei Datenqualitätskriterien, Häufigkeit der Bewirtschaftung bis hin zu Realtime- sowie Performance-Anforderungen.

Auch für die Ableitungen, also die Anwendung von Geschäftsregeln in der Ableitungsdatenbank bzw. im Propagation Layer, findet das Domänenkonzept Anwendung.[10] Und wie bereits dargelegt gibt es dabei auch Ableitungen (Rechenkerne, Gruppen von Geschäftsregeln), die mehrere Domänen überspannen. Angewendet auf das 3NF-Modell in einem Core Data Warehouse ergeben sich dann Zuordnungen der einzelnen Entitäten zu Domänen und Ableitungsgruppen wie in Abbildung 4–5 schematisch visualisiert.

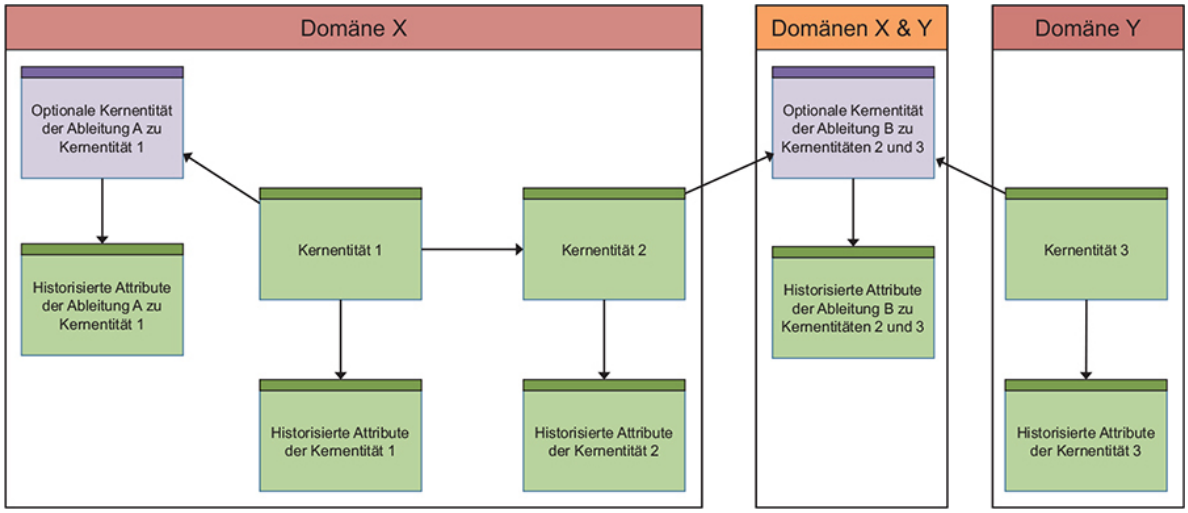


Abb. 4–5 *Ableitungen im Domänenkonzept des 3NF-Modells*

Die Anwendung von Domänen zur Gruppierung der Entitäten nach Service Level, Ladezyklizitäten oder anderen Kriterien fördert auch eine inkrementelle Vorgehensweise und ist damit gut geeignet für agile Methoden. Eine Aufsplittung der Entitäten innerhalb einer Domäne ist dann ein weiteres Mittel zur Erlangung einer noch größeren Unabhängigkeit der Entitäten untereinander und damit einer noch besseren Unterstützung im Change-Prozess, also im Fall der iterativen Änderung von Modellen. Für die Unterstützung agiler Methodik in der Entwicklung von BI-Anwendungen ist damit das Domänenkonzept eine ganz wichtige Komponente.

4.3.4 Bewertung der 3NF-Modellierung im Core Data Warehouse

Aufgrund der granularen Struktur kann ein Core Data Warehouse in der 3NF-Form oftmals bereits die Anforderungen an einen Staging-Bereich abdecken. Dazu müssen die unverändert übernommenen Attribute aus den Quellen Bestandteil des Core-Data-Warehouse-Modells sein. In diesen Fällen ist der Staging-Bereich temporärer Art und es besteht keine Notwendigkeit, diesen dauerhaft aufzubewahren.

Da in dem 3NF-Modell die granularen Daten gesammelt und integriert sind und für die weitere Distribution bereitstehen, sind damit die wesentlichen Kernaufgaben eines Core Data Warehouse als zentrale Komponente in einer Hub-and-Spoke-Architektur erfüllt.

Außerdem repräsentiert ein 3NF-Modell im Core Data Warehouse die fachlichen Beziehungen und da es sich nicht an den einzelnen Zyklen der

Weiterentwicklung orientiert, haben geänderte Anforderungen im Regelfall, sofern die Art der Harmonisierung oder die extrahierten Daten davon betroffen sind, auch eine Auswirkung auf das Datenmodell. Dadurch entsteht ein Aufwand für Anpassung und Test im Core Data Warehouse. Die Art der Änderung, die hier bereits erledigt ist, braucht nicht mehr auf dem Weg zu den Data Marts erfolgen, insofern handelt es sich zu einem Teil nur um eine Verlagerung des Aufwands, der mit einem Change verbunden ist.

Die Aspekte der Historisierung sind in dem Paradigma der 3NF-Modellierung des Core Data Warehouse ebenso berücksichtigt wie die Anforderungen an deren Reproduzierbarkeit. Dabei hat sich die Trennung der Kernentität von den Versions-entitäten mit den Attributen bewährt.

Aus der Klassifikation der Aufgaben der Datenintegration ergibt sich, dass nur die Aufgaben der Filterung und Bereinigung im ETL-Prozess zwischen Staging-Bereich und Core Data Warehouse liegen. Die Aggregation und Anreicherung finden im Regelfall erst auf dem Weg zum Reporting Layer bzw. zu den Data Marts statt. Der Einsatz einer Ableitungsdatenbank zur Kapselung mehrfach verwendeter Transformationen ist dabei eine sinnvolle Zwischenschicht.

Ein auf Basis der 3NF-Modellierung gestaltetes Core Data Warehouse hat tendenziell einen höheren Grad an Agilität im Umgang mit Change-Prozessen im Vergleich zur Star-Schema-Modellierung. Auch die damit einhergehenden Aufgaben für Entwicklung und Test weisen in diesem Vergleich eine geringere Komplexität auf. Wegen der vorhandenen Abhängigkeiten und der spezifisch ausgeprägten Struktur zwischen einzelnen Tabellen im 3NF-Modell ist die automatisierte Ausführung von Tests zwar möglich, diese ändern sich jedoch potenziell mit jeder umgesetzten Änderung und eine automatisierte Erstellung von Testfällen ist daher als eher schwierig zu bewerten.

4.4 Data-Vault-Ansatz

Die Gestaltung eines Core Data Warehouse mit der Star-Schema-Methodik und der strikten Modellierung konform zur dritten Normalform kann auch Nachteile mit sich bringen.

Die Grenzen der Star-Schema-Modellierung im Core Data Warehouse sind dabei insbesondere dadurch gegeben, dass die Granularität und Dimensionierung festgelegt sind. Aber insbesondere der Prozess des Change Management und der damit verbundene Aufwand für Entwicklung und Test ist

ein gewichtiger Grund, gerade bei kleinen Iterationsschritten, wie sie in agilen Entwicklungsmethoden zu finden sind, nach Alternativen zu suchen.

Auf dieser Grundlage der Argumentation entstand im Jahr 2000 die Data-Vault-Methode zur Modellierung eines Core Data Warehouse, die von Dan Linstedt in [Linstedt 2011] publiziert wurde.[11]

4.4.1 Tabellentypen im Data-Vault-Modell

Data Vault ist eine Methode zur Data-Warehouse-Modellierung und entgegen den bekannten Ansätzen speichert ein Data-Vault-Modell Schlüssel, deskriptive Attribute und Beziehungsinformationen konsequent in getrennten Tabellen. Ein Data-Vault-Modell hat daher drei Hauptkomponenten:

- **Hub**: Tabelle mit natürlichen Schlüsseln wie etwa Rechnungsnummer, Mitarbeiternummer, Kundennummer, SKU
- **Satellit:** Tabelle mit beschreibenden Informationen zu einem Hub oder Link, etwa Kundenname, Produktbezeichnung, Belegart, Positionsart etc.
- **Link**: Tabelle zur Abbildung der Beziehung zwischen Hubs und Links (Beziehungen zwischen Links sind erlaubt, vgl. dazu [Linstedt 2011])

In einem Hub sind keine Beziehungsinformationen und auch keine deskriptiven Attribute (Details) abgebildet. Ein Hub besteht aus dem künstlichen Schlüssel (HUBID), dem unternehmensweit eindeutigen Business Key, dem Ladezeitstempel und der Quelle.

In der Tabelle repräsentiert das Ladedatum bzw. der Zeitstempel (hier ein Tagesdatum) den Zeitpunkt, wann dieser Business Key (die Kundennummer) in das Data Warehouse erstmals geladen wurde. Die Source beschreibt, aus welchem System der Business Key stammt.

Der Hub definiert somit die Liste der eindeutigen Business Key und deren Zuordnung (Mapping) zu künstlichen Schlüsseln (sogenannten *Surrogate Keys*, die daher auch oft mit SID für *Surrogate Identifier* bezeichnet sind). Daher darf ein Hub auch nicht erneut aufgebaut werden und einmal eingefügte Sätze bleiben in dieser Tabelle und werden nicht gelöscht. Ausnahmen können allerdings dann gegeben sein, wenn beispielsweise in ein B2B-Kundenobjekt B2C-Kundendaten versehentlich geladen wurden. Dies ist eine erlaubte Korrektur, wobei aber abhängige Daten in den anderen Tabellen des Modells ebenfalls zu korrigieren sind.

Für die HUBID kann neben einer Sequenznummer auch eine auf andere Art generierte eindeutige ID genutzt werden. Denkbar sind dazu *Unique IDs* (UUID oder GUID), wie sie u. a. in Java eingesetzt werden. Besonders interessant sind dabei Verfahren, die für den gleichen Business Key reproduzierbar den gleichen künstlichen Schlüssel generieren. Zu diesen Verfahren gehören die verschiedenen Methoden zur Hash-Key-Bildung.

Aus der Perspektive der Datenintegrationsprozesse ist die Bewirtschaftung eines Hubs ein reines Insert-only-Verfahren und daher sehr leicht und performant implementierbar.

Die Satellit-Tabellen umfassen alle beschreibenden Attribute aller Quellsysteme und bilden die vollständige Historie im Data Warehouse ab. Eine Satellit-Entität besteht aus dem Fremdschlüssel zum Hub, dem Ladezeitstempel, den Attributen und der Datenquelle.

Zu einem Hub kann es mehrere Satelliten geben. Die Satelliten werden z. B. nach der Art der Daten, der Änderungshäufigkeit und der Quelle aufgeteilt. Die Prozesse der Bewirtschaftung eines Satelliten sind auch einfach und leicht zu automatisieren, da lediglich neue Datensätze zu schreiben sind.

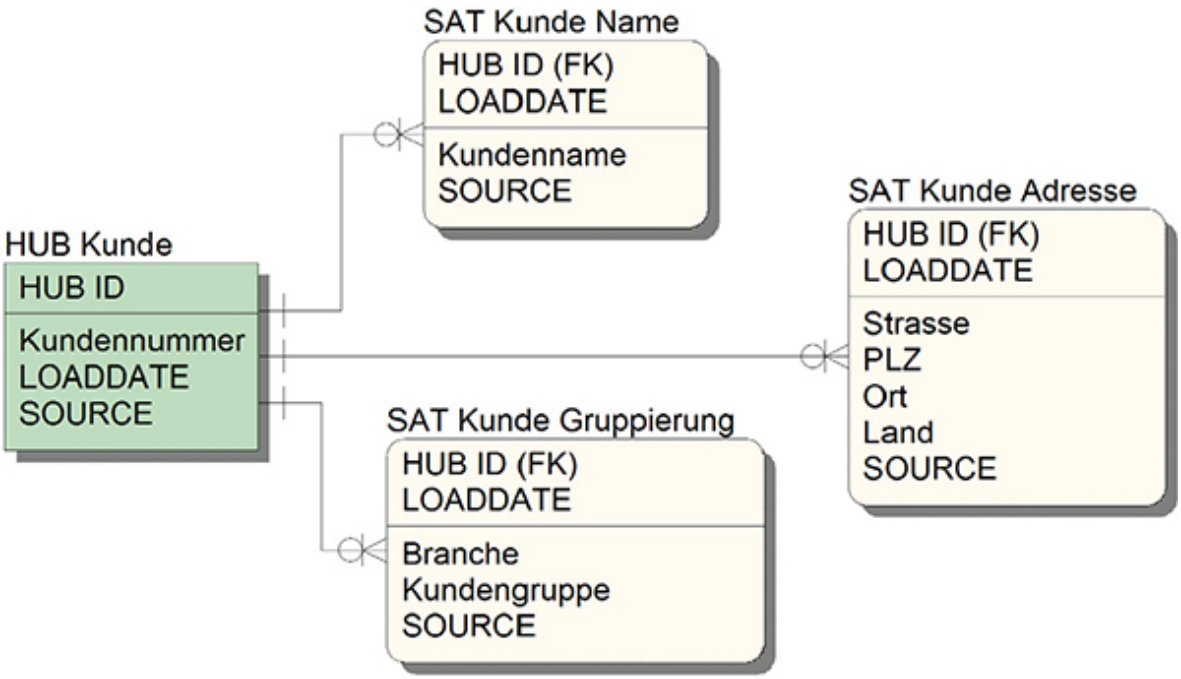


Abb. 4–6 *Hub mit mehreren Satellit-Tabellen*

Während es in einem Hub nur eine Zeile für jede Ausprägung eines Businessschlüssels gibt, ist dies in einem Satelliten anders. Mit jedem Ladelauf ist zu prüfen, ob sich an den Attributen im Satelliten etwas geändert hat und

wenn dies der Fall ist, erfolgt die Einfügung eines neuen Satzes in dem Satelliten mit dem Zeitstempel des Ladens.

Offenbar ist die Art der Aufteilung der Attribute auf verschiedene Satelliten ein Freiheitsgrad der Modellierung, der aber die Aspekte der Bewirtschaftung genauso adressiert wie Aspekte der Datenredundanz. Dem Ansatz des Anchor-Modeling [Rönnebeck et al. 2010] folgend müsste sogar jedes Attribut in einer eigenen Tabelle gespeichert werden. In der Praxis ist es ein Abwägen von Eigenschaften der Änderungshäufigkeit, dem Quellsystem und fachlichen Kombinationen von Attributen.

Wenn die Attribute auf mehrere Satelliten verteilt sind, können diese in der Konsequenz auch unterschiedlichen Änderungszyklen unterworfen sein. In Abfragen mit mehreren Satelliten an einem Hub sind die Zeitstempel pro Hub einzeln zu ermitteln [Hahne 2014]. Somit ist dann pro Hub eine *Sub-Select-Klausel* notwendig, um die korrekten Zeitstempel zu ermitteln. Dies kann durch den Einsatz sogenannter *Point-in-Time-(PIT)*-Tabellen seitens der Abfrage vereinfacht werden [Hahne 2014]. Diese Konzepte zur Verbesserung der Abfrage-Performance an ein Data-Vault-Modell erlangen zunehmende Bedeutung, da entgegen der ursprünglichen Sichtweise, dass ein Data Vault nicht direkt abgefragt werden sollte, heute vermehrt Konzepte der Virtualisierung in BI-Architekturen zum Einsatz kommen. Die Implementierung virtueller Data Marts etwa in Form von Views führt aber dazu, dass die Performance der Abfrage an ein Data-Vault-Modell viel kritischer zu betrachten ist. Dies funktioniert jedoch für granulare Data Marts mit einem Fokus auf die erste Analyse von nicht sonderlich aufbereiteten Datenbeständen sehr gut. Die Virtualisierung stößt bei der Implementierung umfangreicher Geschäftsregeln schnell an die Grenzen. Gerade im Kontext agiler Vorgehensweisen ermöglichen diese virtuellen Data Marts auf den Data-Vault-Modellen im Core Data Warehouse eine einfache und schnelle Möglichkeit, die Daten dem Fachbereich zur Analyse zu übergeben. Die Festlegung und Validierung von Geschäftsregeln können dann viel agiler im Sinne eines kontinuierlichen Refactorings erfolgen.

Der Einsatz von Point-in-Time-Tabellen hat einerseits den Vorteil einer etwas besseren Performance bei Abfragen, da nur noch ein Sub-Select benötigt wird. Andererseits muss diese Tabelle aber auch im Rahmen der Bewirtschaftung eines Data-Vault-Modells mit gefüllt werden. Daraus können sich potenziell Inkonsistenzen ergeben, zumindest aber gibt es eine weitere Abhängigkeit für eine konsistente Abfrage an das Modell.

Zu einem Data-Vault-Modell gehören neben den bisher dargestellten Hub- und Satellit-Tabellen auch noch die Link-Tabellen, über die Beziehungen zwischen HubTabellen abgebildet werden. Grundprinzip bei Data Vaults ist dabei, dass ausschließlich Many-to-many-Beziehungen berücksichtigt werden[12], und zwar auch dann, wenn es sich um eine fachliche 1:n-Beziehung handelt. Dies hat den Vorteil, dass eine Änderung der Kardinalität einer Beziehung nicht zu kaskadierenden Modelländerungen führt.

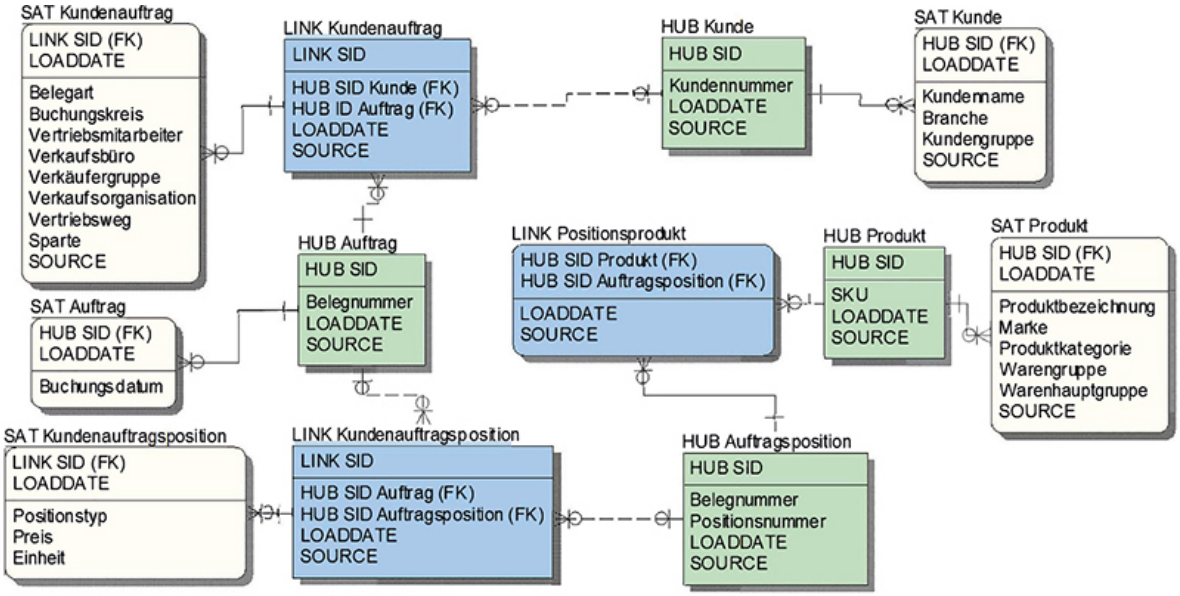


Abb. 4–7 *Data-Vault-Modell für Core-Data-Warehouse-Auftragsdaten*

Ein Link ist eine Tabelle zur Abbildung einer Beziehung vom Grad größer gleich zwei. Dabei stehen alle beteiligten Tabellen zum Link in einer 1:n-Beziehung. Die Beziehung zwischen den beteiligten Hub-Tabellen ist somit eine Many-to-many-Beziehung.

Der Link Kundenauftrag in Abbildung 4–7 hat als Primärschlüssel die Fremdschlüssel zu den in der Beziehung beteiligten Hubs Kunde und Auftrag. Mit dem ersten Auftreten der Beziehung in den Quelldaten wird diese Kombination von Schlüsseln in die Link-Tabelle geschrieben. Dies geschieht zusammen mit dem Zeitstempel des Auftretens und der Quelle, aus der diese Information stammt. Eine explizite Historisierung von dieser Beziehung auf Basis einer aus dem Quellsystem übermittelten Gültigkeit der Beziehung ist dabei nicht vorgesehen.

Der Primärschlüssel der Link-Tabelle ist demzufolge ein zusammengesetzter Schlüssel, der aus den Fremdschlüsseln zu den Hub-Tabellen besteht. Sollen auch Attribute in Form von Satelliten an diese Beziehung gebunden werden, ist dazu ein künstlicher Primärschlüssel notwendig. Hierzu dient eine Link-ID. Aus

Gründen der Flexibilität hat es sich bewährt, in Link-Tabellen grundsätzlich künstliche Schlüssel einzusetzen. Bei der Nutzung von Hashfunktionen für die Berechnung der künstlichen Schlüssel in den Hub-Tabellen ist es möglich, für den Link einen Hash-Wert über alle beteiligten Business Key zu benutzen.[13]

Da die Belegnummer ein fachlicher wichtiger Schlüssel ist, ergibt sich daraus bereits ein erster Hub. Ebenso formt auch die Position eines Auftrags einen Businessschlüssel, der Schlüssel ist aber der zusammengesetzte Schlüssel aus der Belegnummer und der Position. Generell führen Transaktionen aus den operativen Prozessen immer zu Hubs (siehe [Hultgren 2012]).

Weitere Business Key in dem Modell sind der Kunde (Auftraggeber) und das Produkt (SKU). Darüber hinaus sind noch weitere Felder potenziell als fachliche Schlüssel einzustufen, wobei sich dies ggf. erst später im Ablauf eines Projekts ergibt.

Die beiden anderen Link-Tabellen in dem Beispiel verdeutlichen die Variante, wenn zusätzlich Satellit-Tabellen an den Link angebunden sein sollen, es also Attribute zu der Beziehung gibt. Im konkreten Beispiel ist dies die Beziehung zwischen dem Kunden und dem Auftrag, die Link-Tabelle dazu hat den Namen Link Kundenauftrag. An dieser Tabelle ist zu erkennen, dass nunmehr der Primärschlüssel ein künstlicher Schlüssel ist (LINK SID). Die Fremdschlüssel zu den Hubs der Beziehung wandern in den abhängigen Teil, stellen aber gemeinsam einen alternativen Schlüssel der Tabelle dar. Dieser Schlüssel LINK SID dient nun in dem angehängten Satelliten zur Aufnahme der Attribute des Auftragskopfs. Streng genommen ist es genauso möglich, in diesem Fall diesen Satelliten auch an den Hub-Auftrag zu hängen, da die Attribute wie z. B. Verkaufsbüro oder Belegart bereits eindeutig für den Auftragskopf sind. Es gibt aber viele Beziehungen, deren Attribute nur im Zusammenspiel der beteiligten Hub-Tabellen eindeutig sind und daher zwingend diese Vorgehensweise benötigen.

4.4.2 Zeitstempel und Formen der Historisierung

Die bisher genutzten Beispiele basieren alle auf einer einfachen Angabe des Ladezeitpunkts als Zeitstempel in den Tabellen. Dies hat den Vorteil, dass die Operationen zur Bewirtschaftung eines Data-Vault-Modells reine Insert-Prozesse sind. Es werden also jeweils nur neue Sätze geschrieben, jedoch keine existierenden Sätze gelöscht oder aktualisiert. Ausnahmen davon stellen lediglich kontrollierte Korrekturen dar, die z. B. aufgrund von korrupten

Datenlieferungen entstanden und inhaltlich falsch sind. Der Nachteil dieser Vorgehensweise ist bei den Skripten für Abfragen an ein Data-Vault-Modell deutlich geworden, es sind dabei komplexe Sub-Select-Statements notwendig, um die passenden Zeitstempel in den Tabellen zu filtern.

Mit dem Zeitstempelverfahren können Abfragen die korrekte Version anhand der Attribute *Gültig Von* und *Gültig Bis* bereits mit einem Between-Statement ermitteln. Dieses Verfahren kann auf ein Data-Vault-Modell angewendet werden und führt zu Strukturen in den Satellit-Tabellen, die Gültigkeitsfelder als *LOADDATE Begin* und *LOADDATE End* nutzen. Dies verdeutlicht auch, dass hierfür im Wesentlichen die Zeit der Verarbeitung im Data Warehouse, also die technische Gültigkeit, herangezogen wird.

Die Einführung des Feldes LOADDATE End hat einen signifikanten Einfluss auf die Bewirtschaftungsprozesse, da dies eine Abkehr von dem reinen Insert-only-Verfahren auf den Satellit-Tabellen zur Folge hat. Abfragen sind demgegenüber einfacher zu formulieren, da eine zeitliche Between-Abfrage hier ein komplexes Sub-Select unnötig macht. Eine verbesserte Abfrage-Performance steht also im Zielkonflikt mit schnellen und einfachen ETL-Prozessen. Eine Möglichkeit, um dies zu vermeiden, stellt die Auslagerung der Information der zeitlichen Terminierung in eine eigene Tabelle dar, dies ist dann aber kein Standard-Data-Vault-Modell mehr. Diese Tabelle muss dann bei Abfragen auch immer mit berücksichtigt werden.

Eine bitemporale Historisierung kennt das Data-Vault-Modell in dieser Form nicht direkt, kann jedoch diesbezüglich angepasst werden. Eine Möglichkeit zur Berücksichtigung fachlicher und technischer Gültigkeiten ist in Abbildung 4–8 dargestellt.

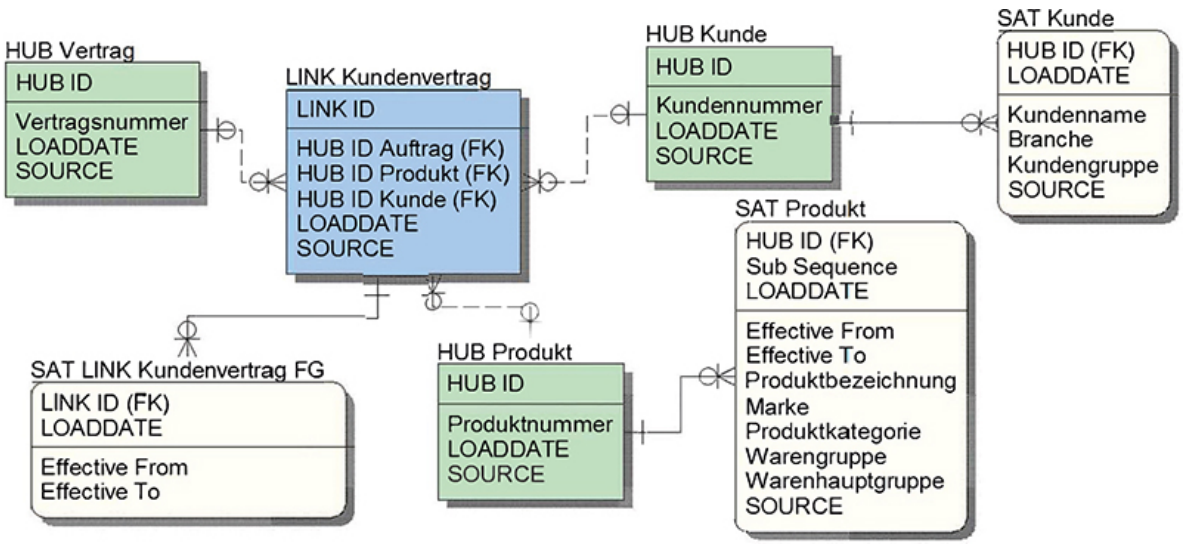


Abb. 4–8 *Bitemporale Historisierung im Data Vault*

Die Historisierung der Beziehungen kann über sogenannte *effective satellites* erfolgen. Dies sind Satellit-Tabellen, in denen die fachliche Gültigkeit einer Beziehung (repräsentiert in der Link-Tabelle) abgelegt ist. In Abbildung 4–7 ist die Link-Tabelle des Kundenvertrags so modelliert. Für die Ablage mehrerer fachlicher Versionen zu einem technischen Gültigkeitsdatum dienen die *multi active satellites*. Dies ist exemplarisch für den Satelliten zum Produkt so abgebildet.

4.4.3 Harmonisierung von fachlichen Schlüsseln

Die Hub-Tabellen bilden die wesentlichen konstituierenden Bausteine eines Data-Vault-Modells und ergeben sich aus den fachlichen Schlüsseln. Alle in operativen Systemen genutzten Business Key sind dafür geeignete Kandidaten. Vielfach ist es jedoch so, dass ein fachlicher Schlüssel wie z. B. eine Kundennummer sich nicht nur aus einem Quellsystem ergibt, sondern in mehreren Quellsystemen entsteht. Sind dabei die auftretenden Nummernkreise disjunkt, können diese problemlos in den gleichen Hub aufgenommen werden, da dort bereits die Zuordnung zum Quellsystem grundsätzlich verankert ist. Diese ist aber nicht Bestandteil des Schlüssels. Es handelt sich somit nicht um einen zusammengesetzten fachlichen Schlüssel, bestehend etwa aus dem Quellsystem und der Kundennummer.

In einem Hub können daher überlappende Nummernkreise nicht ohne Weiteres abgebildet werden. Im Konzept der Data-Vault-Modellierung dient dazu der sogenannte *Business Vault*, ein abgeleitetes Data-Vault-Modell, in dem diese Schritte der fachlichen Harmonisierung verschiedenster, auch überlappender

Schlüsselbereiche aus verschiedenen operativen Systemen erfolgt (siehe [Linstedt 2011]).

Dieses Verfahren der innerhalb der Architektur nachgelagerten Anwendung von Regeln lässt sich als Best Practice in einer Data-Vault–Architektur formulieren. Je änderungsanfälliger eine Geschäftsregel ist, umso weiter oben und näher am Nutzer sollte diese Regel Berücksichtigung finden. In dieser Hinsicht können, in Abgrenzung zum Ansatz der 3NF-Modellierung, die Schritte der Harmonisierung bereits von den Schritten der Filterung getrennt werden. Im Data Vault wird in der ersten Schicht noch keine Regel Berücksichtigung finden, die sich im Sinne einer Geschäftsregel ändern kann. Die Schritte der Harmonisierung, die in einem 3NF-Modell bereits vor dem Core Data Warehouse stattfinden, sind hier nur zum Teil berücksichtigt, die volatilen Regeln sind zwischen dem ersten Data Vault und der nächsten Schicht, dem sogenannten *Business Vault*, angesiedelt.

4.4.4 Bewertung der Data-Vault-Methode

Eine wesentliche Eigenschaft eines Data-Vault-Modells ist dessen Stabilität in Bezug auf Änderungen. Dies postulierte Dan Linstedt bereits als notwendige Bedingung während der Entstehung der Methode.[14]

Kommt es zu Änderungen eines Staging-Modells sind die Auswirkungen auf das Modell des Core Data Warehouse zu betrachten. In einem Data Vault gilt das Grundprinzip, dass einmal erstellte Tabellen sich nicht mehr ändern. Änderungen werden nur durch neue Tabellen aufgefangen. Dadurch ist gewährleistet, dass bereits in Produktion befindliche Tabellen und deren Bewirtschaftung in Form von ETL-Prozessen auch weiterhin Bestand haben und auch gar nicht erneut getestet werden müssen, da diese von den Änderungen nicht betroffen sind. Dieses Verhalten führt neben den als besonders einfach zu automatisierenden Schritten der Modellgenerierung auch zu der als besonders effizient hervorzuhebenden automatisierten Erstellung und Ausführung von Tests.

Für die Änderungen bedeutet dies, dass es ggf. neue Hubs gibt, darüber hinaus aber nur neue Satelliten zur Abbildung der hinzugekommenen Attribute sowie neue Links. Alle vorher genutzten Tabellen bleiben erhalten, denn nur wenn alle bisherigen Modellbestandteile vollständig erhalten bleiben, kann von einer Unanfälligkeit gegenüber Änderungen gesprochen werden.

Nach den bisherigen Ausführungen wird deutlich, dass die Data-Vault-Methode sehr gut geeignet ist, um ein Core Data Warehouse zu modellieren. Mit einem Fokus auf die schnelle Beladung und nahezu keinem Änderungsbedarf im Falle einer Änderung auf Quellsystemseite liegt die Stärke des Ansatzes einerseits darin, Quellsysteme mit hoher Ladegeschwindigkeit anzubinden, und andererseits im relativ geringen Aufwand, mit dem das u. a. durch Mechanismen der Generierung von ETL-Prozessen geschehen soll.

Diese Vorteile führen aber auch zu Strukturen, durch die Abfragen an Data-Vault-Modelle komplex werden und daher nicht für den Endanwender geeignet sind. Ein Core Data Warehouse auf Basis von Data-Vault-Modellen sollte daher nicht direkt für Abfragen und Analysen genutzt werden.

Ein Data-Vault-Modell im Core Data Warehouse deckt die Anforderungen an einen Staging-Bereich im Regelfall mit ab, daher ist dieser nur von temporärer Art und bedarf keiner besonderen Aufbewahrung. Da die Daten im Sinne eines SPOT auch gesammelt, integriert und dann weiter verteilt werden, sind die Aufgaben des Core Data Warehouse selbst auch erfüllt.

Im Falle der Änderungen der Quellsysteme oder der fachlichen Anforderungen wird das Core-Data-Warehouse-Modell nur erweitert. Dadurch ist der Aufwand für Test und Anpassung sehr gering. Jedoch sind dafür die Anpassungen in den nachgelagerten Schichten etwa im Reporting Layer erforderlich und auch größer als beispielsweise beim Star-Schema-Modell als zentralem Paradigma für das Core Data Warehouse. Der Aufwand wird also – teilweise nachgelagert – trotzdem anfallen.

Aspekte der Historisierung auf Basis der technischen Kenntniserlangung im Data Warehouse sind umfänglich gegeben. Für eine echte fachliche Historisierung, die auch Bestandteil der gelieferten Daten von den Quellsystemen sein kann, ist das Modell geringfügig zu verändern.

Ein auf Basis der Data-Vault-Methode gestaltetes Core Data Warehouse hat tendenziell einen hohen Grad an Agilität im Umgang mit Change-Prozessen.

4.5 Fazit

In der aktuellen Diskussion des Aufbaus effizienter BI-Architekturen, die die Heterogenität an Strukturen, Daten und Anforderungen beherrschen, sind zunehmend Anforderungen an Agilität – im Sinn von Time-to-Market – sowie Aspekte der Kostensenkung für Entwicklung und Betrieb relevant. Dabei hat sich

gezeigt, dass durch Automatisierung und Standardverfahren der Generierung Modelle und insbesondere Aufgaben der Datenintegration günstig zu realisieren sind. Durch die sich ergebenden kürzeren Entwicklungszeiten sind kürzere Releasezyklen möglich, eine wichtige Voraussetzung für die agile Entwicklung analytischer Applikationen.

Durch die zunehmende Globalisierung von Unternehmen entstehen auch immer globalere Data-Warehouse-Lösungen, in denen eine stringente Trennung in Online-Zeit und Batch-Zeit nach herkömmlichen Mustern über alle Bereiche hinweg nicht mehr vorkommt. Im vermehrt anzutreffenden sogenannten 24x7-Modus bedarf es einer Entzerrung unter expliziter Berücksichtigung von Performance-Aspekten auch in Realtime-Umgebungen.

In diesem Umfeld haben sich Konzepte der Gestaltung bewährt, die die Prozesse der Datenintegration entzerren und insbesondere unterschiedliche Service Level und Ladezyklizitäten ermöglichen. Konzepte der Virtualisierung und des Self-Service-BI erleichtern die effiziente Nutzung und Implementierung der resultierenden Modelle. Dazu zählen Domänenkonzepte im 3NF-Modell und die Gestaltung von Data-Vault-Modellen.

In Situationen, in denen die BI-Architektur heterogene Anforderungen an die Datenqualität, die Häufigkeit der Datenbewirtschaftung, die Performance und die Verfügbarkeit auszubalancieren hat, ist der Einsatz von Domänenkonzepten und Data-Vault-Modellen ein sehr guter Ansatz, um in diesem Geflecht die Komplexität beherrschbar zu machen und einen Ausgleich zwischen den verschiedenen Anforderungen herzustellen. Bereiche unterschiedlichster Charakteristik können dadurch nebeneinander und trotzdem integriert in der logischen Gesamtarchitektur betrieben werden.

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

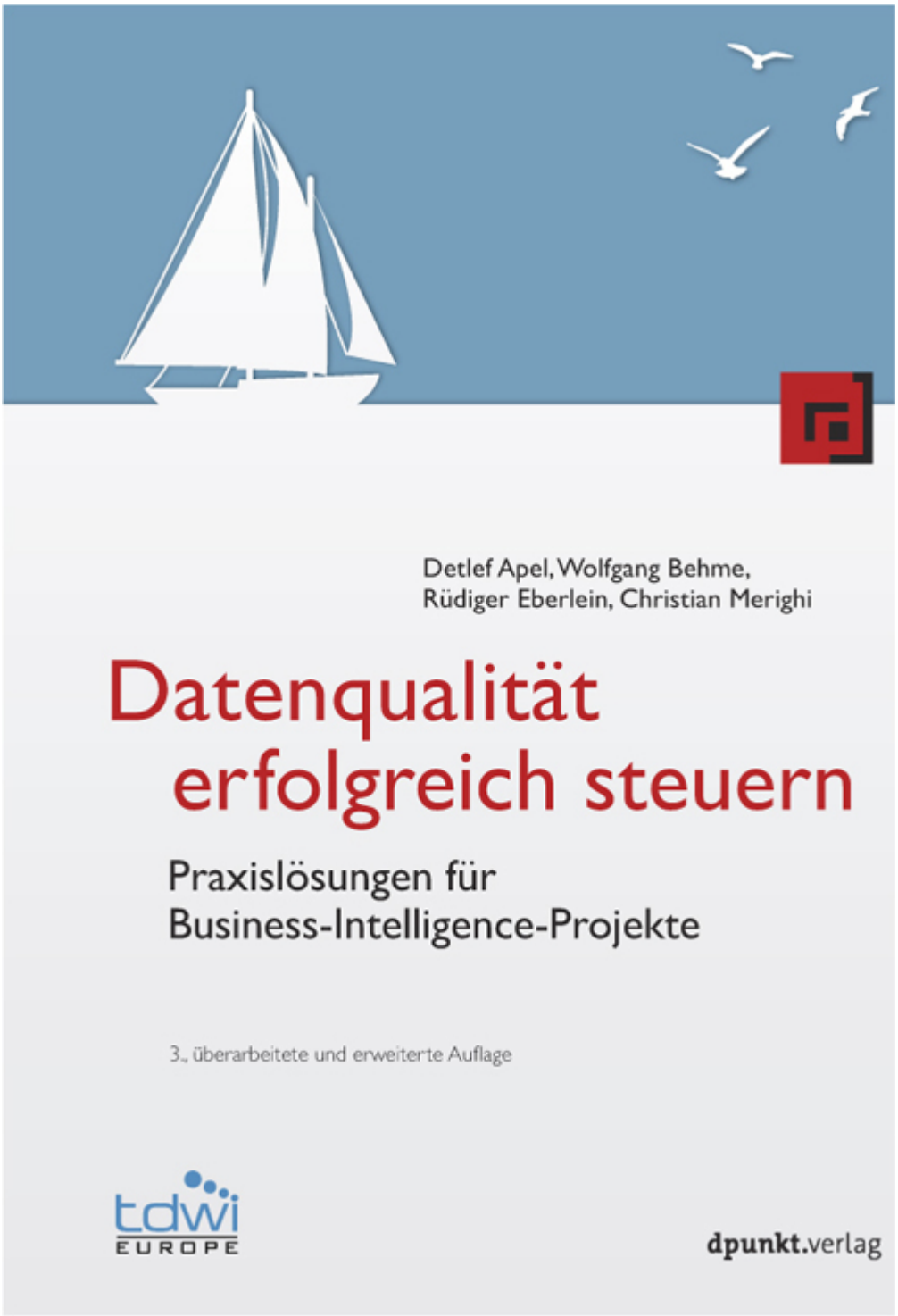
U

V

W

Y

Z



2015, 390 Seiten, Festeinband

€ 69,90 (D)

ISBN 978-3-86490-042-6

In der Edition TDWI erscheinen Titel, die vom dpunkt.verlag gemeinsam mit dem TDWI Germany e.V. ausgewählt und konzipiert werden. Inhaltliche Schwerpunkte

dieser Reihe sind Business Intelligence und Data Warehousing

Detlef Apel · Wolfgang Behme · Rüdiger Eberlein · Christian Merighi

Datenqualität erfolgreich steuern

Praxislösungen für Business-Intelligence-Projekte

3., überarbeitete und erweiterte Auflage

Immer mehr Unternehmen begreifen ein gutes Datenqualitätsmanagement als einen entscheidenden Wettbewerbsvorteil.

Anhand praktischer Beispiele zeigt Ihnen dieses Buch, wie Sie die Qualität Ihrer Daten zielorientiert und nachhaltig verbessern können. Analysieren Sie die Ursachen und Auswirkungen schlechter Datenqualität und erfahren Sie, welche Investitionen sich wirklich lohnen. Lernen Sie die Grundlagen des Datenqualitätsmanagements kennen, die technische Umsetzung mit passgenauen Werkzeugen sowie die praktische Um -setzung in einem kompletten Zyklus eines BI-Projekts.

Die 3. Auflage wurde komplett überarbeitet. Als neues Thema wurde Big Data aufgenommen, da es für die Welt der Business Intelligence eine neue Evolutionsstufe darstellt und somit Auswirkungen auf das Datenqualitätsmanagement hat.

2015, 446 Seiten, Festeinband

Euro 69,90 (D)

ISBN 978-3-86490-043-3

»Das Buch kann somit gleichermaßen BI-Praktikern empfohlen werden, die u.a. die konkreten Checklisten, die klaren Best Practices und konkreten Beispiele schätzen werden, wie auch Lesern aus Forschung und Lehre, die einen Überblick

über die relevanten Ansätze im Kontext der BI-Strategie und BI-Organisation suchen.«

Aus dem Geleitwort von Dr. Henning Baars, Universität Stuttgart

In der Edition TDWI erscheinen Titel, die vom dpunkt.verlag gemeinsam mit dem TDWI Germany e.V. ausgewählt und konzipiert werden. Inhaltliche Schwerpunkte dieser Reihe sind Business Intelligence und Data Warehousing

Tom Gansor · Andreas Totok

Von der Strategie zum Business Intelligence Competency Center (BICC)

Konzeption – Betrieb – Praxis

2., überarbeitete und aktualisierte Auflage

Business Intelligence (BI) stellt in Unternehmen die Versorgung mit entscheidungsorientierten Informationen sicher. Fachanwender und IT-Spezialisten beklagen allerdings häufig die Qualität der Informationen, die in der Regel auf eine fehlende Gesamtstrategie oder die unzureichende Organisation zurückzuführen ist.

Die Autoren erörtern in diesem Buch die verschiedenen Probleme im Einsatz von BI und zeigen praktische Lösungsansätze auf. Sie entwickeln ein Vorgehensmodell, das Inhalte, Architektur und Organisation berücksichtigt. Dabei werden die verschiedenen Varianten eines Business Intelligence Competency Center (BICC), deren organisatorische Verankerung sowie Rollen und Aufgaben beschrieben.

Die 2. Auflage vertieft neue, wesentliche Aspekte des BICC wie Big Data, Mobile BI, Agile BI und Visual BI.