Data-Warehouse-Design

Seminar: Business Intelligence –

Teil I: OLAP & Data Warehousing

von Jörg Ramser

Betreuer: Dipl. Inform. Wolfgang Mahnke

18.07.2003

Gliederung

1 Einleitung

Unternehmen sind mit der Hilfe von aktuellen Informationen in der Lage, sich Wettbewerbsvorteile gegenüber ihren Konkurrenten zu sichern. Sie nutzen diese Informationen als Basis zur Entscheidungsfindung und erhoffen sich dadurch, in möglichst kurzer Zeit Tendenzen bzw. Verhaltensmuster in den Datenbeständen zu finden. Dadurch können Korrelationen zwischen zwei oder mehreren relevanten Unternehmensgrößen der Gegenwart und Vergangenheit abgeleitet werden. Aber auch Risiken sind schnell erkennbar und somit vermeidbar.
Zur Verwaltung dieser Informationen werden so genannte Data Warehouses genutzt. Darunter versteht man im Allgemeinen eine von den operativen Systemen losgelöste Datenbank, in der alle Daten, die in einem Unternehmen anfallen, zusammengetragen werden. Diese unternehmensweite Datenbasis ist Grundlage für alle Systeme, die das Management unterstützen. Die Bedeutung, die Data Warehouses heutzutage haben, wird klar, wenn man betrachtet, dass große Unternehmen schon heute Terra- beziehungsweise Petabyte an Daten für Analysezwecke in Data Warehouses abgelegt haben und die Tendenz steigend ist. Für einen weiteren Überblick und Anwendungsbeispiele von Data Warehouses sei hier auch auf [Seil03] verwiesen.
Diese Arbeit befasst sich im Rahmen des Seminars *Business Intelligence - OLAP & Data Warehousing* mit dem Design eines Data Warehouses. Dazu wird in Kapitel 2 der Schemaentwurf und die Umsetzung der bei Data Warehouses für Tools und Anwender vorhandenen multidimensionalen Sicht auf die Daten behandelt. Hierbei wird direkt auf die aus der beschriebenen Nutzung ableitbaren Besonderheiten des Data Warehouses eingegangen, wie z.B:

- multidimensionale Daten
- ein überwiegend lesender Zugriff
- die Forderung nach Möglichkeiten der Historisierung der Daten
- die Verwaltung von Datenmengen im Terra- bzw. Petabytebereich

Der Wert der Daten eines Data Warehouses ist maßgeblich davon abhängig, wie gut das Data-Warehouse-System die Daten bereitstellen und der Anwender sie interpretieren kann. Da Metadaten stark an dieser Thematik beteiligt sind, beschäftigt sich Kapitel 3 mit den Metadaten eines Data Warehouses. Bevor auf das Design eines Metadatenschemas eingegangen wird, betrachtet Kapitel 3 die besondere Rolle der Metadaten beim Data Warehouse und die in diesem Bereich vorherrschenden Standards. Abschließend wird der Entwurf eines Metadatenschemas für Data Warehouses behandelt.

2 Umsetzung des multidimensionalen Datenmodells

Den Datenanalyseprogrammen und Anwendern wird zur Erstellung ihrer Abfragen eine multidimensionale Sicht auf die Daten geboten. Allerdings muss die interne Verwaltung der Daten nicht nach dem multidimensionalen Datenmodell geschehen. Aufgrund ihrer technischen Reife stellt eine Speicherung in relationalen Datenbanken einen guten Ansatz dar. Diese Variante wird in Abschnitt 2.1 behandelt. In Abschnitt 2.2 wird die multidimensionale Speicherung betrachtet.

2.1 Relationale Speicherung

Dieser Abschnitt beschäftigt sich zunächst mit der Repräsentation der verschiedenen multidimensionalen Konstrukte im relationalen Datenmodell und den dabei auftretenden Schwierigkeiten. Anschließend wird die Umsetzung multidimensionaler Anfragen betrachtet.

2.1.1 Abbildungsmöglichkeiten auf Relationen

Es sollen nun nach ein paar allgemeinen Bemerkungen verschiedene Techniken der Abbildung auf relationale Datenbanken mit ihren Vor- und Nachteilen aufzeigt werden.
Zunächst ist bei der Abbildung zu beachten, dass sich Analyseanwendungen in Anfrage- und Aktualisierungscharakteristik von OLTP (On-Line Transactional Processing) Anwendungen zum Teil stark unterscheiden, so dass die dort verwendeten Techniken des Schemaentwurfs wie beispielsweise Normalisierung nicht unbedingt den gewünschten Effekt zeigen.
Eine gute Abbildung von multidimensionalen Konstrukten auf relationale DB-Schemata zeichnet sich aus durch

- die Möglichkeit multidimensionale Anfragen effizient in die entsprechenden relationalen Anfragen zu übersetzen und abzuarbeiten;
- einen möglichst geringen Verlust an anwendungsbezogener Semantik;
- eine einfache und schnelle Wartbarkeit der relationalen Tabellen, z.B. beim Laden von neuen Daten.

Im Folgenden wird die Abbildung des Beispieldatenwürfels in Abbildung 1 gezeigt. Zum besseren Verständnis geschieht dies zuerst ohne Klassifikationshierarchien. *Klassifikationshierarchien* bilden mittels einer Baumstruktur eine Abstraktionshierarchie über die Dimensionselemente einer Dimension. Dabei werden die Stufen des Baumes als *Dimensionsstufen* bezeichnet.

Faktentabelle
Die relationale Speicherung eines Datenwürfels ohne die Klassifikationshierarchien ist einfach zu realisieren. Hierfür bedient man sich einer so genannten Faktentabelle (engl. fact table). Bei dieser wird einfach ein Teil der Spalten als die Dimensionen des Würfels und der andere Teil der Spalten als Kenngrößen aufgefasst. So ist der Datenwürfel in Abbildung 1 äquivalent zur Tabelle in Abbildung 2.

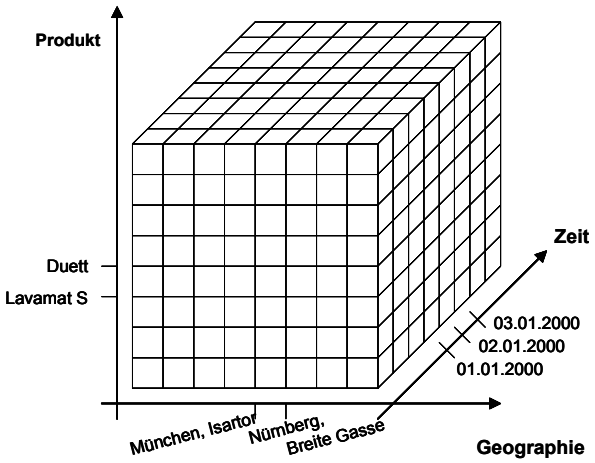

Abbildung 1: Beispielwürfel

Produkt (Dimension)	Geographie (Dimension)	Zeit (Dimension)	Verkäufe (Kenngröße)	Preis (Kenngröße)
Duett	Nürnberg, Breite Gasse	03.01.2000	2	800
Duett	München, Isartor	03.01.2000	3	1200
Lavamat S	München, Isartor	03.01.2000	2	1500

Abbildung 2: Faktentabelle zum Beispielwürfel

Es bleibt noch die Abbildung der Klassifikationshierarchien. Für diese Abbildung eignen sich neben dem Snowflake- bzw. dem Starschema auch Mischformen dieser beiden Schemata.

Snowflake-Schema
Beim Snowflake-Schema wird für jede Klassifikations*stufe* eine eigene Relation angelegt. Diese Relation enthält dann zum einem eine ID zur Unterscheidung der verschiedenen Knoten der Stufe und zum anderen beschreibende Attribute. Die 1:n-Beziehung zwischen zwei übereinander liegenden Klassifikationsstufen wird mit Hilfe eines Fremdschlüssels von der tiefer liegenden auf die nächst höhere Stufe dargestellt. In Abbildung 3 ist beispielhaft eine graphische Darstellung des Snowflake-Schemas zu sehen. Da dieses entfernt an eine Schneeflocke (eng. snowflake) erinnert, bekam dieses Schema den Namen Snowflake-Schema. [KRRT98]

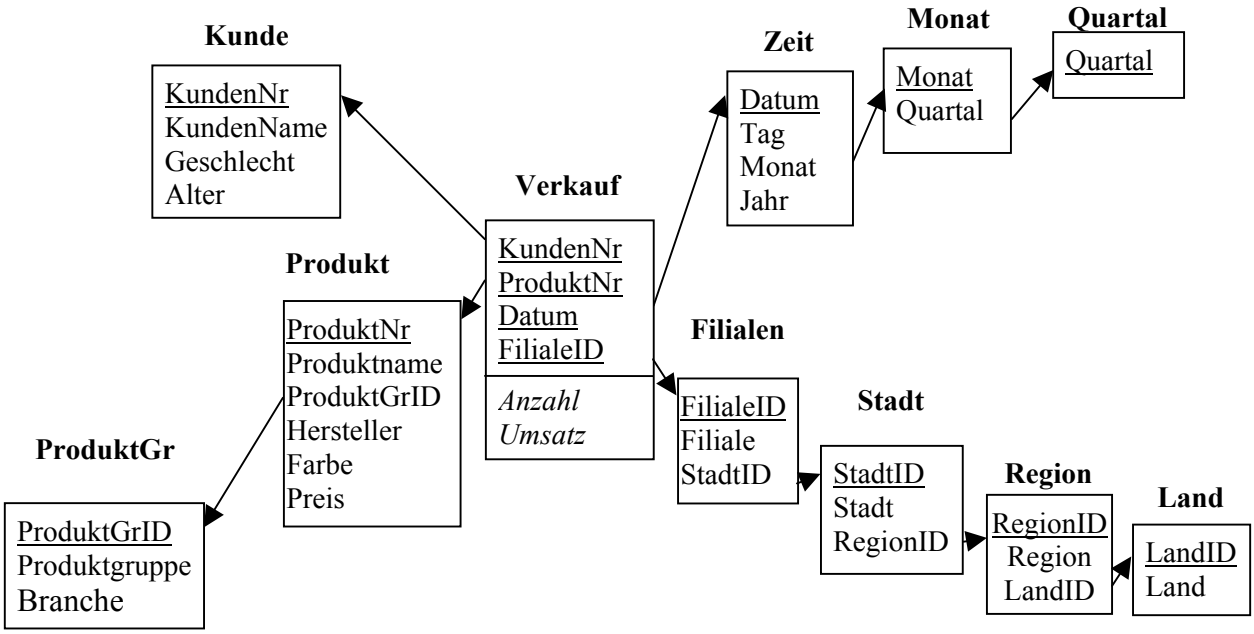

Abbildung 3: Umsetzung des Beispiels in ein Snowflake-Schema

Die Kennzahlen werden bei diesem Schema mit Hilfe der oben beschriebenen Faktentabelle verwaltet, wobei die Dimensionsspalten aus Fremdschlüsseln auf die Dimensionselemente der niedrigsten Stufe bestehen. Alle diese Fremdschlüssel zusammen bilden dann den Primärschlüssel der Faktentabelle.
Bleibt noch zu bemerken, dass das Snowflake-Schema bezüglich der funktionalen Abhängigkeiten, die durch die Klassifikationshierarchien entstehen (z.B. FilialeID $\rightarrow$ StadtID, StadtID $\rightarrow$ LandID) normalisiert ist. Hieraus resultiert auch der größte Nachteil des Snowflake-Schemas, nämlich die Vielzahl der durch diese Normalisierung entstehenden Tabellen. Diese müssen dann bei Anfragen miteinander verbunden werden, worunter natürlich die Performanz leidet [KRRT98].

Star-Schema
Den durch Normalisierung entstehenden Nachteil des Snowflake-Schemas versucht das Star-Schema durch Denormalisierung der zu einer Dimension gehörenden Tabellen zu vermeiden. [KRRT98] Ergebnis dieser Denormalisierung ist, dass pro Dimension nur noch eine Tabelle benötigt wird. Abbildung 4 zeigt nun obiges Beispiel als Star-Schema. Auch hier kann der Name des

Schemas aus der Form der graphischen Darstellung, welche an einen Stern (engl.: Star) erinnert, abgeleitet werden.

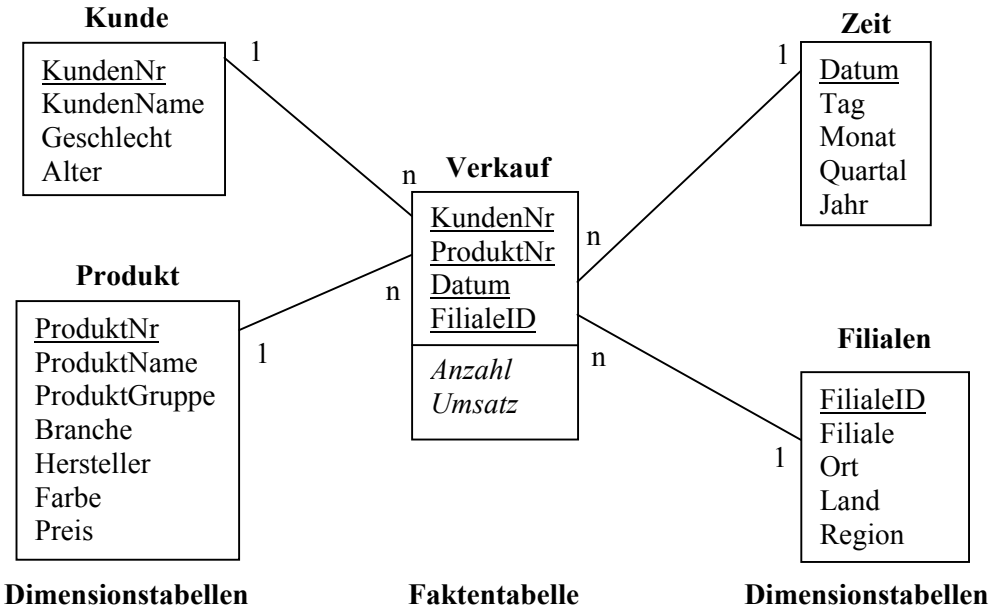

Abbildung 4: Das Beispiel als Star-Schema modelliert

Da die Dimensionstabellen nun nicht normalisiert sind, führt dies zu gewissen Redundanzen in diesen Tabellen. So bestimmt beispielsweise die Lage einer Filiale in einem bestimmten Ort auch gleichzeitig die Zugehörigkeit zu einem Land. Auch bei diesem Schema werden die Kennzahlen in gleicher Weise mittels Faktentabelle verwaltet.
Trotz der Nachteile durch die Redundanzen ist das Starschema dennoch häufig dem Snowflake-Schema vorzuziehen.

- Die Redundanzen liegen lediglich in den Dimensionstabellen. Diese haben aber im Allgemeinen ein geringes Datenvolumen im Vergleich zur Faktentabelle, so dass dies meistens nicht weiter ins Gewicht fällt. Hier existieren jedoch auch Ausnahmen.
- Änderungen an den Klassifikationen, und somit an den Dimensionstabellen sind nur selten nötig, so dass die Änderungsanomalien, die durch die Denormalisierung entstehen, nicht so in Gewicht fallen.
- Einschränkungen für Anfragen hingegen finden häufig auf höherer Granularitätsstufe statt. Da beim Snowflake-Schema jedes Mal teure Verbundoperationen nötig wären, während dies beim Star-Schema entfällt, kann man von einer deutlichen Steigerung der Anfragegeschwindigkeit ausgehen.
- Ein weiterer Vorteil des Star-Schemas liegt in der Einfachheit seiner Struktur, was sich vor allem bei manuellen Anfragen als Vorteil erweist.

Da dies nur charakteristische Aussagen sind, welche keinesfalls Allgemeingültigkeit besitzen, ist die Entscheidung zwischen Star- oder Snowflake-Schema nicht abschließend zu klären. Häufig werden daher Mischformen eingesetzt.

Mischformen von Star- und Snowflake-Schema
Um die spezifischen Nachteile von Star- und Snowflake-Schema zu vermeiden bzw. deren spezifische Vorteile zu nutzen kann man einzelne Dimensionen im Stile eines Star- und andere im Stile eines Snowflake-Schemas ablegen. Dabei spricht prinzipiell das Vorhandensein folgender Eigenschaften der jeweiligen Dimension für deren Abbildung mittels Snowflake-Schema:

- Frequenz der Änderungen *hoch*
- *Große* Anzahl von Dimensionselementen auf niedrigster Stufe
- Anzahl der Klassifikationsstufen innerhalb einer Dimension *hoch*
- Innerhalb einer Dimension *viele* Aggregate materialisiert

Galaxie
Es kommt zwar vor, dass die Modellierung eines Star-Schemas mit nur einer Faktentabelle auskommt. Dies ist aber lediglich dann möglich, wenn die Kenngrößen durch genau dieselben Dimensionen beschrieben werden können. Dies ist in der Realität allerdings meistens nicht der Fall, da sehr viele Kenngrößen mit jeweils unterschiedlichen Dimensionen existieren. Beispielhaft seien hier die Kenngrößen Verkäufe, Umsatz, Preis und Lagerbestand, von denen nur Verkäufe und Umsatz die gleichen Dimensionen haben, erwähnt. Diese beiden können daher auch als ein Würfel modelliert und dann in einer Faktentabelle gespeichert werden. Für alle anderen Kenngrößen muss aber jeweils eine eigene Faktentabelle erzeugt werden.
Dieses Schema mit mehren Faktentabellen, die teilweise mit den gleichen Dimensionstabellen verknüpft sind, nennt man Multi-Faktentabellen-Schema oder Galaxie [KRRT98].

Semantikverluste
Nachdem nun verschiedene Abbildungsvarianten betrachtet wurden, soll abschließend noch auf die ihnen anhaftenden Semantikverluste eingegangen werden. Der erste Punkt ist hier, dass im multidimensionalen Datenmodell zwischen Dimensionen und Kenngrößen unterschieden wurde. Schaut man sich aber die Faktentabelle an, so ist hier nicht mehr direkt zwischen Dimension und Kenngröße zu unterscheiden. Man kann diese Unterscheidung nur implizit daraus ersehen, dass Attribute Fremdschlüsselbeziehungen zu Dimensionstabellen haben. Auch die Unterscheidung in der Dimensionstabelle zwischen beschreibenden Attributen und den Hierarchie aufbauenden Attributen ist nicht mehr möglich. Die Frage, welche Stufe klassifiziert die andere und wie verlaufen die Drill-Pfade ist in relationalen Darstellungen auch nicht mehr zu beantworten. Es geht also auch der Aufbau der einzelnen Dimensionen verloren. Alle diese Probleme machen den Einsatz von Metadaten nötig, die genau diese Informationen explizit enthalten.

2.1.2 Umsetzung multidimensionaler Anfragen

Nachdem die einzelnen Abbildungsvarianten auf relationale Schemata erläutert worden sind, soll jetzt die Umsetzung der multidimensionalen Anfragen in relationale Anfragen auf den oben behandelten Schemata erläutert werden. Wobei die Umsetzung multidimensionaler Anfragen natürlich prinzipiell von der gewählten Abbildungsvariante (z.B. Snow, Star…) abhängt. Prinzipiell besteht eine Anfrage aber meist aus einer Aggregatanfrage, die aus einem (n+1)-Wege-Verbund zwischen den n Dimensionstabellen und der Faktentabelle sowie Restriktionen auf den Dimensionen besteht. In Anlehnung an den Begriff des Star-Schemas wird ein solcher Mehrfachverbund als Star-Join bezeichnet.

Star-Join-Anfragemuster
Für das Star-Join-Anfragemuster soll folgende beispielhafte Anfrage betrachtet werden: *„Wie viele Artikel der Produktgruppe Waschgeräte wurden 2000 pro Monat in den unterschiedlichen Regionen an männliche Kunden verkauft?“* Angewendet auf das oben betrachtete Star-Schema ergibt sich die in Abbildung 5 dargestellte Anfrage. Wobei in der SELECT-Klausel der Anfrage die aggregierten Kenngrößen (im Beispiel SUM(Verkauf.Anzahl)) und die gewünschte Ergebnisgranularität (im Beispiel Filiale.Region, Zeit.Monat), welche auch in der GROUP BY-Klausel auftauchen, angegeben werden. In der FROM-Klausel werden die Dimensionen, bezüglich derer Einschränkungen vorgenommen werden, und die Faktentabelle aufgeführt. Die WHERE-Klausel enthält neben den Verbundbedingungen auch die verlangten Restriktionen.

```
SELECT Filialen.Region,                                  --Ergebnissgranulariät
       Zeit.Monat,                                       --Ergebnissgranulariät
       SUM(Verkauf.Anzahl)                               --aggregierte Kenngröße

FROM Verkauf, Zeit, Produkt, Filialen, Kunde

WHERE Verkauf.ProduktNr = Produkt.ProduktNr              --Joinbedingung
  AND Verkauf.Datum = Zeit.Datum                         --Joinbedingung
  AND Verkauf.FilialeID = Filialen.FilialeID             --Joinbedingung
  AND Verkauf.KundenNr = Kunde.KundenNr                  --Joinbedingung

  AND Produkt.Produktgruppe = "Waschgeräte"              --Restriktionen
  AND Zeit.Jahr = 2000                                   --Restriktionen
  AND Filiale.Land = "Deutschland"                       --Restriktionen
  AND Kunde.Geschlecht = "männlich"                      --Restriktionen

GROUP BY Filiale.Region,                                 --Ergebnissgranulariät
         Zeit.Monat                                      --Ergebnissgranulariät
```

Abbildung 5: Beispielanfrage als Star-Join-Anfrage

Der Cube-Operator

Für statistische Auswertungen werden häufig Zwischen- und Gesamtsummen benötigt. Um dies entsprechend des oben stehenden Anfragemusters zu realisieren, müsste man für jede Kombination von Gruppierungsattributen eine eigene Teilanfrage machen und dann all diese Teilanfragen mittels UNION miteinander verknüpfen. So wären bei N Gruppierungsattributen 2^N Teilanfragen nötig. Wobei noch zu beachten ist, dass jedes Attribut, über das gerade hinwegaggregiert wird, mit einem konstanten Wert - zum Beispiel NULL - gefüllt werden müsste. In [GBLP96] wurde daher als verkürzte Schreibweise der so genannte Cube-Operator eingeführt, der automatisch nach jeder Kombination der übergebenen Attribute gruppiert. Später wurde dieser Operator auch in SQL99 übernommen [SQL99]. Ein weiterer Vorteil des Cube-Operators ist die optimierte Berechnung, da Zwischenergebnisse vom DBS wieder verwendet werden können. Zum Thema Cube-Operator sei hier auch auf die Arbeit [Rond03] verwiesen. Die Abbildung 6 zeigt ein Beispiel für den Cube-Operator, das Ergebnis der Anfrage ist in Abbildung 7 zu sehen.

```
SELECT Produkt.Hersteller, Zeit.Jahr, Produkt.Farbe,
       SUM (Verkauf.Anzahl)

FROM Verkauf, Produkt, Zeit

WHERE Verkauf.ProduktNr = Produkt.ProduktNr
  AND Produkt.Branche = 'Automobil'
  AND Verkauf.Datum = Zeit.Datum
  AND Verkauf.KundenNr = Kunde.KundenNr

GROUP BY CUBE (Produkt.Hersteller, Zeit.Jahr, Produkt.Farbe);
```

Abbildung 6: Beispiel für den Cube-Operator

Hersteller	Jahr	Farbe	Anzahl
VW	1998	Rot	800
VW	1998	Weiß	600
VW	1998	Blau	600
...	...	...	...
VW	1998	ALL	2000
...	...	ALL	...
Ford	2000	ALL	2000
VW	ALL	Rot	3400
...	...	...	...
Ford	ALL	Blau	...
ALL	1998	Rot	...
...	...	...	...
VW	...	ALL	ALL
...	...	...	...
ALL	1998	ALL	...
...	...	...	...
ALL	ALL	Rot	...
...	...	...	...
ALL	ALL	ALL	19500

Abbildung 7: Beispielanfrageergebnis für Anfrage mit Cube-Operator

2.1.3 Versionisierungs- und Evolutionsaspekte

Allgemein können Änderungen die Klassifikationshierarchien, das zugehörige Klassifikationsschema oder das Würfelschema betreffen.
Um relationale Datenbanken um temporale Aspekte zu erweitern, existieren prinzipiell die Möglichkeiten der Attribut-Zeitstempelung oder die der Tupel-Zeitstempelung.
Bei der Attribut-Zeitstempelung wird zu jedem Attribut mit Zeitabhängigkeit eine Gültigkeit gespeichert und zwar in Form eines Gültigkeitsanfangszeitpunkts und Gültigkeitsendzeitpunkts. Das heißt es existieren innerhalb eines Tupels Zeitversionen für die verschiedenen zeitabhängigen Attribute. Hierbei ist anzumerken, dass die heutigen relationalen Datenmodelle die Attribut-Zeitstempelung noch nicht unterstützen und hierfür temporale Erweiterungen nötig sind.
Bei der Tupel-Zeitstempelung werden zu jedem Tupel zwei Spalten hinzugefügt, welche den Anfang bzw. das Ende der Gültigkeit dieses Tupels angeben. Das hat zur Folge, dass die Änderung *eines* Attributs dazu führt, dass eine neue Version des *ganzen* Tupels angelegt wird, also Redundanzen nach sich zieht [KnMy96]. Trotz dieses Nachteils wird aufgrund ihrer Einfachheit die Tupel-Zeitstempelung in vielen kommerziellen Systemen eingesetzt.

Klassifikationshierarchieänderungen
Allgemein ist hier zu sagen, dass eine Änderung an der Klassifikationshierarchie nur selten nötig sein wird, da sich die Dimensionsstrukturen nur langsam verändern [Kimb96a], [Kimb96b].
Die erste Realisierungsalternative für Klassifikationshierarchieänderungen ist das Überschreiben oder „Update in place“, hierbei wird einfach das bestehende Tupel modifiziert. Da hierdurch die Forderung nach einer nachvollziehbaren Historie nicht erfüllt wird, kann diese Variante als nicht geeignet angesehen werden.
Der zweite Ansatz verwendet das Prinzip der Tupelversionierung, indem er jedes Tupel mit einer Versionsnummer versieht. Im Allgemeinen wird hierzu einfach der Primärschlüssel um diese Nummer erweitert. Dadurch ist das Problem des Verlusts der historischen Daten gelöst. Nachteil ist hier, dass die Zeit nicht explizit modelliert ist, da sie nur in Zusammenhang mit der Faktentabelle rekonstruiert werden kann.
Der dritte Ansatz setzt nun Zeitattribute ein, so dass die Zeit jetzt explizit modelliert ist. Diese Zeitattribute werden wie oben beschrieben jedem Tupel in Form eines Gültigkeitsanfangs- und eines Gültigkeitsendeattributs hinzugefügt.

Schemaänderungen
Prinzipiell sind Schemaänderungen auf dem Klassifikationsschema bzw. dem Würfelschema möglich. Beide Änderungen sind jedoch mit weitreichenden Konsequenzen verbunden, denn Änderungen auf Schemaebene ziehen Änderungen der Metadaten nach sich. Diese wiederum bedingen Änderungen der Instanzen.
Bei der Änderung des Schemas werden unterschiedliche Strategien verfolgt:

- *Schemaevolution:* Diese Strategie geht von einem Überschreiben auf Metaebene aus. Aufgrund des Überschreibens sind keine Zeitattribute nötig. Allerdings sind Änderungen der Instanzen von Nöten. Eine Erweiterung der Metaebene um Zeit- bzw. Versionsattribute ist nicht nötig.
- *Schemaversionisierung:* Hier findet die Versionen bzw. Zeitkontrolle auf der Metaebene statt, hierfür wird das Metaschema um Attribute für Versionen bzw. Zeit erweitert. Die Auswahl der gewünschten Version geschieht in der Regel bei der Anfrage. Änderungen der Instanzen zur Änderungszeit sind nicht nötig.

2.2 Multidimensionale Speicherung

In diesem Abschnitt soll gezeigt werden, wie eine direkte Speicherung des multidimensionalen Modells in multidimensionalen Datenbankmanagementsystemen (MDBMS), also ohne den Umweg über die relationalen Tabellen, aussieht.

2.2.1 Datenstrukturen

Hier sollen zuerst die Datenstrukturen Dimension und Würfel genauer betrachtet werden.

Dimension
Zur Speicherung wird eine Dimension als eine endliche *geordnete* Liste von Dimensionswerten aufgefasst, welche sowohl die Dimensionselemente als auch die höheren Klassifikationsstufen enthält. Indem man die Dimensionswerte auf eine Indexmenge abbildet erreicht man ihre Ordnung.
Eine Dimension D mit m Dimensionswerten kann man demnach formal als m-Tupel darstellen:

$$D = (x_1^D , x_2^D \ ... \ x_m^D)$$

Würfel
Einen mehrdimensionalen Würfel kann man durch die Kombination mehrer Dimensionen definieren. Hierbei hilft die Vorstellung, dass die n Dimensionen einen n-dimensionalen Raum aufspannen. Die m verschiedenen Werte einer Dimension teilen diesen Würfel in m verschiedene parallele Ebenen. Der Schnittpunkt von n Ebenen eins n-dimensionalen Würfels bezeichnet eine Zelle, in welcher dann die Kenngrößen abgelegt werden. Diese Zelle kann eindeutig über ein n-Tupel von Dimensionswerten angesprochen werden, wobei der i. Wert des Tupels ein Wert der Dimension D_i ist. Es bietet sich also folgende Schreibweise an:

$$W = ((D_1, D_2, ..., D_n), (M_1:Typ_1, ..., M_m:Typ_m)),$$

D_i : Dimensionen, $M_i:Typ_i$: Kenngrößen des Würfels mit Datentyp

Beispielsweise:
Verkauf = ((Produkt, Filiale, Zeit), (Anzahl: integer, Umsatz: long))

Besonders betrachtet werden müssen leere Zellen und dabei vor allem die Ursache warum sie leer sind (z.B. kein Wert verfügbar oder Wert aufgrund der Semantik der Anwendung nicht sinnvoll). Für die verschiedenen Fälle muss dann jeweils ein Symbol zur Darstellung definiert werden.

Virtuelle Würfel
Im multidimensionalen Modell kann zwischen konkret gespeicherten Daten und den daraus abgeleiteten Daten unterschieden werden. Abgeleitete Daten wie Gewinn oder prozentualer Umsatz entstehen aus Berechnungen auf gespeicherten oder anderen abgeleiteten Daten.
Die Berechnungsfunktion bildet einen vorhandenen Würfel auf einen anderen, abgeleiteten Würfel ab, dieser Würfel ist dann virtuell. Auf virtuelle Würfel kann man alle Operationen, die auch für physische Würfel angeboten werden, anwenden. Virtuelle Würfel kann man somit mit dem Sichtenmechanismus in relationalen Datenbanksystemen vergleichen.
Zur Auswertung von Daten eines virtuellen Würfel werden die Daten, die in der Berechnungsvorschrift des virtuellen Würfels aufgeführt sind, gelesen und gemäß der definierten Operationen dynamisch zum Anfragezeitpunkt verknüpft. Ein virtueller Würfel kann sich auch auf andere virtuelle Würfel beziehen. Die dynamische Berechnung des virtuellen Würfels wird nur für die Bereiche des virtuellen Würfels (Unterwürfel, Ebenen, einzelne Zellen) vorgenommen, die auch für die Resultatbildung notwendig sind.

Teilwürfel
Die Ebenen eines Würfels in Bezug auf eine bestimmte Dimension kann man über die Dimensionswerte dieser Dimension identifizieren. Die Ebenen bilden somit Schnitte im Würfel. Die Ebenen beziehungsweise Schnitte entlang einer Dimension können auch kombiniert werden. Es entstehen dann mehrere parallele Ebenen oder eine dickere Ebene, falls die verwendeten Dimensionswerte in der Werteliste aufeinander folgen.
Kombiniert man nun mehrere Ebenen beziehungsweise mehrere Schnitte von jeweils unterschiedlichen Dimensionen, so entsteht dadurch ein Teilwürfel, dass Ebenen miteinander über eine Schnittmenge verknüpft werden. Diese Teilwürfel sind zunächst nur virtuell, denn sie entstehen erst zur Laufzeit durch die Schnittbildung. Ansonsten zeichnen sich Teilwürfel durch die gleichen Eigenschaften wie die Ursprungswürfel aus.

Klassifikationshierarchien und Aggregationen
Wie bereits oben erwähnt umfassen die Dimensionswerte sowohl die Dimensionselemente als auch die Knoten der höheren Klassifikationsstufen. Letztere bilden somit weitere Ebenen im Würfel hinsichtlich der jeweiligen Dimension. Die baumartigen Zuordnungen der Klassifikationshierarchie werden auf Zuordnungen zwischen Klassifikationsstufen im Würfel abgebildet. In Abbildung 8 ist eine solche Zuordnung zu sehen.

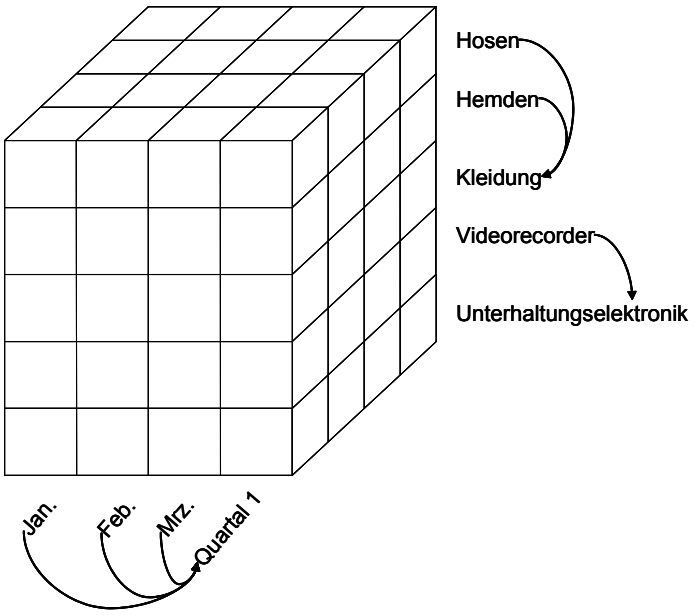

Abbildung 8: Hierarchiezuordnungen

Die Aggregationen, die auf und durch die Klassifikationshierarchien definiert sind, bestimmen die Werte, die für die höheren Stufen der Klassfiktionshierarchien gelten. Prinzipiell sind für die Be-

rechnung dieser Aggregationen zwei verschiedene Ansätze denkbar. Eine Variante ist die „Echtzeitberechnung“, also eine Berechnung immer bei Bedarf, die andere denkbare Variante ist eine Vorberechnung (engl.: batch). Auf die Vor- bzw. Nachteile der einzelnen Varianten soll hier in diesem Rahmen allerdings nicht eingegangen werden.

2.2.2 Speicherung multidimensionaler Daten

In diesem Abschnitt wird nun die direkte multidimensionale Speicherung der Daten mit Hilfe multidimensionaler Arrays betrachtet.

Array-Speicherung

Bei der multidimensionalen Speicherung wird der mehrdimensionale Würfel in einem multidimensionalen Array, dessen Indizes die Koordinaten der Würfelzellen bilden, gespeichert. Dieses multidimensionale Array muss nun wiederum zum Speichern in eine 1-dimensionale Liste überführt werden, dieser Vorgang wird Linearisieren genannt. Die Linearisierungsreihenfolge bei der Array-Linearisierung ist für den Spezialfall des dreidimensionalen Arrays in Abbildung 9 dargestellt.

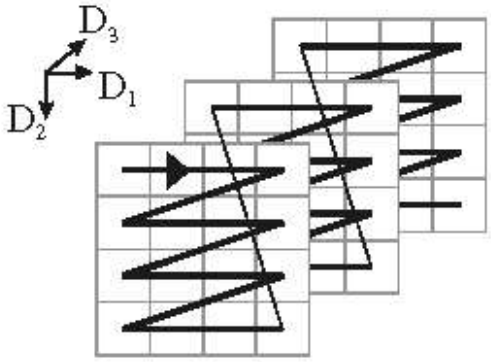

Abbildung 9: Linearisierungsreihenfolge für Array-Linearisierung

Allgemein lässt sich daraus folgende Berechnungsvorschrift für den Array-Index der Zelle z $(x_1,x_2,...x_n)$ eines Würfels

$$W = ((D_1, D_2, ..., D_n) , (M_1{:}Typ_1, ..., M_m{:}Typ_m))$$

ableiten:

$$Index(z) = x_1 + (x_2-1) \cdot |D_1| + (x_3-1) \cdot |D_1| \cdot |D_2| + ... + (x_n-1) \cdot |D_1| \cdot ... \cdot |D_{n-1}|$$

$$= 1 + \sum_{i=1}^{n} (x_i - 1) \cdot \prod_{j=1}^{i-1} |D_i|$$

Probleme bei der Array-Speicherung

Ein großes Problem bei der Array-Speicherung stellt die Tatsache dar, dass in der Realität die Würfel zumindest auf Detailebene dünn besetzt sind. Für die Berechnung der Position einer Zelle in dem eindimensionalen Array ist es notwendig, dass für leere Zellen Platz reserviert wird, sie also physisch vorhanden sind. Dieses Problem existiert nicht auf den höheren Aggregationsstufen, da es für das Vorhandensein eines Aggregationswertes völlig ausreichend ist, wenn nur ein Wert der zugeordneten unteren Ebenen vorhanden ist.

2.3 Vergleich von relationaler und multidimensionaler Speicherung

Dieser Abschnitt stellt die Vor- und Nachteile der relationalen bzw. multidimensionalen Speicherung gegenüber.

Zuerst werden nun die Vor- und Nachteile der relationalen Speicherung betrachtet.

Vorteile:

- Verwendet bewährte Datenbanktechnologie
- Zugriff auf das Data Warehouse erfolgt mittels Standard-SQL, welches von vielen Anwendern verstanden und Herstellen unterstützt wird
- Datenimport leicht möglich
- Sicherheitsmechanismen sind bereits auf relationaler Ebene vorhanden
- Große Datenmengen können verarbeitet werden
- Leichte Skalierbarkeit

Nachteile:

- Standard-SQL ist für multidimensionale Analysen nur bedingt ausreichend
- Semantikverluste durch die Abbildung multidimensionaler Daten auf relationale Datenbanken
- Multidimensionale Anfragen müssen auf SQL-Anfragen abgebildet werden, dadurch Performanzverluste

Dem gegenüber stehen die Vor- und Nachteile der multidimensionalen Speicherung:

Vorteile:

- Hohe Anfragegeschwindigkeit
- Effiziente multidimensionale Speicherstrukturen
- Meist eigene, multidimensionale Abfragesprache, intuitiv verständlicher als SQL

Nachteile:

- Problematik der dünn besetzten Würfel muss gelöst werden
- Skalierbarkeit eingeschränkt, Grenze ist nicht absolut festzusetzen, sie variiert von Hersteller zu Hersteller
- Proprietäre MDDB-Systeme werden eingesetzt, keine Abfragesprache als Standard definiert

Aufgrund der Tatsache, dass beide Verfahren einige Vor- bzw. Nachteile haben, kann keinem der Verfahren generell der Vorzug gegeben werden. Ein Versuch die Vorteile beider Verfahren zu vereinigen stellt die Hybride Speicherung im nächsten Kapitel dar.

2.4 Hybride Speicherung

Der Ansatz der Hybriden Speicherung (Hybrides OLAP) soll die Vor bzw. Nachteile der einzelnen Ansätze verbinden. Grundsätzliche Idee ist hier, Detaildaten, die meistens in großer Anzahl vorliegen, in einer relationalen Datenbank zu speichern und aggregierte Daten in multidimensionalen Strukturen abzulegen, um einen schnellen Zugriff auf die obersten Klassifikationsstufen, welche in der OLAP-Verarbeitung oftmals zuerst analysiert werden, zu gewährleisten. Auf den oberen Klassifikationsstufen besteht auch das Problem der dünn besetzten Datenbestände nicht, eines der Probleme der multidimensionalen Speicherung.
Der Zugriff erfolgt über die multidimensionale Datenbank durch ein multidimensionales Anfragewerkzeug. Bei der Anfrage muss dann jeweils entschieden werden, ob die Daten in der multidimensionalen Welt vorliegen, ob sie in der relationalen Datenbank vorliegen und von dort gelesen werden müssen oder die dritte Möglichkeit, dass sie nur aus Berechnungen von Daten aus beiden Welten zu gewinnen sind. Die Entscheidungsfindung über den nötigen Beschaffungsweg der Daten ist für den Anwender transparent. Die Art der Verteilung auf relationale bzw. multidimensionale Datenbanken ist anwendungsspezifisch.
Diese Kombination beider Datenbankmanagementsysteme vereint die Vorteile und überwindet gleichzeitig die Nachteile beider Welten.

3 Metadaten beim Data Warehousing

Bei der Qualität der Daten eines Data Warehouses spielen die Metadaten eine große Rolle. Der Begriff Metadaten umfasst sowohl die Schemadaten der beteiligten Datenbanken, als auch Aktualisierungsdaten oder Zusatzinformationen, die in einem Data Warehouse System entstehen.

3.1 Die Rolle von Metadaten beim Data Warehousing

Metadaten sind notwendig um Daten besser verstehen, verwalten und benutzen zu können. Hierbei stammt die klassische Definition des Begriffs Metadaten aus dem Bereich der Datenbanken: Dort versteht man unter Metadaten die Definition der Struktur der Datenbank, d.h. die Schemainformation [BaGü01]. Diese Sicht wurde inzwischen aufgrund neuer Bedürfnisse ausgeweitet, so dass sich nach [BaGü01] folgende Definition ergibt:

„Unter dem Begriff Metadaten versteht man gemeinhin jede Art von Information, die für den Entwurf, die Konstruktion und die Benutzung eines Informationssystems nötig wird."

Auch beim Data Warehouse sind Metadaten unverzichtbar, um Informations-, Schutz- und Sicherheitsbedürfnisse der verschiedenen Anwender und Softwarekomponenten abzudecken. Hierfür werden Metadaten in allen Phasen des Data Warehousing produziert und konsumiert. Beim Data Warehouse ist hierfür eine eigene Verwaltungskomponente, der Metadatenmanager, zuständig. Prinzipiell kann man die Art der Nutzung wie folgt unterscheiden [BaGü01].

- *Passiv:* Die passive Nutzung der Metadaten dient der konsistenten Dokumentation, welche von allen Akteuren im Data Warehouse genutzt werden kann.
- *Aktiv:* Bei der aktiven Nutzung werden Transformationsregeln als Metadaten gespeichert, die dann zur Ausführungszeit von Werkzeugen interpretiert und ausgeführt werden können.
- *Semiaktiv:* Bei der semiaktiven Nutzung werden Strukturinformationen wie beispielsweise Tabellendefinitionen und Konfigurationsspezifikationen im Repositorium gespeichert. Diese Metadaten können zur Überprüfung eingesetzt werden (zum Beispiel kann ein Anfrageparser prüfen ob die Typinformationen korrekt sind). Von der aktiven unterscheidet sich die semiaktive Verwendung durch die Tatsache, dass die Metadaten bei der semiaktiven Verwendung nicht direkt zur Ausführung eines Prozesses verwendet werden.

Mit den Metadaten will man die effektive Beschaffung von Informationen ermöglichen, hierbei zählen folgende Aspekte:

- *Datenqualität:* Bei der Gewährleistung von Datenqualität spielen die Kriterien Konsistenz, Korrektheit und Vollständigkeit eine Rolle. Damit diese Aspekte im Data Warehouse gewährleistet werden können, müssen Regeln zur Überprüfung definiert werden, die bei jedem Aktualisierungsprozess ausgeführt werden. Ebenfalls wichtig bei der Gewährleistung von Datenqualität ist das Vorhandensein von *Nachvollziehbarkeitsinformationen*. Diese umfassen Metadaten über das Quellsystem von Daten, die Erstellungszeit, den Autor, die Bedeutung der Daten zum Zeitpunkt der Erfassung und vieles weitere. Nur mir diesen Metainformationen ist es dem Anwender möglich, den Weg eines einzelnen Datenelements von der Quelle bis ins Data Warehouse zu rekonstruieren und die Genauigkeit und damit die Qualität der gelieferten Information zu überprüfen.
- *Terminologie:* Um eine einheitliche Interpretation der Daten zu ermöglichen, ist eine einheitliche Terminologie notwendig. Vorrausetzung für eine einheitliche Terminologie wiederum ist das Vorhandensein eines Metadatenmanagementsystems als einzige Informationsquelle.
- *Datenanalyse:* Metadaten über die Bedeutung von Daten, die verwendete Terminologie, Kennzahlensysteme und weitere unterstützen den Anwender bei der Analyse der Daten

in Form von effizienter Formulierung präziser Anfragen an das Data Warehouse und erlauben die korrekte Interpretation der zurückgelieferten Werte.

Die zweite Zielsetzung der Metadaten besteht darin, den Aufbau und den laufenden Betrieb eines Data Warehouses zu vereinfachen. Im Einzelnen ist das:

- *Automatisierung der Administrationsprozesse:* Die Ausführung der Prozesse eines Data Warehouse benötigt Scheduling- beziehungsweise Konfigurationsmetadaten.
- *Systemintegration:* Zur Schema- und Datenintegration sind Informationen über die Struktur und Bedeutung der einzelnen Quellen und des Zielsystems von Nöten. Dies setzt einheitliche und konsistente Metadaten voraus.
- *Schutz- und Sicherheitsaspekte:* Damit nicht jedes Werkzeug eigene Schutzfunktionen realisieren muss, werden die Zugriffs- und Benutzerrechte des Data Warehouses als Metadaten gespeichert.
- *Flexibler Softwareentwurf:* Sich häufig ändernde semantische Aspekte werden nicht in der Anwendung, sondern explizit als Metadaten gespeichert und erhöhen damit die Wiederverwendbarkeit und Wartbarkeit der Anwendung.

3.2 Metadatenmanagement

Die Speicherung und Verwaltung der Metadaten findet in einem Repositorium statt, welches selbst wiederum auf der Basis eines Datenbankmanagementsystems realisiert wird. Die Bedürfnisse des Anwendungsgebietes bestimmen die Struktur (Metadatenschema) des Repositoriums sowie die Semantik der zu speichernden Metadaten.
Die Anforderungen an ein Repositorium sind unterteilt in Anforderungen an die Funktionalität und Anforderungen an die Architektur, welche in diversen Standards definiert sind.
Die Struktur und der Inhalt eines Repositoriums werden bestimmt vom zu modellierenden Informationssystem. Um komplexe Informationssysteme adäquat zu modellieren, sind mindestens vier Ebenen notwendig, wobei jede Ebene die Konstrukte oder Sprache zur Definition der darunter liegenden Ebene enthält. Es ergeben sich nach [BaGü01] folgende 4 Ebenen:

- Ebene 0 enthält die effektiven Daten (Objektdaten) wie beispielsweise die Kundendaten.

Die darüber liegenden Ebenen enthalten die eigentlichen Metainformationen:

- Auf Ebene 1 befinden sich die *Metadate*n als Modell des zu modellierenden Informationssystems, zum Beispiel das Datenbankschema oder auch die Prozessbeschreibungen.
- Ebene 2 definiert die Sprachelemente der Ebene 1 (*Metamodell* auch *Metadatenschema* genannt). Hier ist auch das konzeptuelle Schema des Repositoriums angesiedelt.
- Ebene 3 schließlich enthält das *Metametamodell*, das die verschiedenen Sprachen der Ebene 2 vereinigt.

3.2.1 Anforderungen an Repositorien

Wie bereits oben erwähnt gliedern sich die Anforderungen an die Repositorien in Anforderungen an die Funktionalität und Architektur. Hier sollen nun die wichtigsten Aspekte nach [Bern98] aufgeführt werden.

Funktionalität
Diese Anforderung kann man gliedern in Anforderungen an den Anwenderzugriff, die Interoperabilität und das Änderungsmanagement.

Anwenderzugriff
Die Hauptaufgabe eines Repositoriums ist, den Anwendern die Informationen zu Verfügung zu stellen, die sie zur Erfüllung ihrer Aufgaben benötigen. Dabei muss sich die Benutzerführung an den Kenntnisstand des jeweiligen Anwenders anpassen.
Das Repositorium stellt Mechanismen zur Navigation, Selektion, Filterung und manuellen Aktualisierung von Metadaten zur Verfügung:

- *Navigation:* Steuerung der Navigation im Repositorium erfolgt durch das Metadatenschema. Ausgehend von einem konkreten Metadatenelement soll der Anwender die Möglichkeit haben, sich anhand existierender Beziehungen zu anderen Elementen navigieren zu können.
- *Selektion:* Die Struktur des Repositoriums (Metadatenschema) muss die Anfrage nach bestimmten Kriterien unterstützen. Ein Beispiel ist die Selektion der Aktivitäten, die einen bestimmten Ladeprozess definieren.
- *Filterung:* Beim Filtern von Metadaten wählt man die Elemente anhand von Suchkriterien aus, die nicht zwangsweise durch die Struktur des Repositoriums vorgegeben sein müssen. Ein Beispiel für Filtern ist die Suche nach Schlüsselbegriffen innerhalb textueller Beschreibungen.
- *manuelle Aktualisierung:* Um eine manuelle Aktualisierung des Repositoriums zu unterstützen, sind ausgefeilte Konzepte zur Benutzerführung bereitzustellen, um die Konsistenz des Repositoriums zu gewährleisten. Um zum Beispiel die Eingabe langer Sequenzen von verbundenen Elementen zu ermöglichen, erweitert man das Metamodell um zusätzliche Teilmodelle, die die Prozesse der Eingabe formalisieren und damit die Generierung angepasster Eingabemasken ermöglichen.

Interoperabilität
Eine Interaktion von Werkzeugen mit dem Repositorium beziehungsweise zwischen Repositorien untereinander erfordert:

- Die Definition eines umfassenden Austauschformates, in dem sich Metadaten importieren und exportieren lassen;
- eine umfassende Programmierschnittstelle (API);
- ein erweiterbares Metamodell, dem ohne viel Aufwand den Domänen angepasste Metadatentypen hinzugefügt werden können.

Änderungsmanagement
In diesem Zusammenhang werden von einem Repositorium folgende Komponenten und Dienste erwartet:

- *Versions- und Konfigurationsverwaltung*
- *Notifikationsmechanismus:* Ein Notifikationsmechanismus erlaubt die Verbreitung von Änderungshinweisen an Werkzeuge und Anwender, die ihr Interesse an solchen Hinweisen angemeldet haben.
- *Auswirkungsanalysen*: Auswirkungsanalysen ermöglichen dem Administrator die Auswirkungen von geplanten Änderungen im Repositorium vor der effektiven Änderungen zu evaluieren

Architektur
In Abbildung 10 sind sowohl die Datenquellen als auch die Werkzeugarten, die in die Metadatenverwaltung involviert sind, zu sehen. Zuerst werden nun die Komponenten der Abbildung mit ihren Funktionen aufgeführt.

- *Metadatenmanager:* Der Metadatenmanager ist für die Verwaltung des Repositoriums zuständig, er stellt die persistente Speicherung und Wiederanlaufverfahren zur Verfügung. Falls die Datenquellenmetadaten nicht im Repositorium abgelegt wurden, ist der Metadatenmanager auch dafür zuständig deren Integration mit dem Repositorium zu sichern. Alle Zugriffe auf das Repositorium laufen über die Schnittstellen des Metadatenmanagers.

- *Anwenderzugriffswerkzeuge:* Anwenderzugriffswerkzeuge sind für die im obigen Absatz Funktionalität vorgestellten Funktionalitäten gegenüber den Anwendern zuständig.
- *Data-Warehouse-Manager:* Der Data-Warehouse-Manager benutzt das Repositorium als Ablage für Steuerungsinformationen jeglicher Art. Zur Laufzeit werden diese Steuerungsinformationen den entsprechenden Werkzeugen (z.B. Administrations- oder Analysewerkzeuge) zur Verfügung gestellt, die sie dann interpretieren und ausführen. Diese Werkzeuge können selbst Metadaten produzieren und sie im Repositorium ablegen.
- *Analysewerkzeuge:* Analysewerkzeuge stellen den Anwendern die gewünschten Metadaten, wie zum Beispiel Auswertungen oder die Bedeutungen einzelner Datenelemente, zur Verfügung.
- *Datenbeschaffungswerkzeuge:* Datenbeschaffungswerkzeuge legen technische Metadaten ab, die dem Systemadministrator nützlich sind, Beispiele hierfür sind statistische Werte über die Anzahl geladener Datensätze oder Log-Dateien.
- *Entwicklungswerkzeuge:* Entwicklungswerkzeuge verwenden Metadaten für den Entwurf neuer Anwendungen im Data-Warehouse-System.

Die Gesamtarchitektur der Metadatenverwaltung kann unterschiedlich realisiert sein. Generell bietet sich an, sämtliche Metadaten in einem einzigen Repositorium abzulegen. Man nennt dies *zentralisierte Metadatenverwaltung*: Metadaten werden zentral und konsistent verwaltet, der Zugriff erfolgt einheitlich für alle Anwender. Dieser Ansatz ist in der realen Welt oft nicht realisierbar: Es existieren daher zwei Alternativen, eine komplett *dezentralisierte* und eine *föderierte* Verwaltung von Metadaten. Bei der *dezentralisierten Metadatenverwaltung* sind die einzelnen Repositorien völlig unabhängig. Es wird lediglich versucht, den Austausch von Metadaten mit Hilfe von Standards zu unterstützen. Eine *föderierte Metadatenverwaltung* ist eine Mischung aus zentraler und dezentraler Verwaltung, es wird zwar eine globale, konzeptuelle Sicht auf die Metadaten eines Unternehmens geboten, allerdings sind die einzelnen Repositorien weiterhin autonom in der Pflege der Daten.

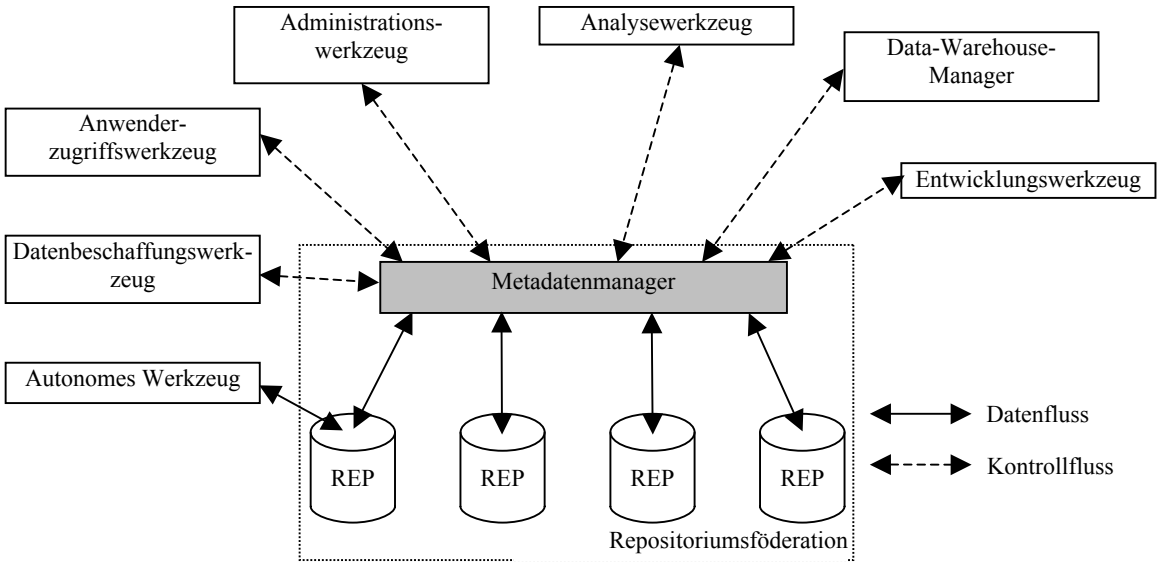

Abbildung 10: Repositoriumsföderation und Werkzeuge

3.2.2 Repositorium- und Metadatenaustauschstandards

Dieses Kapitel behandelt die wichtigsten Standardisierungsbestrebungen und Ansätze in Bezug auf Repositorien und den Austausch zwischen ihnen.

Repositoriumstandards
Repositoriumstandards stellen Referenzarchitekturen dar, die von Herstellern der Repositoriumssysteme realisiert werden sollen, um ihre Produkte universell einsetzbar zu machen. In diesem Zusammenhang sind folgende Standards von Bedeutung:

- *Information resource dictionary system (IRDS):* Der IRDS-Standard wurde 1990 von der ISO definiert und behandelt die Anforderungen und die Architektur eines Repositoriums.
- *Portable Common Tool Environment (PCTE):* Der PCTE-Standard, wurde ebenfalls 1990 von der European Computer Manufacturer's Association (ECMA) definiert und beschreibt die Basis für eine standardisierte Softwareentwicklungsumgebung.

Austauschstandards
Um die geforderte Interoperabilität zwischen Repositorien zu erreichen ist eine Standardisierung von Austauschformaten unumgänglich. Dabei existieren folgende Techniken:

- XML-basierte Austauschstandards;
- Case Data Interchange Format (CDIF): Der CDIF-Standard ist der einzige nennenswerte Austauschstandard, der nicht auf XML basiert.

3.3 Data-Warehouse-Metadatenschemata

In diesem Kapitel soll das Metamodell eines Repositoriums (Metadatenschema) betrachtet werden. Das Metadatenschema definiert die Metadatentypen und deren Beziehungen untereinander, um sie dann den Anwendern und Werkzeugen zur Verfügung zu stellen. Im Bereich der Standards für Metamodelle existieren drei Modelle:

- CWM (Common Warehouse Metamodel) [CWM01] der Object Management Group (OMG), unterstützt von 17 namhaften Unternehmen, wie z.B. IBM
- OIM (Open Information Model) der Meta Data Coalition, einer Vereinigung von 50 Firmen, wie z.B. Microsoft
- Zachman-Framework, vorgeschlagen von John A. Zachman

Das CWM und das OIM basieren auf UML und verwenden XML als Basis für den Austausch von Metadaten. Das Zachman-Framework ist eine allgemeine Architektur von Informationssystemen, die auch als Organisationsschema für Metadaten dienen kann. Beispielhaft wird in diesem Zusammenhang das CWM betrachtet.

CWM (Common Warehouse Metamodel)
Das Ziel des CWM ist die Unterstützung eines einfachen Austauschs von Data-Warehouse-Metadaten zwischen Werkzeugen und Repositorien.
Abbildung 11 zeigt die einzelnen Teilmodelle und die Abhängigkeiten zwischen ihnen. Als Notation wurde bei CWM die UML gewählt.

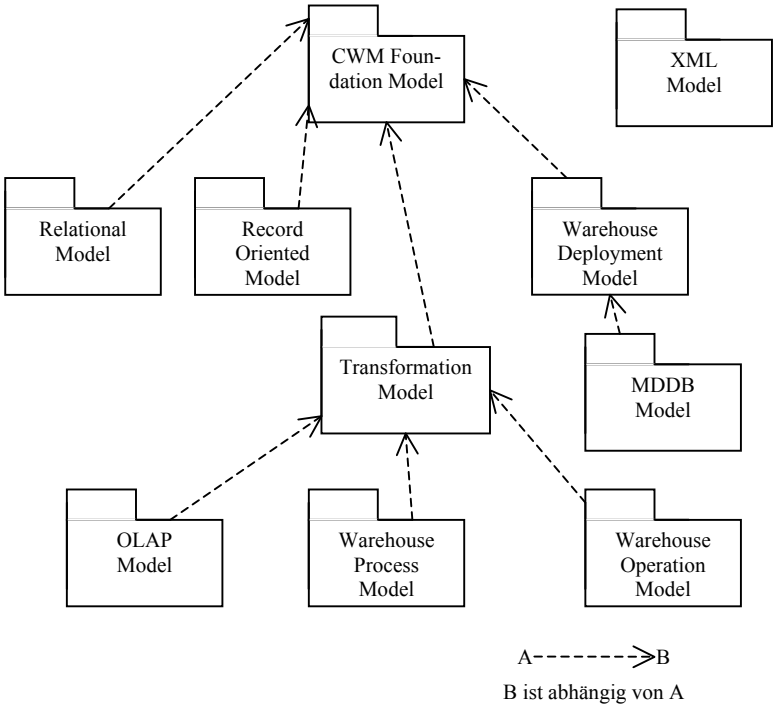

Abbildung 11: Common Warehouse Metamodell

Im Folgenden werden nun die Funktionen der einzelnen Teilmodelle gezeigt [CWM01].

- Im *CWM Foundation Model* werden allgemeine Konzepte und Strukturen zur Verwendung in den anderen Teilmodellen definiert.
- Im *Relational Model* werden die relationalen Datenstrukturen beschrieben.
- Das *Multidimensional Database (MDDB) Model* ist eine generische Repräsentation einer multidimensionalen Datenbank.
- Das *XML Model* enthält die Klassen zur Beschreibung der XML-Datenstrukturen
- Das *Warehouse Operation Model* enthält Klassen zur Dokumentation der alltäglichen Data Warehouse-Prozesse.
- Das *Record Oriented Model* behandelt das Konzept eines Records.
- Im *Transformation Model* werden Transformationen zwischen unterschiedlichsten Arten (objektorientiert, relational, multidimensional etc.) von Quell- und Zieldateien aufgeführt.
- Das *OLAP Model* definiert ein Metamodell der grundlegenden OLAP-Konstrukte, die von den OLAP-Werkzeugen und OLAP-Anwendungen verwendet werden.
- Das *Warehouse Process Model* dokumentiert den Prozessfluss zur Ausführung von Transformationen im Data Warehouse.
- Das *Warehouse Deployment Model* enthält die Elemente zur Definition der Verwendung von Hard- und Software.

3.4 Entwurf eines Schemas zur Verwaltung der Metadaten

In diesem Abschnitt wird gezeigt, welche Metadatentypen in einem Schema für Data-Warehouse-Repositorien verfügbar sein sollten.
Dieses Thema wird aus unterschiedlichen Perspektiven betrachtet. Das hier betrachtete Modell nach [BaGü01] beschränkt sich auf die Darstellung der Klassen und ihrer Beziehungen untereinander. Die betrachteten Ausschnitte sollen keineswegs den Anspruch der Vollständigkeit erheben, sondern sollen als erweiterbare Grundlage verstanden werden.

3.4.1 Funktionale Aspekte

Die funktionalen Aspekte können noch grob in Transformationsprozesse und multidimensionale Datenbanken aufgeteilt werden.

Transformationsprozesse

Hier soll die Modellierung von Transformationsprozessen gezeigt werden. Abbildung 12 zeigt das entsprechende UML-Diagramm.
Hierbei ist eine *Transformation* eine vom Anwender definierte atomare Einheit der Verarbeitung, aus der sich durch geordnete Zusammenfassung zu einer Gruppe, die *TransformationGroup,* eine logische Arbeitseinheit bilden lässt. Der *TransformationProcess*, eine Spezialisierung von *Process*, ist eine Zusammenfassung von diesen logischen Arbeitseinheiten zu physisch auszuführenden Prozessen.
Diese drei Elemente sind nun Verfeinerungen eines abstrakten *ExecutionElements*. Dieses bestimmt welche Datenobjekte (*source*) zu welchen Datenobjekten (*target*) verarbeitet werden können, wobei die Datenobjekte in der *DataObjectSets* zu Mengen zusammengefasst werden. Die zweite Aufgabe eines *ExecutionElements* ist die Herstellung einer Verbindung zu einem *ActivationElement*, dieses wiederum stellt eine Realisierung des abstrakten *ExecutionElements* dar. Die Klasse *Process* dient der Erweiterung um weitere Prozessarten.

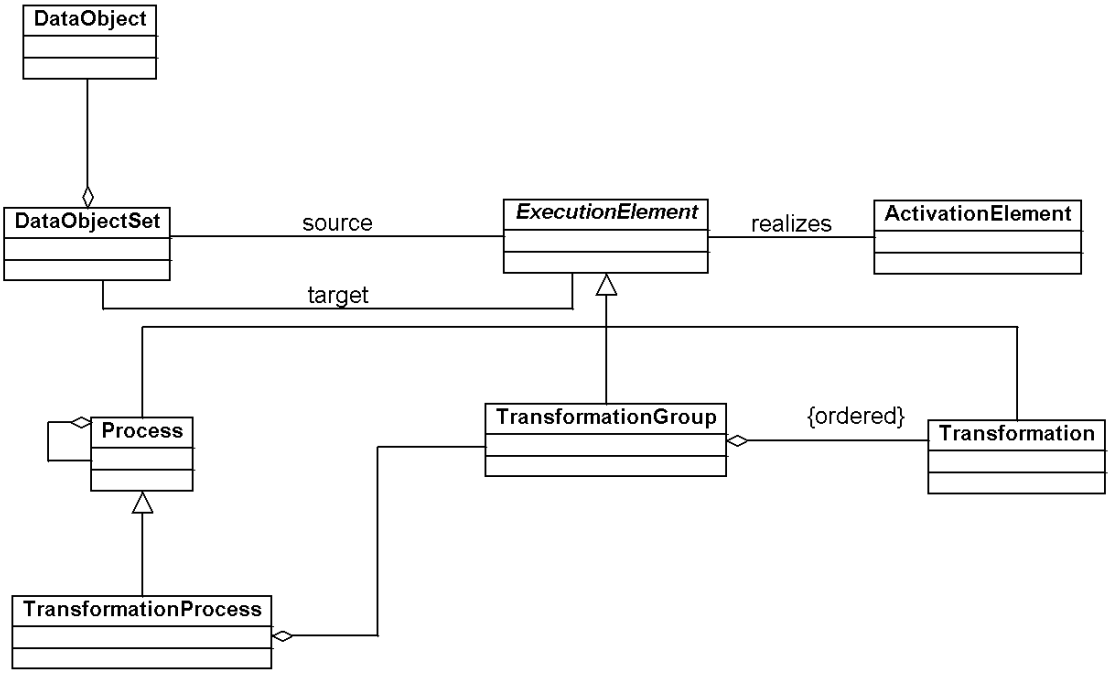

Abbildung 12: Transformationsprozesse

Multidimensionale Datenbanken

In Abbildung 13 ist ein allgemeines Modell mit den wichtigsten Elementen multidimensionaler Datenbanken gezeigt, wobei *Schema* ein Behälter ist, der alle Elemente des Modells direkt oder indirekt beinhaltet Das S*chema* wiederum besteht aus einer Aggregation der UML-Klasse *Class*, diese wiederum stellt die Oberklasse zu *Cube* und *Dimension* dar. Ein *Cube* selbst besteht über eine Aggregation aus mehren *Dimensionen*. Auf einer *Dimension* sind nun mehrere Hierarchien definiert, dargestellt durch eine Aggregation. Eine *Hierarchie* ihrerseits besteht aus einer geordneten Liste von Dimensionsobjekten. Die Klasse *Dimensionsobject* und die Klasse der Kenngrößen *Measure* sind eine Spezialisierung der Klasse *Attribute*. Über die *ordered*-Verbindung zwischen *Class* und A*ttribute* kann nun modelliert werden, welche Dimensionsobjekte zu welchen Dimensionen und welche Kennzahlen zu welchen Würfeln gehören.

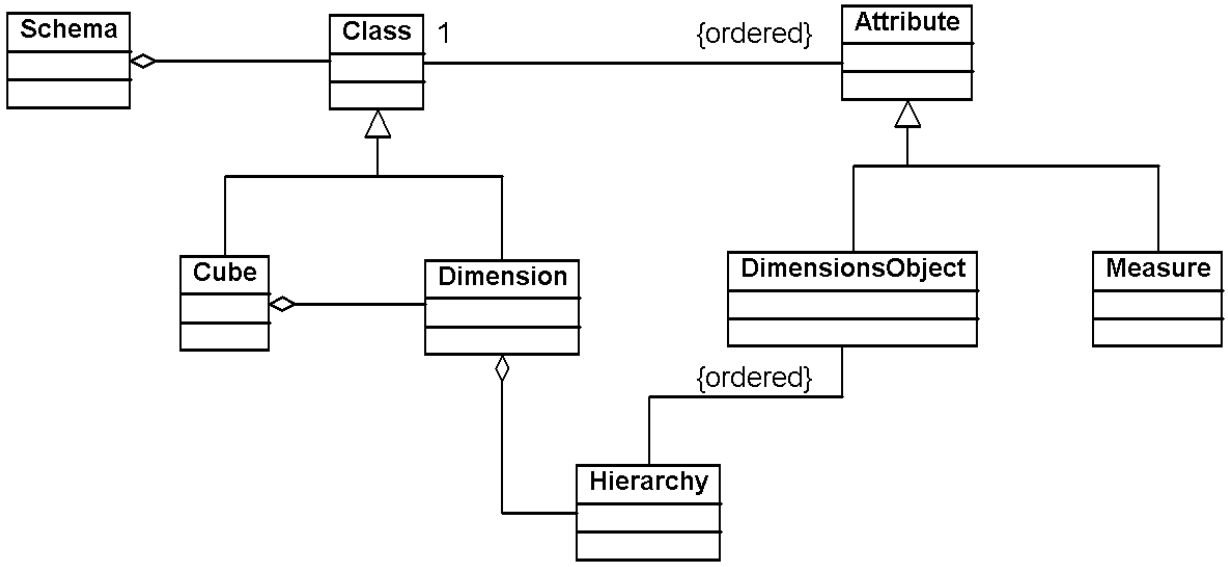

Abbildung 13: Multidimensionales Schema

3.4.2 Personen, Organisationen und Aufgaben

In Abbildung 14 ist die Modellierung von Personen, Organisationsstrukturen und Aufgaben zu sehen. Alle handlungsfähigen Akteure wie Personen, Organisationseinheiten, Werkzeuge und Systeme sind in der zentralen Klasse *Actor* zusammengefasst. Die Akteure ihrerseits sind zuständig für die Ausführung einer Aufgabe (T*ask*). Die Ausführung geschieht wiederum über einen P*rocess*. Die Zugriffrechte sind dem Akteur mittels einer Rolle (R*ole*) zugeordnet.

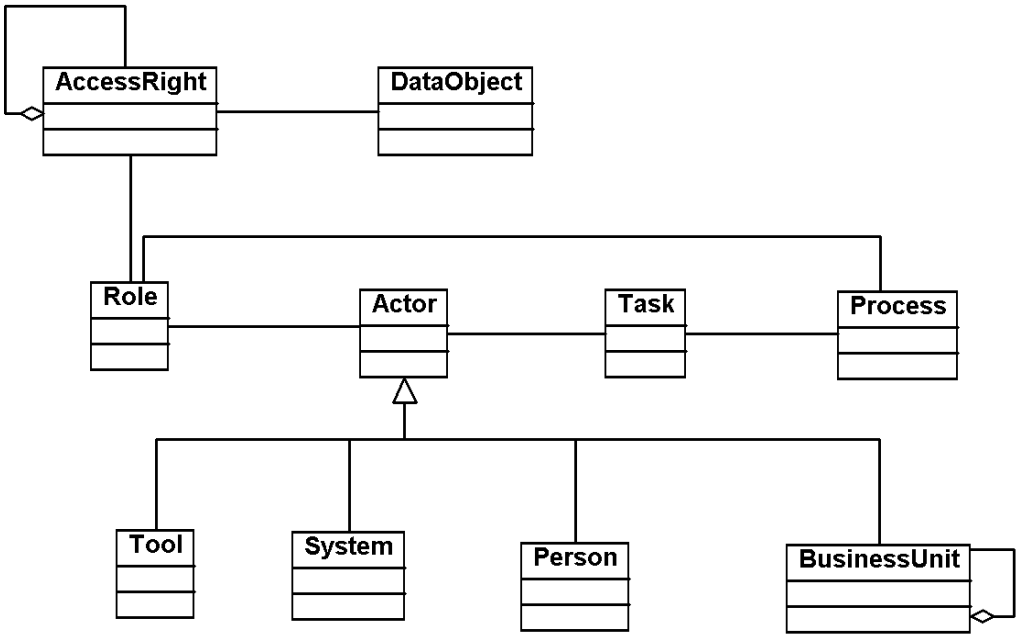

Abbildung 14: Personen, Organisation und Aufgaben

3.4.3 Business-Metadaten

Alle hier in diesem Ausschnitt und in Abbildung 15 für Business-Metadaten vorgestellten Klassen sind zusammengefasst in der Klasse *BusinessObject*. Es folgt eine Auflistung dieser Klassen mit einer kurzen Beschreibung:

- *BusinessTerm*: Ein Begriff, ein Wort oder Ausdruck mit einer Bedeutung für den Endanwender.
- *Terminology*: Sie fasst mehrere BusinessTerms zu einer logischen Einheit zusammen. Terminologien können auch aus Terminologien bestehen.
- *Reports*: Auswertungen wie z.B. Listen, Diagramme etc.
- *BusinessRule*: Es existieren zwei Arten von Geschäftsregeln:
    - *ActionRule*: Vorbedingungen die erfüllt sein müssen, damit eine Aktion ausgeführt werden kann.
    - *InferenceRule*: Beschreibt die Herleitung von Domänenwissen, welches nicht explizit gespeichert wird.
- *BusinessGoal*: Definiert ein Ziel für eine Geschäftseinheit
- *BusinessFigure*: Kennzahlen, die es erlauben die Effizienz eines Prozesses zu messen.

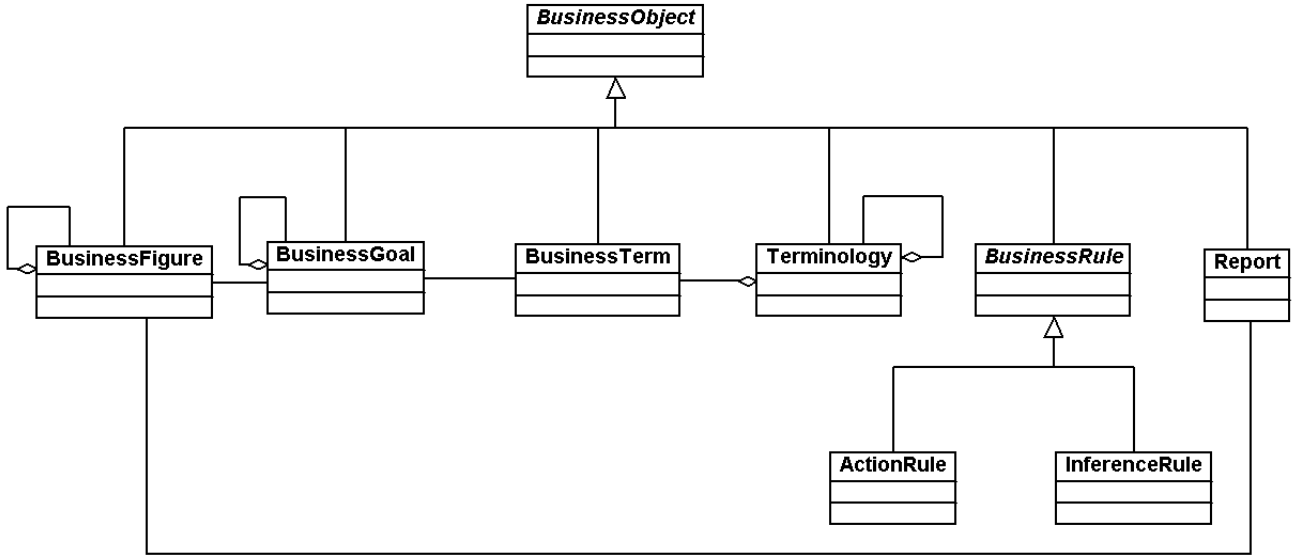

Abbildung 15: Business-Metadaten

3.4.4 Abstraktionsstufen

Eine mögliche Aufteilung in Abstraktionsstufen für die eben beschriebenen Modelle stellt *Beschreibung, Spezifikation* und *Realisierung* dar.
Wobei *Beschreibung* als Text in natürlicher Sprache und die *Spezifikation* in einer formalen Darstellung wie zum Beispiel Syntaxbäume zu finden sein könnte. Die Darstellung der *Realisierung* könnte durch eine Assoziationsverbindung zu einer anderen Klasse des Metamodells erfolgen.

3.4.5 Zusammenfassung

In diesem Kapitel wurde die Rolle der Metadaten für das Data Warehouse betrachtet. Es wurde festgestellt, dass kein prinzipieller Unterschied zwischen Metadaten für das Data Warehouse und allgemeinen Metadaten besteht. Danach wurde eine Einteilung der Metadaten in 4 Ebenen vorgenommen, um die Modellierung der Metadaten zu vereinfachen. Es wurde auf die Anforderungen an die Metadaten – aufgeteilt in Anforderungen an die Funktionalität und Architektur – eingegangen. Und obwohl noch kein einheitlicher Standard akzeptiert ist, wurde neben diversen vorhandenen Austauschstandards auch der CWM (Common Warehouse Metastandard) betrachtet.
Abschließend wurden die verschiedenen Teilaspekte zum Entwurf eines Schemas zur Verwaltung der Metadaten behandelt.

4 Zusammenfassung

In Kapitel 2 dieser Arbeit wurden Realisierungsmöglichkeiten für die physische Umsetzung des multidimensionalen Datenmodells diskutiert. Es wurde hier die Abbildung des multidimensionalen Modells auf ein relationales Datenbanksystem betrachtet. Hier liegen die Schwierigkeiten in der Transformation und der Konvertierung der multidimensionalen Strukturen, der Übersetzung bei der Anfrageverarbeitung sowie dem Semantikverlust durch die Konvertierung. Daneben existiert die Möglichkeit der direkten Verwendung eines multidimensionalen Datenbanksystems. Das Problem dieser Systeme liegt in der fehlenden Standardisierung des multidimensionalen Modells sowie dem Fehlen einer einheitlichen Anfragesprache.
Mittlerweile haben sich die relationalen Systeme weitgehend durchgesetzt. Die Gründe hierfür sind in der weit verbreiteten und ausgereiften Technologie der relationalen Datenbanken zu sehen. Weiterhin zeigen sie ein besseres Skalierungsverhalten als multidimensionale Systeme, deren Stärken vor allem eine höhere Anfragegeschwindigkeit in Anwendungen mit geringem Datenumfang und eine intuitivere Darstellung der Daten sind. Sinnvoll ergänzen können sich beide Ansätze in einer kombinierten Architektur der so genannten hybriden Speicherung mit einem großen, relational implementierten Data Warehouse und davon abhängigen, kleineren multidimensionalen Systemen.
In Kapitel 3 wurden Metadaten für Data Warehouses betrachtet, dabei gilt, dass Metadaten üblicherweise eine heterogenere Struktur als Applikationsdaten haben. Sie beschreiben Daten- und Systemaspekte auf unterschiedlichen Abstraktionsstufen unter Verwendung verschiedener Formalisierungsgrade. Damit wird erreicht, dass alle Anwender und alle Softwarekomponenten des Data-Warehouse-Systems unterstützt werden und eine einheitliche Sicht auf die Daten haben. Dies stellt hohe Anforderungen an die Flexibilität der Speicherungs- und Austauschmechanismen. Der geforderte Funktionsumfang eines Metadatenmanagementsystems im Bereich des Data Warehousing unterscheidet sich prinzipiell nicht von einem System zur Verwaltung allgemeiner Metadaten. Als besondere Voraussetzungen zur erfolgreichen Integration eines Metadatenmanagementsystems in ein Data-Warehouse-System sind allerdings ausgereifte Benutzerführung sowie Interoperabilität zu nennen. Im Gegensatz zum umfassenden Angebot an Datenbeschaffungs- und Analysewerkzeugen gibt es im Bereich der allgemein einsetzbaren Metadatenverwaltung nur wenige Anbieter. Auch ist die Einigung auf ein einheitliches Format zur Repräsentation und Austausch von Metadaten noch nicht absehbar. Den vorhandenen Standards gemeinsam ist allerdings die Verwendung von UML zur Repräsentation und XML als Basis für ein Austauschformat. Dies weckt die Hoffnung, dass eine Vereinheitlichung erreicht werden kann.

Referenzen

[BaGü01] Bauer, A., Günzel, H.: Data Warehouse Systeme dpunkt.verlag, Heidelberg, 2001

[Bern98] Bernstein, P.A.: Repositories and Object Oriented Databases. In: *SIGMOD Record*, Vol. 27, No. 1, 1998, S. 88-96

[CWM01] o.V.; *Common Warehouse Metamodel (CWM) Specification,* 2001, elektronisch verfügbar unter: http://www.omg.org/docs/ad/01-02-01.pdf, Aufruf am 03.07.2003

[GBLP96] Gray, J.; Bosworth, A.; Layman, A.; Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: *Proceedings of the 12th International Conference on Data Data Engineering* (ICD'96, New Orleans (LA), USA, 26. Feb. – 1. März), 1996, S. 152-159

[Kimb96a] Kimball, R.: *The Data Warehouse Toolkit, John Wiley & Sons Inc.,* New York, 1996, S. 100-105

[Kimb96b] Kimball, R.: Slowly changing dimensions. Unlike OLTP systems, data warehouses can track historical data. In: *DBMS online 9* (4), 1996, elektronisch verfügbar unter: http://www.dbmsmag.com/9604d05.html, Aufruf am 31.10.1999

[KnMy96] Knolmayer, G.; Myrach, T.: Zur Abbildung zeitbezogener Daten in betrieblichen Informationssystemen. In: *Wirtschaftsinformatik 38 (1)*, 1996, S. 63-74

[KRRT98] Kimball, R.; Reeves, L.; Ross, M.; Thornwaite, W.: *The Data Warehouse Lifecycle Toolkit.* Wiley, 1998

[Rond03] Rondot, R.; *Anfragesprachen für On-Line Analytical Processing (OLAP),* 2003, elektronisch verfügbar unter: http://wwwdvs.informatik.uni-kl.de/courses/seminar/SS2003/ausarbeitung3.pdf, Aufruf am 03.07.2003

[Seil03] Seiler, M.; *OLAP & Data Warehousing, eine Einführung,* 2003, elektronisch verfügbar unter: http://wwwdvs.informatik.uni-kl.de/courses/seminar/SS2003/ausarbeitung1.pdf, Aufruf am 03.07.2003

[SQL99] o.V.: ANSI/ISO/IEC 9075-2:1999 Information technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation) ISO, 1999