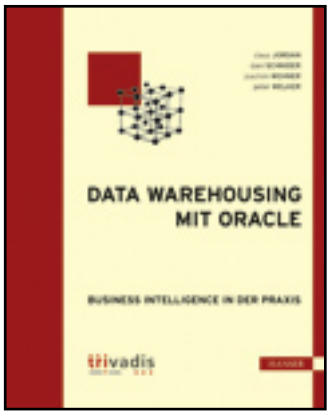
HANSER



Leseprobe

Claus Jordan, Dani Schnider, Joachim Wehner, Peter Welker

Data Warehousing mit Oracle

Business Intelligence in der Praxis

ISBN: 978-3-446-42562-0

Weitere Informationen oder Bestellungen unter

http://www.hanser.de/978-3-446-42562-0

sowie im Buchhandel.

2 Data Warehouse-Grundlagen

Bevor wir uns mit dem Aufbau eines Data Warehouses im Detail beschäftigen, werden in diesem Kapitel die notwendigen Grundlagen erläutert.

- Abschnitt 2.2 beschreibt die grundlegende Architektur eines Data Warehouses und stellt kurz die verschiedenen Komponenten einer DWH-Architektur vor.
- In Abschnitt 2.3 werden verschiedene Methoden der Datenmodellierung sowie die empfohlene Vorgehensweise für die Modellierung von Data Warehouses erklärt.
- In Abschnitt 2.4 finden Sie einen Vorschlag für Namenskonventionen im Data Warehouse, der auch als Basis für die nachfolgenden Kapitel verwendet wird.

2.1 Einführung

Ein Data Warehouse (DWH) stellt die technische Infrastruktur zur Verfügung, die benötigt wird, um Business Intelligence betreiben zu können. Sein Zweck ist es, Daten aus unterschiedlichen Datenquellen zu integrieren und eine historisierte Datenbasis zur Verfügung zu stellen, welche für Standard- und Ad-hoc-Reporting, OLAP[1]-Analysen, Balanced Scorecards, BI-Dashboards und weitere BI-Anwendungen eingesetzt werden kann. Ein DWH ist ein abfrageorientiertes System, mit welchem auf eine Sammlung von historisierten Daten über einen längeren Zeitpunkt zugegriffen werden kann.

Durch diese Ausgangslage ergeben sich einige Unterschiede zwischen einem operativen System (auch OLTP[2]-System genannt) und einem Data Warehouse. Während in einem OLTP-System mehrere bis viele Anwender gleichzeitig Daten einfügen, ändern und löschen, ist dies bei einem DWH-System nicht der Fall. Die einzigen

1 OLAP = Online Analytical Processing

2 OLTP = Online Transaction Processing

„Anwender“, die in ein Data Warehouse schreiben, sind die ETL-Prozesse, welche Daten von den Quellsystemen ins DWH laden. Auch die Art der Abfragen ist unterschiedlich. In operativen Systemen werden typischerweise spezifische Informationen in einem großen Datenbestand gesucht, beispielsweise die letzten zehn Banktransaktionen eines bestimmten Kunden. In einem Data Warehouse hingegen werden meistens Auswertungen über große Datenmengen ausgeführt und aggregiert, zum Beispiel die Summe über alle Verkäufe an alle Kunden einer bestimmten Region.

Um diesen unterschiedlichen Bedürfnissen gerecht zu werden, werden DWH-Datenbanken anders aufgebaut als OLTP-Datenbanken. Architektur, Design und Datenmodellierung funktionieren im DWH-Umfeld nicht auf die gleiche Weise, wie es viele erfahrene Architekten, Datenmodellierer und Entwickler gewohnt sind, die hauptsächlich im Bereich von OLTP-Datenbanken tätig sind. Auf die spezifischen Bedürfnisse von DWH-Systemen wird im folgenden Abschnitt eingegangen.

2.2 Architektur

Die Komplexität und Erweiterbarkeit eines Data Warehouses ist weitgehend abhängig von der verwendeten Architektur. Deshalb ist es in jedem DWH-Projekt von Anfang an wichtig, dass eine saubere Architektur definiert und implementiert wird. Auch wenn dies zum Beginn des Projektes nach Mehraufwand aussieht, zahlt sich eine konsequente Aufteilung in verschiedene DWH-Komponenten im späteren Projektverlauf und im operativen Betrieb des Systems aus.

Leider wird oft der Fehler gemacht, dass aufgrund von knappen Terminvorgaben wesentliche Architekturgrundsätze missachtet und „Abkürzungen“ bzw. „Schnellschüsse“ implementiert werden. Diese Ad-hoc-Lösungen führen früher oder später zu Problemen. Der Aufwand, diese wiederum zu beheben, ist oft größer als der Aufwand, von Anfang an eine saubere Lösung zu realisieren.

2.2.1 DWH-Komponenten

Ein Data Warehouse besteht typischerweise aus verschiedenen Komponenten und Datenflüssen zwischen diesen Komponenten. Auch wenn nicht jedes DWH-System alle Komponenten umfassen muss, lässt sich jedes Data Warehouse auf eine Grundarchitektur, wie sie in Abbildung 2.1 dargestellt ist, zurückführen.

Um den Zweck der einzelnen Komponenten in einer DWH-Architektur zu erklären, werden nachfolgend Vergleiche aus dem „realen Leben“ gemacht. Nehmen wir an, das DWH sei ein großes Lebensmittelgeschäft. Auch dort gibt es verschiedene Bereiche (Komponenten), die jeweils einem bestimmten Zweck dienen.

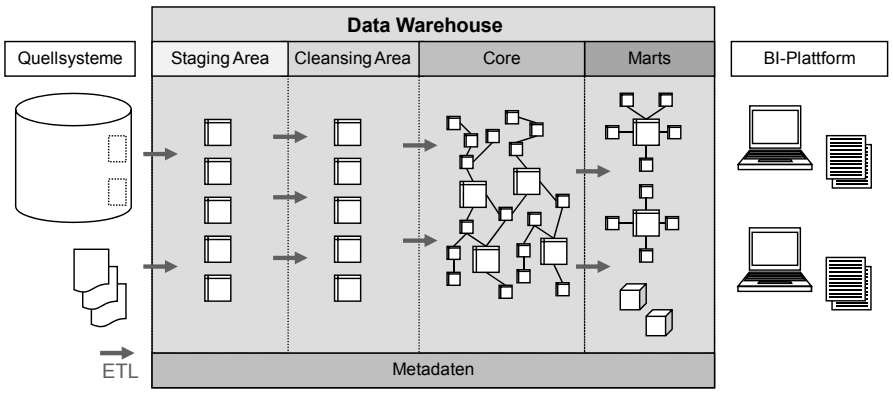


Abbildung 2.1 Grundarchitektur eines Data Warehouses

Folgende Komponenten gehören zu einer vollständigen DWH-Architektur:

- *Staging Area*: Daten aus unterschiedlichen Quellsystemen werden zuerst in die Staging Area geladen. In diesem ersten Bereich des DWH werden die Daten so gespeichert, wie sie angeliefert werden. Die Struktur der Stage-Tabellen entspricht weitgehend der Schnittstelle zum Quellsystem. Beziehungen zwischen den einzelnen Tabellen bestehen keine. Jede Tabelle enthält die Daten der letzten Lieferung, welche vor der nächsten Lieferung gelöscht werden.

  In einem Lebensmittelgeschäft entspricht die Staging Area der Laderampe, an der die Lieferanten (Quellsysteme) ihre Waren (Daten) abliefern. Auch dort werden immer nur die neuesten Lieferungen zwischengelagert, bevor sie in den nächsten Bereich überführt werden.

- *Cleansing*[3] *Area*: Bevor die gelieferten Daten ins Core geladen werden, müssen sie bereinigt werden. Fehlerhafte Daten müssen entweder ausgefiltert, korrigiert oder durch Singletons (Defaultwerte) ergänzt werden. All diese Bereinigungsschritte werden in der Cleansing Area durchgeführt. Auch bestimmte Transformationen, z.B. Währungsumrechnungen oder fachliche Vereinheitlichungen bzw. Ermittlungen von Codes werden hier durchgeführt.

  Im Lebensmittelgeschäft kann die Cleansing Area mit dem Bereich verglichen werden, in dem die Waren für den Verkauf kommissioniert werden. Die Waren werden ausgepackt, Gemüse und Salat werden gewaschen, das Fleisch portioniert, ggf. mehrere Produkte zusammengefasst und alles mit Preisetiketten versehen. Die Qualitätskontrolle der angelieferten Ware gehört ebenfalls in diesen Bereich.

- *Core*: Die Daten aus den verschiedenen Quellsystemen werden über die Staging und Cleansing Area in einer zentralen Komponente, dem Core, zusammengeführt und dort über einen längeren Zeitraum, oft mehrere Jahre, gespeichert. Eine

[3] Das Wort „Cleansing“ wird englisch „klensing“ ausgesprochen und nicht „kliinsing“.

Hauptaufgabe des Core ist es, die Daten aus den unterschiedlichen Quellen zu integrieren und nicht mehr getrennt nach Herkunft, sondern themenspezifisch strukturiert zu speichern. Oft spricht man bei thematischen Teilbereichen im Core von „Subject Areas“. Die Daten werden im Core so abgelegt, dass historische Daten zu jedem späteren Zeitpunkt ermittelt werden können.

Das Core sollte die einzige Datenquelle für die Data Marts sein. Direkte Zugriffe von Benutzern auf das Core sollten möglichst vermieden werden.

Das Core kann mit einem Hochregallager verglichen werden. Waren werden so abgelegt, dass sie jederzeit auffindbar sind, aber der Zugriff darauf ist nur internen Mitarbeitern möglich. Kunden haben im Lager nichts zu suchen – außer vielleicht bei IKEA. Im Gegensatz zu einem Hochregallager bleiben die Daten aber auch dann im Core erhalten, nachdem sie an die Data Marts übertragen wurden.

- *Marts*: In den Data Marts werden Teilmengen der Daten aus dem Core so aufbereitet abgespeichert, dass sie in einer für die Benutzerabfragen geeigneten Form zur Verfügung stehen. Jeder Data Mart sollte nur die für die jeweilige Anwendung relevanten Daten bzw. eine spezielle Sicht auf die Daten enthalten. Das bedeutet, dass typischerweise mehrere Data Marts für unterschiedliche Benutzergruppen und BI-Anwendungen definiert werden. Dadurch kann die Komplexität der Abfragen reduziert werden. Das erhöht die Akzeptanz des DWH-Systems bei den Benutzern.

  Die Data Marts sind die Marktstände oder Verkaufsgestelle im Lebensmittelgeschäft. Jeder Marktstand bietet eine bestimmte Auswahl von Waren an, z.B. Gemüse, Fleisch oder Käse. Die Waren werden so präsentiert, dass sie von der jeweiligen Kundengruppe akzeptiert, also gekauft werden.

- *ETL-Prozesse*: Die Daten, die von den Quellsystemen als Files, Schnittstellentabellen oder über einen View Layer zur Verfügung gestellt werden, werden in die Staging Area geladen, in der Cleansing Area bereinigt und dann im Core integriert und historisiert. Vom Core werden aufgrund von fachlichen Anforderungen Teilmengen oder oft auch nur Aggregate in die verschiedenen Data Marts geladen. All diese Datenflüsse werden unter dem Begriff ETL (Extraction, Transformation, Loading) zusammengefasst. Die Extraktion der Daten aus den Quellsystemen findet in der Regel außerhalb des DWH-Systems statt – nämlich in den Quellsystemen selbst. Als Transformationen werden alle Datenumformungen, Bereinigungen, Anreicherungen mit Zusatzinformationen und Aggregationen bezeichnet. Schließlich werden die Daten in die Zieltabellen der nächsten Komponente geladen.

  Die ETL-Prozesse sind die Mitarbeiter des Lebensmittelgeschäfts, die unterschiedliche Arbeiten verrichten müssen, damit die Lebensmittel vom Lieferanten bis hin zum Kunden gelangen.

- *Metadaten*: Für den reibungsfreien Betrieb des Data Warehouses werden unterschiedliche Arten von Metadaten benötigt. Fachliche Metadaten enthalten fachliche Beschreibungen aller Attribute, Drill-Pfade und Aggregationsregeln für die

Frontend-Applikationen und Codebezeichnungen. Technische Metadaten beschreiben z.B. Datenstrukturen, Mapping-Regeln und Parameter zur ETL-Steuerung. Operative Metadaten beinhalten alle Log-Tabellen, Fehlermeldungen, Protokollierungen der ETL-Prozesse und vieles mehr. Die Metadaten bilden die Infrastruktur eines DWH-Systems und werden als „Daten über Daten" beschrieben.

Auch in unserem Lebensmittelgeschäft braucht es eine funktionierende Infrastruktur – von Wegweisern zur Kasse bis hin zur Klimaüberwachung der Frischwaren.

2.2.2 Transformationsschritte

Mithilfe der ETL-Prozesse werden die Daten von den Quellsystemen ins Data Warehouse geladen. In jeder DWH-Komponente werden dabei unterschiedliche Transformationsschritte durchgeführt, wie in Abbildung 2.2 dargestellt.

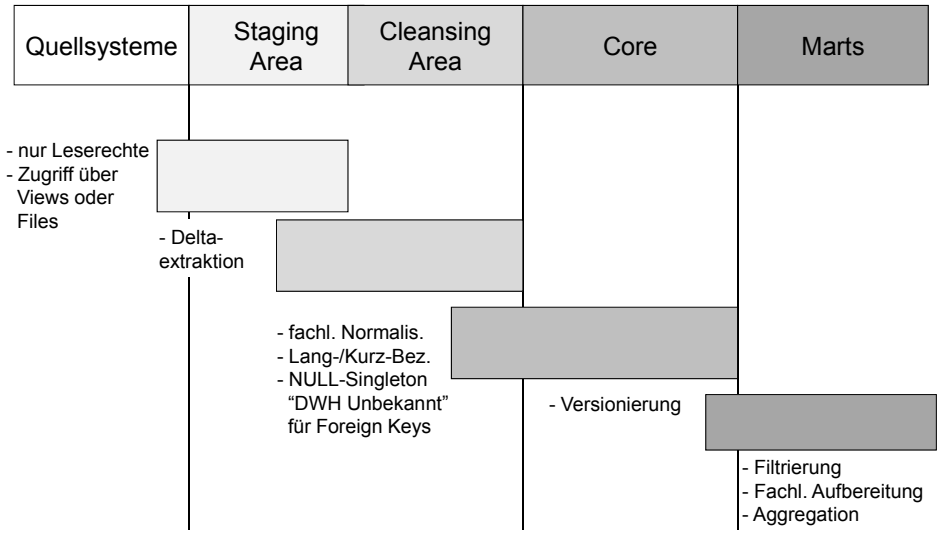


Abbildung 2.2 Transformationsschritte in den DWH-Komponenten

- *Quellsystem → Staging Area*
  Wird direkt auf ein Quellsystem zugegriffen, sollte als Schnittstelle zwischen Quellsystem und DWH ein View Layer definiert werden. Der Zugriff der ETL-Prozesse auf das Quellsystem erfolgt ausschließlich über diese Views. Auf diese Weise kann eine gewisse Unabhängigkeit der ETL-Prozesse gegenüber Strukturänderungen auf dem Quellsystem erreicht werden. Die Views können außerdem für die Delta Extraktion verwendet werden, indem sie so implementiert werden, dass nur die jeweils relevante Teilmenge der Daten in den Views zur Verfügung steht. Ein Beispiel dafür sind Subscriber Views von Oracle Change Data Capture (siehe Abschnitt 3.4.4).

Als Alternative zum direkten Zugriff auf das Quellsystem werden häufig Dateien als Schnittstelle zwischen Quellsystem und DWH verwendet. Die Extraktion der Daten in die Dateien erfolgt auf dem Quellsystem und wird meistens außerhalb des DWH-Projektes realisiert.

Die Daten, ob über Views oder Files geliefert, werden unverändert in die Staging Area geladen.

- *Staging Area → Cleansing Area*
  Beim Laden in die Cleansing Area werden die Daten geprüft, bereinigt und mit zusätzlichen Attributen angereichert. Dazu gehört zum Beispiel die Ermittlung von Lang- und Kurztexten aus den fachlichen Attributen der Quellsysteme. Fehlende oder fehlerhafte Attribute und Foreign Keys werden durch Singletons ersetzt.

  Fehlerhafte Datensätze können je nach Anforderungen ignoriert, automatisch korrigiert, durch Singletons ersetzt oder in Fehlertabellen geschrieben werden. Fehlertabellen können als Basis für Fehlerprotokolle oder manuelle Korrekturen verwendet werden. Bei solchen aufwendigen Varianten der Fehlerbehandlung muss allerdings organisatorisch geklärt werden, wer für die Fehlerkorrekturen verantwortlich ist.

- *Cleansing Area → Core*
  Nachdem die Daten in der Cleansing Area in die benötigte Form aufbereitet wurden, werden sie ins Core geladen. In diesem Schritt findet die Versionierung der Stammdaten[4] statt, d.h. es wird für jeden Datensatz geprüft, ob sich etwas geändert hat und somit eine neue Version erstellt werden muss. Je nachdem, ob für das Core ein relationales oder dimensionales Datenmodell verwendet wird, gibt es verschiedene Varianten der Versionierung von Stammdaten.

  Die Bewegungsdaten[5] werden historisiert. Weil sich Bewegungsdaten nachträglich nicht mehr ändern, heißt das, dass laufend neue Daten eingefügt und über einen längeren Zeitraum gespeichert werden. Oft besteht die Anforderung, dass Bewegungsdaten nach einer gewissen Zeit – in der Regel nach mehreren Jahren – aus dem Core gelöscht werden.

  Aggregationen werden im Core nicht durchgeführt. Die Bewegungsdaten werden auf der Detaillierungsstufe, die geliefert wird, gespeichert.

---

[4] Stammdaten (oder Referenzdaten) sind zustandsorientierte Daten, die sich im Laufe der Zeit ändern können. Um die Änderungen im Core nachvollziehbar abspeichern zu können, werden die Daten versioniert. Das heißt, dass für jede Datenänderung ein neuer Datensatz im Core eingefügt wird. Die Versionierung von Stammdaten wird im Abschnitt 4.5.1 erklärt.

[5] Bewegungsdaten (oder Transaktionsdaten) sind ereignisorientierte Daten, die aufgrund eines bestimmten Bewegungsvorgangs (Transaktion) entstehen und nachträglich nicht mehr geändert werden. Sie sind immer mit einem Ereigniszeitpunkt (z.B. Transaktionsdatum) verbunden. Die Historisierung von Bewegungsdaten wird im Abschnitt 4.5.2 beschrieben.

- *Core → Marts*
  Die Transformationen vom Core in die Data Marts bestehen aus der Filtrierung der Daten auf die für jeden Data Mart erforderliche Teilmenge, der fachlichen Aufbereitung der Dimensionen[6] in die gewünschten Hierarchiestufen sowie der Aggregation der Bewegungsdaten auf die Granularität der Faktentabellen.

2.2.3 Architekturgrundsätze

Obwohl sich die Architektur vieler DWH-Systeme in Details unterscheidet und oft auch unterschiedliche Namen für die Komponenten verwendet werden, gibt es ein paar wichtige Architekturgrundsätze, die auf jeden Fall berücksichtigt werden sollten. Vereinfachungen der Architektur sind erlaubt, aber die wichtigsten Komponenten sollten auf keinen Fall weggelassen werden.

Jedes Data Warehouse besitzt ein Core
Data Marts werden nie direkt aus den Quellsystemen geladen. Einzige Datenquelle für die Data Marts ist das Core[7], welches als „Single Source of Truth" verwendet wird.

Das Core dient als zentrale Integrationsplattform innerhalb des Data Warehouses. Daten aus verschiedenen Quellsystemen müssen in eine vergleichbare Form transformiert werden. Dieser Aufwand soll für jedes Quellsystem nur einmal gemacht werden und nicht für jeden Data Mart separat.

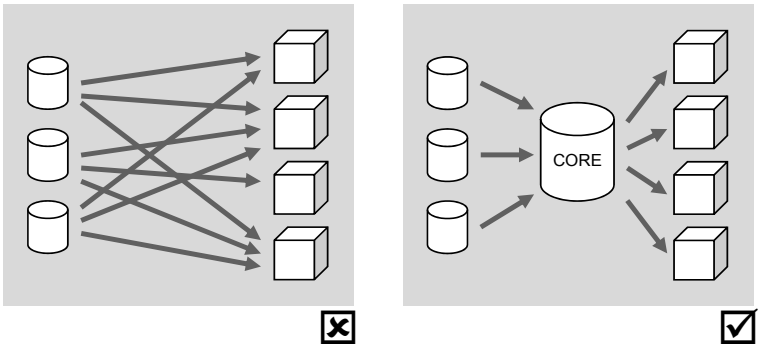


Abbildung 2.3
ETL-Prozesse mit und ohne Core

Bei mehreren Quellsystemen und mehreren Data Marts wird schnell ersichtlich, dass es keine Lösung sein kann, für jede gewünschte Kombination separate ETL-Prozesse zu definieren. Die Komplexität der ETL-Prozesse wird rasch sehr hoch (siehe Abbildung 2.3 links). Durch die mehrfache Implementation kann nicht mehr garantiert werden, dass die Business Rules in allen Data Marts gleich umgesetzt werden. Nur

6 Die Begriffe Dimensionen, Hierarchien und Fakten sind Elemente der dimensionalen Modellierung und werden im Abschnitt 2.3.2.1 erläutert.

7 In Ausnahmefällen können einzelne Data Marts auch aus einem ODS (Operational Data Store) geladen werden, wie in Abschnitt 4.4 beschrieben wird.

durch eine zentrale Komponente wie das Core kann sichergestellt werden, dass die Daten in den unterschiedlichen Data Marts konsistent und somit vergleichbar sind (siehe Abbildung 2.3 rechts).

Eine Ausnahme ist es, wenn nur ein Quellsystem und ein Data Mart vorhanden sind. In diesem Fall kann auf ein Core verzichtet werden. Trotzdem ist es zu empfehlen, diese Komponente zumindest architektonisch vorzusehen, denn typischerweise werden an ein bestehendes Data Warehouse immer neue Anforderungen gestellt, sodass zu einem späteren Zeitpunkt weitere Data Marts dazu kommen. Ein pragmatischer Weg ist es deshalb, zu Beginn ein Core zu realisieren und den ersten Data Mart als „virtuellen Data Mart" (d.h. View Layer) zu implementieren.

Ein weiterer Grund für das Core ist die Historisierung der Daten. Im Core werden alle Bewegungsdaten über eine längere Zeitperiode – oft mehrere Jahre – gespeichert, und die Stammdaten werden versioniert. Dies erlaubt, bei Bedarf einen Data Mart neu laden zu können. Existiert kein Core, so können die historischen Daten nicht mehr rekonstruiert werden, da sie in der Regel in den Quellsystemen nicht mehr vorhanden sind.

Benutzer greifen nie direkt aufs Core zu
Für Abfragen werden Data Marts zur Verfügung gestellt. Das Core hat einen anderen Zweck und ist nicht für direkte Benutzerabfragen optimiert.

Die Datenstrukturen im Core sind darauf ausgelegt, Rohdaten in historisierter Form zu archivieren. Abfragen auf die Core-Tabellen können deshalb – je nach Datenvolumen, Datenmodell und verwendeter Historisierungslogik – sehr komplex werden und somit zu langen Antwortzeiten führen. Das Core ist nicht für Abfragen optimiert.

Für die Benutzer (und für viele BI-Tools) ist es wichtig, dass die SQL-Abfragen möglichst einfach und immer nach einem ähnlichen Muster erfolgen können. Deshalb werden die Daten in geeigneter Form in Data Marts zur Verfügung gestellt.

Wenn die Anforderung besteht, dass bestimmte Benutzergruppen direkt auf das Core zugreifen dürfen, sollte dies so gelöst werden, dass ein View Layer („virtueller Data Mart") zur Verfügung gestellt wird. Die Views enthalten dann beispielsweise die Historisierungslogik, welche die jeweils gültige Version zu einem bestimmten Zeitpunkt zurückliefert. Durch den View Layer können die Abfragen vereinfacht werden, und es besteht eine gewisse Unabhängigkeit der Abfragen von der physischen Speicherung der Daten im Core.

Pro Anwendungsbereich wird ein Data Mart erstellt
Ein universeller Data Mart, der alle Bedürfnisse abdeckt, führt zu hoher Komplexität und langen Antwortzeiten. Dies entspricht nicht den Wünschen der Anwender.

Je nach Anwendungsbereich, Benutzergruppe und verwendetem Frontend-Tool bestehen unterschiedliche Anforderungen an die Datenstrukturen in den Data Marts. Granularität der Daten, historisches Zeitfenster, Häufigkeit der Aktualisierung, aber

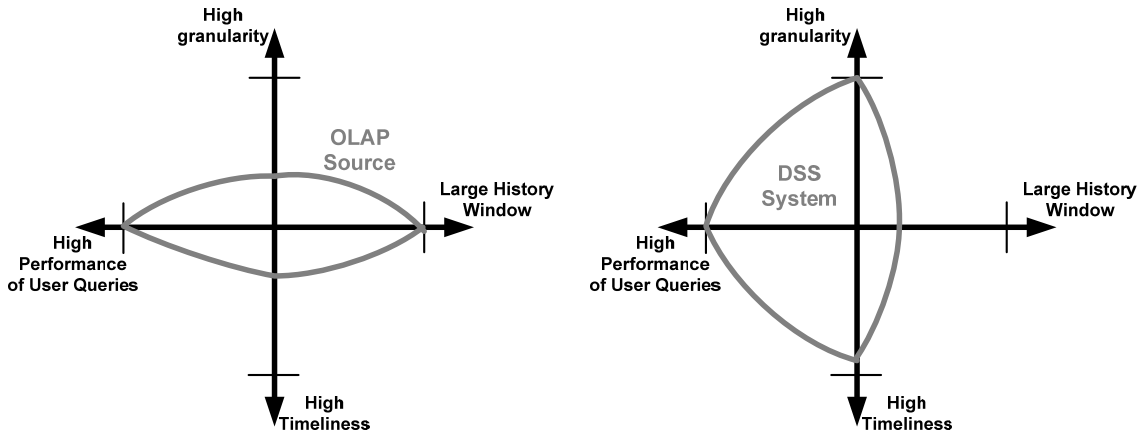


Abbildung 2.4 Unterschiedliche Anforderungen an Data Marts

auch die geforderten Antwortzeiten können sehr unterschiedlich sein. In Abbildung 2.4 wird dies anhand von zwei Data Marts – einen für OLAP-Analysen und einen für ein Decision Support System (DSS) – illustriert.

Für das Reporting werden typischerweise Detaildaten mit feiner Granularität benötigt, während mit einem OLAP-Tool häufig nur auf aggregierte Daten auf hohen Hierarchiestufen zugegriffen wird. Für Standardreports, die einmal täglich berechnet und als PDF-Dateien den Benutzern zur Verfügung gestellt werden, gelten andere Performanceanforderungen als für Online-Analysen, die auf einem Data Mart ausgeführt werden.

Außerdem sollte jeder Data Mart auf die wesentlichen Daten reduziert werden. Nicht jede Benutzergruppe oder Abteilung benötigt sämtliche Dimensionen und Hierarchiestufen für ihre Abfragen. Ein Data Mart mit Dutzenden von Dimensionen ist für die Endbenutzer oft verwirrend. Es empfiehlt sich deshalb, spezifische Data Marts mit den relevanten Dimensionen und Verdichtungsstufen zur Verfügung zu stellen. Diese können viel besser auf die jeweiligen Bedürfnisse der Benutzer optimiert werden.

2.3 Modellierung

Um ein logisches Datenmodell für das Core und für die Data Marts zu erstellen, gibt es unterschiedliche Ansätze. Im Core wird oft ein relationales Datenmodell verwendet, aber es ist je nach Anforderungen und Design des Data Warehouses auch möglich, ein Core mit dimensionalem Datenmodell zu erstellen. In den Data Marts kommt hingegen fast immer die dimensionale Modellierung zum Zug. In den nachfolgenden Abschnitten werden die beiden Arten der Datenmodellierung kurz vorgestellt und anschließend erläutert, was geeignete Vorgehensweisen für die Datenmodellierung im Data Warehouse sind.

2.3.1 Relationale Modellierung

Die klassische relationale Modellierung in der 3. Normalform, wie sie typischerweise in OLTP-Systemen eingesetzt wird, eignet sich nur beschränkt für DWH-Applikationen. Die Modellierung des Core erfolgt jedoch häufig mittels relationaler Datenmodellierung.

2.3.1.1 Normalformen

Ziel der Normalisierung eines relationalen Datenmodells ist es, Redundanz zu vermeiden. Der ursprüngliche Grund, damit Speicherplatz zu sparen, ist heute nicht mehr relevant. Nach wie vor wichtig ist es aber, durch die Vermeidung mehrfach gespeicherter Daten sicherzustellen, dass die Datenkonsistenz gewährleistet werden kann. Weil oft von der „3. Normalform" gesprochen wird, seien hier kurz die Definitionen der wichtigsten Normalformen erwähnt:[8]

- **1. Normalform**: Eine Relation befindet sich in der 1. Normalform, wenn ihre Attribute nur einfache Attributswerte aufweisen.
- **2. Normalform**: Eine Relation ist in der 2. Normalform, wenn sie in der 1. Normalform ist und jedes nicht zum Identifikationsschlüssel gehörige Attribut voll von diesem abhängig ist.
- **3. Normalform**: Eine Relation befindet sich in der 3. Normalform, wenn sie in der 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, transitiv von diesem abhängt.

In der Datenbanktheorie wird dem Normalisierungsprozess viel Bedeutung zugewiesen. In der Praxis wird bei der Erstellung relationaler Datenmodelle jedoch meistens implizit normalisiert, d.h. es wird gleich zu Beginn ein Modell in der 3. Normalform erstellt, ohne dass die einzelnen Normalisierungsschritte im Detail durchlaufen werden. Ein Datenmodell in der 3. Normalform wird oft auch als „normalisiertes Datenmodell" bezeichnet.

2.3.1.2 Darstellung von relationalen Datenmodellen

Relationale Datenmodelle werden typischerweise als Entity-Relationship-Diagramme dargestellt. Die Entitäten (oder Entitätsmengen) werden als Tabellen implementiert. Die Beziehungen (Relationships) zwischen den Entitäten werden als Fremdschlüsselbeziehungen bzw. Foreign Key Constraints implementiert.

---

[8] Die Definitionen stammen aus dem Buch „Informationssysteme und Datenbanken" von Carl August Zehnder (Zehnder 2005). Es gibt neben den drei beschriebenen Normalformen noch weitere, die aber in der Praxis keine Bedeutung haben.

2.3.1.3 Historisierung

Spezielle Techniken müssen für die Historisierung von Daten im relationalen Datenmodell angewendet werden. Insbesondere bei Stammdaten, die sich im Laufe der Zeit ändern können, müssen spezielle Modelle verwendet werden, um die Daten versionieren zu können. Die Problematik dabei ist, dass zu einem bestimmten Datensatz mehrere Versionen existieren können, dass aber Beziehungen nicht auf eine Version verweisen, sondern auf die übergeordnete Entität (also das Element, zu welchem die Versionen gehören).

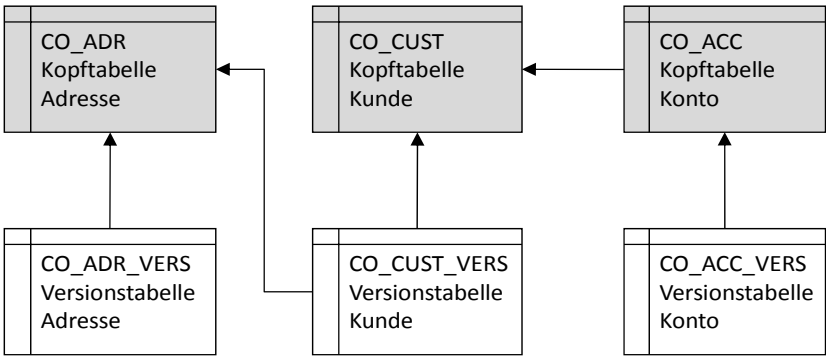


Abbildung 2.5 Beispiel für historisiertes Datenmodell mit Kopf- und Versionstabellen

Eine Modellierungsart, die sich dafür eignet, ist die Modellierung mittels Kopf- und Versionstabellen (siehe Beispiel in Abbildung 2.5). Pro logischer Entitätsmenge werden zwei Tabellen erstellt:

- Die *Kopftabelle* enthält die nichtveränderlichen (statischen) Attribute, insbesondere den Identifikationsschlüssel der Entität.
- Die *Versionstabelle* enthält die veränderlichen (dynamischen) Attribute sowie ein Gültigkeitsintervall. Jeder Eintrag in der Versionstabelle entspricht einer Version der Entität und verweist auf die übergeordnete Entität in der Kopftabelle.
- *Beziehungen* (Foreign Keys) zeigen immer auf die Kopftabelle. Sofern die Beziehung unveränderlich ist, wird der Fremdschlüssel in der Kopftabelle gespeichert (statische Fremdschlüsselbeziehung). Kann sich die Beziehung im Laufe der Zeit ändern, steht der Fremdschlüssel in der Versionstabelle (dynamische Fremdschlüsselbeziehung).

Statische und dynamische Fremdschlüsselbeziehungen

Nicht nur beschreibende Attribute können sich im Laufe der Zeit ändern, sondern auch Beziehungen zwischen Entitäten. Dies jedoch nicht in allen Fällen. Die Beziehung zwischen einem Kunden und seiner Adresse ist dynamisch. Zieht der Kunde an eine neue Adresse, ändert sich der Fremdschlüssel auf die Adresstabelle. Die Beziehung zwischen einem Bankkonto und einem Kunden hingegen ist statisch. Das Konto gehört immer der gleichen Person und kann nicht auf einen anderen Kunden übertragen werden.

Die Datenmodellierung mit Kopf- und Versionstabellen ist sehr flexibel und eignet sich auch für bitemporale Historisierung.

Bitemporale Historisierung
Bei der bitemporalen Historisierung wird unterschieden zwischen der fachlichen Gültigkeit einer Entität und dem technischen Änderungszeitpunkt. Der Zeitpunkt, wann eine Datenänderung durchgeführt wird, kann vor oder nach dem Zeitpunkt liegen, wann die Änderung in Kraft tritt. Bei den Abfragen muss dann entsprechend definiert werden, welcher fachliche und technische Abfragezeitpunkt für die gewünschte Auswertung relevant ist.

2.3.2 Dimensionale Modellierung

Die dimensionale Datenmodellierung eignet sich sehr gut für die Modellierung von Data Marts, aber auch für das Core. Im Gegensatz zur relationalen Modellierung geht es hier nicht mehr primär um die Vermeidung von Redundanz, sondern um eine geeignete Darstellung von hierarchischen Zusammenhängen.

Dimensional ist nicht gleich multidimensional
Dimensionale Modelle können mittels relationaler oder multidimensionaler Technologie implementiert werden. Der Begriff „dimensional“ bezeichnet somit die Art der Modellierung, der Begriff „multidimensional“ hingegen die Art der Implementierung.

2.3.2.1 Elemente des dimensionalen Modells

Bei der dimensionalen Modellierung werden nicht Entitäten und Beziehungen modelliert, sondern es wird ein logisches Datenmodell bestehend aus Dimensionen und Fakten erstellt. Die grundlegenden Begriffe werden hier kurz erläutert:

- *Dimensionen* enthalten beschreibende Informationen, die als Filter- und Gruppierungsmerkmale im dimensionalen Modell dienen. Die Attribute einer Dimension können verwendet werden, um die Resultatmenge einzuschränken (Filter) oder nach bestimmten Kriterien zu gruppieren. Oft enthalten die Dimensionen verschiedene Verdichtungsstufen, die in *Hierarchien* abgebildet werden. Die Hierarchien dienen als Basis für Drill-Up und Drill-Down in OLAP-Tools. Die unterste Hierarchiestufe wird als *Granularität* der Dimension bezeichnet.
- *Kennzahlen* sind numerische Werte, die auf verschiedene Verdichtungsstufen aggregiert werden können. Besteht die Aggregation aus dem Aufsummieren der Kennzahlen, sind die Kennzahlen *additiv*. Bei *non-additiven* Kennzahlen werden andere Aggregationsfunktionen (z.B. Durchschnitt, Minimum, Maximum) verwendet.
- *Fakten* bezeichnen die Verbindungen von Kennzahlen zu einer Kombination von Dimensionen. Fakten haben immer einen Zeitbezug, d.h. zumindest eine der Dimensionen ist die Zeitdimension. Jeder Fakteintrag bezieht sich auf einen be-

stimmten Zeitpunkt. Das kann ein Ereigniszeitpunkt sein (z.B. Transaktionsdatum, Verkaufsdatum) oder ein Messzeitpunkt (z.B. letzter Tag des Monats).

Typisch für Fakten ist es, dass sie sich nachträglich nicht mehr ändern, d.h. ein Fakteintrag, der einmal ins DWH geschrieben wurde, wird nachträglich nicht mehr verändert.[9] Korrekturen (z.B. Stornos) werden als Negativbuchungen ins DWH geschrieben.

2.3.2.2 Darstellung von dimensionalen Datenmodellen

Auch dimensionale Datenmodelle können als Entity-Relationship-Diagramme dargestellt werden (z.B. als Star Schema oder Snowflake Schema). Diese Darstellung ist geeignet für das physische Datenbankdesign, aber nicht gut für die Abbildung der hierarchischen Zusammenhänge. Für die Analyse der fachlichen Anforderungen und für die logische Datenmodellierung sehr gut geeignet ist die ADAPT-Notation, die im Abschnitt 2.3.6 beschrieben wird.

2.3.2.3 Slowly Changing Dimensions

Die Historisierung von Fakten ist sehr einfach, da sich Fakten nicht mehr ändern, nachdem sie einmal geschrieben wurden. Der Inhalt der Dimensionen hingegen kann sich im Laufe der Zeit verändern. Diese Thematik wird als Slowly Changing Dimensions (SCD) bezeichnet (Kimball et al. 2008). Folgende drei Arten von Slowly Changing Dimensions werden unterschieden:

- *SCD I*: Keine Historisierung – Ursprüngliche Attributwerte werden überschrieben.
- *SCD II*: Vollständige Historisierung – Bei Änderungen wird ein neuer Versionsdatensatz geschrieben (Closed Version + New Version).
- *SCD III*: Historisierung der letzten Änderung – Ursprüngliche Attributwerte werden in separaten Attributen gespeichert.

SCD III hat in der Praxis keine Bedeutung. Die Entscheidung, ob SCD I oder SCD II eingesetzt werden soll, hängt von den fachlichen Anforderungen ab. Ist für die Abfragen immer die aktuelle Sichtweise relevant, sollte SCD I eingesetzt werden. Ist für die Abfragen jedoch die Sichtweise zum Ereigniszeitpunkt der Fakten maßgebend (was meistens der Fall ist), sollte SCD II eingesetzt werden. Da SCD I aus SCD II abgeleitet werden kann, aber nicht umgekehrt, sollte für eine vollständige Historisierung im (dimensionalen) Core immer SCD II verwendet werden. Damit kann analog zur Historisierung in der relationalen Modellierung auch mit einem dimensionalen Core eine lückenlose Historisierung der Stammdaten (in diesem Fall der Dimensionen) sichergestellt werden.

---

[9] Dieser Grundsatz wird leider in vielen DWH-Projekten ignoriert. Werden UPDATE-Statements auf Fakten durchgeführt, deutet dies meistens auf einen Fehler bei der Datenmodellierung hin. Änderungen auf Fakten sind höchstens auf technischen Attributen (z.B. Storno-Flag) erlaubt.

2.3.3 Vorgehensweise

Um ein Data Warehouse zu modellieren, also die Datenmodelle für Staging Area, Cleansing Area, Core und Data Marts zu entwickeln, gibt es verschiedene Vorgehensweisen. Nachfolgend sollen die zwei typischen Ansätze vorgestellt werden. In den meisten DWH-Projekten werden Mischformen davon eingesetzt.

2.3.3.1 Anforderungsgetriebene Modellierung (Top-down)

Bei diesem Ansatz werden zuerst nur die fachlichen Anforderungen betrachtet und daraus die Datenmodelle der Data Marts erstellt. Daraus kann das Datenmodell des Core abgeleitet werden. Erst dann werden die Quellsysteme untersucht, um zu ermitteln, woher die Daten beschafft werden können. Im Rahmen einer Gap-Analyse wird untersucht, welche erforderlichen Daten nicht in der gewünschten Form zur Verfügung stehen und unter Umständen von weiteren Datenquellen beschafft werden müssen.

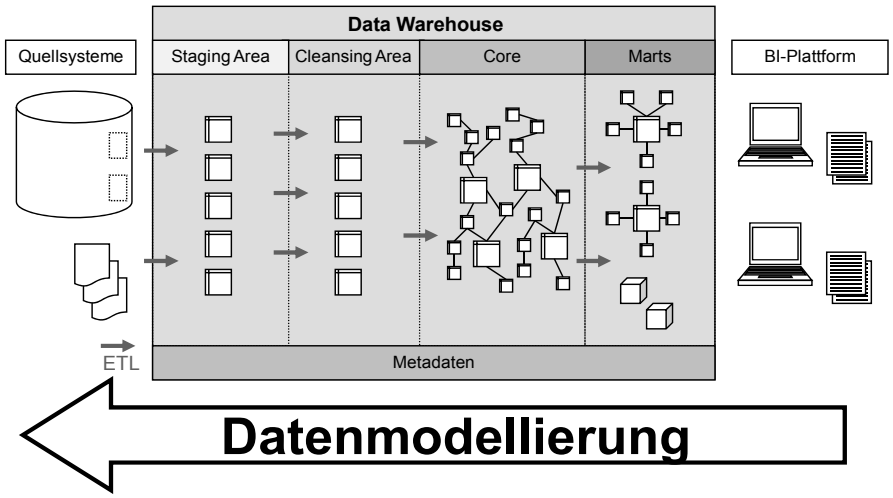


Abbildung 2.6 Top-down-Ansatz für Datenmodellierung

Der Vorteil der in Abbildung 2.6 dargestellten Vorgehensweise ist, dass nur die fachlich relevanten Informationen im DWH gespeichert werden. Allerdings kann dies auch zu einem Nachteil werden: Werden in weiteren Phasen zusätzliche Anforderungen an das DWH gestellt, fehlen die dazu benötigten Informationen im Core, und das Datenmodell sowie die Schnittstellen zu den Quellsystemen und die ETL-Prozesse müssen nachträglich erweitert werden. Dieser Ansatz eignet sich deshalb nur dann, wenn die fachlichen Anforderungen an das Data Warehouse von Anfang an klar und einigermaßen vollständig sind. Nachträgliche Erweiterungen sind möglich, aber aufwendig.

Wichtig bei diesem Ansatz ist deshalb, dass die Fachbereiche, die mit den Data Marts arbeiten werden, bereits zu Beginn des Projekts in die Entscheidungsprozesse eingebunden werden. Welche Informationen in welcher Form in den Data Marts zur Verfügung stehen sollen, muss von den zuständigen Fachbereichen festgelegt werden. Aus diesem Grund sollten geeignete Fachvertreter bei der Anforderungsanalyse und der logischen Modellierung der Data Marts involviert sein.

Die logische Datenmodellierung der Data Marts, also das Festlegen von Dimensionen und Hierarchien sowie Kennzahlen und Fakten, sollte von der Informatik in Zusammenarbeit mit den zuständigen Fachbereichen durchgeführt werden. Die physische Datenmodellierung der Data Marts, insbesondere die technische Umsetzung mittels relationaler oder multidimensionaler Technologie, wird ausschließlich von der Informatik gemacht. Die Fachseite wird für diese Aufgaben nicht einbezogen.

Bei der Datenmodellierung des Core besteht die Hauptschwierigkeit darin zu entscheiden, welche zusätzlichen Informationen im Core gespeichert werden sollen. Werden nur genau die Daten ins Core übernommen, die in den Data Marts benötigt werden, tritt bei zusätzlichen oder der Erweiterung von bestehenden Data Marts das Problem auf, dass das Core auch erweitert werden muss. Dies ist mit viel Aufwand verbunden. Deshalb sollte von Anfang an bei der Modellierung des Core überlegt werden, welche Daten fachlich relevant sein können und somit „auf Vorrat“ im Core abgelegt werden sollen. Ein empfehlenswerter Ansatz ist es, die Daten in der feinsten Granularität, die vom Quellsystem geliefert wird, im Core abzulegen und in den Data Marts auf die erforderliche Verdichtungsstufe zu aggregieren.

2.3.3.2 Quellsystemgetriebene Modellierung (Bottom-up)

Der Bottom-up-Ansatz geht von den zur Verfügung stehenden Daten der Quellsysteme aus, wie in Abbildung 2.7 dargestellt. Vorerst wird ermittelt, welche Informationen für das Data Warehouse relevant sind. Daraus wird das Datenmodell für das Core erstellt. Basierend auf dem Core werden dann für unterschiedliche Bedürfnisse spezifische Data Marts erstellt. Wenn sich bei der Modellierung der Data Marts herausstellt, dass gewisse Informationen nicht zur Verfügung stehen, muss das Core entsprechend erweitert werden.

Diese Vorgehensweise wird oft dann verwendet, wenn die Anforderungen an das DWH unklar oder nur ansatzweise vorhanden sind. Darin besteht auch die Gefahr dieses Ansatzes: Oft werden zu viele oder die falschen Daten ins DWH übernommen, und das Core tendiert zu einem Archivierungssystem für Daten, die in keinem Data Mart benötigt werden. Wenn nur ein Quellsystem vorhanden ist, wird das Core mit großer Wahrscheinlichkeit nichts anderes als eine historisierte Kopie des Quellsystems werden, indem das gleiche oder ein sehr ähnliches Datenmodell verwendet wird.

Ein reiner Bottom-up-Ansatz sollte als Vorgehensweise für die Datenmodellierung eines DWH möglichst vermieden werden, da es typischerweise zu DWH-Systemen

mit vielen Daten führt, aber nicht den erforderlichen Informationen, die in den Data Marts benötigt werden.

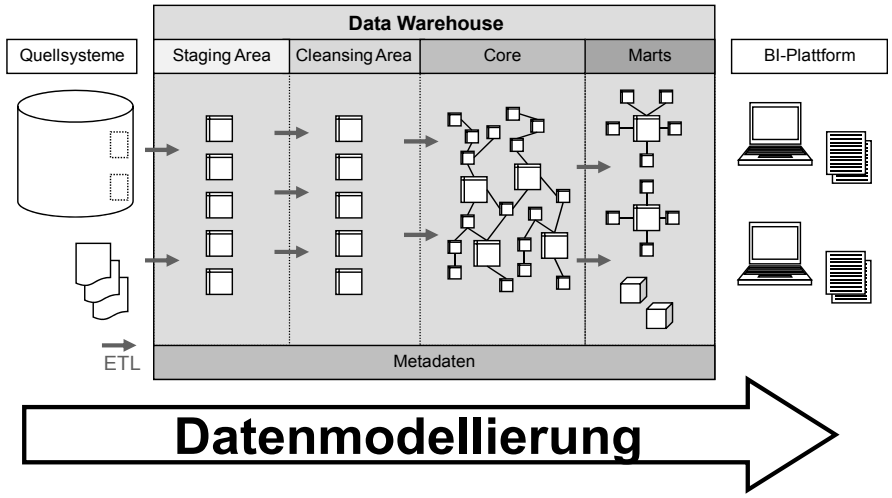


Abbildung 2.7 Bottom-up-Ansatz für Datenmodellierung

2.3.3.3 Kombination von Top-down und Bottom-up

Eine sinnvolle Vorgehensweise ist es, in einem ersten Schritt die Top-down-Modellierung zu wählen, um dann in einer zweiten Phase die von den Quellsystemen ausgehende Bottom-up-Modellierung durchzuführen.

Durch die Kombination der beiden Modellierungsvarianten kann sichergestellt werden, dass die Anforderungen an die Data Marts erfüllt sowie zusätzlich fachlich relevante Informationen ins Core übernommen werden, die momentan zwar noch nicht benötigt werden, aber in Zukunft von Interesse sein könnten.

2.3.4 Modellierung des Core

Beim Erstellen des Core-Datenmodells taucht immer wieder die Frage auf, ob ein relationales Modell in der 3. Normalform oder ein dimensionales Modell mit Star- bzw. Snowflake Schema verwendet werden soll. Die korrekte Antwort lautet: „Es kommt darauf an … ". Die Frage sollte nicht auf eine Glaubensfrage zwischen Inmon[10]- und Kimball[11]-Anhängern reduziert werden, sondern je nach Anforderungen und Umfeld des aktuellen Projekts beurteilt werden.

[10] William H. Inmon gilt als „Vater des Data Warehousing". Sein Buch „Building the Data Warehouse" (Inmon 2005) gilt als DWH-Standardwerk.

[11] Ralph Kimball hat mit seinem Buch „The Data Warehouse Toolkit" (Kimball, Ross 2002) wesentlich zur Verbreitung der dimensionalen Modellierung beigetragen.

2.3.4.1 Relationales Core

Wird das Core primär als Integrationsplattform verschiedener Quellsysteme betrachtet, die als Basis für Data Marts unterschiedlichster Struktur dienen kann, ist es naheliegend, für das Core ein relationales Datenmodell in der 3. Normalform zu erstellen. In einem relationalen Core ist das Datenmodell normalisiert. Redundante Datenhaltung ist nicht vorgesehen, auch nicht in aggregierter Form. Alle Daten werden ausschließlich in der feinsten Granularität gespeichert. Hierarchische Zusammenhänge können in einer Art Snowflake Schema gespeichert werden, aber im Unterschied zur dimensionalen Modellierung sind hier auch Beziehungen zwischen Dimensionen erlaubt. Die Versionierung der Stammdaten erfolgt im relationalen Core, um die Nachvollziehbarkeit aller Änderungen gewährleisten zu können. Bewegungsdaten werden in der Granularität im Core abgelegt, in der sie von den Quellsystemen geliefert werden. Aggregationen finden erst beim Laden der Data Marts statt.

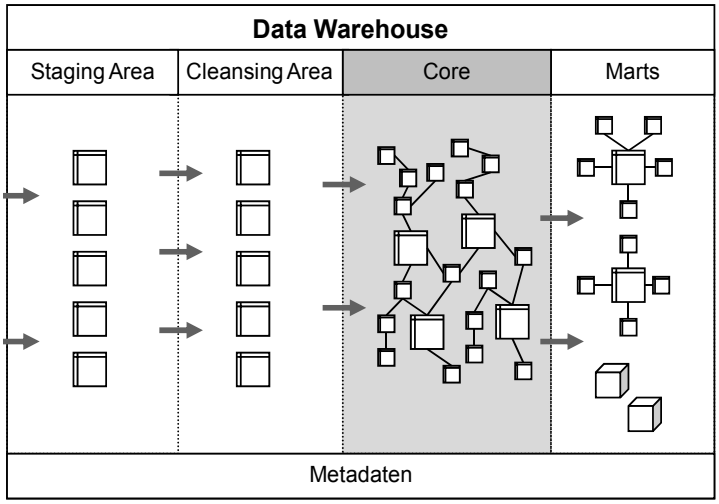


Abbildung 2.8
Data Warehouse mit relationalem Core

Ein relationales Core wie in Abbildung 2.8 ist vor allem dann zweckmäßig, wenn Data Marts mit unterschiedlichen Strukturen beliefert werden sollen oder wenn das Data Warehouse Daten an weitere Zielsysteme oder nicht-dimensionale Data Marts liefern soll. In diesen Fällen werden die Daten je nach Bedürfnissen der Data Marts oder Zielsysteme mittels ETL-Prozessen in eine geeignete Form transformiert. Ein wesentlicher Vorteil des relationalen Core ist die hohe Flexibilität, da das Datenmodell die fachlichen Zusammenhänge abbildet und nicht auf die Strukturen der Data Marts optimiert bzw. eingeschränkt ist. Das bedeutet aber auch, dass die ETL-Prozesse zum Laden der Data Marts komplexer werden, da in diesem Schritt die Transformation in ein dimensionales Modell vorgenommen werden muss. Außerdem besteht die Gefahr, dass bei einem relationalen Core zu stark das Datenmodell des (Haupt-)Quellsystems berücksichtigt wird, dass also eine typische quellsystemgetriebene Vorgehensweise (Bottom-up) verwendet wird.

2.3.4.2 Dimensionales Core

Werden aus dem Core ausschließlich dimensionale Data Marts beliefert, in welcher die Daten in jeweils ähnlicher Form zur Verfügung stehen, kann auch für das Core ein dimensionales Datenmodell verwendet werden. Die einzelnen Data Marts unterscheiden sich nur durch Anzahl und Granularität der Dimensionen. Das Core-Datenmodell enthält alle Dimensionen, die in mindestens einem Data Mart vorkommen, in feinster Granularität. Auch hier finden im Core noch keine Aggregationen statt.

Ein dimensionales Core wird in jedem Fall mittels relationaler Technologie implementiert. Ob dabei Star- oder Snowflake Schemas verwendet werden, ist nicht relevant, weil typischerweise keine Abfragen direkt auf das Core ausgeführt werden. Snowflake Schemas können vorteilhaft sein, da daraus einfacher Dimensionen unterschiedlicher Granularität abgeleitet werden können. Dimensionen werden im dimensionalen Core generell mittels SCD II versioniert, um die Nachvollziehbarkeit aller Änderungen gewährleisten zu können. Bei Bedarf kann daraus beim Laden der Data Marts SCD I abgeleitet werden. Fakten werden in der Granularität im Core abgelegt, wie sie von den Quellsystemen geliefert werden. Aggregationen finden erst beim Laden der Data Marts statt.

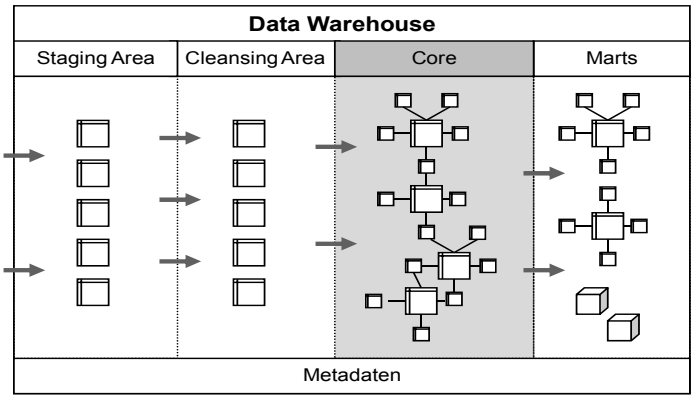


Abbildung 2.9
Data Warehouse mit dimensionalem Core

Ein dimensionales Core wie in Abbildung 2.9 wird meistens dann verwendet, wenn zu Beginn des DWH-Projekts die Anforderungen bereits klar sind und feststeht, welche Data Marts mit welchen Dimensionen zur Verfügung stehen müssen. Voraussetzung dafür ist, dass eine konsequente anforderungsgetriebene Vorgehensweise (Top-down) verwendet wird und dass ausschließlich dimensionale Data Marts beliefert werden, die alle einen ähnlichen Aufbau haben. Der große Vorteil des dimensionalen Core besteht darin, dass die ETL-Prozesse zum Laden der Data Marts sehr einfach sind. Sie bestehen nur aus Filter- und Aggregationsregeln. Komplexere Transformationen sind kaum notwendig, da die Umformung in ein dimensionales Modell bereits beim Laden des Core durchgeführt wird. Grenzen sind dem Modell gesetzt, wenn die gleichen Informationen in unterschiedlichen Data Marts eine andere Bedeutung

haben, beispielsweise in einem Data Mart als beschreibendes Attribut einer Dimension, in einem anderen Data Mart als Kennzahl.

2.3.4.3 Entscheidungsmatrix

Je nach Projekt, Kunde, Anforderungen und Komplexität des DWH-Systems muss entschieden werden, ob das Core relational oder dimensional modelliert werden soll. Die Entscheidungsmatrix in Tabelle 2.1 soll helfen, diese Grundsatzentscheidung zu fällen.

Tabelle 2.1 Entscheidungsmatrix relationales/dimensionales Core

Kriterium	Relationales Core	Dimensionales Core
Komplexität des DWH Das Core bezieht Daten aus verschiedenen Quellsystemen und beliefert unterschiedliche Data Marts und Zielsysteme.	Geeignet für Enterprise-DWHs, welche Data Marts und Zielsysteme mit unterschiedlichsten Datenstrukturen beliefern müssen.	Geeignet für DWH-Systeme, die mehrere dimensionale Data Marts beliefern müssen, welche die gleichen oder ähnliche Datenstrukturen verwenden.
Struktur der Data Marts Die Struktur der verschiedenen Data Marts, die aus dem Core beliefert werden, bestimmen das Datenmodell des Core.	Es gibt neben dimensionalen Data Marts auch Zielsysteme oder Datenempfänger, welche Daten in einer anderen Form benötigen.	Alle Data Marts haben ein dimensionales Datenmodell, und das Core enthält alle Dimensionen und Fakten, die in mindestens einem Data Mart vorkommen.
Anforderungen an DWH Die fachlichen Anforderungen an das DWH bestimmen die Vorgehensweise und damit das Datenmodell des Core.	Können die Anforderungen von der Fachseite zu Projektbeginn noch nicht klar festgelegt werden, wird anhand der vorhandenen Daten der Quellsysteme entschieden, was ins Core geladen wird.	Aufgrund der Anforderungen können die Dimensionen und Fakten der Data Marts festgelegt und daraus das Datenmodell des Core abgeleitet werden.
Vorgehensweise Die Vorgehensweise der Datenmodellierung beeinflusst das Core-Datenmodell.	Quellsystemgetriebene Modellierung (Bottom-up) oder anforderungsgetriebene Modellierung (Top-down)	Anforderungsgetriebene Modellierung (Top-down)

2.3.5 Modellierung der Data Marts

Für Data Marts wird in den meisten Fällen ein dimensionales Datenmodell verwendet, das entweder mittels relationaler oder multidimensionaler Technologie implementiert werden kann. Ebenfalls zur Anwendung kommen Data Marts mit „flachen Tabellen“, d.h. denormalisierten Datenstrukturen, in denen die Informationen in der Form aufbereitet werden, wie sie für die Benutzer dargestellt werden sollen.

2.3.5.1 ROLAP

Wird ein dimensionales Modell mit relationaler Datenbanktechnologie realisiert, spricht man von ROLAP (Relational OLAP). Dimensionen und Fakten werden in Dimensions- und Faktentabellen gespeichert. Die Dimensionstabellen enthalten alle beschreibenden Merkmale der Dimensionen und werden bei den Abfragen zur Filterung und Gruppierung der Daten verwendet. Die Hierarchien innerhalb einer Dimension können in einer Tabelle zusammengefasst oder in separaten Tabellen pro Hierarchiestufe gespeichert werden. Je nach Art der Abbildung von Hierarchien wird das physische Datenmodell als Star Schema oder Snowflake Schema bezeichnet (siehe Abbildung 2.10).

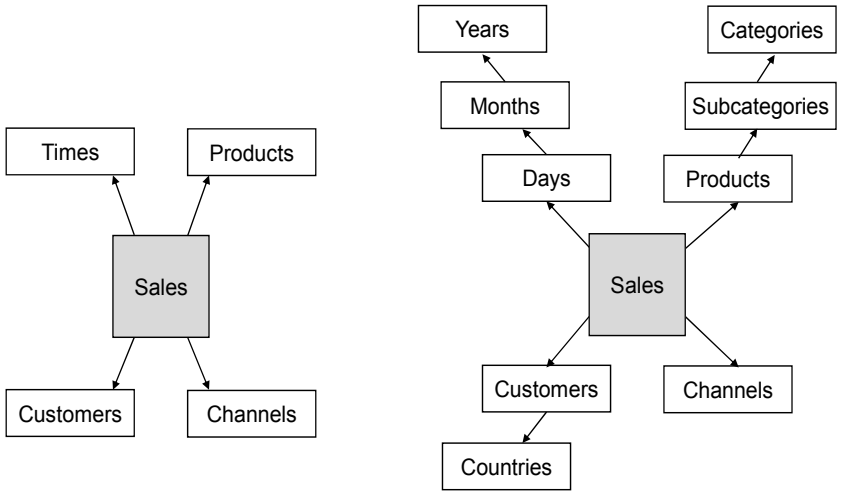


Abbildung 2.10 Star Schema (links) und Snowflake Schema (rechts)

Die Faktentabellen enthalten die Kennzahlen sowie Fremdschlüsselbeziehungen zu den Dimensionstabellen der untersten Hierarchiestufe. Je nach Granularität des Data Marts können Faktentabellen sehr groß werden. Obwohl durch geeignete Performancemaßnahmen (Partitionierung, Materialized Views, Star Transformation, Indexierung) die Abfragen beschleunigt werden können, sollte bereits bei der Modellierung des Data Marts darauf geachtet werden, dass die Granularität nicht möglichst fein ist, sondern nur den benötigten Anforderungen genügt.

Ob das physische Datenmodell eines Data Marts als Star- oder Snowflake Schema realisiert wird, hängt von den Frontend-Tools ab, mit welchen darauf zugegriffen wird. Da die meisten Tools sowohl Star- als auch Snowflake Schemas unterstützen, ist es in den allermeisten Fällen zweckmäßiger, ein Star Schema zu implementieren. Die Abfragen sind einfacher, da weniger Tabellen verbunden werden müssen, was allgemein die Performance verbessert.

2.3.5.2 MOLAP

Wird ein dimensionales Modell mit multidimensionaler Datenbanktechnologie realisiert, spricht man von MOLAP (Multidimensional OLAP). In multidimensionalen Datenbanksystemen werden dimensionale Modelle als sogenannte „Cubes“ gespeichert, obwohl ein multidimensionaler „Würfel“ deutlich mehr als drei Dimensionen besitzen kann.

In einem multidimensionalen Cube werden die Kennzahlen meistens auf allen Hierarchiestufen vorberechnet und abgespeichert. Dies ermöglicht eine sehr gute Abfrageperformance, da im Gegensatz zur relationalen Implementierung während der Abfragen keine Aggregationen durchgeführt werden müssen. Multidimensionale Cubes sind aus diesem Grund vor allem für Data Marts mit sehr hohen Performanceanforderungen geeignet.

Das inkrementelle Laden von multidimensionalen Cubes kann unter Umständen aufwendig sein, da darauf geachtet werden muss, dass Daten nicht versehentlich mehrfach geladen werden und damit das Ergebnis verfälscht wird. Aus diesem Grund empfiehlt es sich, die Cubes vor dem Laden zu löschen und danach neu zu laden. Dies ist jedoch nur möglich, wenn dieser Ladeprozess nicht zu lange dauert. Aus diesem Grund sollten die multidimensionalen Cubes nicht zu groß sein. Hier ist besonders darauf zu achten, dass die Anzahl der Dimensionen klein gehalten und die Granularität nicht zu fein gewählt wird.

Data Marts auf das Wesentliche beschränken

Ein Data Mart sollte immer nur einem spezifischen Anwendungsbereich dienen. Ein universeller Data Mart mit vielen Dimensionen und feiner Granularität auf allen Dimensionen führt zu komplexen Abfragen, schlechten Antwortzeiten und frustrierten Anwendern. Deshalb gilt generell: Besser mehrere separate Data Marts als eine „eierlegende Wollmilchsau“. Dies gilt besonders bei der multidimensionalen Implementierung von Data Marts, da auf diese Weise die Größe und Komplexität der Cubes klein gehalten werden kann.

Multidimensionale Data Marts werden oft für Applikationen eingesetzt, mit denen interaktive Abfragen auf verschiedenen Hierarchiestufen durchgeführt werden (Drill-up, Drill-down, Slice & Dice). Für solche Anwendungen gelten sehr hohe Performanceanforderungen, die mit relationalen Data Marts nur durch zusätzlichen Aufwand (z.B. Erstellung von Materialized Views) erfüllt werden können.

2.3.5.3 Entscheidungsmatrix

Die Entscheidung, ob ein Data Mart relational oder multidimensional implementiert wird, hängt häufig von verschiedenen Faktoren ab, beispielsweise von den bereits vorhandenen Lizenzen für bestimmte Datenbankprodukte. Es besteht jedoch keine Notwendigkeit, alle Data Marts eines DWH-Systems mit der gleichen Technologie zu realisieren. Stehen beide Technologien zur Auswahl, sollte aufgrund der Anforderungen entschieden werden, ob ROLAP oder MOLAP zum Einsatz kommt. Die Mat-

rix in Tabelle 2.2 soll bei der Entscheidung helfen, welche Technologie für einen spezifischen Data Mart besser geeignet ist.

Tabelle 2.2 Entscheidungsmatrix ROLAP/MOLAP

Kriterium	ROLAP	MOLAP
Hohe Abfrageperformance Die Antwortzeiten für die einzelnen Abfragen sollen im Sekundenbereich liegen.	- Um eine gute Performance zu erreichen, sind spezielle Maßnahmen (z.B. MViews, Partitionierung) nötig.	+ + + Da die meisten Daten auf allen Hierarchiestufen vorberechnet sind, ist die Performance der Abfragen sehr hoch.
Abfragen auf Aggregationen Es werden ausschließlich oder vorwiegend Abfragen auf hohen Aggregationsstufen gemacht.	+ Für häufig verwendete Aggregationen können (und sollten) Materialized Views erstellt werden.	+ + + Da die meisten Daten auf allen Hierarchiestufen vorberechnet sind, ist die Performance der Abfragen sehr hoch.
Feine Granularität Es muss möglich sein, mittels Drill-down auf Detaildaten zuzugreifen.	+ Die Faktentabelle wird groß, aber durch Partitionierung kann dieses Problem gelöst werden.	- Eine feine Granularität führt zu einem großen Cube mit vielen leeren Zellen (sparse density).
Hohe Aktualität Es werden laufend (z.B. täglich) neue Daten in den Data Mart geladen.	+ + + Zusätzliche Fakten oder Änderungen von Dimensionen mittels SCD I oder SCD II stellen kein Problem dar.	- Das inkrementelle Laden von Cubes ist oft aufwendig, weshalb häufig der ganze Cube neu berechnet wird.
Lange Historisierung Für Auswertungen sollen Fakten über mehrere Jahre zur Verfügung stehen.	+ + Sofern die Faktentabelle nach Datum partitioniert ist, können historische Daten über mehrere Jahre im Data Mart gespeichert werden.	+ Multidimensionale Cubes können entweder „partitioniert" werden oder es besteht die Möglichkeit, Abfragen über mehrere Cubes zu machen.
Hohe Anzahl Dimensionen Der Data Mart hat sehr viele unterschiedliche Dimensionen.	- Eine hohe Anzahl Dimensionen führt zu vielen Joins bei den Abfragen, weshalb die Anzahl Dimensionen klein gehalten werden sollte.	- Eine hohe Anzahl Dimensionen führt zu vielen Kombinationen und großen Cubes, weshalb die Anzahl Dimensionen klein gehalten werden sollte.

2.3.5.4 Flat Tables

Für Standard-Reporting oder Datenlieferungen an externe Zielsysteme werden die Daten immer in der gleichen Struktur abgefragt. Für diese Art von Anwendungen ist es zweckmäßig, den Data Mart so zu modellieren, dass die Daten direkt in der erforderlichen Struktur abgespeichert werden. Eine typische Modellierungsart ist die Denormalisierung der Daten in „flache Tabellen", in welchen beschreibende Attribute und Kennzahlen nebeneinander dargestellt werden.

Da die Logik zur Aufbereitung der Daten in den ETL-Prozessen zwischen Core und Data Mart implementiert wird, sind die Abfragen auf einen solchen Data Mart sehr einfach und bestehen nur noch im Lesen der denormalisierten Tabelle.

Solche Data Marts können auch als „virtuelle Data Marts“ implementiert werden. In diesem Fall werden die Daten nicht physisch in einer Flat Table gespeichert, sondern der Data Mart besteht nur aus Views, welche auf die Basistabellen im Core zugreifen und die Daten in denormalisierter Form darstellen.

2.3.6 ADAPT-Notation für fachliche Anforderungen

Es gibt verschiedene Darstellungsmöglichkeiten für dimensionale Datenmodelle, aber nicht alle eignen sich für die Besprechung der fachlichen Anforderungen mit den Vertretern der Fachabteilungen. Ein Entity-Relationship-Diagramm ist eine nützliche Darstellung, um mit einem DBA das physische Datenbankdesign zu diskutieren, aber für die Darstellung der fachlichen Bedürfnisse nicht sonderlich gut geeignet. Die hierarchischen Zusammenhänge innerhalb der Dimensionen sind in einem Entity-Relationship-Diagramm nicht oder nur schwer sichtbar.

Eine Designmethode, die sich für die Erstellung dimensionaler Modelle sehr gut eignet, ist ADAPT (Application Design for Analytical Processing Technologies) der Firma Symmetry Corporation (http://www.symcorp.com). ADAPT erlaubt die Darstellung von Dimensionen mit Hierarchien und Hierarchiestufen sowie Fakten und deren Zuordnung zu Dimensionen. Mit ADAPT-Diagrammen lassen sich nicht nur die hierarchischen Zusammenhänge innerhalb der Dimensionen gut darstellen, sondern auch die Granularität der Fakten.

ADAPT-Symbole können in Visio mittels des „FREE ADAPT Visio Stencil“[12] verwendet werden. Dort ist ebenfalls ein White Paper „Getting Started with ADAPT“[13] zu finden, das eine gute Einführung in die ADAPT-Notation bietet.

Ein ADAPT-Diagramm ist nicht ein Ersatz für ein Entity-Relationship-Diagramm eines Data Marts, sondern dient vor allem zur Dokumentation der fachlichen Anforderungen. Die Diagramme werden – nach einer kurzen Einführung in die ADAPT-Notation – sowohl von Vertretern der Fachabteilungen als auch der Informatik verstanden. Sie können somit gut verwendet werden, um die fachlichen Anforderungen zu erarbeiten und zu dokumentieren. Abbildung 2.11 auf der nächsten Seite zeigt ein Beispiel für ein solches ADAPT-Diagramm.

---

12 http://www.symcorp.com/downloads/ADAPTv3_visio_stencil.zip

13 http://www.symcorp.com/downloads/ADAPT_white_paper.pdf

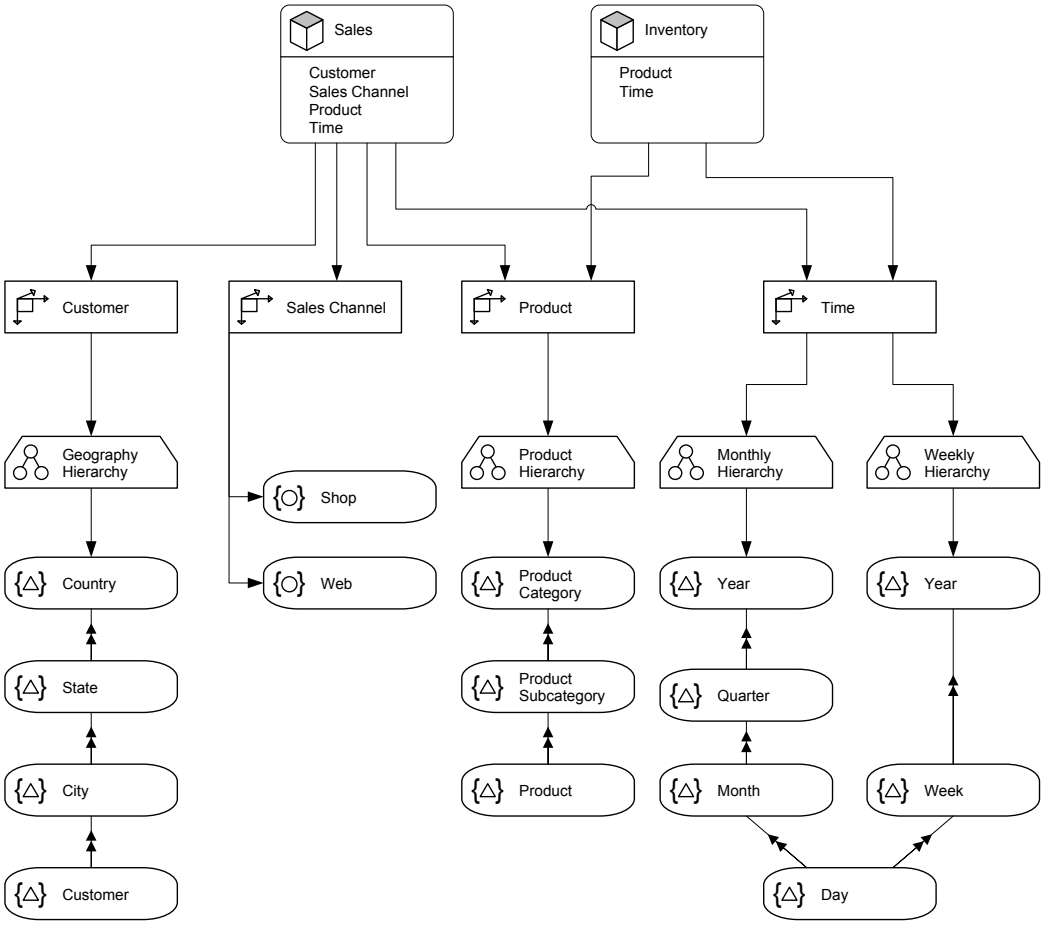


Abbildung 2.11 Beispiel für ADAPT-Diagramm eines Data Marts

Praxistipp: ADAPT-Notation für Anforderungsanalyse

Als Ansatz hat sich gut bewährt, zusammen mit Vertretern der Fachabteilungen Workshops durchzuführen. Darin werden die fachlichen Anforderungen ermittelt und unter Verwendung der ADAPT-Notation dokumentiert. Dazu zeichnet der Workshop-Moderator die entsprechenden ADAPT-Diagramme auf dem Laptop und projiziert sie gleichzeitig via Beamer, damit sie für alle sichtbar sind. Nach dem Workshop werden die Diagramme bereinigt (nur Darstellung, nicht Inhalt!) und als Protokoll des Workshops an die Teilnehmer verschickt. Diese Version dient dann als Basis für den nächsten Workshop.

Die ADAPT-Diagramme zeigen nur die hierarchischen Zusammenhänge der Dimensionen sowie eventuell spezifische Attribute, Ausprägungen oder Wertebereiche. Für eine vollständige Auflistung aller Attribute und Datentypen ist diese Notation nicht geeignet. Deshalb wird für das physische Datenmodell des Data Marts zusätzlich ein Entity-Relationship-Diagramm erstellt, das dann auch als Basis für die Erstellung der Dimensions- und Faktentabellen dient.

Bei Data Marts, die mittels MOLAP-Technologie implementiert werden, kann die Definition der Hierarchien und Drill-Pfade direkt aus den ADAPT-Diagrammen abgeleitet werden. Auch hier müssen zusätzlich die Attribute der einzelnen Hierarchiestufen dokumentiert werden.

2.4 Konventionen

Die vier Autoren sind gemeinsam mit ihren Kollegen in vielen Data Warehouse-Projekten bei den unterschiedlichsten Kunden tätig. In fast jedem Projekt gibt es individuelle Namenskonventionen. Gründe dafür können zum Beispiel sein, dass

- der Kunde seine eigenen festen Definitionen hat, von denen er auch im DWH nicht abweichen möchte,
- jeder Designer bzw. Berater seinen eigenen Designstil in das Projekt einbringt,
- externe IT-Dienstleister das Design vorgegeben haben,
- es inzwischen einige BI/DWH-Literatur mit eigenen Konventionen gibt und
- die eingesetzten Design-Tools bestimmte Namenskonventionen vorgeben (z. B. legt der SCD II Wizard des Oracle Warehouse Builders alle Dimensionsspalten selber an, die allerdings im Nachhinein noch geändert werden können).

Warum kann eine einheitliche Namenskonvention im Projekt oder sogar unternehmensweit von Vorteil sein?

- Neue Projektmitarbeiter – sowohl von Dienstleistern als auch vom Kunden – können sich schneller in das bestehende Projekt einarbeiten. Man spricht also die gleiche Sprache.
- Da alle Tabellen gleiche Namenskonventionen und die gleiche Sortierreihenfolge der Spalten haben, kann schneller und mit geringerer Fehlerquote implementiert werden.
- Quality Checks sind wesentlich leichter zu erstellen bzw. einfach generierbar (siehe Kapitel 3.2).
- An einen internen/externen Mailverteiler geschickte Frage-/Antwort-Mails mit spezifischen Designfragen werden von Dritten schneller verstanden und beantwortet.
- Entwickelte Teillösungen können durch geringfügige Modifikationen in anderen Projekten wieder verwendet werden. Gemäß den Erfahrungen der Autoren kann dies eine Steigerung der Entwicklungsgeschwindigkeit von 30-50 % bedeuten.

Hinweis

Beim Einsatz des OWBs kann dies ideal durch den Einsatz der Skriptsprache OMB Plus erfolgen, mit dessen Hilfe von Objekten bis hin zu Mappings alles generiert werden kann, was auch manuell mit dem OWB möglich ist. Dies ist gerade bei sich oft ähnelnden Designschritten von Vorteil. Neben dem Zeitgewinn wird zusätzlich die Designqualität erhöht, da sich deutlich seltener sogenannte Leichtigkeitsfehler einschleichen.