In der Komplexitätsfalle

Wie bleibt ein Data Warehouse erweiterbar?

Juri Urbainczyk
iteratec GmbH
Inselkammerstraße 4
82008 München-Unterhaching
juri.urbainczyk@iteratec.de

1 Abstract

Business Intelligence (BI) Systeme und Data Warehouses (DWH) sind wichtig für jedes Unternehmen, um Management-Entscheidungen auf der Basis von relevanten Informationen zu unterstützen. Ein DWH zu etablieren ist teuer. Die Wartung und Erweiterung nach der Inbetriebnahme ist meist teurer als erwartet – wobei mit der Zeit der Aufwand für Erweiterungen leider eher steigt als sinkt. Damit wächst das Risiko von unternehmerischen Fehlentscheidungen aufgrund der sinkenden Datenqualität und später Verfügbarkeit von Auswertungen. Der unmittelbare Grund liegt häufig im Vorgehen, welches die Komplexität des Datenmodells mit jeder Erweiterung anwachsen lässt. Dieses White Paper analysiert diese Situation und stellt einen Ansatz vor, mit dem das DWH auf Erweiterungen vorbereitet wird und somit die Kosten und Risiken in späteren Projektphasen gesenkt werden können.

2 Das Data Warehouse wächst

Gerade in Zeiten eines schwierigen wirtschaftlichen Umfelds ist eine ausreichende Transparenz als Basis für strategische Entscheidungen der Unternehmensführungen immer wichtiger. Business Intelligence Systeme wie z.B. ein Data Warehouse sollen diese Entscheidungen durch Bereitstellung der notwendigen Informationen vorbereiten und unterstützen. Viele Unternehmen haben bereits ein BI System aufgebaut und haben Erfahrungen mit einem solchen Projekt gesammelt.

DWH Projekte sind extrem komplex, da potenziell die gesamte Fachlichkeit des Unternehmens berührt wird und enorme Datenintegrationsaufwände zu leisten sind. Zu Beginn des Projekts ist diese Komplexität jedoch nicht unmittelbar sichtbar, da ein DWH nie in einem Big Bang entsteht, sondern inkrementell wächst, was schon allein darin begründet ist, dass anfangs vermutlich nur einige wenige Berichte notwendig sind, und daher auch nicht alle Daten sofort integriert werden müssen. Das DWH wächst also blumenkohlförmig. Dieses Problem lässt sich im Übrigen nicht dadurch lösen, dass man initial *alle* Daten in das DWH überführt. Dadurch schafft man höchsten eine Kopie des operativen Datenmodells und verlagert das Problem auf die nächste Projektphase.

Die Schwierigkeiten beginnen nach einigen Inkrementen: es stellt sich heraus, dass der Aufwand für die Erweiterung des DWH mit der Zeit eher anwächst als kleiner wird. Dazu kommen Probleme mit falschen Reports und inkonsistenten Daten. Irgendwann kommt ein Punkt, an dem der Aufwand für die Erweiterung des DWH den Nutzen übersteigt. Es wird zu teuer, das DWH weiter zu warten und zu pflegen. Woran liegt das?

Auslöser für Erweiterungen am DWH sind Anforderungen aus dem Fachbereich: „Wir brauchen einen neuen Report, der so ähnlich aussieht…!“ Wenn man Glück hat, bekommt man eine grobe Beschreibung der gewünschten Auswertung. Da nur selten alle Informationen für diese Auswertung im DWH verfügbar sind, müssen die notwendigen Daten bestimmt und aus den Quellsystemen extrahiert werden. An dieser Stelle fangen die Probleme an: man begibt sich auf die Suche nach den Quelldaten, die den Wünschen des Fachbereichs entsprechen. Glaubt man, sie gefunden zu haben, werden die Daten kurzentschlossen in das Data Warehouse überführt. Vor allem bei Faktentabellen hat man dann den geringsten Aufwand, wenn man für die Daten nahezu 1:1 das gleiche Datenmodell wie im operativen System wählt. Meistens werden auch noch einige neue Dimensionen nötig.

Dieses Vorgehen führt dazu, dass das Datenmodell des DWH mit jeder Erweiterung um nicht wenige neue Tabellen anwächst. Leider haben diese Tabellen meist eine Struktur, die nicht zu den bisherigen DWH Tabellen passt, da sie aus anderen Quellsystemen stammen. Manche Daten sind nun mehrfach im DWH: da auf eine

grundlegende Analyse verzichtet wurde, hat man nicht erkannt, dass ein Teil der neuen Daten nur eine andere Ausprägung von schon im DWH vorhandenen Daten sind. Das gilt insbesondere für die Dimensionen.

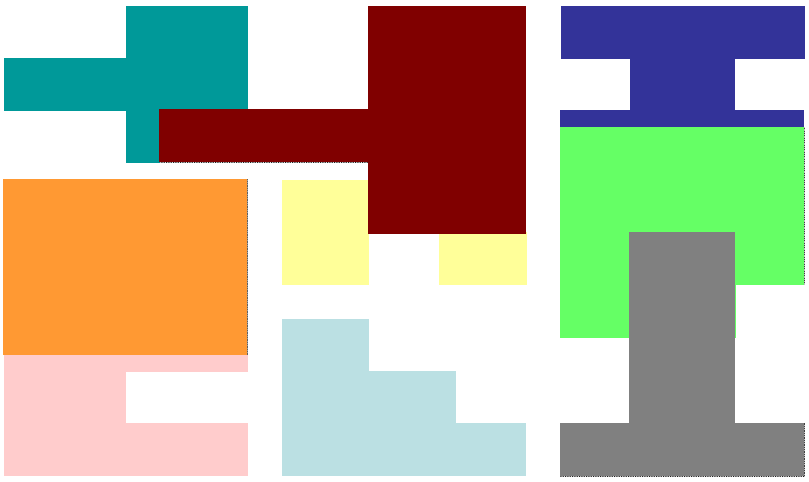

Abbildung 1: Das Datenmodell nach einer Reihe von Erweiterungen: die „Teilmodelle“ passen nicht zusammen. Das Datenmodell ist inkonsistent und redundant.

Auf diese Weise kopiert man die operativen Datenmodelle eines nach dem anderen in das DWH. Damit handelt man sich unmerklich viele weitere Probleme ein: zwar kann man den unmittelbaren Bedarf des Fachbereichs befriedigen, doch wächst das Datenmodell unkontrolliert und ungeplant weiter. Man hat auf einmal dieselben Daten mehrfach oder nur mit leicht abweichender Semantik oder Struktur im Data Warehouse. Viele Attribute liegen nun auch (unter anderen Namen) mehrfach vor, wobei die Wertemengen jeweils leicht abweichen dürften. Die Redundanz nimmt zu während die Konsistenz abnimmt.

Nicht nur dass das DWH dadurch unnötig groß wird; es wird auch immer schwieriger zu erweitern. Denn die Daten, die man auf diese Weise so „schnell“ zur Auswertung gebracht hat, haben in den meisten Fällen jetzt eine Struktur, die nicht allgemeingültig und nicht wieder verwendbar ist. Bei der nächsten Erweiterung wird man nicht mehr genau wissen, was denn nun die Unterschiede zwischen diesen Daten sind und welche man für den Bericht hernehmen soll. Es kann somit nicht mehr auf diese sehr speziellen Daten aufgesetzt werden und man muss erneut - leicht andere - Daten aus dem operativen System übernehmen. Dadurch wachsen die Aufwände für zukünftige Erweiterungen.

Ein Beispiel: für ein Automobil gibt es im Händlersystem die Statuswerte „bestellt“, „angefordert“, „produziert“, „geliefert“ und „übergeben“. Das Produktionssystem verfolgt jedoch die Statuswerte „bestellt“, „geprüft“, „geplant“, „fertig gestellt“. Jetzt könnte man im DWH jedes dieser Attribute getrennt speichern (z.B. jeweils als separate Dimension). Wird jedoch irgendwann ein Bericht benötigt, der den gesamten Lebenszyklus des Automobils verfolgen soll, kann dieser nicht dargestellt werden oder er muss algorithmisch aus den einzelnen Attributen zusammengeführt werden. Dabei tut sich ein weiterer Stolperstein auf, da der Statuswert „bestellt“ in einem Kontext „bestellt durch den Kunden“ und im anderen „bestellt vom Händler“ bedeutet. Vermutlich wird man für den neuen Bericht den Statuswert wieder aus einem anderen System (z.B. SAP) extrahieren, mit erneut abweichender Semantik und Speicherung, was die Komplexität im DWH weiter erhöht. Man geht festen Schritts der Komplexitätsfalle entgegen.

3 Das unternehmensweite Datenmodell

„Natürlich!“ wird nun der Datenbank-Experte rufen. „Man muss auch vorher ein unternehmensweites Datenmodell für die Auswertungen bestimmen, und das wird dann mit dem DWH umgesetzt.“. Dieser Ansatz hat auf den ersten Blick viel für sich. Zuerst überlegt man sich, was man eigentlich auf welche Weise auswerten möchte, und baut dann das System nach diesen Überlegungen auf. Dadurch bleibt das Datenmodell immer konsistent, übersichtlich und erweiterbar. William H. Inmon vertritt genau diese These, dass eine DWH Projekt auf Basis des unternehmensweiten Datenmodells (UWDM) durchgeführt werden sollte [INMON].

Die Erfahrung zeigt jedoch, dass die Dinge nicht so einfach sind. Um das unternehmensweite Berichtsdatenmodell zu erstellen, kann man z.B. die operativen Systeme analysieren und deren Datenmodelle vereinheitlichen. Die Erstellung eines UWDM aber ist eine Sisyphusaufgabe ungeheuren Ausmaßes, an der bereits viele Unternehmen gescheitert sind. Selbst wenn es schließlich gelingt, ein solches Datenmodell zu konstruieren, wird es meist nur noch von denen verstanden, die es modelliert haben. Auch die Probleme der Validität und der Wartung des UWDM sind ungelöst. Alternativ kann man die Anwender befragen, welche Daten sie in ihren Berichten gerne sehen würden (z.B. beschrieben in [NUSS]). Das erscheint machbar und sinnvoll. Aber auch dieses Vorgehen führt nicht zur Lösung des Problems.

Warum? Das Ergebnis einer solchen Befragung spiegelt lediglich die Wünsche der Anwender zu einem bestimmten Zeitpunkt wider. Im Verlauf des DWH Projekts werden jedoch ganz andere Ideen und Vorstellungen auf den Tisch kommen, an die vorher niemand gedacht hat. Hier ist es nicht anders als in anderen Projekten: der Appetit kommt beim Essen. Darüber hinaus sind die Struktur und die Art der möglichen Berichte begrenzt durch die Struktur der Daten in den operativen Systemen. Werden z.B. Buchungen nur taggenau erstellt, kann man keinen Bericht erstellen, der diese auf Stundenbasis anzeigt – zumindest nicht ohne die operativen Systeme und die Geschäftsprozesse des Unternehmens zu ändern (was weit über den Scope und die Möglichkeiten eines DWH Projekts hinausgeht). Die Anwender richten sich aber meist nicht nach den Möglichkeiten der operativen Datenmodelle (schon eher nach der Präsentation in den operativen Systemen) und werden somit Berichtswerte definieren, die durch ein DWH nicht realisierbar sin. D.h. die Vorabfestlegung auf bestimmte Auswertungen und Berichte wird der Realität nicht gerecht und führt zu instabilen und unrealistischen Anforderungen.

4 Lösungsansatz

Das DWH Datenmodell kann nicht im Vorhinein vollständig definiert werden. Was kann dann getan werden, um der Komplexitätsfalle zu entkommen? Man wird damit umgehen müssen, dass der Fachbereich Anforderungen für neue Berichte an das DWH stellt. Das DWH wird also inkrementell dynamisch wachsen, und das ist auch gut so, denn nur so kann der gigantische Umfang der zu analysierenden Daten nach und nach dem Anwender zur Verfügung gestellt werden. Die Vorgaben für die Inkremente kommen aus den fachlichen Anforderungen, also aus dem, was ausgewertet werden soll: die Werte, die im Bericht angezeigt werden sollen, müssen auf irgendeine Weise auch dort hinterlegt sein. Das bedeutet, dass Änderungen am Datenmodell zu erwarten sind, wenn neue Berichte benötigt werden. Das Datenmodell muss also auf jeden Fall so ausgelegt werden, dass es leicht zu ändern ist: es muss also leicht verständlich, konsistent und wenig redundant sein. **Die Erweiterbarkeit muss eine der wesentlichen Eigenschaften des Datenmodells sein**. Diese Eigenschaft des Datenmodells muss mit der Zeit stabil bleiben, darf also nicht durch die Erweiterungen selbst gefährdet werden. Es muss also ein Prozess für die Erweiterungen gefunden werden, der den Erhalt des ausbaufähigen Datenmodells sicherstellt. Wenn es also durch das Vorgehen gelingt, Redundanz und Inkonsistenz zu vermeiden, bleibt das DWH Datenmodell verständlich und erweiterbar!

Die Redundanz wiederum ergibt sich durch das ungesteuerte Übernehmen von Datenmodellen aus den operativen Systemen in das DWH. Die Folgerung muss also lauten: d**ie Datenmodelle der operativen Systeme und des Data Warehouse sind streng zu trennen**. Das ist nicht nur hilfreich, um die Komplexität zu reduzieren, sondern auch, um direkte Abhängigkeiten zwischen den Systemen zu vermeiden. Änderungen der Datenstrukturen in den operativen Systemen würden sonst direkt auf das DWH durchschlagen. Ein Werkzeug, um diese Trennung der Datenmodelle umzusetzen, gibt es bereits in der DWH Architektur: das ETL-Tool. Es bildet die Schnittstelle zwischen operativem System und DWH und entkoppelt die beiden Datenmodelle voreinander. Aber welches Vorgehen ist aber zu wählen, um das DWH Datenmodell zu bestimmen?

Die Trennung der Datenmodelle lässt sich nur bewerkstelligen, wenn man bei jeder Erweiterung des DWH das operative Modell nicht unbesehen übernimmt, sondern durch eine Analyse herausfindet, wie das Datenmodell sinnvoll zu erweitern ist (wenn überhaupt, denn vielleicht sind die Daten, die benötigt werden, ja schon im DWH – eventuell „unter anderem Namen“?). Es ist also **konzeptionelle Vorarbeit bei jeder Erweiterung** notwendig. Diese Konzeptarbeit kann nur erfolgreich sein, wenn

a) das existierende DWH Datenmodell erweiterbar ist und
b) die fachliche Bedeutung der fraglichen Daten der operativen Systeme verstanden werden.

Man muss z.B. erkennen, was es genau bedeutet, wenn in einem Feld des Datensatzes der Wert „20“ oder „21“ steht. Oftmals nämlich stellt sich heraus, dass es inhaltlich das gleiche bedeutet, wie der Wert „XY“ in einem ganz anderen Attribut eines anderen Datensatzes. Manchmal ist die Situation aber auch genau umgekehrt: die Attribute haben zwar genau (oder in etwa) den gleichen Namen, dahinter steckt jedoch eine abweichende oder vollkommen andere fachliche Bedeutung. Hier gibt es also eine Menge Fallgruben, die bei der Analyse

aufzudecken und zu klären sind. Für eine solche Analyse ist i. allg. die Beteiligung des jeweils technisch zuständigen Teams und des betroffenen Fachbereichs notwendig, denn das Wissen über alle operativen Systeme ist bei weitem zu umfangreich, um es in wenigen Personen im DWH Projekt bündeln zu können. Erschwerend kommt hinzu, dass häufig keine oder nur grobe Dokumentation für die operativen Systeme und ihre Datenmodelle vorhanden ist. Das aber macht die genaue Analyse der zu importierenden Daten nur umso notwendiger. Die fachliche und technische Analyse der Datenbestände allein ist jedoch nicht ausreichend. Anschließend muss eine **aktive Datenmodellierung** auf DWH Seite erfolgen, wobei die Trennung von DWH und operativem Datenmodell zu berücksichtigen ist.

Abbildung 2: Das Datenmodell nach gesteuerter Erweiterung: die „Teilmodelle“ passen zusammen.

Wird z.B. eine bestimmte fachliche Aussage auf operativer Seite durch 3 Attribute kodiert, kann diese Information im DWH aus gutem Grund in ein Attribut zusammengezogen werden. Ein anderes Beispiel: für die Attributwerte „20“ und „21“ ist genau nicht ein Typ im DWH zu definieren, der die Werte „20“ oder „21“ annehmen kann. Die Werte aus dem operativen System dürfen nicht unüberlegt übernommen werden. Vielleicht sind manche Werte gar nicht mehr im Gebrauch und nur noch aus historischen Gründen im operativen System zu finden? Die Übernahme solcher „Junk“ Attribute würde das DWH Datenmodell mit unnötigen Werten belasten und die Komplexität unnötig in die Höhe treiben. Nur für die sinnvollen Werte sind Pendants im DWH zu definieren, die dann z.B. „bestellt“ und „geprüft“ lauten. Die Datenstrukturen der operativen Systeme haben sich bei vielen Anwendern aber schon so verinnerlicht, dass sie tatsächlich lieber operative Schlüssel wie „20“ und „21“ in den Berichten sehen wollen als Klartext. Dafür werden dann diese Zahlenwerte als zusätzliches Attribut mitgeführt (was im Zweifel bedeuten kann, für mehrere Quellsysteme unterschiedliche Wertemengen mitzuführen). Laut Kimball [KIMBALL] sind Dimensionstabellen genau der Ort, wo solche erweiterten Attribute abgelegt werden sollten. Die Wertemenge im DWH jedoch lautet „bestellt“ und „geprüft“, und deren Semantik sollte in einem zusätzlichen Kommentarfeld hinterlegt werden (denn nicht immer wird auch die nächste Änderung von den gleichen Personen durchgeführt). Bei einer nachfolgenden Erweiterung sind neue Entitäten im operativen System auf diese fachliche Bedeutung hin zu hinterfragen.

Bei der Datenmodellierung trifft man auch immer auf die Frage, ob man nun aus den Daten des operativen Systems eine Dimension oder ein Fakt machen sollte. Dies ist eine Entscheidung, die viel Erfahrung erfordert, denn die jeweiligen Konsequenzen sind nicht immer unmittelbar einsichtig. Die detaillierte Erörterung dieser Frage würde auch den Rahmen dieses Dokuments sprengen. Um das Datenmodell des DWH einfach zu halten, ist es auf jeden Fall empfehlenswert, **eine bestimmte fachliche Entität nur einmal im DWH abzubilden**: d.h. entweder als Dimension oder als Fakt, nicht aber als beides. Es ist jedoch ganz normal, dass bestimmte Entitäten in Berichten auf beiden Seiten auftreten können: z.B. kann der Umsatz pro Mitarbeiter ausgewertet werden (Mitarbeiter ist Dimension) oder auch die Anzahl der Mitarbeiter pro Niederlassung (Mitarbeiter ist Fakt). Um dies zu erreichen, aber das Datenmodell im DWH einfach zu halten, sollte eine entsprechende Umformung beim Übergang in den Data Mart durchgeführt werden.

Bei der Übernahme der Daten vom operativen System in das DWH (ETL) sind Transformationen durchzuführen, die das Datenmodell des operativen Systems auf das DWH abbilden. Das Datenmodell des DWH wird sich aber ändern, wenn neue Daten hinzukommen. Nicht nur durch eventuelle neue Entitäten an sich, sondern auch weil schon existierende Entitäten modifiziert werden müssen (z.B. werden Aufzählungstypen neue Werte bekommen, alte werden umbenannt etc.). Bei Verwendung des Push-Prinzips (s. Anhang) folgt daraus, dass die Transformationen im ETL möglichst auf Seiten des DWH und nicht schon im operativen System erfolgen müssen. Würden sie im operativen System durchgeführt, würde jede Erweiterung des DWH potenziell Änderungen in anderen operativen Systemen nach sich ziehen. Eine Ausnahme bilden fachliche Algorithmen der operativen Systeme, die bei einer Nachimplementierung im ETL eine starke Abhängigkeit des ETL vom operativen System zur Folge hätte: bei Änderungen der fachlichen Logik im operativen System müsste auch der Algorithmus im ETL geändert werden.

Eine weitere Konsequenz dieser Überlegung ist auch, dass die Daten im operativen System möglichst nah an der Quelle extrahiert werden sollten (s. [CASERTA]). Denn alle weiteren Stationen der Daten im operativen System,

bedeuten nur potenzielle Verunreinigung mit Fehlern und Transformationen, die – wie oben begründet – im ETL durchgeführt werden sollten. Des Weiteren sind sekundäre Systeme immer eher in Gefahr, abgelöst oder ganz abgeschafft zu werden, so dass man eventuell auf Sand baut, wenn man die Daten erst am Ende der Verarbeitungskette extrahiert.

5 Zusammenfassung

Das DWH kann nur durch eine gesteuerte inkrementelle Erweiterung wachsen. Dabei ist es notwendig Redundanz und Komplexität im Datenmodell gering zu halten, und zwar über die gesamte Lebenszeit des DWH hinweg. Daher ist ein Vorgehen notwendig, das diesen Notwendigkeiten Rechnung trägt. Bei jeder Erweiterung müssen also folgende Schritte durchgeführt werden:

- Analyse der betroffenen operativen Datenmodelle in Zusammenarbeit mit dem Wartungsteam und/oder dem Fachbereich
- Anschließend aktive Datenmodellierung des erweiterten DWH Datenmodells

Kurz: erst nachdenken, dann handeln in überschaubaren Schritten (Inkrementen).

Für die Datenmodellierung lassen sich folgende **Regeln** zur Hilfestellung angeben:

1. Die Datenmodellierung muss sich an den Anforderungen für die gewünschten neuen Reports ausrichten.
2. Redundanz ist zu vermeiden. Bei jeder Entität ist zu klären, ob diese nicht bereits im DWH vorhanden ist. Häufig führen neue Daten nämlich nur zu neuen Attributen einer bereits existierenden Entität.
3. Eine Entität sollte entweder nur als Dimension oder als Fakt in das DWH integriert werden, jedoch nicht beides. Sollte für Auswertungen beides benötigt werden, ist der Übergang zum Data Mart zu nutzen, um die Daten entsprechend aufzubereiten.
4. *Eine* fachliche Bedeutung sollte in *einem* Attribut (und nicht in mehreren) kodiert werden.
5. Ein besonderes Augenmerk muss auf Aufzählungstypen gerichtet werden, die meist als Dimensionen abgebildet werden: oft existiert bereits ein Attribut gleichen Typs im DWH ohne dass dies offensichtlich erkennbar ist. Häufig müssen die Wertebereiche untersucht und integriert werden. „Junk“ Werte sind auszufiltern.
6. Attribute sind in einer durchgängigen Form zu speichern, d.h. alle Datumsangaben in einem Format, alle Beträge in einem Format etc.
7. Die fachliche Bedeutung eines Werts sollte als Kommentarfeld im Datenmodell hinterlegt werden.
8. Änderungen an bereits im DWH existierenden Entitäten sind einer reinen Erweiterung um neue Entitäten vorzuziehen. Der Aufwand ist gut investiert, denn dadurch bleibt das DWH erweiterbar.
9. Konventionen zur Benennung von Attributen, Tabellen und anderen Entitäten sind unerlässlich.
10. Daten sollten möglichst nah dem Ort ihres Entstehens extrahiert werden.
11. Bei Verwendung des Push-Prinzips: Transformationen und Filter im ETL sind bevorzugt auf Seiten des DWH durchzuführen und nicht im operativen System.

6 Ausblick

Die in diesem Text beschriebene Methode geht davon aus, dass ein DWH nur in einem inkrementellen Ansatz aufgebaut werden kann. Die Herkulesaufgabe, alle Daten eines Unternehmens auswertbar aufzubereiten, ist nicht in einem Schritt durchführbar. Für den langfristigen Erfolg des inkrementellen Ansatzes ist das Datenmodell von entscheidender Bedeutung: es muss so aufgebaut sein, dass es leicht erweiterbar ist. Nur mit einem gut strukturierten Datenmodell sind zurückliegende Änderungen zu verstehen und zukünftige Erweiterungen mit darstellbarem Aufwand umsetzbar.
Die in diesem Text angegebenen Regeln basieren aus Erfahrungen des Autors in verschiedenen DWH Projekten und sind somit empirischer Natur. Es handelt sich um Richtlinien, die verfeinert und ergänzt werden müssen. Ein ebenso offener Punkt ist die Frage, was zu tun ist, wenn man bereits in der Komplexitätsfalle gefangen ist. Kann ein DWH System so umgebaut werden, dass es anschließend wieder inkrementell wachsen kann? Wie ist die Redundanz und Inkonsistenz des Datenmodells messbar, um zu entscheiden, an welchem Punkt man sich befindet. Diese Fragen müssen genauer untersucht und beantwortet werden.

7 Anhang: DWH Referenzarchitektur

Es ist Stand der Technik, Data Warehouse Systeme in einer dreistufigen Architektur aufzubauen. Diese drei Stufen werden im Folgenden mit *Staging*, *Core Data Warehouse* und *Data Mart* bezeichnet (s. Abbildung). Eine tiefgehende Beschreibung der Referenzarchitektur würde den Rahmen dieses Dokuments sprengen. Es wird daher der Fokus auf einen kurzen Überblick und auf die Einführung auf die für das Verständnis des Papiers notwendigen Begriffe gelegt.

Ein *Data Mart* ist eine Bezeichnung für eine Datenbank oder ein Datenbankschema, das speziell auf die Anforderungen der Präsentationen von Berichten an den Endbenutzer zugeschnitten ist. Um performant zu sein, sind die Daten im Data Mart so abgelegt, dass sie ideal zu den Bedürfnissen der jeweiligen Reports passen: sie z.B. denormalisiert, aggregiert und kumuliert, dass nur noch wenig Rechenaufwand notwendig ist, um die Daten anwendergerecht im Read-Only-Modus anzuzeigen. Da die verschienen Gruppen der Anwender unterschiedliche Berichte und Daten sehen wollen, stellt ein Data Mart i. allg. nur einen fachlich begrenzten Ausschnitt aus den gesamten Daten des Data Warehouse zur Verfügung.

Im *Core Data Warehouse* werden alle auswertbaren Daten gespeichert, die anschließend den verschiedenen Data Marts zur Verfügung gestellt werden. Es enthält also die Obermenge aller Daten aus allen Data Marts. Um diese extreme Datenmenge trotzdem beherrschbar zu halten, werden die Daten im Core Data Warehouse normalisiert gespeichert. Die Daten liegen dort entweder als *Dimension* oder als *Fakt* (Berichtswert) vor. Fakten sind diejenigen Kennzahlen oder Messgrößen, die nach den Dimensionen ausgewertet werden können, z.B. Umsatz (Fakt) pro Mitarbeiter (Dimension). Auch die Pflege der Dimensionen, die nicht immer durch Re-Import aus den operativen Systemen geschehen kann, erfolgt im Core Data Warehouse, bevor die dann geänderten Daten wieder an die Data Marts weitergegeben werden.

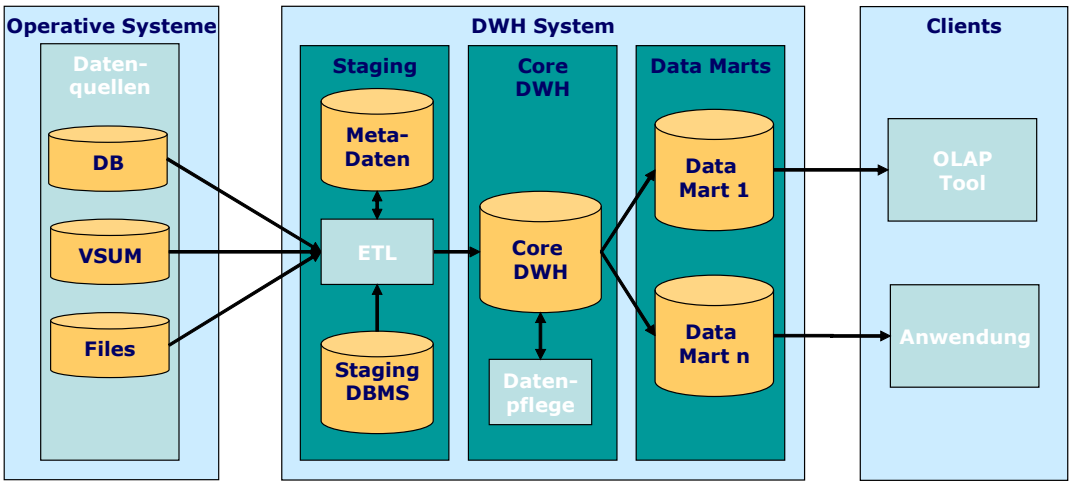

Abbildung 3: DWH Referenzarchitektur

Über den Bereich *Staging* gelangen die Daten aus den operativen Systemen in das Core Data Warehouse. Er enthält sowohl die Metadaten für den ETL-Prozess (also z.B. die Beschreibungen der Datenstrukturen und Verarbeitungsregeln) als auch die während des ETL-Vorgangs zwischengespeicherten Daten. Müssen z.B. Daten erst noch mit anderen Daten zusammengeführt werden, bevor sie gesäubert und normalisiert in das Core Data Warehouse gelangen, werden sie im Staging Bereich zwischengespeichert. Man unterscheidet zwischen Pull- und Push-Prinzip: beim Pull-Prinzip werden die Daten vom ETL-Tool aus den Datenbeständen des operativen Systems extrahiert, während beim Push-Prinzip das operative System die Daten extrahiert und dem ETL-Tool z.B. als Flat Files zur Verfügung stellt.

8 Literatur

[INMON] William H. Inmon, Bulding the Data Warehouse, John Wiley & Sons Inc, 2002

[KIMBALL] Ralph Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite, The Data Warehouse Lifecycle Toolkit, John Wiley & Sons Inc, 1998

[NUSS] Mark A. Nusselein, Inhaltliche Gestaltung eines Data Warehouse-Systems am Beispiel einer Hochschule, Bayrisches Staatsinstitut für Hochschulforschung und -planung (2003)

[CASERTA] Ralph Kimball and Joe Caserta, The Data Warehouse ETL Toolkit, John Wiley & Sons Inc, 2004