Evolution von Dimensionsdaten und Dimensionshierarchien

Johann Eder, Christian Koncilia

Universität Klagenfurt
Institut für Informatik-Systeme
{eder,koncilia}@isys.uni-klu.ac.at

Zusammenfassung Das wichtigste Einsatzgebiet von Data Warehouses ist die multidimensionale Analyse der Daten mittels OLAP-Tools. Die Dimensionen dienen dabei typischerweise der Beschreibung der Zeit, der Kennzahlen und der (betrieblichen) Strukturen wie zum Beispiel *Abteilungen*, *Kostenstellen* oder auch *Kostenarten*. Solche Strukturen sind jedoch dynamisch [7, 11, 1], was zu laufenden Anpassungen der multidimensionalen Sicht auf die Daten führt. Wir zeigen in dieser Arbeit, wie man mit solchen Strukturänderungen umgehen kann ohne Abstriche bzgl. der Konsistenz und Aussagekraft von Analysen machen zu müssen. Dieser Ansatz basiert im wesentlichen auf der Erweiterung des bekannten Data Warehouse Konzepts um temporale Aspekte, und Transformationsfunktionen die zur Hinterlegung von Beziehungen zwischen unterschiedlichen Strukturversionen dienen.

1 Einleitung & Motivation

Data Warehouses bzw. Data Marts sind meist materialisierte Views über strukturierte oder semi-strukturierte Datenbestände aus unterschiedlichen Datenquellen [14, 8]. Eines der wichtigsten Anwendungsgebiete von Data Warehouses besteht in der Analyse von unternehmensrelevanten Datenbeständen mittels On-Line Analytical Processing (OLAP) Technologien [2]. Dazu wird eine multidimensionalen Sicht auf die zu analysierenden Kennzahlen, wie z. B. Gewinn, Umsatz oder Absatz erzeugt.

Ein systemimmanentes Problem von Data Warehouses und OLAP-Systemen ist der Umgang mit Schemaänderungen. Gerade in analytischen Systemen ist aber eine korrekte Berücksichtigung von Schemaänderungen notwendig, da im Unterschied zu herkömmlichen Produktionssystemen, kein Snapshot zu einem bestimmten Zeitpunkt (siehe [12]) sondern Daten für mehrere Perioden über die Dimension „Zeit" betrachtet werden. Soll z. B. die ökonomische Entwicklung von europäischen Staaten über die letzten 20 Jahre analysiert werden, so ist es von essentieller Bedeutung, die Wiedervereinigung Deutschlands, die Trennung der Tschechoslowakei, etc. korrekt zu berücksichtigen.

Im folgenden verstehen wir unter einer Schema- oder Strukturänderung Modifikationen an Ausprägungen, die als Dimensionsdaten in der Analyse verwendet werden.

In derzeit eingesetzten OLAP-Systemen lassen sich bei Strukturbrüchen (wie z. B. der Aufnahme eines neuen Produkts in das Produktportfolio, oder dem Auflösen einer Vertriebsniederlassung) keinerlei Beziehungen zwischen den unterschiedlichen Strukturversionen angeben. Bei der Datenanalyse kommt es dann unweigerlich zu Problemen bzgl. der Aussagekraft der Kennzahlen, sobald Analysen erstellt werden, die Daten aus Perioden vor und nach Strukturänderungen verwenden.

Ein kurzes Beispiel: Eine Abteilung A wird mit März 2000 in zwei Abteilungen A_1 und A_2 aufgespalten. Will man nun eine Analyse über alle Monate des Jahres 2000 erstellen, so liegen für die Abteilung A nur Jänner- und Feberdaten vor – für die Abteilungen A_1 und A_2 hingegen nur Daten ab März. Es wird also vorausgesetzt, daß der Benutzer der erstellten Analyse entsprechendes Domain-Wissen mitbringt und sich darüber im klaren ist, daß die Abteilungen A, A_1 und A_2 irgendwie in Beziehung zueinander stehen. In sehr vielen Fällen lassen sich jedoch solche Beziehungen auch formalisieren und könnten dementsprechend im System hinterlegt werden, was gerade für Trendanalysen und Mehrperiodenvergleiche wichtig wäre. Im angegebenen Beispiel könnte man so hinterlegen, daß man den Umsatz der Abteilung A_1 für die Perioden vor dem März 2000 z. B. als Funktion $Umsatz(A_1, Periode) = Umsatz(A, Periode) * 0,3$ angeben kann, oder umgekehrt, daß die Mitarbeiteranzahl $M\#$ der Abteilung A für alle Perioden ab März 2000 der Funktion $M\#(A, Periode) = M\#(A_1, Periode) + M\#(A_2, Periode)$ entspricht. Durch die Verwendung solcher Funktionen kann sichergestellt werden, daß Analysen auch über einen Strukturbruch hinweg erstellt werden können.

Weitere wichtige Anwendungsgebiete wären der korrekte Umgang mit Einheitenänderungen (z. B. von DM auf EURO), oder Indexbereinigungen über die Zeit.

In diesem Artikel werden wir einen Ansatz präsentieren, wie auch bei dynamischen Strukturänderungen korrekte Auswertungen erstellt werden können. Dazu sind folgende Erweiterungen eines herkömmlichen Data Warehouse notwendig:

- **Temporale Erweiterung**: Daten müssen mit einem Zeitbezug historisiert gespeichert werden.
- **Schema Versionen**: aus der temporalen Haltung von Dimensionsdaten ergibt sich die Notwendigkeit, mit unterschiedlichen Strukturversionen, die wir als Schema Versionen bezeichnen, umzugehen.
- **Transformationsfunktionen**: Daten müssen von einer Schema Version in eine andere mittels Funktionen transformiert werden können.

Aus Platzgründen wird in dieser Arbeit auf eine detaillierte, formale Definition der Funktionen zur Datentransformation verzichtet. Wir verweisen hier auf [4] wo eine auf Matrizenkalkülen beruhende formale Definition gegeben wird und auch detaillierter auf die Beantwortung von Anfragen an ein solches System eingegangen wird.

2 Temporale Erweiterung eines DWH

Eine multidimensionale Sicht auf Daten besteht im wesentlichen aus einer Menge von Dimensionen die einen mehrdimensionalen Raum bzw. Datenwürfel aufspannen [13, 10]. Zellen in diesem Raum beinhalten die zu analysierenden Werte und werden über die Dimensionen bzw. deren Ausprägungen referenziert. Üblicherweise wird ein solcher Datenwürfel durch eine Zeit-, eine Kennzahlen- und mehrere Strukturdimensionen aufgespannt.

Die Zeitdimension definiert dabei die Granularität der Gültigkeitszeit (der Zeitraum in dem ein Objekt in der Realwelt den angegebenen Zustand, bzw. die angegebene Wertebelegung aufweist [9]) – also das „Chronon[1]", bzw. den kleinsten nicht weiter zerlegbaren Zeitintervall (Monate, Tage, Stunden, ...). Im Allgemeinen hat die Zeitdimension eine hierarchische Struktur wie z. B. $Jahr \rightarrow Quartal \rightarrow Monat$.

Die Kennzahlendimension beschreibt die zu analysierenden Kennzahlen inkl. deren hierarchischem Zusammenhang. Sie beschreibt also „*Was*" analysiert wird. Typische Beispiele wären Absatz, Umsatz oder Kosten. Kennzahlen werden oft auch als Fakten bezeichnet.

Die Strukturdimensionen legen fest, „*Wie*" die definierten Kennzahlen analysiert werden können. Beispiele für solche Strukturdimensionen wären *Produkte* : *Produktgruppen* $\rightarrow$ *Produkte* oder *Gebiet* : *Staat* $\rightarrow$ *Stadt* $\rightarrow$ $Filiale$, etc.

Eine sorgfältige Anforderungsanalyse kann Schemaänderungen innerhalb der Zeit- und Kennzahlendimensionen weitgehend minimieren (wenn auch nicht ausschließen). Änderungen innerhalb der Strukturdimensionen werden aber auch nach sorgfältigster Analyse laufend auftreten, da Unternehmensstrukturen sehr dynamisch sind.

Eine Dimension selbst besteht aus einer Menge von Dimensiondaten und deren hierarchischen Beziehungen. Dimensionsdaten entsprechen dabei der Extension der entsprechenden Dimensionen. So gehören zur Dimension *Zeit* z. B. die Dimensiondaten *1999*, *April* und *14. April* die in der hierarchischen Beziehung 1999 $\rightarrow$ *April* $\rightarrow$ 14.*April* stehen. Die hierarchischen Beziehungen definieren dabei den Konsolidierungspfad.

Die bisher intuitive Beschreibung eines multidimensionalen Systems soll nun durch eine formale Beschreibung ergänzt werden. Einige Arbeiten wie [1] oder [6] geben eine solche formale Definition. Für unsere Arbeit benötigen wir jedoch eine Definition die einfach um temporale Aspekte erweitert werden kann.

Ein multidimensionales System besteht aus:

i.) Einer Anzahl von Dimensionen $N + 1$.
ii.) Einer Menge von Dimensionen $\mathcal{D} = \{D_1, ..., D_N, F\}$ wobei F eine Kennzahlendimension ist, und D_i sonstige Dimensionen sind.
iii.) Einer Anzahl von M Dimensiondaten.
iv.) Einer Menge von Dimensiondaten $\mathcal{DD} = \mathcal{DD}_{D_1} \cup ... \cup \mathcal{DD}_{D_N} \cup \mathcal{DD}_F = \{DM_1, ..., DM_M\}$ wobei $\mathcal{DD}_F$ die Menge aller Kennzahlen, $\mathcal{DD}_{D_i}$ die Menge aller Dimensionsdaten der Dimension D_i, und $DM_i =< DM_{id}, Key, D_i, \mathcal{UDA} >$ ist. DM_{id} ist eine für jedes Dimensionsdatum eindeutige Kennung welche nicht verändert werden kann (ähnlich einer O_{id} in Objekt-Orientierten Datenbanksystemen). Key ist ein benutzerdefinierter Schlüssel (z. B. die Produktnummer) der innerhalb von D_i eindeutig ist. D_i ist die Dimension der das entsprechende Dimensionsdatum zugeordnet ist. $\mathcal{UDA}$ ist eine Menge von benutzerdefinierten Attributen (User-Defined Attributes) wie z. B. der Bezeichner oder die Farbe eines Produktes.

[1] [9] definiert ein Chronon als „a non-decomposable time interval of some fixed, minimal duration"

v.) Einer Menge von hierarchischen Zuordnungen $\mathcal{H} = \{H_1, ..., H_O\}$ wobei $H_i =< DM_{id}^C,\ DM_{id}^P,\ Level >$. DM_{id}^C ist die Kennung eines Dimensionsdatums und DM_{id}^P die Kennung des dazugehörigen „Vaters" oder $\emptyset$ wenn DM_{id}^C ein Top-Level Dimensionsdatum ist (ein Dimensionsdatum welches innerhalb der Dimension auf höchster Ebene angesiedelt ist). *Level* ist ein Wert aus $\{0...L\}$ wobei L die Anzahl der Ebenen ist, und *Level* die Ebene ist, welcher DM_{id}^C zugeordnet ist. Alle „Blätter" (Dimensiondaten ohne Nachfolger) befinden sich auf Level 0. Zyklen sind innerhalb von $\mathcal{H}$ nicht zulässig.

vi.) Eine Funktion $cval : (DM_{D_1}, ..., DM_{D_N}, DM_F) \rightarrow value$ welche jedem Vektor $(DM_{D_1}, ..., DM_{D_N}, DM_F)$ einen eindeutigen Wert zuordnet wobei $(DM_{D_1}, ..., DM_{D_N}, DM_F) \in \mathcal{DD}_{D_1} \times ... \times \mathcal{DD}_{D_N} \times \mathcal{DD}_F$ gilt.

Daraus folgt, daß ein Datenwürfel C definiert ist durch die Funktion *cval*. Der Definitionsbereich dieses Datenwürfels $dom(C)$ ist die Menge aller Vektoren. Der Bildbereich dieses Datenwürfels $ran(C)$ ist die Menge aller Zellwerte.

Das so definierte multidimensionale System wird nun um temporale Aspekte erweitert. Unser Ansatz berücksichtigt dabei die Gültigkeitszeit. Die Transaktionszeit (der Zeitpunkt zu dem eine Werteänderung dem Datenbanksystem bekannt gegeben wurde [9]) bleibt vorerst unberücksichtigt.

Wir fassen dazu eine Dimension als gerichteten Graphen auf. Dabei entspricht jedes Dimensionsdatum einem Knoten und jede hierarchische Beziehung zwischen Dimensionsdaten einer Kante. Ein multidimensionales System, welches die Gültigkeitszeit unterstützen soll, muß alle Knoten und Kanten eines solchen Graphen mit Zeitstempeln der Form $[T_s, T_e]$ versehen. T_s steht dabei für den Startzeitpunkt und T_e für den Endzeitpunkt. Steht der Endzeitpunkt noch nicht fest, so wird dieser durch ∞ repräsentiert. Weiters gilt, daß $T_e \geq T_s$. $[T_s, T_e]$ definiert also den Intervall von wann bis wann ein bestimmter Zustand gültig war. In der Arbeit von P. Chamoni und S. Stock [3] werden lediglich Kanten mit Zeitstempeln versehen. Dies reicht jedoch nicht aus, wenn auch Attributänderungen an den Dimensiondaten selbst historisiert werden sollen.

Unsere weiter oben gegebene formale Definition eines multidimensionalen Systems wird nun also wie folgt erweitert:

i.) Ein Dimensionsdatum ist ein Tupel $DM_i =< DM_{id},\ Key,\ D_i,\ \mathcal{UDA},\ [T_s, T_e] >$. DM_{id}, D_i und $\mathcal{UDA}$ sind wie weiter oben beschrieben definiert. *Key* ist ein benutzerdefinierter Schlüssel der innerhalb von D_i für jeden Zeitpunkt $T_s \leq T \leq T_e$ eindeutig ist. $[T_s, T_e]$ repräsentiert die Gültigkeitszeit des Dimensionsdatums.

ii.) Eine hierarchische Beziehung ist ein Tupel $H_i =< DM_{id}^C,\ DM_{id}^P,\ Level,\ [T_s, T_e] >$. DM_{id}^C, DM_{id}^P und *Level* sind wie oben beschrieben definiert. $[T_s, T_e]$ definiert dabei die Gültigkeitszeit der Beziehung zwischen DM_{id}^C und DM_{id}^P.

3 Schema Versionen

Mittels der in [9] beschriebenen temporalen Projektion und temporalen Selektion können nun sogenannte *Schema Versionen* (SV) ermittelt werden. Intuitiv kann eine Schema Version definiert werden als eine multidimensionale Struktur die für einen bestimmten Zeitraum gültig ist, und in der alle für diesen Zeitraum definierten Dimensiondaten und hierarchischen Beziehungen gültig sind. Das bedeutet also, daß es innerhalb einer Schema Version keine unterschiedlichen Versionen eines Dimensionsdatums oder einer hierarchischen Beziehung geben kann, bzw. umgekehrt, daß für jede Änderung eines Dimensionsdatums oder einer hierarchischen Beziehung automatisch eine neue Schema Version generiert wird, so für diesen Zeitintervall noch keine entsprechende Schema Version existiert.

Formal definieren wir eine Schema Version wie folgt: Eine Schema Version ist ein 4-Tupel $< SV_{id},\ T,\ \{\mathcal{DD}_{D_1,SV_{id}},\ ...,\ \mathcal{DD}_{D_N,SV_{id}},\ \mathcal{DD}_{F,SV_{id}}\},\ \mathcal{H}_{SV_{id}} >$ wobei SV_{id} eine eindeutige Kennung der Schema Version, und T die Gültigkeitszeit der Schema Version in der Form $[T_s, T_e]$ ist. $\mathcal{DD}_{D_i,SV_{id}}$ ($\mathcal{DD}_{D_i,SV_{id}} \subseteq \mathcal{DD}_{D_i}$) ist die Menge aller Dimensiondaten die der Dimension D_i zugeordnet sind und die zu jedem Zeitpunkt P mit $T_s \leq P \leq T_e$ gültig sind. $\mathcal{DD}_{F,SV_{id}}$ ($\mathcal{DD}_{F,SV_{id}} \subseteq \mathcal{DD}_F$) ist die Menge aller Kennzahlen die zu jedem Zeitpunkt P mit $T_s \leq P \leq T_e$ gültig sind. $\mathcal{H}_{SV_{id}}$ ($\mathcal{H}_{SV_{id}} \subseteq \mathcal{H}$) ist die Menge aller hierarchischen Beziehungen die zu jedem Zeitpunkt P mit $T_s \leq P \leq T_e$ gültig sind.

Konzeptuell betrachtet existiert für jede Schema Version SV ein korrespondierender Datenwürfel. Eine Schema Version beinhaltet also nicht alle Änderungen zu einem bestimmten Zeitpunkt, sondern den für einen bestimmten Zeitraum gültigen Datenwürfel. Wir möchten jedoch an dieser Stelle explizit darauf hinweisen, daß im physischen Datenmodell Faktoren wie Performance, Normalisierungsgrad, Redundanz, etc. berücksichtigt werden.

Wie bereits beschrieben ergeben sich Schema Versionen aus Änderungen in Dimensionsdaten oder Dimensionshierarchien. Die dazu notwendigen Funktionen werden im folgenden Abschnitt erläutert.

3.1 Strukturänderungen

Die Dimension *Zeit* definiert ein Chronon C. Da dieses Chronon festlegt, in welcher Granularität bzgl. der Zeit Daten im Data Warehouse vorliegen und analysiert werden können, würde das Berücksichtigen von Strukturänderungen innerhalb dieses Chronons kein mehr an Informationsgehalt mit sich bringen. Strukturänderungen brauchen daher nur mit der durch C vorgegeben Granularität erfasst werden.

Desweiteren definieren wir ein Data Warehouse (DWH) als eine nicht-leere, endliche Menge von Schema Versionen $DWH = \{SV_1, ..., SV_n\}$. Wie bereits beschrieben ist jede Schema Version ein 4-Tupel der Form $< SV_{id}, T, \mathcal{DD}_{SV_{id}}, \mathcal{H}_{SV_{id}} >$. Das so definierte Data Warehouse muß eine aufeinanderfolgende Sequenz von Tupeln $< SV_{id}, T_i, \mathcal{DD}_{SV_{id}}, \mathcal{H}_{SV_{id}} >$ sein, sodaß gilt $T_i = [T_{i,s}, T_{i,e}]$ und $T_{i,s} = T_{(i-1),e} + C$.

Dimensionsdaten können mit Hilfe der drei Funktionen `INSERT`, `DELETE` und `UPDATE` geändert werden. Diese sind wie folgt definiert:

Insert: Die Funktion `INSERT` fügt ein neues Dimensionsdatum ein und ist definiert als `INSERT(`DM`,` T_s`)`. DM ist das neue Dimensionsdatum und ist definiert als Tupel $< Key, D_i, \mathcal{UDA}, DM_{id}^P >$. Key, D_i, $\mathcal{UDA}$ und DM_{id}^P sind dabei wie in Abschnitt 2 beschrieben definiert. T_s gibt den Zeitpunkt an, ab dem das Dimensionsdatum gültig ist. Es definiert also den Zeitintervall $[T_s, \infty]$. Eine eindeutige Kennung DM_{id} wird dem Dimensionsdatum zugewiesen.

Update: Die Funktion `UPDATE` ändert ein existierendes Dimensionsdatum und ist definiert als `UPDATE(`Key`,` D_i`,` DM'`,` T_s`)`. Key und D_i geben an, welches Dimensionsdatum geändert werden soll. DM' ist ein Tupel $< Key, \mathcal{UDA}, DM_{id}^P >$ welches die neuen Werte für das Dimensionsdatum enthält. Eine `UPDATE` Funktion resultiert in zwei Aktionen: der Endzeitpunkt der Gültigkeitszeit des zu aktualisierenden Dimensiondaten wird auf $T_s - C$ gesetzt. Zusätzlich wird ein neues Dimensionsdatum eingefügt, dessen Gültigkeitszeit auf $[T_s, \infty]$ gesetzt wird.

Delete: Die Funktion `DELETE` „löscht" ein existierendes Dimensionsdatum und ist definiert als `DELETE(`DM`,` T_e`)`. DM referenziert eindeutig das Dimensionsdatum welches „gelöscht" werden soll und ist definiert als ein Tupel $< Key, D_i >$. T_e definiert den Zeitpunkt bis zu welchem dieses Dimensionsdatum gültig war. Eine `DELETE` Funktion setzt den Zeitstempel des Dimensionsdatums von $[T_s, \infty]$ auf $[T_s, T_e]$.

Existiert keine Schema Version mit einem entsprechenden Zeitstempel der aus den Parametern der Funktion `INSERT`, `UPDATE` oder `DELETE` folgt, so wird eine neue Schema Version generiert.

Durch Verwendung dieser drei Basisfunktionen können komplexere Modifikationen an Dimensionsdaten vorgenommen werden. Wir unterscheiden dabei zwischen den folgenden Änderungen, die in [4] detailliert beschrieben wurden:

1.) SPLIT: Ein Dimensionsdatum wird in n Dimensiondaten aufgespalten.
2.) MERGE: n Dimensiondaten werden zu einem Dimensionsdatum zusammengefaßt.
3.) CHANGE: Ein oder mehrere Attribute eines Dimensionsdatums ändern sich.
4.) MOVE: Die hierarchische Zuordnung eines Dimensionsdatums ändert sich.
5.) NEW-MEMBER: Ein neues Dimensionsdatum wird eingefügt.
6.) DELETE-MEMBER: Ein existierendes Dimensionsdatum wird gelöscht.

4 Beziehungen zwischen Schema Versionen

Bisher wurde gezeigt, wie ein multidimensionales System um temporale Aspekte erweitert werden kann, und welche Funktionen notwendig sind um Änderungen in einem solchen System vornehmen zu können.

Wir werden nun zeigen, wie Beziehungen zwischen Dimensiondaten unterschiedlicher Schema Versionen hinterlegt werden können. Solche Beziehungen werden z. B. hinterlegt um dem System mitzuteilen, daß die Abteilung *„Recht & Finanzen"*, welche in der Schema Version SV_i gültig ist, nach einer Aufspaltung in die Abteilungen *„Recht"* und *„Finanzen"*, in irgendeiner Beziehung mit diesen Abteilungen steht, welche in der Schema Version SV_{i+1} gültig sind.

Durch die Verwendung von solchen Beziehungen kann ein Benutzer Daten von einer Schema Version, bzw. Struktur in eine andere Schema Version umrechnen lassen. Er kann also z. B. die Umsätze für Jänner 1999 nach der aktuell gültigen Vertriebsstruktur analysieren.

Um solche Beziehung zu hinterlegen, führen wir sogenannte *„Transformationsfunktionen"* ein. Eine Transformationsfunktion ist eine generische Funktion, die jeweils ein Dimensionsdatum aus einer Schema Version,

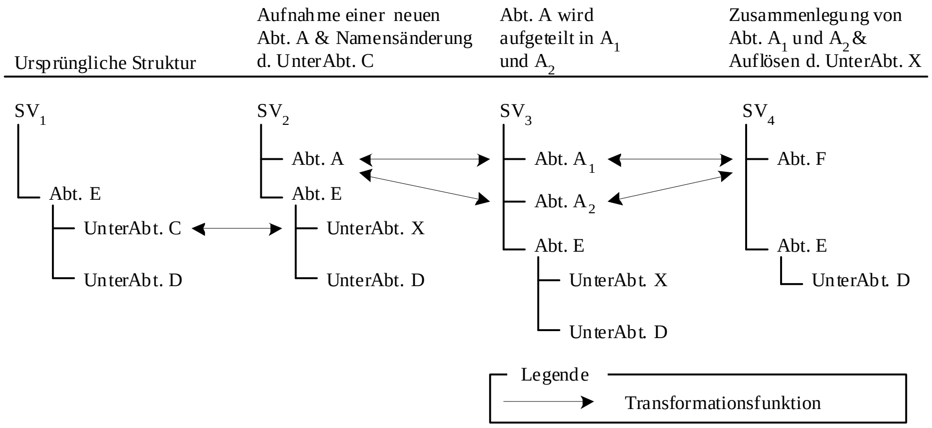

Abbildung 1. Ein Beispiel für Strukturänderungen und Transformationsfunktionen

unter Verwendung eines Gewichtungsfaktors, mit einem Dimensionsdatum einer anderen Schema Version verbindet. Transformationsfunktionen sind weder injektiv noch surjektiv. D. h., ein Dimensionsdatum kann über mehrere Funktionen mit unterschiedlichen Dimensionsdaten einer andere Schema Version in Beziehung stehen (notwendig bei einer Split-Operation), und umgekehrt.

Transformationsfunktionen können dabei beliebige Schema Versionen miteinander in Verbindung setzen. Dies ermöglicht Analysen entlang beider Zeitrichtungen, also z. B. Analysen der Jännerumsätze nach der im Februar gültigen Struktur, oder Analysen der Februarumsätze nach der im Jänner gültigen Struktur.

Um solche Anfragen an das System zu stellen, definiert der Benutzer zuerst implizit oder explizit eine Struktur (Schema Version) nach der die Daten analysiert werden sollen. Das Ergebnis der Anfrage kann jedoch Daten aus beliebig vielen Schema Versionen beinhalten, die dann jeweils in die vom Benutzer angegebene Schema Version transformiert werden.

Zu beachten ist weiters, daß es unterschiedliche Beziehungen für unterschiedlichen Kennzahlen geben kann. So kann hinterlegt werden, daß die Produkte „P_1" und „P_2" bzgl. des Umsatzes in einer anderen Beziehung stehen, als bzgl. des Absatzes.

Des weiteren machen wir uns eine Beschränkung von multidimensionalen Datenbanksystemen zunutze, um unser System möglichst einfach zu halten, ohne den Anspruch auf Allgemeingültigkeit aufzugeben. In multidimensionalen Datenbanksystemen ergeben sich die Werte von „inneren Knoten" (also von Dimensiondaten denen zumindest ein Nachfolger zugeordnet ist) stets durch die Anwendung einer Funktion auf die Werte der entsprechenden „Blätter". So ergibt sich zum Beispiel der Umsatz einer Produktgruppe aus der Summe der Umsätze aller Produkte die dieser Produktgruppe zugeordnet sind. Für unser Modell bedeutet dies, das Transformationsfunktionen stets Blätter miteinander in Beziehung setzen. Oder, in anderen Worten: Wenn für eine Anfrage das Überleiten von Daten aus einer Schema Version in eine andere Schema Version notwendig ist, so passiert dies stets auf Level-0 Ebene. Nachdem die Daten auf dieser Ebene transformiert wurden, können die Werte der inneren Knoten wie in multidimensionalen System üblich konsolidiert werden.

4.1 Definition von Transformationsfunktionen

Wie bereits beschrieben werden Transformationsfunktionen verwendet, um Daten (Zellwerte) für bestimmte Kennzahlen und bestimmte Dimensiondaten unter Verwendung von Gewichtungsfaktoren von einer Schema Version in eine andere zu transformieren. Gewichtungsfaktoren entsprechen dabei im Wesentlichen einer prozentuellen Angabe.

Wir definieren dazu eine Operation $MapF$ wie folgt: $MapF(SV_j, SV_k, DM_{id}, DM'_{id}, \{M^1_{id}, ..., M^n_{id}\}, w)$.

i.) SV_j und SV_k sind Kennungen unterschiedlicher Schema Versionen.
ii.) DM_{id} und DM'_{id} sind eindeutige Kennungen für Dimensiondaten für die $DM_{id} \in SV_j$ und $DM'_{id} \in SV_k$ gilt. DM_{id} und DM'_{id} müssen Dimensiondaten der selben Dimension sein.
iii.) $\{M^1_{id}, ..., M^n_{id}\}$ ist eine nicht leere, endliche Menge von Kennzahlen-Kennungen und $\exists f : f \in F \wedge f_{id} = M^i_{id}$.
iv.) w ist der Gewichtungsfaktor.

Für Dimensiondaten die sich von einer Schema Version auf eine andere nicht ändern, wird implizit eine Transformationsfunktion mit dem Gewichtungsfaktor $w = 1$ hinterlegt.

Transformationsfunktionen können zeitlich aufeinanderfolgende (z. B. $SV_1 \leftrightarrow SV_2$) aber auch zeitlich nicht aufeinanderfolgende (z. B. $SV_1 \leftrightarrow SV_3$) Schema Versionen miteinander in Beziehung setzen. Zwei Schema Version SV_i und SV_k werden dabei als aufeinanderfolgend bezeichnet wenn $T_{s,i} = T_{e,k} + C$ oder $T_{s,k} = T_{e,i} + C$ gilt.

Das Beispiel in Abb. 1 zeigt mehrere strukturelle Änderungen in einer Dimension „*Abteilungen*". So wurde z. B. das Dimensionsdatum „*UnterAbt.C*" der Schema Version SV_1 zu „*UnterAbt.X*" in SV_2 umbenannt, „*Abt.A*" wurde von der Schema Version SV_2 zu SV_3 aufgesplittet in „$Abt.A_1$" und „$Abt.A_2$", usw.

Bezüglich der Kennzahl „Umsatz" U könnte dieses Beispiel unter Verwendung der folgenden Transformationsfunktionen dargestellt werden:

1.) $MapF(SV_1, SV_2, UnterAbtC, UnterAbtX, \{U\}, 1)$
2.) $MapF(SV_2, SV_1, UnterAbtX, UnterAbtC, \{U\}, 1)$
3.) $MapF(SV_2, SV_3, AbtA, AbtA_1, \{U\}, 0.3)$
4.) $MapF(SV_2, SV_3, AbtA, AbtA_2, \{U\}, 0.7)$
5.) $MapF(SV_3, SV_2, AbtA_1, AbtA, \{U\}, 1)$
6.) $MapF(SV_3, SV_2, AbtA_2, AbtA, \{U\}, 1)$
7.) und so weiter...

Diese Menge von Funktionen definiert zum Beispiel, daß bzgl. der Kennzahl „Umsatz" die Abteilung A_1 in der Schema Version SV_3 gleich 30% der Abteilung A in der Schema Version SV_2 ist (siehe Funktion Nr. 3 in obiger Liste). Umgekehrt wird hinterlegt, daß die Abteilung A in der Schema Version SV_2 gleich der Summe von A_1 und A_2 in der Schema Version SV_3 ist (siehe Funktionen Nr. 5 und 6).

5 Fazit

Wir haben in dieser Arbeit einen Ansatz vorgestellt um Data Warehouses, bzw. multidimensionale Datenstrukturen um temporale Aspekte zu erweitern. Dieser Ansatz erlaubt einerseits die Historisierung von Änderungen in Dimensionsdaten und Dimensionshierarchien, und andererseits die korrekte Transformation von Daten zwischen unterschiedlichen Strukturversionen.

Dies ermöglicht ein „sauberes" Design von Data Warehouses, welches ohne „workarounds" auskommt, um mit Strukturänderungen umzugehen. Wir gehen weiters davon aus, daß dieses System die korrekte Interpretation von Abfrageergebnissen erleichtert da der Benutzer von der Pflicht entbunden wird, detailliertes Wissen über die historische Entwicklung der Strukturen zu besitzen.

Derzeit wird dieser Ansatz unter Verwendung einer relationalen und einer objekt-relationalen Datenbank prototypisch implementiert. Außerdem soll das System um dynamische Gewichtungsfaktoren und benutzerdefinierte Transformationsfunktionen erweitert werden.

Literatur

[1] BLASCHKA, M. ; SAPIA, C. ; HÖFLING, G.: On Schema Evolution in Multidimensional Databases. In: *Proc. of the DaWak99 Conference*. Florence, Italy, 1999

[2] CABIBBO, L. ; TORLONE, R.: A Logical Approach to Multidimensional Databases. In: *Proc. EDBT*, 1998

[3] CHAMONI, P. ; STOCK, S.: Modellierung temporaler multidimensionaler Daten in Analytischen Informationssystemen. In: KRUSE (Hrsg.) ; RUDOLF (Hrsg.) ; SAAKE (Hrsg.) ; GUNTER (Hrsg.): *Arbeitsbericht 14*. Magdeburg : Otto-von-Guericke-Universität Magdeburg, 1998, S. 93–106

[4] EDER, J. ; KONCILIA, C.: Evolution of Dimension Data in Temporal Data Warehouses. In: *Technical Report* (2000). – URL: http://www.ifi.uni-klu.ac.at

[5] ETZION, O. (Hrsg.) ; JAJODIA, S. (Hrsg.) ; SRIPADA, S. (Hrsg.): *Temporal Databases: Research and Practise*. Springer-Verlag, 1998 (Lecture Notes in Computer Science 1399)

[6] GYSSENS, M. ; LAKSHMANAN, L.: A Foundation for Multi-Dimensional Databases. In: VIJAYARAMAN, T. M. (Hrsg.) ; BUCHMANN, Alejandro P. (Hrsg.) ; MOHAN, C. (Hrsg.) ; SARDA, Nandlal L. (Hrsg.): *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, 1996, Mumbai (Bombay), India*, Morgan Kaufmann, 1996

[7] HURTADO, C. ; MENDELZON, A. ; VAISMAN, A.: Updating OLAP Dimensions. In: *Proceedings of the ACM second international workshop on Data warehousing and OLAP*. Kansas City, MO USA, 1999

[8] INMON, W.: *Building the Data Warehouse*. 2. New York : John Wiley and Sons, 1996
[9] JENSEN, C. S. (Hrsg.) ; DYRESON, C. E. (Hrsg.): *A consensus Glossary of Temporal Database Concepts - Feb. 1998 Version*. Springer-Verlag, 1998, S. 367–405. – in [EJS98]
[10] LI, C. ; WANG, X.: A Data Model for Supporting On-Line Analytical Processing. In: *ACM* CIKM 96 (1996)
[11] SAP AMERICA, INC. AND SAP AG: Data Modelling with BW - ASAP for BW Accelerator. (1998). – White Paper: URL: http://www.sap.com
[12] SNODGRASS, R.: A Taxonomy of Time in Databases. In: *ACM* (1985)
[13] VASSILIADIS, P. ; SELLIS, T.: A Survey of Logical Models for OLAP Databases. In: *SIGMOD Record 28*, 1999
[14] WU, M. ; BUCHMANN, A.: Research Issues in Data Warehousing. In: *BTW'97* (1997)