Logische Modelle für On-Line Analytical Processing

Burkhard Schäfer

27. Juni 2003

Inhaltsverzeichnis

1 Einführung

In Unternehmen, deren Geschäftsprozesse DV-gestützt abgewickelt werden, fallen große Mengen von Daten an. Es werden daher Verfahren benötigt, diese Daten geeignet zu analysieren, um daraus Informationen zu gewinnen und Entscheidungen zu stützen.

Der Einsatz eines *Data Warehouse* ermöglicht es, die anfallenden Daten sinnvoll zusammen zu tragen und abzulegen. Mittels *On-Line Analytical Processing (OLAP)* können nun interaktive Analysen auf einem solchen Data Warehouse durchgeführt werden, um die gewünschten Einsichten zu erhalten.

Da die in einem Data Warehouse abgelegten Datensätze in der Regel sehr detailliert sind, bietet OLAP Mechanismen an, Daten geeignet zu aggregieren und so Informationen zu liefern, die als Grundlage für Entscheidungen dienen können.

OLAP wird häufig zur Analyse von Verkaufsdaten eingesetzt. Dies geschieht, um die Leistungsfähigkeit einer Unternehmung bewerten zu können. Die im Folgenden verwendeten Beispiele orientieren sich an dieser Thematik.

Die Kapitel 2 und 3 dieser Ausarbeitung stellen die logischen OLAP-Konzepte auf anschauliche Art und Weise dar. In Kapitel 4 wird auf die notwendigen theoretischen Grundlagen eingegangen. Es wird hierfür ein geeignetes Datenmodell definiert und die darauf erforderlichen Operationen beschrieben. Kapitel 5 stellt eine Zusammenfassung dar.

2 Mehrdimensionale Datenobjekte

Betrachtet man die digitale Repräsentation eines Geschäftsvorfalls in einer Datenbank, so finden sich eine Reihe von Attributen, die den Vorfall genauer beschreiben. Diese Attribute können auch Referenzen darstellen. In einer relationalen Datenbank kann so mittels Fremdschlüsselbeziehungen auf Daten verwiesen werden, die in anderen Tabellen stehen.

Um eine Untersuchung dieser Vorfälle zu ermöglichen, ist es wünschenswert, alle Informationen, die einen Vorfall charakterisieren, zusammenzuführen. Im relationalen Modell entspricht dies einem Verbund über alle Tabellen. So können aus den Rohdaten der Vorfälle relativ einfach aggregierte Daten (Summen, Durchschnitt, ...) berechnet werden.

Abbildung 1 zeigt einige relationale Tabellen, deren Inhalte Geschäftsvorfälle beschreiben.

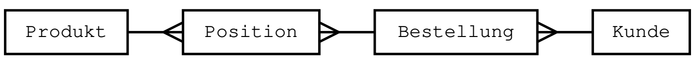

Abbildung 1: Relationales Datenbankschema der Rohdaten

Durch Denormalisierung ergeben sich Tupel, die Attribute aus allen Quellrelationen enthalten.

```
Sale (id, date, ...,                          -- aus Bestellung
      quantity, ...,                          -- aus Position
      fname, lname, city, country, ...,       -- aus Kunde
      productname, price, ... )               -- aus Produkt
```

Es gilt nun, zu entscheiden, welche der verfügbaren Attribute in eine Analyse einbezogen werden sollen. Unter Berücksichtigung dieser Attribute werden nun die *Maße* und *Dimensionen* des Datenmodells bestimmt und in einem sog. *Hypercube* dargestellt.

2.1 Hypercubes

Unter einem *Hypercube* versteht man einen n-dimensionalen Würfel, der aus Zellen besteht, die aggregierte Werte (*Maße*) beinhalten. Über verschiedene Operationen ist es möglich, innerhalb des Hypercubes zu navigieren und die enthaltenen Daten in unterschiedlichen Detaillierungsgraden zu betrachten.

Maß (engl.: measure). Als *Maße* werden Attribute bezeichnet, deren Wert von anderen Attributen abhängt. Auf diesen Attributen werden in der Regel Aggregatsfunktionen berechnet. Die Auswahl der aggregierten Tupel lässt sich vom Benutzer zur Laufzeit bestimmen. Als Maße werden in der Regel die Attribute verwendet, die untersucht werden sollen. Im Beispiel bietet sich hierzu der `Verkaufserlös` an.

Dimension. Unter einer Dimension versteht man Attribute, die ein Maß näher bestimmen (“determining attributes”). Bei der Untersuchung von Verkaufsabschlüssen eignen sich z.B. `Zeit`, `Ort` oder `Produkt` als Dimensionen.

2.2 Hierarchien

Die gewählten n Dimensionen spannen einen n-dimensionalen Hypercube auf. Dabei bietet es sich häufig an, *Hierarchien* auf Dimensionen zu definieren, die die Navigation und die Bildung sinnvoller Aggregate unterstützen. Hierarchien bestehen aus einer Menge von *Mitgliedern* (engl.: members), die in einer baumartigen Struktur organisiert sind. Die Mitglieder stellen dabei die Blätter des Baumes dar. Hierarchien verkörpern also Part-Of-Relationen.

Für die Dimension `Ort` bietet sich die in Abb. 2 angegebene Hierarchie an. Abbildung 3 zeigt, wie eine dazu passende Wertebelegung aussehen könnte.

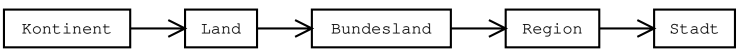

Abbildung 2: Hierarchie auf der Dimension `Ort`

Eine Dimension kann mehr als eine Hierarchie besitzen. Dies ist sinnvoll, wenn unterschiedliche Kriterien zur Navigation herangezogen werden sollen. So könnte

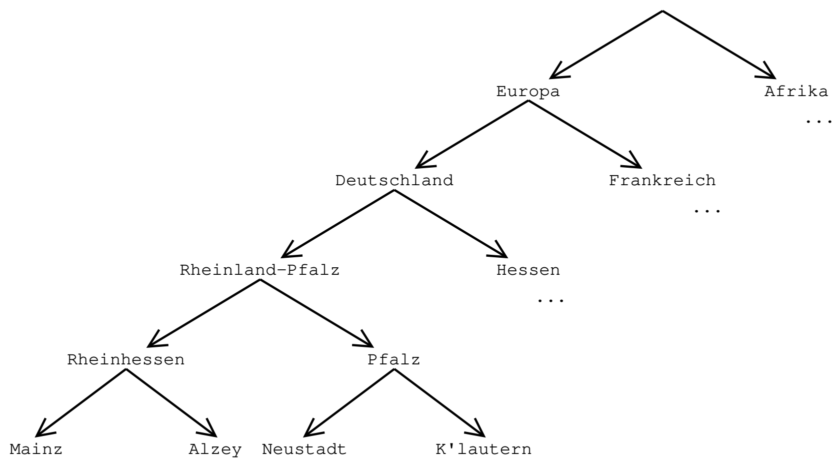

Abbildung 3: Wertebelegung für Beispielhierarchie

die Dimension `Ort` auch nach geographischen Merkmalen (Insel / Festland, Bergregion / Flachland), gesellschaftlichen Gesichtspunkten (Einwohnerzahl, Bevölkerungsdichte) oder politischen Randbedingungen (Planwirtschaft / Marktwirtschaft) klassifiziert werden.

Zur Navigation innerhalb einer Hierarchie stehen die Operationen *Drill-Down* und *Roll-Up* (siehe Abschnitt 4.5.3) zur Verfügung. Somit können Granularität und Aggregationsebenen der betrachteten Daten interaktiv verändert werden. [AGS98]

3 OLAP-Operationen

Sobald ein OLAP-Würfel erstellt ist, stehen einige Operationen zur Verfügung, die Daten manipulieren, aggregieren und darstellen.

3.1 Pivoting

Beim Pivoting handelt es sich um eine Operation, die hauptsächlich der Darstellung dient. Sie erlaubt es, die Ausrichtung der Dimensionen eines Würfels zu verändern, also Dimensionen zu vertauschen [AGS98]. Die Daten an sich werden nicht verändert, lediglich ihre Anordnung im Würfel. Abbildung 4 veranschaulicht dies an einem Beispiel.

3.2 Slicing und Dicing

Häufig ist für eine Analyse nur eine Teilmenge der im Hypercube enthaltenen Werte interessant. Die Einschränkung der betrachteten Daten wird in OLAP als *Slicing* und *Dicing* [AGS98] bezeichnet.

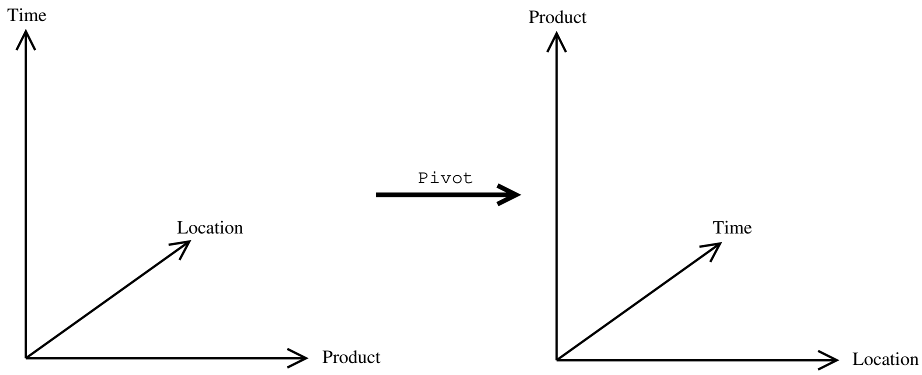

Abbildung 4: Pivot-Operation

Dabei wird eine Beschneidung des Hypercube entlang der Dimensionen vorgenommen. Für jede Dimension kann angegeben, welche Werte berücksichtigt werden sollen. Manche OLAP-Konzepte erlauben auch die Angabe von Intervallen sowie die Selektion nach Rang, wodurch man die höchsten oder niedrigsten Werte erhält.

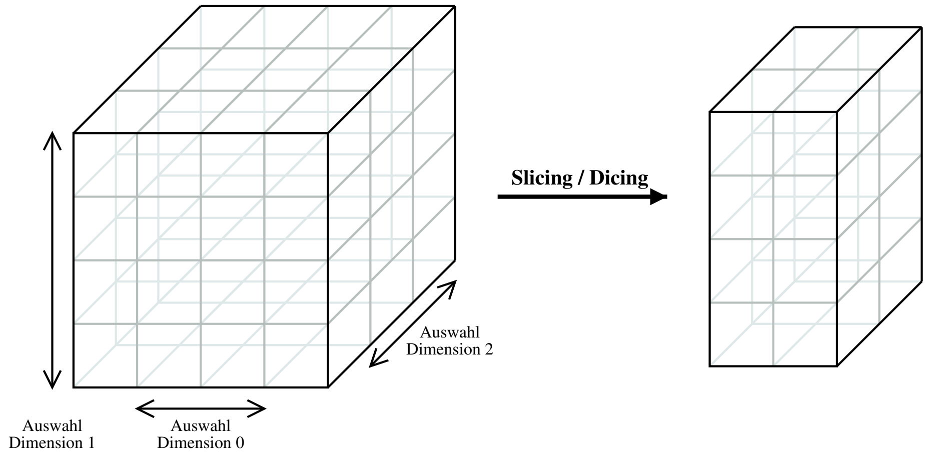

Abbildung 5: Slicing und Dicing

Wird nur auf einer Dimension eine Beschränkung definiert, spricht man von *Slicing*. Bei Restriktionen auf mehreren Dimensionen, spricht man von *Dicing*.

Abbildung 5 zeigt, wie aus einem Hypercube eine Auswahl auf den Wertebereichen der Dimensionen getroffen wird. Es entsteht der rechts dargestellte (kleinere) Cube, der die unerwünschten Daten ausblendet.

Bleibt nach Anwendung der beschriebenen Restriktionsoperation nur noch ein einzelner Wert für eine Dimension übrig, so kann diese zur Verbesserung der Übersicht entfernt werden. Diese Operation wird auch als *Destroy Dimension* bezeichnet.

3.3 Drill-Down und Roll-Up

Um Daten in unterschiedlichen Detaillierungsgraden zu betrachten, können die Operation *Drill-Down* und *Roll-Up* herangezogen werden. Sie erlauben es, innerhalb einer Dimension zwischen verschiedenen Aggregationsebenen zu navigieren. Diese Navigation orientiert sich in der Regel an einer Hierarchie.

Durch *Drill-Down* und *Roll-Up* werden implizit die zu aggregierenden Rohdaten ausgewählt, die in einer Zelle des Hypercube dargestellt werden sollen. Wie in Abbildung 6 zu erkennen ist, sorgt *Roll-Up* für eine stärker aggregierte Sicht, während *Drill-Down* die Daten in einer detaillierteren Darstellungsweise anbietet, also quasi als "Zoom-Funktion" fungiert.

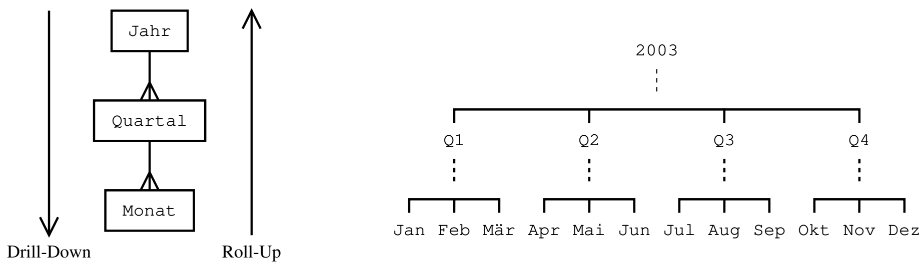

Abbildung 6: Navigation durch Hierarchien mittels Drill-Down und Roll-Up

Gelegentlich wird unter Drill-Down zusätzlich die Hinzunahme weiterer Dimensionen verstanden. So könnten bei einer Verkaufsdatenanalyse beispielsweise erst ab einem bestimmten Detaillierungsgrad die unterschiedlichen Kundengruppen mit betrachtet werden.

In [GSS01] wird mit *Intelligent Roll-Ups* ein Ansatz beschrieben, mit dem der Benutzer in der Hierarchienavigation gestützt wird. Damit ist es möglich, von einer Norm abweichende Werte durch geschicktes Ausführen von Roll-Ups genauer zu untersuchen.

3.4 Push und Pull

In [AGS98] werden die Operationen *Push* und *Pull* eingeführt. Sie erlauben es, Dimensionen in Maße zu konvertieren (Push) und umgekehrt (Pull). Solche Operationen werden benötigt, wenn anfangs noch nicht fest steht, welche Attribute als Maße und welche als Dimensionen betrachtet werden sollen.

Push. Bei Anwendung der Push-Operation werden die Werte einer bestimmten Dimension, die zu einem Tupel gehören, dem Tupel hinzugefügt. Dies wird in Abbildung 7 veranschaulicht. Optional kann die betroffene Dimension daraufhin aus dem Hypercube gelöscht werden.

Pull. Die Pull-Operation erzeugt aus einem Maß eine neue Dimension. Das Maß wird dabei aus dem Hypercube entfernt und dient nun als zusätzliche Koordinate der Zelle, aus der es stammt.

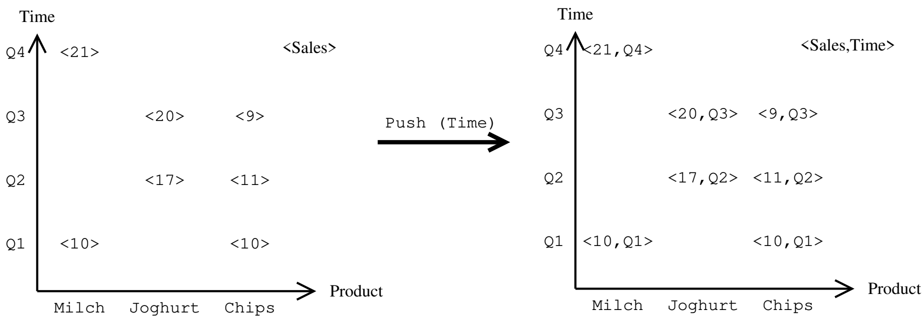

Abbildung 7: Funktionsweise von *Push* ohne Löschen der Dimension

4 Formale Grundlagen

In diesem Abschnitt wird ein Datenmodell vorgestellt, mit dem Hypercubes repräsentiert werden können. Passend dazu werden einige Operationen eingeführt, mit denen die aus Kapitel 3 bekannte OLAP-Funktionalität abgebildet werden kann.

Das Modell orientiert sich an [AGS98], wurde jedoch gegenüber der Quelle vereinheitlicht und erweitert.

4.1 Datenmodell

Die Grundidee des vorgestellten Datenmodells ist die Gleichbehandlung von Maßen und Dimensionen. Erst bei der Darstellung soll entschieden werden, welche Attribute als Maße und welche als Dimensionen betrachtet werden. Aus dieser Idee ergeben sich eine Reihe von Vereinfachungen gegenüber dem in [AGS98] präsentierten Ansatz. Diese werden im Verlauf dieses Kapitels näher erläutert.

Ein Hypercube ist ein Tupel, das sich wie folgt zusammensetzt:

$$C = (k, dn, dt, e, v)$$

Dabei besitzt der Hypercube C folgende Eigenschaften:

Größe	Bedeutung	Beispiel
k	Anzahl der Attribute	5
$dn = (dn_1, \ldots, dn_k)$	Namen der Attribute	('Datum', 'Kunde', 'Ort', 'Produkt', 'Preis')
$dt = (dt_1, \ldots, dt_k)$	Typen der Attribute	(`Date`, `String`, `String`, `String`, `Float`)
e	Menge der Tupel	{(2002-01-01, 'Müller', 'München', 'Frischmilch', 0.80), (2002-01-02, 'Meier', 'Hamburg', 'Joghurt', 0.40), ...}
v	Anzahl Attribute, die als Dimension betrachtet werden	4

Bei den Einträgen k, dn, dt und v handelt es sich um Metadaten, die den Hypercube näher beschreiben. Der in der Tabelle beispielhaft skizzierte Cube verfügt also über fünf Attribute, von denen die ersten vier als Dimension und das letzte als Maß betrachtet werden.

Ein Hypercube C ist stets Element der Menge $Cube$, die alle Hypercubes enthält. Es gilt also

$$C \in Cube$$

$Cube$ist definiert als

$$Cube = \mathbb{N}_+ \times DN \times DT \times E \times \mathbb{N}_0$$

mit

$$\begin{aligned} DN &= String^k \\ DT &= Dom^k \\ E &= dt_1 \times \cdots \times dt_k \end{aligned}$$

Mit Dom wird hier die Menge aller Datentypen bezeichnet. Künftig bezeichnen wir mit dn ein Element der Menge DN, mit dt ein Element der Menge DN und mit e ein Element von E.

Es wird festgelegt, dass ein Hypercube mindestens ein Attribut besitzen muss, also $k \geq 1$. Weiterhin können höchstens k Attribute als Dimensionen betrachtet werden. Es gilt also zusätzlich die Voraussetzung $v \leq k$.

Um einen Hypercube zu erzeugen, kann eine relationale Tabelle mehr oder weniger direkt in die gegebene Struktur übernommen werden. Ist eine Tabelle mit n Spalten gegeben, so besitzt auch der Cube n Attribute, also $k = n$. Das Tupel dn wird mit den Namen der Spalten belegt. Die Typen der Spalten werden in dt gespeichert. Schließlich werden alle Tupel der Relationalen Tabelle in die Menge e übernommen.

4.2 Dimensionen und Maße

In dem vorgestellten Datenmodell werden Dimensionen und Maße gleich behandelt: Sie werden als Attribute dargestellt, deren Werte in der Menge e eines Hypercubes abgelegt werden.

Soll jedoch ein Hypercube visualisiert werden, ist eine Unterscheidung zwischen Dimensionen und Maßen notwendig. Diese wird durch die Eigenschaft v des Würfels getroffen. Diese beschreibt, welche Attribute als Dimensionen betrachtet werden. Dabei wird stets von links an gezählt. Bei $v = 0$ ist jedes Attribut ein Maß; bei $v = k$ wird jedes Attribut als Dimension betrachtet.

$$\begin{array}{ll} v = 1 & dn = (\underline{Product}, Customer, Location, Time, Price) \\ v = 3 & dn = (\underline{Product, Customer, Location}, Time, Price) \\ v = 5 & dn = (\underline{Product, Customer, Location, Time, Price}) \end{array}$$

Abbildung 8: Dimensionen bei unterschiedlichen Werten für v

Abbildung 8 zeigt beispielhaft, wie sich aus dem Wert von v die Unterteilung der Attribute ergibt. Die als Dimension betrachteten Attribute sind unterstrichen markiert.

Mittels der Operationen *pull* und *push* kann v in- bzw. dekrementiert werden. Zuvor kann die Anordnung der Attribute mittels der *swap*-Funktion geändert werden. So kann frei bestimmt werden, welche Attribute in welcher Reihenfolge von den genannten Operationen betroffen sind.

Ein großer Vorteil des vorgestellten Datenmodells ist, dass für die Konvertierung zwischen Maßen und Dimensionen kein großer Aufwand betrieben werden muss. Gerade bei interaktiven Anwendungen dürften sich hierdurch spürbare Leistungssteigerungen ergeben.

4.3 Operationen

4.3.1 swap

$$swap : Cube \times \mathbb{N}_+ \times \mathbb{N}_+ \to Cube$$

Übersicht: $swap(C, i, j)$ vertauscht im Hypercube C die Attribute i und j.

Definition:

$$swap : \frac{(k, (dn_1, \ldots, dn_i, \ldots, dn_j, \ldots, dn_k), (dt_1, \ldots, dt_j, \ldots, dt_j, \ldots, dt_k)}{(k, (dn_1, \ldots, dn_j, \ldots, dn_i, \ldots, dn_k), (dt_1, \ldots, dt_j, \ldots, dt_i, \ldots, dt_k)} \cdots$$
$$\cdots \frac{(e_1, \ldots, e_i, \ldots, e_j, \ldots, e_k), v)}{(e_1, \ldots, e_j, \ldots, e_i, \ldots, e_k), v)}$$

Beim Betrachten der Inferenzregel fällt auf, dass in jedem der drei Tupel dn, dt und e jeweils die Stellen i und j vertauscht werden. Es werden also die Daten und die Metadaten angepasst. Durch Verknüpfung mehrerer Funktionsaufrufe der Art *swap* können alle Attribute des Hypercube beliebig vertauscht werden. Dies ist beispielsweise für *Pivoting* (siehe 4.5.1) nützlich.

4.3.2 restrict

$$restrict : Cube \times \mathbb{N}_+ \times Praed \to Cube$$

Übersicht: $restrict(C, i, P)$ entfernt alle Elemente aus dem Hypercube C, für die das Attribut i das Prädikat P nicht erfüllt.

Definition:

$$restrict : \frac{(k, dn, dt, e, v)}{(k, dn, dt, \{\widetilde{e} \mid \widetilde{e} \in e \wedge P(\widetilde{e}_i)\}, v)}$$

Die Operation *restrict* wird benötigt, um durch Anwendung von Restriktionen die Daten auswählen zu können, die betrachtet werden sollen. In der Ergebnismenge sind nur die die Elemente $\widetilde{e} \in e$ enthalten, für die das Prädikat P für das Attribut i wahr ist, also der Ausdruck $P(\widetilde{e}_i)$ erfüllt wird.

Diese Funktion kommt für *Slicing* und *Dicing* (siehe auch 3.2) zum Einsatz.

4.3.3 transform

$$transform : Cube \times \mathbb{N}_+ \times \mathcal{F}_{dn} \times \mathcal{F}_{dt} \times \mathcal{F}_e \to Cube$$

Übersicht: $transform(C, i, f_{dn}, f_{dt}, f_e)$ wendet die gegebenen Transformationsfunktionen auf die Dimension i des gegebenen Hypercubes C an.

Definition:

$$transform : \frac{(k, (dn_1, \ldots, dn_i, \ldots, dn_k), (dt_1, \ldots, dt_i, \ldots, dt_k),}{(k, (dn_1, \ldots, f_{dn}(dn_i), \ldots, dn_k), (dt_1, \ldots, f_{dt}(dt_i), \ldots, dt_k),} \cdots$$
$$\cdots \frac{\{(e_{1,j}, \ldots, e_{i,j}, \ldots, e_{k,j}), \ldots\}, v)}{\{(e_{1,j}, \ldots, f_e(e_{i,j}), \ldots, e_{k,j}), \ldots\}, v)}$$

mit

$$\begin{aligned} f_{dn} : String \to String \qquad & f_{dn} : dn_i \mapsto \widetilde{dn_i} \\ f_{dt} : Dom \to Dom \qquad & f_{dt} : dt_i \mapsto \widetilde{dt_i} \\ f_e : dt_i \to \widetilde{dt_i} \qquad & f_e : e_i \mapsto \widetilde{e_i} \end{aligned}$$

Die hier vorgestellte Funktion $transform$ erlaubt es, die Werte eines bestimmten Attributs i mittels einer Abbildungsfunktion f_e umzusetzen. Da sich dabei nicht nur der Wert sondern auch der Datentyp eines Attributs ändern kann, muss die Transformation auch auf Ebene der Metadaten vollzogen werden. Über die Funktion f_{dn} wird hierzu der Name des Attributs transformiert, während f_{dt} den Datentyp anpasst. Auch an der Signatur von f_e ist zu erkennen, dass der Quelltyp vom Zieltyp abweichen kann.

Eine Anwendung von $transform$ ist die Navigation in Hierarchien. Dies wird in Abschnitt 4.4 genauer beschrieben.

4.3.4 aggregate

$$aggregate : Cube \times \mathcal{F}_{aggr}^{k-v} \to Cube$$

Übersicht: aggregiert alle Tupel des Hypercubes C, die sich in den Dimensionsattributen gleichen, mittels der übergebebenen Aggregationsfunktionen f_{aggr_i}.

Definition:

$$aggregate : \frac{(k, dn, dt, e, v)}{(k, dn, dt, e', v)}$$

mit der Voraussetzung

$$v > 0$$

wobei e' mit folgendem Algorithmus berechnet wird:

```
e' := ∅
for d ∈ project(1) ∘ ... ∘ project(v) ∘ C do
{
    // für jedes Maß...
    for i ∈ (v + 1, ..., k) do
    {
        // Multimenge (mit Duplikaten) aller Werte des
        // Maßes i erzeugen
        values_i := {t_i | t ∈ e ∧ (t_1 = d_1 ..., t_v = d_v)}

        // Aggregatsfunktion auf Multimenge anwenden
        aggr_i := f_aggr_i(values_i)
    }

    // neues Tupel zusammensetzen aus Dimensionen
    // und aggregierten Maßen
    newTuple := (d_1, ..., d_v, aggr_v+1, ..., aggr_k)

    // neues Tupel der Ergebnismenge hinzufügen
    e' := e' ∪ newTuple
}

return (e')
```

Die Funktion *aggregate* fasst jeweils die Tupel zusammen, die in allen Dimensionsattributen übereinstimmen. Dies wird erreicht, in dem zunächst durch Projektion alle vorkommenden Wertekombinationen für die Dimensionsattribute ermittelt werden. Es werden also die Tupel nach ihren Dimensionsattributen gruppiert.

Für jede dieser Gruppierungen werden dann die Wertemengen der Maße bestimmt. Diese Wertemengen werden mit der Funktion f_{aggr_i} zu einem einzelnen Wert aggregiert. Die aggregierten Werte aller Maße werden nun zusammen mit den Werten der Dimensionsattribute zu einem neuen Tupel zusammengesetzt, welches in die Ergebnismenge eingefügt wird. Pro Gruppierung, also pro Zelle im OLAP-Würfel, entsteht also ein Tupel mit den aggregierten Maßen.

4.3.5 project

$$project : Cube \times \mathbb{N}_+ \to Cube$$

Übersicht: $project(C, i)$ entfernt das Attribut i aus dem Hypercube C.

Definition:

$$project : \frac{(k, (dn_1, \ldots, dn_k),}{(k-1, (dn_1, \ldots, dn_{i-1}, dn_{i+1}, \ldots, dn_k),} \cdots$$

$$\cdots \frac{(dt_1, \ldots, dt_k), \{(e_{1,j}, \ldots, e_{k,j}), \ldots\}, v)}{(dt_1, \ldots, dt_{i-1}, dt_{i+1}, \ldots, dt_k), \{(e_{1,j}, \ldots, e_{i-1,j}, e_{i+1,j}, \ldots, e_{k,j}), \ldots\}, v')}$$

mit

$$\begin{array}{ll} v' = v & \text{falls } i > v \\ v' = v - 1 & \text{falls } i \leq v \end{array}$$

unter der Voraussetzung

$$k > 1 \wedge v < k$$

Die Funktion *project* liefert einen Hypercube, aus dem das Attribut i entfernt wurde. Dies geschieht, indem aus jedem Tupel $t \in e$ der i-te Wert entfernt wird. Ebenso wird den Metadaten-Tupeln dn und dt jeweils der i-te Eintrag entnommen. Da ein Attribut entfernt wird, muss auch die Anzahl der Attribute k dekrementiert werden.

$$\begin{array}{ll} k = 5 & dn = (\underline{Product, Customer, Location}, Time, Price) \\ & \downarrow project(C, 3) \\ k = 4 & dn = (\underline{Product, Customer}, Time, Price) \end{array}$$

Abbildung 9: Beispiel für *project*

Die Operation kann nur auf Hypercubes angewandt werden, die mehr als ein Attribut besitzen, da sonst der Ergebniswürfel kein Attribut besäße. Dies widerspräche der Definition eines Hypercubes (4.1).

Zusätzlich muss $v < k$ gelten, damit es nach dem Entfernen des Attributs i nicht mehr Dimensionen als Attribute geben kann. Es ist wichtig, zu beachten, dass e nur eine einfache Menge ist. Folglich werden bei der Projektion auch Duplikate entfernt.

Die Funktion *project* wird benötigt, wenn für eine interaktive Ansicht nur bestimmte Attribute betrachtet werden sollen. Die nicht benötigten Attribute können mit *project* vorher entfernt werden.

4.3.6 copy

$$copy : Cube \times \mathbb{N}_+ \rightarrow Cube$$

Übersicht: $copy(C, i)$ fügt dem Hypercube C ein neues Attribut $k + 1$ hinzu und füllt dieses mit den Werten des Attributs i.

Definition:

$$copy : \frac{(k, (dn_1, \ldots, dn_k), (dt_1, \ldots, dt_k), \{(e_{1,j}, \ldots, e_{k,j}), \ldots\}, v)}{(k+1, (dn_1, \ldots, dn_k, dn_i), (dt_1, \ldots, dt_k, dt_i), \{(e_{1,j}, \ldots, e_{k,j}, e_{i,j}), \ldots\}, v)}$$

Als Rückgabewert stellt die Funktion *copy* einen Hypercube bereit, in dem das Attribut i dupliziert wurde. Die Tupel in der Wertemenge e werden um eine Stelle verbreitert, um den duplizierten Wert aufnehmen zu könne. Das neue

$$
\begin{array}{ll}
k = 5 & dn = (\underline{Product, Customer, Location}, Time, Price) \\
 & \downarrow copy(C, 3) \\
k = 6 & dn = (\underline{Product, Customer, Location}, Time, Price, Location)
\end{array}
$$

Abbildung 10: Beispiel für *copy*

Attribut wird dabei hinten, also an der Stelle $k+1$, angehängt. Hierdurch erhöht sich die Anzahl der Attribute im Würfel C, so dass k inkrementiert werden muss. Die Metadaten in dn und dt werden ebenfalls angepasst.

Abbildung 10 stellt vereinfacht dar, wie ein Attribut durch die *copy*-Funktion dupliziert wird. Der Einfachheit halber wurden nur die Auswirkung auf k und dn abgebildet.

4.3.7 push

$$push : Cube \rightarrow Cube$$

Übersicht: $push(C)$ betrachtet das Attribut an Stelle v künftig als Maß.

Definition:

$$push : \frac{(k, dn, dt, e, v)}{(k, dn, dt, e, v-1)}$$

unter der Voraussetzung

$$v > 0$$

Die Funktion $push$ dient dazu, eine zusätzliche Dimension künftig als Maß zu betrachten. Wie in Abschnitt 4.2 erläutert, bestimmt v, welche Attribute als Dimensionen betrachtet werden. Mit der $push$-Funktion wird v nun dekrementiert. Es wird also das vormals an Stelle v stehende Attribut künftig als Maß betrachtet.

$$
\begin{array}{ll}
v = 3 & dn = (\underline{Product, Customer, Location}, Time, Price) \\
 & \downarrow push \\
v = 2 & dn = (\underline{Product, Customer}, Location, Time, Price)
\end{array}
$$

Abbildung 11: Beispiel für $push$-Operation

In Abbildung 11 wird vereinfacht gezeigt, wie eine Dimension (`Location`) in ein Maß umgewandelt wird. Es fällt hierbei auf, dass die eigentlichen Daten für eine $push$-Operation nicht manipuliert werden müssen, sondern lediglich ein neuer Wert für v gesetzt werden muss.

Es ist zu beachten, dass $push$ nicht äquivalent zu der in 3.4 vorgestellten OLAP-Operation $push$ ist. In 4.5.4 wird der Zusammenhang zwischen den beiden Operationen genauer beleuchtet.

4.3.8 pull

$$pull : Cube \rightarrow Cube$$

Übersicht: $pull(C)$ betrachtet das Attribut an Stelle $v+1$ künftig als Dimension.

Definition:

$$pull : \frac{(k, dn, dt, e, v)}{(k, dn, dt, e, v+1)}$$

unter der Voraussetzung

$$v < k$$

Die *pull*-Operation ist das Gegenteil zu *push* (siehe 4.3.7). Sie dient dazu, ein bisher als Maß betrachtetes Attribut $v+1$ zukünftig als Dimension zu verwenden. Dies geschieht durch inkrementieren von v. Dazu muss $v < k$ vorausgesetzt werden, da ein Hypercube C niemals mehr Dimensionen als Attribute besitzen kann.

$$\begin{array}{ll} v = 2 & dn = (\underline{Product, Customer}, Location, Time, Price) \\ & \downarrow pull \\ v = 3 & dn = (\underline{Product, Customer, Location}, Time, Price) \end{array}$$

Abbildung 12: Beispiel für *pull*-Operation

Bei Betrachtung des Beispiels in Abbildung 12 wird deutlich, dass *pull* eine vorher ausgeführte *push*-Operation aus Abschnitt 4.3.7 rückgängig macht (vgl. Abb. 11).

4.4 Hierarchien

Wie bereits aus Abschnitt 2.2 bekannt, dienen Hierarchien dazu, die in einem Hypercube enthaltenen Daten auf unterschiedlichen Aggregationsebenen zu untersuchen. In der Regel benutzt man Hierarchien, die auf einer Dimension definiert sind. Hier werden Hierarchien allgemeiner, nämlich auf beliebigen Attributen, betrachtet. Diese Möglichkeit ergibt sich aus der äqivalenten Behandlung von Maßen und Dimensionen im vorgestellten Modell.

Ein Roll-Up, also das Betrachten einer höheren Aggregationsebene, besteht immer aus zwei Schritten:

Transformation. Die Datentupel werden in den Wertebereich der höheren Hierarchieebene transformiert (f_e). Außerdem werden die Metadaten angepasst (f_{dn}, f_{dt})

Aggregation. Die transformierten Daten werden aggregiert (f_{agg}).

In den folgenden beiden Abschnitten wird auf diese Schritte ausführlich eingegangen und die Realisierung im Modell erläutert.

Grundsätzlich können auf jedem Attribut Hierarchien definiert werden. Solche Hierarchien stellen eine Part-Of-Relation dar und bestehen aus einer oder mehreren Hierarchiestufen. Stufe 1 ist stets die triviale Hierarchieebene, nämlich die unmodifizierte Sicht auf die Dimension. Für jede weitere Hierarchieebene $n > 1$ müssen geeignete Abbildungsfunktionen definiert werden, um von Ebene $n - 1$ auf Ebene n zu gelangen. Eine Hierarchieebene n kann also erst erreicht werden, nachdem alle darunter liegenden Ebenen $1, \ldots, n - 1$ traversiert worden sind.

4.4.1 Transformation

Der erste Schritt in einem Roll-Up stellt die Transformation dar. Hierfür werden für jede Hierarchieebene $n > 1$ drei Funktionen definiert:

1. $f_{e_{n-1 \to n}}$ transformiert die Daten von Hierarchieebene $n - 1$ nach n
2. $f_{dn_{n-1 \to n}}$ transformiert den Namen des Attributs
3. $f_{dt_{n-1 \to n}}$ transformiert den Datentyp des Attributs

Im folgenden wird eine Beispielhierarchie auf der Dimension `Ort` konstruiert. Es sollen, wie in Abbildung 13 gezeigt, drei Hierarchieebenen eingeführt werden: `Stadt`, `Region` und `Bundesland`.

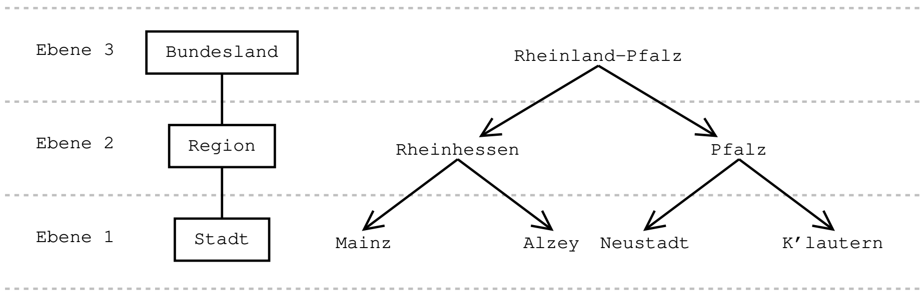

Abbildung 13: Ebenen der Hierarchie auf `Ort`

Hierzu werden für jede Hierarchieebene die drei Transformationsfunktionen angegeben:

Ebene 1. Da Ebene 1 die triviale Hierarchieebene ist, werden keine Funktionen benötigt. Streng genommen wird für jede Funktion die Identität verwendet.

Ebene 2.

$$
\begin{aligned}
f_{e_{1\to 2}} &: Mainz \mapsto Rheinhessen,\\
&\quad Alzey \mapsto Rheinhessen,\\
&\quad Neustadt \mapsto Pfalz,\\
&\quad Kaiserslautern \mapsto Pfalz\\
f_{dn_{1\to 2}} &: Stadt \mapsto Region\\
f_{dt_{1\to 2}} &: String \mapsto String
\end{aligned}
$$

Ebene 3.

$$
\begin{aligned}
f_{e_{2\to 3}} &: Rheinhessen \mapsto Rheinland-Pfalz,\\
&\quad Pfalz \mapsto Rheinland-Pfalz\\
f_{dn_{2\to 3}} &: Region \mapsto Bundesland\\
f_{dt_{2\to 3}} &: String \mapsto String
\end{aligned}
$$

Damit sind die Hierarchieebenen vollständig beschrieben. Es ist nun möglich, ausgehend von Ebene 1 iterativ beliebige Hierarchieebenen zu erreichen. Dabei werden sowohl die Daten als auch die Metadaten im Hypercube entsprechend umgesetzt.

Für jede Dimension können beliebig viele Hierarchien eingeführt werden. Es müssen hierzu lediglich für jede Ebene das notwendige Funktionstripel f_e, f_{dn} und f_{dt} definiert werden.

4.4.2 Berechnung von Aggregaten

Während der in 4.4.1 beschriebene Transformation kommt es häufig vor, dass mehrere Tupel auf die gleiche Zelle im OLAP-Würfel abgebildet werden. Dies ist dann der Fall, wenn die Werte aller als Dimension betrachteten Attribute übereinstimmen.

Betrachten wir einen Würfel $C \in Cube$ mit $v = 4$ und

$$dn = (\underline{Kunde, Produkt, Ort, Zeit}, Betrag)$$

der in der Dimension `Zeit` die Hierarchieebene `Monat` besitzt und folgende Datentupel enthält:

$$
\begin{aligned}
(Meyer, Joghurt, Frankfurt, April, 20.00)\\
(Meyer, Joghurt, Frankfurt, Mai, 25.00)\\
(Meyer, Milch, Frankfurt, April, 10.00)\\
(Meyer, Milch, Frankfurt, Juni, 8.00)
\end{aligned}
$$

Wird nun ein Roll-Up auf dem Attribut `Zeit` durchgeführt, um von der Hierarchiestufe Monat zur Stufe Quartal zu gelangen, werden die Daten durch die Funktion f_e wie folgt modifiziert:

$$(Meyer, Joghurt, Frankfurt, Q2, 20.00)$$
$$(Meyer, Joghurt, Frankfurt, Q2, 25.00)$$
$$(Meyer, Milch, Frankfurt, Q2, 10.00)$$
$$(Meyer, Milch, Frankfurt, Q2, 8.00)$$

Die ersten beiden Tupel liegen in der gleichen Zelle des Würfels, da sie in allen Dimensionen übereinstimmen. (Lediglich die Maße unterscheiden sich.) Das gleiche gilt für die letzten beiden Tupel. Da dies nicht gewünscht ist, wird auf allen Tupeln, die in die gleiche Zelle abgebildet wurden, ein Aggregat über die Maße berechnet. Das Ergebnis der Aggregation wird nun als Tupel der Zelle zugeordnet; die ursprünglichen Tupel werden verloren.

Nun wird die *aggregate*-Operation angewandt. Wählt man als Aggregationsfunktion $f_{Betrag} : \mathbb{R}^* \to \mathbb{R}$ die Summenfunktion, also

$$f_{Betrag}(R) = \sum_{r \in R} r$$

so werden die Daten zu folgenden zwei Tupeln zusammengefasst werden:

$$(Meyer, Joghurt, Frankfurt, Q2, 45.00)$$
$$(Meyer, Milch, Frankfurt, Q2, 18.00)$$

So erhält durch Aggregation jede Zelle im Würfel einen eindeutigen Wert.

Durch Aggregation ergibt sich eine Verkleinerung der Menge e, da Tupel entfernt werden. Es geht also Information verloren. Aus diesem Grund ist eine *aggregate*-Operation nicht reversibel. Dies muss berücksichtigt werden, wenn eine Drill-Down-Funktionalität realisiert werden soll. Auf diese Fragestellung wird in Abschnitt 4.5.3 genauer eingegangen.

4.5 Ausdrucksmächtigkeit

Um das beschriebene Datenmodell und die darauf definierten Operationen sinnvoll einsetzen zu können, muss man sich davon überzeugen, dass alle Funktionen, die von einem OLAP-System erwartet werden, auf das vorgestellte Modell abgebildet werden können.

Es wird daher im folgenden gezeigt, dass das Modell eine ausreichende Ausdrucksmächtigkeit besitzt. Dazu wird für jede OLAP-Operation eine äquivalente Folge von Operationen im Modell angegeben. Jedoch soll betont werden, dass die hier gewählten Darstellungen der Anschaulichkeit wegen gewählt wurden. Sie genügen daher nicht unbedingt den Anforderungen an einen formalen Beweis.

4.5.1 Pivoting

Da bei einer Pivoting-Operation lediglich Achsen vertauscht werden, lässt sie sich relativ einfach durch eine oder mehrere *swap*-Operationen nachstellen. Mit *swap* werden die betroffenen Attribute so oft geeignet vertauscht, bis die Zielkonstellation erreicht ist.

Dies lässt sich an einem Beispiel gut nachvollziehen. Die in Abbildung 4 auf Seite 6 dargestellte Pivot-Operation transformiert die Dimensionen (`Product`, `Time`, `Location`) nach (`Location`, `Product`, `Time`). Überträgt man dies auf das neue Datenmodell, so ergibt sich

$$
\begin{aligned}
dn &= (Product, Time, Location) \\
&\qquad\qquad\qquad\quad \downarrow swap(C, 1, 2) \\
dn &= (Time, Product, Location) \\
&\qquad\qquad\qquad\quad \downarrow swap(C', 1, 3) \\
dn &= (Location, Product, Time)
\end{aligned}
$$

Die Pivot-Operation konnte also in diesem Fall durch zwei *swap*-Operationen im Modell nachvollzogen werden. Es ist offensichtlich, dass für jede Pivot-Operation mindestens eine äquivalente Folge von *swap*-Operationen existiert (ohne Beweis).

4.5.2 Slicing und Dicing

Die bereits vorgestellten Operationen *Slicing* und *Dicing* erlauben es, Einschränkungen auf dem Wertebereich eines Hypercubes vorzunehmen. Dabei kann Dicing auf eine Reihe von Slicing-Operationen zurückgeführt werden. Es muss hier also nur noch untersucht werden, ob Slicing mit den im Modell verfügbaren Operationen nachgebildet werden kann.

Slicing basiert darauf, Zellen aus dem Würfel zu entfernen, die außerhalb eines gewünschten Bereichs liegen, der üblicherweise auf einer Dimension spezifiziert wird. Dies entspricht dem Entfernen von Tupeln aus der Menge e im Modell. Wird nun die Restriktion, die durch Slicing ausgeführt wird, als Prädikat ausgedrückt, kann die $restrict$-Funktion mit diesem Prädikat P verwendet werden, um die Restriktion auf der gewünschten Dimension, also dem Attribut i, zu erzielen.

Als Beispiel soll hier ein Slicing-Vorgang auf der Dimension `Time` dienen. Es sollen alle Verkäufe selektiert werden, die seit dem 1.01.2003 erfolgt sind. Man formuliert also das Prädikat P mit

$$P(u) \quad gdw \quad u > \text{"1.01.2003"}$$

Angenommen, es liegt ein Hypercube $C \in Cube$ der Gestalt

$$C = (3, (Product, Time, Location), dt, e, v)$$

vor. Dann ist für die Dimension `Time` das Attribut $i = 2$ zu wählen. Die Operation

$$restrict(C, i, P)$$

entspricht nun dem betrachteten Slicing-Vorgang.

So lange es möglich ist, die Slicing-Bedingung als Prädikat zu formulieren, kann also jede *Slicing*- oder *Dicing*-Operation mit den dargestellten Operationen vollzogen werden.

4.5.3 Drill-Down und Roll-Up

Aus Abschnitt 4.5.3 ist bereits bekannt, dass *Drill-Down* und *Roll-Up* der Navigation zwischen verschiedenen Hierarchieebenen dienen. Diese Funktionalität stellt im vorgestellten Modell ebenfalls zur Verfügung. In 4.4 wurde erläutert, wie Hierarchien im Modell modelliert werden können. Also müssen die gewünschten Hierarchien wie beschrieben formal hinterlegt werden.

Um nun einen Roll-Up auf einem Hypercube C durchzuführen, wird zunächst die $transform$-Funktion ausgeführt, um Werte und Metadaten des Hypercubes zu transformieren. Auf das Ergebnis wird dann die $aggregate$-Funktion angewandt. In der Schreibweise des Modells entspricht dies

$$C' = aggregate(f^*_{aggr}) \circ transform(f_{e_{1\to 2}}, f_{dn_{1\to 2}}, f_{dt_{1\to 2}}) \circ C$$

Damit befindet man sich auf Hierarchieebene 2. Wenn eine höhere Hierarchieebene erreicht werden soll, werden mehrere Roll-Ups in Folge durchgeführt (Transformationskette):

$$C^* \quad = \quad \underbrace{aggregate_{n-1} \circ transform_{n-1\to n}} \circ \ldots \circ \underbrace{aggregate_1 \circ transform_{1\to 2}} \circ C$$

Auf jeder Ebene kann eine andere Aggregatsfunktion $aggregate_i$ eingesetzt werden. Wird auf jeder Ebene die gleiche Funktion verwendet, kann in manchen Fällen (genauer: wenn die Aggregatsfunktion distributiv ist), darauf verzichtet werden, sie auf jeder Ebene zu berechnen. Dann genügt das Berechnen nachdem alle Transformationen durchgeführt sind. Ein Beispiel für eine solche Funktion ist die Summenfunktion Σ:

$$\widetilde{C} = \Sigma \circ transform_{n-1\to n} \circ \ldots \circ transform_{1\to 2} \circ C$$

Ein *Drill-Down* sorgt für größere Probleme. In der Regel ist ein Roll-Up verlustbehaftet. Es ist also nicht möglich, den Vorzustand zu rekonstruieren, wenn kein Zugriff auf die ursprünglichen Daten besteht. Hier hilft die Überlegung, dass ein Drill-Down erst durchgeführt werden kann, wenn sich der Benutzer auf einer Hierarchieebene $n > 1$ befindet.

Diese Ebene kann nur durch Roll-Ups erreicht worden sein. Fasst man diese Roll-Ups nun als Kette von Transformationen auf, kann ein Drill-Down durchgeführt werden, indem der korrespondierende Roll-Up aus der Transformationskette entfernt wird:

$$C^* = \underbrace{aggregate_{n-2} \circ transform_{n-2 \to n-1}} \circ \ldots \circ \underbrace{aggregate_1 \circ transform_{1 \to 2}} \circ C$$

Eine (eher seltene) Ausnahme besteht, wenn die in der Hierarchiedefinition verwendeten Funktionen bijektiv sind. Dann existieren nämlich Umkehrfunktionen, mit denen die ursprünglichen Werte berechnet werden können. In solchen Fällen besitzt ein Roll-Up ($transform$) eine Umkehrfunktion ($transform^{-1}$), die für Drill-Down verwendbar ist. Es gilt dann also

$$transform \circ transform^{-1} = id$$

Es können also sowohl Roll-Ups als auch Drill-Downs mittels der im Modell definierten Operationen vollzogen werden.

4.5.4 Push und Pull

Die OLAP-Funktionen Push und Pull existieren, wie in 3.4 erläutert, um Maße und Dimensionen ineinander umwandeln zu können.

Soll eine bestimmte Dimension in ein Maß umgewandelt werden, kommt auch im Modell die *push*-Operation zum Einsatz. Hier ist allerdings zu beachten, dass nicht beliebige Attribute konvertiert werden können, da $push$ keinen geeigneten Parameter (z.B. einen Attributindex) akzeptiert. Aus diesem Grund muss zuvor die Anordnung der Attribute durch (evtl. wiederholte) Anwendung der *swap*-Operation geeignet angepasst werden.

Für diese Anpassung kann Algorithmus 1 zum Einsatz kommen. Das so konvertierte Attribut i steht nun nicht mehr als Maß zur Verfügung. Ist dies gewünscht, kann der Algorithmus um eine *copy*-Operation erweitert werden, die das Attribut i dupliziert. Der entsprechende Funktionsaufruf lautet `copy(C, v+1)` und ist vor dem `return`-Statement einzufügen.

Auch die in OLAP bekannte Pull-Funktionalität, also das erzeugen einer neuen Dimension aus einem Maß, wird im Modell unterstützt. Hierfür steht die Funktion *pull* zur Verfügung. Genau wie bei $push$ besteht die Einschränkung, dass nicht beliebige Maße konvertiert werden können. Es muss wieder mittels $swap$ die Anordnung geeignet angepasst werden, damit das gewünschte Attribut an Position $v + 1$ liegt. Algorithmus löst diese Aufgabe.

Es können also die aus OLAP bekannten Push- und Pull-Operationen auf das Modell abgebildet werden.

Algorithm 1 *push* eines bestimmten Attributs i

Eingabe: Hypercube $C \in Cube$ mit $C = (k, dn, dt, e, v)$, Index $i \in \mathbb{N}_+$ des zu pushenden Attributs

Ausgabe: Hypercube $C' \in Cube$

```
// swap, damit zu pushende Dimension an Stelle
// v steht (also “rechts außen”)
C_swapped = swap(C, 1, v)
// push durchführen
C' = push(C_swapped)
// Ende
return(C')
```

Algorithm 2 *pull* eines bestimmten Attributs i

Eingabe: Hypercube $C \in Cube$ mit $C = (k, dn, dt, e, v)$, Index $i \in \mathbb{N}_+$ des zu pullenden Attributs

Ausgabe: Hypercube $C' \in Cube$

```
// swap, damit zu pullendes Maß an Stelle
// v steht (also “links außen”)
C_swapped = swap(C, v + 1, k)
// pull durchführen
C' = pull(C_swapped)
// Ende
return(C')
```

5 Zusammenfassung

Diese Ausarbeitung hat einen Überblick über die logischen Konzepte des Online Analytical Processing (OLAP) gegeben. Nach einer Einführung, die die Anwendungsfelder von OLAP skizziert, wurden Mehrdimensionale Datenobjekte vorgestellt. Eine solche Struktur ist der Hypercube.

Die in OLAP üblichen Operationen wurden in Kapitel 3 vorgestellt und anhand von Beispielen erläutert. Sie bieten dem Benutzer die Möglichkeit, die in einem Hypercube enthaltenen Daten zu manipulieren. Geeignete Anwendungsprogramme gestatten das interaktive Anwenden dieser Operationen. So kann ein Benutzer schnell einen Überblick über große Datenmengen gewinnen.

In Kapitel 4 wurde versucht, OLAP-Konzepte formal zu fassen. Es wurde hierzu ein Modell aufgestellt, das einen Hypercube beschreibt sowie einige Operationen darauf definiert. Dabei wurde besonderen Wert auf die äquivalente Behandlung von Maßen und Dimensionen, sowie die Modellierung von Hierarchien gelegt.

Im letzten Teil der vorliegenden Ausarbeitung wurde die Ausdrucksmächtigkeit des Modells untersucht. Es konnte festgestellt werden, dass es mit dem Modell möglich ist, alle üblichen OLAP-Operationen elegant und kompakt auszudrücken.

Literatur

[HKa01] Han, J., Kamber, M.: *Data Mining - Concepts and Techniques*
Morgan Kaufmann Publishers, 2001, ISBN 1-55860-489-83

[AGS98] Agrawal, R., Gupta, A., Sarawagi, S.: *Modeling Multidimensional Databases*
IBM Research Report, 1998, IBM Almaden Research Center

[SBH99] Sapia, C., Blaschka, M., Höfling, G.: *An Overview of Multidimensional Data Models for OLAP*
Report des Bayerischen Forschungszentrums für Wissensbasierte Systeme, http://citeseer.nj.nec.com/sapia99overview.htm

[GSS01] Sathe, G., Sarawagi, S.: *Intelligent Rollups in Multidimensional OLAP Data*
The VLDB Journal, 2001, S. 531-540