Das optimale SAS Data Warehouse

Alfred Kruse
Rahewinkel 34
22115 Hamburg
Alfred.3.Kruse@gmx.de

Zusammenfassung

Es gibt viele Möglichkeiten, ein Data Warehouse aufzubauen. Tatsächlich existiert jedoch nur eine optimale Architektur, die eine geringstmögliche Fehlerquote, minimierte Wartungsanforderungen, schnellstmögliche Batchverarbeitung und größtmöglichen Komfort der Anwender in sich vereint.
Der Weg dahin wird, unter Berücksichtigung der Eigenheiten von SAS, dargestellt.
Begonnen wird mit der Extraktion, bei der schon die ersten (organisatorischen) Fehlerquellen möglich sind. Es wird aufgezeigt wie und was aus den operativen Systemen extrahiert werden sollte. Im nächsten Schritt wird die (detaillierte) Struktur einer optimalen DWH-Architektur diskutiert und dargestellt. Batchverarbeitung, Wartungsbedingungen und Dokumentationsanforderungen in Zusammenhang mit dem SAS-System ergeben dann weitere Optimierungen für die Verzeichnisstruktur. Ergänzt wird dies durch Beispiele aus dem Betrieb eines SAS-DWH, zusammen mit Erfahrungen aus mehreren verschiedenen DWH-Architektur.

Schlüsselwörter: Data Warehouse, BI, Business Intelligence, Datenaufbereitung, BI Architektur

1 Theorie des Data Warehouse

Der Begriff Business Intelligence (BI) bezeichnet den Prozeß, fragmentierte Daten in handlungsgerichtetes Wissen umzuformen. Dies ist nichts anderes als die technische Umsetzung der menschlichen Fähigkeit, aus einer Fülle von Wahrnehmungen (Daten) das Relevante zu selektieren, zu reflektieren und daraus eine Aktion abzuleiten. Von diesem Gesichtspunkt aus ist die BI für jedes Unternehmen unverzichtbar.
Als Basis der Business Intelligence hat sich das Data Warehouse (DWH) durchgesetzt. Hier werden unterschiedliche operative Anwendungen technisch integriert, semantisch vereinheitlicht und die entstandenen Daten organisationsweit für Auswertungen zur Verfügung gestellt. Als ein an der Unternehmenssteuerung beteiligtes Informationssystem (dispositives bzw. analytisches Informationssystem) gelten für das DWH andere Regeln als für operative Anwendungen [1]:

Operative Anwendung

- unterstützt die Arbeit des Sachbearbeiters
- aktuelle & detaillierte Daten
- Online-Aktualität

- viele Transaktionen kleiner Datenmengen
- voraussagbare kontinuierliche Systemlast

Dispositive Anwendung
- unterstützt die Arbeit des Managements
- aufbereitete aktuelle & historische Daten, detailliert und aggregiert
- Batch-Aktualität
- wenige Analysen großer Datenmengen
- bedarfsabhängig schwankende Systemlast

Aufgrund dieser wesentlichen Unterschiede zwischen operativen und dispositiven Anwendungen treten folgende Probleme bei einer direkten Verbindung beider Systeme auf [2]:

- **Mangelnde Integration**: Die Erreichbarkeit von Daten unterschiedlicher operativer Datenquellen ist nur mit hohem Synchronisationsaufwand zu realisieren. Bedingt durch die redundante und verstreute Datenhaltung treten Dateninkonsistenzen auf. Zeitreihenanalysen und vergangenheitsbezogene Fragestellungen können operativ nur sehr schwer realisiert werden.
- **Mangelnde Antwortzeiten**: Operative Systeme sind für die Ausführung lokaler Update-Operationen optimiert - große verdichtete Datenvolumina können nur mit langen Antwortzeiten abgefragt werden.
- **Mangelnde Lastbalancierung**: Benutzen operative und dispositive Anwendungen dieselben Datenbanken, so muß die Rechenkapazität zwischen ihnen aufgeteilt werden. Da die Lastverteilung bei analytischen Informationssystemen grundsätzlich ungleichmäßig und nicht vorhersehbar ist, können keine komplexen Abfragen an die Datenbanken gestellt werden, ohne die Leistung des operativen Systems erheblich zu verringern.

Zusammenfassend ergibt sich die Konsequenz, zwischen der operativen Anwendung und dem analytischem Auskunftssystem eine Zwischenschicht einzuführen. Diese erhält die Daten der operativen Anwendungen periodisch, bereitet sie auf und stellt sie den unterschiedlichsten analytischen Anwendungen einheitlich zur Verfügung ("single point of truth").
Die aus dem operativen Bereich benötigten Daten werden zunächst in einem ETL-Bereich extrahiert, transformiert (pausibilisiert) und in die zentrale Datenbasis geladen. Hier werden sie in verschiedenen Stufen integriert, aggregiert und denormalisiert, in einem letzten Prozeß werden die so vorbereiteten Daten bereichspezifisch unterschiedlichsten Anwendergruppen zur Verfügung gestellt (Data Marts).
Da die zentrale Datenbasis aus rein physikalischen Gründen nicht vollständig für jeden Tag vorgehalten werden kann, werden nach jeder Belieferung die vorher bestehenden Daten mit den aktuellen überschrieben (Aging-Verfahren). Um keine Informationen zu verlieren, ist der Aufbau eines Historisierungsbereiches notwendig.

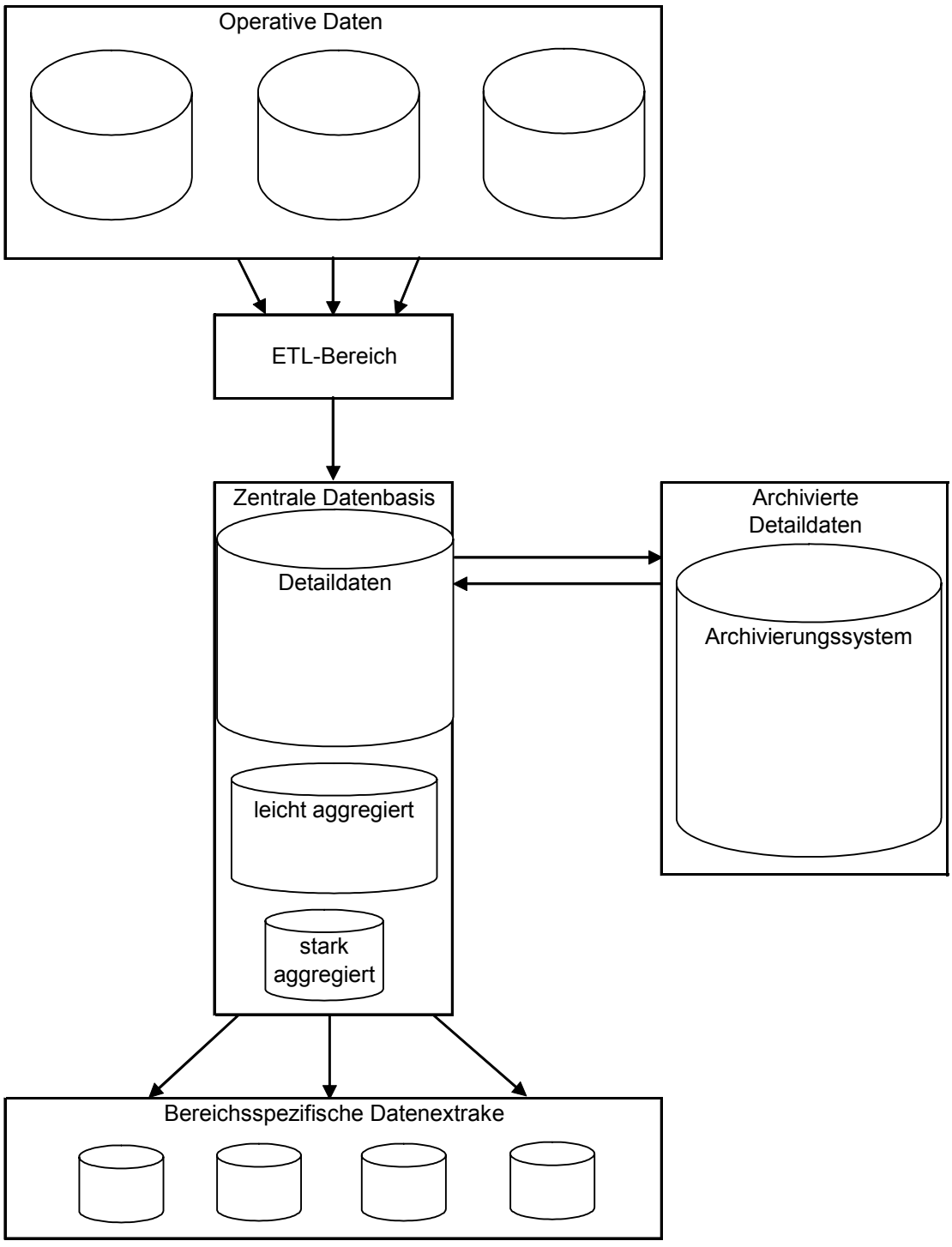


Abbildung 1: Allgemeingültige Referenzarchitektur eines DWH [3] [4]

Für die Umsetzung dieser allgemeingültigen Referenzarchitektur eines Data Warehouse existieren drei verschiedene Architekturalternativen [5]:

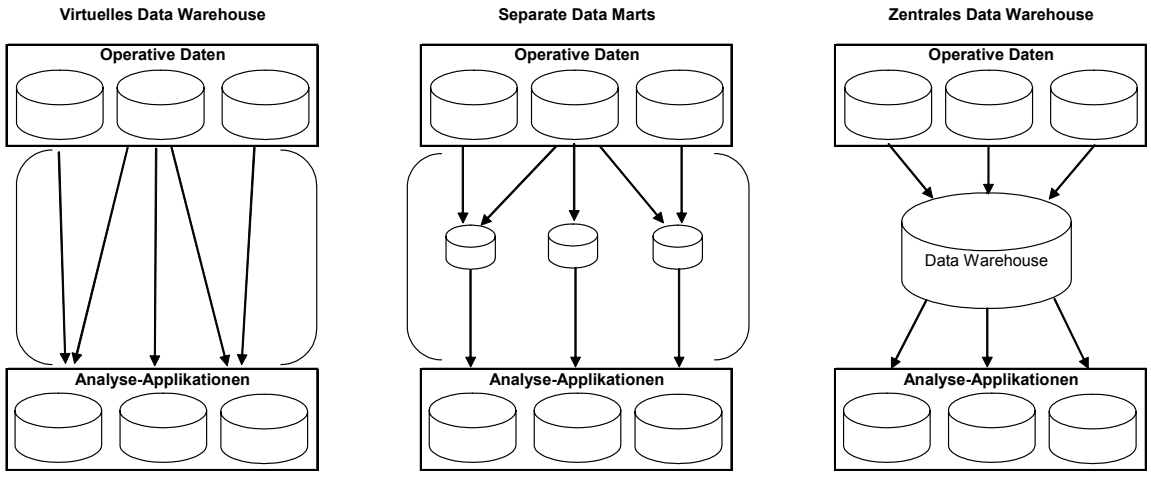


Abbildung 2: DWH-Typen [5]

Als **virtuelles Data Warehouse** werden Architekturen bezeichnet, die für den Benutzer zwar als eigenständiges System erscheinen, auf eine redundante Speicherung der operativen Daten jedoch verzichten. Der Vorteil dieser Systemalternative besteht in der schnellen und kostengünstigen Implementierung, da auf die Anschaffung einer neuen technischen Infrastruktur verzichtet werden kann. Nachteile bestehen jedoch in der zusätzlichen Belastung der operativen Anwendungen und mangelhaften Antwortzeiten, da vorberechnete Aggregate und Caching strukturell entfallen müssen. Weiter sind keine Zeitreihenanalysen machbar (es stehen nur die aktuellen Daten des operativen Systems zur Verfügung), d.h. die Aussagefähigkeit ist extrem eingeschränkt.
Diese Probleme treten nicht auf, wenn **separate Data Marts** als applikationsspezifische Datenspeicher zwischengeschaltet werden. Diese Data Marts sind von geringer Komplexität und speziell auf die von ihnen versorgte Applikation abgestimmt, also ebenfalls relativ schnell und kostengünstig implementiert. Da aber die Entwicklung dieser Data Marts dezentral verläuft, werden damit unzusammenhängende Insellösungen geschaffen, die einen hohen Wartungsaufwand zwangsläufig nach sich ziehen. Aber auch falls hier eine unternehmensweite Sicht geschaffen wird, weist der Ansatz separater Data Marts weitere Nachteile auf. Aus der dezentralen Organisation resultiert eine höhere Netzbelastung, die Berechnung übergreifender Aggregate ist strukturell kaum möglich, redundante Speicherung identischer Datenbestände zwangsläufig.

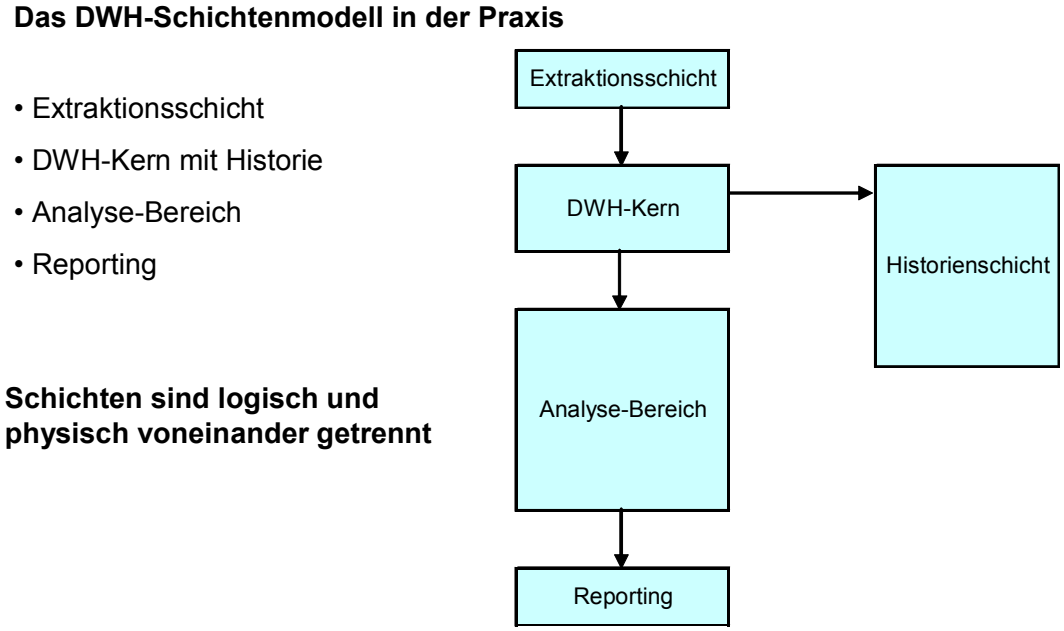
In der Theorie wird dann die Vereinigung aller Data Marts als Data Warehouse verstanden (DWH Bus Architecture). [6] Beseitigt man die Redundanzen der Data Mart-Architektur, gelangt man zur Konzeption des **zentralen Data Warehouse**. Hier wird ein zentrales System, streng von den operativen Anwendungen getrennt, mit Daten gefüllt, die wiederum in ein einheitliches Unternehmensmodell überführt werden.

Alle Benutzer des DWH greifen mit den verschiedensten Zugriffswerkzeugen auf eine einheitliche Datenbasis zu.
Derartige unternehmensweite Systeme erfordern aufgrund ihrer Komplexität und den hohen Anforderungen an die Datenqualität einen langwierigen Implementierungsprozeß, i.e. einen hohen Kosten- und Zeitaufwand. Dies verschärft sich noch, wenn man berücksichtigt, daß bei dieser Lösung weitere Hard- und Software-Kosten anfallen, die weder beim virtuellem DWH noch bei der Architektur der separaten Data Marts benötigt werden.
Demgegenüber stehen aber die Vorteile einer solchen zentralen Architektur. Zum einen erzwingt dieser Ansatz notwendigerweise die Entwicklung eines gemeinsamen konsistenten Unternehmensdatenmodells. Dies hat zur Folge, daß nur bei diesem Ansatz das Prinzip der einheitlichen Datensicht gewährleistet ist. Technisch folgen aus einer homogenen Architektur und Systemlandschaft außerdem deutlich geringere Betriebskosten, insbesondere bei Wartung und Administration. Da kein Netzwerk von verflochtenen Datenbanken bei analytischen Abfragen aktiviert werden muß und ein effizientes Caching-Verfahren im zentralen Data Warehouse zwangsläufig implementiert wird, sind Antwortzeiten und Netzbelastung auch deutlichst geringer als bei den anderen beiden Alternativen. Ein weiterer Vorteil besteht in der zentralen Dokumentation, die die Klammer zwischen dispositiven und operativen Anwendungen darstellt.
Ein virtuelles Data Warehouse als auch die Architektur separater Data Marts lassen sich also kurzfristig und kostengünstig implementieren, ziehen aber hohe Folgekosten im Betrieb nach sich. Ein zentrales Data Warehouse hat hohe Implementierungs-, aber geringere Folgekosten. Diese Kostenanalyse verschärft sich, wenn man den Betrieb dieser Architekturen über die IT hinweg betrachtet. Bei jeder der drei Architekturmodelle soll eine einheitliche Datensicht implementiert werden. Dies bedingt hohe Abstimmungsaufwände nicht nur bei den Fachverantwortlichen, sondern ebenfalls bei der technischen Umsetzung. Da für die virtuelle und Data Mart–Architektur jede Analyseanwendung dezentral versorgt wird, wird jedesmal auch (mindestens) ein Ansprechpartner benötigt. Da sich diese, basierend auf der technischen Umsetzung, personell voneinander unterscheiden, ist der pesonelle und zeitliche Aufwand unverhältnismäßig hoch. Im Gegensatz dazu sind bei der Architektur des zentralen DWH keine aufwendigen technischen Abstimmungen notwendig, für Betrieb als auch Fachanwender existiert ein einziger wohldefinierter Ansprechpartner.

2 Architektur

Nach meinen Erfahrungen hat sich für das DWH das in der Theorie beschriebene Schichtenmodell auch im praktischem Betrieb als optimal herauskristallisiert. Das DWH ist also architektonisch folgendermaßen aufgebaut :

- Extraktionsschicht
- DWH-Kern
- Analyse-Bereich
- Reporting



Die Extraktionsschicht besteht dabei aus den aus der operativen Anwendung extrahierten und bereits nach SAS konvertierten Daten, der Analyse-Bereich aus Data Marts, die für das Reporting oder AdHoc-Abfragen der Fachabteilung speziell designt sind. Der DWH-Kern ergibt sich dann zwangsläufig als Summe aller Bestände auf dem Weg dorthin. Wie wir weiter unten sehen werden, ist hier tatsächlich das Herz des DWH angesiedelt. Diese vier Schichten sind nicht nur logisch, sondern auch physisch getrennt, d.h. ein Programm lässt sich immer genau einer Schicht zuordnen, es gibt keine Überläufer.
Als sehr effizient hat sich das Prinzip "Ein Programm – ein Verzeichnis" herausgestellt. Statt also über den ShareServer in (unübersichtliche) Großverzeichnisse hineinzuschreiben, schreibt jedes Programm in sein eigenes Verzeichnis. Nebenbei beschleunigt dies noch den Batch, denn nach meinen Erfahrungen benötigt der ShareServer nicht wenig Prozessorkapazitäten und behindert die I/O deutlich.

Der Programmname sollte dabei aus 6 Zeichen bestehen, 3 Buchstaben und drei Zahlen. Die ersten drei Buchstaben beschreiben das Sachgebiet, die nächsten drei Zahlen sind

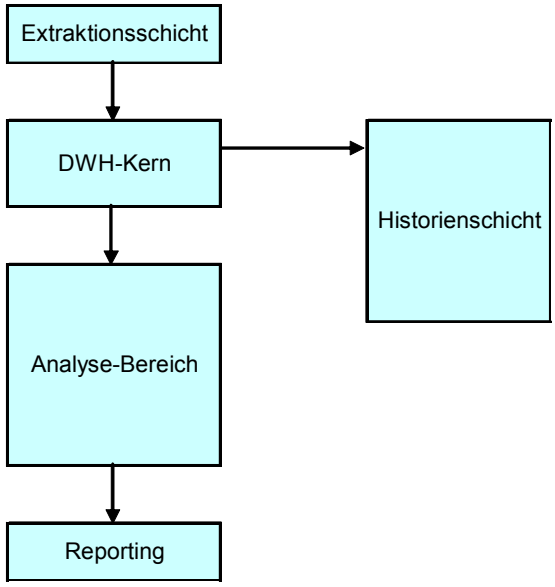
Laufnummern. Behält man über das Schichtenmodell hinweg die Zahlen bei, ist bei der Nennung eines Programmnamens Stellung im Batch ebenso klar wie der fachliche Inhalt. Diese Nomenklatur stammt noch aus Großrechnerzeiten, bei denen ein Job durch 8 Zeichen begrenzt war. Nichtsdestotrotz hat sie sich bewährt und bietet mit minimaler Redundanz ein Maximum an Information.

Weiter ist mit diesem Prinzip der Batch eines DWH selber einfachst aufzusetzen : Jedes Programm läuft zum frühestmöglichen Zeitpunkt, d.h. wenn alle benötigten Dateien / Verzeichnisse erzeugt sind. Natürlich sind die Autoexecs in dieser Architektur komplizierter, jedes Verzeichnis muß hier einzeln angegeben werden. Aber gerade dadurch ist zwangsläufig dokumentiert, welche Daten jedes Programm benutzt. Ganz davon abgesehen sind damit technisch fehlende Abhängigkeiten unmöglich gemacht worden.

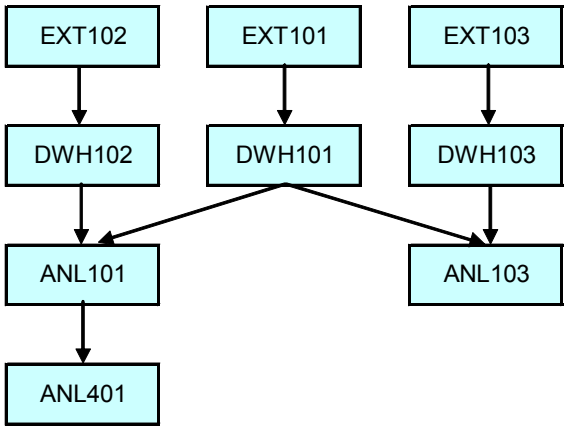
Programmtechnische Umsetzung

- 1 Programm, 1 Verzeichnis
- Programmname = Verzeichnisname
- Programmname TTTzzz
- Dateinamen sprechend
- Variablennamen sprechend

- **Batch ergibt sich kanonisch**
- **Orientierung optimal**
- **Technische Dokumentation**



Beispiel : SAP/ZGP



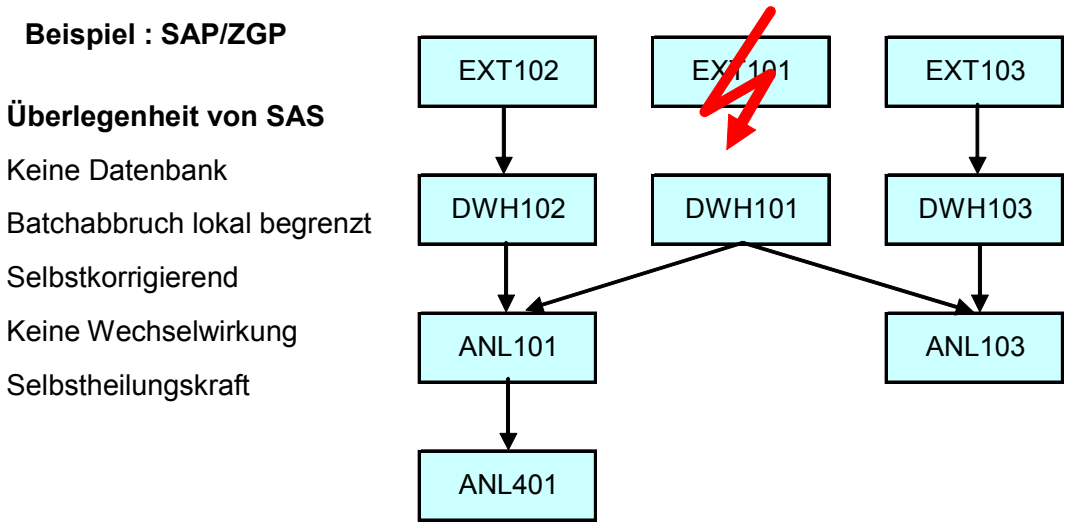
• **Extraktionsschicht**
EXT101 Partnerstammdaten
EXT102 Partnerrollen (VDGPO)
EXT103 Partnerbeziehungen

• **DWH-Kern**
DWH101 Partnerstammdaten
DWH102 Partnerrollen
DWH103 Partnerbeziehungen

• **Analyse-Bereich**
ANL101 Rollen / Stammdaten
ANL103 Beziehungen / Stammdaten
ANL401 Hauptdarlehensnehmer CML

An diesem Beispiel sieht man sehr deutlich, daß, obwohl die Nomenklatur nicht eineindeutig einen Bestand bzw. Programmnamen definiert, man mittels dieser Methodik eine übersichtliche intrinsische Dokumentation des DWH geschaffen hat.
Eine solche Nomenklatur sollte aber nur für Verzeichnisse aufgesetzt werden. Dateien und Variablen hingegen sollten sprechende Bezeichnungen bekommen. Damit wird verhindert, daß sich bei den DWH-Entwicklern Expertenwissen aufbaut. Zusätzlich dazu habe ich festgestellt, daß damit die Programmierung auch einfacher und intuitiver wird. Natürlich kann man beispielsweise Datumsfelder mit dem Präfix „DT“ abkürzen, der Rest des Namens sollte jedoch sprechend sein (Beispiel : das Vertragsdatum eines Kredites sollte danach im DWH als DT_Vertrag codiert sein). Ebenso können Prozentwerte mit dem Präfix PZT gekennzeichnet sein. Vorsehen sollte man sich jedoch vor der Häufung solcher Variablen-Gruppierungen, oftmals ist hier weniger mehr.

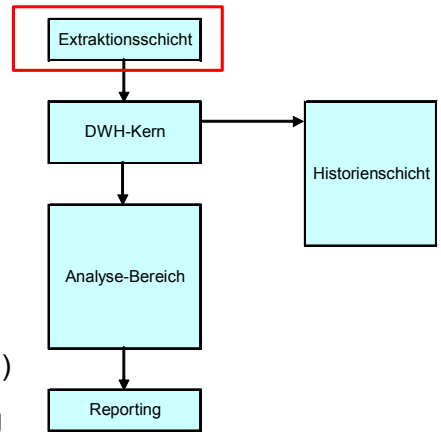
An dieser Stelle wird auch die Überlegenheit von SAS gegenüber anderen DWH-Systemen deutlich. SAS ist keine Datenbank und von daher bereits technisch deutlich stabiler als Datenbank-Systeme. Zusammen mit der oben dargestellten Architektur ist aber auch eine fachlich wesentlich höhere Stabilität gegeben. Nehmen wir im obigem Beispiel einmal einen Abbruch beim Einlesen der Stammdaten an, d.h. das Programm EXT101 liefert keine aktuellen Daten. Wir müssen uns in SAS keine Gedanken um technische Konsistenzen machen, Partnerrollen und –beziehungen können mit aktuellen Informationen erstellt werden. Ab DWH-Kern läuft der gesamte Batch normal weiter, nur steht für einen Tag im DWH nicht die aktuelle, sondern die gestrige Partnerstamm-Information zur Verfügung. Am nächsten Tag kommen auch diese Daten wieder aktuell, durch den Merge mit dem bestehendem Bestand werden alle Daten aktualisiert – das DWH in dieser Architektur heilt sich selbst. Auch hat das Batchproblem keine Auswirkungen auf andere Bestände, es ist architektonisch als auch technisch lokal begrenzt.



Extraktion

Details Extraktionsschicht

- **nah am operativen System**
- **nur Originaldaten, keine Delta-Versorgung**
- **im operativen System nachvollziehbar**
- **das operative System nachbildend**
- **SAP-Module = Originaldaten** (CML-Kapitalien)
- **keine Sorts, keine Joins, keine Verarbeitung**
- **Dokumentation Variablennamen operatives System → DWH**
- **keine Konsistenzprüfungen o.ä.**



Die Extraktionsschicht besteht aus den aus der operativen Anwendung extrahierten und bereits nach SAS konvertierten Daten. Ich empfehle hier nichts sonstiges mit den selektierten Daten zu machen, noch nicht einmal eine Sortierung. Dies hat nach meiner Erfahrung den Vorteil, daß im Fehlerfall relativ schnell eine präzise Analyse möglich ist. Jede Modifikation der Daten erschwert dies deutlich.
Aus den operativen Systemen sollten nur die Originaldaten, die auch für den operativen Betrieb nötig sind, möglichst 1:1 übernommen werden. Keinesfalls sollte man versuchen, derartige (technische) Daten in ein anderes technisches Konzept zu zwängen oder sie bereits bei der Extraktion fachlich zu modifizieren. Erfahrungsgemäß führt dies bei der Implmentierung als auch im späterem Batch-Betrieb zu nicht vorhersehbaren technischen und fachlichen Problemstellungen. Aus diesem Grund empfehle ich, prinzipiell nur Originaldaten der operativen Systeme in das DWH zu extrahieren. "Originaldaten" bedeutet nicht, daß diese Daten auch original so in dieser Form in einer SAP-Tabelle vorhanden sein müssen. So werden die Kapitalien eines Kredites in SAP/CML bei der Online-Anzeige über ein Modul on the fly errechnet – trotzdem sind dies Originaldaten.

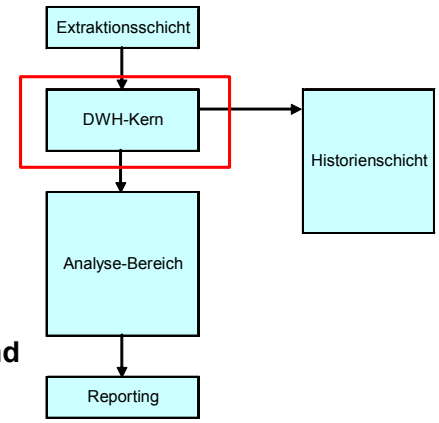
Man sollte auch nicht den Fehler machen, derartige Berechnungsmodule innerhalb des DWH nachzuprogrammieren. Nach meinen Erfahrungen ergeben sich zwischen der Nachprogrammierung und dem operativem Original immer Differenzen, auch die Synchronizität ist nicht gewährleistet. In einem solchen Fall ist es besser, mit einer geringeren Aktualität zu leben und ein solches Modul etwa nur einmal pro Woche Daten liefern zu lassen, statt täglich unzuverlässige Daten zu erzeugen.
Es ist ebenfalls nicht zwingend notwendig, die in SAP (oder einem sonstigem transaktionsorientiertem operativen System) vorhandenen Tabellen unbedingt in genau dieser Form zu extrahieren. Es kann sehr wohl Sinn machen, Stammdaten, die in verschiedenen Tabellen abgelegt sind, bereits bei der Extraktion zusammenzuführen. Dies gilt al-

lerdings genau und nur bei 1:1-Zusammenhängen, Denormalisierungen und fachliche Joins sollte man hier unterlassen.
Dieses Vorgehen hat auch den weiteren Vorteil, Verantwortlichkeiten präzise festzulegen. Das operative System stellt genau und nur die technischen Daten bereit – erst das DWH als dispositive Datendrehscheibe formt diese technische Sicht zu fachlichen Informationen um und ist damit verantwortlich für die Erstellung der korrekten fachlichen Zusammenhänge.

DWH-Kern

Details DWH-Kern

- **technische Abbildung der operativen Welt**
- **Mittler operatives / dispositives System**
- **nur normalisierte Strukturen**
- **im operativen System nachvollziehbar**
- **das operative System strukturell nachbildend**
- **„Gedächtnis des DWH“**
- **keine Verarbeitung**
- **Variablennamen entsprechen der dispositiven Welt**
- **Konsistenzprüfungen und Plausibilisierungen, Breakpoints**
- **Indizierung für Online-Anzeige (Suche)**



Im DWH-Kern werden die (normalisierten) Daten aus der Extraktionsschicht ebenfalls als normalisierte Bestände im DWH abgelegt. Im Gegensatz zur Extraktionsschicht ist der DWH-Kern persistent, einmal in das DWH überführte Daten bleiben bis zur Zerfallsgrenze erhalten. Hier ist auch der Ort der ersten Plausibilisierung von Daten : Beim Übergang von der Extraktionsschicht zum DWH-Kern sollte man prüfen, ob alle Primärschlüssel gefüllt und innerhalb des richtigen Zahlenkreises liegen.
Ebenfalls ist empfehlenswert, beim Einlesen in das DWH einen Zeitstempel zu vergeben. Hier macht es auch Sinn, das operative System, aus dem die Daten stammen, als Id in den Primärschlüssel zu integrieren und im Falle von mandantenfähigen DWH einen logischen, vom operativem System unabhängigen Mandanten zu vergeben.

Im DWH-Kern sollte nichts gelöscht werden. Der tagesaktuelle Bestand aus der Extraktionsschicht wird an den bestehenden herangemerged, nicht mehr gelieferte Datensätze erhalten ein Kennzeichen "logisch gelöscht", das sogenannte Reorganisationskennzeichen. Mittels dieser Methodik erhält das DWH eine Art "Gedächtnis", es vergisst einmal gelieferte Datensätze erst, wenn die Zerfallsgrenze erreicht ist.
Die Indizierung einer Tabelle aus dem DWH-Kern sollte nicht starr nach dem Primärschlüssel erfolgen. Im Gegenteil, oftmals ist dies kontraproduktiv, da platzfressend.

Besser ist es hier, nach den erfahrungsgemäß in der Online-Suche benötigten Variablen zu indizieren, während sich die davon unabhängige Sortierung immer nach den Batch-Erfordernissen richten sollte.
Ebenfalls zu dieser Schicht gehört die Historisierung. Diese sollte aber, um den Batch zu beschleunigen, nicht im gleichen Programm wie der Aufbau des DWH-Kerns stattfinden. Für die Historisierung bietet SAS mit der PROC COMPARE eine besonders elegante Möglichkeit der Implementierung. Diese Form der Programmierung hat den Vorteil, daß sie extrem stabil ist, es findet immer ein Abgleich mit den letzten, der Historisierungschicht als aktuell gemeldeten Daten statt. Daher ist auch kein Vorhalten einer Gestern-Datei notwendig.

Der DWH-Kern ist bis zu einem gewissem Grad "nur" eine technische Abbildung der operativen Welt. Erst beim Übergang in den Analyse-Bereich werden Kennzahlen gebildet und fachliche Sichten erzeugt. Dies ist natürlich eine extreme Beschränkung für den DWH-Kern, hat aber den Vorzug, daß man an jedem Punkt eines DWH sich der Herkunft und Qualität der Daten ziemlich sicher ist. Allerdings macht es auch keinen Sinn, aufgrund dieses Prinzips die gleiche Kennzahl mehrfach zu berechnen. Auch die Konsolidierung mehrerer (mehr oder weniger) fachlich gleichartiger Systeme ist mit diesem Prinzip nicht wirklich zu implementieren – es fehlt eine Art Zwischenbereich, der einerseits nicht mehr rein technisch getrieben, andererseits noch nicht wirklich aus fachlichen Anforderungen heraus benötigt wird.

Analyse-Bereich & Reporting

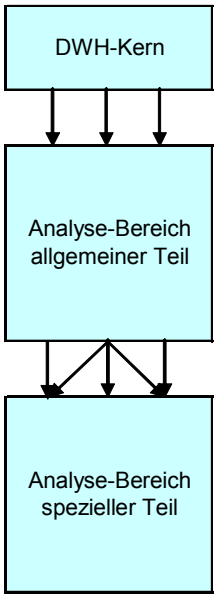
Details Analyse-Bereich

- **dispositive Darstellung**
- **nur noch fachlich mit operativem System vergleichbar**
- **Bildung von Kennzahlen**

Probleme

- **Batcherfordernisse**
- **Konsolidierung mehrerer operativer Systeme**
- **Einmalige Berechnung von Kennzahlen**

Differenzierung des Analyse-Bereichs in einen allgemeinen und einen speziellen Teil



Die Lösung der Probleme besteht in der Differenzierung des Analyse-Bereichs in eine allgemeinen und einen speziellen Teil. So werden beispielsweise in SAP/ZGP die Stammdaten eines Partners und dessen verschiedene Funktionen innerhalb von SAP/CML (Rollen) in verschiedenen Tabellen gehalten. Im Kern-DWH bleibt diese Trennung erhalten. Für Reports und sonstige Auswertungen benötigt man aber immer die Partnerstammdaten des Partners für eine bestimmte Rolle, oftmals ist es sogar notwendig, verschiedenste Rollen zusammen mit den unterschiedlichsten Stammdaten auf die Kreditebene hochzudrehen, d.h. als einen Datensatz anzuzeigen. Für solche Auswertungen ist ein Merge von Stammdaten und Rollen notwendig, den man aber wiederum nicht für jeden Report wiederholen will. Also wird dieser Merge einmal als allgemeiner Analyse-Bestand abgelegt, alle weiteren Bestände, die diese Informationen benötigen, referenzieren auf diesen Bestand.

Diese allgemeinen Bestände des Analyse-Bereichs sind zunächst batchgetrieben, d.h. sie werden durch die Notwendigkeit eines Batch-Tunings erzeugt. Oftmals stellt sich ein solcher Bestand jedoch für die Fachabteilung als (einzige) Möglichkeit dar, seltene Ad-Hoc-Abfragen direkt auf die allgemeinen Analyse-Bestände laufen zu lassen. Hier werden ebenfalls die allgemeingültigen Kennzahlen berechnet.

Details Analyse-Bereich : allgemeiner Teil

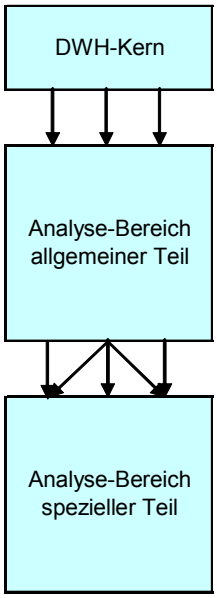
- **oftmals batchgetrieben**
- **Kennzahlen-System**
- **Standard-Denormalisierungen**
- **komplette Darstellung einzelner Fachentitäten**
- **inklusive erledigter Datensätze**

Beispiel SAP/ZGP

Hauptdarlehenspartner

Merge an Kredite, Objekte, Grundbuchdaten, …

Oftmals die einzige Möglichkeit für Fachanwender, komplexe AdHoc-Abfragen durchzuführen



Weniger seltene Abfragen der Fachabteilungen und abteilungsspezifische Fragestellungen werden im speziellem Teil des Analyse-Bereiches vorbereitet. Hier werden Tabellen genau so erstellt, wie der Fachbereich es sich wünscht. Diese Vorgehensweise hat neben der wesentlichen Vereinfachung von Abfragen für die Fachanwender zusätzlich den Vorteil, daß die Bestände extrem klein (von der Anzahl der Observations ebenso wie von der Variablenzahl) sind.

Details Analyse-Bereich : spezieller Teil

- **immer fachgetrieben**
- **spezielle fachliche Sicht**
- **nicht notwendig „vollständig“**

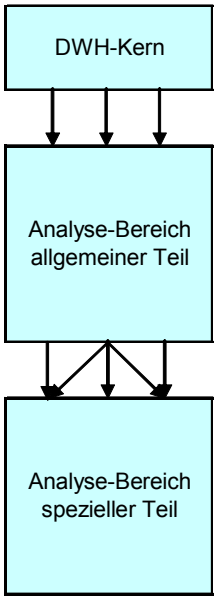
Beispiel SAP/ZGP

Alle Hauptdarlehenspartner mit einer Konzernzugehörigkeit

Darstellung des Konzern-Engagements

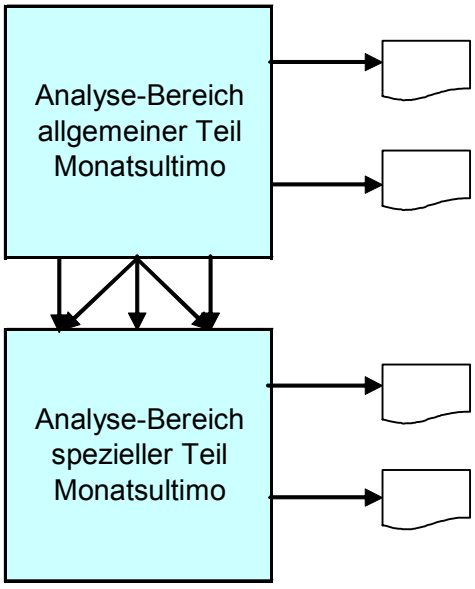
Speziell für individuelle fachliche Sichten designt

Oftmals nur für einen einzigen Zweck nutzbar



Snapshot-Historisierung

- **Reporting nur auf ANL-Beständen**
- **M/Q/J-Sicherungen auf Platte**
- **Reproduzierbarkeit des Ultimos**
- **Einfacher Zugriff auch für FA**

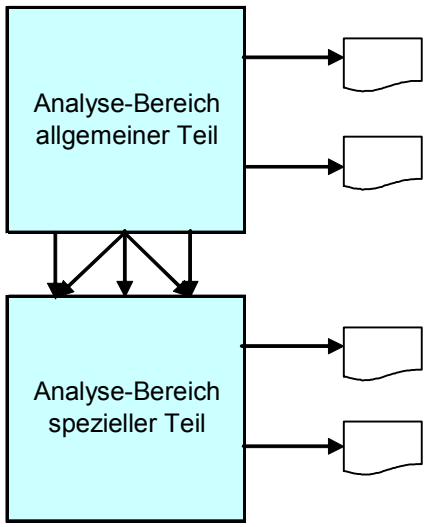


Mit diesen speziell für den einzelnen Fachbereich erzeugten Tabellen bietet sich die Möglichkeit, zusätzliche eingefrorene Stände (Monatsultimo, Quartal, Jahresbestand) zu halten, ohne das DWH extrem aufzublähen. Diese "Snapshot-Historisierung" ist von der Historisierung im DWH-Kern wesentlich verschieden : Es wird eine fachliche Sicht zu einem Stichtag eingefroren und nicht jede Veränderung zu jedem Stichtag gesondert protokolliert.

Das darauf folgende Reporting ist dann relativ trivial : Basierend auf speziellen Analyse-Beständen erzeugt eine einfache PROC REPORT eine Liste bzw. eine PROC EXPORT ein Daten-File. Wichtig ist hierbei, daß ein Report keinen Datensammler enthält (das macht bereits der Analyse-Bestand) und daß keine neuen Kennzahlen errechnet werden. Es empfiehlt sich hier, lieber einen Analyse-Bestand mehr zu bauen als dieses Konzept zu durchbrechen.

Reporting : möglichst einfach

- **ZUGRIFF NUR AUF ANALYSE-BEREICH**
- **keine temporäre Datensammlung**
- **keine neuen Kennzahlen**
- **Besser : Analyse-Bestand / Report**
- **Report = proc report**
- **Nachvollziehbar durch ANL-Bestand**
- **keine Probleme / Fehler im Report**
- **effektiv : uninteressant ☺**



3 Security

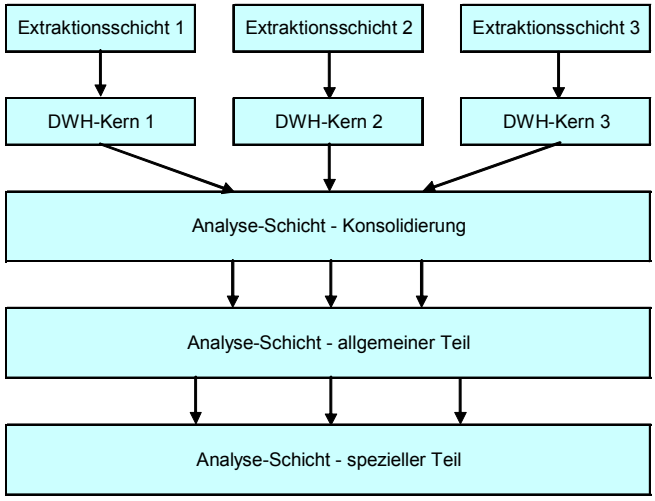
Für die Fachabteilung ist es optimal, genau und nur auf die Analyse-Bestände zuzugreifen. Es macht sogar Sinn, aus Security-Gründen den Zugriff der Fachabteilung darauf zu beschränken und einzelne Bestände auch selektiv nur einzelnen Fachabteilungen zugänglich zu machen. Falls sich beispielsweise innerhalb der Kundendaten sensible Informationen befinden (und diese unbedingt im DWH abgelegt werden müssen), kann man durch diese Methode den generellen Zugriff auf die Daten des DWH grundsätzlich jeder Fachabteilung gestatten, den Zugriff auf sensible Kundendaten aber auf diejenigen beschränken, die es unbedingt benötigen.

Dies gilt nicht für die DWH-Entwickler, diese benötigen den unbeschränkten Zugriff auf alle DWH-Informationen. Es macht auch meiner Erfahrung nach keinen Sinn, Entwickler hier auf Testsysteme zu beschränken. Um die Komplexität der Produktion nachzubilden, muß das Testsystem praktisch der Produktion entsprechen, von jedem möglichem Datensatz müssen mindesten 5 Test-Datensätze vorhanden sein, um alle Programmierungsfehler abfangen zu können. Bei nur 50 verschiedenen relevanten Variablen ergeben sich so 5exp50 Varianten – von denen wahrscheinlich 90 % im produktivem System garnicht vorkommen, bei einer Beschränkung der DWH-Entwickler auf ein Testsystem aber unumgänglich sind. Neben den erhöhten Kosten wird auch der Zugriff der Fachabteilung auf unnötig kompliziert, da ihr von den DWh-Entwicklern keine Analysen abgenommen werden können. Ganz davon abgesehen, daß ein Fachtest innerhalb dieser Testumgebung auch sehr aufwendig ist und eine gespiegelte operative Umgebung voraussetzt. Wesentlich effektiver ist es hier, DWH-Entwicklern Zugriff auf produktive Daten zu erlauben. Es werden dann exakt die Probleme, die in der Produktion vorliegen, behoben und keine theoretisch möglichen abgefangen. Auch kann die Fachabteilung im produktivem operativem System die Korrektheit einer DWH-Auswertung überprüfen, mehrfache Testiterationen werden vermieden. Die DWH-Entwicklung selber wird wesentlich vereinfacht (und somit die Kosten reduziert), da man bei der Änderung eines Programms kein komplexes Umfeld schaffen, sondern nur auf die für dieses Programm nötigen Bestände aus dem produktivem DWH zugreifen muß. Auch technisch komplexere Analysen können aus der Fachabteilung an das DWH-Team delegiert werden und werden somit schneller und kostengünstigerer.

4 Konsolidierung mehrerer operativer Datenbasen

Mehrere operative Systeme

- **anwendungsnahe DWH-Kerne**
- **Konsolidierung erst im ANL-Bereich**
- **evtl. Konsolidierungsschicht**



Aus der oben dargestellten Architektur ergibt sich das Vorgehen bei mehreren operativen Systemen kanonisch : Für jedes System werden getrennte DWH-Kerne erzeugt, die erst im Analyse-Bereich zusammengeführt werden. Eventuell ist es zweckmäßig, diese Konsolidierung als spezielle Schicht zusätzlich einzuführen, um die notwendigen Fachlichkeiten innerhalb eines Programm-Schemas zu bündeln.

5 Schlußbemerkungen

Die Programmierung innerhalb des DWH sollte so einfach wie möglich sein, je einfacher, desto besser. Insbesondere der Selbstverwirklichung von Programmierern muß hier frühzeitig ein Riegel vorgeschoben werden. Auch vor extremer Makroprogrammierung und generischen Programmen sollte man sich hüten, meiner Erfahrung nach ist individuelles Coding als auch Cut&Pace wesentlich effektiver, kostengünstiger und wartungsfreier als diese Alternativen.

Die Dokumentation des DWH haben wir mit der dargestellten Architektur nahezu komplett nebenbei miterledigt. Datasets ebenso wie Variablen haben sprechende Namen, die eine detaillierte Dokumentation in vielen Fällen obsolet machen. Da eine relativ primitive Verzeichnisstruktur vorliegt, können wir mit einer einfachen PROC CONTENTS und der entsprechenden Autoexec eine tagesgenaue automatisierte Dokumentation des gesamten DWH zur Verfügung stellen.

Beachtet man alle diese Punkte, so kann man leicht ein kostengünstiges DWH effektiv bereitstellen. Das heisst aber nicht, daß man sklavisch dieser Architektur folgen muß ! Es gibt immer Bereiche, bei denen es Sinn machen kann, die Architektur zu durchbrechen. Hat man einen solchen Punkt identifiziert, so ist es sinnvoller, die Architektur anzupassen oder eine Ausnahme zu definieren anstatt auf Biegen und Brechen dem Buchstaben der Architektur zu folgen. Die (theoretische) Architektur lebt immer mit dem (praktisch erzeugtem) DWH, sonst wird sie sich nach kurzer Zeit selbst ad absurdum führen.

Literatur

[1] Roland Kapeller: Business Intelligence und Metadatenmodellierung (2003)

[2] Sean Kelly: Das Data Warehouse in Action (1997)

[3] Kemper / Finger: Datentransformationen für das DWH (1999)

[4] Inmon, William H.: Building the Data Warehouse (1996)

[5] Schinzer / Bange: Werkzeuge zum Aufbau analytischer Informationssysteme (1999)

[6] Kimball, Ralph: The Data Warehouse Toolkit (1996)

[7] Determann, Lorenz: Modellierung analytischer Informationssysteme (2002)