Temporale Auswertungsformen in OLAP

Dissertation

zur Erlangung der Würde eines Doktors der Staatswissenschaften

vorgelegt der

Wirtschaftswissenschaftlichen Fakultät der Universität Basel

von Andreas Thurnheer

von Berneck, SG

Basel, 2003

Genehmigt von der Wirtschaftswissenschaftlichen Fakultät der Universität Basel auf Antrag von Prof. Dr. Markus Lusti und Prof. Dr. Friedrich Rosenkranz.

Basel, den 10. April 2003

Der Dekan

Prof. Dr. Werner Müller

Vorwort

Die vorliegende Dissertation entstand in den Jahren 1998 bis 2002 und ist das Ergebnis meiner Forschungstätigkeit am Wirtschaftswissenschaftlichen Zentrum der Universität Basel. Es war mir ein Anliegen, neue Ideen nicht nur zu formulieren und in das Forschungsgebiet analytischer Datenbanken zu integrieren, sondern auch ihre Realisierbarkeit zu testen und zu beurteilen. Dabei ist ein Prototyp entstanden, den ich als Ergebnis interessierten Kreisen nicht vorenthalten möchte. Die folgende WWW-Seite enthält eine Projektzusammenfassung und verweist auf die entwickelten Anwendungen sowie auf deren Quellcode: `http://www.wwz.unibas.ch/wi/projects/afm/index.htm`

Bei meiner Arbeit bin ich von zahlreichen Personen auf vielfältige Weise unterstützt worden. Ihnen möchte ich an dieser Stelle herzlich danken. Ganz besonderer Dank richtet sich an Prof. Dr. Markus Lusti für die Anregung und Betreuung dieser Dissertation. Seine Unterstützung und Vertrauen haben wesentlich zum Gelingen meiner Arbeit beigetragen. Ein spezieller Dank gebührt auch Prof. Dr. Friedrich Rosenkranz für die Übernahme des Koreferats sowie dem Förderverein des Wirtschaftswissenschaftlichen Zentrums für seine Unterstützung meiner Forschungsarbeit.

Dank möchte ich auch den Mitarbeiterinnen und Mitarbeitern der Abteilung Wirtschaftsinformatik aussprechen. Als gute Kollegen und Kritiker standen sie mir hilfreich zur Seite. Ein besonderes Dankeschön richtet sich an Andreas Born, Thomas Zuber, Ute Trapp, Alessio Ishizaka, Patrick Wirz, Martin Fischer, Yudon Tsering, Pius Ten Hacken, Daniela Zappatore und Dorota Smyk.

Herzlichen Dank auch an meine Frau Martina und an meine Eltern, die meinen Werdegang stets unterstützten und förderten.

Andreas Thurnheer, im Mai 2003

Zusammenfassung

Online Analytical Processing (OLAP) ist eine Analysemethode, mit der Endbenutzer auch ohne Programmierkenntnisse in der Lage sind, komplexe multidimensionale Auswertungen ad-hoc zu definieren. Die meisten analytischen Datenmodelle basieren auf dem Sternschema, das zwischen Dimensionen und Fakten unterscheidet. Während die Fakten die zentralen betrieblichen Erfolgsgrössen abbilden, enthalten die Dimensionen die Kriterien, welche die Auswahl, Zusammenfassung und Navigation der Fakten ermöglichen. Die Fakten sind in den meisten dimensionalen Schemata zeitbezogen abgebildet, weil Analysen häufig nicht nur auf aktuellen, sondern auch auf vergangenen Daten basieren. Obwohl Dimensionsdaten im Zeitablauf auch veränderbar sind, sehen viele OLAP-Systeme ihre zeitbezogene Speicherung nicht vor. Dies scheint zunächst auch nicht notwendig zu sein, weil die Anwender nicht die Dimensionsdaten, sondern die Fakten ins Zentrum ihrer Analyse stellen.

Änderungen in den Dimensionsdaten führen jedoch dazu, dass ein Faktwert im Lauf der Zeit von verschiedenen Versionen eines Dimensionsobjekts abhängig abgebildet wird und identische Auswertungen, die zu verschiedenen Zeitpunkten gemacht werden, unter Umständen unterschiedliche Resultate ergeben. Abhängig von der verwendeten Aktualisierungsmethode ordnet das OLAP-System die Fakten den aktuellen, den ursprünglichen oder den zu ihrer Erfassungszeit gültigen Dimensionsdaten zu. Den Anwendern steht bei der Analyse jedoch nur eine der erwähnten temporalen Auswertungsformen zur Verfügung, weil das OLAP-System ein Datenobjekt nicht gleichzeitig mit verschiedenen Methoden aktualisieren kann ohne mehrere Datenmodelle parallel pflegen zu müssen.

Die vorliegende Arbeit untersucht deshalb Methoden, welche die Abbildung verschiedener temporaler Auswertungsformen in OLAP unterstützen. Nach dem Vergleich bekannter Modellierungskonzepte wird das aggregierte Faktenmodell eingeführt. Dieses erweitert ein konventionelles OLAP-System um eine neue Klasse temporaler Aggregate, mit deren Hilfe sich eine alternative temporale Auswertungsform abbilden lässt. Die Definition, die Verwaltung und der Zugriff auf diese temporalen Aggregate wird grösstenteils von den Metadaten gesteuert, so dass weder Systementwickler noch Endbenutzer bei der Administration oder Anwendung des erweiterten Modells mit der zusätzlichen Komplexität konfrontiert wird.

Weitere Informationen befinden sich im Internet unter folgender Adresse:

www.wwz.unibas.ch/wi/projects/afm

Summary

Online Analytical Processing (OLAP) enables end-users to define complex multidimensional queries without the use of any programming skills. Most analytical data models are based on the star schema, which differentiates between dimensions and facts. While facts represent the main business measures, dimensions contain the criteria to select, aggregate and navigate them. Since decisions not only depend on current but also on historical data, facts are stored in a time-dependent way. Although dimension data may also change over time, many OLAP systems do not facilitate their time-dependent storage. As end-users do not focus on dimension data in their analysis but on facts, it seems that temporal aspects of dimension data can be disregarded at first view.

However, changes in dimension data may result in a situation, where a fact value is assigned to different versions of a dimension object over time. In this case, identical queries executed in different points of time possibly lead to unequal results. Depending on the applied modeling technique and update strategy, OLAP systems assign facts to the current, the original or to this version of the dimension data which was valid at the time the facts were recorded. Since OLAP systems can not update a data object with different methods at the same time without having to maintain several data models simultaneously, it can only provide one temporal perspective to dimension data.

This dissertation determines methods to enable the representation of different temporal perspectives in an OLAP system. After the comparison of already known modeling approaches, we introduce the Aggregated Fact Model. This method extends a conventional OLAP system with a new class of temporal aggregates, providing an alternative temporal perspective to dimension data. As the definition, administration and analysis process is mostly driven by meta data, neither administrators nor end-users will be faced with additional complexity.

Further Information can be found on the internet at the following address:

www.wwz.unibas.ch/wi/projects/afm

Inhalt

Abbildungen

Abkürzungen

ADO	ActiveX Data Objects
ADOX	ActiveX Data Objects Extensions
COM	Common Object Model
ETL	Extraktion, Transformation, Laden
FASMI	Fast Analysis Shared Multidimensional Information
MDX	Multidimensional Expression Language
MS	Microsoft
OCX	OLE Custom Control
OLAP	On-Line Analytical Processing
ROLAP	Relationales OLAP
MOLAP	Multidimensionales OLAP
HOLAP	Hybrides OLAP
DOLAP	Desktop OLAP
OLE	Object Linking and Embedding
QBE	Query By Example
SCD	Slowly Changing Dimensions
SQL	Structured Query Language
VB	Visual Basic
VBA	Visual Basic for Applications

Teil I

Grundlagen

1.1 Einführung

1.1.1 Einleitung

Information ist die Grundlage jedes betriebswirtschaftlichen Handelns: Die richtige Information zur richtigen Zeit am richtigen Platz ist ein wichtiger Erfolgsfaktor im Wettbewerb.[1] Data Warehouses unterstützen Entscheidungsträger durch die bereinigte, vereinheitlichte, zeitbezogene und analyseorientierte Abbildung strategisch relevanter Daten.

Während viele der hinter dem Data Warehouse-Konzept stehenden Ideen nicht neu sind, haben technische Weiterentwicklungen und die Einführung neuer Werkzeuge dazu geführt, dass nicht nur Programmierer, sondern auch anwendernahe Bereiche der Informatik oder die Fachbereiche selbst analytische Aufgaben wahrnehmen können. Diese Aufgabenverlagerung verbessert die Informationsqualität, weil der Entscheidungsprozess nicht mehr bei jeder Datenanalyse unterbrochen wird und die Ergebnisse schneller zur Verfügung stehen. Damit Endbenutzer bei der Informationsgewinnung weder über- noch unterfordert werden, müssen die Analysemethoden und Werkzeugklassen auf ihre Fähigkeiten und Anforderungen abgestimmt sein. Programmierte Abfragen sind zwar mächtig, können jedoch nur von Experten erstellt werden. Vordefinierte Berichte sind hingegen unflexibel und beantworten individuelle Fragestellungen häufig nicht befriedigend. An dieser Stelle setzt Online Analytical Processing (OLAP) an. Diese Analysemethode überträgt die Multidimensionalität der betrieblichen Welt auf ein logisches Datenmodell und bereitet die analytischen Daten so auf, dass Endbenutzer auch ohne die Verwendung einer komplexen Abfragesprache flexible Auswertungen erstellen können.

Die meisten OLAP-Datenmodelle basieren auf dem Sternschema, das zwischen Fakten und Dimensionen unterscheidet. Während die Fakten die zentralen betrieblichen Erfolgsgrössen repräsentieren, enthalten die Dimensionen die Kriterien, die ihre Auswahl, Zusammenfassung und Navigation ermöglichen.[2]

Viele strategische Analysemethoden wie etwa die Zeitreihenanalyse basieren nicht nur auf aktuellen, sondern auch auf vergangenen Daten. Damit Endbenutzer Zugriff auf historische Daten haben, sind die Fakten zeitbezogen gespeichert. Ein Aspekt, den viele OLAP-Systeme nicht adäquat berücksichtigen, ist die Zeitabhängigkeit der Dimensionsdaten. Die unzureichende Abbildung von Änderungen in Dimensionsdaten kann das Nachvollziehen vergangener Aus-

[1] Vgl. Behme, Mucksch 1998: S. 5 und 24
[2] Vgl. Lusti 2001: S. 144

wertungen verhindern und erschwert die richtige Interpretation gegenwärtiger Analyseergebnisse.

1.1.2 Forschungsziele

Die vorliegende Arbeit setzt sich mit der Abbildung zeitbezogener Dimensionsdaten in OLAP auseinander. Dabei steht nicht die vollständige Historisierung aller Dimensionsdaten, sondern deren temporale Zuordnung zu den Fakten im Vordergrund. Dazu wird nach der Erörterung bereits bekannter Methoden ein eigener Ansatz zur Behandlung veränderbarer Dimensionsdaten in Form eines Prototyps illustriert und kritisch beurteilt. Die Ziele dieser Arbeit orientieren sich an der Beantwortung der folgenden Fragestellungen:

- **Temporale Auswertungsformen**

  Welche Probleme entstehen durch Dimensionsdatenänderungen? Welche Version der Dimensionsdaten muss bei welcher Art der Fragestellung den Fakten zugeordnet werden?

- **Abbildungsmethoden**

  Können Konzepte aus der temporalen Datenbanktheorie auf Sternschemata übertragen werden? Welche alternativen Methoden zur Darstellung temporaler Auswertungsformen in OLAP gibt es? Inwiefern weichen in diesen Ansätzen die Anforderungen an Benutzer und System von einer konventionellen OLAP-Anwendung ab?

- **Werkzeugprototyp**

  Wie grenzt sich die eigenentwickelte Methode zur Abbildung temporaler Auswertungsformen in OLAP von den bekannten Ansätzen ab? Wie lässt sich die spezifizierte Methode in ein herkömmliches OLAP-System integrieren? Welche Konsequenzen sind hinsichtlich temporaler Funktionalität, Benutzerfreundlichkeit und Systemverhalten aufgrund dieser Integration zu erwarten?

1.1.3 Aufbau

Die Arbeit besteht aus drei Teilen mit jeweils drei Hauptkapiteln. **Teil I** gibt einen Überblick über die Grundlagen analytischer Informationssysteme. Während **Kapitel 1.2** OLAP in den Data Warehouse-Prozess einordnet, führt **Kapitel 1.3** in die bei OLAP angewandte multidimensionale Modellierung ein.

Teil II befasst sich mit der Abbildung temporaler Daten. **Kapitel 2.1** zeigt anhand eines Beispiels, welche Folgen eine Dimensionsdatenänderung auf Analyseergebnisse haben kann. Die anschliessende Einführung der *temporalen Auswertungsformen* diskutiert, welche Möglichkeiten der Zuordnung zwischen den verschiedenen Versionen der Dimensionsdaten und den Fakten im Sternschema sinnvoll sind und in welchem Zusammenhang sie angewandt werden. Der Rest des Kapitels geht auf den Ursprung, die Identifikation und die Übertragung von Datenänderungen ein. Damit schafft dieses Kapitel die Grundlagen, die zum weiteren Verständnis bei der Abbildung verschiedener temporaler Auswertungsformen in OLAP notwendig sind. **Kapitel 2.2** gibt einen Überblick über die in der temporalen Datenbanktheorie bekannten Grundkonzepte zur Modellierung veränderbarer Daten. **Kapitel 2.3** überträgt die in relationalen Datenmodellen angewandten temporalen Konzepte auf das Sternschema und macht einen Vorschlag zur Abbildung und Analyse zeitbezogener Dimensionsdaten. Anschliessend werden bekannte *alternative* Vorgehensweisen zur Darstellung verschiedener temporaler Auswertungsformen in OLAP präsentiert und bewertet.

Teil III führt das *aggregierte Faktenmodell* ein. **Kapitel 3.1** spezifiziert seine Funktionsweise und führt dabei die Anwendung temporaler Aggregate ein. Danach werden die konzeptionellen Grenzen dieses Modellansatzes besprochen. **Kapitel 3.2** zeigt, wie die im vorhergehenden Kapitel spezifizierten Prozesse im Prototyp umgesetzt wurden. Es beschreibt dabei insbesondere die bei der Implementation verwendeten Funktionen zur Definition, Wartung und Analyse der temporalen Aggregate. **Kapitel 3.3** vergleicht schliesslich das aggregierte Faktenmodell mit den in Kapitel 2.3 vorgestellten Methoden und erörtert die noch zu verbessernden Modelleigenschaften.

Mit Ausnahme von Kapitel 1.1, 2.3 und 3.3 enthalten alle Hauptkapitel am Schluss eine Zusammenfassung. Während eine Zusammenfassung im einführenden und abschliessenden Kapitel nicht nötig ist, enthält Kapitel 2.3 am Schluss eine Bewertung, welche die zentralen Kriterien der vorgestellten Konzepte bereits zusammengefasst darstellt.

Der Anhang enthält ein Glossar eingeführter und verwendeter Begriffe. Im Text markiert ein kleines ‣ Dreieck, dass der nachfolgende Begriff im Glossar definiert wird.

1.2 Analytische Informationssysteme

Online Analytical Processing (▸ OLAP) ist eine effiziente Analysemethode, mit welcher Endbenutzer auch ohne Programmierkenntnisse in der Lage sind, komplexe multidimensionale Auswertungen ad-hoc zu definieren. Der Endbenutzerzugriff ist ein wichtiger, aber vergleichsweise kleiner Bereich in einem analytischen Informationssystem. Dieses Kapitel beschreibt deshalb zunächst die Grundlagen zur Bereitstellung analytischer Daten anhand der zentralen Eigenschaften und Funktionen eines ▸ Data Warehouses. Dabei wird die Zeitorientierung analytischer Daten als eine von vier zentralen Merkmalen eines Data Warehouses erörtert. Auf die Behandlung temporaler Daten geht diese Arbeit dann im zweiten und dritten Hauptkapitel ein.

Die Ausführungen zur Beispielarchitektur zeigen, welche Komponenten und Datenverarbeitungsschritte in einem Data Warehouse anzutreffen sind und in welchem Umfeld multidimensionale Datenbanken eingesetzt werden. Anschliessend untersucht die Arbeit, welche Analysemethoden und Werkzeugklassen den verschiedenen Anwendergruppen zur Verfügung stehen. Der letzte Teil dieses Kapitels konzentriert sich schliesslich auf OLAP und beschreibt die zentralen Funktionen aus der konzeptionellen, der externen und der internen Sicht.

1.2.1 Data Warehousing

Ausgangslage

Im Wettbewerb spielt die Zeit eine immer bedeutendere Rolle. Eine Unternehmung muss im Markt schnell und richtig handeln, damit sie sich gegen die Konkurrenz behaupten kann. Mit relevanter und aktueller Information lassen sich qualitativ bessere Entscheidungen treffen.[3] Das Erzielen von Wettbewerbsvorteilen erfordert jedoch eine *effiziente* Analyse von unternehmensinternen und -externen Daten.

Die operativen Datenbanksysteme, die für die Verarbeitung des täglichen Geschäfts zuständig sind, enthalten zwar einen grossen Teil der benötigten Infor-

[3] Vgl. Holthuis 1999: S. 16

mation, eignen sich jedoch schlecht für Datenanalysen. Beeinträchtigt wird die Informationsgewinnung in operativen Systemen durch: [4]

- Heterogene Datenstrukturen
- Mangelnde Datenqualität
- Personalengpässe
- Belastung der bestehenden Infrastruktur
- Fehlende historische Daten

Definition

Ein Data Warehouse setzt bei diesen Mängeln an und ermöglicht es, Information von einer einheitlichen, bereinigten und analyseorientierten Datenbasis zu gewinnen. Data Warehouse-Daten werden meist von den operativen Daten physisch getrennt, damit die operativen Geschäfte nicht durch analytische Tätigkeiten beeinflusst werden.[5]

Der Begriff Data Warehouse wurde erstmals von Inmon definiert: Ein Data Warehouse bezeichnet eine themenorientierte, integrierte, zeitbezogene und dauerhafte Sammlung von Informationen zur Entscheidungsunterstützung des Managements.

- **Themenorientierung**

  Das Datenmodell einer analytischen Datenbank soll nach Geschäftsobjekten wie *Kunde* oder *Produkt* organisiert sein. Die bei operativen Systemen wichtigen innerbetrieblichen, prozessorientierten Abläufe und Funktionen sind bei Data Warehouses nur von untergeordnetem Interesse. Weil sich viele Analyseprozesse nicht vordefinieren lassen, müssen die Daten so vorliegen, dass die Entscheidungsträger sie aus möglichst vielen Blickwinkeln betrachten können. Ein benutzerfreundliches, themenorientiertes Datenmodell trägt dazu bei, dass Entscheidungsträger nicht nur *vordefinierte* Berichte, sondern auch ▸ *Ad-hoc*-Analysen mit geeigneten Werkzeugen selbständig durchführen können.

[4] Vgl. Wieken 2000: S. 13-16 oder Berson, Smith 1997: S. 12
[5] Inmon 1992: S. 25

- **Integration**

  Die Data Warehouse-Daten werden aus verschiedenen Quellen extrahiert, bereinigt und vereinheitlicht. Die Verwaltung aller entscheidungsrelevanter Daten in einem homogenen System mit einheitlichen Datenstrukturen bildet die Grundlage für eine unternehmensweite, konsistente und qualitativ hochwertige Informationsbasis.

- **Zeitorientierung**

  Im Gegensatz zur *zeitpunktgenauen* Betrachtung der Daten in operativen Systemen werden die Daten in Data Warehouses so gespeichert, dass sie über verschiedene *Zeiträume* analysiert werden können. Alte Datenbestände bleiben auf diese Art mehrere Jahre bestehen und lassen sich mit aktuellen Informationen vergleichen.

- **Dauerhaftigkeit**

  Aufgrund des dokumentarischen Charakters analytischer Daten werden diese, abgesehen von der zyklischen Aktualisierung des Integrationsprozesses im Laufe der Zeit, nicht verändert. Der Benutzer soll keine Modifikationen an der Datenbasis vornehmen können.

Die *enge* Data Warehouse-Definition, die sich auf die Datenbasis beschränkt, wurde von verschiedenen Autoren ergänzt.[6] Neben der eigentlichen Datensammlung und deren Verwaltung werden heute sowohl die Anbindung, Extraktion und Transformation operativer und externer Daten als auch die Erweiterung in Richtung Analyse- und Präsentationswerkzeuge in einem weiteren Sinne zum Begriff *Data Warehouse* gezählt.

Ein Data Warehouse im *weiteren Sinn* ist somit kein einzelnes Produkt, sondern vielmehr eine Sammlung von Technologien, die es den Anwendern ermöglicht, schnellere und bessere Entscheidungen zu treffen.[7] Für das Data Warehouse im engeren Sinn, das nur die abteilungsübergreifende analytische Datenbasis umfasst, wird hier der Begriff *zentrales Data Warehouse* verwendet.

[6] Vgl. Schnizer et al. 1999: S. 15
[7] Jarke et al. 2000: S. 1

Architektur

Wegen der bestehenden Infrastruktur, den betrieblichen Anforderungen sowie der in der Regel stetigen Weiterentwicklung hat jede Data Warehouse-Architektur eine eigene Charakteristik. Trotz dieser Individualität können grundlegende Gemeinsamkeiten festgestellt werden, mit welchen sich die Architekturen in vier Grundtypen klassifizieren lassen: virtuelle, zentralisierte, koordinierte und hierarchische Data Warehouse-Architektur.

Abbildung 1.2.1 zeigt die relevanten Komponenten und Prozesse anhand einer hierarchischen Architektur. Die analytischen Daten werden in dieser Architektur nicht nur im *zentralen Data Warehouse*, sondern zusätzlich in ▸*Data Marts* verwaltet. Data Marts sind subjekt- beziehungsweise abteilungsspezifische Data Warehouses, die dezentral von Fachabteilungen bewirtschaftet werden. Die gesamten analytischen Daten werden erst zentral gespeichert und anschliessend den Data Marts zur Verfügung gestellt. Die Trennung analytischer Daten auf verschiedene Ebenen ermöglicht eine zielgerichtete Aufgabenverteilung des Systems: Das zentrale Data Warehouse stellt die Datenintegration, die Data Marts den fachbereichsabhängigen Sachbezug sicher.

Die Verbindung eines zentralen Data Warehouses mit Data Marts ist vorteilhaft, wenn die Daten nicht nur lokal, sondern auch abteilungsübergreifend genutzt werden. Dafür stellt diese Architektur die höchsten Anforderungen an Ressourcen, Entwicklungs- und Wartungsaufwand.

Die redundante Datenhaltung auf mehreren analytischen Ebenen kann bei der Wartung und Weiterentwicklung eines Data Warehouses eine wichtige Rolle spielen. Die Speicherung historischer Daten im zentralen Data Warehouse ermöglicht zum Beispiel Modellanpassungen auf Data Mart-Ebene, ohne dass historische Informationen verloren gehen oder in der geänderten Datenbank fehlen.

Die folgenden Punkte beschreiben die Datenverarbeitungsschritte im Data Warehouse:

1. **Extraktion**

   Die analytisch relevanten Daten werden aus unterschiedlichen in- und externen Quellsystemen extrahiert und zur Bereinigung in der sogenannten *Staging Area* temporär gespeichert.[8] Regelsysteme bereinigen und vereinheitlichen unterschiedliche Formate oder Bezeichnungen in den Rohdaten.

[8] Vgl. Bange, Schnizer 2000: S. 10-11

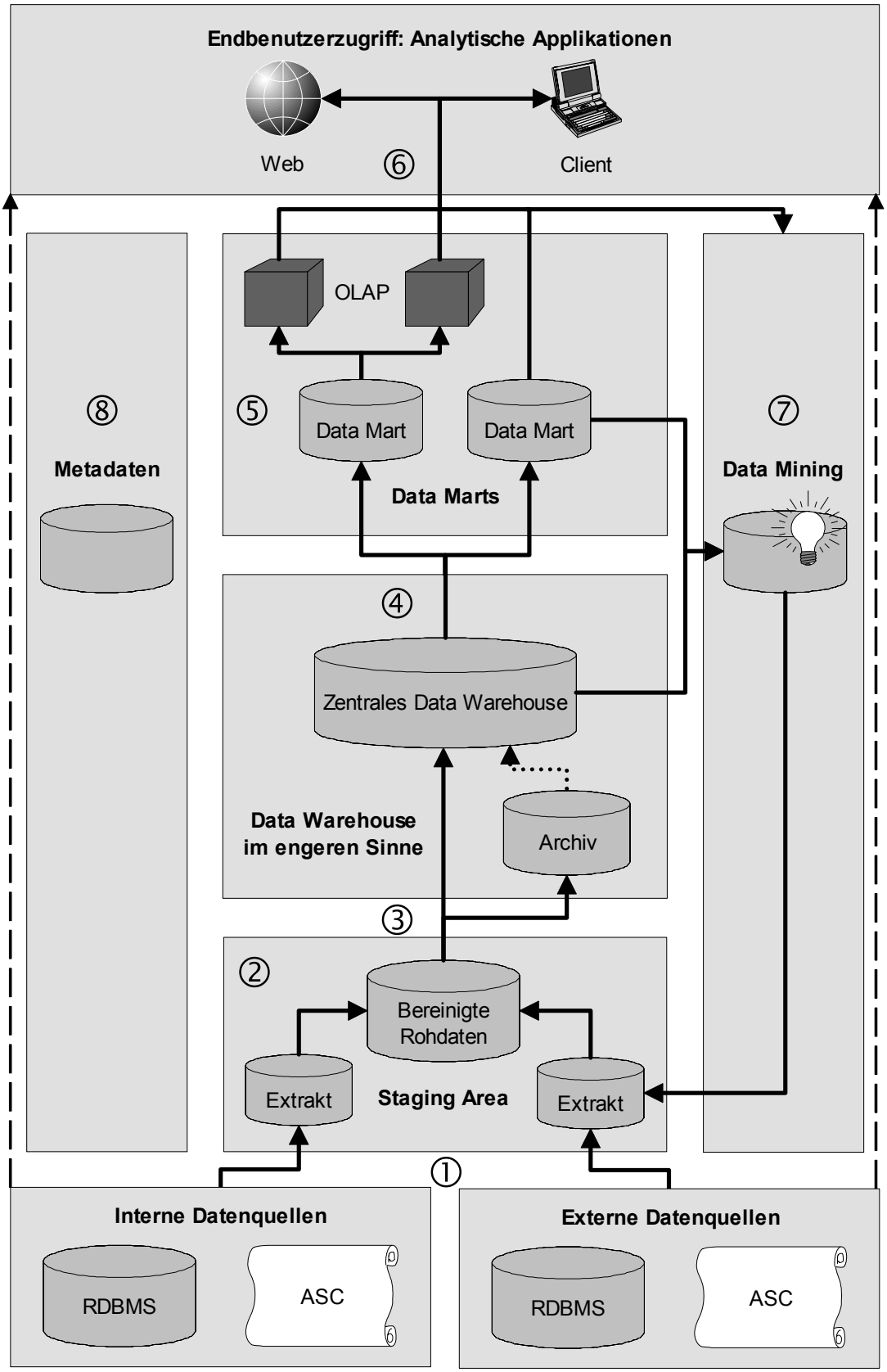


Abbildung 1.2.1: Beispielarchitektur

Das Extraktionssystem darf verbleibende Inkonsistenzen nicht übernehmen, sondern muss diese zur manuellen Fehlerbehebung dokumentieren. Die Informationsqualität nimmt zu, wenn die protokollierten Fehler entweder durch Korrekturen im Quellsystem oder durch neue Regeln im Extraktions- und Transformationsprozess bereinigt werden.

2. Transformation

Die Daten müssen teilweise angepasst und verdichtet gespeichert oder aus anderen Werten abgeleitet werden, damit sie sich an die richtige Stelle im Data Warehouse einfügen lassen. Daten, die später historisch betrachtet werden sollen, werden mit temporalen Attributen ergänzt.

3. Laden

Der Ladeprozess übermittelt die transformierten Daten an das Data Warehouse. Die periodische Aktualisierung des Datenbestandes erfolgt entweder vollständig oder inkrementell. Die inkrementelle Aktualisierung benötigt komplexe Mechanismen, mit welchen sie Änderungen in den Quellsystemen verfolgen kann. Das vollständige Laden ist zeitaufwändiger und nur möglich, falls die alten Daten in den Quellsystemen noch vorhanden sind. Damit sich vergangene Daten in jedem Fall vollständig wiederherstellen lassen, muss neben dem zentralen Data Warehouse auch ein Archiv bedient werden.

4. Datenbereitstellung

Das zentrale Data Warehouse stellt die Daten für alle abteilungsspezifischen Data Marts zur Verfügung. Auch hier werden die Daten periodisch vollständig oder inkrementell aktualisiert. Die inkrementelle Aktualisierung ist an dieser Stelle weniger problematisch, weil Änderungen über Extraktions-, Transformations- und Ladeprotokolle (ETL), die in der Staging Area erstellt werden, oder über die zeitabhängige Datenspeicherung im zentralen Data Warehouse abrufbar sind.

5. Multidimensionale Datenbanken (OLAP)

Die Data Mart-Daten können je nach Verwendungszweck in multidimensionale Datenbanken geladen oder direkt mit geeigneten Werkzeugen analysiert werden. Dedizierte multidimensionale Datenbanken werden dann eingesetzt, wenn das Datenvolumen begrenzt ist und die Anwender besonders hohe Anforderungen an die Abfrageeffizienz stellen.

6. Endbenutzerzugriff

Der Endbenutzer kann die vorhandenen Daten mit dedizierten Analysewerkzeugen oder über verbreitete Büroanwendungen (z.B. Webbrowser, Tabellenkalkulationswerkzeug, lokale DBMS) abrufen.

7. Data Mining

Aufgrund bestimmter Fragestellungen werden die benötigten Daten aus einem Data Mart oder dem zentralen Data Warehouse in eine davon getrennte Datenbank geladen, problemspezifisch transformiert und mit ▸Data Mining-Methoden analysiert. Die Erkenntnisse liegen dann einerseits dem Benutzer vor, können andererseits aber auch wieder ins Data Warehouse integriert werden.

8. Metadaten

▸Metadaten beschreiben die eigentlichen Daten und Prozesse und werden in einem Metadatenbanksystem gesammelt.[9] Beispiele für Metadaten sind Namen von Tabellen und Attributen, Zugriffsrechte, Aktualisierungszeitpunkte oder Indizes. Die Hauptaufgabe der Metadaten ist einerseits die Minimierung des Aufwandes für Entwicklung und Unterhalt eines Data Warehouses und andererseits die Verbesserung und Vereinfachung der Informationsgewinnung für sämtliche Benutzer.[10] Die in Abbildung 1.2.1 dargestellte zentrale Metadatenbank stellt einen Idealzustand dar, der in der Praxis kaum zu finden ist. Häufig verwendet jede Datenbankverwaltung eigene Metadaten, die dann mehr oder weniger gut zwischen den unterschiedlichen Systemen ausgetauscht werden können.

Operatives Berichtswesen

In Abbildung 1.2.1 zeigen zwei (gestrichelte) Pfeile von den Datenquellen direkt auf die analytischen Applikationen. Damit wird angedeutet, dass das operative Berichtswesen nicht durch ein Data Warehouse ersetzt werden kann, denn analytische Informationssysteme unterstützen operative Entscheidungen nur eingeschränkt: Einerseits sind die Daten nicht aktuell, andererseits können benötigte Detailinformationen bei der Verdichtung im ETL-Prozess verloren gehen. Der Nutzen eines Data Warehouses in operativen Bereichen liegt besonders im vereinheitlichten Datenbestand, in der Möglichkeit, externe Daten zu verwenden, und im Auffinden von Mängeln im operativen Datenbestand.

[9] Lusti 2001: S. 417
[10] Vgl. Staud et al. 1999

1.2.2 Analyseprozess

Mit der Entwicklung des Data Warehouse-Konzepts konnte die Grundlage für ein komplettes Informationssystem geschaffen werden. Das Data Warehouse stellt die benötigten Daten in einer einheitlichen Form, zeitabhängig, dauerhaft und sachbezogen sämtlichen analytischen Anwendungen zur Verfügung. Die Daten sind von grossem Nutzen, wenn sie von vielen Anwendern analysiert werden und häufig in die Entscheidungsfindung einfliessen. Die *Analysemethoden*, *Werkzeuge* und *Fähigkeiten der Endbenutzer* müssen dabei so aufeinander abgestimmt sein, dass die Benutzer weder unter- noch überfordert sind. Damit sich die jeweils adäquaten Analysemethoden und Werkzeuge den Benutzern zuordnen lassen, müssen zuerst die Anwendergruppen und deren Anforderungen definiert werden.

Anwenderklassen

Die Benutzergruppen unterscheiden sich einerseits hinsichtlich der Inhalte und Detaillierung der benötigten Informationen, andererseits nach Art und Häufigkeit der Nutzung.[11] Während die *inhaltliche Ausprägung* der Informationsnutzung durch die Datenmodellierung berücksichtigt wird, trägt die *Art der Datenanalyse* den Fähigkeiten des Anwenders Rechnung. Im Allgemeinen werden drei Anwenderklassen unterschieden:[12]

- Gelegenheitsanwender (benötigen einen einfachen Informationszugang)
- Fachanwender (erstellen Analysen ohne Programmierkenntnisse)
- Spezialisten (definieren komplexe Analysen)

Eine generelle, personenbezogene Abgrenzung, wie sie hier vorgestellt wird, ist in der Praxis nicht einfach. Ein Anwender kann in einem Bereich zu den Spezialisten gehören, in einem anderen jedoch nur ein Fachbenutzer sein.[13]

Methoden

Analysen können mit prozeduralen und deklarativen Abfragesprachen, mit OLAP oder mit Methoden des Data Mining erstellt werden. Während Gelegenheitsanwender selten neue Abfragen selbst formulieren, sondern vordefinierte

[11] Vgl. Wieken 1999: S. 35
[12] Vgl. Kimball 1998: S. 390 oder Berson, Smith 1997: S. 224
[13] Vgl. Wieken 1999: S. 38

oder parametrisierbare Berichte nutzen, sind programmierte Abfragen und Data Mining-Methoden aufgrund ihrer Komplexität den Spezialisten vorbehalten.

Fachanwender sind darauf angewiesen, möglichst flexible Ad-hoc-Abfragen zu erstellen. Da sie sich jedoch nicht mit komplizierten Abfragesprachen und Datenmodellen auseinandersetzen möchten, mussten in der Vergangenheit viele Analysen Spezialisten übergeben werden. Bei diesem Vorgehen ist der Zeitaufwand hoch und die Analysequalität mangelhaft, weil der Weg der Entscheidungsfindung meist nicht nur über ein einziges Auswertungsresultat führt. Fachanwender müssen deshalb Analysemethoden einsetzen, die es ihnen erlauben, Fragestellungen selbständig, schrittweise und interaktiv zu beantworten.

Online Analytical Processing (OLAP) entspricht heute den Anforderungen der Fachbenutzer zur Erstellung benutzerfreundlicher, flexibler, mehrdimensionaler Auswertungen am ehesten, weil es die Erstellung von Ad-hoc-Abfragen auch ohne die Verwendung einer komplexen Abfragesyntax ermöglicht. Anwendungen, die OLAP unterstützen, gehören deshalb zu den zentralen Komponenten in einem Data Warehouse. Kapitel 1.2.3 geht anschliessend auf die wichtigsten OLAP-Funktionen ein.

Werkzeuge

Die verschiedenen Analysemethoden müssen den entsprechenden Benutzern in Form von konkreten Anwendungen zur Verfügung stehen. Obwohl die Auswahl und Entwicklung geeigneter Frontends weniger Ressourcen benötigt als etwa die Informationsselektion und -speicherung, gehören die Analysewerkzeuge zu den entscheidenden Erfolgsfaktoren eines Data Warehouse-Projekts. Ein System, das komplex in der Bedienung ist oder einen zu kleinen Funktionsumfang hat, wird zu wenig oder gar nicht genutzt. Zu den verbreiteten Werkzeugklassen zählen:

- OLAP-Frontends
- Tabellenkalkulationsprogramme
- Berichtshefte
- Webbrowser
- Entwicklungsumgebungen

Abbildung 1.2.2 zeigt eine Einordnung der Werkzeugklassen in Abhängigkeit von den Ausprägungen *Analysefreiheit des Anwenders* (Flexibilität) und *Geschwindigkeit der Anwendungsentwicklung*. Während der Funktionsumfang in Standardanwendungen wie OLAP-Frontends oder Tabellenkalkulationspro-

gramme stark vorgegeben ist, können mit Entwicklungsumgebungen individuell zugeschnittene Analysewerkzeuge realisiert werden. Der Aufwand bei der Anwendungsentwicklung ist dementsprechend bei den Entwicklungsumgebungen am höchsten.

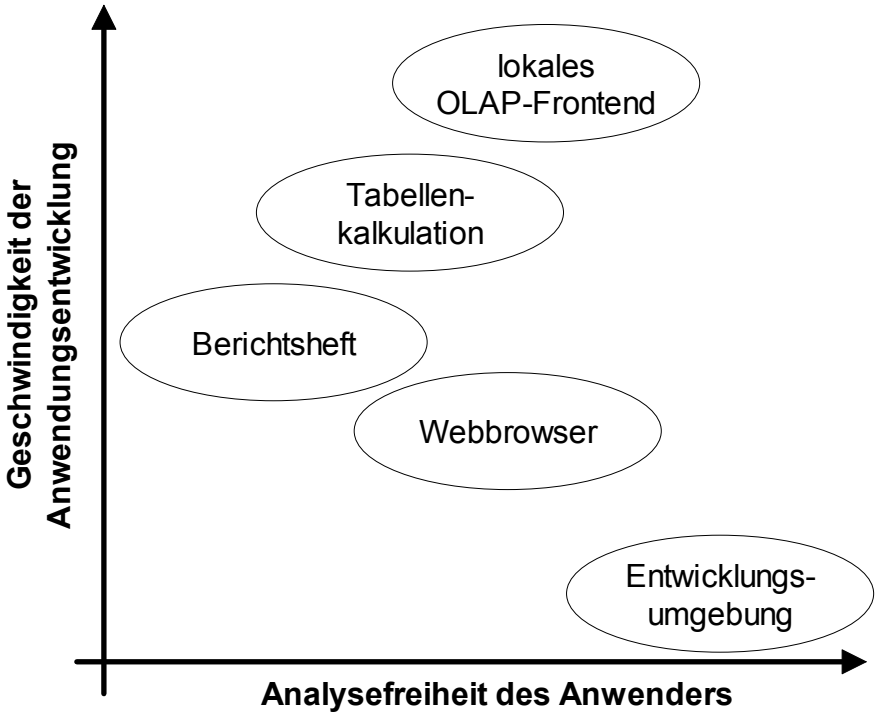


Abbildung 1.2.2: Analysefreiheit vs. Geschwindigkeit[14]

Tabellenkalkulationsprogramme oder Internetapplikationen werden auch zur Anwendungsentwicklung verwendet, so dass die Zuordnung der Werkzeugklassen jeweils vom Verwendungszweck abhängig ist.

Damit die Endbenutzer OLAP besser in ihr gewohntes Arbeitsumfeld integrieren können, unterstützen viele Hersteller diese Analysemethode nicht nur in dedizierten Frontends, sondern auch in webbasierten Anwendungen oder in Tabellenkalkulationsprogrammen. Daneben ist es in bestimmten Systemen auch möglich eigene OLAP-Werkzeuge zu entwickeln.

[14] Vgl. Schnizer et al. 1999: S. 63

1.2.3 OLAP

Das Konzept wurde vom Begründer der relationalen Datenbanktechnologie, E. F. Codd, 1993 eingeführt.[15] Obwohl OLAP unabhängig von der Data Warehouse-Technologie entwickelt wurde, stehen die beiden Systeme heute in einem engen Zusammenhang. OLAP wird besonders zur Datenanalyse auf der Data Mart-Ebene genutzt.

In operativen Umgebungen wird vor allem die relationale Datenbanktechnologie eingesetzt. Die Daten in diesen transaktionsorientierten Systemen sind änderungsintensiv: Integritätsbedingungen und normalisierte Schemata stellen die Konsistenz der Daten sicher und optimieren den Speicherbedarf.

In analytischen Umgebungen werden relationale Datenbanken sowohl im zentralen Data Warehouse als auch auf Data Mart-Ebene verwendet. Die gute ▸Skalierbarkeit, die Möglichkeit, auch grosse Datenbestände verwalten zu können, sowie die vertraute Technologie sind besonders bei der Datenbereitstellung von Vorteil. Der Datenzugriff hingegen ist für Endbenutzer komplex oder aber eingeschränkt. Eine komplizierte Abfragesyntax sowie nicht überschaubare und analyseinadäquate Datenmodelle erschweren den betrieblichen Anwendern den Zugang zu flexiblen Ad-hoc-Analysen. Vordefinierte Analysen und Berichte können zwar den benutzerunfreundlichen Datenzugriff umgehen, sind aber wenig flexibel. Abfragen auf normalisierten Schemata sind zudem ressourcenintensiv und langsam. Die Verfügbarkeit und Abfragegeschwindigkeit eines Systems kann sich direkt auf den interaktiven Analyseprozess und somit auf die Qualität der Entscheidungsunterstützung auswirken. Abgesehen von der Abfragegeschwindigkeit und den adäquaten Analysemethoden sollte ein OLAP-System auch mehrbenutzerfähig und multidimensional sein sowie alle benötigten Daten zur Verfügung stellen.[16]

OLAP-Systeme kombinieren eine einfache Datennavigation in einem mehrdimensionalen Modell mit Analysefunktionen, wie sie etwa in Tabellenkalkulationsprogrammen zu finden sind. Der nächsten Abschnitte gehen mit der konzeptionellen, der externen und der internen Sicht auf dieses mehrdimensionale Modell ein, erörtern die zentralen Analysefunktionen und beschreiben die Alternativen bei der Speicherung der OLAP-Daten.

---

[15] Vgl. Lusti 2001: S. 153
[16] Vgl. Oehler 2000: S. 33: Oehler beschreibt die FASMI-Regeln

Konzeptionelle Sicht: Mehrdimensionales Modell

In mehrdimensionalen Systemen wird zwischen *identifizierenden* und *quantifizierenden* Attributen unterschieden.[17] Die quantifizierenden Attribute entsprechen den zu analysierenden managementkritischen Grössen und werden ▸ *Fakten* oder ▸ *Indikatoren* genannt. Ein Fakt allein kann noch keine bedeutende Information liefern. Die Aussage, dass zum Beispiel der Umsatz gleich eintausend Franken ist, wirft sofort neue Fragen auf: In welchem Zeitraum, mit welchem Produkt oder durch welche Abteilung wurde der Betrag erwirtschaftet? Die Informationen liefern die identifizierenden Attribute und entsprechen im mehrdimensionalen Modell den *Hierarchieebenen*, durch welche die Auswertungs*dimensionen* strukturiert sind. Das Ziel von Hierarchien ist die Festlegung von Konsolidierungspfaden, auf denen die Datenbank aggregierte Zahlen berechnen kann.[18] Die logische Verknüpfung von ausgewählten Fakten und den entsprechenden ▸ Dimensionen wird ▸ *Würfel* oder ▸ *Cube* genannt, obwohl meistens mehr als drei Dimensionen vorhanden sind.

Abbildung 1.2.3 zeigt eine mögliche Struktur einer Produktdimension mit deren Hierarchieebenen und den in diesem Beispiel zulässigen Werten. Die dargestellte Struktur könnte auf horizontaler als auch auf vertikaler Ebene ergänzt werden.

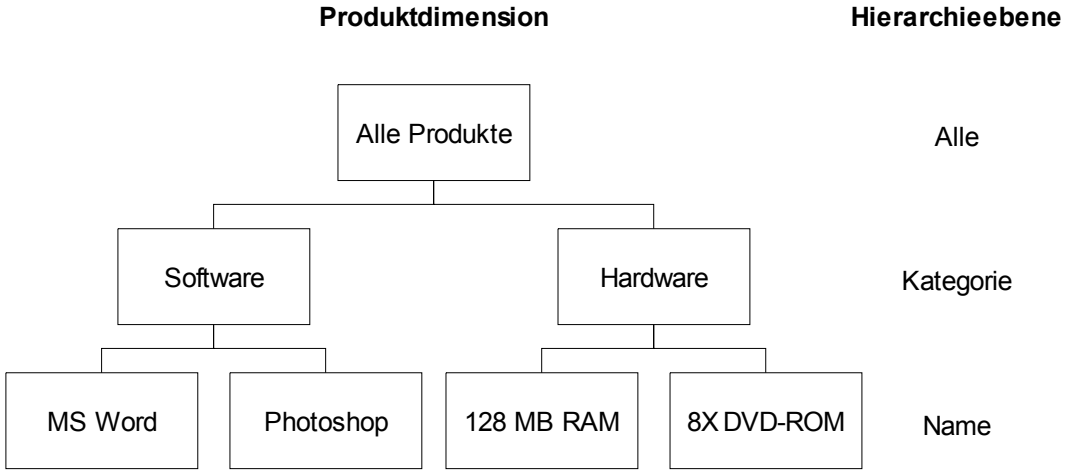


Abbildung 1.2.3: Beispiel einer Produktdimension und deren Hierarchie

Abbildung 1.2.4 illustriert eine zweidimensionale Ansicht des Umsatzes nach den Dimensionen PRODUKT (Hierarchiestufe: Name) und ZEIT (Hierarchiestufe: Jahr). Jeweils eine Spalte und eine Zeile enthält die summierten Werte (TOTAL). Aus dieser Sicht wird deutlich, dass Fakten numerische, aggregierbare Grössen darstellen sollten. Wird der Umsatz zum Beispiel nicht auf der

[17] Vgl. Oehler 2000: S. 46
[18] Engels 1996: S. 22

Hierarchiestufe *Name*, sondern auf der Ebene *Kategorie* analysiert, muss die OLAP-Anwendung die Fakten aus den Detaildaten verdichten.

		ZEIT			
		1998	1999	2000	TOTAL Jahr
PRODUKT	MS Word	100	200	250	550
	Photoshop	250	500	800	1550
	TOTAL Software	350	700	1050	2100

Abbildung 1.2.4: Umsatzanalyse nach den Dimensionen Zeit und Produkt

In bestimmten Situationen kann es notwendig sein, Fakten nicht nur nach Dimensionswerten, sondern auch in Abhängigkeit zu anderen Fakten auszuwerten. In diesem Fall müssen aus *kontinuierlichen*, numerischen Werten *diskrete* Werte gebildet werden. Die Einteilung der Umsätze in verschiedene Grössenklassen würde im dargestellten Beispiel die Analyse eines anderen Fakts nach Kunden und diesen Umsatzgruppen ermöglichen.

Obwohl Bildschirm- oder Papierausgaben auf zwei Dimensionen beschränkt sind, lassen sich mit OLAP mehrdimensionale Analysen erstellen, indem zwei Dimensionen über eine Spalte resp. Zeile angezeigt und die anderen Dimensionen auf ein einzelnes Element fixiert werden.[19] Eine solche mehrdimensionale Analyse könnte zum Beispiel die Betrachtung des Umsatzes in Abhängigkeit von den Perioden (Zeit) und Produkten (vgl. Abbildung 1.2.4) darstellen. Damit in dieser Abfrage mehr als zwei Dimensionen benötigt werden, sind zum Beispiel nur Kunden mit einer guten Bonität zu berücksichtigen, die in der Region *Nordwestschweiz* und über den Vertriebsweg *Direktverkauf* Waren bezogen haben.

Die Werte aller Fakten sind somit über jede Kombinationsmöglichkeit der Dimensionselemente abrufbar. Diese flexible Betrachtung von betriebswirtschaftlich interessanten Grössen aus unterschiedlichen Blickwinkeln ermöglicht dem Anwender, auch unvorhersehbare Analysepfade abzubilden.

[19] Vgl. Engels 1996: S. 16

Externe Sicht: Navigation und Analyse

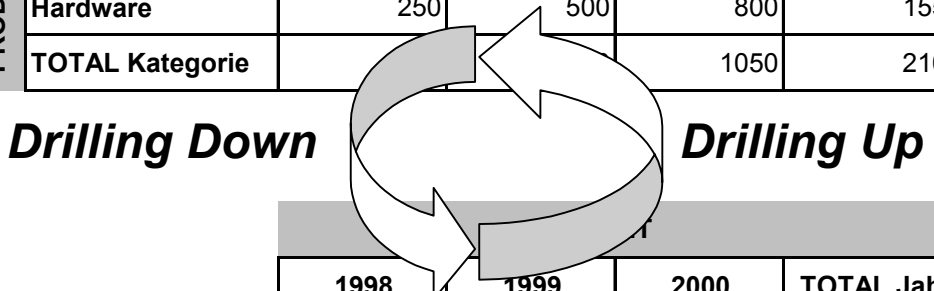
Datennavigation

Die Navigation im Datenbestand erfolgt nahezu vollständig mausgesteuert. Der Benutzer wählt die beiden darzustellenden Dimensionen sowie einen Fakt aus und platziert sie mittels Drag- und Drop-Verfahren in die entsprechenden Zeilen, Spalten und Datenfelder.[20] Diese Ausgangsabfrage stellt zunächst die oberste Hierarchiestufe der beiden Dimensionen und dementsprechend eine Übersicht mit hochverdichteten Daten dar, von der aus der Benutzer in detailliertere Ebenen vorstossen oder andere Sichten auswählen kann. Die wichtigsten Navigationsmöglichkeiten dazu sind Drilling Down and Up sowie Slicing and Dicing.

- **Drilling Down and Up**

  Der Benutzer kann die Fakten entlang der Dimensionshierarchie detaillierter darstellen (▸ Drilling Down) oder verdichten (▸ Drilling Up).

		ZEIT			
		1998	1999	2000	TOTAL Jahr
PRODUKT	Software	100	200	250	550
	Hardware	250	500	800	1550
	TOTAL Kategorie			1050	2100



		1998	1999	2000	TOTAL Jahr
PRODUKT	128 MB RAM	350	700	1050	2100
	8X DVD-ROM	1400	1100	800	3300
	TOTAL Hardware	1750	1800	1850	5400

Abbildung 1.2.5: Drilling Down and Up

[20] Die Bedienung ist produktspezifisch. Die Erläuterungen beziehen sich auf Microsoft Excel oder Cognos Powerplay.

In Abbildung 1.2.5 wird die Drilling Down-Funktion in der Produktdimension von der Ebene *Kategorie* auf die detailliertere Ebene *Name* und die umgekehrt wirkende Drilling Up-Funktion dargestellt.

- **Slicing and Dicing**

  Das Filtern des Datenbestandes nach einzelnen Dimensionselementen wird Slicing (Schneiden) genannt. Damit werden nur ausgewählte Schnitte aus dem mehrdimensionalen Modell betrachtet. Dicing (Würfeln) wechselt die dargestellten Dimensionsachsen, ohne die restlichen Einstellungen zu verändern.[21]

- **Weitere Navigationsmöglichkeiten**

  OLAP-Daten enthalten oft keine Einzeltransaktionen, sondern bereits auf der detailliertesten Ebene zusammengefasste Werte. Damit der Benutzer trotzdem die Möglichkeit hat, alle Informationen zu analysieren, kann er in einigen OLAP-Systemen mittels ▸ *Drilling Through* auf das zentrale Data Warehouse oder die operative Datenbasis zugreifen.

  In bestimmten Fällen kann es notwendig sein, mehrere Datenwürfel für eine Analyse zu kombinieren. Der Benutzer kann mittels ▸ *Drilling Across*, sofern dies vorgesehen ist, auf die Daten anderer Würfel zugreifen.

  Ergänzend soll an dieser Stelle noch die Möglichkeit zur Verwendung von dedizierten, multidimensionalen Abfragesprachen erwähnt werden, die einige OLAP-Systeme anbieten. Diese sind aber mit ihrer an SQL angelehnten Syntax mehr für die Entwicklung multidimensionaler Anwendungen als für die benutzerfreundliche Ad-hoc-Analyse interessant und sollen im Folgenden nicht weiter untersucht werden.

Datenanalyse

Anwender können mit der zur Verfügung stehenden Navigationsfunktionalität der OLAP-Werkzeuge einfach auf detaillierte und verdichtete Werte zugreifen. Zum Beispiel lässt sich der Reingewinn jeder Filiale im Kanton Basel im Jahr 1998 ohne aufwändige Analysevorgänge ermitteln.

Viele Fragestellungen sind jedoch komplexer und stellen einen Bedarf an die Analyse, der über das einfache Abrufen von Daten hinausgeht. Die zuvor gestellte Abfrage könnte etwa so umformuliert werden, dass die Filialen im Kan-

[21] Vgl. Oehler 2000: S. 177

ton Basel gesucht sind, die 1998 einen überdurchschnittlichen Reingewinn erwirtschaftet haben. Solche Analysen erfordern neben dem Auffinden der Grunddaten auch die Anwendung weitergehender Funktionen.

Zu den verbreiteten *Analysefunktionen* gehören:

- Absolute und relative Darstellung
- Komplexe Filter
- Hervorhebung von Abweichungen
- Mathematische Grundoperationen
- Ranglisten
- Statistische Funktionen
- Visualisierung

Einige Hersteller haben in ihren OLAP-Produkten Schnittstellen zu anderen Anwendungen wie etwa Data Mining-Werkzeugen oder Tabellenkalkulationsprogrammen geschaffen und verknüpfen damit die OLAP-Funktionalität mit deren Analysemöglichkeiten.

Interne Sicht: Datenverwaltung

Mehrdimensionale Daten werden in der *konzeptionellen Sicht* einheitlich als kartesisches Produkt dargestellt, das sich aus der Schnittmenge der einzelnen Dimensionen ergibt.[22] Die physische Speicherung dieser Daten ist grundsätzlich mit zwei unterschiedlichen Technologien möglich: relational oder multidimensional. Abgesehen von den in Kapitel 2.3.2 beschriebenen alternativen Ansätzen zur Abbildung zeitbezogener Dimensionsdaten[23] basieren die Ausführungen zu temporalen Daten im zweiten und dritten Hauptkapitel auf dem relationalen Speichermodell.

Relationales OLAP

Relationales OLAP (▸ROLAP) nutzt die klassische relationale Technologie zur Speicherung der Daten. Die Multidimensionalität wird dabei durch besondere Modellierungsmethoden wie dem ▸Stern- oder ▸Schneeflockenschema erzielt. OLAP-Server übersetzen die mehrdimensionalen Abfragen in SQL-Befehle und leiten diese an das relationale Datenbanksystem weiter. Die Da-

[22] Vgl. Schnizer et al. 1999: S. 47
[23] Vgl. Kapitel 2.3.2: S. 87

tenbank verarbeitet die Abfrage und sendet die angeforderten Daten zurück an den OLAP-Server. Dieser bereitet das Resultat mehrdimensional auf und überträgt es schliesslich in die Frontend-Anwendung des Benutzers (vgl. Abbildung 1.2.6). Abgesehen von Metadaten werden keine weiteren Daten im OLAP-Server physisch gespeichert. Dadurch eignet sich ROLAP auch zur Verwaltung von Daten mit hoher Periodizität.[24] Besondere Stärken sind zudem die relativ gute Skalierbarkeit und damit verbunden die Möglichkeit, auch sehr grosse Datenbestände verwalten zu können.

Relationale Datenbanksysteme sind nicht auf die Verarbeitung multidimensionaler Datenstrukturen optimiert, wodurch einige Aggregations- und Berechnungsformeln nur unzureichend mit SQL definiert werden können.[25] Andere Funktionen wie das Transaktionskonzept werden von den abfragenden Zugriffen gar nicht benötigt.

Multidimensionales OLAP

Die Datenspeicherung im multidimensionalen OLAP (▸ MOLAP) erfolgt in proprietären Datenbanksystemen, die eine auf multidimensionale Verarbeitung optimierte Array-Struktur verwenden. Bei der Erstellung von Würfeln, also den mehrdimensionalen Datenbanken, werden bei MOLAP im Gegensatz zum ROLAP nicht nur Metadaten, sondern alle benötigten Objektdaten vom Quellsystem übertragen.

Die Ausrichtung des physischen an das konzeptionelle Modell begünstigt vor allem die Abfrageeffizienz. Die Nachteile sind im Vergleich zum ROLAP eine geringere Periodizität, ein eingeschränktes Datenvolumen und eine redundante Datenhaltung.

Hybrides OLAP

Die Vorzüge beider Technologien können durch den kombinierten Einsatz in einer OLAP-Umgebung genutzt werden: Abgeleitete Daten, die ein begrenztes Speichervolumen beanspruchen, zur Analyse aber relativ häufig verwendet werden, speichert das System in der multidimensionalen Datenbank. Der Zugriff auf die selten benötigten aber umfangreichen Detaildaten erfolgt in diesem Fall über die relationale Datenbank. Dieser Ansatz wird hybrides OLAP (▸ HOLAP) genannt und von immer mehr Herstellern unterstützt. Natürlich werden dadurch nicht alle Nachteile umgangen. Der langsame Zugriff auf die Detaildaten sowie die redundante Datenhaltung bleiben bestehen. Ein Aspekt,

[24] Vgl. Marx 1999: S. 72
[25] Vgl. Wieken 1999: S. 86

der besonders bei der Entwicklung zum Tragen kommt, stellt die Risikoverminderung dar. Mit der HOLAP-Variante brauchen sich die Entwickler nicht schon zu Beginn für die eine oder andere Technologie festzulegen, sondern können auch später, im laufenden Betrieb noch entscheiden, ob sich die Beschränkung auf ROLAP oder MOLAP lohnen würde.

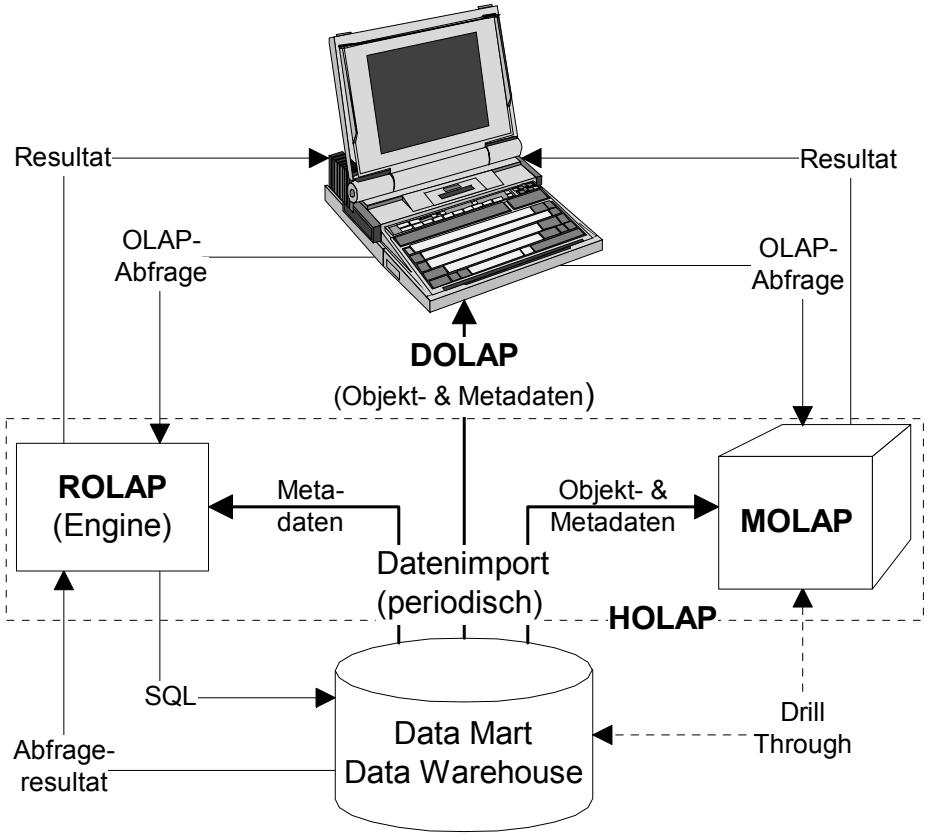


Abbildung 1.2.6: OLAP-Varianten

Desktop-OLAP

Eine weitere Variante stellt das Desktop-OLAP (▸ DOLAP) dar, bei dem die Daten von einer Datenbank auf dem Client transferiert und anschliessend serverunabhängig analysiert werden können. Die Datenmenge ist dementsprechend limitiert. DOLAP kann sowohl auf der relationalen als auch auf der multidimensionalen Technologie basieren.[26]

[26] Vgl. Schnizer et al. 1999: S. 54

Zusammenfassung

Operative Systeme sind für die taktische und strategische Entscheidungsunterstützung ungeeignet. Der analytische Zugriff auf operative Daten wird besonders durch heterogene Strukturen und eine mangelnde Datenqualität erschwert. Eine historische Betrachtung der Information ist nicht möglich, weil operative Daten häufig nicht zeitabhängig gespeichert werden.

In einem Data Warehouse werden unternehmensinterne und -externe Informationen so aufbereitet, dass Anwender auf einen qualitativ hochwertigen, homogenen Datenbestand zugreifen können. Bei der periodischen Aktualisierung des analytischen Datenbestandes werden die betrieblichen Erfolgsfaktoren nicht überschrieben, sondern zeitbezogen gespeichert. Die Datenverwaltung kann zentral in einem System, dezentral in mehreren fachbereichsspezifischen Systemen oder auch kombiniert in einem mehrstufigen System erfolgen.

Damit die bereitgestellten Daten möglichst häufig genutzt werden, müssen die Analysemethoden und die Werkzeuge auf die Anforderungen und Fähigkeiten der Anwender abgestimmt sein. OLAP ist eine Ad-hoc-Analysemethode, die durch verschiedene Werkzeugklassen unterstützt wird und den Anwendern flexible, mehrdimensionale Auswertungen ohne besondere Kenntnisse einer Abfragesyntax ermöglicht.

Die Verwaltung der OLAP-Daten erfolgt relational oder multidimensional. Multidimensionales OLAP verwendet proprietäre, auf die multidimensionale Verarbeitung optimierte Datenstrukturen. Relationales OLAP basiert auf der relationalen Datenbanktechnologie und bildet die Multidimensionalität durch besondere Modellierungsmethoden ab.

1.3 Datenmodellierung

Dieses Kapitel beschreibt die besonderen Eigenschaften der Datenmodelle im Data Warehouse und führt mit der dimensionalen Datenmodellierung ein zentrales Konzept zur Abbildung analytischer Daten ein. Der letzte Teil des Kapitels erörtert die wichtigsten Prozesse bei der Definition und Aktualisierung eines OLAP-Datenwürfels.

1.3.1 Datenmodelle für Data Warehouses

Datenmodelle sind die zentralen Elemente bei der Gestaltung einer Data Warehouse-Umgebung.[27] Abhängig von der gewählten Architektur werden Modelle für ein zentrales Data Warehouse und/oder mehrere Data Marts definiert. Die folgenden Abschnitte grenzen die Anforderungen dieser Modelle voneinander ab und zeigen damit, auf welchen Voraussetzungen das in OLAP verwendete dimensionale Modell basiert.

Analytische Datenbanken unterscheiden sich von operativen Systemen vor allem durch ihre Zielsetzung: Operative Daten sind änderungsintensiv, analytische abfrageintensiv. Diese unterschiedliche Zielverfolgung wirkt sich entscheidend auf die Datenstruktur und somit auf die Datenmodellierung aus. Normalisierte Datenmodelle, die sich in transaktionsorientierten operativen Systemen bewährt haben, eignen sich nicht für Ad-hoc-Analysen, weil sie unübersichtlich und abfrageineffizient sind.

Anforderungen an analytische Datenmodelle

Die Anforderungen an operative und analytische Datenmodelle weichen nicht nur voneinander ab, sondern sind teilweise vollkommen konträr. Die folgenden Merkmale stellen die zentralen Modellierungskriterien in analytischen Systemen dar:[28]

- **Integration**

  Die verschiedenen, teilweise heterogen vorliegenden Datenstrukturen der Quellsysteme sind im Data Warehouse vereinheitlicht und konsistent abzu-

[27] Vgl. Wieken 1999: S. 147
[28] Vgl. Wieken 1999: S. 150

bilden. Dabei müssen nur die Daten berücksichtigt werden, die für analytische Prozesse notwendig sind.

- **Verständlichkeit**

  Das Datenmodell muss übersichtlich sein und sich nach betriebswirtschaftlichen Sachverhalten orientieren. Die Ausrichtung der Datenstrukturen auf die Bedürfnisse der Anwender erleichtert das Verständnis und ermöglicht eine einfachere Erstellung von Auswertungen, die einen direkten Datenzugriff erfordern. Auch analytische Anwendungen, die nicht auf eine dedizierte Modellierung angewiesen sind, können von einer endbenutzerorientierten Strukturierung der Daten profitieren, weil die Sicht auf das zugrundeliegende Datenmodell für den Benutzer weniger aufwändig transformiert werden muss.

- **Historisierung**

  Ein analytisches Informationssystem muss aktuelle *und* vergangene Daten über einen bestimmten Zeitraum abbilden können. Einerseits bleiben Auswertungen auch noch zu einem späteren Zeitpunkt nachvollziehbar, andererseits können viele analytische Fragestellungen erst durch die historische Betrachtung der relevanten Kennzahlen beantwortet werden.[29]

- **Detaillierungsgrad**

  Es ist wichtig, dass die Anwender die Daten in dem von ihnen gewünschten Detaillierungsgrad vorfinden. Viele operative Daten sind jedoch für analytische Anwendungen zu detailliert und können verdichtet in das Data Warehouse übernommen werden. Dieses Vorgehen ist laufzeit- und speichereffizient, birgt jedoch die Gefahr, dass im Nachhinein bestimmte Detailinformationen nicht abrufbar sind.

- **Zugriff**

  Der Zugriff auf analytische Daten muss *schnell* sein. Die interaktive Entscheidungsfindung wird durch lange Wartezeiten unterbrochen und führt im schlimmsten Fall zu einem Verzicht der Systemnutzung. Die Anwendung adäquater Modellierungsmethoden wie Denormalisierung oder Speicherung vorberechneter Werte kann die Abfrageeffizienz bedeutend erhöhen.

---

[29] Vgl. Kapitel 1.2.1: S. 7

Aufteilung der Modellierungsanforderungen

Die Daten werden in einer analytischen Umgebung einerseits in einem zentralen Data Warehouse und andererseits in mehreren fachbereichsspezifischen Data Marts verwaltet (hierarchische Architektur). Diese mehrstufige Datenhaltung ermöglicht die Verteilung der Aufgaben auf die verschiedenen Ebenen und führt zu einer grösseren Spezialisierung bei der Gestaltung der jeweiligen Datenmodelle.

Die *Historisierung* und der adäquate *Detaillierungsgrad* müssen auf allen Ebenen berücksichtigt werden. Das zentrale Data Warehouse unterliegt als Datenlieferant aller Data Marts jedoch immer strengeren Modellierungsauflagen. Deshalb sind die Daten im zentralen Data Warehouse mindestens so detailliert und historisiert wie jene in den Data Marts.

Die *Integration* aller Datenstrukturen gehört zu den Modellierungsanforderungen, die das zentrale Data Warehouse erfüllen muss. Ein Data Mart beinhaltet nur noch eine fachbereichsspezifische Teilmenge der gesamten Datenstrukturen. Dabei sollte ein Data Mart so modelliert werden, dass Data Mart-übergreifende Abfragen nur selten notwendig sind.[30]

Der endbenutzergesteuerte Zugriff auf die analytischen Daten erfolgt in einem Data Warehouse meist auf der Data Mart-Ebene. Die *Verständlichkeit* und der effiziente *Datenzugriff* sind deshalb besonders bei der Modellierung der Data Marts zu berücksichtigen. Diese Kriterien zur Endbenutzerfreundlichkeit fliessen dann in Kapitel 2.3.3 und 3.3.3 in die Bewertung ein, welche die Konzepte zur Behandlung temporaler Daten in OLAP vergleichen.

Modellierung des zentralen Data Warehouse

Das zentrale Data Warehouse wird durch ein einheitliches Datenmodell beschrieben, das mindestens die von den Data Marts geforderte Detailgenauigkeit aufweist. Dabei werden die Daten so abgebildet, dass veränderbare Werte zeitbezogen analysierbar sind. Die zuvor dargestellten Anforderungen schränken die Modellierung des zentralen Data Warehouses ein, bestimmen jedoch keine eindeutige Modellierungsstrategie. Abhängig von Grösse und Verwendungszweck haben sich in der Praxis zwei Gestaltungsvarianten etabliert.[31]

Die erste Variante orientiert sich an den operativen Datenbanken und verwendet ein normalisiertes oder nur leicht denormalisiertes Datenmodell. Dieser Ansatz berücksichtigt besonders die Anforderungen an die *Speicherkapazität*

[30] Lusti 2001: S. 137
[31] Vgl. Firestone 1998

und den *Wartungsaufwand*, weil die Datenbank wenig redundante Daten speichern und verwalten muss.

Die zweite Variante richtet sich eher nach der Modellierung der Data Marts und umfasst alle Dimensionen und Faktentabellen auf einem übergeordneten, grösst möglichen Nenner. Aus dieser Sicht ist ein zentrales Data Warehouse nichts anderes als eine Vereinigung aller Data Marts.[32] Diese Modellierungsvariante vereinfacht den Ladeprozess der Data Marts und ermöglicht einen benutzerfreundlicheren Zugriff auf die Daten des zentralen Data Warehouses.

Modellierung von Data Marts

Data Marts werden durch fachbereichsspezifische Datenmodelle beschrieben, die sowohl benutzerfreundlich als auch effizient analysierbar sind und die Daten in ihrem historischen Zusammenhang abbilden. Data Marts enthalten tendenziell weniger Detaildaten sowie mehr vorberechnete und abgeleitete Werte als ein zentrales Data Warehouse.

Data Marts werden (multi-)dimensional modelliert. Modelle wie das Sternschema oder das Schneeflockenschema ermöglichen flexible und effiziente Analysen unternehmensrelevanter Kennzahlen in Abhängigkeit verschiedener Einflussfaktoren. Die dimensionale Modellierung setzt das bei OLAP angewandte logische Schema auf die relationale Datenbanktechnologie um.

Während die Modellierung des zentralen Data Warehouse keinen Einfluss auf die weiteren Ausführungen hat und deshalb nicht weiter betrachtet wird, beschreibt das folgende Kapitel die Konzipierung dimensionaler Datenmodelle, um anschliessend daraus die zentralen OLAP-Verwaltungsfunktionen zu erörtern.

[32] Vgl. Kimball 1998: S. 27

1.3.2 Dimensionale Modellierung

Das Sternschema

Die Modellierung relationaler Datenbanken auf Data Mart-Ebene richtet sich nach dem dimensionalen Modell, bei dem zwischen *quantifizierenden* und *identifizierenden* Attributen unterschieden wird.[33] Im Mittelpunkt steht die Faktentabelle, welche die quantifizierenden Attribute enthält. Jeder Teilschlüssel der Faktentabelle verweist auf einen Primärschlüssel einer Dimensionstabelle. Abbildung 1.3.1 zeigt ein einfaches Sternschema mit vier Dimensionstabellen (GEOGRAFIE, ZEIT, KUNDE und PRODUKT) und drei Kennzahlen (Umsatz, Kosten und Gewinn). Die Schlüsselattribute sind jeweils unterstrichen und in fetter Schrift dargestellt. Der Beziehungstyp wird mit der Krähenfuss-Notation dargestellt.

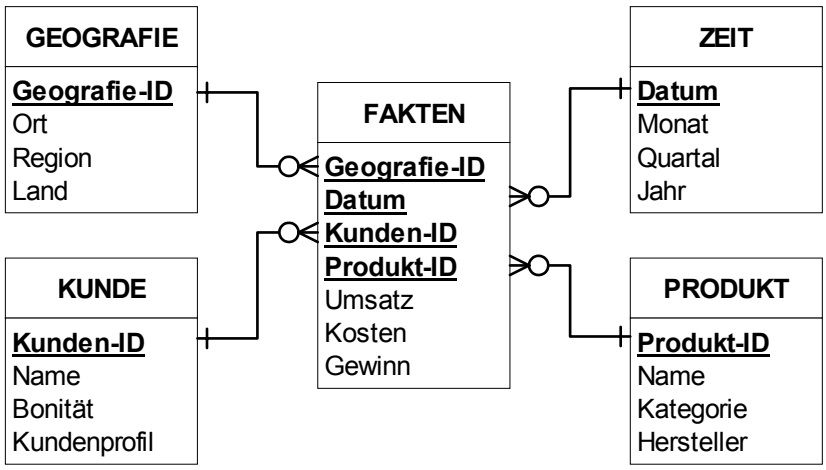


Abbildung 1.3.1: Ein Sternschemabeispiel

Fakten

Fakten sind meist aggregierbare, kontinuierliche Attribute, die ein betriebliches Erfolgskriterium messen.[34] Die in Abbildung 1.3.1 dargestellten Kennzahlen stellen Bewegungszahlen dar, die über bestimmte *Zeiträume* ermittelt werden. Andere Fakten wie etwa ein Lagerbestand oder ein Aktienkurs sind Bestandesgrössen, die zu jedem *Zeitpunkt* abrufbar sind. Bewegungs- und Bestandesgrössen werden deshalb häufig unterschiedlich über Zeiträume verdichtet: Bewegungsgrössen lassen sich summieren, während bei Bestandesgrössen eher Mit-

[33] Vgl. Kapitel 1.2.3: S. 17
[34] Vgl. Lusti 2001: S. 144

telwerte, Minimum-, Maximum- oder andere statistische Funktionen zum Tragen kommen.[35]

Dimensionen

Dimensionen enthalten meistens symbolische und diskrete Kriterien, die zur Auswahl, Zusammenfassung und Navigation der Fakten dienen.[36] Die Dimensionselemente sind häufig hierarchisch strukturiert. In Abbildung 1.3.1 stehen zum Beispiel die Elemente der Dimension GEOGRAFIE in einer hierarchischen Beziehung zueinander. Jeder Wert des Basiselements Ort gehört zu einer bestimmten Region und diese wiederum zu einem bestimmten Land. Der Wechsel zu einer übergeordneten Ebene *verdichtet*, die Wahl einer untergeordneten Ebene *detailliert* die Kennzahlen. Dieses Vorgehen entspricht dem Drilling Up beziehungsweise Drilling Down bei OLAP. Eine wesentliche Schwäche des Sternschemas ist die fehlende Darstellungsmöglichkeit der hierarchischen Beziehungen einzelner Dimensionselemente.

Bei der Modellierung der Dimensionen sind drei *Spezialfälle* zu beachten:

- Die *Zeitdimension* nimmt eine besondere Stellung bei der Modellierung eines analytischen Datenmodells ein. Sie ermöglicht die zeitbezogene Analyse aller Fakten und ist deshalb in den meisten Sternschemata vorhanden. Im Gegensatz zu anderen Dimensionen kann ihre Struktur nicht beliebig verändert werden. Das Datum 20.2.2002 gehört zum Beispiel zum Monat Februar und zum Jahr 2002. Hingegen sind Zuordnungen benutzerdefinierter Ergänzungen wie das Fiskaljahr oder die Ferienzeit nicht explizit vorgegeben.
- *Rekursive Dimensionen* können Elemente verschiedener Hierarchieebenen in einem einzigen Attribut speichern. Im Attribut Mitarbeiter können zum Beispiel die Namen ohne die Angabe expliziter Hierarchieebenen gespeichert werden (vgl. Abbildung 1.3.2). Die Hierarchie lässt sich durch die Angabe des nächsten Vorgesetzten rekursiv erstellen. Dieses Vorgehen ist dann vorteilhaft, wenn die Tiefe des Hierarchiebaumes unausgeglichen ist.

[35] Vgl. Stock 2001: S. 87
[36] Vgl. Lusti 2001: S. 144

Mitarbeiter-ID	Mitarbeiter	Vorgesetzter
100	Thommen	300
200	Brauer	400
300	Moser	500
400	Liechti	
500	Hug	400
600	Meier	200

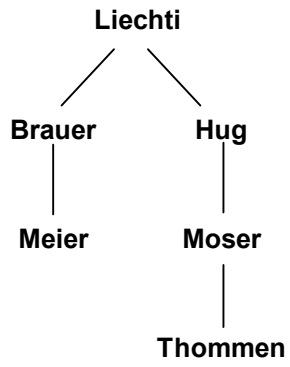


Abbildung 1.3.2: Rekursiv erstellte Dimensionshierarchie

- Unter der *Kennzahlendimension* wird manchmal auch die Faktentabelle verstanden. Hier soll jedoch die Möglichkeit zur Verwendung einer von der Faktentabelle getrennten Kennzahldimension angesprochen werden. Die ursprüngliche Faktentabelle kann in eine Faktentabelle und eine Kennzahlendimension aufgespaltet werden, wie dies in Abbildung 1.3.3 dargestellt ist. Vorteilhaft an dieser Modellierungsvariante ist die Möglichkeit, Fakten zu einem späteren Zeitpunkt hinzuzufügen, ohne dass leere Einträge für frühere Perioden entstehen, wie dies beim Einfügen einer neuen Spalte in der Faktentabelle der Fall wäre. Im Gegenzug können die Speicheranforderungen durch die Vervielfachung der Datensätze in der Faktentabelle steigen.

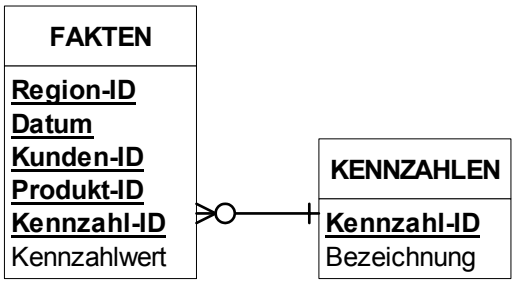


Abbildung 1.3.3: Kennzahlendimension

Abfragen in multidimensional strukturierten Datenbanken

Der Zugriff auf die Daten eines Sternschemas, das physisch in einer relationalen Datenbank abgebildet ist, erfolgt in der Regel mit SQL. Viele analytische Anwendungen verbergen diese Abfragesyntax vor dem Endbenutzer und stellen den SQL-Code nach Angabe von Dimensionen und Fakten automatisch zusammen. Diese Automatisierung ist einfach, weil die Grundstruktur beim Sternschema im Gegensatz zu normalisierten Schemata einheitlich ist.

Eine SQL-Abfrage, die auf ein Sternschema wie in Abbildung 1.3.1 zugreift, könnte folgende Struktur haben:

```
TRANSFORM <Aggregatsfunktion>(<FAKTENTABELLE>.<Fakt>)
SELECT    <Zeilendimensionsattribut>
FROM      <FAKTENTABELLE>,
          <ZEILENDIMENSION>, <SPALTENDIMENSION>,
          <FILTERDIMENSION 1>, ..., <FILTERDIMENSION n>
WHERE     <FAKTENTABELLE>.<Zeilendimensionsschlüssel> =
             <ZEILENDIMENSION>.<Zeilendimensionsschlüssel>
  AND     <FAKTENTABELLE>.<Spaltendimensionsschlüssel> =
             <SPALTENDIMENSION>.<Spaltendimensionsschlüssel>
  AND     <FAKTENTABELLE>.<Filterdimensionsschlüssel 1> =
             <FILTERDIMENSION 1>.<Filterdimensionsschlüssel 1>
          ...
  AND     <FAKTENTABELLE>.<Filterdimensionsschlüssel n> =
             <FILTERDIMENSION n>.<Filterdimensionsschlüssel n>
  AND     <FILTERDIMENSION 1>.<Filterdimensionsattribut 1> =
             <Filterwert 1>
          ...
  AND     <FILTERDIMENSION n>.<Filterdimensionsattribut n> =
             <Filterwert n>
ORDER BY  <Zeilendimensionsattribut>,
             <Spaltendimensionsattribut>
GROUP BY  <Zeilendimensionsattribut>
PIVOT     <Spaltendimensionsattribut>;
```

Der hier verwendete SQL-Code verwendet eine Erweiterung von Microsoft Access (**TRANSFORM**, **PIVOT**), mit der sich die Daten ohne weiteren Programmieraufwand als Kreuztabelle darstellen lassen. Angaben in spitzigen Klammern <...> sind Parameter, die entweder vom Benutzer durch die Angabe der Dimensions- und Faktelemente sowie mit Hilfe der Metadaten bestimmt werden.

Beispielabfrage[37]

Der Anwender möchte wissen, wie sich der Umsatz in den letzten Jahren für alle Produktkategorien in der Region Nordwestschweiz entwickelt hat. Er zieht dazu in seinem OLAP-Werkzeug das Jahr auf die *Spaltenüberschrift*, die Produktkategorie auf die *Zeilenüberschrift* und den Umsatz in die freie Fläche seines Tabellenblatts. Als *Filter* wählt er aus der Dimensionskategorie Region die Nordwestschweiz aus. Anhand dieser Information und der Metadaten des Sternschemas kann die Applikation den folgenden SQL-Code generieren:

```
TRANSFORM SUM(FAKTEN.Umsatz)
SELECT    PRODUKT.Kategorie
FROM      FAKTEN, PRODUKT, ZEIT, GEOGRAFIE
WHERE     FAKTEN.Produkt-ID = PRODUKT.Produkt-ID
  AND     FAKTEN.Datum = ZEIT.Datum
  AND     FAKTEN.Geografie-ID = GEOGRAFIE.Geografie-ID
  AND     GEOGRAFIE.Region = „Nordwestschweiz"
ORDER BY  PRODUKT.Kategorie, ZEIT.Jahr
GROUP BY  PRODUKT.Kategorie
PIVOT     ZEIT.Jahr;
```

Abbildung 1.3.4 zeigt das Resultat der zuvor definierten Abfrage anhand eines Zahlenbeispiels. Navigationsfunktionen von OLAP wie Drilling Up and Down oder Slicing and Dicing verändern im Abfragegerüst die jeweiligen Parameterwerte. Das Abfragegerüst selbst bleibt bei jeder Abfrage unverändert.

		ZEIT		
		1999	**2000**	**2001**
PRODUKT	**Schokolade**	600000	721000	698000
	Backwaren	800000	815000	845000
	Milchprodukte	248900	215000	187400
	Gefrierfleisch	573000	478600	412000
	Kassenprodukte	100200	145000	123000

Abbildung 1.3.4: Resultat der Beispielabfrage

[37] basiert auf dem Sternschema der Abb. 1.3.1

Alternative Modellierungsvarianten

Neben dem Sternschema gibt es weitere Ansätze zur Modellierung der Data Marts.[38] Die extremste Form ist eine *einzige Tabelle*, die alle Daten in der ersten Normalform abbildet.[39] Eine derart stark denormalisierte Sicht hat gegenüber dem Sternschema den Vorteil, dass Abfragen häufig effizienter sind. Die Benutzer müssen sich beim direkten Datenbankzugriff zwar keine Gedanken über die Tabellenverbindungen machen, können jedoch nicht mehr unmittelbar zwischen Fakt- und Dimensionswerten unterscheiden. Besonders nachteilig wirken sich die höheren Anforderungen an die Speicherkapazität aus, wodurch diese Modellierungsvariante nur in kleineren Data Marts genutzt werden kann.

Entgegengesetzte Ziele verfolgt das *Schneeflockenschema*, indem es die Dimensionstabellen normalisiert abbildet. Voluminöse beschreibende Attribute können auf diese Art abgespalten und in getrennte Tabellen eingefügt werden. Dieses Vorgehen ist speichereffizienter und fortschreibungsfreundlicher, verschlechtert jedoch die Verständlichkeit und erhöht die Komplexität bei der Abfrageerstellung. Die Veränderung der Laufzeiteffizienz ist abhängig von der auszuführenden Abfrage. Werden nur Dimensionsattribute benötigt, die in direkter Beziehung zu der Faktentabelle stehen, dann müssen bei den Verbundoperationen weniger Daten als beim Sternschema verarbeitet werden und die Abfrage wird schneller ausgeführt. Zusätzliche Verknüpfungsoperationen verlangsamen hingegen die Abfrageeffizienz. Das Normalisieren der Dimensionstabellen wird häufig dann angewandt, wenn Dimensionen sehr gross sind oder der Data Mart hauptsächlich als Datenlieferant für MOLAP-Datenbanken dient.[40]

1.3.3 Entwicklung und Aktualisierung eines Würfels

OLAP-Werkzeuge stellen die Daten unabhängig von ihrer physischen Speicherung multidimensional dar. Falls die auszuwertenden Daten bereits in einem multidimensionalen Schema vorliegen, sind die Abfrageergebnisse ohne zusätzliche Transformationsschritte und somit schnell verfügbar. In einem Data Warehouse basieren deshalb viele OLAP-Anwendungen auf multidimensional modellierten Data Marts.

Die folgenden Abschnitte beschreiben die grundlegenden Schritte bei der Entwicklung und Aktualisierung eines OLAP-Würfels anhand des Sternschema-

[38] Vgl. Gluchowski 1997: S. 64
[39] Vgl. Wieken 1999: S. 185
[40] Vgl. Wieken 1999: S. 184

beispiels aus Abbildung 1.3.1. Obwohl sich die Ausführungen auf den OLAP-Server von Microsoft[41] beziehen, können die wichtigsten Funktionen auch auf Produkte anderer Hersteller übertragen werden. Nicht beachtet werden Funktionen, wie die Datenpartitionierung oder die Benutzerrechteverwaltung, die in Bezug auf die Modellierung oder Aktualisierung eines OLAP-Würfels eine untergeordnete Rolle spielen.

Definition der OLAP-Strukturen

Ein Data Mart beinhaltet sämtliche Dimensions- und Faktentabellen, die für die Entwicklung aller fachbereichsspezifischen OLAP-Würfel erforderlich sind. Ein Würfel benötigt dabei genau eine Faktentabelle und mehrere Dimensionstabellen. Während eine Faktentabelle meistens spezifisch auf einen OLAP-Würfel ausgerichtet ist, kann eine Dimensionstabelle wie etwa die Zeitdimension in mehreren Würfeln verwendet werden.

1. Quelldaten definieren

Die Bestimmung der Quelldatenbank erlaubt den Zugriff auf die in das Modell zu integrierenden Dimensions- und Faktattribute. Die Quelldatenbank entspricht in einer Data Warehouse-Umgebung dem Data Mart.

2. Fakten auswählen

Aus den in der Quelldatenbank vorhandenen Tabellen können die gewünschte Faktentabelle und die darin enthaltenen Fakten ausgewählt werden. Zur Erstellung des Beispielwürfels werden die Fakten Umsatz und Kosten ausgewählt. Der abgeleitete Fakt Gewinn wird später definiert.

3. Dimensionen und Hierarchiestrukturen bestimmen

Eine Dimension kann aus Attributen mehrerer Tabellen (Schneeflockenschema) oder einer einzelnen Tabelle (Sternschema) zusammengestellt sein. Nach der Wahl des Quellschemas können die Elemente einer Dimension bestimmt werden. Es sind nur Dimensionsattribute auswählbar, die in Beziehung zu den Faktattributen stehen. Beispiel: Ort, Region und Land für die Dimension GEOGRAFIE. Das Attribut Geografie-ID dient nur zur Verknüpfung der Dimensionstabelle mit der Faktentabelle und wird nicht als eigenes Dimensionselement übernommen. Verbundattribute sind meist numerisch,

[41] Microsoft OLAP Services 2000

damit das Datenvolumen der Faktentabelle nicht unnötig vergrössert wird. Anschliessend kann die Dimensionshierarchie nach den gewünschten Verdichtungspfaden der Fakten strukturiert werden. Der Definitionsprozess wiederholt sich solange, bis alle Dimensionen bestimmt sind.

4. Modell anpassen

Modellanpassungen sind notwendig, wenn die von der OLAP-Anwendung bestimmten Standardeinstellungen nicht mit den Anforderungen der Anwender übereinstimmen. Zum Beispiel sollte überprüft werden, ob die ▸ *Aggregatsfunktionen* und *Tabellenverknüpfungen* richtig gesetzt wurden. Des Weiteren lassen sich zusätzliche, berechnete Fakten definieren. Dazu gehören unter anderen Durchschnittswerte, die mit Hilfe des Quotienten SUMME(Fakt)/ANZAHL(Fakt) bestimmbar sind. Die OLAP-Anwendung leitet den im Sternschemabeispiel enthaltenen Fakt Gewinn aus der Differenz Umsatz - Kosten ab.

5. Speichermodell bestimmen

Das physische Speichermodell bestimmt, *wie* und *wo* die Daten des OLAP-Würfels gespeichert werden. Während die würfelbeschreibenden Metadaten immer in der OLAP-Datenbank verwaltet werden, bleiben die Objektdaten bei *ROLAP* in der ursprünglichen, relationalen Datenbank. *MOLAP* verwaltet hingegen auch die Objektdaten in der proprietären, multidimensionalen OLAP-Datenbank. Das *HOLAP*-Speichermodell schliesslich verwendet beide Verfahren, indem es Detaildaten relational und zusammengefasste Werte multidimensional abspeichert.

6. Aggregate definieren

Die Faktentabelle enthält ausschliesslich *detaillierte* Daten. Jede *aggregierte* Sicht, die zum Beispiel durch eine Drilling Up-Operation angefordert wird, muss zur Laufzeit berechnet werden. Damit die OLAP-Anwendung gleiche Abfragen nicht wiederholt berechnen muss, kann sie bestimmte Abfrageresultate im Voraus ermitteln und diese zur Laufzeit den Benutzern in Form von ▸ *materialisierten Sichten* zur Verfügung stellen. Im Gegensatz zu ▸ *virtuellen Sichten*, die nur die Abfragestruktur enthalten, umfassen materialisierte Sichten das eigentliche Abfrageresultat.[42] Die OLAP-Anwendung greift bei Verfügbarkeit direkt auf die vorberechneten Werte zu

[42] Vgl. Gupta, Mumick 1999a: S. 3

und verbessert damit die Antwortzeit. Die Verwaltung und der Zugriff auf die materialisierten Sichten werden vor den Endbenutzern verborgen.

Materialisierte Sichten benötigen im Gegensatz zu virtuellen Sichten zusätzlichen Speicherplatz und mehr Zeit für die periodische Datenaktualisierung.[43] Weil ihr Grenznutzen zudem mit jeder zusätzlichen Sicht abnimmt[44], sollten bei Kapazitäts- oder Zeitengpässen nur die ▸Aggregationen materialisiert werden, die häufig genutzt oder besonders viel Rechenzeit erfordern.

Die Einflussnahme auf die Gestaltung präaggregierter Daten ist produktabhängig. Microsoft OLAP Services erlaubt zum Beispiel nur die Festlegung des zusätzlichen Speicherbedarfs oder des erwarteten Leistungsgewinns bei der Abfrageeffizienz. Explizite Angaben wie „aggregiere die Fakten nach den Dimensionselementen Jahr, Hersteller und Ort“ sind nicht möglich.

7. Daten laden

Nachdem das OLAP-System die Dimensions- und Faktentabellen definiert hat, kann es die benötigten Daten laden. Abhängig vom gewählten Speichermodell werden die detaillierten und die aggregierten Daten in die OLAP-Datenbank übertragen (MOLAP) oder nur die Aggregate berechnet und anschliessend in den Data Mart zurückgeschrieben (ROLAP).

Aktualisierung der OLAP-Daten

Die OLAP-Daten müssen periodisch aktualisiert werden, damit auch neue oder geänderte Daten in die Analysen einfliessen können. Die meisten OLAP-Werkzeuge unterstützen sowohl das *vollständige* als auch das *inkrementelle* Aktualisieren der Quelldaten.

- **Vollständige Aktualisierung**

  Die vollständige Aktualisierungsmethode löscht die bestehenden Daten und lädt sie anschliessend neu in die OLAP-Datenbank. Die vorberechneten Daten der Aggregatstabellen werden nach dem Laden der Detaildaten neu definiert. Der vollständige Aktualisierungsprozess stellt nicht nur alle neuen Daten bereit, sondern aktualisiert auch bereits vorhandene Daten.

---

[43] Vgl. Gupta, Mumick 1999b: S. 43
[44] Vgl. Thomsen et al. 1999: S. 121

- **Inkrementelle Aktualisierung**

  Die inkrementelle Aktualisierungsmethode lädt nur die Daten in die OLAP-Datenbank, die seit dem letzten Ladenvorgang in der Quelldatenbank neu hinzugekommen sind. Die Aggregatstabellen werden mit den neuen Daten ergänzt und aktualisiert. Bereits bestehende Detaildaten werden durch den Aktualisierungsprozess nicht tangiert.

Die inkrementelle Aktualisierungsmethode ist effizienter, weil weniger Daten zu verarbeiten sind. Da diese Fortschreibungsmethode lediglich neue Quelldaten in das Zielsystem einfügt, ist sie zur Aktualisierung bereits bestehender Detaildaten nicht geeignet. Veränderungen in der Zieldatenbank muss das System daher mit der vollständigen Aktualisierungsmethode abbilden. Änderungen werden zum Beispiel durch Korrekturen oder nachträgliches Einfügen von Datensätzen verursacht. Besonders häufig sind Dimensionsdatenänderungen. Während die Fakten bei einem periodischen Aktualisierungsprozess durch die zeitabhängige Speicherung nur ergänzt werden, sind die Dimensionsdaten häufig nicht zeitbezogen abgebildet und werden deshalb bei einer Änderung mit dem aktuellen Wert überschrieben. Auch wenn die vollständige Aktualisierungsmethode nur partiell, zum Beispiel auf die Dimensionstabellen angewandt wird, müssen auch die aggregierten Daten der materialisierten Sichten zumindest partiell neu berechnet werden.

Zusammenfassung

Operative und analytische Systeme stellen unterschiedliche Anforderungen an die Datenmodellierung. Normalisierte Datenmodelle, die vielfach in operativen Umgebungen eingesetzt werden, sind zwar fortschreibungsfreundlich, aber schlecht verständlich und abfrageineffizient.

Dimensionale Datenmodelle wie das Sternschema oder das Schneeflockenschema entsprechen den Bedürfnissen der Endbenutzer in analytischen Umgebungen besser, weil die zentralen Kennzahlen in direkter Abhängigkeit zu den relevanten betrieblichen Ausprägungen abgebildet sind. Die Reduktion der Datenobjekte auf eine Fakten- und wenige Dimensionstabellen verbessert neben der Benutzerfreundlichkeit auch die Abfrageeffizienz. Anwender können Adhoc-Analysen zudem ohne Kenntnisse einer Abfragesprache erstellen, weil sich Abfragen in dimensionalen Datenmodellen einfach vorstrukturieren lassen.

Analytische Informationssysteme müssen betriebliche Erfolgsgrössen oft in ihrem historischen Kontext abbilden. In vielen Sternschemata werden die Fakten deshalb in Abhängigkeit einer Zeitdimension gespeichert. Die Attribute der Zeitdimension bestimmen die Gültigkeitsperioden der Fakten in verschiedenen Verdichtungsstufen.

Damit nicht alle Berechnungen zur Laufzeit erstellt werden müssen, definieren OLAP-Systeme zusätzliche Tabellen zur Speicherung aggregierter Abfrageresultate. Die Anwendung pflegt den verdichteten Datenbestand parallel zu Detaildaten und greift bei der Abfrageverarbeitung selbständig auf ihn zu.

Analytische Daten werden periodisch aktualisiert. OLAP-Systeme unterstützen neben der vollständigen Datenaktualisierung auch partielle Methoden. Die inkrementelle Aktualisierung lädt nur neue Daten in die OLAP-Datenbank und ergänzt die vorberechneten Abfrageresultate. Änderungen an bereits bestehenden Daten können allerdings nur durch eine vollständige Aktualisierung abgebildet werden.

Teil II

Temporale Daten in OLAP

2.1 Datenänderungen

Eine zentrale Anforderung an analytische Informationssysteme ist die Zeitorientierung. Das folgende Kapitel untersucht, welche Auswirkungen Datenänderungen im dimensionalen Modell haben und zeigt mit der Definition der temporalen Auswertungsformen, welche historischen Betrachtungsweisen bei Dimensionsdatenänderungen möglich sind. Voraussetzung für die Abbildung von Datenänderungen in einem analytischen System ist deren Identifikation im Quellsystem. Der letzte Teil des Kapitels befasst sich deshalb mit Methoden der Änderungsverfolgung.

2.1.1 Problematik dynamischer Datenmodelle

Viele betriebliche Entscheidungsprozesse erfordern eine historische Betrachtung der zentralen Erfolgsfaktoren. Die *zeitpunktgenaue Betrachtung* eines Aktienkurses ist zum Beispiel wenig aussagekräftig, wenn keine früheren Kursstände verfügbar sind. Erst der *historische Vergleich* der zu untersuchenden Grössen lässt Aussagen zu wie *„der Aktienkurs ist gesunken“* oder *„die Aktie ist heute nur noch halb soviel wert wie letztes Jahr“*. In einem Data Warehouse werden die zentralen Erfolgsfaktoren deshalb nicht überschrieben, sobald neue Werte verfügbar sind, sondern durch eine zeitbezogene Speicherung ergänzt. Abbildung 2.1.1 zeigt ein Beispiel einer Faktentabelle, in der alle Fakten mit Hilfe des Teilschlüssels Datum zeitbezogen gespeichert sind.

Datum	Kunden-ID	Produkt-ID	Geografie-ID	Umsatz
01.02.2002	4711	1234	101	20
01.02.2002	5555	5236	145	50
...	...	...	...	...
15.02.2002	4711	1234	101	40

Abbildung 2.1.1: Beispiel einer Faktentabelle mit zeitbezogenen Daten

Auch wenn der Kunde 4711 zwei Wochen nach seinem letzten Besuch in der selben Filiale das gleiche Produkt kauft, bleibt die alte Information bestehen, weil sich der Erfassungszeitraum verändert hat. Die hier abgebildete ▸ Granularität der Gültigkeitszeit von einem Tag fasst mehrere identische Einkäufe während des selben Tages in einem Datensatz zusammen.

Analytische Systeme erfassen Bewegungsgrössen wie Umsatz, Kosten oder Produktionsmengen nur dann, wenn auch eine Transaktion stattgefunden hat.

Bestandesgrössen wie Lagerbestände oder Aktienkurse werden hingegen periodisch ermittelt und in Abhängigkeit des jeweiligen Erfassungszeitraums abgespeichert. Die zeitlich lückenlose Abbildung der Bestandesgrössen führt dazu, dass selbst dann neue Datensätze eingefügt werden, wenn keine Änderungen zum vorhergehenden Zustand vorliegen. Das Abbilden zeitbezogener Daten über mehrere Jahre stellt deshalb hohe Anforderungen an die Speicherkapazität analytischer Systeme.

Im dimensionalen Modell lassen sich die Fakten mit Hilfe einer Zeitdimension auch über längere Zeiträume als die in der Faktentabelle vorgegebene Periodizität aggregieren. Eine explizite Zeitdimension ist nicht notwendig, wenn der Anwender die Fakten nur über Monate oder Jahre aggregieren will. SQL-Anweisungen ermöglichen die direkte Verknüpfung dieser Zeitinformation mit dem Datumsfeld der Faktentabelle. Datenbanken können jedoch andere Perioden wie Quartale, Wochen, Arbeitstage, Ferienzeit oder Fiskaljahr nicht direkt aus einem Datum ableiten und sind für die Nutzung dieser Zeiträume auf eine entsprechende Definition angewiesen.[45]

Abbildung 2.1.2 zeigt ein Beispiel einer Zeitdimension. Es ist nicht notwendig, den Primärschlüssel der Zeitdimension und den entsprechenden Fremdschlüssel der Faktentabelle zwingend mit einem Datumsdatentyp abzubilden. Ein künstlicher Schlüssel in einem Zahlenformat verschlechtert zwar die Verständlichkeit, ist jedoch platzsparender und erlaubt auch die Zuordnung spezieller Perioden wie unbekannte oder nicht zulässige Daten.[46]

Datum	Monat	Quartal	Jahr	Ferienzeit
28.02.2002	Februar	Winter	2002	nein
01.03.2002	März	Winter	2002	nein
...	...	...	...	...
22.07.2002	Juli	Sommer	2002	ja

Abbildung 2.1.2: Beispiel einer Zeitdimension

Die Verknüpfung der Faktentabelle mit der Zeitdimension ermöglicht die zeitbezogene Analyse aller Fakten in verschiedenen vorgegebenen Verdichtungsstufen.

[45] Vgl. Kimball 1996a: S. 33
[46] Vgl. Kimball 1998: S. 192

Die folgende Beispielabfrage berechnet den gesamten Umsatz im Winter 2002:

```
SELECT SUM(Umsatz)
FROM   FAKTEN, ZEIT
WHERE  FAKTEN.Datum = ZEIT.Datum
  AND  Quartal = „Winter"
  AND  Jahr = 2002;
```

Die zeitbezogene Speicherung der Fakten bildet zwar Änderungen in der Faktentabelle adäquat ab, berücksichtigt jedoch keine Modifikationen der Dimensionsdaten. Weil *Dimensionsdatenänderungen* weniger häufig als *Faktänderungen* auftreten und bei vielen Ad-hoc-Analysen nicht die Dimensions-, sondern die Faktdaten im Zeitablauf verfolgt werden müssen, scheint der Zeitbezug der Dimensionsdaten von untergeordnetem Interesse zu sein. Zwingend historisch abzubildende Daten lassen sich mitunter als Fakt modellieren und damit in den gewünschten Zeitbezug setzen.

Die unzureichende Berücksichtigung von Dimensionsdatenänderungen kann jedoch die historische Betrachtung der Fakten beeinträchtigen. Das Hinzufügen *neuer* Werte in die Dimensionstabellen ist unproblematisch. Neue Daten haben keinen Bezug zu älteren Werten in der Faktentabelle und liefern keine falschen Analysewerte. Änderungen *bestehender* Dimensionsdaten wirken sich hingegen auch auf die zeitbezogene Abbildung vergangener Fakten aus.

Fallbeispiel

Der Hersteller des Schokoriegels Chnuschpi plant eine Verkaufsoffensive und vereinbart mit einem Detailhandelsgeschäft, dass Chnuschpi ab dem Jahr 2002 nicht mehr im Schokoladenregal, sondern ausschliesslich im Kassenbereich aufliegt. Nach dieser Neupositionierung ändert das Detailhandelsgeschäft die Produktkategorie von *Schokolade* nach *Kassenprodukte*. Abbildung 2.1.3 zeigt die Produktdimension vor der Änderung. Grundlage dieses Beispiels soll wiederum das Sternschema aus Abbildung 1.3.1 sein.[47]

[47] Vgl. Kapitel 1.3.2: S. 31

Produkt-ID	Name	Kategorie	Hersteller
1022	Chnuschpi	Schokolade	Vander
1023	N&N	Schokolade	Saturn
...	...	...	...

Abbildung 2.1.3: Produktdimension vor der Datenänderung

Abbildung 2.1.4 zeigt die von der Produktdimension abhängige Faktentabelle. Aus Übersichtlichkeitsgründen sind die Umsätze bereits über die einzelnen Jahre aggregiert dargestellt. Zudem wird auf die Darstellung der Kunden- und Geografiedimension verzichtet.

Jahr	Produkt-ID	Umsatz
1999	1021	2100
1999	1022	500
1999	1023	1500
...	...	...
2000	1022	600
...	...	...
2001	1022	700
...	...	...
2002	1022	1800

Abbildung 2.1.4: Faktentabelle

Damit die Veränderungen durch die neue Produktkategoriezuordnung einfacher sichtbar werden, sind die Umsätze mit Ausnahme des Schokoriegels *Chnuschpi* über alle Produktkategorien und Jahre konstant bei 10000 festgelegt. Das Ergebnis der Umsatzanalyse in Abhängigkeit von Jahr und Produktkategorie ist in Abbildung 2.1.5 dargestellt. Die Umsätze des Schokoriegels werden zu der Produktkategorie *Schokolade* gerechnet.

		ZEIT			
		1999	**2000**	**2001**	**TOTAL Jahr**
PRODUKT	**Backwaren**	10000	10000	10000	30000
	Schokolade	10500	10600	10700	31800
	...	...	...	...	...
	Kassenprodukte	10000	10000	10000	30000
	TOTAL Kategorie	80500	80600	80700	241800

Abbildung 2.1.5: Abfrageresultat vor der Dimensionsdatenänderung (2001)

Im Jahr 2002 wird die Produktkategorie des Schokoriegels *Chnuschpi* von Schokolade auf Kassenprodukte geändert. Abbildung 2.1.6 zeigt die Produktdimension nach der Datenänderung.

Produkt-ID	**Name**	**Kategorie**	**Hersteller**
1022	Chnuschpi	***Kassenprodukte***	Vander
1023	N&N	Schokolade	Saturn
...	...	...	...

Abbildung 2.1.6: Produktdimension nach der Datenänderung

Als Konsequenz dieser Änderung werden alle Fakten mit der Produkt-ID = 1022 der Kategorie *Kassenprodukte* zugeordnet, obwohl die älteren Fakten zum Zeitpunkt der Umsatzerzielung zur Kategorie *Schokolade* gehört haben. Das Ergebnis der Umsatzanalyse im Jahre 2002 ändert sich dementsprechend wie Abbildung 2.1.7 zeigt. Die Änderungen gegenüber der Abfrage aus dem Jahr 2001 sind kursiv dargestellt.

		ZEIT				
		1999	**2000**	**2001**	**2002**	**TOTAL Jahr**
PRODUKT	**Backwaren**	10000	10000	10000	10000	40000
	Schokolade	*10000*	*10000*	*10000*	10000	40000
	...	...	...	...	...	...
	Kassenprodukte	*10500*	*10600*	*10700*	11800	43600
	TOTAL Kategorie	80500	80600	80700	81800	243600

Abbildung 2.1.7: Abfrageresultat nach der Dimensionsdatenänderung (2002)

Änderungen an bestehenden Dimensionsdaten wie der Wohnortswechsel eines Kunden, die Neuklassifizierung von Produktkategorien oder die Bonitätsanpas-

sung eines Schuldners beeinflussen die Zuordnung der Dimensionselemente zu den Fakten *rückwirkend*. Die Anwender werden weder über die vorgenommenen Veränderungen informiert, noch können sie den alten Zustand der Daten abrufen. Die veränderte historische Sicht auf die Fakten kann deshalb zu unverständlichen Analyseergebnissen und im schlimmsten Fall zu falschen Interpretationen führen.

2.1.2 Temporale Auswertungsformen

Das zuvor dargestellte Überschreiben vergangener Dimensionsdaten mit neuen Werten führt nicht zu einer *falschen*, sondern zu einer *anderen* temporalen Sicht auf die Fakten. Jeder Zustand einer über die Zeit veränderbaren Dimensionshierarchie stellt eine von vielen möglichen Auswertungsstrukturen dar, die sich mit den Fakten verknüpfen lässt. Welche dieser Strukturen gerade die richtige ist, hängt von den Anforderungen der Benutzer und deren Fragestellungen ab.[48] Zu den wichtigsten ▸ temporalen Auswertungsformen gehören die ▸ aktuelle Sicht, die ▸ faktbezogene historische Sicht, die ▸ dimensionsbezogene historische Sicht und die ▸ ursprüngliche Sicht.[49]

Aktuelle Sicht

Wie im Beispiel zuvor illustriert wurde, wird die aktuelle Sicht durch das Überschreiben vergangener Dimensionsdaten erzielt. Die Fakten werden dem Anwender so präsentiert, als wären sie nie anderen Dimensionsdaten zugeordnet gewesen. Die aktuelle Sicht ist dann sinnvoll, wenn der Anwender die Fakten unter den gleichen, zur Zeit gültigen Rahmenbedingungen vergleichen soll. Im Fallbeispiel könnte die Frage gestellt werden, wie sich der Umsatz der Schokoladenprodukte über die Zeit verändert hat. Mit dem Wegfall des Schokoriegels Chnuschpi liegt der Umsatz im Jahr 2002 tiefer als zuvor. Die aktuelle Sicht ordnet hingegen diesen Schokoriegel nicht mehr den Schokoladenprodukten zu und zeigt damit, wie sich der Umsatz der restlichen Schokoladeprodukte über die Zeit verändert hat.

[48] Vgl. Wieken 1999: S. 146
[49] Vgl. Kimball 2000a

Faktbezogene historische Sicht

Die faktbezogene historische Sicht ermittelt für jeden Datensatz in der Faktentabelle die Gültigkeitszeit und ordnet ihm anschliessend die zu dieser Zeit gültigen Dimensionsstrukturen zu. Die Auswertungsergebnisse bleiben auch nach Dimensionsdatenänderungen nachvollziehbar, weil die Fakten unabhängig vom Auswertungszeitpunkt immer im gleichen dimensionalen Kontext abgebildet werden. Diese temporale Auswertungsform eignet sich deshalb besonders zur Analyse absoluter Zahlen. Die Frage nach dem Umsatz der Kassenprodukte im Jahr 2000 wird mit der faktbezogenen historischen Sicht mit 10000 beantwortet. Die aktuelle Sicht hingegen beantwortet die gleiche Frage mit 10600, weil der Schokoriegel mit einem Umsatz von 600 aus dieser Perspektive rückwirkend zu den Kassenprodukten zählt.

Dimensionsbezogene historische Sicht

Die dimensionsbezogene historische Sicht bestimmt die zu einem benutzerdefinierten Zeitpunkt gültigen Dimensionsdaten und ordnet diese allen Fakten zu. Im Gegensatz zur faktbezogenen historischen Sicht werden die Fakten nicht individuell, sondern mit einer einzigen Dimensionsstruktur verknüpft. Die dimensionsbezogene historische Sicht ist dann identisch mit der aktuellen Sicht, falls das aktuelle Datum als Bezugszeit angegeben wird. Dementsprechend wird diese Sicht angewandt, wenn der Anwender die Fakten unter einer identischen, jedoch in der Vergangenheit liegenden Dimensionsstruktur vergleichen will. Durch das Festlegen der Bezugszeit aller Dimensionsdaten auf das Jahr 2000 könnte eine Abfrage zum Beispiel ermitteln, wie sich der Umsatz der Schokoladenprodukte entwickelt hätte, wenn der Schokoriegel nicht an der Kasse ausgestellt worden wäre.

Ursprüngliche Sicht

Die ursprüngliche Sicht verwendet ausschliesslich den ersten Zustand der Dimensionsdaten und verknüpft diesen mit allen Fakten. Die Fakten werden so dargestellt, wie wenn sich die Dimensionsdaten im Verlaufe der Zeit nie geändert hätten. Wie bei der dimensionsbezogenen historischen Sicht und der aktuellen Sicht können die Fakten mit der ursprünglichen Sicht unter gleichbleibenden Dimensionsstrukturen verglichen werden.

Abbildung 2.1.8 zeigt, dass sich nicht alle Fragestellungen mit einer einzigen temporalen Auswertungsform beantworten lassen.

Fragestellung	aktuell	fakt-bezogen	dimensions-bezogen	ursprünglich
Wie sehen die absoluten Umsatzzahlen aus?		X		
Wie haben sich die Umsatzzahlen absolut verändert?	X		X	X
Wie haben sich die Umsatzzahlen relativ verändert?	X		(X)	X
Was wäre gewesen, wenn es keine Umstrukturierung gegeben hätte?				X
Welche Bedeutung haben die Produkte anteilig am Gesamtumsatz?		X		

Abbildung 2.1.8: Eignung temporaler Auswertungsformen[50]

Die dimensionsbezogene historische Sicht berücksichtigt als einzige temporale Auswertungsform nicht sämtliche, sondern nur jene Dimensionsdaten, die zum gewählten Bezugszeitpunkt bereits gültig waren. Die Analyse relativer Kennzahlen wie der Umsatz pro Mitarbeiter oder der Produktumsatz im Verhältnis zum Gesamtumsatz ist aufgrund der unvollständig abgebildeten Daten problematisch. Damit die relativen Veränderungen über verschiedene Zeiträume vergleichbar sind, müssen die Bezugszahlen ebenfalls mit derselben temporalen Auswertungsform und mit derselben Gültigkeitszeit bestimmt werden.

Die adäquate Verwendung der verschiedenen temporalen Auswertungsformen verbessert die Analysequalität, weil der Zeitbezug der Dimensionsdaten auf die Fragestellung abgestimmt wird. Die faktbezogene historische Sicht eignet sich besonders bei der historischen Betrachtung einzelner Werte. Vergleiche müssen hingegen häufig unter unveränderten Rahmenbedingungen stattfinden. Dem tragen die anderen temporalen Auswertungsformen Rechnung, weil sie alle Fakten in Abhängigkeit von gleichbleibenden Dimensionsstrukturen abbilden. Die aktuelle Sicht wird häufig gegenüber der ursprünglichen oder der dimensionsbezogenen Sicht bevorzugt, weil das Interesse der Benutzer eher bei der gegenwärtigen als bei einer vergangenen Dimensionsstruktur liegt.[51]

Mit dem konventionellen Ansatz, der die Dimensionsdaten im Sternschema überschreibt, werden immer nur die aktuellen Dimensionsdaten den Fakten zugeordnet. Kapitel 2.3 und folgende befassen sich mit der Modellierung aller genannten temporalen Auswertungsformen.

[50] Vgl. Wieken 1999. S. 146
[51] Vgl. Stock 2001: S. 121

2.1.3 Änderungsverfolgung

Bevor das System die Daten im gewünschten historischen Kontext abspeichern kann, muss es die Anpassungen in den Quelldaten erkennen und die benötigte Information von der Quell- in die Zieldatenbank kopieren. Im Folgenden werden Methoden zur Identifikation und Übertragung der Datenänderungen vorgestellt sowie auf die Ursachen von Anpassungen am analytischen Datenbestand erörtert. Der Einsatz der Fortschreibungsmethoden hängt von den bestehenden Datenbanksystemen und den konkreten Anforderungen an die Datenaktualität ab. Die Anpassungsursachen sowie die Anforderungen an die zeitbezogene Datenhaltung entscheiden schliesslich, wie das System die Daten im Einzelfall aktualisieren muss (Löschen, Einfügen oder Überschreiben).[52] Beispiel: Die Bereinigung eines vorhandenen Fehlers erfordert das einfache Überschreiben des alten Werts. Ein neuer Datensatz, der durch einen weiteren Einkauf erfasst wurde, darf hingegen bestehende Daten nicht modifizieren. Er wird im veränderten historischen Kontext zeitbezogen abgespeichert.

Identifikation und Übertragung von Änderungen

Änderungen werden in bestimmten Intervallen oder unmittelbar nach jeder einzelnen Anpassung von den Quell- an die Zieldatenbanken weitergegeben. Während Intervallmethoden nur die letzte Änderung innerhalb der vorgegebenen Aktualisierungsperiode erfassen, lassen sich mit den Echtzeitmethoden alle Änderungen abbilden. Im Gegenzug stellen die Echtzeitmethoden höhere Anforderungen an die Ressourcen oder die Datenbanktechnologie der Quellsysteme. Die Quell- und Zieldatenbanken werden nicht genauer spezifiziert, weil nicht nur in der operativen Datenbasis, sondern auch im Data Warehouse Fortschreibungsmethoden benötigt werden, die Änderungen erkennen und adäquat übertragen.

Intervallmethoden

- **Momentaufnahmen**

  Von den Quelldaten wird eine Momentaufnahme gemacht und diese Information in die Zieldatenbank integriert. Die Ladeoperation kann dabei die bestehenden Daten vollständig ersetzen oder mit der aktuellen Momentaufnahme überschreiben. Im Gegensatz zum Ersetzen bleibt beim Überschreiben jener Teil der alten Daten erhalten, der in der neuen Momentaufnahme

[52] Vgl. Bokun, Taglienti 1998

nicht mehr vorkommt. Historische Daten lassen sich mit dieser Methode nur dann integrieren, wenn die Quellsysteme bereits zeitbezogene Daten enthalten.

- **Vorher/Nachher-Vergleich**

  Das Ladeprogramm vergleicht die Quelldaten mit den bereits vorhandenen Daten. Die gefundenen Änderungen werden anschliessend in der Zieldatenbank abgebildet. Diese Methode ist sehr aufwändig, denn das System muss alle bereits vorhandenen Datensätze auf Veränderungen untersuchen. Dafür kann diese Lademethode auch historische Daten abbilden, weil sie bestehende Daten abhängig von der gewählten Aktualisierungsregel nicht nur überschreiben, sondern auch mit den neuen Daten ergänzen kann.

- **Zeitstempel in den Quelldaten**

  Das Volumen der zu ladenden Daten kann reduziert werden, wenn die Quelldatenbank den Zeitpunkt der Änderung in zusätzlichen Attributen festhält. Das System vergleicht die Zeitinformation aller Quelldatensätze mit dem Datum der letzten Aktualisierung und ladet nur die modifizierten Datensätze in die Zieldatenbank. Im Gegensatz zu den beiden zuvor beschriebenen Methoden wird die Aufgabe der Änderungsidentifizierung auf das Quellsystem übertragen.

Echtzeitmethoden

- **Analyse des Transaktionsprotokolls**

  Die Protokollverarbeitung ist in vielen Datenbanksystemen vorhanden und gut optimiert. Die Nutzung des Transaktionsprotokolls ist deshalb eine effiziente Methode zur Verfolgung von Änderungen und wird häufig auch bei der Datenbankreplikation eingesetzt.[53] Ein Überwachungsprogramm wertet die Einträge im Protokoll aus und gibt die erforderlichen Datenänderungen an die Zieldatenbank weiter.

[53] Vgl. Hammergren 1998: S. 121

- **Trigger**

  Trigger sind gespeicherte Programme, die dann ausgeführt werden, wenn bestimmte Ereignisse wie etwa Datenänderungen auftreten. Falls die Quelldatenbank Trigger unterstützt, kann sie Änderungen durch die vordefinierten Programme an die Zieldatenbank weitergeben. Die Leistung der Quelldatenbank wird von dieser Methode beeinträchtigt, weil sie jede triggerauslösende Änderung doppelt durchführen muss.

- **Direkte Erfassung**

  Bei der direkten Erfassung der Datenänderungen schreibt die Anwendung des Quellsystems nicht nur die Quelldaten, sondern gleichzeitig auch die Daten des Zielsystems fort. Obwohl diese Methode sehr effizient sein kann, ist sie in vielen Situationen nicht anwendbar. Zum Einen können bestehende Anwendungen aufgrund mangelnder Kenntnisse häufig nicht mehr an die neuen Anforderungen angepasst werden, zum Anderen unterscheiden sich die Änderungsmethoden in Quell- und Zieldatenbanken. Die Änderung eines einzelnen Werts im Quellsystem kann zum Beispiel aufgrund der zeitabhängigen Speicherung in der Zieldatenbank das Hinzufügen eines ganzen Datensatzes erfordern. Die unterschiedliche Änderungsbehandlung erhöht die Komplexität und beeinträchtigt dann die Datenbankleistung.

Ursprung von Änderungen

Datenänderungen werden in Data Warehouses nicht direkt von Endbenutzern, sondern von den operativen Quellsystemen oder von Anpassungsprozessen in der analytischen Umgebung initiiert. Häufige Ursachen der Anpassungen sind Fehlerbereinigungen, das Nachtragen fehlender Werte oder Veränderungen im Zeitablauf.

Fehlerbereinigungen und das Nachtragen fehlender Daten sind aufwändig, weil diese Änderungen häufig nicht mit laufzeiteffizienten Methoden wie der inkrementellen Aktualisierung erkannt werden oder das System den vorliegenden Änderungstyp nicht bestimmen kann.

- **Fehlerbereinigungen**

  Fehlerhafte Daten können vom Quellsystem übernommen oder im Transformationsprozess falsch verarbeitet worden sein und müssen deshalb unabhängig von der zeitbezogenen Datenhaltung aktualisiert werden.

- **Nachtragen fehlender Werte**

  In vielen Anwendungsfällen kann es vorkommen, dass Werte für gewisse Attribute im Data Warehouse fehlen. Falls die fehlenden Werte zu einem späteren Zeitpunkt verfügbar sind, muss das System sie im richtigen historischen Kontext nachtragen. Existieren die Datensätze mit den fehlenden Werten bereits, dann kann die Aktualisierungsmethode die bestehenden Werte einfach ergänzen. Ganze Datensätze lassen sich hingegen nur durch Einfügeoperationen nachtragen.

- **Veränderungen im Zeitablauf**

  Die häufigsten Anpassungen werden durch historisch bedingte Veränderungen verursacht. Im Laufe der Zeit verändern zum Beispiel die Lieferanten ihre Konditionen, die Kunden ihre Adressen oder die Mitarbeiter ihre Abteilung. Neben diesen extern vorgegebenen Veränderungen gibt es auch interne Anpassungsprozesse wie die Aufteilung einer Verkaufsregion in mehrere kleinere Gebiete. Solche Änderungen können durch Aktualisieren des alten Datensatzes oder, falls das System die Daten zeitbezogen abspeichern soll, mit einer anderen geeigneten Methode abgebildet werden. Die Modellierung zeitbezogener Daten wird im nächsten Kapitel eingeführt.

Zusammenfassung

Im Gegensatz zu den Fakten werden die Dimensionsdaten häufig nicht zeitbezogen abgespeichert. Das Überschreiben der nicht mehr gültigen Dimensionsdaten stellt alle Fakten immer in Beziehung zu den gegenwärtigen Dimensionsdaten. Abfragen, die sich auf vergangene Dimensionsdaten beziehen, können mit dieser aktuellen Sicht nicht erstellt werden.

Ein analytisches Informationssystem muss deshalb auch andere temporale Auswertungsformen wie die faktbezogene historische Sicht, die dimensionsbezogene historische Sicht oder die ursprüngliche Sicht unterstützen. Im Gegensatz zu den anderen temporalen Auswertungsformen ermittelt die faktbezogene historische Sicht für jeden Faktdatensatz die Gültigkeitsdauer und ordnet ihm anschliessend die zu dieser Zeit gültigen Dimensionsstrukturen zu.

Vergleichende Analysen, die alle Fakten unter den selben unveränderbaren Auswertungsstrukturen untersuchen, können mit allen anderen temporalen Auswertungsformen erstellt werden. Die aktuelle Sicht wird jedoch häufiger benötigt, weil das Interesse der Benutzer eher bei der gegenwärtigen als bei einer vergangenen Dimensionsstruktur liegt. Viele Fragestellungen können deshalb entweder mit der aktuellen oder der faktbezogenen historischen Sicht beantwortet werden.

Damit das analytische System Datenänderungen im gewünschten historischen Kontext abbilden kann, muss es zuerst Änderungen im Quellsystem identifizieren und diese anschliessend in ihre Datenbanken übertragen. Während Intervallmethoden weniger komplex sind und vom Data Warehouse direkt gesteuert werden können, müssen Echtzeitmethoden von den Quellsystemen unterstützt werden. Echtzeitmethoden berücksichtigen dafür alle Datenänderungen und übermitteln sie unmittelbar. Die analytischen Daten sind dadurch aktueller und enthalten mehr Änderungsinformationen.

Die zeitbezogene Abbildung analytischer Daten muss besonders dann berücksichtigt werden, wenn Anpassungen aufgrund von historisch bedingten Änderungen erfolgen. Falsche Werte, die zum Beispiel durch Eingabefehler entstanden sind, können hingegen durch ein einfaches Überschreiben korrigiert werden, weil sowohl vergangene als auch aktuelle Fakten mit den richtigen Dimensionsdaten zu verknüpfen sind.

2.2 Grundkonzepte der Modellierung temporaler Daten

Wie im vorhergehenden Kapitel gezeigt wurde, können die Anwender alle Fakten mit Hilfe eines Zeitstempelattributs oder einer expliziten Zeitdimension historisch analysieren. Die restlichen Dimensionen enthalten jedoch keinen Zeitbezug. Das Grundmodell des Sternschemas unterstützt deshalb jeweils nur eine von mehreren temporalen Auswertungsformen.

Dieses Kapitel führt die grundlegenden Konzepte der temporalen Datenhaltung ein und betrachtet die notwendigen Datenmodellanpassungen in relationalen Datenbanksystemen. Das nächste Kapitel zeigt anschliessend, wie sich diese Konzepte auf das dimensionale Modell übertragen lassen und erörtert deren Nutzen in OLAP-Systemen.

2.2.1 Temporale Datenhaltung

Zeitmodelle

Im *kontinuierlichen* Zeitmodell existiert zwischen zwei Punkten auf der Zeitachse immer ein weiterer Zeitpunkt. Dieses Modell kann nicht verlustfrei auf Computer übertragen werden, weil sich zu einem Zeitpunkt weder ein Vorgänger noch ein Nachfolger eindeutig bestimmen lässt.[54] Die Zeit wird deshalb als Folge diskreter Zeiteinheiten abgebildet, die isomorph zu den natürlichen Zahlen einschliesslich Null oder einer Untermenge von ihnen ist.[55]

Im *diskreten* Zeitmodell kann die Zeitachse als Menge von nicht mehr teilbaren Zeitpunkten angesehen werden. Die kleinste in einem bestimmten Zusammenhang relevante Zeiteinheit wird als ▸ *Chronon* bezeichnet.[56] Die Granularität der Zeitmessung, das heisst die Grösse des Chronons, ist vom Anwendungsbereich abhängig.[57] Während eine höhere Granularität durch Aggregation erreicht wird, lassen sich Daten meist nur mit einem Genauigkeitsverlust disaggregieren.

Viele Objekte weisen eine längere Zeitbindung auf als ein Chronon. Ununterbrochene Zeiträume können durch Intervalle beschrieben werden, die durch

---

[54] Vgl. Steiner 1997: S.16
[55] Vgl. Knolmayer, Myrach 1996: S. 64
[56] Vgl. Jensen, Dyreson 1998: S. 376
[57] Vgl. Stock 2001: S. 32

einen Anfangs- und einen Endzeitpunkt oder durch eine bestimmte Dauer wie etwa eine Woche definiert sind.[58]

Klassifikation der Zeit

In Datenbanksystemen wird häufig zwischen benutzerdefinierter Zeit, Gültigkeitszeit und Transaktionszeit unterschieden.

- **Benutzerdefinierte Zeit**

  Die benutzerdefinierte Zeit wie etwa das Attribut *Geburtsdatum* hat zwar zeitliche Eigenschaften, spielt aber bei der zeitbezogenen Datenhaltung keine Rolle. Das Datenbanksystem behandelt sie wie gewöhnliche Attribute. Unter Berücksichtigung der Domänenintegrität kann die benutzerdefinierte Zeit jeden Zeitpunkt in der Vergangenheit, Gegenwart oder Zukunft einnehmen.

- **Gültigkeitszeit**

  Die Gültigkeitszeit einer Objektversion gibt den Zeitraum an, in welchem das Objekt in der Realwelt den beschriebenen Zustand aufweist.[59] Ändert sich der Zustand eines Objekts, dann werden die neuen Werte zusätzlich zu den bereits vorhandenen Daten in Abhängigkeit von der veränderten Gültigkeitszeit gespeichert. Muss jedoch ein bereits abgespeicherter Zustand aufgrund eines Fehlers berichtigt werden, dann ersetzen die korrekten Werte die vorhandenen Daten ohne neuen Gültigkeitsbezug. Die Gültigkeitszeit selbst kann im Nachhinein ebenfalls verändert werden und sowohl vergangene als auch zukünftige Zeiträume beschreiben.

- **Transaktionszeit**

  Die Transaktionszeit einer Objektversion ist der Zeitpunkt, an dem die Änderungen im Datenbanksystem dokumentiert werden.[60] Fehlerhafte Werte bleiben bei der Korrektur mit der entsprechenden Transaktionszeit in der Datenbank eingetragen. Dadurch können die Anwender auf alle Daten zugreifen, die irgendwann einmal in die Datenbank eingetragen wurden. Die Transaktionszeit wird von der Systemzeit übernommen und liegt des-

[58] Vgl. Jensen et al. 1998: S. 377
[59] Vgl. Jensen et al. 1998: S. 370
[60] Vgl. Jensen et al. 1998: S. 371

halb weder vor der Datenbankerstellung noch in der Zukunft und ist nachträglich nicht veränderbar.

Klassifikation temporaler Datenbanken

Von den drei vorgestellten Zeitlinien können die Gültigkeitszeit und die Transaktionszeit die Daten in einen temporalen Bezug setzen. Abhängig vom Einsatz der beiden Zeitlinien lassen sich vier Datenbanktypen unterscheiden (vgl. Abbildung 2.2.1).[61]

	keine Gültigkeitszeit	**Gültigkeitszeit**
keine Transaktionszeit	Schnappschuss-Datenbank	historische Datenbank
Transaktionszeit	Rollback-Datenbank	bitemporale Datenbank

Abbildung 2.2.1: Klassifikation der Datenbanken

- **Schnappschuss-Datenbanken**

  Schnappschuss-Datenbanken berücksichtigen weder Gültigkeits- noch Transaktionszeit und können deshalb nur *einen* zeitlichen Zustand abbilden (zum Beispiel den aktuellen). Der alte Zustand eines Werts wird bei dessen Änderung gelöscht. Der Anwender hat anschliessend keine Möglichkeit mehr, auf den vorhergehenden Zustand zuzugreifen.

- **Historische Datenbanken**

  Historische Datenbanken berücksichtigen nur die Gültigkeitszeit. Die Anwender können die historische Entwicklung der Daten ausschliesslich aus der heutigen Sicht verfolgen, weil nicht mehr zutreffende Zustände der Gültigkeitszeit beim Aktualisieren verloren gehen. Die Bezeichnung *historisch* wird in diesem Zusammenhang nicht nur für in der Vergangenheit, sondern auch in der Zukunft gültige Daten verwendet.[62]

- **Rollback-Datenbanken**

  Rollback-Datenbanken berücksichtigen nur die Transaktionszeit. Die Daten werden so abgebildet, dass sich jeder vergangene Zustand der Datenbank

[61] Vgl. Stock 2001: S. 38
[62] Vgl. Jensen et al. 1998: S. 372

später rekonstruieren lässt. Rollback-Datenbanken werden auch als reine Einfüge-Datenbanken angesehen, weil sie abgesehen von der Transaktionszeit keinen bestehenden Wert überschreiben oder löschen.

- **Bitemporale Datenbanken**

  Bitemporale Datenbanken sind eine Kombination aus historischer Datenbank und Rollback-Datenbank, denn sie berücksichtigen sowohl die Gültigkeits- als auch die Transaktionszeit. Erst mit bitemporalen Datenbanken ist bestimmbar, welche Datenwerte zu einem bestimmten Zeitpunkt als gültig erachtet wurden.[63]

2.2.2 Temporale Klassifikation der Attribute

Die Modellierung zeitbezogener Daten setzt eine differenzierte Betrachtung der temporalen Attributeigenschaften voraus, denn gewisse Attribute ändern sich im Verlauf der Zeit gar nicht oder es ist nicht erforderlich, den vorhandenen Zeitbezug abzubilden.

Die Unterscheidung zwischen zeitabhängigen und zeitunabhängigen Attributen alleine ist häufig unzureichend. Ein als zeitunabhängig deklariertes Attribut kann zum Beispiel nicht veränderbar sein oder muss aus der Sicht der Anwender nicht zeitbezogen abgespeichert werden. Eine genauere temporale Differenzierung unterstützt das Verständnis und damit die Modellierung zeitbezogener Daten.

Kaiser unterscheidet Zeitstempelattribute, zeitunabhängige Attribute, zeitabhängige Attribute im weiteren Sinn, zeitabhängige Attribute im engeren Sinn und zeitabhängige zyklische Attribute:[64]

- **Zeitstempelattribute**

  Zeitstempelattribute sind auf die Domäne *Zeit* definiert und bestimmen den Zeitpunkt oder das Zeitintervall der Gültigkeits- oder Transaktionszeit aller zugeordneten Datenobjekte. Die Anwendung von Zeitstempelattributen wird im nächsten Kapitel eingeführt.[65]

---

[63] Vgl. Knolmayer, Myrach 1996: S. 65
[64] Vgl. Kaiser 2000: S. 108
[65] Vgl. Kapitel 2.2.3: S. 64

- **Zeitunabhängige Attribute**

  Die Werte zeitunabhängiger Attribute bleiben für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps konstant. Typische Beispiele zeitunabhängiger Attribute sind Geburtsdatum und Geburtsort einer Person oder abgeleitete Kategorien der Zeitdimension wie etwa Monat oder Jahr.

- **Zeitabhängige Attribute im weiteren Sinn**

  Die Werte zeitabhängiger Attribute im weiteren Sinn können sich für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps ändern. Die zeitbezogene Betrachtung dieser Attribute ist jedoch in der modellierten Realität *nicht erforderlich*. Sie enthalten deshalb ausschliesslich aktuelle Werte. Allgemeingeltende Beispiele dieser Attributart können nicht gegeben werden, weil die Klassenzuteilung abhängig von den Anforderungen der Benutzer ist. Der Name eines Kunden oder die Adresse eines Mitarbeiters gehört zum Beispiel zu den zeitabhängigen Attributen im weiteren Sinn, wenn für den Anwender ausschliesslich aktuelle Werte dieser Attribute relevant sind.

- **Zeitabhängige Attribute im engeren Sinn**

  Die Werte zeitabhängiger Attribute im engeren Sinn können sich für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps ändern. Die zeitbezogene Betrachtung *ist erforderlich*, weil die Wertevolution dieser Attribute nachvollziehbar sein muss. Die Datenbank enthält von dieser Attributart deshalb nicht nur die aktuelle, sondern auch vergangene und gegebenenfalls zukünftige Werte. Auch hier ist die Klassenzuteilung von den Anforderungen der Benutzer abhängig. Soll die berufliche Entwicklung der Mitarbeiter nachvollziehbar sein, dann wird zum Beispiel die Funktion der Mitarbeiter zeitbezogen abgebildet.

- **Zeitabhängige zyklische Attribute**

  Die Werte zeitabhängiger zyklischer Attribute können sich für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps ändern. Im Gegensatz zu den zeitabhängigen Attributen im engeren Sinn müssen nicht nur Änderungen, sondern auch die Tatsache, dass sich ein Wert in einem vorgegebenen Intervall nicht verändert hat, abgebildet werden. Zu dieser Attributart gehören zum Beispiel die Fakten in einem dimensionalen Datenmodell. Sie werden periodisch und unabhängig von vorhergehenden Werten ergänzt.

2.2.3 Zeitbezogene Daten in relationalen Datenmodellen

In der Literatur sind verschiedene Modelle und Datenbanksprachen zur zeitbezogenen Erweiterung des relationalen Datenmodells vorgestellt worden.[66] Dieses Kapitel zeigt anhand eines Beispiels die grundlegenden Erweiterungen zur Darstellung zeitbezogener Daten im relationalen Datenmodell. Andere logische Datenmodelle wie das objektorientierte, das hierarchische oder das Netzwerk-Datenmodell werden im Hinblick auf die zeitbezogene Modellierung von OLAP-Datenbanken nicht berücksichtigt, denn diese Systeme basieren auf der relationalen oder dann auf proprietären multidimensionalen Technologien.

Zeitstempel

Wie in den vorhergehenden Abschnitten erläutert wurde, muss bei der zeitbezogenen Modellierung einerseits zwischen den beiden Zeitlinien Gültigkeitszeit und Transaktionszeit, andererseits zwischen verschiedenen Klassen zeitbezogener Attribute unterschieden werden. Zudem lassen sich bestimmte Intervalle nicht nur mit zwei zeitgrenzenmarkierenden Attributen, sondern auch mit einem einzigen Attribut darstellen.

Das relationale Datenmodell bietet in seiner Grundform keine adäquate Unterstützung für die Realisierung zeitbezogener Informationssysteme.[67] Deshalb werden benutzerdefinierte Zeitattribute, sogenannte Zeitstempel eingeführt, durch die sich die Gültigkeits- und Transaktionszeit eines Datenobjekts abbilden lässt.

Intervallabbildung

1. Zeitstempelung mit einem Attribut

Ein Zeitstempelattribut bildet die Zeitbindung eines Objekts ab, falls die Bindungsdauer ein Chronon aufweist.[68] Im folgenden Beispiel wird eine Tabelle mit einem Zeitstempelattribut (Zeitpunkt) erweitert (vgl. Abbildung 2.2.2). Die Grösse des Chronons ist hier gleich einem Tag.

---

[66] Vgl. zum Beispiel Stock 2001: S. 55
[67] Vgl. Knolmayer, Myrach: S. 68
[68] Vgl. Knolmayer, Myrach 1996: S. 64

Kunden-ID	Name	Ort	Geburtsdatum	Zeitpunkt
1000	Meier	Basel	01.10.1954	01.01.2001
1001	Huber	Glarus	24.05.1968	01.01.2001
...	...	...	...	...
1000	Meier	Basel	01.10.1954	02.01.2001
...	...	...	...	...
1000	Meier	Basel	01.10.1954	27.11.2001
...	...	...	...	...
1000	Meier	Zürich	01.10.1954	28.11.2001

Abbildung 2.2.2: Zeitpunkt-Zeitstempelung

Durch die Primärschlüsselerweiterung Zeitpunkt lassen sich zu einem Kunden nicht nur eine, sondern mehrere Versionen abbilden (im dargestellten Beispiel wird jeden Tag eine neue Version gespeichert). Die zu verwaltende Versionsmenge ist von der Grösse des Chronons abhängig.

Viele Objekte weisen eine längere Bindungsdauer als ein Chronon auf. Abbildung 2.2.2 zeigt zum Beispiel, dass die Daten des Kunden Meier vom 1.1.2001 bis zum 27.11.2001 unverändert bleiben. Zur Darstellung ununterbrochener Zeiträume ist eine Intervall-Zeitstempelung vorteilhafter, weil sie unveränderte Objektversionen in einem ▸Tupel zusammenfassen kann. Intervalle können mit einem einzelnen Zeitstempelattribut abgebildet werden, wenn sie sich nicht überschneiden oder überlappen und auch keine Lücken dazwischen liegen (das auf den Endzeitpunkt eines Intervalls folgende Chronon ist gleichzeitig Startzeitpunkt des nächsten Intervalls).[69] Abbildung 2.2.3 zeigt diese Intervalldarstellung: Kunde Meier hat vom 1.1.2001 bis 27.11.2001 in Basel gewohnt.

Kunden-ID	Name	Ort	Geburtsdatum	Intervallstart
1000	Meier	Basel	01.10.1954	01.01.2001
1001	Huber	Glarus	24.05.1968	01.01.2001
...	...	...	...	...
1000	Meier	Zürich	01.10.1954	28.11.2001
...	...	...	...	...

Abbildung 2.2.3: Intervall-Zeitstempelung mit einem Zeitstempelattribut

[69] Vgl. Allen 1983: S. 836

2. Zeitstempelung mit zwei zeitgrenzenmarkierenden Attributen

Die Abbildung eines Intervalls in einem Tupel mit zwei zeitgrenzenmarkierenden Zeitstempeln lässt auch überschneidende oder nicht aufeinanderfolgende Intervallbeziehungen zu. Die Information über Beginn und Ende eines Intervalls ist zudem für jedes Tupel direkt ersichtlich und muss nicht über zwei Tupel ermittelt werden, wie dies bei einem Zeitstempelattribut der Fall ist. Abbildung 2.2.4 zeigt die mit zwei Zeitstempeln erweiterte Kundentabelle.

Kunden-ID	Name	Ort	Geburtsdatum	Intervallstart	Intervallende
1000	Meier	Basel	01.10.1954	01.01.2001	27.11.2001
1001	Huber	Glarus	24.05.1968	01.01.2001	31.07.2001
...	...	...	...	...	...
1001	Huber	Luzern	24.05.1968	01.07.2001	∞
...	...	...	...	...	...
1000	Meier	Zürich	01.10.1954	28.11.2001	∞
...	...	...	...	...	...

Abbildung 2.2.4: Intervall-Zeitstempelung mit zwei Zeitstempelattributen

Das Intervallende der letzten Objektversion bleibt häufig unbestimmt (∞), weil ihre Änderungszeitpunkte meist nicht im Voraus bekannt sind. Im dargestellten Beispiel hat Kunde Huber während eines Monats sowohl in Glarus als auch in Luzern gewohnt. Diese Art der Intervallbeziehung (Überschneidung) wäre mit einem einzelnen Zeitstempelattribut nicht darstellbar.

Zeitstempelmethoden

1. Tupel-Zeitstempelung

Die Tupel-Zeitstempelung erweitert alle Tupel in einer Tabelle mit Zeitstempelattributen. Somit gehören alle zuvor illustrierten Beispiele zu dieser Zeitstempel-Kategorie (vgl. Abbildungen 2.2.2 bis 2.2.4). Änderungen werden nicht durch das Überschreiben bestehender, sondern durch das Einfügen neuer Tupel mit entsprechend aktualisierten Zeitstempeln abgebildet. Der Anwender kann somit jederzeit auch auf vergangene Zustände zurückgreifen. Da jedes Tupel jeweils nur eine Objektversion enthält, ist die vollständige Geschichte eines Objekts auf mehrere Tupel verteilt (vertikale temporale Anomalie).[70] Wenn sich nicht alle Attributwerte gleichzeitig ändern, führt das Einfügen neu-

[70] Vgl. Steiner 1997: S. 23

er Tupel zu Redundanz (horizontale temporale Anomalie). Zum Beispiel ändert sich das Attribut Geburtsdatum in der Kundentabelle nie. Es wird unnötigerweise redundant gespeichert, weil bei der Modellierung der dargestellten Tabelle nicht zwischen verschiedenen Klassen zeitabhängiger Attribute differenziert wurde.

2. Attribut-Zeitstempelung

Die Attribut-Zeitstempelung unterscheidet zwischen zeitabhängig und zeitunabhängig zu speichernden Attributen. Sie beseitigt sowohl die vertikale als auch die horizontale temporale Anomalie, indem die gesamte Geschichte in einem Attribut abgebildet wird. Abbildung 2.2.5 zeigt die Attribut-Zeitstempelung.

Kunden-ID	Name	Ort	Geburtsdatum
1000	Meier	{(01.01.2001, 27.11.2001, Basel); (28.11.2001, ∞, Zürich)}	01.10.1954
1001	Huber	{(01.01.2001, 31.07.2001, Glarus); (01.07.2001, ∞, Luzern)}	24.05.1968
...	...	...	...

Abbildung 2.2.5: Attribut-Zeitstempelung

Die Tabellen liegen bei der Attribut-Zeitstempelung nicht in der ersten Normalform[71] vor und können deshalb nicht direkt in kommerziell verfügbaren relationalen Datenbanksystemen abgebildet werden.[72] Die zusammengesetzten und ▸Wiederholungsgruppen enthaltenden Attribute lassen sich zwar durch temporale Normalisierungskonzepte[73] auflösen, das Ergebnis entspricht jedoch nicht mehr der Attribut-Zeitstempelung. Einerseits sind die Versionen der zeitbezogenen Attribute wiederum auf mehrere Tupel verteilt (vertikale temporale Anomalie), andererseits erfordert die Normalisierung nicht für jedes Attribut eine eigene Tabelle. Zeitabhängige Attribute, deren Ausprägungen sich immer gleichzeitig ändern, können gemeinsam in einer Tabelle abgebildet werden. Dieser Modellierungsprozess führt zu einer nach temporalen Eigenschaften differenzierenden Tupel-Zeitstempelung. Abbildung 2.2.6 zeigt das aus der Kundentabelle hervorgegangene normalisierte Modell.

[71] Die Werte zeitabhängiger Attribute sind nicht atomar.
[72] Vgl. Stock 2001: S. 45
[73] Vgl. Navathe, Ahmed 1993

KUNDEN

Kunden-ID	Name	Geburtsdatum
1000	Meier	01.10.1954
1001	Huber	24.05.1968
...	...	...

KUNDENORT

Kunden-ID	Ort	Intervallstart	Intervallende
1000	Basel	01.01.2001	27.11.2001
1001	Glarus	01.01.2001	31.07.2001
1000	Zürich	28.11.2001	∞
1001	Luzern	01.07.2001	∞
...	...	...	...

Abbildung 2.2.6: Normalisierte Kundentabelle

Da der Wohnort an sich nicht veränderbar ist, sondern nur die Beziehung zwischen den Kunden und den Orten, kann hier auf eine Tabelle ORTE verzichtet werden.

Berücksichtigung der Zeitlinien

Die bisherigen Ausführungen zu der Modellierung zeitbezogener Daten haben nur eine Zeitlinie berücksichtigt. Die Unterscheidung zwischen *Gültigkeits-* und *Transaktionszeit* erfordert die Verwendung zusätzlicher Zeitstempel. Im Gegensatz zur Gültigkeitszeit sind bei der Transaktionszeit alle Attribute unabhängig von ihrem Zeitbezug als veränderbar zu betrachten. Auch fehlerhafte Werte zeitunabhängiger Attribute müssen sich korrigieren lassen und sind dementsprechend ebenfalls mit der Transaktionszeit zu versehen. Abbildung 2.2.7 zeigt die Kundentabelle, in der die Gültigkeitszeit *und* die Transaktionszeit für alle Tupel mit jeweils zwei zeitgrenzenmarkierenden Zeitstempeln abgebildet sind.

Kunden-ID	Name	Ort	Geburtsdatum	GZ-Start	GZ-Ende	TZ-Start	TZ-Ende
1000	Meier	Basel	01.10.1954	01.01.2001	∞	01.01.2001	15.12.2001
1000	Meier	Basel	01.10.1954	01.01.2001	27.11.2001	15.12.2001	∞
1000	Meier	Zürich	01.10.1954	28.11.2001	∞	15.12.2001	∞
...	...	...	...	...	...	...	...
1001	Huber	Genf	24.05.1968	01.01.2001	∞	01.01.2001	06.01.2001
1001	Huber	Glarus	24.05.1968	01.01.2001	∞	06.01.2001	15.06.2001
1001	Huber	Glarus	24.05.1968	01.01.2001	31.07.2001	15.06.2001	∞
1001	Huber	Luzern	24.05.1968	01.07.2001	∞	15.06.2001	∞
...	...	...	...	...	...	...	...

Abbildung 2.2.7: Kundentabelle mit Gültigkeits- und Transaktionszeit

Die Transaktionszeit (TZ) wird wie die Gültigkeitszeit (GZ) mit zwei Zeitstempelattributen dargestellt. TZ-Start bestimmt dabei, wann das Tupel in die Tabelle eingefügt wird und TZ-Ende wird bei der Korrektur fehlerhafter Werte gesetzt. Das Beispiel in Abbildung 2.2.7 zeigt zwei unterschiedliche Änderungen. Beide Kunden wechseln ihren Wohnort. Bei Kunde Huber wird jedoch am 6.1.2001 zusätzlich der Wohnort korrigiert.

Entitätsintegrität und Datenzugriff[74]

Der Primärschlüssel muss bei der Modellierung zeitbezogener Daten zur Sicherung der ▸ *Entitätsintegrität* erweitert werden. Der neue Primärschlüssel setzt sich aus dem bisherigen Primärschlüssel sowie den Zeitstempelattributen zusammen. Er gewährleistet, dass kein Tupel eingefügt werden kann, dessen Schlüsselwerte mit jenen eines schon existierenden Tupels übereinstimmen. Integritätsregeln wie keine Zulassung von überlappenden oder überdeckenden Intervallen können jedoch nicht vom erweiterten Primärschlüssel sichergestellt werden. In diesem Fall müssen die Integritätsbedingungen von Anwendungsprogrammen oder Triggern abgebildet werden.

Im Gegensatz zu herkömmlichen Tabellen erfolgt der *Datenzugriff* bei temporal erweiterten Tabellen nicht durch die einfache Übereinstimmung mit einem Schlüsselwert. Anfragen wie *„wo hat Kunde Huber am 1.9.2001 gewohnt“* erfordern vielmehr eine intervallbezogene Überprüfung des gesuchten Datums.[75]

[74] Vgl. Knolmayer, Myrach 1996: S. 69
[75] Ausnahme: Zeitpunkt-Zeitstempelung, vgl. Abbildung 2.2.2: S. 64

Zusammenfassung

Die Zeit lässt sich als Menge von nicht mehr teilbaren Zeitpunkten in Datenbanksysteme integrieren. Die kleinste relevante Zeiteinheit wird als Chronon bezeichnet. Objekte, die eine längere Zeitbindung als ein Chronon aufweisen, können durch Intervalle beschrieben werden.

In temporalen Datenbanken wird zwischen der Gültigkeitszeit und der Transaktionszeit unterschieden. Die Gültigkeitszeit ist benutzerdefiniert und gibt den Zeitraum an, in welchem das Objekt in der Realwelt den beschriebenen Zustand aufweist. Die Transaktionszeit wird vom System festgelegt und gibt an, wann ein Datensatz angelegt und wann er verändert wurde. Erst durch die Berücksichtigung beider Zeitlinien ist bestimmbar, welche Datenwerte zu einem bestimmten Zeitpunkt als gültig erachtet wurden.

Nicht alle Attribute sind veränderbar oder müssen als veränderbar betrachtet werden. Die Unterscheidung in zeitabhängige und zeitunabhängige Attribute trägt wesentlich zum Verständnis des Datenmodells und zur Reduktion des Verwaltungs- und Ressourcenaufwands bei.

Zeitpunkte und Intervalle können mit Zeitstempelattributen abgebildet werden. Bei einer Intervall-Zeitstempelung ist zu berücksichtigen, dass der Datenzugriff nicht durch das blosse Übereinstimmen der Primärschlüsselattribute erfolgen kann, sondern ein intervallbezogener Vergleich des gesuchten Datums notwendig ist.

Während die Implementierung der Attribut-Zeitstempelung im relationalen Modell problematisch ist, kann die Tupel-Zeitstempelung einfach umgesetzt werden. Jede Änderung wird durch ein neues Tupel abgebildet. Die daraus entstehende Redundanzbildung kann durch temporale Normalisierungskonzepte vermieden werden. Eine auf diese Art modellierte Tabelle enthält, abgesehen vom Primärschlüssel, ausschliesslich Attribute, die sich gleichzeitig ändern.

2.3 Zeitbezogene Daten im dimensionalen Modell

Das folgende Kapitel beschäftigt sich mit der Abbildung der *temporalen Auswertungsformen* im dimensionalen Modell.[76] Es untersucht zunächst, welche der im vorhergehenden Kapitel dargestellten Methoden sich zur Modellierung zeitbezogener Fakten und Dimensionen eignen. Das daraus abgeleitete Modell wird anschliessend mit bekannten Modellierungsansätzen verglichen. Zu den zentralen Beurteilungskriterien gehört neben der Möglichkeit zur flexiblen Abbildung der temporalen Auswertungsformen auch die Implementier- und Wartbarkeit sowie die Benutzerfreundlichkeit in typischen kommerziellen OLAP-Systemen.

2.3.1 Temporale Erweiterung der Tabellen

Ein Sternschema besteht aus einer Faktentabelle und mehreren Dimensionstabellen. Grundvoraussetzung temporaler Analysen ist die zeitbezogene Abbildung der Fakten. Deshalb wird im folgenden davon ausgegangen, dass alle Fakten *immer* über ein Zeitstempelattribut und/oder eine Zeitdimension in ihrem historischen Kontext abgespeichert sind. Abbildung 2.3.1 zeigt ein einfaches Sternschema, bei dem sich die Fakten mit dem Zeitstempelattribut Datum und der Zeitdimension ZEIT über verschiedene temporale Verdichtungsstufen analysieren lassen.[77]

Die Nichtschlüssel- und Zeitstempelattribute sind im Sternschemabeispiel bereits gemäss ihren temporalen Eigenschaften gekennzeichnet.[78] Die Zuordnung der Attributarten erfolgt nach den Anforderungen der Anwender. Die hier vorgenommene Klassifikation soll lediglich als Beispiel dienen:

- In der *Faktentabelle* sind neben dem Zeitstempelattribut Datum die Fakten Umsatz, Kosten und Gewinn temporal klassifiziert. Die Fakten gehören zu den zeitabhängigen zyklischen Attributen, weil das System ihren Zustand periodisch von neuem abbilden muss.
- Die *Zeitdimension* enthält ausschliesslich zeitunabhängige Attribute, weil ihr Primärschlüssel bereits zur Domäne Zeit gehört. Zu einem bestimmten Datum können zum Beispiel nicht mehrere Versionen eines Jahrs existieren.

---

[76] Vgl. Kapitel 2.1.2: S. 50
[77] Vgl. auch Abbildung 1.3.1: S. 31
[78] Die Attribute sind hinsichtlich der von Kaiser aufgestellten Kategorien temporaler Attributarten klassifiziert. Vgl. Kapitel 2.2.2: S. 62

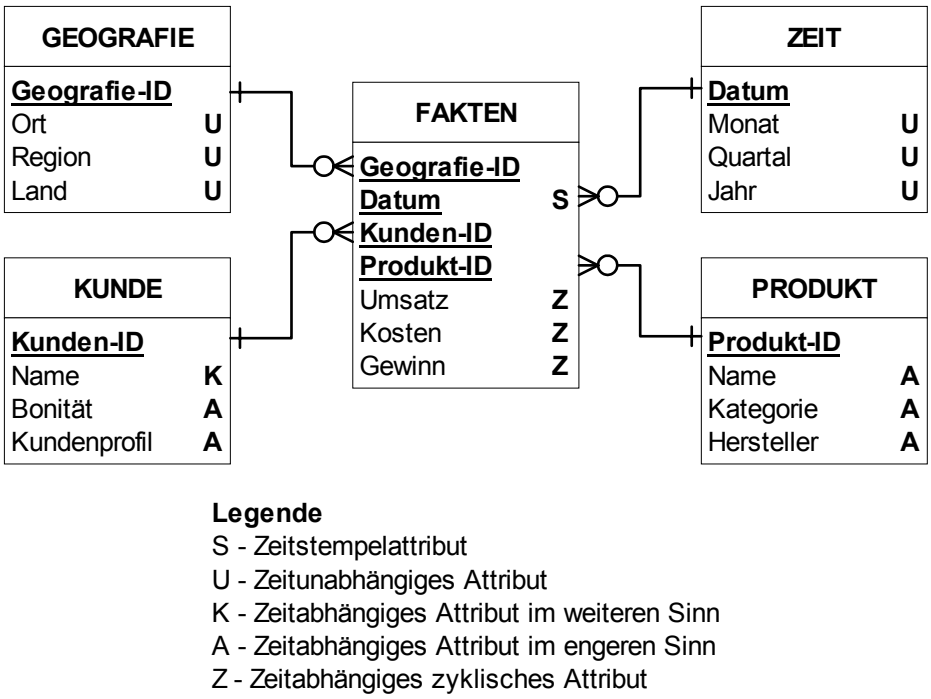


Abbildung 2.3.1: Sternschema mit temporal klassifizierten Attributen

- Die Attribute der *Geografiedimension* sind hier ebenfalls als zeitunabhängig eingestuft. Politisch bedingte geografische Veränderungen sind zwar möglich, treten aber äusserst selten auf. Demgegenüber müssen von der Unternehmung beeinflussbare geografische Attribute wie etwa die Verkaufsregion häufig zeitbezogen betrachtet werden. Solche Attribute sind in der abgebildeten Geografiedimension jedoch nicht vorhanden.
- In der *Kundendimension* sind die Attribute Bonität und Kundenprofil als zeitabhängig im engeren Sinn, das Attribut Name als zeitabhängig im weiteren Sinn gekennzeichnet. Diese Unterscheidung bedeutet, dass sich alle Attributwerte im Laufe der Zeit verändern können, aber nur Anpassungen der Bonitätsklasse und des Kundenprofils zeitbezogen abzubilden sind. In diesem Fall wird davon ausgegangen, dass nur ein einziger Zustand des Kundennamens (zum Beispiel der aktuelle) für das Unternehmen relevant ist. Einerseits ist eine Namensänderung selten, andererseits wird ein Verkauf nicht durch den Namen, sondern durch betriebswirtschaftliche Kundenmerkmale wie die Bonität oder spezielle Interessen (Kundenprofil) beeinflusst.
- Die Attribute Name, Kategorie und Hersteller der *Produktdimension* sind als zeitabhängig im engeren Sinn klassifiziert. Im Gegensatz zum Kundennamen wird beim Produktnamen angenommen, dass sich eine Änderung auf den Verkauf, beziehungsweise auf die Fakten auswirken kann und deshalb zeitbezogen abzubilden ist.

Erweiterung der Faktentabelle

Die Faktentabelle enthält bereits ein Zeitstempelattribut (Datum), das den Gültigkeitszeitpunkt aller Fakten (Umsatz, Kosten, Gewinn) bestimmt. Es muss überprüft werden, inwiefern eine Erweiterung der Tabelle um die *Transaktionszeit* oder die Verwendung der Intervall-Zeitstempelung anstelle der *Zeitpunkt-Zeitstempelung* die temporale Abbildung der Fakten verbessern kann.

Transaktionszeit

Während Änderungen im Zeitablauf mit der Gültigkeitszeit abgebildet werden, lassen sich Korrekturen oder nachträglich eingefügte Daten mit der Transaktionszeit identifizieren. Die Berücksichtigung der Transaktionszeit ermöglicht den Zugriff auch auf solche Fakten, die aus der heutigen Sicht nicht mehr gültig sind. In der Vergangenheit erstellte Analyseergebnisse bleiben dadurch auch zu einem späteren Zeitpunkt noch nachvollziehbar.

Abbildung 2.3.2 zeigt eine mit der Transaktionszeit erweiterte Faktentabelle. Die Attribute Geografie-ID, Kunden-ID und Produkt-ID sind aus Platzgründen abgekürzt dargestellt (G-ID, K-ID und P-ID). Das ursprüngliche Attribut der Gültigkeitszeit Datum wurde in G-Datum umbenannt und die Transaktionszeit T-Datum eingefügt.

G-ID	K-ID	P-ID	G-Datum	T-Datum	Umsatz	Kosten	Gewinn	Zeile
10	11	100	1.1.2001	16.1.2001	100	80	20	1
10	27	152	1.1.2001	16.1.2001	80	25	55	2
...	...	...	...	...	...	...	...	3
10	11	100	1.1.2001	25.2.2001	50	40	10	4
...	...	...	...	...	...	...	...	5
50	22	133	21.10.2000	28.2.2001	60	50	10	6
...	...	...	...	...	...	...	...	7

Abbildung 2.3.2: Um die Transaktionszeit erweiterte Faktentabelle

Unterstützung bei der Änderungsabbildung

Die Zeile 1 in Abbildung 2.3.2 wäre ohne die Transaktionszeit durch Zeile 4 ersetzt worden. Zeile 4 zeigt eine Korrektur, die zum Beispiel durch die Rückgabe eines fehlerhaften Artikels notwendig wurde. Das Tupel in Zeile 4 enthält gegenüber jenem in Zeile 1 die veränderten Fakten und ein neues Transaktionsdatum. Das Gültigkeitsdatum bleibt dagegen unverändert. Der Zustand der Fakten kann sowohl vor als auch nach der Korrektur analysiert werden, weil

das ursprüngliche Tupel in Zeile 1 durch die Erweiterung des Primärschlüssels mit der Transaktionszeit erhalten bleibt.

Unterstützung bei der Änderungsidentifikation

Änderungen können durch den Einsatz der Transaktionszeit im Quellsystem besser erkannt und ins Zielsystem übertragen werden, weil diese im Gegensatz zur Gültigkeitszeit Modifikationen vollständig abbildet und dabei immer Bezug auf die aktuelle Systemzeit nimmt. Beispiel: OLAP-Datenbanken laden ihre Daten in bestimmten Intervallen aus dem zentralen Data Warehouse oder einem Data Mart. Zur Reduzierung des Ladevolumens nutzen sie häufig inkrementelle Methoden, die den Zeitpunkt der letzten Aktualisierung mit der Gültigkeitszeit der vorhandenen Daten vergleichen.[79] Das System lädt dadurch nur Daten, die noch nicht vorhanden sind. Nachträgliche Korrekturen oder neu hinzugekommene Daten, deren Gültigkeitszeit bereits weit in der Vergangenheit liegt, lassen sich jedoch mit dieser Lademethode nicht berücksichtigen. Das Tupel in Zeile 6 wurde zum Beispiel erst etwa vier Monate nachdem es gültig war in die Datenbank gespeichert. Aufgrund des alten Gültigkeitsdatums geht der Ladeprozess davon aus, dass das Tupel bereits vier Monate zuvor geladen wurde und infolgedessen bereits der OLAP-Anwendung zur Verfügung steht.

Wenn die Faktentabelle in der Quelldatenbank hingegen auch die Transaktionszeit enthält, können die Ladeprozesse anstelle der Gültigkeitszeit die Transaktionszeit mit dem Zeitpunkt der letzten Aktualisierung vergleichen und damit *alle* Veränderungen und Ergänzungen in den OLAP-Datenbestand integrieren.

Beurteilung

Die Transaktionszeit vergrössert die Faktentabelle. Einerseits erfordert jede Korrektur das Anlegen eines neuen Tupels, andererseits wird jedes Tupel um ein zusätzliches Attribut erweitert. Neben dem Ressourcenaufwand erhöht sich auch die Analysekomplexität. Der Anwender muss vor jeder zeitbezogenen Abfrage die gewünschte Transaktionszeit festlegen. Ansonsten aggregiert eine Auswertung die Fakten über alle Transaktionszeiten einer Gültigkeitsversion. Unter einer Gültigkeitsversion wird hier jede Version eines zeitbezogenen Entitätstyps mit derselben Gültigkeitszeit verstanden. Die folgende Beispielabfrage rechnet fälschlicherweise den Umsatz der Zeile 1 und 4 (vgl. Abbildung 2.3.2) zusammen, weil die Transaktionszeit nicht explizit vorgegeben wurde.

[79] Vgl. Zeitstempel in Quelldaten: S. 54

Beispielabfrage:

```
SELECT SUM(Umsatz)
FROM   FAKTEN
WHERE  G-Datum = 1/1/2001
  AND  G-ID = 10
  AND  K-ID = 11
  AND  P-ID = 100;
```

Der Anwender muss deshalb ergänzend auch den Bezugszeitpunkt bestimmen, damit die Abfrage die gewünschten Resultate liefert. Dies erfüllt zum Beispiel die Bedingung:

```
AND T-Datum = 25/2/2001
```

Oder falls das genaue Transaktionsdatum dem Anwender nicht bekannt ist:

```
AND T-Datum =
   (SELECT MAX(T-Datum)
    FROM FAKTEN
    WHERE T-Datum <= 30/1/2001
      AND  G-ID = 10
      AND  K-ID = 11
      AND  P-ID = 100)
```

30.1.2001 steht hier als Beispiel für ein gewünschtes Bezugsdatum. Die Unterabfrage ermittelt entsprechend der angegebenen Parameter die nächst kleinere Transaktionszeit.

Falls die Transaktionszeit ausschliesslich zur Identifikation von Datenänderungen dienen soll, ist es ausreichend, für jedes Objekt eine einzige Gültigkeitsversion zu verwalten. Die Transaktionszeit bestimmt dann lediglich den letzten Änderungszeitpunkt eines Objekts und muss deshalb nicht mehr zum Primärschlüssel gehören. Das Datenvolumen der Faktentabelle wird in diesem Fall noch durch das Änderungszeitstempelattribut, aber nicht durch weitere Datensätze belastet. Der Speicherbedarf eines zusätzlichen Attributs in der Faktentabelle ist dennoch nicht zu vernachlässigen. Neben der Verwendung von Zeitstempelattributen sollten deshalb auch Alternativen zur Verfolgung von Änderungen der Quelldaten in Betracht gezogen werden.[80]

[80] Vgl. Kapitel 2.1.3: S. 53

Der Nutzen der Transaktionszeit für die Entscheidungsunterstützung ist im Vergleich zur Gültigkeitszeit als gering einzustufen. Die Endbenutzer analytischer Systeme verlassen sich meist auf die jeweils aktuellsten Versionen vergangener und heutiger Daten, weil diese am wenigsten Fehler enthalten. Fragestellungen wie *„Hat der Entscheidungsträger aufgrund der ihm damals zur Verfügung stehenden Daten richtig gehandelt?“* erfordern zwar den Einbezug der Transaktionszeit, gehören bei der Unterstützung der taktischen und strategischen Entscheidungsfindung eher zu den Ausnahmen.

Wenn die Transaktionszeit dennoch zu berücksichtigen ist, lassen sich bei ihr wie auch bei der Gültigkeitszeit verschiedene temporale Auswertungsformen unterscheiden.[81] Die aktuelle Sicht zeigt zum Beispiel alle Fakten in Beziehung zu den derzeitig gültigen Dimensionsdaten an. Die Fakten und die aktuellen Dimensionsdaten können aber auch schon (mehrfach) korrigiert worden sein, so dass nicht nur bei der Gültigkeitszeit, sondern auch bei der Transaktionszeit zwischen dem ersten, einem beliebigen, dem faktbezogenen historischen oder dem letzten Zustand zu unterscheiden ist. Von dieser Differenzierung wird im Weiteren abgesehen, weil Analysen auf vergangenen Gültigkeitsversionen der Dimensionsdaten wie auch bei den Fakten selten erforderlich sind.

Intervallzeitstempelung

Die Intervall-Zeitstempelung hat gegenüber der Zeitpunkt-Zeitstempelung den Vorteil, dass sie unveränderte Objektversionen in einem einzigen Tupel zusammenfassen kann. Abbildung 2.3.3 zeigt die modifizierte Beispielfaktentabelle, die anstelle eines zeitpunktbestimmenden Zeitstempelattributs zwei intervallgrenzenmarkierende Zeitstempelattribute enthält.

G-ID	K-ID	P-ID	Intervallstart	Intervallende	Umsatz	Kosten	Gewinn	Zeile
10	4711	1022	1.1.2001	1.1.2001	50	40	10	1
10	4711	1022	3.1.2001	3.1.2001	20	8	12	2
...	...	...	...	...	...	...	...	3
20	1000	1048	3.1.2001	4.1.2001	25	15	10	4
...	...	...	...	...	...	...	...	5

Abbildung 2.3.3: Intervall-Zeitstempelung in der Faktentabelle

Analytische Systeme können Bewegungsgrössen wie Umsatz, Kosten oder Gewinn im Gegensatz zu Bestandesgrössen wie Vorratsmengen oder Vermögen nicht kontinuierlich, sondern nur punktuell bei einer vorhandenen Transak-

[81] Vgl. Kapitel 2.1.2: S. 50

tion erfassen. Deshalb ist es selten, dass Bewegungsgrössen wie in Zeile 4 dargestellt über mehrere Zeitpunkte unverändert bleiben. Solche Fakten sind in der Regel bereits nach einem einzigen Zeitpunkt nicht mehr gültig (vgl. Zeilen 1 und 2). Einzig das Nichtvorhandensein der Faktwerte stellt einen oft auftretenden, gleichbleibenden Zustand dar. Beispiel: Kunde 4711 kauft nie in der Region 30 ein. Nullwerte lassen sich jedoch implizit aus fehlenden Tupeln ableiten und müssen nicht explizit gespeichert werden.

Beurteilung

Die Intervall-Zeitstempelung kann die Menge der Tupel in der Faktentabelle reduzieren, wenn alle Fakten Bestandesgrössen sind. Eine Verbesserung der Speichereffizienz lässt sich jedoch nur unter günstigsten Bedingungen erzielen: Je länger die Zeitbindung der Fakten und je häufiger sich alle Fakten gleichzeitig ändern, desto kleiner ist der Speicherverbrauch. Die Zeitpunkt-Zeitstempelung ermöglicht durch die geeignete Wahl des Chronons ebenfalls längere Zeitbindungen wie einen Monat oder ein Jahr, ist jedoch bei der Zeitraumdefinition nicht so flexibel wie die Intervall-Zeitstempelung.

Die Intervall-Zeitstempelung erfordert komplexere Abfragestrukturen als die Zeitpunkt-Zeitstempelung. Einerseits ist die Bezugszeit nicht durch Übereinstimmung, sondern durch Intervallvergleiche zu suchen, andererseits müssen die Fakten bei einer Abfrage umständlicher als bei der Zeitpunkt-Zeitstempelung über die Zeit aggregiert werden. Zeile 4 in Abbildung 2.3.3 zeigt, dass der Umsatz für jeden Tag jeweils gleich 25 war. Damit sich der Umsatz über das gesamte oder einen Teil des Intervalls zusammenfassen lässt, muss die Anzahl der im betreffenden Zeitraum enthaltenen Chroni bekannt sein. Im erwähnten Beispiel sind dies zwei (Tage). Der Umsatz ist in diesem Beispiel folglich gleich 50: Anzahl der relevanten Chroni (2) multipliziert mit dem Wert des entsprechenden Faktattributs (25).

Die Änderungscharakteristik der Fakten - häufige Änderungen der Bewegungsgrössen und nicht gleichzeitige Änderungen der Bestandesgrössen - sowie die komplexere Abfragestrukturen sprechen gegen die Nutzung der Intervall-Zeitstempelung der Faktentabelle. Die weiteren Modellierungsüberlegungen gehen deshalb nicht von der Intervall-, sondern von der Zeitpunkt-Zeitstempelung der Faktentabelle aus.

Erweiterung der Dimensionstabellen

Im Gegensatz zu den Fakten weisen die Tupel der Dimensionstabellen im konventionellen Sternschema keinen Zeitbezug auf. Jede Änderung der Dimensionsdaten hat zur Folge, dass der vergangene Zustand verloren geht. Abgesehen

von der aktuellen Sicht lassen sich folglich keine weiteren temporalen Auswertungsformen darstellen. Die zeitabhängigen Attribute im engeren Sinn müssen zur Unterstützung mehrerer temporaler Auswertungsformen so modelliert werden, dass sie in ihrem historischen Zusammenhang analysierbar sind.

Wie bei der Faktentabelle ermöglichen Zeitstempelattribute in den Dimensionstabellen die historische Abbildung aller Werte. Zeitabhängig zu speichernde Tupel werden dazu um ein oder zwei Zeitstempelattribute erweitert. Ein *einzelnes* Zeitstempelattribut kann entweder den Gültigkeitszeitpunkt oder den Gültigkeitsbeginn der neuen und damit implizit das Gültigkeitsende der vorhergehenden Objektversion bestimmen. Mit *zwei* intervallgrenzenmarkierenden Zeitstempelattributen kann der gesamte Gültigkeitszeitraum einer Objektversion in einem einzigen Tupel abgespeichert werden.[82]

Während viele Fakten bei jedem Zeitpunkt einen neuen Wert annehmen, bleiben Dimensionsdaten meist über mehrere aufeinanderfolgende Zeitpunkte konstant. Deshalb ist die Intervall-Zeitstempelung gegenüber der Zeitpunkt-Zeitstempelung nicht nur die übersichtlichere, sondern auch die speichereffizientere Methode zur Abbildung zeitabhängiger Dimensionsdaten.

Abbildung 2.3.4 und 2.3.5 zeigen zwei Varianten der Intervallzeitstempelung anhand der Dimensionstabelle PRODUKT. Die erste Variante verwendet ein Zeitstempelattribut weniger und ist deshalb speichereffizienter. Zudem ist sie fortschreibungsfreundlicher als die zweite Variante, weil der Aktualisierungsprozess einerseits nur ein Zeitstempelattribut anpassen, andererseits keine Intervallüberschneidungen oder -lücken überprüfen muss.[83] Lücken und Überschneidungen der Gültigkeitsintervalle sind bei den Dimensionsdaten nicht zugelassen, weil zu jedem Fakt und Bezugszeitpunkt genau ein Dimensionstupel zuordenbar sein muss. Das Attribut Intervallende in Abbildung 2.3.5 muss deshalb nicht zum Primärschlüssel gehören.

Produkt-ID	Intervallstart	Name	Kategorie	Hersteller
1022	5.1.1999	Chnuschpi	Schokolade	Vander
1023	27.1.1999	N&N	Schokolade	Venus
...	...	...	...	...
1022	1.1.2001	Chnuschpi	Kassenprodukte	Vander
...	...	...	...	...

Abbildung 2.3.4: Produktdimension mit einem Zeitstempelattribut

[82] Vgl. Kapitel 2.2.3: S. 64
[83] Vgl. Kimball 2000c

Die zweite Variante (Abbildung 2.3.5) speichert das Intervallende explizit in einem zusätzlichen Zeitstempelattribut. Ein nicht vorhandener Wert beim Attribut Intervallende bedeutet hier, dass die entsprechende Version heute noch gültig ist.

Produkt-ID	Intervallstart	Intervallende	Name	Kategorie	Hersteller
1022	5.1.1999	31.12.2000	Chnuschpi	Schokolade	Vander
1023	27.1.1999		N&N	Schokolade	Venus
...	...	...	...	...	...
1022	1.1.2001		Chnuschpi	Kassenprodukte	Vander
...	...	...	...	...	...

Abbildung 2.3.5: Produktdimension mit zwei Zeitstempelattributen

Wie das nachfolgende Beispiel einer temporalen SQL-Abfrage zeigt, vereinfacht die Speicherung des gesamten Gültigkeitsintervalls die Abfrageformulierung und verbessert in den meisten Fällen die Laufzeiteffizienz.[84]

Beispielabfrage: *„Zu welcher Kategorie hat das Produkt 1022 am 7.7.2001 gehört?“*

Variante 1: Tupel enthalten nur ein Zeitstempelattribut (Intervallstart)

```
SELECT Kategorie
FROM   PRODUKT
WHERE  Produkt-ID = 1022
  AND  Intervallstart =
       (SELECT MAX(Intervallstart)
        FROM   PRODUKT
        WHERE  Produkt-ID = 1022
        AND Intervallstart <= 7/7/2001);
```

[84] Vgl. Kimball 2000c

Variante 2: Tupel enthalten zwei Zeitstempelattribute (Intervallstart und -ende)

```
SELECT Kategorie
FROM   PRODUKT
WHERE  Produkt-ID = 1022
AND    Intervallstart <= 7/7/2001
AND    (Intervallende >= 7/7/2001 OR Intervallende IS NULL);
```

Die Abfrage der Variante 2 kann die Bezugszeit direkt mit den Gültigkeitsintervallen der Dimensionstupel vergleichen. Die Abfrage und das Modell sind besser verständlich, weil in jedem Tupel nicht nur eine Intervallgrenze, sondern die gesamte Gültigkeitszeit abgebildet ist. Variante 1 muss dagegen zunächst den grössten Wert des Zeitstempelattributs Intervallstart suchen, der kleiner oder gleich dem gewünschten Bezugsdatum ist. Anschliessend lässt sich der gefundene Wert als Kriterium für das Zeitstempelattribut der Dimensionstabelle nutzen.

Der Vergleich eines Bezugsdatums mit den Intervallgrenzen legt bei Variante 2 die Nutzung des **BETWEEN ... AND** Operators (SQL) nahe. Dies ist möglich, wenn das Intervallende auch bei einem aktuell gültigen Tupel nicht leer ist, sondern zum Beispiel den höchst möglichen Datumswert der Datenbank enthält. Die beiden Intervallgrenzenvergleiche der Variante 2 könnten in diesem Fall durch die folgende SQL-Anweisung ersetzt werden:

```
...WHERE 7/7/2001 BETWEEN Intervallstart AND Intervallende...
```

Abbildung 2.3.6 zeigt das temporal erweiterte Sternschemabeispiel. Die Dimensionstabellen KUNDE und PRODUKT wurden durch zwei Zeitstempelattribute ergänzt, um die darin enthaltenen zeitabhängigen Attribute in ihrem historischen Kontext abzubilden. Die erste Variante, die nur ein Zeitstempelattribut verwendet, soll hier aufgrund der komplexeren Abfrageformulierungen nicht weiter verfolgt werden.

Die Kunden- und die Produkttabelle können durch die Primärschlüsselerweiterung mehrere Versionen eines zeitbezogenen Entitätstyps enthalten. Der Beziehungstyp dieser Tabellen musste deshalb so angepasst werden, dass sich jedes Faktattribut mehreren Dimensionstupel zuordnen lässt (m:n-Beziehung).

Der Kundenname wird bei jeder Änderung ebenfalls zeitbezogen gespeichert, obwohl er als zeitabhängig im weiteren Sinn deklariert wurde. Abhilfe würde hier eine Abspaltung des Kundennamens in eine eigene Tabelle schaffen. Dadurch würde das Modellverständnis jedoch eher verschlechtert als verbessert.

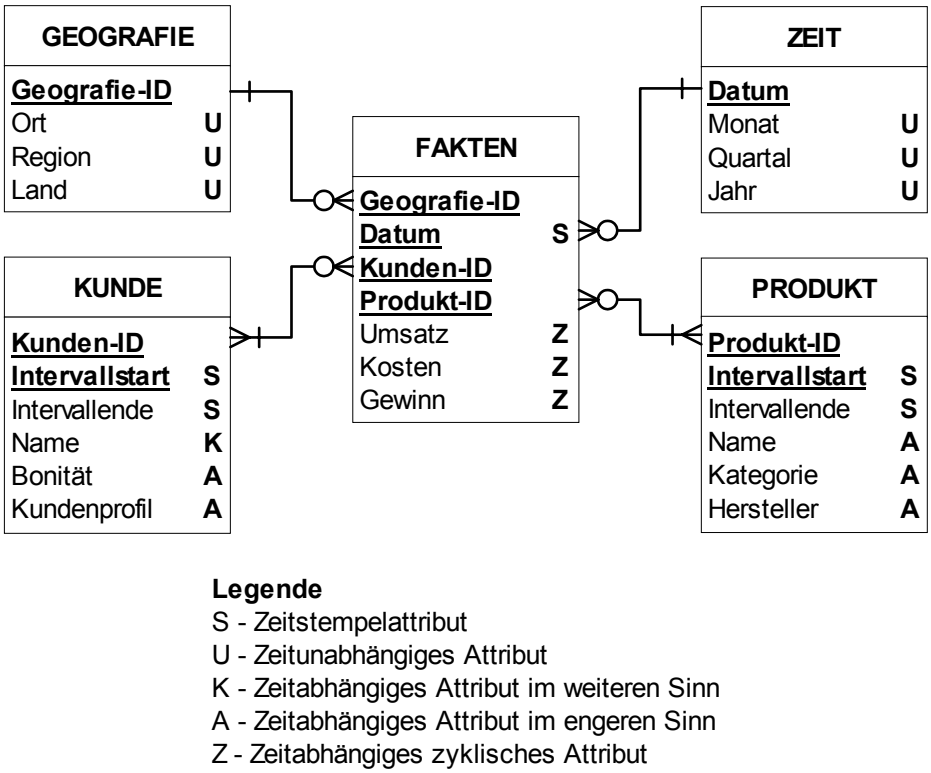


Abbildung 2.3.6: Temporal erweitertes Sternschema

Temporale Auswertungsformen im erweiterten Sternschema

Nachdem die veränderbaren Dimensionsdaten zeitbezogen abgebildet sind, können die Fakten nicht mehr nur in Beziehung zu den aktuellen, sondern auch zu vergangenen Versionen der Dimensionsdaten analysiert werden. Zu unterscheiden sind die vier temporalen Auswertungsformen:[85]

- Aktuelle Sicht
- Faktbezogene historische Sicht
- Dimensionsbezogene historische Sicht
- Ursprüngliche Sicht

Beispielabfrage: Der Umsatz soll für jedes Jahr und jede Produktkategorie angezeigt werden.

[85] Vgl. Kapitel 2.1.2: S. 71

Die SQL-Abfrage könnte auf dem ursprünglichen, nicht temporal erweiterten Sternschema folgendermassen aussehen (vgl. Abbildung 2.3.1):[86]

```
TRANSFORM SUM(FAKTEN.Umsatz)
SELECT    PRODUKT.Kategorie
FROM      FAKTEN, PRODUKT, ZEIT
WHERE     ZEIT.Datum = FAKTEN.Datum
  AND     FAKTEN.Produkt-ID = PRODUKT.Produkt-ID
GROUP BY  PRODUKT.Kategorie
PIVOT     ZEIT.Jahr;
```

Die Verwendung derselben Abfrage im temporal erweiterten Sternschema führt zu einer Mehrfachzählung der Faktwerte, falls die Produktdimension mehrere Versionen eines Produkts enthält. Die Abfrage muss so angepasst werden, dass nur die gewünschte Version der Produkte den Fakten zugeordnet wird. Die nächste Abfrage enthält deshalb eine zusätzliche Bedingung für die temporal erweiterte Produktdimension:

```
TRANSFORM SUM(FAKTEN.Umsatz)
SELECT    PRODUKT.Kategorie
FROM      FAKTEN, PRODUKT, ZEIT
WHERE     ZEIT.Datum = FAKTEN.Datum
  AND     FAKTEN.Produkt-ID = PRODUKT.Produkt-ID
  AND     PRODUKT.Intervallstart <= <Bezugsdatum>
  AND     (PRODUKT.Intervallende >= <Bezugsdatum>
            OR PRODUKT.Intervallende IS NULL)
GROUP BY  PRODUKT.Kategorie
PIVOT     ZEIT.Jahr;
```

Der Platzhalter `<Bezugsdatum>` wird abhängig von der gewünschten temporalen Auswertungsform ersetzt:

- Aktuelle Sicht: `<Bezugsdatum>` = `NOW`
- Faktbezogene historische Sicht: `<Bezugsdatum>` = `FAKTEN.Datum`
- Dimensionsbezogene historische Sicht: `<Bezugsdatum>` = benutzerdefiniert

[86] In der Abfrage wird die von MS Access unterstützte SQL-Erweiterung zur Definition einer Kreuztabelle verwendet (TRANSFORM ... PIVOT).

NOW steht für eine SQL-Funktion, die das aktuelle Datum aus der Systemuhr liest.[87] <Bezugsdatum> ist bei der dimensionsbezogenen historischen Sicht ein von den Endbenutzern zu definierendes Datum (zum Beispiel 19.3.2000).

Die noch nicht dargestellte *ursprüngliche Sicht* verwendet bei der Abfrage jeweils die erste Version der zeitbezogenen Dimensionsobjekte. Im Gegensatz zu den anderen temporalen Auswertungsformen lässt sich der Gültigkeitsbezug in diesem Fall nicht direkt durch die Zuordnung eines vorgegebenen Datums oder Datenbankfeldes ermitteln. Zunächst ist für jedes Objekt die kleinste Gültigkeitszeit zu bestimmen:[88]

```
CREATE VIEW ERSTE_PRODUKTVERSION AS
   SELECT   PRODUKT.Produkt-ID,
              MIN(PRODUKT.Intervallstart) AS MinIntervallstart
   FROM     PRODUKT
   GROUP BY PRODUKT.Produkt-ID;
```

Das Resultat wird anschliessend mit der Produktdimension verknüpft. Die Abfrage zeigt deshalb alle Fakten in Abhängigkeit zur ersten Version jeder Produktkategorie an:

```
TRANSFORM SUM(FAKTEN.Umsatz)
SELECT    PRODUKT.Kategorie
FROM      FAKTEN, ZEIT, ERSTE_PRODUKTVERSION
WHERE     ZEIT.Datum = FAKTEN.Datum
  AND     PRODUKT.Produkt-ID = ERSTE_PRODUKTVERSION.Produkt-ID
  AND     PRODUKT.Intervallstart =
            ERSTE_PRODUKTVERSION.MinIntervallstart
GROUP BY  PRODUKT.Kategorie
PIVOT     ZEIT.Jahr;
```

Alle Abfragen sind soweit parametrisierbar, dass die Endbenutzer für jede Dimension nur die gewünschte temporale Auswertungsform und bei der dimensionsbezogenen historischen Sicht zusätzlich das Bezugsdatum angeben müssen.

---

87 Die Standardkonstante (ab SQL92) wäre CURRENT_DATE (vgl. Groff, Weinberg 1999: S. 84). Diese wird jedoch von MS Access nicht unterstützt.

88 Anstelle einer VIEW müsste bei MS Access eine gespeicherte Abfrage verwendet werden.

Beurteilung

1. Funktionalität

Das temporal erweiterte Sternschema unterstützt *alle* zuvor definierten Auswertungsformen. Neben einfachen Datenänderungen wie etwa die Umbenennung eines Produktnamens lassen sich durch die zeitbezogene Speicherung der Dimensionstupel auch Modifikationen der Hierarchiezuordnung abbilden. Solche Anpassungen sind zum Beispiel ein Bonitätsklassenwechsel eines Kunden oder eine Veränderung der Kategoriezugehörigkeit eines Produkts.

Nicht darstellbar sind Modifikationen an der Dimensionshierarchie. Das Hinzufügen, Umbenennen oder Löschen einer Hierarchieebene wird durch die hier verwendete temporale Schemaerweiterung nicht zeitbezogen berücksichtigt. Dies bedeutet zum Beispiel, dass durch das Löschen eines Dimensionsattributs alle darunter gespeicherten Werte verloren gehen.

Jede Dimension, die ein oder mehrere zeitbezogene Attribute im engeren Sinn enthält, wird als Ganzes im historischen Kontext abgebildet. Die fehlende temporale Unterscheidung der Dimensionsattribute führt zu zusätzlicher Redundanz.

2. Benutzerfreundlichkeit

Das Sternschema ist durch die temporalen Erweiterungen weniger gut verständlich. Neben den konventionellen Ausprägungskriterien enthalten Dimensionstabellen auch Zeitstempelattribute. Die Anwender müssen bei der Analysedefinition diese inhaltlich verschiedenen Attributklassen unterscheiden und anwenden können. Die Abfragedefinition ist zwar komplexer, lässt sich jedoch wie auch beim konventionellen Sternschema für gewisse Anwendungsbereiche wie etwa die OLAP-Navigation so parametrisieren, dass die neu zur Verfügung stehenden temporalen Auswertungsformen keine zusätzlichen Anforderungen an die Endbenutzer stellen.

Analysen auf dem erweiterten Sternschema sind weniger effizient, weil die Datenbank bei jeder Abfrage zunächst die benötigten Klassifikationshierarchien in Abhängigkeit von der gewünschten temporalen Auswertungsform auswählen muss. Zudem ist es sehr aufwändig, zu jeder Version der Dimensionsattribute präaggregierte Fakten zu verwalten. Die eingeschränkte Optimierungsmöglichkeit verursacht deshalb besonders bei zeitbezogenen Abfragen längere Wartezeiten.

3. Einfluss auf die Ressourcen

Die zeitbezogen gespeicherten Dimensionsdaten benötigen aufgrund der zusätzlichen Attribute (Zeitstempel) und Tupel (Objektversionen) mehr Speicherressourcen. Die Zunahme des Speicherbedarfs ist im Verhältnis zum Gesamtverbrauch als eher gering einzustufen, denn die in der Regel sehr viel grössere Faktentabelle wird in diesem Modell nicht verändert.

Die Anforderungen an die periodischen Aktualisierungsprozesse sind ebenfalls etwas höher. Anstelle des einfachen Überschreibens eines nicht mehr gültigen Tupels muss eine Änderung bei zeitbezogenen Dimensionsdaten durch das Einfügen eines neuen Tupels abgebildet werden. Zusätzlich sind die Zeitstempelattribute nachzuführen. Das Intervallende des nicht mehr gültigen Tupels wird gleich dem Wert des Intervallstarts (Änderungsdatum) abzüglich eines Chronons gesetzt. Auf diese Weise wird sichergestellt, dass sich die Gültigkeitsintervalle eines zeitbezogenen Entitätstyps nicht überschneiden und auch keine Lücken entstehen.

4. Unterstützung des erweiterten Sternschemas in OLAP-Datenbanken

Die spezifizierten Modellerweiterungen lassen sich durch das Hinzufügen der Zeitstempelattribute und durch Anpassungen der Aktualisierungsprozesse im zentralen Data Warehouse und den Data Marts realisieren. Damit den Endbenutzern auch die gewünschte temporale Funktionalität zur Verfügung steht, müssen die Abfragestrukturen der analytischen Anwendungen entsprechend ergänzt werden.

Im Gegensatz zu statischen Berichten generieren OLAP-Anwendungen ihren Abfragecode zur Laufzeit. Datenbankentwickler können folglich die Abfragestrukturen nicht im Voraus an das veränderte Modell anpassen. Die OLAP-Anwendung muss die Verwendung der Zeitstempel bereits vorsehen oder eine geeignete Abfrageschnittstelle zur Verfügung stellen, mit welcher Endbenutzer den Abfragecode gemäss der gewünschten temporalen Auswertungsform modifizieren können. Der manuelle Eingriff bei der Abfragedefinition ist aufwändig und vielen Anwendern von Ad-hoc-Analysewerkzeugen nicht zumutbar. Einige kommerziell verfügbare OLAP-Systeme stellen zudem bestimmte Anforderungen an das Datenmodell, welche die Implementation des temporal erweiterten Schemas verhindern. Diese Systeme gehen von einem konventionellen Stern- oder Schneeflockenschema aus, bei dem die Dimensionstabellen immer in einer 1:n-Beziehung zu den Fakten stehen. Die zwischen den Fakten und den zeitbezogenen Dimensionsdaten bestehenden m:n-Beziehungen werden hingegen nicht unterstützt.

Berücksichtigung veränderbarer Dimensionsstrukturen

Das vorgestellte temporal erweiterte Sternschema bildet neben den Fakten auch alle Dimensionsdaten zeitbezogen ab. Es vernachlässigt jedoch *Strukturänderungen*, die durch das Einfügen, Umbenennen oder Löschen eines Attributs verursacht werden. Veränderbare Dimensionsstrukturen können durch ein Modell berücksichtigt werden, das die hierarchische Zuordnung der Dimensionsattribute ebenfalls zeitbezogen abbildet. Ein solches Modell wird zum Beispiel von Stock[89] oder Eder/Koncilia[90] vorgeschlagen. Stellvertretend für diese Modellierungsvarianten zeigt Abbildung 2.3.7 den Ansatz von Stock anhand der Produktdimension.

Die Ebenen der Klassifikationshierarchien werden nicht durch die Angabe expliziter Attributnamen, sondern durch entsprechende Werte in den Tupeln beschrieben. Jedes Tupel enthält ein Identifikationsattribut (ID), Intervallzeitstempelattribute (I-Start, I-Ende), den Namen der Hierarchieebene (Ebene), den dazugehörigen Wert dieser Ebene (Wert) sowie den Verweis auf den in der Hierarchie übergeordneten Wert (ID-ÜG, I-Start-ÜG). Der Primärschlüssel besteht aus dem Identifikationsattribut ID und dem Zeitstempelattribut I-Start.

ID	I-Start	I-Ende	Wert	ID-ÜG	I-Start-ÜG	Ebene
10	1.1.1999	31.12.2000	Chnuschpi	14	1.1.1999	Name
11	1.1.1999	31.12.2000	N&N	15	1.1.1999	Name
12	1.1.1999	31.12.2000	Raiser	15	1.1.1999	Name
13	1.1.2000	31.12.2000	Caofina	16	1.1.2000	Name
14	1.1.1999	31.12.2000	Schokolade	18	1.1.1999	Kategorie
15	1.1.1999	31.12.2000	Schokolade	17	1.1.1999	Kategorie
16	1.1.2000	31.12.2000	Schokogetränke	18	1.1.1999	Kategorie
17	1.1.1999		Saturn			Hersteller
18	1.1.1999		Vander			Hersteller
...	...	...	...	...	...	...
10	1.1.2001		Chnuschpi	18	1.1.1999	Name
11	1.1.2001		N&N	17	1.1.1999	Name
12	1.1.2001		Raiser	17	1.1.1999	Name
13	1.1.2001		Caofina	18	1.1.1999	Name

Abbildung 2.3.7: Strukturänderungen in einer Produktdimension

[89] Vgl. Stock 2001: S. 142
[90] Vgl. Eder, Koncilia 2001

Mit dieser Dimensionstabelle lässt sich die Dimensionshierarchie zu jedem gewünschten Bezugszeitpunkt rekursiv aufbauen.[91] Ist ein Wert oder eine hierarchische Zuordnung nicht mehr gültig, dann wird die Dimensionstabelle mit neuen Tupeln ergänzt. Das Beispiel in Abbildung 2.3.7 zeigt, dass die Ebene Kategorie im Jahr 2001 nicht mehr gültig ist. Die Elemente der Ebene Name werden deshalb ab dem 1.1.2001 nicht mehr den Elementen der Ebene Kategorie, sondern den entsprechenden Elementen der Ebene Hersteller zugeordnet. Obwohl die alte Dimensionsstruktur nicht mehr gültig ist, können die Anwender sie durch die Wahl eines Bezugsdatums vor dem Jahr 2001 noch verwenden.

Beurteilung

Abgesehen von der zusätzlichen Berücksichtigung der Strukturänderungen gelten hier dieselben Kritikpunkte wie bei jenem temporal erweiterten Modell, das nur Datenänderungen zeitbezogen abbilden kann. Das Verständnis, die Abfragedefinition und die Datenaktualisierung werden jedoch durch die Aufteilung der hierarchischen Attributabhängigkeiten auf mehrere Tupel zusätzlich erschwert. OLAP-Systeme müssen zur Unterstützung dieses Modells nicht nur die Verwendung von Zeitstempeln, sondern auch eine adäquate Nutzung rekursiver Dimensionen vorsehen.

2.3.2 Alternative Ansätze

Viele kommerzielle OLAP-Systeme unterstützen die Verwendung temporal erweiterter Dimensionstabellen nicht.[92] Deshalb ist diese Werkzeugklasse zur Abbildung der temporalen Auswertungsformen häufig auf *alternative* Modellierungsmethoden angewiesen. Im folgenden werden drei Ansätze erörtert, die sich mit der Modellierung mehrerer temporaler Auswertungsformen in OLAP-Systemen beschäftigen:

- Slowly Changing Dimensions
- Alternative Fremdschlüssel
- Zeitschalterverfahren

[91] Vgl. Rekursive Dimensionen in Kapitel 1.3.2: S. 32
[92] Vgl. Oehler 2000: S. 118

Slowly Changing Dimensions (SCD)[93]

Kimball bezeichnet die Dimensionen als „slowly changing“, weil sich die Dimensionsdaten im Vergleich zu den Fakten weniger schnell verändern. Er beschreibt in diesem Konzept drei Möglichkeiten im Umgang mit Datenänderungen in Dimensionstabellen.

1. Überschreiben (SCD Typ 1)

Die erste Methode überschreibt die bestehenden Daten. Dies ist die einfachste Möglichkeit und stellt bei den meisten OLAP-Systemen das Standardvorgehen dar. Eine Historisierung findet jedoch nicht statt, weil sämtliche vergangenen Zustände der Dimensionsdaten dadurch verloren gehen. Das Überschreiben ermöglicht von den vier temporalen Auswertungsformen lediglich die Abbildung der *aktuellen Sicht*. Abbildung 2.3.8 zeigt dieses Vorgehen anhand eines Beispiels. Die Kategorie des Produkts 1022 wird von *Schokolade* nach *Kassenprodukte* geändert.

Produktdimension ***vor*** der Änderung:

ID	Name	Kategorie	Hersteller
...	...	...	...
1022	Chnuschpi	Schokolade	Vander
...	...	...	...

Produktdimension ***nach*** der Änderung:

ID	Name	Kategorie	Hersteller
...	...	...	...
1022	Chnuschpi	***Kassenprodukte***	Vander
...	...	...	...

Abbildung 2.3.8: Überschreiben bestehender Daten

2. Hinzufügen (SCD Typ 2)

Die zweite Methode verändert bestehende Daten nicht. Sie fügt die aktuellen Werte in neuen Tupeln hinzu. Ein künstliches Schlüsselattribut KID erweitert die Dimensionstabelle, weil das ursprüngliche Primärschlüsselattribut ID die Entitätsintegrität nicht mehr sicherstellen kann. Diese Anpassung muss auch in

[93] Vgl. Kimball 1996a: S. 100-106

der Faktentabelle berücksichtigt werden, in der nun anstelle des alten ebenfalls das entsprechende neue Schlüsselattribut abzubilden ist. Dieses Vorgehen ermöglicht die Darstellung der *faktbezogenen historischen Sicht*, weil vergangene Fakten den entsprechend früher gültigen Dimensionsdaten und neue Fakten der aktuellen Version der Dimensionsdaten zugeordnet sind. Abbildung 2.3.9 zeigt die Produktdimension nach der Änderung. In diesem Beispiel zählen die Fakten des Produkts *Chnuschpi*, vor der Kategorieänderung zur Schokoladenkategorie, danach zu den Kassenprodukten.

KID	ID	Name	Kategorie
...	...	...	...
100	1022	Chnuschpi	Schokolade
...	...	...	...
748	***1022***	***Chnuschpi***	***Kassenprodukte***
...	...	...	...

Abbildung 2.3.9: Hinzufügen veränderter Daten

Bei der Frage nach dem Umsatz des Produkts *Chnuschpi* werden anstelle eines einzigen Resultats zwei oder mehrere Werte erscheinen, weil dasselbe Produkt durch mehrere Tupel beschrieben wird. Dieses Problem kann durch eine Gruppierung über das ursprüngliche Schlüsselattribut ID gelöst werden.

Die Darstellung der *aktuellen Sicht* ist theoretisch möglich, jedoch in OLAP-Anwendungen unverhältnismässig aufwändig. Die dazu notwendige Abfragesequenz zur Ermittlung der aktuellen Sicht könnte etwa wie folgt skizziert werden:

1. Finde zu jedem Dimensionsobjekt (Produkt) die neuste Version.

2. Erstelle eine View, in der die Daten aller Versionen eines Dimensionsobjekts mit Ausnahme des Primärschlüssels durch die Daten der aktuellen Version ersetzt werden.

3. Verwende zur eigentlichen Abfragedefinition anstelle der ursprünglichen Dimension die zuvor erstellte View.

Die *ursprüngliche Sicht* lässt sich nach dem gleichen Muster wie die aktuelle Sicht ermitteln. Anstelle der neusten Version wird jedoch die älteste verwendet. Die *dimensionsbezogene Sicht* ist nicht möglich, weil in diesem Modell die Dimensionstabellen keine Zeitstempel enthalten. Sollten dennoch Zeitstempelattribute in die Dimensionstabellen eingeführt werden, entspräche dieses Modell dem bereits vorgestellten temporal erweiterten Sternschema.

3. Den Vergangenen und neuen Zustand speichern (SCD Typ 3)

Die dritte Methode überschreibt die bestehenden Daten. Sie bildet deshalb genau wie die erste Methode die *aktuelle Sicht* ab. Die Dimensionstabelle enthält jedoch für zeitabhängig zu speichernde Werte jeweils ein weiteres Attribut, das einen vergangenen wie etwa den vorhergehenden oder den ursprünglichen Zustand speichert. Abbildung 2.3.10 zeigt, dass die Produktdimension zwei Attribute für die Produktkategorie enthält. Während Kategorie_ursprünglich immer denselben Wert enthält, wird Kategorie_aktuell bei einer Änderung überschrieben.

Produktdimension ***vor*** der Änderung:

ID	Name	Kategorie_ursprünglich	Kategorie_aktuell	Hersteller
...	...	...	...	...
1022	Chnuschpi	Schokolade	Schokolade	Vander
...	...	...	...	...

Produktdimension ***nach*** der Änderung:

ID	Name	Kategorie_ursprünglich	Kategorie_aktuell	Hersteller
...	...	...	...	...
1022	Chnuschpi	Schokolade	***Kassenprodukte***	Vander
...	...	...	...	...

Abbildung 2.3.10: Speicherung des ursprünglichen und aktuellen Zustands

Ein zusätzliches Attribut ermöglicht neben der *aktuellen Sicht* auch die Abbildung einer vergangenen, zum Beispiel der *ursprünglichen Sicht*. Dazwischenliegende Zustände werden hingegen nicht gespeichert. Deshalb eignet sich diese Methode besonders für Attribute, die sich selten ändern und nur wenige Zustände annehmen können wie etwa das Attribut *Zivilstand*.

Beurteilung

Die von Kimball vorgeschlagenen Methoden zur Behandlung zeitabhängiger Dimensionsdaten sind im Gegensatz zu den temporal erweiterten Schemata einfacher zu implementieren. Die drei Varianten basieren auf dem konventionellen Sternschema und setzen dabei weder den Umgang mit Zeitstempeln, noch die Abbildung von m:n-Beziehungen voraus. Jede der drei Modellierungsmethoden unterstützt jeweils nur eine (SCD Typ 1 und 2) beziehungsweise zwei (SCD Typ 3) direkt ableitbare temporale Auswertungsformen. Die kombinierte Verwendung der häufig benötigten *aktuellen* und *faktbezogenen historischen Sicht* wird vom SCD Typ 2 nur bedingt unterstützt. Während die faktbezogene historische Sicht direkt zur Verfügung steht, können die *aktuelle*

oder die *ursprüngliche Sicht* nur dann aus den bestehenden Daten abgeleitet werden, wenn die OLAP-Anwendung eine mächtige Abfragesprache wie SQL zur Verfügung stellt und die Endbenutzer auch fähig sind, diese adäquat einzusetzen. Die Voraussetzung solcher Kenntnisse widerspricht jedoch der Anforderung, dass Anwender in einer OLAP-Umgebung ohne die Verwendung von Programmierlogik in der Lage sein sollen, auch komplexe Auswertungen zu definieren.[94] Deshalb muss sich der Entwickler bereits bei der Definition eines Datenwürfels für eine bestimmte temporale Auswertungsform entscheiden. Falls dennoch mehrere temporale Auswertungsformen abzubilden sind, können entsprechend mehrere Datenwürfel erstellt werden.

Alternative Fremdschlüssel[95]

Die Variante 2 des „Slowly Changing Dimensions"-Konzept hat den Nachteil, dass sie den Endbenutzern nur die faktbezogene Sicht ohne zusätzlichen Abfrageaufwand ermöglicht. Ausgehend von diesem Modell hat Marco eine Modellierungsvariante vorgestellt, mit der sich neben der *faktbezogenen historischen Sicht* auch die *aktuelle Sicht* direkt nutzen lässt. In der Faktentabelle wird dazu für beide Sichten ein jeweils getrenntes Fremdschlüsselattribut zu jeder zeitbezogenen Dimension gespeichert. Abbildung 2.3.11 zeigt die Struktur der Fakten- und Produktdimensionstabelle. Die Fremdschlüssel Produkt-ID_historisch und Produkt-ID_aktuell der Faktentabelle lassen sich beide mit dem Primärschlüssel Produkt-ID der Dimensionstabelle verbinden (1:n-Beziehung). Vor der Änderung sind die Werte der beiden Fremdschlüsselattribute noch identisch.

Dimensionstabelle ***vor*** der Änderung:

Produkt-ID	**Name**	**Kategorie**	**Hersteller**
...	...	...	...
1022	Chnuschpi	Schokolade	Vander
...	...	...	...

Faktentabelle ***vor*** der Änderung:

Produkt-ID_historisch	**Produkt-ID_aktuell**	**Jahr**	**Umsatz**
...	...	...	...
1022	1022	2001	700
...	...	...	...

Abbildung 2.3.11: Produktdimension und Faktentabelle vor der Änderung

[94] Vgl. Oehler 2000: S. 33
[95] Vgl. Marco 2000: S. 219

Der vergangene Wert des Attributs Produkt-ID_aktuell wird, wie in Abbildung 2.3.12 dargestellt, in der Faktentabelle nach der Änderung durch den neuen Wert ersetzt. Wie beim SCD Typ 2 könnte hier auch ein weiteres Attribut in die Dimensionstabelle eingeführt werden, mit dem sich die auf verschiedene Tupel verteilten Versionen nach ihrer Objektzugehörigkeit gruppieren lassen.

Dimensionstabelle ***nach*** der Änderung:

Produkt-ID	**Name**	**Kategorie**	**Hersteller**
...	...	...	...
1022	Chnuschpi	Schokolade	Vander
...	...	...	...
1078	Chnuschpi	***Kassenprodukte***	Vander
...	...	...	...

Faktentabelle ***nach*** der Änderung:

Produkt-ID_historisch	**Produkt-ID_aktuell**	**Jahr**	**Umsatz**
...	...	...	...
1022	***1078***	2001	700
...	...	...	...
1078	***1078***	***2002***	***1800***
...	...	...	...

Abbildung 2.3.12: Produktdimension und Faktentabelle nach der Änderung

Wenn das Primärschlüsselattribut Produkt-ID in einer Abfrage mit dem Fremdschlüsselattribut Produkt-ID_historisch verknüpft wird, ordnet die Datenbank jene Dimensionswerte den Fakten zu, die beim Erfassungszeitpunkt gültig waren (*faktbezogene historische Sicht*). Benutzt der Anwender jedoch den alternativen Fremdschlüssel Produkt-ID_aktuell, dann ordnet die Datenbank auch vergangene Fakten den aktuellen Dimensionswerten zu (*aktuelle Sicht*).

Falls die OLAP-Anwendung die alternative Zuordnung der Fremdschlüsselattribute nicht zulässt, kann das gewünschte Ergebnis auch durch die Verwendung eines Duplikats der Dimensionstabelle erzielt werden. Mit der entsprechenden Benennung der Dimensionstabellen wie etwa *PRODUKT_HISTORISCH* und *PRODUKT_AKTUELL* ist es für die Anwender offensichtlich, welche Dimension das Abfrageresultat im gewünschten temporalen Zusammenhang darstellt.

Beurteilung

Die Verwaltung alternativer Fremdschlüssel ermöglicht die Nutzung der beiden wichtigsten temporalen Auswertungsformen in einem einzigen Schema, ohne den Endbenutzer mit komplexem Abfragecode zu überfordern. Die *aktuelle* und die *faktbezogene historische Sicht* können zudem gemeinsam in einer Abfrage kombiniert werden. Der Umsatz lässt sich zum Beispiel sowohl in Abhängigkeit von den aktuellen Werten der Produktdimension als auch von vergangenen Werten der Kundendimension darstellen.

Im Gegensatz zu allen zuvor vorgestellten Modellen werden hier bestehende Werte der Faktentabelle verändert. Die Erweiterung der Faktentabelle um jeweils ein Attribut pro Dimensionstabelle stellt hohe Anforderungen an die Speicherressourcen und den Wartungsaufwand.

Zeitschalterverfahren[96]

Viele OLAP-Systeme sehen die zeitbezogene Speicherung der Daten in der Faktentabelle vor. Zeitstempel in den Dimensionstabellen werden hingegen kaum unterstützt. Das Zeitschalterverfahren passt sich diesen Rahmenbedingungen an und hinterlegt die Gültigkeitsinformation jeder zeitbezogenen Dimension in einer eigenen Faktentabelle. Die *faktbezogene historische Sicht* kann durch die Verrechnung dieser Gültigkeitsinformation (Zeitschalter) mit der ursprünglichen Faktentabelle abgeleitet werden.

Beispiel

Als Zielgrösse soll in diesem Beispiel der Umsatz in Abhängigkeit der Zeit (Jahr) und Produktkategorie (Kategorie) in der faktbezogenen historischen Sicht dargestellt werden. Zur Vereinfachung der Illustration enthält das Ausgangsschema nur die Faktentabelle und Produktdimensionstabelle (vgl. Abbildung 2.3.13).

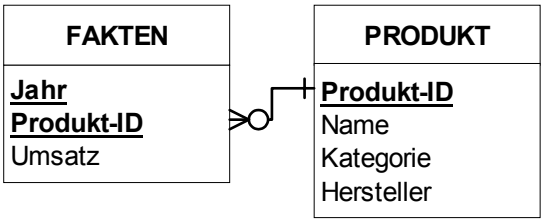


Abbildung 2.3.13: Vereinfachtes Sternschemabeispiel

[96] Vgl. Oehler 2000: S. 118

Die Produktdimension wird in eine Faktentabelle transformiert und mit zusätzlichen Attributen ergänzt:

- Das Zeitstempelattribut Jahr und dessen Werte sind identisch mit jenen der ursprünglichen Faktentabelle. Die vorhandenen Attribute werden dadurch in einen zeitlichen Zusammenhang gesetzt.
- Das Faktattribut Zeitschalter erhält den Wert 1 für jede gültige Produktstruktur. Dieses Attribut ist nur notwendig, falls das OLAP-System explizit ein Faktattribut zur Definition der Faktentabelle benötigt. Ansonsten kann der Wert 1 auch über die SQL-Funktion `Count()` ermittelt werden.

Abbildung 2.3.14 zeigt die ursprüngliche und die aus der Produktdimension abgeleitete Faktentabelle. Die Produktfaktentabelle bildet in jeder Periode die entsprechend gültige Struktur der Produktdimension ab.

FAKTEN

Jahr	Produkt-ID	Umsatz
...	...	...
2001	1022	700
2001	1023	500
2002	1022	1800
2002	1023	300
...	...	...

PRODUKTFAKTEN

Produkt-ID	Jahr	Name	Kategorie	Hersteller	Zeitschalter
...	...	...	...	...	...
1022	2001	Chnuschpi	Schokolade	Vander	1
1023	2001	N&N	Schokolade	Saturn	1
...	...	...	...	...	...
1022	2002	Chnuschpi	***Kassenprodukte***	Vander	1
1023	2002	N&N	Schokolade	Saturn	1
...	...	...	...	...	...

Abbildung 2.3.14: Beispiel der ursprünglichen und der neuen Faktentabelle

Anhand dieser Faktentabellen lässt sich nun die gewünschte Auswertung definieren. Die erste Abfrage ermittelt den Umsatz nach Produkt und Periode aus der ursprünglichen Faktentabelle (vgl. Abbildung 2.3.15).

		ZEIT	
		2001	**2002**
PRODUKT	**...**	...	...
	1022	700	1800
	1023	500	300
	...	...	...

Abbildung 2.3.15: Ermittlung des Umsatzes nach Produkt-ID und Jahr

Die zweite Abfrage bestimmt, welche Kombinationen aus Produkt-ID und Kategorie in jeder Periode gültig waren. Das Resultat wird in Abbildung 2.3.16 verschachtelt dargestellt, weil bei dieser Abfrage drei Auswertungskriterien (JAHR, PRODUKT-ID und KATEGORIE) involviert sind.

		JAHR	
PRODUKT-ID	**KATEGORIE**	**2001**	**2002**
...	**...**	...	...
1022	**Schokolade**	1	NULL
1023	**Schokolade**	1	1
...	**...**	...	...
1022	**Kassenprodukte**	NULL	1
1023	**Kassenprodukte**	NULL	NULL
...	**...**	...	...

Abbildung 2.3.16: Ermittlung der Gültigkeit nach Produkt-ID, Kategorie und Jahr

Die Nullwerte zeigen an, dass kein entsprechender Eintrag in der Produktfaktentabelle verfügbar ist und damit diese Kombination von PRODUKT-ID und KATEGORIE zum betreffenden Zeitpunkt nicht gültig war. Der Umsatz lässt sich nun in Abhängigkeit von der Kategorie und dem Jahr bestimmen, indem die beiden Abfrageresultate aus Abbildung 2.3.15 und 2.3.16 miteinander verrechnet werden:

```
[Gesuchter Wert] = [Zeitschalter] * XRef(Umsatz)
```
[97]

[97] *XRef* steht hier für eine Funktion, die den Wert eines Fakts aus einem anderen Würfel anhand der entsprechenden Dimensionswerten ermittelt. Die Benennung dieser Funktion ist produktspezifisch und wird zum Beispiel bei MS Analysis Services *LookupCube* genannt.

Die Werte der Zeitschalter (1 oder NULL) aus der zweiten Abfrage (Abbildung 2.3.16) werden mit den entsprechenden Umsätzen aus der ersten Abfrage (Abbildung 2.3.15) multipliziert. Das Resultat dieser Berechnung ist in Abbildung 2.3.17 dargestellt.

		JAHR	
PRODUKT-ID	**KATEGORIE**	**2001**	**2002**
...	**...**	...	...
1022	**Schokolade**	700	NULL
1023	**Schokolade**	500	300
...	**...**	...	...
1022	**Kassenprodukte**	NULL	1800
1023	**Kassenprodukte**	NULL	NULL
...	**...**	...	...

Abbildung 2.3.17: Ermittlung des Umsatzes nach Produkt-ID, Kategorie und Jahr

Die Aggregation der Umsätze auf die Ebene der Produktkategorie ergibt das gesuchte Ergebnis (vgl. Abbildung 2.3.18). Das Resultat entspricht der faktbezogenen historischen Sicht, weil jedem Fakt genau die Dimensionsebene zugeordnet wurde, die bei der Speicherung des Fakts gültig war.

		ZEIT	
		2001	**2002**
PRODUKT	**...**	...	...
	Schokolade	1200	300
	Kassenprodukte	NULL	1800
	...	...	...

Abbildung 2.3.18: Ermittlung des Umsatzes nach Kategorie und Jahr

Die aktuelle Sicht kann weiterhin mit der ursprünglichen Faktentabelle und den herkömmlichen Dimensionstabellen bestimmt werden.[98]

Beurteilung

Das Zeitschalterverfahren ermöglicht die Abbildung der faktbezogenen historischen Sicht, indem Dimensionsdaten in speziellen Faktentabellen zeitbezogen

[98] Voraussetzung: Dimensionen werden durch Überschreiben der vergangenen Daten und nicht durch Hinzufügen neuer Daten aktualisiert. Vgl. Slowly Changing Dimensions, S. 88

gespeichert und zur Laufzeit den ursprünglichen Fakten zugeordnet werden. Die Verrechnung von Daten aus verschiedenen Faktentabellen setzt die Unterstützung von Multicubes[99] voraus. Die Implementation dieser Methode ist daher nicht in jedem OLAP-System möglich.

Die Verwaltung zusätzlicher Faktentabellen ist aufwändig und stellt hohe Anforderungen an die Speicherressourcen. Zudem ist ein Modell mit mehreren Faktentabellen und zusätzlichen Verrechnungsfunktionen schlechter verständlich als ein konventionelles Sternschema. Die Generierung der faktbezogenen historischen Sicht muss deshalb in einer Anwendung soweit automatisiert sein, dass zur Abfragedefinition keine zusätzlichen Kenntnisse über die darunter liegenden Datenstrukturen erforderlich sind.

2.3.3 Methodenvergleich

Die vorgestellten Modellierungsmethoden bilden den Zeitbezug der Dimensionsdaten entweder direkt in den Dimensionstabellen oder indirekt über das Zeitstempelattribut der Faktentabelle ab. Zur ersten Gruppe gehören die temporal erweiterten Sternschemata. Zeitstempelattribute in konventionell oder rekursiv strukturierten Dimensionstabellen stellen den Zeitbezug aller darin enthaltenen Daten sicher. Die Modellierungsmethoden der zweiten Gruppe bilden die Zeitabhängigkeit der Dimensionsdaten nur partiell ab. Partiell bedeutet hier, dass sich temporale Abfragen gar nicht oder nur indirekt beantworten lassen, wenn zum gewünschten Bezugszeitpunkt keine entsprechenden Fakten vorhanden sind.

Die Beispielfrage *„Zu welcher Kategorie hat das Produkt Chnuschpi am 20.5.2000 gehört?“* ist aus dem temporal erweiterten Sternschema direkt ableitbar, weil alle Dimensionsdaten vom Zeitpunkt ihrer ersten Erfassung bis heute lückenlos durch Gültigkeitsintervalle zeitbezogen gespeichert sind. Diese Gültigkeitsinformation lässt sich in den anderen Modellen nicht *direkt* ermitteln, weil der Bezugszeitpunkt wie hier zum Beispiel auf einen verkaufsfreien Sonntag fallen könnte und das System dann über keine entsprechenden Fakten verfügt. Der Endbenutzer muss deshalb *indirekt* durch das Verändern der Bezugszeit herausfinden, zu welcher Kategorie das Produkt gehört hat.

Diese unvollständige Abbildung zeitabhängiger Dimensionsdaten ist in OLAP meist unproblematisch, weil die Zielgruppe multidimensionaler Systeme viel häufiger Fakten als Dimensionen in den Mittelpunkt ihrer Analyse stellen. Sollte dennoch ein bestimmtes Dimensionsattribut regelmässig in einem tempora-

[99] Vgl. Pendse 2001

len Kontext zu analysieren sein, könnte überprüft werden, inwiefern sich das entsprechende Attribut als Fakt modellieren lässt.

Im Gegensatz zu OLAP steht das zentrale Data Warehouse oder ein Data Mart als Datenlieferant an verschiedene Werkzeugklassen restriktiveren temporalen Rahmenbedingungen gegenüber. In diesen Datenbanksystemen müssen die veränderbaren Daten häufig vollständig zeitbezogen abgebildet werden. Diese Anforderung erfüllen in den präsentierten Ansätzen lediglich die temporal erweiterten Schemata.

Die Auswahl und Bewertung der Kriterien des in Abbildung 2.3.19 dargestellten Vergleichs gehen allerdings nicht von den Anforderungen eines zentralen Data Warehouses oder Data Marts aus, sondern wurden im Hinblick auf die Anwendung in einem OLAP-System festgelegt.

Zeichenerklärung

- ✓ Die Funktionalität wird unterstützt
- **?** Das Modell schliesst die Funktionalität nicht aus. Die Nutzung ist jedoch aufwändig und/oder wird von OLAP-Systemen nicht unterstützt
- **+** überdurchschnittliche Bewertung des Zeilenkriteriums
- **O** durchschnittliche Bewertung des Zeilenkriteriums
- **-** unterdurchschnittliche Bewertung des Zeilenkriteriums

Der Vergleich zeigt die von den Modellierungsmethoden unterstützte Funktionalität (✓/**?**) und bewertet Kriterien zur Benutzerfreundlichkeit und den System-/Wartungsanforderungen (**+**/**O**/**-**). Die folgenden Punkte erörtern die vorgenommene Bewertung, soweit diese nicht bereits aus der Besprechung der einzelnen Modellierungsmethoden hervorgehen.

Temporale Auswertungsformen

- Die dimensionsbezogene Sicht ist auf eine vollständige zeitbezogene Abbildung der Dimensionsdaten angewiesen. Sie wird in diesem Vergleich nur von den Zeitstempelmethoden unterstützt.
- Mit dem SCD Typ 2 ist es möglich, die aktuelle und die ursprüngliche Sicht zu definieren (zum Beispiel mit einer SQL-Abfrage). Die dazu notwendige Erstellung einer Zwischenabfrage ist jedoch aufwändig und wird in vielen OLAP-Systemen nicht unterstützt. Bei den Zeitstempelmethoden ist die Definition der ursprünglichen Sicht ebenfalls auf eine zusätzliche Abfrage angewiesen.

Kriterien	Zeitstempelmethode in konventionellen Dimensionen	Zeitstempelmethode in rekursiven Dimensionen	SCD Typ 1 - Überschreiben (konventionelles Vorgehen)	SCD Typ 2 - Hinzufügen	SCD Typ 3 - zusätzliches Attribut	Alternativer Fremdschlüssel	Zeitschalterverfahren
Temporale Auswertungsformen							
Aktuelle Sicht	✓	✓	✓	?	✓	✓	✓
Faktbezogene historische Sicht	✓	✓		✓		✓	✓
Dimensionsbezogene historische Sicht	✓	✓					
Ursprüngliche Sicht	?	?		?	✓		
Temporale Funktionalität							
Verschiedene Sichten in einer Abfrage	✓	✓			✓	✓	✓
Dimensionsdaten temporal analysierbar	✓	✓					
Temporale Differenzierung der Attribute							
Zeitbezogene Dimensionsstruktur		✓					
Benutzerfreundlichkeit							
Verständlichkeit des Modells	O	-	+	O	O	O	-
Abfrageeffizienz	O	-	+	+	+	+	-
Systemanforderungen / Wartung							
Implementierbarkeit in OLAP-Systemen	-	-	+	+	+	+	O
Speichereffizienz	O	O	+	O	+	-	-
Inkrementelle Aktualisierung	+	+	O	+	O	O	O
Komplexität der Aktualisierungsprozesse	O	-	+	+	+	-	O

Abbildung 2.3.19: Vergleich der Modellierungsmethoden

Temporale Funktionalität

- Das Kriterium *Verschiedene Sichten in einer Abfrage* zeigt an, ob die temporale Auswertungsform nur als Ganzes oder für jede Dimension individuell bestimmbar ist. SCD Typ 1 und 2 unterstützen jeweils lediglich eine Sicht. Die kombinierte Nutzung verschiedener Sichten in einer Abfrage ist deshalb bereits aufgrund der funktionellen Einschränkung nicht möglich.
- Auf die *temporale Differenzierung der Attribute* geht keiner der vorgestellten Ansätze ausreichend ein. Die Unterscheidung zwischen zeitabhängigen und zeitunabhängigen Attributen liesse sich wahrscheinlich am einfachsten beim Zeitschalterverfahren implementieren, indem bei der Definition der zusätzlichen Faktentabellen ausschliesslich zeitabhängige Dimensionsattribute berücksichtigt werden.

Benutzerfreundlichkeit

- Die *Verständlichkeit des Modells* ist umso grösser, je weniger es vom konventionellen Sternschema abweicht. Während rekursive oder als Fakten modellierte Dimensionen schwer interpretierbar sind, müssen sich die Anwender bei den anderen temporalen Modellierungsvarianten gegenüber dem konventionellen Vorgehen mit zusätzlichen Attributen, Tupeln und/oder Tabellen auseinandersetzen.
- Die Modellierungsmethoden, die zum Kriterium *Abfrageeffizienz* überdurchschnittlich bewertet wurden, erlauben die direkte Zuordnung der Dimensionsdaten zu den Fakten und unterstützen die Verwendung präaggregierter Daten. Die Zeitstempelmethode in rekursiven Dimensionen und das Zeitschalterverfahren benötigen hingegen mehrere Schritte bei der Zuordnung der gewünschten Dimensionsdaten zu den Fakten.

Systemanforderungen / Wartung

- Während Werkzeuge im Bereich Wartung und Modellierung Zeitstempel in Dimensionstabellen teilweise bereits automatisch pflegen und damit deren Einsatz in einem Data Mart vorsehen[100], können die meisten kommerziell erhältlichen OLAP-Werkzeuge diese zusätzliche Information nicht adäquat verarbeiten. Die *Implementierbarkeit* des Zeitschalterverfahrens hängt dagegen von der Unterstützung des Multicube-Verfahrens im jeweiligen OLAP-Systems ab.

---

[100] Vgl. zum Beispiel DecisionStream von Cognos.

- Die Bewertung der *Speichereffizienz* hängt nicht nur von der gewählten Modellierungsmethode ab, sondern wird auch von den Eigenschaften jeder individuellen Datenstruktur beeinflusst (Anzahl Attribute, Tupel, Tabellen etc.). Deshalb sind die Bewertungen hier als Tendenzaussagen und nicht als absolut gültig zu verstehen. Die Speichereffizienz wurde unterdurchschnittlich beurteilt, wenn das Modell zusätzliche Tupel oder Attribute in der Fakttabelle (Alternative Fremdschlüssel) oder sogar weitere Fakttabellen benötigt (Zeitschalterverfahren). Verfahren, die gegenüber dem konventionellen Vorgehen (SCD Typ 1) zusätzliche Tupel in den Dimensionstabellen abbilden müssen, wurden als durchschnittlich eingestuft.
- Das OLAP-System muss neben den Detaildaten auch Aggregate periodisch auf den neusten Stand bringen. Aggregate lassen sich im Gegensatz zu den Detaildaten nicht in allen Modellen inkrementell aktualisieren. Wenn das gewählte Modellierungskonzept Veränderungen der Dimensionsdaten durch das Überschreiben vergangener Werte abbildet, müssen alle Aggregate, die sich auf den überschriebenen Wert beziehen, ebenfalls neu berechnet werden. Diese Neuberechnung ist aufwändiger als die *inkrementelle Aktualisierung* und wird deshalb schlechter bewertet.
- Die *Komplexität der Aktualisierungsprozesse* ist im SCD-Konzept am einfachsten. Bestehende Daten werden durch neue ersetzt und/oder überschrieben. Beim Zeitschalterverfahren müssen zusätzliche Tabellen und bei der einfacheren Zeitstempelmethode zusätzliche Dimensionsattribute gewartet werden. Dagegen ist der Abgleich und die Transformation der Quelldaten mit den rekursiv strukturierten Dimensionen aufwändiger, weil neben der Wartung der Zeitstempelattribute die Information eines konventionellen Tupels auf mehrere rekursiv verknüpfte Tupel aufzuteilen ist. Das „Alternative Fremdschlüssel“-Modell muss im Gegensatz zu allen anderen Verfahren bei jeder Veränderung der Dimensionsdaten auch Werte in der Faktentabelle anpassen. Der dazu notwendige Aktualisierungsprozess ist komplex und zeitintensiv.

Teil III

Das aggregierte Faktenmodell

Der Vergleich im vorhergehenden Kapitel hat gezeigt, dass keine der vorgestellten Methoden allen Ansprüchen vollständig genügt:

Die *Zeitstempelmethoden* erfüllen die temporalen Anforderungen am besten, werden aber von vielen OLAP-Systemen nicht adäquat unterstützt. Im zentralen Data Warehouse und in den fachbereichsspezifischen Data Marts kann die Anwendung dieser Modellierungsmethoden dennoch vorteilhaft sein, weil der Zeitbezug der Dimensionsdaten dadurch allen darauf basierenden analytischen Anwendungen zur Verfügung steht.

Die Methoden im *Slowly Changing Dimensions*-Konzept sind im Gegensatz zu den Zeitstempelmethoden in den meisten OLAP-Systemen implementierbar. Die drei Modellierungsvarianten ermöglichen zusammen die Nutzung der häufigsten temporalen Auswertungsformen, sind jedoch in einem Modell nicht beliebig kombinierbar. Die Darstellung der aktuellen und der faktbezogenen historischen Sicht erfordert zum Beispiel die Definition von zwei getrennten Datenwürfeln, weil das OLAP-System die Dimensionsdaten entweder durch Überschreiben oder durch Hinzufügen neuer Werte aktualisieren kann.

Das *alternative Fremdschlüssel*-Modell unterstützt die aktuelle und die faktbezogene historische Sicht in einem einzigen Datenwürfel und lässt sich in vielen OLAP-Systemen nutzen. Wenn jedoch Dimensionsdatenänderungen abzubilden sind, ist der Wartungsaufwand bei dieser Methode besonders hoch, weil der Aktualisierungsprozess im Gegensatz zu den anderen Verfahren auch bestehende Daten in der Faktentabelle anpassen muss.

Das *Zeitschalterverfahren* stellt den komplexesten Ansatz zur Abbildung der faktbezogenen historischen Sicht dar. Die Implementation setzt eine spezielle OLAP-Technologie voraus und ist deshalb nicht in allen Systemen möglich.

Die untersuchten Verfahren zur Unterstützung verschiedener temporaler Auswertungsformen haben gezeigt, dass die notwendigen Erweiterungen häufig zu Verschlechterungen in anwendungskritischen Bereichen führen wie etwa bei der Modellverständlichkeit oder beim Wartungsaufwand. Idealerweise fügt sich die zusätzliche temporale Funktionalität so in ein OLAP-System ein, dass die Anwender ihre Analysen in gewohnter Weise durchführen können und sich die dahinterstehenden Daten in der zur Verfügung stehenden Zeit warten lassen.

Das Kapitel spezifiziert zunächst die Funktionsweise des aggregierten Faktenmodells und zeigt anschliessend einen konkreten Implementationsvorschlag des Modellansatzes.

3.1 Modellansatz

Das aggregierte Faktenmodell erlaubt die Nutzung der *aktuellen* und der *faktbezogenen historischen Sicht*. Es gehört dabei zu den Modellierungsansätzen, die den Zeitbezug der Dimensionsdaten *indirekt* über das Zeitstempelattribut der Faktentabelle abbilden.[101] Das aggregierte Faktenmodell wurde so benannt, weil es eine neue Klasse aggregierter Fakten zur Abbildung der faktbezogenen historischen Sicht nutzt. Im Folgenden wird diese Aggregatsklasse zunächst eingeführt und anschliessend die Funktionsweise des aggregierten Faktenmodells erläutert.

3.1.1 Einführung temporaler Aggregate

Viele OLAP-Systeme unterstützen die Definition von Aggregaten zur Verbesserung der Speichereffizienz.[102] Die zusammengefassten Werte werden im Voraus berechnet und in der Datenbank abgespeichert. Zur Laufzeit greift das System auf die Aggregate zu und spart damit weiteren Berechnungsaufwand. Die Anwender müssen dabei keine Kenntnisse über die Aggregate haben, weil das System bei Vorhandensein autonom auf sie zugreift.

Die zur Abfrageeffizienzsteigerung eingesetzten Aggregate stellen die erste, hier konventionell genannte Klasse von Aggregaten dar. Wie beim Vergleich der temporalen Modellierungsmethoden bereits erwähnt wurde, lassen sich ▸ *konventionelle Aggregate* in einigen Modellen nicht inkrementell, sondern nur vollständig neu berechnen. Diese Einschränkung im periodischen Aktualisierungsprozess gilt, falls Dimensionsdatenänderungen durch einen überschreibenden Zugriff abzubilden sind. Das folgende Beispiel illustriert diese Problematik.

Beispiel: Aktualisierung konventioneller Aggregatstabellen

In diesem Beispiel ändert sich die Produktkategorie des Produkts 1022 von Schokolade zu Kassenprodukte. Werte in *kursiver* Schrift beziehen sich auf das Produkt 1022. ***Fett-kursive*** Schrift hebt die Änderungen im Jahr 2002 gegenüber den Werten aus dem Jahr 2001 hervor. Abbildung 3.1.1 zeigt die Dimensions- und Faktentabelle vor der Dimensionsdatenänderung. Zusätzlich sind in einer weiteren Tabelle (AGGREGATE) aggregierte Fakten gespeichert.

---

[101] Vgl. Kapitel 2.3.3: S. 97
[102] Vgl. Kapitel 1.3.3: S. 38

PRODUKTDIMENSION

Produkt-ID	Name	Kategorie	Hersteller
...	...	...	...
1022	*Chnuschpi*	*Schokolade*	*Vander*
1023	N&N	Schokolade	Saturn
1024	Quatrodent	Kassenprodukte	Pfazer
...	...	...	...

FAKTENTABELLE

Jahr	Produkt-ID	Umsatz
...	...	...
2001	*1022*	*700*
2001	1023	1500
2001	1024	600
...	...	...

AGGREGATE

Jahr	Kategorie	Umsatz
...	...	...
2001	Schokolade	2200
2001	Kassenprodukte	600
...	...	...

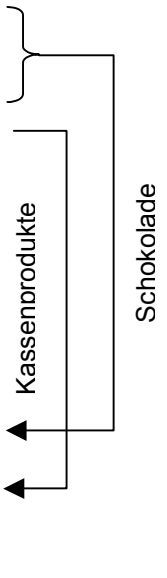


Abbildung 3.1.1: Daten vor der Dimensionsdatenänderung (2001)

In Aggregatstabellen werden die Fakten über ein oder mehrere Dimensionskriterien verdichtet gespeichert.[103] In diesem Beispiel ist der Umsatz in der Tabelle AGGREGATE nicht mehr zu jedem Produkt, sondern nur noch zu jeder Produktkategorie aufgeführt. Umsätze von Produkten, die zur selben Kategorie gehören, werden mit der gewünschten Aggregatsfunktion zusammengefasst. Das aufgeführte Beispiel verwendet die Summenfunktion. Die Definition weiterer Aggregatstabellen ist zwar möglich, jedoch für die Illustration des Beispiels nicht notwendig.

[103] Agarwal et al. 1999: S. 361

Im Jahr 2002 ändert sich die Produktkategorie des Produkts 1022 von Schokolade zu Kassenprodukte. Abbildung 3.1.2 zeigt die veränderte Dimensionstabelle und die Faktentabelle am Ende des Jahrs 2002.

PRODUKTDIMENSION

Produkt-ID	Name	Kategorie	Hersteller
...	...	...	...
1022	*Chnuschpi*	***Kassenprodukte***	*Vander*
1023	N&N	Schokolade	Saturn
1024	Quatrodent	Kassenprodukte	Pfazer
...	...	...	...

FAKTENTABELLE

Jahr	Produkt-ID	Umsatz
...	...	...
2001	*1022*	*700*
2001	1023	1500
2001	1024	600
...	...	...
2002	*1022*	*1500*
2002	1023	1500
2002	1024	500
...	...	...

Abbildung 3.1.2: Faktentabelle nach der Dimensionsdatenänderung (2002)

Aus den Detaildaten der Faktentabelle kann nun berechnet werden, wie hoch der Umsatz für jede Produktkategorie aus der im Jahr 2002 gültigen Sicht ist. Abbildung 3.1.3 zeigt dieses Abfrageresultat.

		ZEIT	
		2001	**2002**
PRODUKT	**Schokolade**	1500	1500
	Kassenprodukte	1300	2000
	...	...	...

Abbildung 3.1.3: Abfrageresultat aus der Faktentabelle ermittelt

Aus der Aggregatstabelle müssen sich dieselben Resultate ableiten lassen wie aus der detaillierten Faktentabelle. Wird die Aggregatstabelle inkrementell aktualisiert, dann verrechnet das System nur die neuen Faktwerte mit den bereits bestehenden Werten. Dadurch wird die vergangene Zuordnung der Dimensionswerte (1022 gehört im Jahr 2001 zur Kategorie Schokolade) mit der aktuellen (1022 gehört im Jahr 2002 zur Kategorie Kassenprodukte) Zuordnung vermischt. Abbildung 3.1.4 zeigt den inkrementellen Aktualisierungsprozess der Aggregatstabelle.

AGGREGATE (t=2001)

Jahr	Kategorie	Umsatz
...	...	...
2001	Schokolade	2200
2001	Kassenprodukte	600
...	...	...

Neu hinzugekommene aggregierte Werte im Jahr 2002

Jahr	Kategorie	Umsatz
...	...	...
2002	Schokolade	1500
2002	Kassenprodukte	2000
...	...	...

AGGREGATE (t=2002)

Jahr	Kategorie	Umsatz
...	...	...
2001	Schokolade	2200
2001	Kassenprodukte	600
...	...	...
2002	Schokolade	1500
2002	Kassenprodukte	2000
...	...	...

Abbildung 3.1.4: Inkrementeller Aktualisierungsprozess der Aggregatstabelle

Das aus der inkrementell aktualisierten Aggregatstabelle ermittelte Abfrageresultat (vgl. Abbildung 3.1.5) stimmt nicht mit dem aus den Detaildaten abgeleiteten Ergebnis überein (vgl. Abbildung 3.1.3).

		ZEIT	
		2001	**2002**
PRODUKT	**Schokolade**	***2200***	1500
	Kassenprodukte	***600***	2000
	...	...	...

Abbildung 3.1.5: Abfrageresultat aus der Aggregatstabelle ermittelt

Während die Werte für das Jahr 2002 bei beiden Abfrageresultaten identisch sind, wird der Umsatz des Produkts 1022 im Jahr 2001 unterschiedlichen Kategorien zugeordnet. Die inkrementelle Aktualisierung bewirkt, dass die Aggregatstabelle lediglich die *neuen* Fakten in Abhängigkeit der veränderten Dimensionsdaten abbilden. Diese Darstellung entspricht der *faktbezogenen historischen Sicht*, weil die aggregierten Fakten immer in Abhängigkeit der zur Erfassungszeit gültigen Dimensionsdaten gespeichert sind (der Umsatz des Produkts 1022 wird im Jahr 2001 der Kategorie Schokolade, im Jahr 2002 der Kategorie Kassenprodukte zugeordnet). Berechnet das System hingegen die Aggregate bei jeder Dimensionsdatenänderung neu aus den Detaildaten, dann speichert es *alle* Fakten in Abhängigkeit der veränderten Dimensionsdaten. Diese Darstellung entspricht der *aktuellen Sicht*, weil die aggregierten Fakten immer in Abhängigkeit zu den aktuell gültigen Dimensionsdaten abgebildet sind (der Umsatz des Produkts 1022 wird in beiden Jahren der Kategorie Kassenprodukte zugeordnet).

In diesem Beispiel wurden die Dimensionsdaten stets durch Überschreiben aktualisiert. Die *detaillierten* Fakten lassen sich deshalb nur in Abhängigkeit zu den aktuellen Dimensionsdaten analysieren (aktuelle Sicht). Damit die *aggregierten* Fakten ebenfalls den gleichen temporalen Sachverhalt darstellen, müssen sie bei Dimensionsdatenänderungen vollständig neu berechnet werden.

Weil die konventionellen Aggregate temporal konsistent zu den detaillierten Fakten sein müssen, führt das aggregierte Faktenmodell eine neue Aggregatsklasse ein, mit deren Hilfe OLAP-Systeme in der Lage sein sollen, zusätzlich die faktbezogene historische Sicht abzubilden. Diese Aggregate werden im Weiteren ▸ *temporale* Aggregate genannt, weil sie in erster Linie nicht wie *konventionelle* Aggregate zur effizienteren Abfrageverarbeitung, sondern zur Abbildung einer weiteren temporalen Auswertungsform bestimmt sind.

Von den temporalen Aggregaten ist bisher lediglich bekannt, dass sie sich auch dann inkrementell aktualisieren lassen, wenn Dimensionsdatenänderungen vorliegen. Die folgenden Abschnitte spezifizieren deshalb die Funktionsweise des aggregierten Faktenmodells und befasst sich dabei insbesondere mit der Definition, Aktualisierung und Analyse der temporalen Aggregate.

3.1.2 Anwendung der temporalen Aggregate

Das aggregierte Faktenmodell unterstützt die aktuelle *und* die faktbezogene historische Sicht. Die Ausführungen über die Funktionsweise beschränken sich im Wesentlichen auf die Darstellung der faktbezogenen historischen Sicht. Die Abbildung der aktuellen Sicht beruht vollständig auf bestehender OLAP-Technologie und wurde bereits im Zusammenhang alternativer Modellierungsansätze erwähnt.[104]

Aktuelle Sicht

Nicht mehr gültige Dimensionsdaten werden im aggregierten Faktenmodell mit neuen Werten überschrieben. Sämtliche Fakten sind dadurch immer in Abhängigkeit der aktuellen Dimensionsdaten gespeichert. Damit die konventionellen Aggregate im selben temporalen Zusammenhang stehen wie die Detaildaten, müssen die von Dimensionsdatenänderungen betroffenen konventionellen Aggregate neu berechnet werden. Das Vorgehen zur Abbildung der aktuellen Sicht entspricht dem bereits bekannten Konzept des Slowly Changing Dimensions Typ 1.[105]

Faktbezogene historische Sicht

Definition der temporalen Aggregatstabellen

Wie das Beispiel zur inkrementellen Aktualisierung gezeigt hat, wird die faktbezogene historische Sicht mit Hilfe temporaler Aggregate abgebildet. Die Aggregatstabelle hat bisher lediglich das Attribut Kategorie der Dimension PRODUKT in einen temporalen Kontext gesetzt. Andere zeitabhängige Attribute wie Name oder Hersteller wurden hingegen vernachlässigt. Zudem haben die Dimen-

[104] Vgl. Kapitel 2.3.2: S. 87
[105] Vgl. Kapitel 2.3.2: S. 88

sionen KUNDE, GEOGRAFIE und ZEIT aus dem Bespielschema (vgl. Abbildung 2.3.1) gefehlt.

Abbildung 3.1.6 zeigt die Faktentabelle und zwei temporale Aggregatstabellen. Die erste Aggregatstabelle stellt die Fakten Umsatz, Kosten und Gewinn in Abhängigkeit der Attribute Kategorie und Datum sowie der restlichen Dimensionsschlüsselwerte Kunden-ID und Geografie-ID dar. Die zweite Aggregatstabelle enthält neben den Faktattributen das zeitabhängige Attribut Bonität aus der Kundendimension, das Zeitstempelattribut Datum sowie die verbleibenden Dimensionsschlüsselwerte Produkt-ID und Geografie-ID.

FAKTENTABELLE

Datum	Kunden-ID	Produkt-ID	Geografie-ID	Umsatz	Kosten	Gewinn
01.02.2002	4711	1022	20	50	20	30
01.02.2002	4711	4055	20	30	15	15
...	...	...	...	...	...	...

AGGREGATSTABELLE_1

Datum	Kunden-ID	Kategorie	Geografie-ID	Umsatz	Kosten	Gewinn
01.02.2002	4711	Schokolade	20	120	70	50
01.02.2002	4711	Backwaren	20	40	20	20
...	...	...	...	...	...	...

AGGREGATSTABELLE_2

Datum	Bonität	Produkt-ID	Geografie-ID	Umsatz	Kosten	Gewinn
01.02.2002	sehr gut	1022	20	350	140	210
01.02.2002	gut	4055	20	300	150	150
...	...	...	...	...	...	...

Abbildung 3.1.6: Faktentabelle und zwei Aggregatstabellen

Anstelle des Fremdschlüsselattributs Produkt-ID der Faktentabelle steht in der ersten Aggregatstabelle das zeitabhängige Dimensionsattribut Kategorie. Dadurch enthält die Aggregatstabelle keine detaillierten Faktwerte mehr für jedes Produkt, sondern nur noch zusammengefasste Werte für jede Produktkategorie. Auch die zweite Aggregatstabelle weicht in einem Attribut und den Faktwerten von der Faktentabelle ab. Anstelle des Attributs Kunden-ID steht das zeitabhängige Dimensionsattribut Bonität. In weiteren Aggregatstabellen lassen sich die restlichen zeitabhängigen Dimensionsattribute auf dieselbe Art integrieren. Bei der Entwicklung der Aggregatstabellen sollten allerdings nur zeitabhängige

Dimensionsattribute im engeren Sinn[106] berücksichtigt werden. Das Einbeziehen zeitunabhängiger Attribute würde zu temporalen Aggregatstabellen führen, deren Inhalte identisch mit ihren konventionellen Pendants sind. Die Integration zeitabhängiger Attribute im weiteren Sinn ist ebenfalls überflüssig, weil hier die temporale Betrachtung im Voraus als nicht relevant eingestuft wurde. Da jedes zeitabhängige Dimensionsattribut die Definition weiterer Aggregatstabellen erfordert, ist die Notwendigkeit zur temporalen Abbildung gegenüber den zusätzlichen Wartungs- und Speicherkosten sorgfältig abzuwägen.

Abbildung 3.1.6 zeigt zwei Aggregatstabellen, die jeweils ein zeitabhängiges Attribut enthalten (Bonität *oder* Kategorie). Die Beschränkung auf ein zeitabhängiges Dimensionsattribut ist nicht ausreichend, um alle möglichen Abfragen zu beantworten. Deshalb müssen weitere Aggregatstabellen definiert werden, in welchen sich zeitabhängige Dimensionsattribute verschiedener Dimensionen kombinieren lassen. Abbildung 3.1.7 zeigt eine Aggregatstabelle, die zwei zeitabhängige Dimensionsattribute enthält (Bonität *und* Kategorie).

AGGREGATSTABELLE_3

Datum	Bonität	Kategorie	Geografie-ID	Umsatz	Kosten	Gewinn
01.02.2002	sehr gut	Schokolade	20	350	140	210
01.02.2002	gut	Backwaren	20	300	150	150
...	...	...	...	...	...	...

Abbildung 3.1.7. Tabelle mit zwei zeitabhängigen Dimensionsattributen

Zur vollständigen Abbildung mehrdimensionaler Abfragekonstellationen sind die in Abbildung 3.1.8 dargestellten Kombinationen von Dimensionsschlüsseln und zeitabhängigen Dimensionsattributen in Aggregatstabellen zu definieren. Die fett markierte Kombination stellt den Primärschlüssel der Aggregatstabelle aus Abbildung 3.1.7 dar.

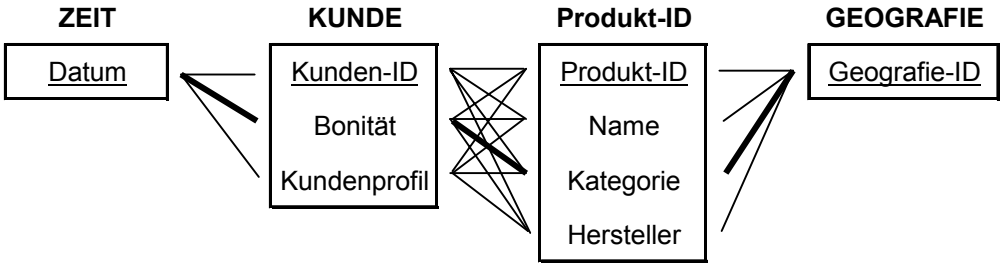


Abbildung 3.1.8: Alle Kombinationen zur Definition der Aggregatstabellen

[106] Vgl. Kapitel 2.2.2: S. 63

Kombinationen zeitabhängiger Attribute, die aus der gleichen Dimension stammen, können auf diese Weise nicht erstellt werden, weil jede Aggregatstabelle in der hier präsentierten Form jeweils nur ein Attribut pro Dimension enthalten kann.

Aktualisierung der temporalen Aggregatstabellen

Nach der Definition der benötigten Aggregatstabellen müssen die detaillierten Objektdaten zusammengefasst und in die Aggregatstabellen geladen werden. Als Datenquelle stehen einerseits die bereits in die OLAP-Datenbank integrierten Fakten- und Dimensionstabellen, andererseits die ursprüngliche Quelldatenbank zur Verfügung (zum Beispiel ein Data Mart oder das zentrale Data Warehouse). Falls die Dimensionsdaten im Quellschema wie etwa im temporal erweiterten Sternschema[107] zeitbezogen gespeichert sind und sich daraus die faktbezogene historische Sicht ableiten lässt, kann das System die temporalen Aggregatstabellen nicht nur inkrementell, sondern auch vollständig aktualisieren.

Abfrage der temporalen Aggregatstabellen

Die Abfragedefinition unterscheidet sich mit Ausnahme der Wahl der gewünschten temporalen Auswertungsform nicht von einer konventionellen OLAP-Abfrage. Der Endbenutzer wählt zunächst die aktuelle oder die faktbezogene historische Sicht und stellt seine Abfrage anschliessend in gewohnter Weise zusammen. Die OLAP-Anwendung bestimmt anhand der gewünschten temporalen Auswertungsform und der Abfrageparameter die richtigen Tabellen und führt die Abfrage aus.

Damit sich zeitbezogene, nicht zeitbezogene und nicht zeitbezogen abzubildende Dimensionsattribute gemeinsam in einer Abfrage verwenden lassen, soll das System in der Lage sein, temporale Aggregatstabellen mit den Dimensionstabellen zu verknüpfen. Abbildung 3.1.9 zeigt diese Einbindung der Aggregatstabellen in das bestehende Sternschema anhand eines Beispiels.

[107] Vgl. Kapitel 2.3.1: S. 71

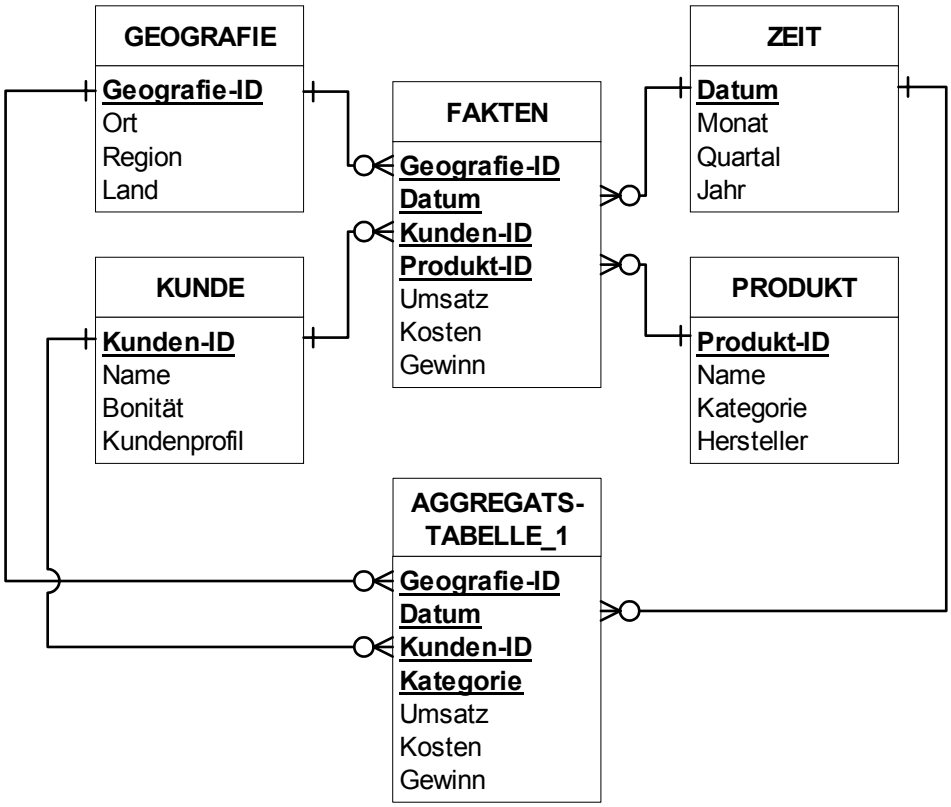


Abbildung 3.1.9: Temporale Aggregatstabelle im Sternschema

Der Endbenutzer kann hier zum Beispiel den Umsatz abhängig von der Produktkategorie (zeitabhängig), dem Kundennamen (nicht zeitabhängig) und dem Jahr ermitteln. Verwendet die OLAP-Anwendung dazu die Tabellen AGGREGATSTABELLE_1, KUNDE und ZEIT, dann werden die Fakten in der *faktbezogenen historischen Sicht* der Produktkategorie zugeordnet. Verwendet sie hingegen die Tabellen FAKTEN, KUNDE, PRODUKT und ZEIT, dann werden alle in der Abfrage genutzten Dimensionsattribute gemäss der *aktuellen Sicht* den Fakten zugeordnet.

Prozessablauf

Abbildung 3.1.10 fasst den Prozessablauf von der Definition der zeitabhängigen Dimensionsattribute bis zur Rückgabe eines in der faktbezogenen historischen Sicht erstellten Abfrageresultats zusammen. Das aggregierte Faktenmodell erweitert ein bestehendes OLAP-System, bei dem Fakten, Dimensionen und konventionelle Aggregate bereits vorhanden sind.[108]

[108] Vgl. Kapitel 1.3.3: S. 36

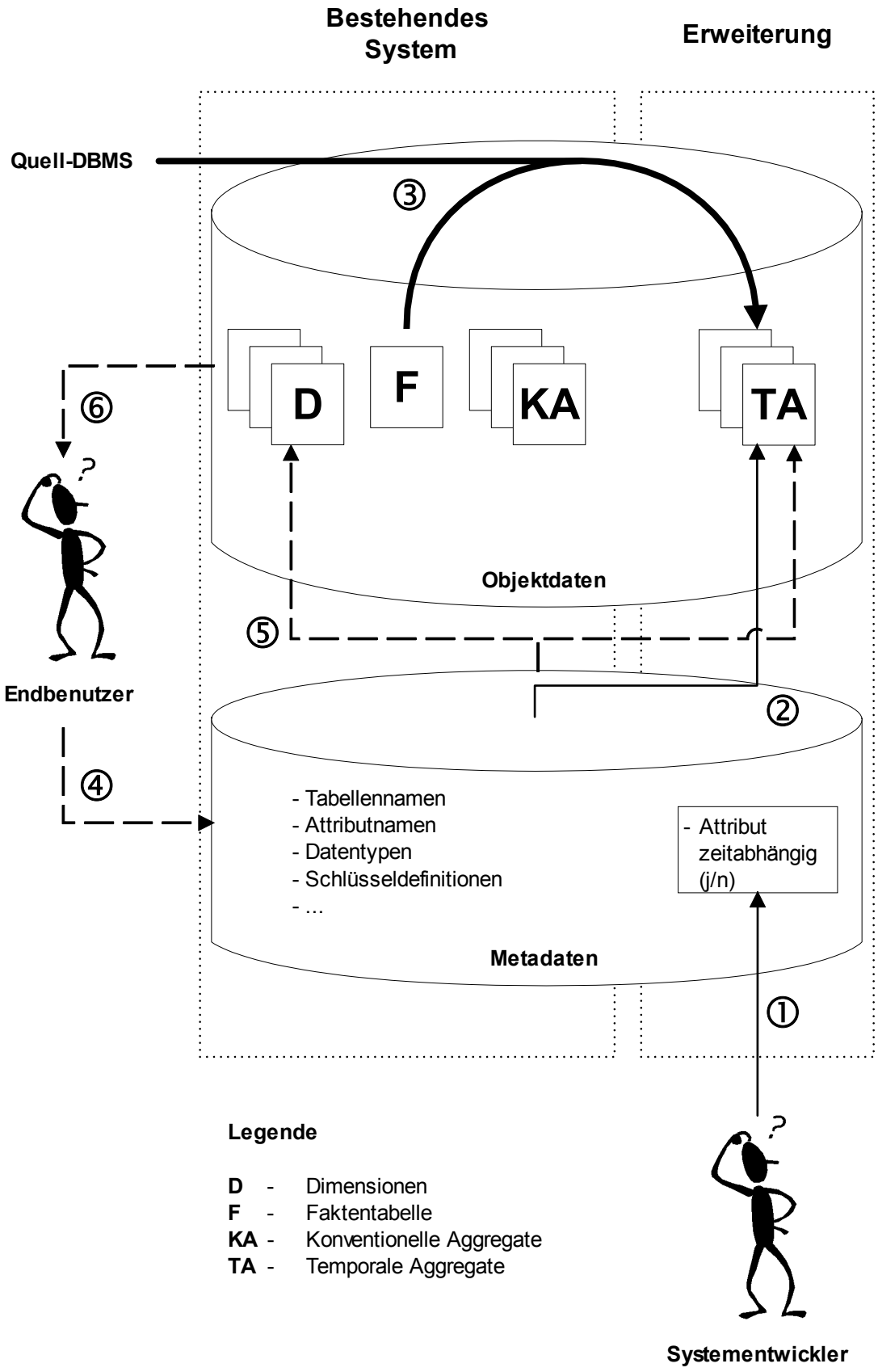


Abbildung 3.1.10: Ablaufprozess im aggregierten Faktenmodell

Ablauf der Modelldefinition

1. Der Systementwickler identifiziert die zeitabhängigen nach den Anforderungen der Endbenutzer. Der temporale Status aller Dimensionsattribute wird in der Metadatenbank gespeichert.
2. Das OLAP-System definiert anhand der Metadaten die benötigten temporalen Aggregatstabellen.
3. Das OLAP-System berechnet die verdichteten Daten entweder indirekt aus den bereits bestehenden Fakten- und Dimensionstabellen oder direkt aus den Quelldaten und fügt diese in die entsprechenden Aggregatstabellen ein.

Ablauf des Abfrageprozesses

4. Ein Endbenutzer definiert eine multidimensionale Abfrage, deren Resultat in der faktbezogenen historischen Sicht oder der aktuellen Sicht dargestellt sein soll.
5. Das OLAP-System bestimmt anhand der Abfrageparameter und der Metadaten die richtigen Tabellen und führt die Abfrage aus.
6. Der Anwender erhält das gewünschte Abfrageresultat.

3.1.3 Grenzen des aggregierten Faktenmodells

Fehlende Unterstützung alternativer Hierarchiepfade

Die Primärschlüssel der temporalen Aggregatstabellen enthalten höchstens ein zeitabhängiges Attribut pro Dimension. Endbenutzer können deshalb nie mehrere zeitabhängige Attribute aus derselben Dimension gleichzeitig in einer Abfrage verwenden. Diese Einschränkung ist besonders bei Dimensionsattributen aus alternativen Hierarchieebenen problematisch, weil deren Verknüpfung in einer Abfrage eher benötigt wird als die kombinierte Verwendung von Attributen aus über- oder untergeordneten Hierarchieebenen. Alternative Hierarchieebenen sind zum Beispiel die Produktkategorie und der Hersteller eines Produkts, denn ein Hersteller lässt sich nicht eindeutig zu einer Kategorie und eine Kategorie nicht eindeutig einem Hersteller zuordnen. Abbildung 3.1.11 zeigt eine Dimensionsstruktur ohne (GEOGRAFIE) und eine mit alternativen Hierarchieebenen (PRODUKT).

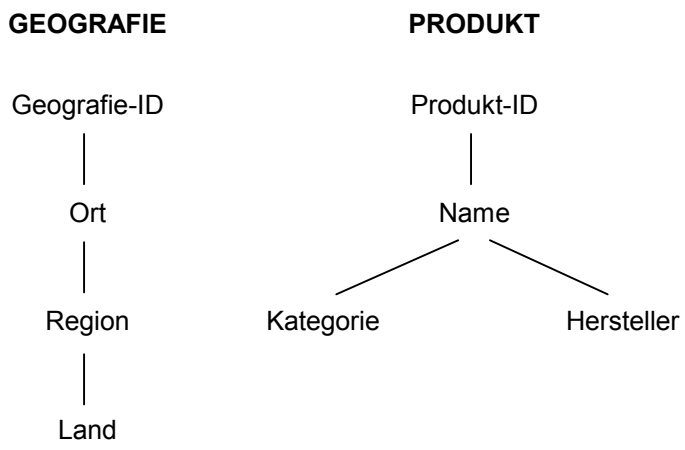


Abbildung 3.1.11: Dimension ohne und mit alternativen Hierarchieebenen

Beispiel 1 (zeitabhängige Dimensionsattribute aus alternativen Hierarchieebenen): „Zeige den Umsatz des Herstellers *Vander*, aber nur für die Kategorie *Schokolade*." Diese Abfrage kann das aggregierte Faktenmodell nicht in der faktbezogenen historischen Sicht darstellen.

Beispiel 2 (zeitabhängige Dimensionsattribute aus übergeordneten Hierarchieebenen): „Zeige den Umsatz des Herstellers *Vander*, aber nur für das Produkt *Chnuschpi*." Diese Abfrage ist auch nicht möglich. Im Gegensatz zur Abfrage im ersten Beispiel ist die Einschränkung jedoch weniger relevant, weil die Endbenutzer auch direkt nach dem Umsatz des Produkts *Chnuschpi* fragen könnten. Dies setzt allerdings voraus, dass die Dimensionselemente eindeutig sind und zum Beispiel kein Hersteller ausser *Vander* ein Produkt namens Chnuschpi herstellt. Eindeutige Dimensionswerte lassen sich zum Beispiel durch die Kombination von unter- und übergeordneten Dimensionswerten bei der Datentransformation im ETL-Prozess erzielen. Der Produktname enthielte dann anstelle des Werts *Chnuschpi* den Wert *Chnuschpi_Vander*.

Die Problematik alternativer Hierarchieebenen lässt sich vermeiden, wenn jeder Hierarchiepfad in einer eigenen Dimension modelliert wird. Diese Vorgehensweise führt allerdings zu weiterer Redundanz und einem entsprechend höheren Speicherverbrauch.

Unvollständige Historisierung der Dimensionsdaten

Das aggregierte Faktenmodell bildet den Zeitbezug der Dimensionsdaten indirekt über das Zeitstempelattribut der Aggregatstabellen ab. Weil die Dimensionsdaten keinen eigenen temporalen Bezug aufweisen und die Dimensionswerte durch einen überschreibenden Zugriff aktualisiert werden, lassen sich Veränderungen der Dimensionsdaten nicht historisch betrachten.

Zusammenfassung

Das aggregierte Faktenmodell wendet zur Abbildung der *aktuellen Sicht* dasselbe Vorgehen wie beim SCD Typ 1 an. Der periodische Aktualisierungsprozess überschreibt alle nicht mehr gültigen Dimensionsdaten mit neuen Werten und ermöglicht damit, dass die Anwender die Fakten abhängig von den aktuellen Dimensionsdaten analysieren können.

OLAP-Systeme verbessern die Abfrageeffizienz, indem sie aggregierte Fakten in getrennten Tabellen verwalten. Diese vorberechneten Abfrageresultate werden hier als *konventionelle Aggregate* bezeichnet. In Aggregatstabellen können Fakten nicht nur zeitbezogen, sondern auch abhängig von zeitbezogenen Dimensionsattributen gespeichert sein. Diese Datenstruktur ermöglicht die Abbildung jeweils einer der vier temporalen Auswertungsformen. Allerdings ist es nicht möglich, die temporale Auswertungsform in konventionellen Aggregatstabellen frei zu wählen, denn die Fakten müssen zur Sicherstellung der Konsistenz sowohl in konventionellen Aggregatstabellen als auch in der detaillierten Faktentabelle denselben dimensionalen Kontext aufweisen.

Das aggregierte Faktenmodell führt deshalb eine neue Aggregatsklasse ein, mit deren Hilfe sich die Fakten unabhängig vom dimensionalen Kontext der detaillierten Faktentabelle in einer zusätzlichen temporalen Auswertungsform abbilden lassen. In diesen *temporalen Aggregatstabellen* werden die Fakten im Gegensatz zur Faktentabelle nicht gemäss der aktuellen, sondern entsprechend der *faktbezogenen historischen Sicht* abgespeichert. Damit unterstützt das aggregierte Faktenmodell die aktuelle *und* die faktbezogene historische Sicht.

Die bei den temporalen Aggregatstabellen anwendbare Aktualisierungsmethode ist abhängig von der temporalen Datenhaltung der Quelldatenbank. Das aggregierte Faktenmodell kann die temporalen Aggregatstabellen vollständig oder inkrementell aktualisieren, wenn die Dimensionsdaten in der Quelldatenbank zeitbezogen abgespeichert sind. Andernfalls steht nur die inkrementelle Aktualisierung zur Verfügung. Der inkrementelle Aktualisierungsprozess stellt in jedem Fall sicher, dass ausschliesslich neue Fakten im veränderten dimensionalen Kontext abgebildet werden.

Die OLAP-Anwendung ermittelt die zur Beantwortung der Abfrage benötigten Tabellen automatisch anhand der Abfrageparameter und der gewünschten temporalen Auswertungsform. Die Abfragedefinition unterscheidet sich aus der Sicht der Endbenutzer deshalb lediglich in der zusätzlichen Möglichkeit, für jede Dimension die gewünschte temporale Auswertungsform zu bestimmen.

Zu den zentralen Nachteilen des aggregierten Faktenmodells gehört die eingeschränkte Nutzung zeitabhängiger Dimensionsattribute. Da die temporalen Aggregatstabellen aus jeder Dimensionstabelle jeweils höchstens ein zeitabhängiges Attribut enthalten, ist es nicht möglich, mehrere zeitabhängige Attribute aus derselben Dimension zusammen in einer Abfrage zu verwenden. Diese Einschränkung ist besonders bei zeitabhängigen Dimensionsattributen aus alternativen Hierarchiepfaden problematisch, weil sich bei diesen im Gegensatz zu unter- und übergeordneten Hierarchieebenen keine eindeutigen Dimensionselemente definieren lassen.

3.2 Implementation

Die im vorhergehenden Kapitel spezifizierten Prozesse wurden in Form eines Prototyps implementiert. Das folgende Kapitel beschreibt nun die zentralen Funktionen des aggregierten Faktenmodells und präsentiert mit dem Prototyp einen konkreten Implementationsvorschlag. Obwohl das aggregierte Faktenmodell eine funktionelle Ergänzung eines bestehenden OLAP-Systems darstellt, wurde der Prototyp als *eigenständige* Lösung konzipiert. Die Weiterentwicklung eines OLAP-Systems wäre gegenüber der Implementation einer eigenständigen Anwendung produktspezifischer und aufgrund der längeren Einarbeitungszeit auch aufwändiger ausgefallen. Sie hätte allerdings den Vorteil gehabt, dass die zur Abbildung des aggregierten Faktenmodells benötigte OLAP-Funktionalität bereits zur Verfügung gestanden wäre. Die Beschreibung der Implementation befasst sich deshalb auch mit elementaren OLAP-Funktionen, jedoch nur soweit dies im Rahmen des Modellverständnisses notwendig ist.

Abbildung 3.2.1 zeigt den Funktionsbaum des Prototyps. Die gesamte Funktionalität des aggregierten Faktenmodells ist auf zwei Anwendungen aufgeteilt. Die erste Anwendung bildet mit den Modell- und Wartungsfunktionen die Entwicklersicht ab. Die zweite Anwendung stellt die Endbenutzersicht dar und enthält die analyserelevanten Funktionen.

Die folgenden Abschnitte gehen auf diese Funktionen ein und zeigen insbesondere, *wie* die implementierte Anwendung die temporalen Aggregatstabellen definiert, aktualisiert und analysiert.

Die beiden entwickelten Anwendungen sowie deren Quellcode können auf der folgenden Internetseite abgerufen werden:

`http://www.wwz.unibas.ch/wi/projects/afm`

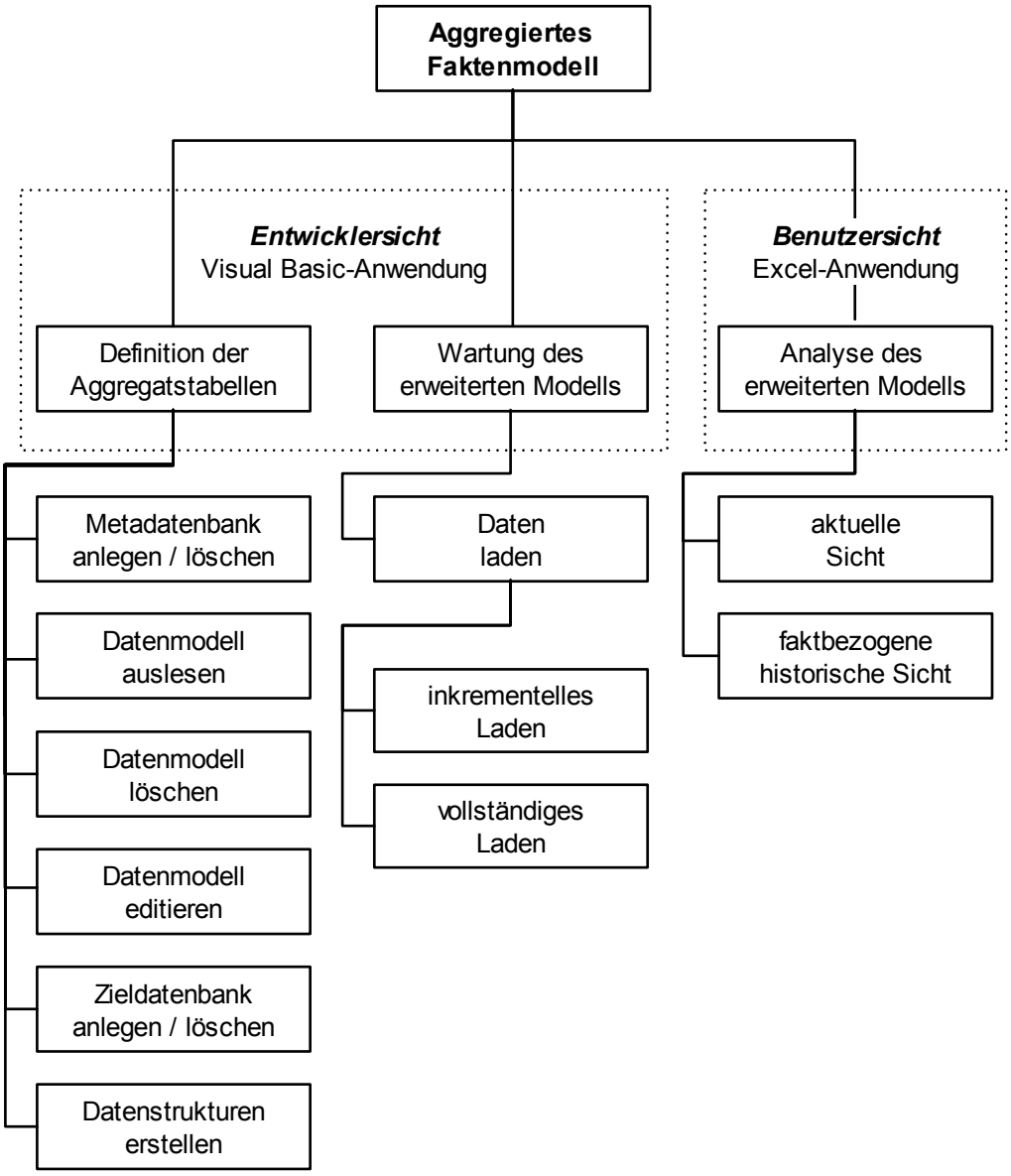


Abbildung 3.2.1: Funktionsbaum des Prototyps

3.2.1 Definition der temporalen Aggregatstabellen

Ausgangslage

Der Prototyp beschränkt sich auf die Abbildung der zentralen Funktionen des aggregierten Faktenmodells und verfügt deshalb gegenüber einem kommerziellen OLAP-System über einen eingeschränkten Funktionsumfang. Die Modellierung der Fakten und Dimensionen sowie die Verwendung konventioneller Aggregate wurden zum Beispiel nicht integriert. Der Prototyp geht bei der De-

finition der temporalen Aggregatstabellen von einem bestehenden Sternschema aus.

Speichermodell

Der Prototyp verwendet zur Verwaltung der Daten ausschliesslich relationale Datenbanken. Die Übertragung des aggregierten Faktenmodells auf MOLAP-Systeme sollte dennoch möglich sein, weil es grösstenteils auf Funktionen basiert, die bereits bei der Verwaltung konventioneller Aggregate genutzt werden.

Quelldatenbank

Die Quelldatenbank enthält die in einem Sternschema modellierten Ausgangsdaten. Damit sich die zusammengehörenden Verbundattribute automatisch ermitteln lassen, setzt der Prototyp voraus, dass die Dimensionstabellen bereits mit der Faktentabelle verknüpft sind.

Zieldatenbank

Nachdem der Prototyp die gewünschten Datenstrukturen bestimmt hat, enthält die Zieldatenbank die temporalen Aggregats- und Dimensionstabellen sowie die Faktentabelle. Die redundante Abbildung der Fakten- und Dimensionstabellen in der Quell- und Zieldatenbank ist nur dann notwendig, wenn die Struktur der Quelltabellen vom Systementwickler oder der Anwendung selbst verändert wird. Unter welchen Umständen dies erforderlich ist und warum die Quell- und Zieldatenbanken im Folgenden immer getrennt dargestellt sind, wird bei der Funktion *Datenmodell auslesen* erörtert. Der Prototyp erstellt neben den temporalen Aggregatstabellen in jedem Fall auch eine Faktentabelle und die dazugehörenden Dimensionstabellen in der Zieldatenbank. Auf die Abbildung konventioneller Aggregatstabellen wurde vollständig verzichtet. Diese sind bei der Verwaltung temporaler Aggregatstabellen nicht relevant.

Metadatenbank

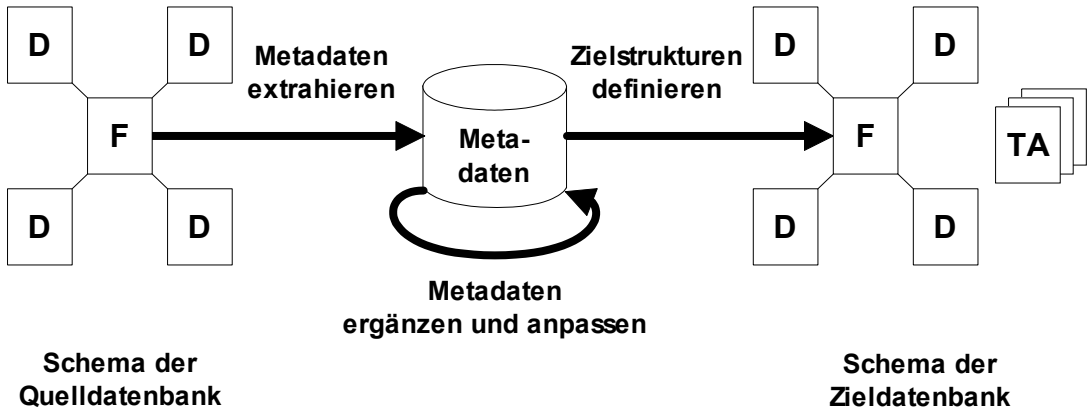
Das aggregierte Faktenmodell benötigt für die Verwaltung der temporalen Aggregate Metadaten der Quell- und Zieldatenmodelle. Der Prototyp verwaltet alle Modelle in einer *getrennten* Metadatenbank, weil bestimmte Eigenschaften wie etwa die Zeitabhängigkeit eines Dimensionsattributs sich weder aus den Metadaten der Quell- oder Zieldatenbank abrufen noch adäquat darin abspeichern lassen. Zudem sind dadurch alle Metadaten eines oder mehrerer Datenmodelle zentral abrufbar, ohne dass die Anwendung bei jeder Funktion auf die Quelldatenbank zurückgreifen muss.

Datenbanktechnologie

Der Prototyp unterstützt *Microsoft Access* Datenbanken und verwendet *Microsoft ActiveX Data Objects (ADO)* für den Objektdaten- und *Microsoft ActiveX Data Objects for Data Definition Language and Security (ADOX)* für den Metadatenzugriff.

Ablauf

Abbildung 3.2.2 zeigt die Quell- und Zielobjekte sowie den Ablauf bei der Definition der temporalen Aggregate im Prototyp: Die Anwendung liest die benötigten Metadaten aus der Quelldatenbank, lässt sie durch den Systementwickler anpassen und definiert daraus das Datenmodell der Zieldatenbank, das neben den Dimensionen und Fakten dann auch die temporalen Aggregate enthält.



Legende: D - Dimensionen, **F** - Faktentabelle, **TA** - Temporale Aggregate

Abbildung 3.2.2: Ablauf bei der Definition der temporalen Aggregate

Metadatenbank anlegen / löschen

Die Metadatenbank umfasst alle benötigten Daten zur Definition, Wartung und Analyse des aggregierten Faktenmodells. Die Funktion *Metadatenbank anlegen* erstellt nach der Angabe des gewünschten Laufwerkpfads eine neue MS Access-Datenbank mit dem in Abbildung 3.2.3 dargestellten Datenmodell. Die Methode `Create` des ADOX-Objekts `Catalog` erstellt die Datenbank, die Methode `Append` fügt die spezifizierten Objekte wie Tabellen, Attribute oder Indizes ein. Abbildung 3.2.4 beschreibt anschliessend die einzelnen Attribute kurz. Die Funktion *Datenmodell editieren* geht dann detaillierter auf die Bedeutung einiger Attribute ein.

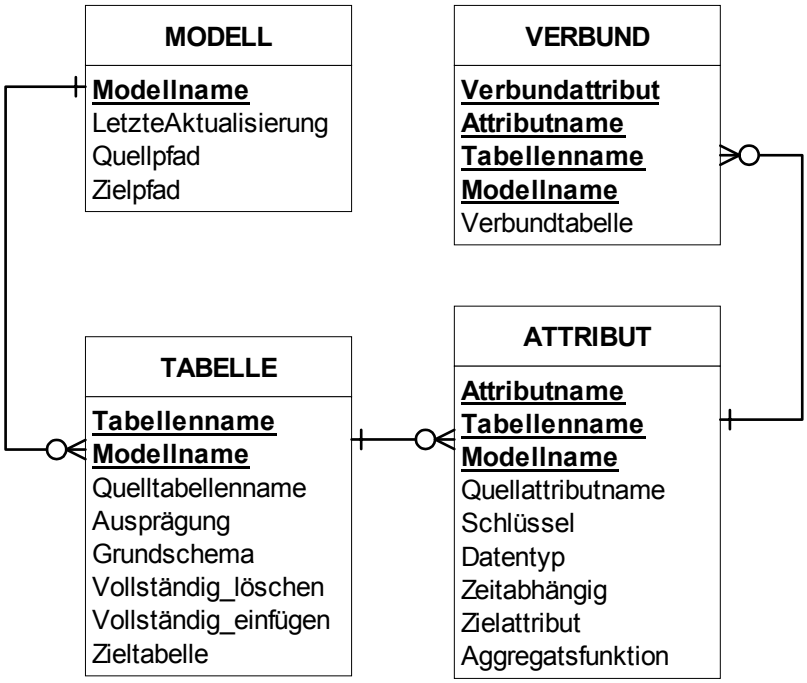


Abbildung 3.2.3: Datenmodell der Metadatenbank

Die Metadatenbank verwaltet eine beliebige Anzahl aggregierter Faktenmodelle, die durch einen eindeutigen Namen definiert sind. Jedes Modell kann mehrere Tabellen und jede Tabelle mehrere Attribute enthalten. Schliesslich verweisen Fremdschlüsselattribute auf Primärschlüsselattribute. Diese Information wird in der Tabelle VERBUND verwaltet.

MODELL

Attributname	Beschreibung	Beispielwert
Modellname	eindeutiger Name des Modells	Verkäufe
LetzteAktualisierung	Datum der letzten Objektdatenaktualisierung	01.05.2002
Quellpfad	Pfad oder Zugriffsparameter der Quelldatenbank	C:\Verkäufe.mdb
Zielpfad	Pfad oder Zugriffsparameter der Zieldatenbank	C:\Verkäufe(AFM).mdb

TABELLE

Attributname	Beschreibung	Beispielwert
Tabellenname	Name der Tabelle in der Zieldatenbank	PRODUKT
Modellname	Fremdschlüsselattribut	Verkäufe
Quelltabellenname	Name der Tabelle in der Quelldatenbank	PRODUKTE
Ausprägung	Art der Tabelle (D, F, TA)	Dimension
Grundschema	Ist die Tabelle in der Quelldatenbank enthalten?	wahr
Vollständig_löschen	SQL-String, der alle Daten in der Tabelle löscht	DELETE * FROM ...
Vollständig_einfügen	SQL-String, der alle Daten neu einfügt	INSERT ...
Zieltabelle	Ist die Tabelle in der Zieldatenbank enthalten?	wahr

ATTRIBUT

Attributname	Beschreibung	Beispielwert
Attributname	Name des Attributs in der Zieldatenbank	PRODUKT_Kategorie
Tabellenname	Fremdschlüsselattribut	PRODUKT
Modellname	Fremdschlüsselattribut	Verkäufe
Quellattributname	Name des Attributs in der Quelldatenbank	Kategorie
Schlüssel	Gehört das Attribut zum Primärschlüssel?	falsch
Datentyp	Datentyp des Attributs	String
Zeitabhängig	Ist das Attribut zeitabhängig zu speichern?	wahr
Zielattribut	Ist das Attribut in der Zieltabelle enthalten?	wahr
Aggregatsfunktion	Mit welcher Funktion wird das Attribut aggregiert?	NULL

VERBUND

Attributname	Beschreibung	Beispielwert
Verbundattribut	Name des Verbundattributs in der Zieldatenbank	NULL
Attributname	Fremdschlüsselattribut	PRODUKT_Kategorie
Tabellenname	Fremdschlüsselattribut	PRODUKT
Modellname	Fremdschlüsselattribut	Verkäufe
Verbundtabelle	Name der Verbundtabelle in der Zieldatenbank	NULL

Abbildung 3.2.4: Funktion der einzelnen Attribute im Metadatenmodell

Die Metadatenbank kann bei Bedarf auch wieder gelöscht werden. Es ist jedoch zu beachten, dass sich die entsprechenden Modelle in den Zieldatenbanken anschliessend nicht mehr aktualisieren lassen.

Datenmodell auslesen

Die Definition eines neuen Modells erfordert die Angabe des Modellnamens sowie des Pfads der Quell-, Ziel- und Metadatenbank. Die Funktion *Datenmodell auslesen* extrahiert mit ADOX-Methoden die benötigten Metadaten aus der

Quelldatenbank und speichert diese zusammen mit den Modellangaben des Systementwicklers in der Metadatenbank.

Die Attribute der Metadatenbank lassen sich in drei Kategorien einteilen. Die Werte der ersten Kategorie werden direkt von der Quelldatenbank bestimmt und sind deshalb nicht veränderbar. Zu dieser gehören die Attribute Quelltabellenname, Grundschema, Quellattributname, Schlüssel und Datentyp. Bei den Attributen der zweiten Kategorie setzt der Prototyp zunächst Standardwerte oder bestimmt den gewünschten Wert anhand von Ableitungsregeln. Der Systementwickler kann diese Werte später unter Berücksichtigung bestimmter Modellrestriktionen nach den Anforderungen der Benutzer verändern. Mit Ausnahme von LetzteAktualisierung, Vollständig_löschen, Vollständig_einfügen gehören alle restlichen Attribute des Modells zur zweiten Kategorie. Die drei Ausnahmen zählen schliesslich zur dritten Kategorie und können nur von der Anwendung und nicht vom Benutzer definiert oder verändert werden. Das Programm bestimmt die Werte dieser Attribute erst bei der Wartung der Objektdaten.

Die Funktion *Datenmodell auslesen* definiert für die Attribute der zweiten Kategorie die in Abbildung 3.2.5 dargestellten Standardwerte.

Attribut	**Standardwert**
Modellname	entwicklerdefiniert
Quellpfad	entwicklerdefiniert
Zielpfad	entwicklerdefiniert
Tabellenname	Quelltabellenname
Ausprägung	Regel 1
Zieltabelle	*wahr*
Attributname	Regel 2
Zeitabhängig	*falsch*
Zielattribut	Regel 3
Aggregatsfunktion	Regel 4
Verbundattribut	Name des entsprechenden Attributs
Verbundtabelle	Name der entsprechenden Tabelle

Abbildung 3.2.5: Standardwerte der Attribute

Regeln

1. Das Attribut Ausprägung erhält den Wert *Dimension*, falls in der Tabelle nicht mehr als zwei Teilschlüsselattribute enthalten sind. Ansonsten nimmt das Attribut den Wert *Fakt* an. Die Anwendung geht davon aus, dass der Primärschlüssel einer Dimensionstabelle aus einem einzigen Attribut besteht oder, falls die Dimensionsdaten zeitbezogen in der Quelldatenbank vorliegen, zusätzlich ein Zeitstempelattribut zum Primärschlüssel gehört.
2. Das Attribut Attributname erhält zunächst denselben Wert wie das Attribut Quellattribut. Nach dem Setzen aller Standardwerte vergleicht das Programm alle Dimensionsattributnamen. Falls zwei identisch sind, erhalten diese einen neuen Namen, der aus dem Tabellennamen und dem ursprünglichen Attributnamen zusammengesetzt ist. Die Eindeutigkeit der Dimensionsattributnamen ist notwendig, weil sie bei der Definition der Aggregatstabellen zusammen in einer Tabelle stehen könnten. Diese Eindeutigkeit überprüft das Programm auch bei der manuellen Anpassung.

3. Das Attribut Zielattribut erhält den Wert *wahr*, falls es sich nicht um ein Zeitstempelattribut einer Dimensionstabelle handelt. Das Programm erkennt Zeitstempelattribute, wenn ihr Name mit „Gültig“ wie etwa *GültigAb* oder *GültigBis* beginnt. Diese Angabe könnte auch verallgemeinert werden, wenn die Metadatenbank ein weiteres Attribut, zum Beispiel Zeitstempel mit den Werten *wahr* oder *falsch* enthielte.

4. Das Attribut Aggregatsfunktion erhält den Wert *NULL*, falls es kein Faktattribut ist, ansonsten den Wert *SUM*. Das Attribut wird als Fakt identifiziert, wenn das entsprechende Attribut Ausprägung den Wert *Fakt* hat und das Attribut nicht zum Primärschlüssel gehört (Schlüssel = *falsch*).

Während die Anpassung des Attributnamens notwendig sein kann, ist die Möglichkeit zur Umbenennung des Tabellennamens für die Modellfunktionalität nicht erforderlich. Sie ermöglicht jedoch eine bessere Anpassung des Zielmodells an die Bedürfnisse der Benutzer, ohne dabei die Komplexität wesentlich zu erhöhen.

Die ausschliessliche Ergänzung des Quellmodells mit Aggregatstabellen und damit die Verschmelzung der Quell- und Zieldatenbank wäre nur möglich, wenn:

- das Quellmodell keine Zeitstempelattribute in den Dimensionstabellen enthält,
- alle Attribute und Tabellen vollständig und unverändert im Zielmodell vorhanden sein sollen,
- und das Quellmodell bereits eindeutige Dimensionsattributnamen enthält.

Die genannten Bedingungen lassen sich einhalten, wenn das Datenmodell der Quelldatenbank bereits von einem OLAP-System aufbereitet wurde. Dann müsste der Prototyp nur die zusätzliche Funktionalität des aggregierten Faktenmodells übernehmen und ausschliesslich die temporalen Aggregate zum bestehenden Quellsystem verwalten. Wie hier jedoch gezeigt wurde, übernimmt die Anwendung bestimmte, zur Abbildung des aggregierten Faktenmodells notwendige OLAP-Funktionen wie etwa die Anpassung der Attributnamen. Deshalb kann der Prototyp auch als rudimentäres OLAP-System betrachtet werden, das keine Funktionen zur Modellierung eines Sternschemas besitzt und aus diesem Grunde bereits ein solches in der Quelldatenbank voraussetzt. Aus dieser Sicht stellt die Quelldatenbank den externen Datenlieferanten (zum Beispiel einen Data Mart) und die Zieldatenbank die eigentliche OLAP-Datenbank dar.

Datenmodell löschen

Die Funktion *Datenmodell löschen* entfernt das gewünschte Modell aus der Metadatenbank. Die im entsprechenden Modell eingetragene Zieldatenbank lässt sich nach dem Löschvorgang nicht mehr aktualisieren. Der Systementwickler wählt dazu die gewünschte Metadatenbank und das darin zu löschende Modell aus. Eine SQL-Anweisung entfernt anschliessend mit den entsprechenden Parametern sämtliche Daten, die zu diesem Modell gehören:

```
DELETE *
FROM   MODELL IN '<Pfad und Name der Metadatenbank>'
WHERE  Modellname = <Name des zu löschenden Modells>;
```

Diese SQL-Anweisung genügt, wenn die ▸ referentielle Integrität in der Metadatenbank jeweils mit Löschweitergabe definiert wurde.

Datenmodell editieren

Die Attribute der zweiten Kategorie (vgl. Funktion *Datenmodell auslesen*) lassen sich nach dem Anlegen eines neuen Modells gemäss den Anforderungen der Anwender anpassen. Abbildung 3.2.6 zeigt die Benutzerschnittstelle, in dem der Systementwickler die Metadaten verändern kann.

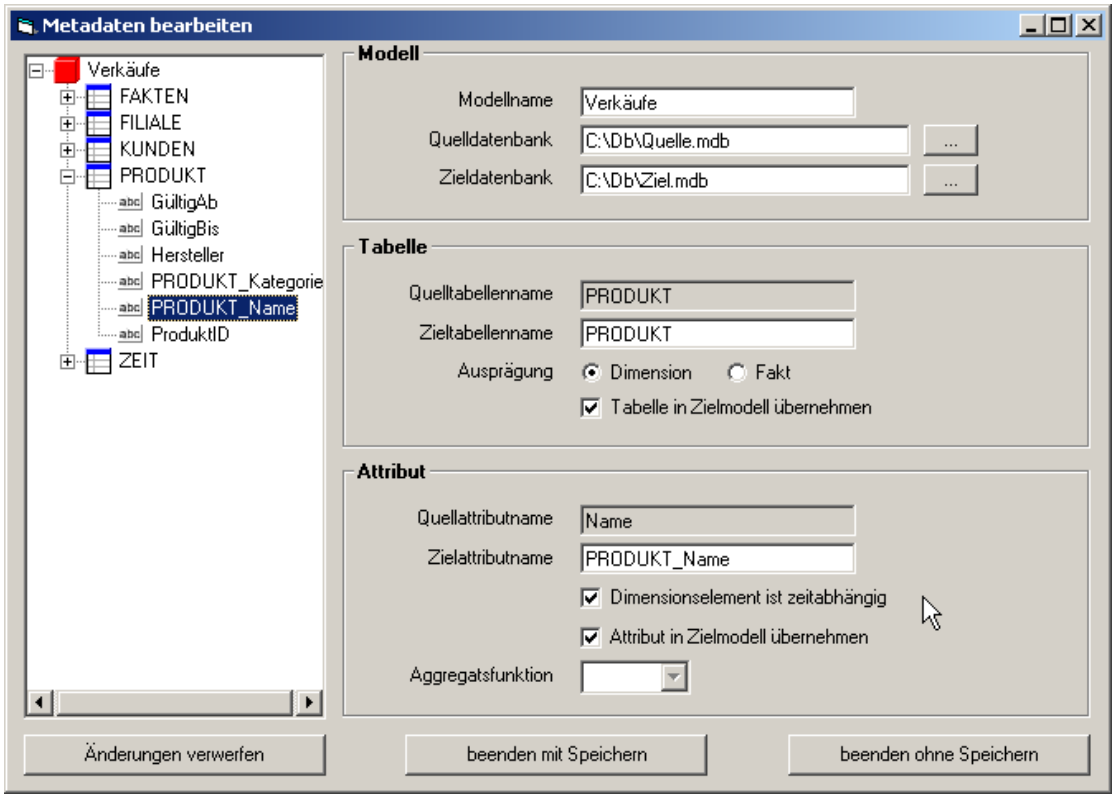


Abbildung 3.2.6: Formular *Metadaten bearbeiten*

Auf der linken Seite wählt der Benutzer die zu bearbeitenden Objekte (Modell, Tabelle, Attribut), und auf der rechten Seite zeigt das Programm die Eigenschaften des ausgewählten Objekts an. Bestimmte Steuerelemente sind immer oder nur im angezeigten Kontext gesperrt und können deshalb nicht verändert werden. Zum Beispiel darf der Benutzer nur bei Fakten eine Aggregatsfunktion definieren oder verändern.

Die wichtigste Anpassung bei der Definition des aggregierten Faktenmodells betrifft das Attribut Zeitabhängig. Der Wert dieses Attributs wird im Formular in einem Checkbox-Steuerelement dargestellt und hat die Bezeichnung *Dimensionselement ist zeitabhängig*. Der Prototyp muss erst dann temporale Aggregatstabellen definieren, wenn der Systementwickler mindestens bei einem Dimensionsattribut diese temporale Eigenschaft von *falsch* auf *wahr* ändert.

Die Funktion *Datenmodell editieren* berücksichtigt vor dem Verändern der Metadaten zahlreiche Zulässigkeitsbedingungen und gibt gegebenenfalls eine Feh-

lermeldung an den Benutzer aus. Zudem führt sie Folgeänderungen, die aufgrund der Benutzerangaben notwendig sind, automatisch durch.

Zu den Zulässigkeitsbedingungen gehören:

- Das Programm akzeptiert bei den veränderbaren Attributen keine Nullwerte.
- Der Modellname muss innerhalb der Metadatenbank, die Zieltabellen- und Attributnamen innerhalb des jeweiligen Modells, eindeutig sein.

Zu den Folgeänderungen gehören:

- Veränderungen der Attribute Attributname oder Tabellenname erfordern eine Anpassung der Attribute Verbundattribut beziehungsweise Verbundtabelle.
- Das Weglassen/Hinzufügen eines Fremdschlüsselattributs in der Faktentabelle bewirkt, dass das Attribut Zieltabelle der entsprechenden Dimensionstabelle auf *falsch*/*wahr* gesetzt wird.

Zieldatenbank anlegen / löschen

Die Funktion *Zieldatenbank anlegen* erstellt eine neue (leere) MS Access-Datenbank. Das Vorgehen ist identisch wie bei der Funktion *Metadatenbank anlegen*: Nachdem der Benutzer den gewünschten Laufwerkspfad und Namen der Datenbank spezifiziert hat, definiert die ADOX-Methode `Create` die leere mdb-Datei.

Die Funktion *Zieldatenbank löschen* ist identisch mit der Funktion *Metadatenbank löschen*. Die beiden Funktionen nutzen deshalb auch die gleichen Prozeduren.

Datenstrukturen erstellen

Nachdem die Metadaten aus der Quelldatenbank ausgelesen und durch den Systementwickler benutzerspezifisch angepasst wurden, definiert die Funktion *Datenstrukturen erstellen* die Faktentabelle, die Dimensionstabellen und die temporalen Aggregatstabellen in der Zieldatenbank.

Die Faktentabelle und die temporalen Aggregatstabellen sind gleich strukturiert: Beide enthalten Faktattribute und einen zusammengesetzten Primärschlüssel. Während die Aggregatstabellen dieselben Faktattribute enthalten wie

die Faktentabelle, unterscheiden sich ihre Primärschlüssel. Deshalb wird zunächst nur die Zusammensetzung des Primärschlüssels untersucht.

Jedes Teilschlüsselattribut einer *Faktentabelle* ist zugleich ein Fremdschlüsselattribut, das auf das Primärschlüsselattribut einer Dimensionstabelle verweist. Abbildung 3.2.7 zeigt diesen Zusammenhang anhand eines Beispiels. *Kursiv* gedruckte Attribute sind zeitabhängig.

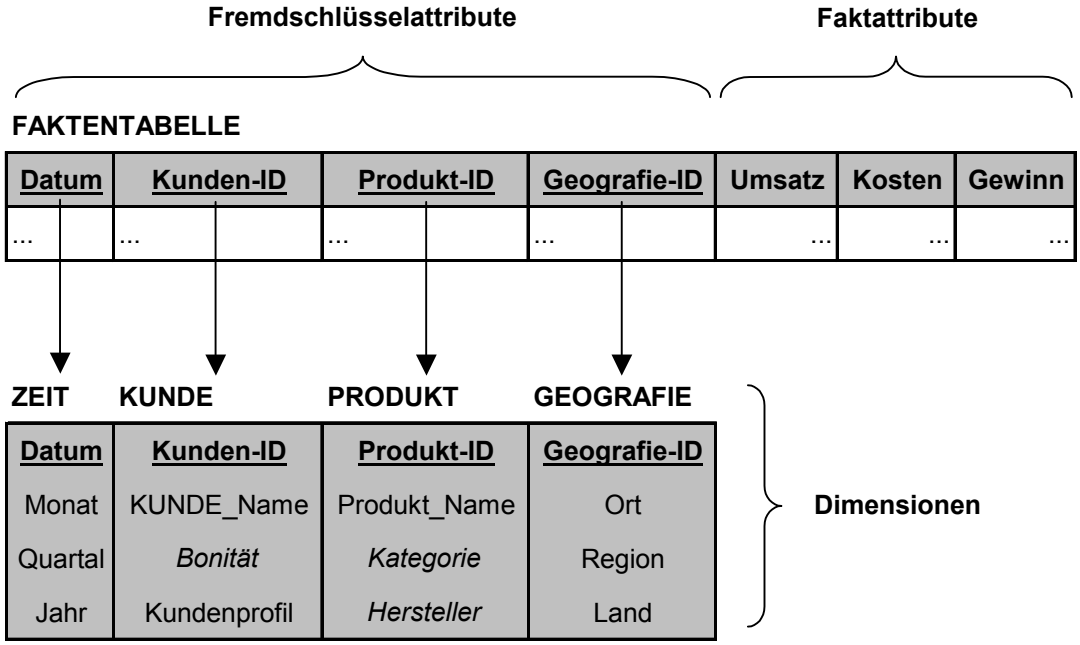


Abbildung 3.2.7: Struktur einer Faktentabelle

Die *temporalen Aggregatstabellen* haben ebenfalls einen zusammengesetzten Primärschlüssel. Die Anzahl Teilschlüssel der temporalen Aggregatstabellen und der Faktentabelle ist immer identisch. Jeder Primärschlüssel einer temporalen Aggregatstabelle enthält aber mindestens ein zeitabhängiges Dimensionsattribut anstelle des ursprünglichen Fremdschlüssels, wie er in der Faktentabelle vorzufinden ist. Abbildung 3.2.8 zeigt eine Aggregatstabelle, in der nicht mehr das Attribut Kunden-ID, sondern das zeitabhängige Attribut Bonität zum Primärschlüssel gehört.

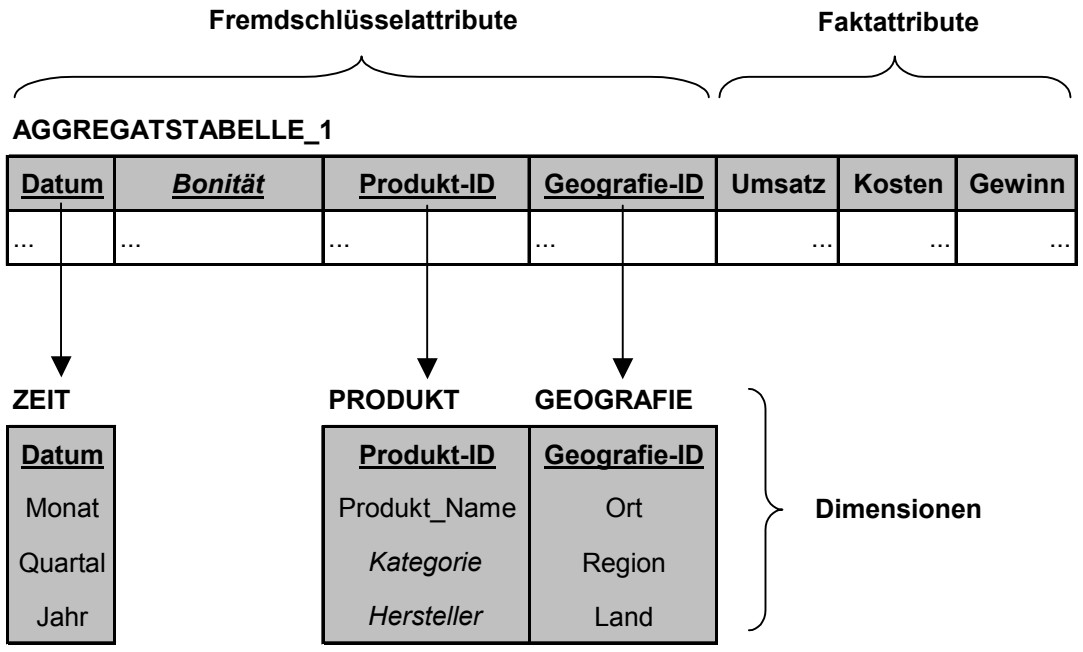


Abbildung 3.2.8: Struktur einer temporalen Aggregatstabelle

Die Primärschlüssel der temporalen Aggregatstabellen leiten sich aus dem Primärschlüssel der Faktentabelle sowie den zeitabhängigen Dimensionsattributen ab. Die zeitabhängigen Attribute ersetzen dabei die ursprünglichen Fremdschlüssel sukzessive, bis alle Kombinationen aus ursprünglichen Fremdschlüsseln und zeitabhängigen Attributen in temporalen Aggregatstabellen definiert sind. Im Beispiel der Abbildung 3.2.9 seien die Attribute Bonität, Kategorie und Hersteller zeitbezogen zu speichern.

Abbildung 3.2.9: Primärschlüssel der Faktentabelle und der Aggregatstabellen

In den ersten drei Aggregatstabellen enthält der Primärschlüssel jeweils ein zeitabhängiges Dimensionsattribut anstelle des entsprechenden Fremdschlüs-

selattributs der Faktentabelle. In der vierten und fünften Aggregatstabelle werden schliesslich auch Kombinationen mit zwei zeitabhängigen Attributen abgebildet. Diese sind zur Beantwortung von Fragen notwendig, die sowohl Hersteller oder Kategorie als auch die Bonität betreffen.

Der Prototyp ermittelt die zur Definition der gesuchten Primärschlüssel benötigten Attribute aus der Metadatenbank und schreibt sie in ein zweidimensionales Datenfeld **RA** (relevante Attribute). Abbildung 3.2.10 zeigt das Ausgangsdatenfeld **RA** und das gesuchte Zieldatenfeld **X** mit den Indizes `i` und `k` beziehungsweise `j` und `i`. Die erste Spalte (`k = 1`) enthält die Fremdschlüsselattribute der Faktentabelle, die auf den Primärschlüssel der Dimensionen `i` verweisen. Die restlichen Spalten enthalten die zeitabhängigen Attribute der Dimensionen `i`.

Datenfeld RA

`k = 1 ... r`

`i = 1 ... n`

	1	2	3
1	Datum		
2	Kunden-ID	Bonität	
3	Produkt-ID	Kategorie	Hersteller
4	Geografie-ID		

Datenfeld X

`i = 1 ... n`

`j = 1 ... m`

	1	2	3	4
1	Datum	Bonität	Produkt-ID	Geografie-ID
2	Datum	Kunden-ID	Kategorie	Geografie-ID
3	Datum	Kunden-ID	Hersteller	Geografie-ID
4	Datum	Bonität	Kategorie	Geografie-ID
5	Datum	Bonität	Hersteller	Geografie-ID

Abbildung 3.2.10: Ausgangs- und Zielfeld

Die Anzahl der Elemente in den Datenfelddimensionen `i`, `j` und `k` werden wie folgt bestimmt:

- `m` ist gleich der Anzahl zu definierender Aggregatstabellen.
- `n` ist gleich der Anzahl Dimensionen im Sternschema beziehungsweise der Anzahl Fremdschlüssel in der Faktentabelle.

- r wird von der höchsten Anzahl zeitabhängiger Attribute einer Dimension bestimmt.

Während sich n und r direkt aus den Metadaten ermitteln lassen, muss m berechnet werden. Die Anwendung kann die Kombinationsmöglichkeiten m mit derselben Formel festlegen, mit der auch die möglichen Faktwerte in einem Datenwürfel bestimmt werden:[109]

$$m = \prod_{i=1}^{n} (r_i + 1) - 1$$

Während r das Maximum zeitabhängiger Attribute aller Dimensionen einschliesslich ursprünglichem Fremdschlüssel der Faktentabelle darstellt, entspricht r_i der Anzahl zeitabhängiger Attribute in der Dimension i *ohne* Fremdschlüsselattribut. Da der ursprüngliche Fremdschlüssel bei der Berechnung von m auch zu berücksichtigen ist, wird jedem r_i noch jeweils ein Element hinzugefügt ($r_i + 1$). Die Kombinationsmöglichkeit, die nur aus den ursprünglichen Fremdschlüsselattributen besteht, wird abgezogen (-1), weil die daraus abgeleitete Tabelle identisch mit der bereits vorhandenen Faktentabelle wäre.

Im aufgeführten Beispiel ergibt die Berechnung das folgende Resultat:

$r_1 = 0,\ r_2 = 1,\ r_3 = 2,\ r_4 = 0,\ n = 4$

$\rightarrow m = [(0 + 1) * (1 + 1) * (2 + 1) * (0 + 1)] - 1 = 5$

Der Algorithmus **Bestimme_Primärschlüssel(i, j)** transformiert das Ausgangsdatenfeld **RA** in das Zieldatenfeld **X**:

```
Bestimme_Primärschlüssel(i, j)                                  (1)
   für jedes Element y(k) ∈ RA(i)                               (2)
      falls j > 1 dann                                          (3)
         für p = 1 bis n                                        (4)
            falls X(p, j) Null ist, dann                        (5)
               X(p, j) = X(p, j - 1)                            (6)
      X(i, j) = RA(k, i)                                        (7)
      falls i < n dann                                          (8)
         Bestimme_Primärschlüssel(i + 1, j)                     (9)
      sonst                                                    (10)
         j = j + 1                                             (11)
```

[109] Vgl. Oehler 2000: S. 130

Erläuterungen

(2) Die Schleife bewirkt zusammen mit dem rekursiven Aufruf (9), dass alle Elemente einer Zeile aus dem Datenfeld **RA** mit allen Elementen der anderen Zeilen kombiniert werden.

(3) Der Algorithmus überprüft, ob die erste Zeile des Zieldatenfelds **x** definiert wird.

(4)-(6) Wenn dies (3) nicht der Fall ist und die Werte in der entsprechenden Zeile noch nicht bestimmt wurden, definiert der Algorithmus die Zeile mit den Werten der vorhergehenden Zeile. Diese Initialisierung ist notwendig, weil der Algorithmus anschliessend nur die Werte jener Elemente bestimmt, die sich vom entsprechenden Element der vorhergehenden Zeile unterscheiden oder im Datenfeld rechts vom veränderten Element liegen.

(8)-(9) Der Algorithmus ruft sich solange auf, bis alle Elemente in der Zeile bestimmt sind (Rekursion).

Weil der ursprüngliche Primärschlüssel der Faktentabelle immer die erste Kombination der Elemente aus dem Datenfeld **RA** darstellt, umfasst das mit dem Algorithmus **Bestimme_Primärschlüssel (i, j)** definierte Datenfeld **x** nicht m, sondern m + 1 Zeilen.

Die Funktion *Datenstrukturen erstellen* legt nach der Definition des Datenfelds **x** die benötigten Aggregatstabellen, die Faktentabelle sowie die Dimensionstabellen in der Zieldatenbank an. Die Aggregatstabellen werden dabei mit einer fortlaufenden Nummerierung benannt. Die AGGREGATSTABELLE_j (j = 1 ... m) setzt sich aus den Primärschlüsselattributen (**x**(i, j + 1), i = 1 ... n) und den Faktattributen zusammen. Die Funktion bestimmt die Datentypen anhand der entsprechenden Attribute aus der Metadatenbank. Im Gegensatz zu den Aggregatstabellen lassen sich die Dimensionstabellen und die Tabellenverknüpfungen direkt aus den Metadaten definieren.

Damit das Endbenutzerwerkzeug anschliessend auf die temporalen Aggregate zugreifen kann, werden die Datendefinitionen der neuen Aggregatstabellen nach dem Erstellen der Datenstrukturen in die Metadatenbank eingefügt. Der Prototyp bestimmt dabei die Werte der in Abbildung 3.2.11 dargestellten Attribute.

Attribut	Standardwert
Tabellenname	AGGREGATSTABELLE_j
Ausprägung	*Aggregat*
Grundschema	*falsch*
Vollständig_löschen	SQL-String für das Löschen der Zieldaten
Vollständig_einfügen	SQL-String für das vollständige Einfügen der Zieldaten
Zieltabelle	*wahr*
Attributname	Regel 1
Quellattributname	entsprechender Quellattributname von Attributname
Schlüssel	Regel 2
Datentyp	entsprechender Datentyp von Attributname
Zeitabhängig	*falsch*
Zielattribut	*wahr*

Abbildung 3.2.11: Ergänzungen in der Metadatenbank

Regeln

1. Das Attribut Attributname erhält den Wert des entsprechenden Attributs aus der Faktentabelle oder, falls es sich um ein zeitabhängiges Dimensionsattribut handelt, aus der betreffenden Dimensionstabelle.

2. Das Attribut Schlüssel erhält den Wert *wahr*, falls das entsprechende Attribut Attributname nicht zu den Faktattributen gehört, ansonsten erhält es den Wert *falsch*.

SQL-Strings

Der Prototyp definiert bei der Erstellung der Datenstrukturen alle statischen Abfragen zur vollständigen Aktualisierung der Tabellen in der Zieldatenbank. Dadurch muss die Anwendung die SQL-Abfragestrings nur einmal bestimmen und kann bei Ausführung der Aktualisierungsprozesse direkt auf sie zurückgreifen. Die vollständige Aktualisierung der temporalen Aggregate ist nur möglich, wenn die Dimensionsdaten der Quelldatenbank bereits zeitbezogen gespeichert sind. Der Prototyp geht dabei von der Intervall-Zeitstempelung mit zwei Zeitstempelattributen aus (GültigAb, GültigBis).[110] Die eigentliche Aktualisierung erfolgt in der Funktion *Daten laden*.

Der Ablauf der vollständigen Aktualisierung wird in zwei Prozesse unterteilt: Die erste Abfrage *löscht* zunächst alle bereits bestehenden Daten und die zweite *fügt* anschliessend die aufbereiteten Quelldaten in die Zieltabellen *ein*. Die

[110] Vgl. Kapitel 2.2.3: S. 66

Elemente in spitzigen Klammern <Element> werden aus der Metadatenbank gelesen. Bei der Zusammensetzung der SQL-Strings sind die Elemente i nur dann zu berücksichtigen, wenn **x**(i, j + 1) ein zeitabhängiges Dimensionsattribut ist.

Löschabfrage (allgemein):

```
DELETE      *
FROM        <Tabellenname> IN '<Zielpfad>';
```

Einfügeabfrage der temporalen Aggregatstabellen:

```
INSERT INTO <Aggregatstabelle j>(<Attributnamen>)
               IN '<Zielpfad>'
SELECT      <Q-Tabellennamen>.<Q-Attributnamen>,
               <Aggregatsfunktion s>(<Q-Faktentabelle>
                  .<Q-Faktattributnamen s>)
FROM        <Q-Faktentabelle>, <Q-Dimensionstabellen>
               IN '<Quellpfad>'
WHERE       <Q-Dimensionstabelle i>.<Q-Dimensionsschlüssel i>
               = <Q-Faktentabelle>
                     .<Q-Faktentabellenfremdschlüssel i>
   AND      <Q-Dimensionstabelle i>.GültigAb
               <= <Q-Faktentabelle>.<Zeitstempelattribut>
   AND      (<Q-Dimensionstabelle i>.GültigBis
               > <Q-Faktentabelle>.<Zeitstempelattribut>
               OR <Q-Dimensionstabelle i>.GültigBis IS NULL)
GROUP BY    <Q-Tabellennamen>.<Q-Attributnamen>;
```

Erläuterungen

- n ist gleich der Anzahl Dimensionen und t ist gleich der Anzahl Faktattribute in der Zieldatenbank.
- <Attributnamen> der j-ten Aggregatstabelle ist die Menge {**x**(1...n, j + 1), <Faktattribut s>}, s = 1...t
- <Q-Tabellennamen> ist die Menge {<Q-Dimensionstabelle i>, <Q-Faktentabelle>}
- GültigAb und GültigBis sind die Intervall-Zeitstempelattribute in den Dimensionstabellen der Quelldatenbank (<Q-Dimensionstabelle i>).

- <Zeitstempelattribut> ist das Zeitstempelattribut der Faktentabelle in der Quelldatenbank (<Q-Faktentabelle>). Es lässt sich mit Hilfe der Metadatenbank bestimmen, ist aber nicht direkt darin enthalten:

```
SELECT Quellattributname AS <Zeitstempelattribut>
FROM   TABELLE, ATTRIBUT
WHERE  Quelltabellenname = <Q-Faktentabelle>
   AND Ausprägung = "Fakten"
   AND Datentyp = 7
   AND Schlüssel = TRUE
   AND TABELLE.Modellname = ATTRIBUT.Modellname
   AND TABELLE.Tabellenname = ATTRIBUT.Tabellenname;
```

  Der Wert *7* des Attributs Datentyp bedeutet, dass der Datentyp des Quellattributnamens als *Datum* definiert sein muss.

Einfügeabfrage der Faktentabelle:

```
INSERT INTO <Faktentabelle>(<Attributnamen>) IN '<Zielpfad>'
SELECT      <Q-Faktentabelle>.<Q-Attributnamen>,
               <Aggregatsfunktion s>(<Q-Faktentabelle>
                  .<Q-Faktattributnamen s>)
FROM        <Q-Faktentabelle> IN '<Quellpfad>'
GROUP BY    <Q-Faktentabelle>.<Q-Attributnamen>;
```

Einfügeabfrage der Dimensionstabellen:

```
INSERT INTO <Dimensionstabelle i>(<Attributnamen>)
               IN '<Zielpfad>'
SELECT      <Q-Dimensionstabelle i>.<Q-Attributnamen>,
FROM        <Q-Dimensionstabelle i> IN '<Quellpfad>'
WHERE       GültigBis IS NULL;
```

Die letzte Zeile (**WHERE** GültigBis **IS NULL**) entfällt bei der Zeitdimension oder bei nicht zeitabhängig gespeicherten Dimensionsdaten, weil diese keine Zeitstempelattribute enthalten.

Im Gegensatz zu den SQL-Strings der vollständigen Aktualisierungsmethode lassen sich die Abfragestrukturen der inkrementellen Aktualisierungsmethode

nicht statisch vordefinieren, weil die Auswahl der neuen Quelldaten jeweils Bezug auf das letzte Aktualisierungsdatum nehmen muss.

3.2.2 Wartung des erweiterten Modells

Nach der Definition der Datenstrukturen in der Zieldatenbank kann der Anwender die Quelldaten aufbereiten und in die Zieldatenbank laden. Die Wartungskomponente umfasst dazu die Funktionen *Daten laden* sowie dessen Unterfunktionen *Daten vollständig laden* und *Daten inkrementell laden*.

Daten laden

Die Funktion *Daten laden* unterstützt zur periodischen Fortschreibung der Dimensionen, Fakten und Aggregate sowohl eine vollständige als auch eine inkrementelle Aktualisierungsmethode. Der Systementwickler wählt die gewünschte Methode sowohl für die Dimensionen als auch für die Fakten/Aggregate über den Menüpunkt *Optionen* → *Aktualisierungsmethode*. Während die vollständige Aktualisierungsmethode sämtliche Daten neu aus der Quelldatenbank lädt, bestimmt die inkrementelle Aktualisierungsmethode die seit der letzten Fortschreibung neu hinzugekommenen Quelldaten und verarbeitet diese mit den bereits bestehenden Zieldaten. Die Funktion benötigt den Zeitpunkt der letzten Aktualisierung, um zwischen den bereits bestehenden und den neuen Quelldaten zu unterscheiden. Deshalb speichert sie den Aktualisierungszeitpunkt nach jeder Fortschreibung im Attribut LetzteAktualisierung der Metadatenbank.

Die referentielle Integrität zwischen der Faktentabelle und den Dimensionstabellen schränkt die Methoden zur Aktualisierung der Dimensionsdaten ein. Der Aktualisierungsprozess darf einen Dimensionsdatensatz nur dann löschen, wenn kein entsprechender Fremdschlüsselwert der Faktentabelle seinen Primärschlüssel referenziert.

Damit die Zieldatenbank die Aktualisierungsanweisungen trotzdem zulässt, stehen dem Prototypen die folgenden Alternativen offen:

- Die Fortschreibungsmethode verwendet keine Lösch- (`DELETE`), sondern lediglich Aktualisierungsanweisungen (`UPDATE`).
- Die Fortschreibungsmethode entfernt die referentielle Integrität vor der Aktualisierung und definiert sie danach wieder.
- In der Zieldatenbank wird auf die referentielle Integrität vollständig verzichtet.

Der Prototyp wendet die zweite Vorgehensweise an. Damit bleibt die referentielle Integrität in der Zieldatenbank erhalten und gleichzeitig stehen im Aktualisierungsprozess auch Löschabfragen zur Verfügung. Dieses Vorgehen setzt allerdings voraus, dass die Programmlogik die Datenintegrität während der Fortschreibung sicherstellt.

Daten vollständig laden

Der Algorithmus zur vollständigen Aktualisierung ist für alle Tabellen gleich. Allerdings lassen sich die temporalen Aggregatstabellen nur dann vollständig aktualisieren, wenn die Dimensionsdaten in der Quelldatenbank bereits zeitbezogen abgebildet sind. Die Funktion *Daten laden* überprüft deshalb zunächst, ob die notwendigen SQL-Strings (Vollständig_löschen, Vollständig_einfügen) der fortzuschreibenden Tabellen in der Metadatenbank vorhanden sind.

Falls die Anwendung die Abfragedefinitionen ermitteln kann, führt sie den SQL-String Vollständig_löschen aus und löscht damit alle bestehenden Daten der zu aktualisierenden Tabellen. Anschliessend lädt sie mit dem SQL-String Vollständig_einfügen die Quelldaten der betroffenen Datenobjekte in die Zieldatenbank.

Daten inkrementell laden

Die inkrementelle Aktualisierungsmethode kann auf die Dimensionstabellen, die Faktentabelle und die temporalen Aggregatstabellen angewandt werden. Die Funktion *Daten inkrementell laden* bestimmt die zur inkrementellen Aktualisierung notwendigen Abfrageparameter wie etwa das letzte Aktualisierungsdatum, die einander entsprechenden Attribut- und Tabellennamen aus der Quell- und Zieldatenbank oder die Aggregatsfunktion der Faktattribute mit Hilfe der Metadatenbank.

Inkrementelle Aktualisierung der Dimensionstabellen

Die inkrementelle Aktualisierung der Dimensionsdaten wählt die neuen Datensätze aus der Quelldatenbank und löscht die Werte aus der Zieldatenbank, die denselben Primärschlüsselwert haben wie die neuen, gerade ausgewählten Datensätze der Quelldatenbank. Anschliessend fügt sie die neuen Datensätze in die Dimensionstabellen der Zieldatenbank ein. Der folgende Algorithmus stellt diesen Prozess formell dar:

```
für jede Dimensionstabelle i in der Zieldatenbank          (1)
   wähle die neuen Datensätze NDS aus                      (2)
      Q-Dimensionstabelle i
   lösche alle Datensätze in der Dimensionstabelle i,      (3)
      welche dieselben Primärschlüssel wie NDS haben
   füge NDS in die Dimensionstabelle i ein                 (4)
```

Erläuterungen

(2) Die Funktion *Daten laden* kann nur zwischen alten und neuen Dimensionsdaten unterscheiden, falls die Dimension ein Zeitstempelattribut enthält. Dies ist bei zeitbezogen gespeicherten Dimensionsdaten und bei der Zeitdimension der Fall. Bei diesen wird eine zusätzliche Bedingung aufgeführt, die das Zeitstempelattribut mit dem Datum der letzten Aktualisierung (LetzteAktualisierung) vergleicht:

SQL-String, der alle Daten der Dimension `i` aus der Quelltabelle selektiert:

```
SELECT <Q-Dimensionstabelle i>.<Q-Attributnamen i>
FROM   <Q-Dimensionstabelle i>;
```

Zusätzliche Bedingung, falls die Quelldaten zeitbezogen gespeichert sind:

```
WHERE  GültigAb > <LetzteAktualisierung>
   AND GültigBis IS NULL
```

Zusätzliche Bedingung, falls es sich um die Zeitdimension handelt:

```
WHERE  <Q-Dimensionstabelle i>.<Q-Dimensionsschlüssel i>
          > <LetzteAktualisierung>
```

Dabei gilt:

- Es werden nur die Quellattribute `<Q-Attributnamen i>` ausgewählt, die auch im Zielmodell vorhanden sind (Zielattribut = *wahr*).
- Die Identifikation zeitbezogener Dimensionsdaten erfolgt anhand der Attribute GültigAb und GültigBis.
- Die Zeitdimension wird dann als solche erkannt, wenn der Datentyp des Primärschlüssels als Datum definiert ist.

(3) Das Löschen stellt sicher, dass sich die Dimensionsdaten einfügen lassen, ohne die Entitätsintegrität zu verletzen.

Inkrementelle Aktualisierung der Faktentabelle

Die inkrementelle Aktualisierung der Faktentabelle wählt die neuen Datensätze aus der Quelldatenbank und fügt sie in die Faktentabelle der Zieldatenbank ein. Im Gegensatz zur Identifikation der neuen Dimensionswerte kann die Aktualisierungsfunktion bei der Faktentabelle immer auf ein Zeitstempelattribut zurückgreifen, weil das aggregierte Faktenmodell die zeitabhängige Speicherung der Faktwerte voraussetzt.

Bei der Implementierung des Prototyps wurde vereinfachend angenommen, dass die Granularität des Zeitstempelattributs in der Quell- und Zieltabelle identisch ist. Unter dieser Voraussetzung sind die aus der Quelldatenbank ausgewählten (neuen) Faktdatensätze noch nicht in der Zieldatenbank vorhanden und können deshalb immer mit einer *Anfügeabfrage* geladen werden:

```
INSERT INTO <Faktentabellenname> (<Faktattributnamen>)
               IN '<Zielpfad>'
SELECT       <Q-Faktentabelle>.<Q-Faktschlüsselattributnamen>,
               <Aggregatsfunktion p>(<Q-Faktentabelle>
                  .<Q-Faktattributnamen s>)
FROM         <Q-Faktentabelle> IN '<Quellpfad>';
WHERE        <Q-Faktentabelle>.<Zeitstempelattribut>
               > <LetzteAktualisierung>
GROUP BY     <Q-Faktentabelle>.<Q-Faktschlüsselattributnamen>;
```

Ist hingegen die Granularität des Zeitstempelattributs in der Faktentabelle der Zieldatenbank (zum Beispiel *Monat*) kleiner als in der Quelldatenbank (zum Beispiel *Tag*), dann kann es notwendig sein, die neuen Faktwerte mit den bereits vorhandenen Faktwerten zu *verrechnen*. Der folgende Aktualisierungsalgorithmus basiert auf der *Summary-Delta Table Method*, welche die Aktualisierungsmethode in Abhängigkeit der bereits vorhandenen Faktdaten in der Zieldatenbank wählt.[111]

[111] Vgl. Mumick et al. 1999: S. 400

```
für jeden neuen Datensatz NDS aus der Q-Faktentabelle          (1)
   wähle den Datensatz DS aus der Faktentabelle, der die       (2)
      gleichen "Group by"-Attribute wie NDS hat
   falls DS nicht gefunden wird, dann                          (3)
      füge NDS in die Faktentabelle ein                        (4)
   sonst (5)
      für alle Aggregatsfunktionen AF in der                   (6)
         Faktentabelle
         falls weder in DS noch in NDS ein Faktwert            (7)
            existiert, dann
            DS.AF = NULL                                       (8)
         sonst                                                 (9)
            falls AF = COUNT oder AF = SUM, dann              (10)
               DS.AF = DS.AF + NDS.AF                         (11)
            falls AF = MIN, dann                              (12)
               DS.AF = MIN(DS.AF, NDS.AF)                     (13)
            falls AF = MAX, dann                              (14)
               DS.AF = MAX(DS.AF, NDS.AF)                     (15)
```

Erläuterungen:

Im Gegensatz zur zuvor verwendeten Anfügeabfrage (`INSERT INTO`), die alle neuen Datensätze mit einer einzigen SQL-Anweisung in die Faktentabelle der Zieldatenbank einfügt, erfolgt die Verarbeitung hier datensatzweise.

`(7)-(8)` Wenn zu einem Faktattribut weder der neue noch der bestehende Datensatz einen Wert enthält, dann ist auch der aktualisierte Wert gleich Null.

`(10)-(15)` Der Algorithmus unterscheidet die Aggregatsfunktionen `COUNT`, `SUM`, `MIN` und `MAX` und bestimmt für jeden Fakt den neuen (aktualisierten) Wert.

Da der Prototyp unterschiedliche Granularitäten der Zeitstempelattribute in der Quell- und Zieldatenbank nicht vorsieht, nutzt er diesen laufzeit*in*effizienteren Algorithmus nicht. Zur Illustration der Implementierbarkeit ist er dennoch im Quellcode enthalten (auskommentiert).

Inkrementelle Aktualisierung der temporalen Aggregatstabellen

Die Fortschreibungsalgorithmen der Faktentabelle lassen sich auch auf die temporalen Aggregatstabellen übertragen. Im Gegensatz zur Faktentabelle sind bei der Aktualisierung der Aggregatstabellen nicht nur eine sondern mehrere Tabellen bei der Auswahl der neuen Quelldaten zu berücksichtigen. Der Aufbau des Abfragestrings wird dadurch zwar komplexer, ändert sich in seiner Grundstruktur jedoch nicht:

```
INSERT INTO <Aggregatstabelle j>(<Attributnamen>)
               IN '<Zielpfad>'
SELECT       <Q-Tabellennamen>.<Q-Attributnamen>,
               <Aggregatsfunktion p>(<Q-Faktentabelle>
                  .<Q-Faktattributnamen s>)
FROM         <Q-Faktentabelle>, <Q-Dimensionstabellen>
               IN '<Quellpfad>'
WHERE        <Q-Dimensionstabelle i>.<Q-Dimensionsschlüssel i>
               = <Q-Faktentabelle>
                    .<Q-Faktentabellenfremdschlüssel i>
   AND       <Q-Faktentabelle>.<Zeitstempelattribut>
               > <LetzteAktualisierung>
GROUP BY     <Q-Tabellennamen>.<Q-Attributnamen>;
```

Da bei der Bestimmung der Aggregate immer auch Dimensionsdaten involviert sind, muss der Aktualisierungsalgorithmus auch überprüfen, ob die Dimensionstabellen zeitbezogen gespeichert sind. Vorhandene Zeitstempel nutzt er, um den gewünschten Zeitbezug der Dimensionsdaten herzustellen (faktbezogene historische Sicht). Die Bedingungen (**WHERE**) der obigen Anfügeabfrage werden dazu mit der folgenden Anweisung ergänzt:

```
...
   AND       <Q-Dimensionstabelle i>.GültigAb
               <= <Q-Faktentabelle>.<Zeitstempelattribut>
   AND       (<Q-Dimensionstabelle i>.GültigBis
               > <Q-Faktentabelle>.<Zeitstempelattribut>
               OR <Q-Dimensionstabelle i>.GültigBis IS NULL)
...
```

3.2.3 Analyse des erweiterten Modells

Nach der Definition und Wartung des aggregierten Faktenmodells stehen die Daten für den Endbenutzerzugriff bereit. Zur Abgrenzung der Endbenutzersicht von der Entwicklersicht wurde die Funktion *Modell analysieren* in einer eigenständigen MS Excel-Anwendung implementiert. MS Excel bietet sich an, weil sich damit dieselben Datenzugriffskomponenten wie in MS Visual Basic (ADO) nutzen lassen, jedoch die Datenausgabe in die bereits vorhandenen Tabellenblätter und die individuelle Weiterverarbeitung einfacher sind.

Das implementierte Endbenutzerwerkzeug ermöglicht dem Anwender, die gewünschten Abfrageparameter über Kombinationsfelder auszuwählen und die Abfrage anschliessend auszuführen. Abbildung 3.2.12 zeigt die Endbenutzerschnittstelle. Da bei der Entwicklung des Prototyps die Abfragegenerierung im Vordergrund steht, fehlen der Benutzerschnittstelle bestimmte Funktionen wie etwa *Drilling Up and Down* , die der Anwender bei einem OLAP-Werkzeug erwarten würde.

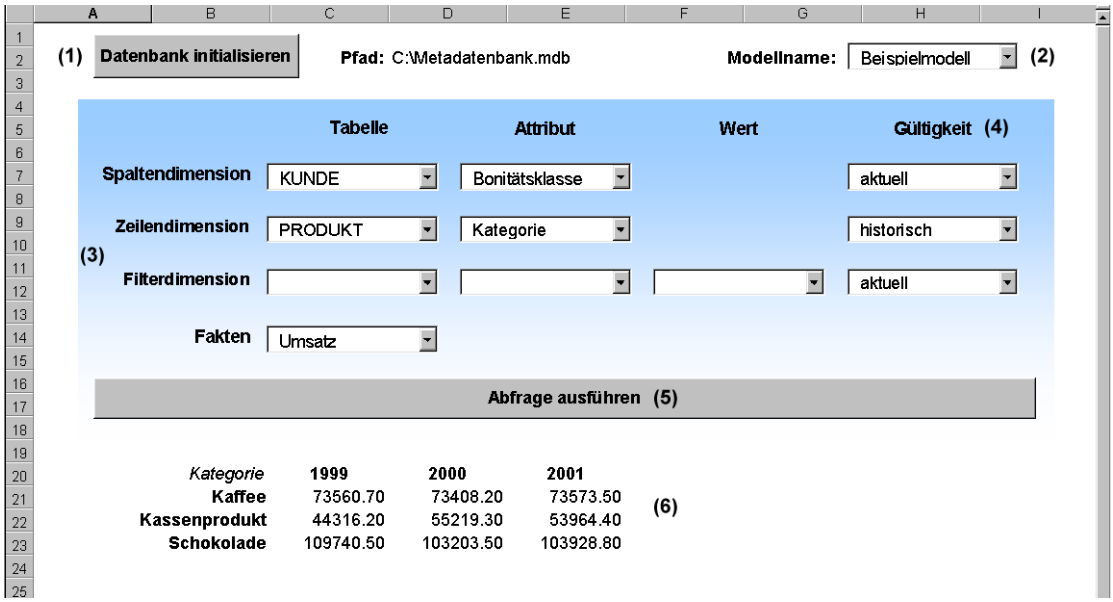


Abbildung 3.2.12: Endbenutzerschnittstelle des Prototyps

Beispielsitzung

1. Der Anwender bestimmt den Pfad der Metadatenbank in Zelle *D2*. Nach dem Drücken der Befehlsschaltfläche *Datenbank initialisieren* liest der Prototyp alle Modellnamen aus der Metadatenbank aus und schreibt sie in das Kombinationsfeld *Modellname*.
2. Nach der Auswahl des gewünschten Modells ermittelt der Prototyp alle im Zielmodell enthaltenen Dimensionen und Fakten und schreibt die Werte in die Kombinationsfelder unter der Überschrift *Tabelle*.

3. Auf die gleiche Art stellt er dem Anwender die restlichen Parameter *Attribut* und *Wert* zur Verfügung. Während die erste Dimension die Spaltenwerte bestimmt, definiert die zweite Dimension die Zeilenwerte. Die Möglichkeit zur Nutzung von Filtern (Slicing) wird mit einer dritten Dimension als Beispiel illustriert. Da die Festlegung eines Vergleichswerts nur bei der Filterdimension notwendig ist, fehlt dieses Kombinationsfeld bei den anderen Dimensionen.

4. Die Kombinationsfelder unter der Überschrift *Gültigkeit* repräsentieren die zentrale Erweiterung aus der Benutzersicht. Der Anwender entscheidet damit für jede Dimension selbständig, ob die Abfrage die Daten gemäss der aktuellen oder der faktbezogenen historischen Sicht ermitteln soll.

5. Die Abfrage wird durch das Drücken der Befehlsschaltfläche *Abfrage ausführen* initiiert.

6. Der Prototyp schreibt die gefundenen Werte der gewählten Spalten- und Zeilendimension sowie die entsprechenden Faktwerte in das Tabellenblatt. Der Anwender kann anschliessend die Parameter beliebig anpassen (3) oder eine auf einem anderen Modell basierende Abfrage definieren (1).

Die Anwendung führt die Abfrage abhängig von der gewünschten temporalen Auswertungsform aus. Der folgende Algorithmus zeigt den bei *Abfrage ausführen* angesprochenen Ablauf des Abfrageprozesses:

```
bestimme die zeitabhängigen Attribute ZA, die in der          (1)
   faktbezogenen historischen Sicht abzubilden sind
falls ZA ≠ ∅ ist, dann                                        (2)
   wähle die temporale Aggregatstabelle TA, bei der gilt      (3)
      ZA ⊆ TA und TA - ZA ⊄ Menge aller zeitabhängiger         (4)
         Dimensionsattribute in einem Modell
   führe die Abfrage mit TA und den restlichen in der         (5)
      Abfrage involvierten Dimensionstabellen aus
sonst                                                         (6)
   führe die Abfrage mit der Faktentabelle und allen          (7)
      in der Abfrage involvierten Dimensionstabellen aus
```

Erläuterungen

(2) Wenn die Abfrage keine zeitabhängigen oder zeitabhängig abzubildenden Dimensionsattribute enthält, lässt sie sich unabhängig von der Einstellung der temporalen Auswertungsform auch ohne die Verwendung temporaler Aggregatstabellen beantworten.

(4) Es ist möglich, dass mehrere temporale Aggregatstabellen die betreffenden zeitabhängigen Attribute enthalten. In diesem Fall ist die Aggregatstabelle auszuwählen, die lediglich die von der Abfrage benötigten zeitabhängigen und in der faktbezogenen historischen Sicht darzustellenden Attribute enthält. Damit wird sichergestellt, dass sich die Aggregatstabelle auch mit Dimensionsattributen verknüpfen lässt, die nicht zeitabhängig sind.

(5) Die Anwendung benötigt zur Beantwortung der Abfrage nur noch Dimensionstabellen, deren Attribute nicht bereits durch die temporale Aggregatstabelle abgebildet sind.

Beispiel

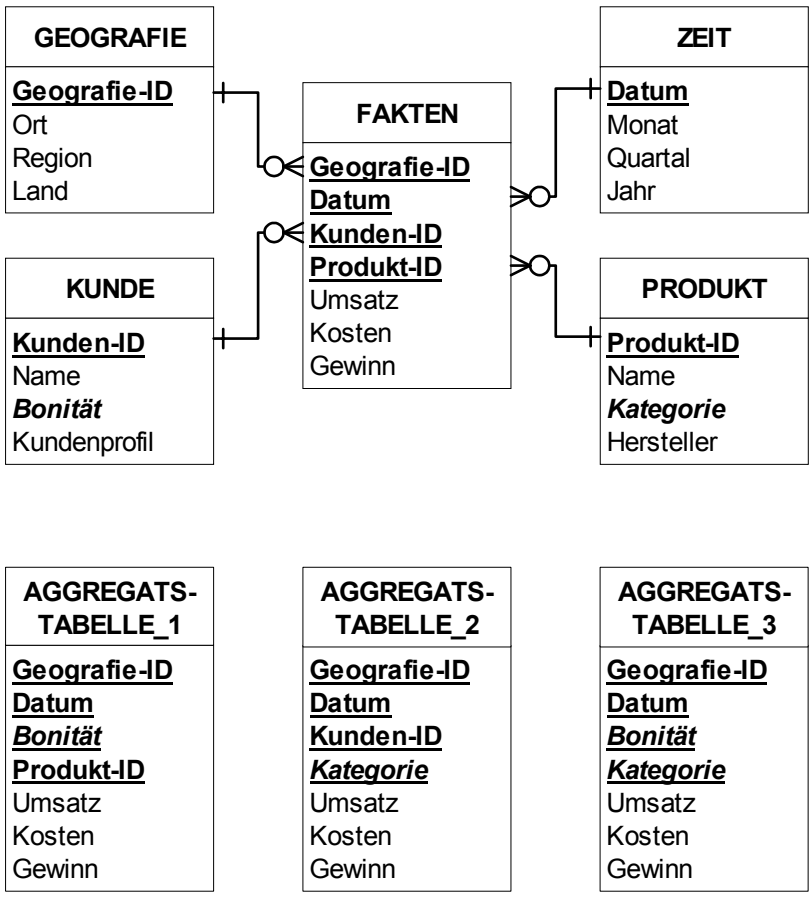


Abbildung 3.2.13: Modell mit zwei zeitabhängigen Dimensionsattributen

Gegeben sei das Schema in Abbildung 3.2.13. Die Attribute Bonität (KUNDE) und Kategorie (PRODUKT) sind mit Hilfe der drei temporalen Aggregatstabellen zeitabhängig abgebildet. In diesem Beispiel soll die Anwendung den Umsatz in Abhängigkeit der vorhandenen Jahre und Produktkategorien in der faktbezogenen historischen Sicht abbilden. Diese Abfrage enthält ein zeitabhängiges Dimensionsattribut (Kategorie), das sowohl in der zweiten als auch in der dritten Aggregatstabelle vorhanden ist. Gestützt auf die zuvor definierte Regel wählt die Anwendung die zweite Aggregatstabelle, weil diese neben der Produktkategorie keine weiteren zeitabhängigen Dimensionsattribute enthält. Diese Wahl ist bei der Beispielabfrage zunächst nicht nachvollziehbar, denn die Fakten sind in der dritten Aggregatstabelle stärker zusammengefasst und wären deshalb effizienter analysierbar. Die zweite Aggregatstabelle hat demgegenüber den Vorteil, dass sie sich über das Fremdschlüsselattribut Kunden-ID auch mit den zeit*un*abhängigen Attributen der Kundendimension verknüpfen lässt und deshalb universeller als die dritte Aggregatstabelle einsetzbar ist. Die gleiche Abfrage, jedoch mit der zusätzlichen Bedingung, dass der Umsatz nur für den Kunden *Meier* gesucht wird, kann die Anwendung daher nur mit der zweiten Aggregatstabelle sowie den Dimensionen ZEIT und KUNDE beantworten. Die hier angewandte Regel zur Ermittlung der geeigneten Aggregatstabelle ist somit nicht immer die effizienteste, führt jedoch stets zum richtigen Resultat.

(7) Der Prototyp verwendet an dieser Stelle immer die Faktentabelle, weil er keine konventionellen Aggregate unterstützt. Ein kommerzielles OLAP-Werkzeug würde gegebenenfalls auch überprüfen, ob die Abfrage mit konventionellen Aggregaten und damit effizienter beantwortbar wäre.

Abfrage basiert auf der Faktentabelle

Die Abfrage basiert auf der Faktentabelle, wenn der Anwender bei allen Dimensionen die Abbildung der aktuellen Sicht wünscht oder die in der faktbezogenen historischen Sicht darzustellenden Dimensionsattribute nicht zeitbezogen definiert sind.

Der Prototyp nutzt die von MS Access unterstützte Kreuztabellenabfrage[112], damit er das Resultat direkt in der gewünschten Form auf dem Bildschirm ausgeben kann. Der zur Ausführung der gewünschten Abfrage benötigte SQL-String enthält einerseits die vom Benutzer bereits definierten Werte, anderer-

[112] Vgl. auch Kapitel 2.3.1: S. 81

seits die mit Hilfe der Metadatenbank zu bestimmenden Parameter <Aggregatsfunktion>, <Faktentabelle> und <Fremdschlüssel>:

```
TRANSFORM <Aggregatsfunktion>(<Faktentabelle>.<Fakt>)
SELECT    <Zeilendimension>.<Zeilendimensionsattribut>
FROM      <Faktentabelle>, <Zeilendimension>,
             <Spaltendimension>
WHERE     <Spaltendimension>.<Spaltendimensionsschlüssel>
             = <Faktentabelle>.<F-Schlüssel Spaltendimension>
  AND     <Zeilendimension>.<Zeilendimensionsschlüssel>
             = <Faktentabelle>.<F-Schlüssel Zeilendimension>
  AND     <Filterdimension>.<Filterdimensionsschlüssel>
             = <Faktentabelle>.<F-Schlüssel Filterdimension>
  AND     <Filterdimension>.<Filterdimensionsattribut>
             = <Filterwert>
GROUP BY  <Zeilendimension>.<Zeilendimensionsattribut>
PIVOT     <Spaltendimension>.<Spaltendimensionsattribut>;
```

Abkürzung: `F-Schlüssel = Fremdschlüssel`

Falls die Datenbank Kreuztabellenabfragen nicht unterstützt, kann die Anwendung die gesuchten Resultate auch mit Hilfe herkömmlicher Auswahlabfragen bestimmen. Dazu ermittelt sie in jeweils einer Abfrage die Spalten- und Zeilendimensionselemente:

```
SELECT DISTINCT <Dimension>.<Dimensionsattribut>
FROM      <Dimension>
ORDER BY <Dimension>.<Dimensionsattribut>;
```

Die Anwendung schreibt die Werte des Spaltendimensionsattributs in die erste Ausgabezeile und die Werte des Zeilenattributs in die erste Ausgabespalte. Dabei speichert sie die Werte beider Dimensionen in jeweils einem Auflistungsobjekt[113].

[113] MS Visualbasic: *Collection*

Die dritte Abfrage bestimmt die Faktwerte in Abhängigkeit der ausgewählten Dimensionskriterien:

```
SELECT <Spaltendimension>.<Spaltendimensionswert>,
          <Zeilendimension>.<Zeilendimensionswert>,
          <Aggregatsfunktion>(<Faktentabelle>.<Fakt>)
FROM   <Faktentabelle>, <Zeilendimension>, <Spaltendimension>
WHERE  <Spaltendimension>.<Spaltendimensionsschlüssel>
          = <Faktentabelle>.<F-Schlüssel Spaltendimension>
  AND     <Zeilendimension>.<Zeilendimensionsschlüssel>
             = <Faktentabelle>.<F-Schlüssel Zeilendimension>
  AND     <Filterdimension>.<Filterdimensionsschlüssel>
             = <Faktentabelle>.<F-Schlüssel Filterdimension>
  AND     <Filterdimension>.<Filterdimensionsattribut>
             = <Filterwert>
GROUP BY <Spaltendimension>.<Spaltendimensionswert>,
            < Zeilendimension>.<Zeilendimensionswert>;
```

Die Anwendung ermittelt mit Hilfe der zuvor definierten Auflistungsobjekten die Ausgabeposition (Spalte/Zeile) jedes Faktwerts und schreibt diesen in die entsprechende Zelle des Tabellenblatts.

Abfrage basiert auf einer temporalen Aggregatstabelle

Das aggregierte Faktenmodell ermöglicht es, die temporale Auswertungsform für jede Dimension unabhängig festzulegen. Der Prototyp ermittelt das Abfrageresultat nur dann mit Hilfe einer temporalen Aggregatstabelle, wenn der Anwender die faktbezogene historische Sicht zu mindestens einem zeitabhängigen Dimensionsattribut verlangt. Das Endbenutzerwerkzeug bestimmt deshalb zunächst, welche der in der faktbezogenen historischen Sicht abzubildenden Attribute zeitabhängig sind. Die Zeilen `(4)`, `(5)` oder `(6)` entfallen in der folgenden SQL-Abfrage, falls das entsprechende Dimensionsattribut (`<Spaltendimensionsattribut>`, `<Zeilendimensionsattribut>`, `<Filterdimensionsattribut>`) in der aktuellen Sicht abzubilden ist.

```
SELECT Attributname                                              (1)
FROM   ATTRIBUT                                                 (2)
WHERE  Modellname = <Modellname>                                 (3)
  AND  (Attributname = <Spaltendimensionsattribut>               (4)
         OR Attributname = <Zeilendimensionsattribut>            (5)
         OR Attributname = <Filterdimensionsattribut>)           (6)
  AND  Zeitabhängig = TRUE                                       (7)
  AND  Zielattribut = TRUE;                                      (8)
```

Wenn die Anwendung ein zeitabhängiges Dimensionsattribut findet, dann sucht sie nach der richtigen Aggregatstabelle. Diese Tabelle hat die Eigenschaft, dass sie alle benötigten, aber keine weiteren zeitabhängigen Dimensionsattribute enthält. Anhand der zuvor definierten Abfrage kennt die Anwendung bereits die Anzahl der zeitabhängigen Dimensionsattribute <c>[114] und deren Namen <1. bis c. zeitabhängiges Attribut>. Die folgende Abfrage bestimmt die gesuchte temporale Aggregatstabelle:

```
SELECT ATTRIBUT_<Anzahl zeitabh. Attribute c>.Tabellenname
FROM   TABELLE, ATTRIBUT, ATTRIBUT AS ATTRIBUT_1, ...,
          ATTRIBUT AS ATTRIIBUT_<c+1>
WHERE  ATTRIBUT.Tabellenname = ATTRIBUT_1.Tabellenname
       ...
  AND  ATTRIBUT_<c-1>.Tabellenname = ATTRIBUT_<c>.Tabellenname
  AND  ATTRIBUT_<c>.Attributname = ATTRIBUT_<c+1>.Attributname
  AND  ATTRIBUT_1.Attributname = <1. zeitabhängiges Attribut>
       ...
  AND  ATTRIBUT_<c-1>.Attributname = <c. zeitabh. Attribut>
  AND  ATTRIBUT.Modellname = TABELLE.Modellname
  AND  ATTRIBUT.Tabellenname = TABELLE.Tabellenname
  AND  TABELLE.Ausprägung = "Temporale_Aggregate"
GROUP BY ATTRIBUT_<c>.Tabellenname
HAVING COUNT(ATTRIBUT_<c>.Attributname) = <c>;
```

Zunächst untersucht die Abfrage für jedes der zuvor ermittelten zeitabhängigen Dimensionsattribute, in welchen Tabellen es vorkommt. Weil die Ergebnisse über das Attribut Tabellenname verknüpft sind, resultieren nur Tabellenna-

[114] Die Anzahl der Datensätze lässt sich zum Beispiel mit der Methode recordcount des ADO-Objekts Recordset ermitteln.

men, die alle der zuvor ermittelten zeitabhängigen Dimensionsattribute enthalten. Mit der Anweisung **HAVING COUNT**(ATTRIBUT_<c>.Attributname) = <c> stellt die Abfrage zugleich sicher, dass die Anzahl der zeitabhängigen Dimensionsattribute in der gesuchten Tabelle gleich ist, wie in der ursprünglichen Abfrage.

Neben der richtigen Aggregatstabelle sind auch noch die restlichen Abfrageparameter mit Hilfe der Metadatenbank zu bestimmen:

- Dimensionen, die keine zeitabhängigen oder zeitabhängig abzubildende Attribute enthalten
- Verbundattribute dieser Dimensionstabellen

Die Anwendung definiert anschliessend eine Kreuztabellenabfrage, wie dies bereits bei der Abfrage gezeigt wurde, die auf der Faktentabelle basiert. Anstelle der Faktentabelle nutzt sie jedoch die gerade gefundene temporale Aggregatstabelle.

Zusammenfassung

Die Implementation zeigt, *wie* sich das aggregierte Faktenmodell realisieren lässt. Bei der Entwicklung des Prototyps wurden zwei Anwendungen erstellt. Die erste bildet mit den Funktionen zur Definition und Aktualisierung der temporalen Aggregate die *Entwicklersicht* ab. Die zweite repräsentiert die *Endbenutzersicht* und beinhaltet die Analysefunktionen, die für den automatisierten Zugriff auf das erweiterte Datenmodell notwendig sind.

Der Prototyp verwaltet sämtliche *Metadaten*, die zur Definition, Aktualisierung und Analyse der temporalen Aggregate erforderlich sind, in einer getrennten Datenbank. Einerseits können nicht alle Metadaten des aggregierten Faktenmodells in die bestehenden Metadatenstrukturen der analytischen Datenbank integriert werden, andererseits vereinfacht die zentrale Verwaltung den Zugriff auf die benötigten Metadaten.

Bei der *Modelldefinition* extrahiert der Prototyp die Metadaten, die das Sternschema der analytischen Datenbank beschreiben. Nachdem der Systementwickler die zeitabhängig abzubildenden Dimensionsattribute identifiziert hat, bestimmt der Prototyp die notwendigen temporalen Aggregatstabellen. Diese werden zusammen mit der Faktentabelle und den Dimensionstabellen in einer neuen Datenbank erstellt. Die Übernahme des bestehenden Sternschemas in eine neue Datenbank ist nicht modellbedingt, sondern auf den eingeschränkten Funktionsumfang des Prototypen zurückzuführen. In einem bestehenden OLAP-System liessen sich Aggregatstabellen auch in der Ausgangsdatenbank verwalten.

Der Prototyp lädt die Daten bei der *Aktualisierung* vollständig oder inkrementell in die mit temporalen Aggregatstabellen erweiterte Zieldatenbank. Die bei der *vollständigen* Aktualisierung genutzten SQL-Strings sind statisch und werden bereits bei der Modelldefinition in der Metadatenbank gespeichert. Die *inkrementelle* Aktualisierung verwendet hingegen Abfragen, die Bezug auf das letzte Aktualisierungsdatum nehmen und sich deshalb erst zur Laufzeit generieren lassen. Der Prototyp geht davon aus, dass die Granularität der Zeitstempelattribute in der Quell- und Zieldatenbank identisch sind, weil dies die Nutzung einfacherer und effizienterer Aktualisierungsalgorithmen ermöglicht.

Vor der Ausführung einer endbenutzerdefinierten *Datenanalyse* ermittelt der Prototyp, welche Dimensionsattribute den Fakten in der faktbezogenen historischen Sicht zuzuordnen sind. Anhand dieser Attribute und der gegebenen Abfrageparameter bestimmt der Prototyp die geeignete temporale Aggregats- oder Faktentabelle sowie die Dimensionstabellen. Damit sich zeitunabhängige, nicht

zeitabhängig darzustellende und zeitabhängige Dimensionsattribute in einer Abfrage kombinieren lassen, wird mit Hilfe der Metadatenbank die Aggregatstabelle bestimmt, die ausser den in der faktbezogenen historischen Sicht abzubildenden keine weiteren zeitabhängigen Dimensionsattribute enthält. Verwendet die Abfrage keine zeitabhängigen oder keine in der faktbezogenen historischen Sicht darzustellenden Dimensionsattribute, dann greift der Prototyp auf die detaillierte Faktentabelle zurück, weil diese den Bezug zu allen aktuellen Dimensionsdaten ermöglicht.

3.3 Diskussion

Mit dem aggregierten Faktenmodell hat die vorliegende Arbeit einen Ansatz präsentiert, der die temporale Funktionalität eines OLAP-Systems erweitert und damit den Anwendern eine bessere Qualität ihrer Datenanalysen ermöglicht. Die folgende Diskussion beurteilt das aggregierte Faktenmodell und vergleicht es mit den anderen bereits vorgestellten Ansätzen. Der Ausblick auf die weiterführenden Arbeiten fasst die zentralen Probleme zusammen und zeigt damit, welche Punkte bei einer Weiterentwicklung besonders zu beachten sind. Der letzte Abschnitt fasst die zentralen Erkenntnisse dieser Arbeit zusammen und erörtert mögliche Anwendungsbereiche des aggregierten Faktenmodells.

3.3.1 Beurteilung

Abbildung 3.3.1 zeigt einen Vergleich der in Kapitel 2.3 vorgestellten Methoden zur Abbildung zeitbezogener Dimensionsdaten und dem aggregierten Faktenmodell. Im Gegensatz zum Vergleich in Abbildung 2.3.19 werden die *Slowly Changing Dimensions*-Konzepte Typ 1 und 2 hier nicht getrennt, sondern gemeinsam als eine einzige Modellierungsalternative betrachtet. Die Eigenschaften der vorgestellten Alternativen lassen sich besser vergleichen, weil dadurch alle Modellansätze beim Kriterium *temporale Auswertungsformen* mindestens die aktuelle *und* die faktbezogene historische Sicht unterstützen.

Die gemeinsame Anwendung des SCD Typ 1 und 2 erfordert die Erstellung zweier getrennter OLAP-Würfel. Dieser Ansatz wird in Abbildung 3.3.1 entsprechend neu bewertet:[115]

- *Temporale Auswertungsformen*: Mit der Definition von zwei OLAP-Würfeln kann der Anwender sowohl eine Abfrage in der aktuellen als auch in der faktbezogenen historischen Sicht definieren.
- *Temporale Funktionalität*: Wie bei der Einzelbewertung unterstützt der SCD-Ansatz keine der weitergehenden temporalen Funktionalitäten. Die gleichzeitige Definition verschiedener temporaler Auswertungsformen ist in einer Abfrage nicht möglich, weil sich sowohl Fakten als auch Dimensionen in den beiden OLAP-Würfeln inhaltlich unterscheiden und damit nicht adäquat verknüpfbar sind.

[115] Der SCD Typ 3 wurde nicht mehr berücksichtigt.

- *Benutzerfreundlichkeit*: Die Verständlichkeit des Modells und die Abfrageeffizienz unterscheiden sich bei der parallelen Nutzung der Datenwürfel nicht von der Einzelbewertung. Konventionelle Aggregate werden sowohl beim SCD Typ 1 als auch beim Typ 2 unterstützt und ermöglichen eine effiziente Abfrageverarbeitung. Hierbei ist zu beachten, dass Aggregate beim SCD Typ 2 nicht konventionellen, sondern temporalen Aggregaten entsprechen, weil sie die faktbezogene historische Sicht abbilden.
- *Systemanforderungen/Wartung*: Bei diesem Kriterium ändert sich die Bewertung der Speichereffizienz. Während die Anwendung einer SCD-Methode in *einem* Würfel relativ speichereffizient ist, benötigen *zwei* Würfel entsprechend mehr Speicherplatz und erhalten im Vergleich zu den anderen Modellierungsmethoden eine unterdurchschnittliche Beurteilung.

Entsprechend dem Vergleich in Abbildung 2.3.19 zeigt Abbildung 3.3.1 die von den Modellierungsmethoden unterstützte Funktionalität (✓/**?**) und bewertet Kriterien zur Benutzerfreundlichkeit und den System-/Wartungsanforderungen (**+**/**O**/**-**).

Temporale Auswertungsformen

Der Vergleich zeigt unter dem Kriterium *temporale Auswertungsformen*, welche der in Kapitel 2.1.2 beschriebenen temporalen Auswertungsformen von den Modellierungsansätzen unterstützt werden.

1. Aktuelle Sicht

Das aggregierte Faktenmodell bildet die aktuelle Sicht durch das Überschreiben der bestehenden, nicht mehr gültigen Dimensionsdaten ab. Das Vorgehen entspricht der Aktualisierungsmethode, die auch beim SCD Typ 1 angewandt wird.

2. Faktbezogene historische Sicht

Das aggregierte Faktenmodell ergänzt ein bestehendes Modell mit temporalen Aggregaten. Diese bilden alle möglichen Abfragekonstellationen mit Beteiligung zeitabhängiger Dimensionsattribute in der faktbezogenen historischen Sicht ab.

Kriterien	Zeitstempelmethode in konventionellen Dimensionen	Zeitstempelmethode in rekursiven Dimensionen	SCD Typ 1 und Typ 2	Alternativer Fremdschlüssel	Zeitschalterverfahren	Aggregiertes Faktenmodell
Temporale Auswertungsformen						
Aktuelle Sicht	✓	✓	✓	✓	✓	✓
Faktbezogene historische Sicht	✓	✓	✓	✓	✓	✓
Dimensionsbezogene historische Sicht	✓	✓				
Ursprüngliche Sicht	?	?	?			
Temporale Funktionalität						
Verschiedene Sichten in einer Abfrage	✓	✓		✓	✓	✓
Dimensionsdaten temporal analysierbar	✓	✓				
Temporale Differenzierung der Attribute						✓
Zeitbezogene Dimensionsstruktur		✓				
Benutzerfreundlichkeit						
Verständlichkeit des Modells	O	-	O	O	-	O
Abfrageeffizienz	O	-	+	+	-	+
Systemanforderungen / Wartung						
Implementierbarkeit in OLAP-Systemen	-	-	+	+	O	-
Speichereffizienz	O	O	-	-	-	-
Inkrementelle Aktualisierung	+	+	+	+	+	+
Komplexität der Aktualisierungsprozesse	O	-	+	-	O	+

Abbildung 3.3.1: Das aggregierte Faktenmodell im Vergleich

Die folgenden Abschnitte besprechen die in Abbildung 3.3.1 dargestellte Bewertung, welche das aggregierte Faktenmodell mit den anderen Modellierungsalternativen vergleicht.

3. Dimensionsbezogene historische Sicht

Die dimensionsbezogene historische Sicht erfordert eine vollständige Historisierung aller zeitabhängigen Dimensionsdaten wie etwa bei der Zeitstempelmethode. Das aggregierte Faktenmodell enthält jedoch lediglich aktuelle Dimensionsdaten und unterstützt diese temporale Auswertungsform daher nicht.

4. Ursprüngliche Sicht

In der vorgestellten Form unterstützt das aggregierte Faktenmodell die ursprüngliche Sicht nicht. Ist diese temporale Auswertungsform dennoch gewünscht, skizzieren die folgenden Strategien mögliche Handlungsalternativen:

1. Das OLAP-Werkzeug ignoriert Änderungen in den Dimensionsdaten und fügt lediglich neue Datensätze hinzu. Diese Vorgehensweise bildet die ursprüngliche Sicht auf Kosten der aktuellen Sicht ab.
2. Die Dimensionstabellen enthalten für jede Dimensionskategorie ein Attribut, das den aktuellen und ein zweites, das den ursprünglichen Zustand abbildet (vgl. SCD Typ 3). Attribute der zweiten Kategorie behalten ihren ersten Zustand bei und sollten deshalb im aggregierten Faktenmodell nicht als zeitabhängig definiert werden. Die Erstellung ganzer Dimensionen anstelle des Hinzufügens zusätzlicher Attribute ist nicht zu empfehlen, weil die Grösse der temporalen Aggregatstabellen unter anderem von der Dimensionsanzahl abhängt.

Die zur Unterstützung der ursprünglichen Sicht notwendigen Anpassungen im aggregierten Faktenmodell sind klein, wirken sich aber auf die Beurteilung der Benutzerfreundlichkeit (Verständlichkeit des Modells) und der Systemanforderungen aus (grössere Dimensions- und Aggregatstabellen, grösserer Aktualisierungsaufwand). Abbildung 3.3.1 zeigt die Bewertung *ohne* die Berücksichtigung der ursprünglichen Sicht.

Temporale Funktionalität

Das Kriterium *temporale Funktionalität* beurteilt weitere temporale Modelleigenschaften, die bei den zuvor genannten temporalen Auswertungsformen noch nicht berücksichtigt wurden.

Verschiedene Sichten in einer Abfrage

Das aggregierte Faktenmodell kann die temporale Auswertungsform für alle in der Abfrage beteiligten Dimensionen individuell anwenden. Damit ist es zum Beispiel möglich, den Fakten die aktuellen Werte der Filterdimension (aktuelle

Sicht) sowie die zum Erfassungszeitpunkt der Fakten gültigen Werte der Spalten- und Zeilendimension (faktbezogene historische Sicht) in einer einzigen Abfrage zuzuordnen. Die Informationsqualität profitiert von dieser flexiblen Wahl der temporalen Auswertungsform, weil der Anwender damit den Zeitbezug in der Abfrage genauer auf seine Problemstellung abstimmen kann.

Dimensionsdaten temporal analysierbar

Temporale Dimensionsdatenanalysen in der Form *„Zu welcher Kategorie hat das Produkt Chnuschpi am 1.1.2001 gehört?“* unterstützt das aggregierte Faktenmodell nicht. Die temporalen Aggregatstabellen enthalten zwar den Zeitbezug einzelner Dimensionsdaten, jedoch sind nie mehrere Attribute derselben Dimension zusammen zeitbezogen abgebildet.

Auch wenn der vorgestellte Prototyp keine Möglichkeit zur temporalen Analyse der Dimensionsdaten bietet, kann die Zuordnung mehrerer zeitbezogener Attribute aus derselben Dimension untersucht werden. Dazu müssen die entsprechenden temporalen Aggregatstabellen untereinander und/oder mit der Faktentabelle verknüpft werden, wie dies der folgende SQL-Code anhand der zuvor genannten Beispielfrage zeigt.[116]

```
SELECT DISTINCT AGGREGATSTABELLE_2.Kategorie                    (1)
FROM FAKTEN, AGGREGATSTABELLE_2, PRODUKT                        (2)
WHERE PRODUKT.Name = "Chnuschpi"                                (3)
  AND FAKTEN.Datum = 1/1/2001                                   (4)
  AND FAKTEN.Produkt-ID = PRODUKT.Produkt-ID                    (5)
  AND FAKTEN.Datum = AGGREGATSTABELLE_2.Datum                   (6)
  AND FAKTEN.Kunden-ID = AGGREGATSTABELLE_2.Kunden-ID           (7)
  AND FAKTEN.Geografie-ID = AGGREGATSTABELLE_2.Geografie-ID     (8)
  AND FAKTEN.Umsatz = AGGREGATSTABELLE_2.Umsatz                 (9)
  AND FAKTEN.Kosten = AGGREGATSTABELLE_2.Kosten                (10)
  AND FAKTEN.Gewinn = AGGREGATSTABELLE_2.Gewinn;               (11)
```

Zeilen (6) bis (11) verknüpfen die Faktentabelle mit der Aggregatstabelle, die das Attribut Kategorie enthält. Da sich nicht alle Teilschlüssel der beiden Tabellen zuordnen lassen (Kategorie ≠ Produkt-ID) kann das Ergebnis mehrdeutig sein. Die Abfrage verknüpft deshalb auch die Faktattribute miteinander.

[116] Das Beispiel verwendet das Datenmodell wie es in Abbildung 3.2.13 dargestellt ist.

Die Schwächen dieses Vorgehens sind:

- die mögliche Mehrdeutigkeit, falls ein anderes Produkt mit denselben Verbundattributen verkauft wurde,
- das Fehlen des Resultats, falls das Produkt zum gewählten Zeitpunkt nicht verkauft wurde,
- die für viele Endbenutzer zu komplexe Abfragedefinition.

Temporale Differenzierung der Attribute

Mit der expliziten Identifikation der zeitabhängigen Dimensionsattributen und ihrer Abbildung in temporalen Aggregatstabellen ist das aggregierte Faktenmodell die einzige der vorgestellten Alternativen, die nicht alle Dimensionsdaten global zeitbezogen, sondern differenziert nach ihrer temporalen Eigenschaft abbildet.

Zeitbezogene Dimensionsstruktur

Das aggregierte Faktenmodell berücksichtigt veränderbare Dimensionsstrukturen nicht. Ein Ansatz zur Implementation dieser Funktionalität kann zum Beispiel über die zeitbezogene Speicherung der Metadaten führen, weil diese die gesamte Information über den Aufbau des aktuellen Datenmodells enthalten. Die zeitbezogene Speicherung der Metadaten würde es erlauben, vergangene Datenmodelle zu rekonstruieren, falls die entsprechenden Objektdaten immer noch in der Quelldatenbank verfügbar sind.

Benutzerfreundlichkeit

Das Kriterium *Benutzerfreundlichkeit* ist in OLAP besonders wichtig, weil nicht nur Spezialisten, sondern auch Fachbenutzer oder gar Gelegenheitsanwender Ad-hoc-Analysen mit dieser Werkzeugklasse erstellen sollen. Das gewählte Vorgehen zur Abbildung zeitabhängiger Dimensionsdaten in OLAP tangiert die Benutzerfreundlichkeit besonders bei der Verständlichkeit des Modells und der Abfrageeffizienz.

Verständlichkeit des Modells

Das aggregierte Faktenmodell ergänzt ein konventionelles Sternschema mit temporalen Aggregatstabellen, ohne dabei die bestehende Datenstruktur zu verändern. Die Endbenutzer werden dabei nicht mit diesen Erweiterungen konfrontiert, weil sowohl Verwaltung als auch Zugriff automatisiert sind. Sollte es

dennoch notwendig sein, die Daten ohne das Endbenutzerwerkzeug zu analysieren, dann können die automatisch generierten Tabellennamen sowie die Anzahl der Aggregatstabellen das Modellverständnis erschweren. In diesem Fall bietet sich die vorherige Konsultation der Metadatenbank an, mit der die gewünschten Datenobjekte einfach zu finden sind.

Abfrageeffizienz

Das aggregierte Faktenmodell bestimmt vor jeder Abfrage, aus welcher Tabelle die Fakten auszuwählen sind. Aufgrund des geringen Datenvolumens der Metadatenbank sind die dazu benötigten Abfragen sehr schnell und die zusätzliche Reaktionszeit wird von den Endbenutzern kaum als Verzögerung wahrgenommen. Die eigentliche Abfrage auf die Objektdaten ist ebenfalls laufzeiteffizient, weil das System bei der Definition der aktuellen Sicht auf die vorberechneten Werte der konventionellen Aggregate und bei der faktbezogenen historischen Sicht auf die Werte der temporalen Aggregate zurückgreifen kann.

Systemanforderungen

Nach den benutzerspezifischen Eigenschaften beurteilen die folgenden Kriterien systembezogene Merkmale des aggregierten Faktenmodells.

Implementierbarkeit in OLAP-Systemen

Kommerzielle OLAP-Systeme setzen zur Verbesserung der Abfrageeffizienz häufig konventionelle Aggregate ein. Obwohl bestimmte Funktionen bei der Verwaltung temporaler und konventioneller Aggregate ähnlich sind, lässt sich das aggregierte Faktenmodell nicht ohne Anpassung eines bestehenden Systems nutzen. Die folgenden Punkte erörtern die zentralen Unterschiede im Umgang mit konventionellen und temporalen Aggregaten.

- **Definition der Aggregatstabellen**

  Das OLAP-System definiert konventionelle Aggregatstabellen, um häufig benötigte oder zeitintensive Abfragen schneller zu verarbeiten. Zur Bestimmung der geeigneten Aggregatstabellen nutzt es die Vorgaben des Systementwicklers, berechnet die idealen Strukturen anhand vorgegebener Algorithmen oder analysiert das bisherige Verhalten der Endbenutzer. Temporale Aggregatstabellen werden hingegen anhand der temporalen Eigenschaft der Dimensionsattribute definiert. In einigen OLAP-Systemen ist es zwar möglich, Attribute als veränderbar zu bestimmen, diese Information wird

jedoch nicht zur Erstellung temporaler Aggregatstabellen, sondern meist zur Abbildung des SCD Typ 2 genutzt.

- **Aktualisierung der Aggregatstabellen**

  Sowohl konventionelle als auch temporale Aggregatstabellen lassen sich inkrementell und vollständig aktualisieren. Allerdings unterliegt die Auswahl der geeigneten Fortschreibungsmethode teilweise gerade entgegengesetzten Einschränkungen. Während das OLAP-System bestimmte Werte der konventionellen Aggregate bei Dimensionsdatenänderungen vollständig neu laden muss, können temporale Aggregate immer inkrementell aktualisiert werden. Die vollständige Fortschreibungsmethode steht bei temporalen Aggregaten dagegen nur zur Verfügung, wenn die Dimensionsdaten der Quelldatenbank zeitbezogen abgebildet sind.

- **Datenanalysen mit Aggregatstabellen**

  Das aggregierte Faktenmodell nutzt dieselben Grundfunktionen zur Auswahl der geeigneten Aggregatstabelle wie konventionelle OLAP-Systeme. Die Bestimmung der geeigneten Aggregatstabelle ist jedoch aufwändiger, wenn der Anwender die temporale Auswertungsform für jedes zeitabhängige Dimensionsattribut individuell und nicht in der gesamten Abfrage identisch definiert. In diesem Fall kann die Abfrage nicht wie in konventionellen Systemen die am stärksten zusammengefasste Sicht auf die Fakten verwenden, sondern muss die Aggregatstabelle auswählen, die so detailliert ist, dass sie sich noch mit den in der aktuellen Sicht darzustellenden Attributen der Dimensionstabellen verknüpfen lässt.

Speichereffizienz

Das aggregierte Faktenmodell definiert neue Tabellen, in welchen es verdichtete Faktwerte abspeichert. Der Speicherverbrauch dieser Aggregatstabellen hängt von der Anzahl der zeitabhängigen Attribute pro Dimension und der Anzahl Dimensionen mit zeitabhängigen Attributen ab. Diese Kriterien bestimmen die Anzahl der notwendigen Aggregatstabellen m, die sich mit der folgenden Formel berechnen lässt:[117]

[117] Vgl. Kapitel 3.2.1, S. 131

$$m = \prod_{i=1}^{n} (r_i + 1) - 1$$

n = Anzahl Dimensionen

r_i = Anzahl zeitabhängiger Attribute der Dimension i

Abbildung 3.3.2 zeigt den Speicherverbrauch in Abhängigkeit der beiden genannten Faktoren. Zur Messung des Speicherverbrauchs wurde eine Beispieldatenbank in MS Access entsprechend dem in Abbildung 2.3.1 dargestellten Sternschema erstellt. Die Faktentabelle enthält 100000, die Geografiedimension 14, die Kundendimension 31, die Produktdimension 15 und die Zeitdimension 1096 Datensätze.

r(i)	Eine Dimension	Drei Dimensionen
0	9.4	9.4
1	18.1	72.6
2	25.1	195.3
3	30.3	415.0

Speicherverbrauch (in Megabytes)

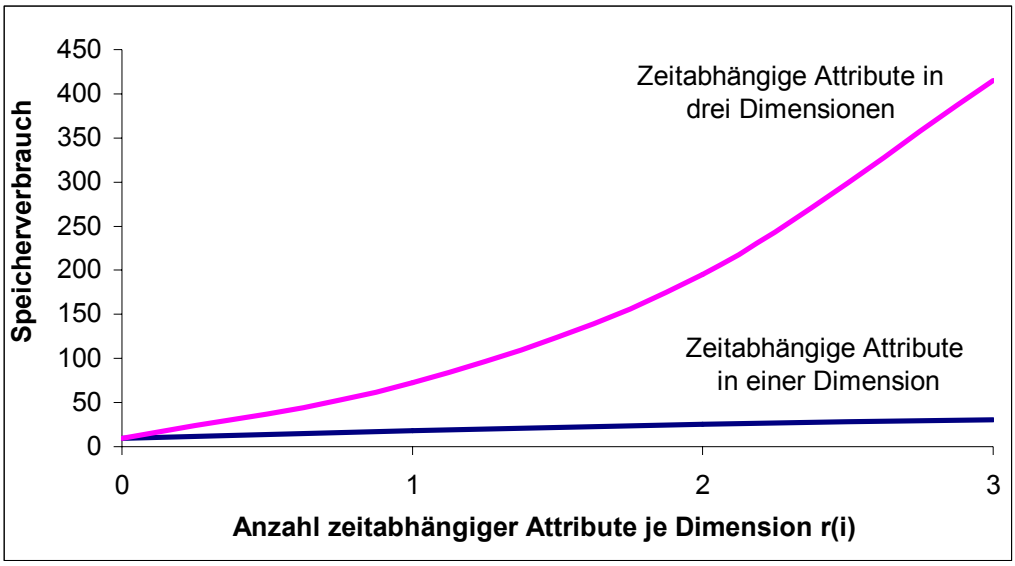


Abbildung 3.3.2: Speicherverbrauch temporaler Aggregate

Die untere Kurve zeigt eine annähernd lineare Zunahme des Datenvolumens, die durch die Definition zeitabhängiger Attribute in *einer* Dimension entsteht.[118] Die abnehmende Steigung der Kurve ist darauf zurückzuführen, dass die Attribute entlang der Dimensionshierarchie vom schwächsten bis zum

[118] Im Beispiel wurden die Attribute der Produktdimension als zeitabhängig definiert.

stärksten zusammenfassenden Attribut ausgewählt wurden. Die obere Kurve zeigt den überproportionalen Speicherverbrauch, der durch die Definition zeitabhängiger Attribute in *drei* Dimensionen entsteht. In diesem Fall sind die zeitabhängigen Attribute zur Darstellung aller möglichen Kombinationen nicht nur einmal, sondern mehrfach in Aggregatstabellen abzubilden. Bei drei zeitabhängigen Attributen in jeder der drei Dimensionen - das Modell enthält dann insgesamt neun zeitabhängige Dimensionsattribute - steigt der Speicherverbrauch gegenüber dem konventionellen Sternschema ohne temporale Aggregatstabellen bereits um mehr als 4000%. Der Systementwickler muss deshalb bereits bei der Identifikation der zeitabhängigen Dimensionsattribute die Folgen des Speicherverbrauchs berücksichtigen und gegebenenfalls auf die temporale Abbildung bestimmter Attribute verzichten.

Die Speichereffizienz ist bei den anderen Modellansätzen jedoch nicht besser, wenn diese ebenfalls Aggregate zur effizienteren Abfrageverarbeitung einsetzen. Bei der SCD-Methode zum Beispiel bildet der erste Datenwürfel die aktuelle Sicht ab. Die zusammengefassten Fakten in diesem Modell entsprechen den konventionellen Aggregaten. Der zweite Datenwürfel bildet die faktbezogene historische Sicht ab. Der Unterschied zwischen den zusammengefassten Werten in diesem und dem aggregierten Faktenmodell besteht in der Freiheit der Aggregatsdefinition. Während die Auswahl der Aggregate im aggregierten Faktenmodell durch die Identifikation der zeitabhängigen Dimensionsattribute vorgegeben ist, können die Aggregate bei der SCD-Methode beliebig definiert werden. Das SCD-Modell ist laufzeitineffizienter und speichereffizienter, wenn es weniger Aggregate enthält. Die Laufzeiteffizienz verbessert sich jedoch nicht, wenn mehr Aggregate als im aggregierten Faktenmodell definiert werden, weil weitere Aggregate keine zeitabhängigen Dimensionsattribute tangieren und deshalb redundant zu den konventionellen Aggregaten sind.

Inkrementelle Aktualisierung

Der durch die Modellerweiterung anfallende Wartungsaufwand kann gering gehalten werden, weil das aggregierte Faktenmodell die temporalen Aggregate immer inkrementell aktualisieren kann.

Im Gegensatz zum Vergleich in Abbildung 2.3.19 bezieht sich die Bewertung der inkrementellen Aktualisierung nicht auf alle, sondern lediglich auf die in der faktbezogenen historischen Sicht abgebildeten Daten, weil die Aktualisierung konventioneller Aggregate bei allen vorgestellten Ansätzen die Neuberechnung der entsprechenden Werte erfordert. Unter diesem Gesichtspunkt erfüllen alle Alternativen dieses Kriterium. Allerdings können die Zeitstempelmethoden und das Zeitschalterverfahren, so wie sie in dieser Arbeit beschrie-

ben wurden, keine Aggregate zur Abbildung der faktbezogenen historischen Sicht verwalten.

Komplexität der Aktualisierungsprozesse

Das aggregierte Faktenmodell verwendet Fortschreibungsmethoden, die von den meisten OLAP-Systemen unterstützt werden und vergleichsweise einfach abzubilden sind. Der Aktualisierungsprozess:

- überschreibt nicht mehr gültige Dimensionsdaten und ergänzt sie mit neuen Werten,
- fügt neue Faktwerte und Aggregate hinzu,
- und berechnet bei Dimensionsdatenänderungen die konventionellen Aggregate neu.

Das OLAP-System wartet weder Zeitstempelattribute (vgl. Zeitstempelmethode) noch verändert es bestehende Faktwerte (vgl. Methode mit alternativen Fremdschlüsseln). Das aggregierte Faktenmodell nutzt damit, abgesehen von der Behandlung der Dimensionsdaten, dieselben Fortschreibungsmethoden wie die SCD-Methode und wird dementsprechend gleich bewertet.

3.3.2 Weiterführende Arbeiten

Die Ausführungen zur Spezifikation und Implementation sowie die anschliessende Bewertung haben gezeigt, dass das aggregierte Faktenmodell in der hier präsentierten Form einerseits noch konzeptionelle Probleme aufweist, andererseits auch Raum für Anpassungen und Verbesserungen offen lässt.

Freie Dimensionswahl

Das aggregierte Faktenmodell erlaubt es nicht, mehrere zeitabhängige Attribute aus derselben Dimension in einer Abfrage zu verwenden. Diese Modelleigenschaft kann die Anwender bereits bei der Erstellung einfacher Analysen behindern. Der erwähnte Lösungsvorschlag, alternative Hierarchiepfade in eigenen Dimensionen zu modellieren und ausschliesslich eindeutige Dimensionswerte zu generieren, ist zwar möglich, benötigt jedoch zusätzliche Ressourcen und erschwert das Modellverständnis. Erst wenn die Aggregatstabellen miteinander adäquat verknüpfbar sind, können die zeitabhängigen Dimensionsattribute ohne Umgehungsstrategie und Einschränkungen flexibel in allen Abfragen genutzt werden.

Temporale Analyse der Dimensionsdaten

Im Gegensatz zu den Fakten sind Dimensionsdaten nicht ausreichend zeitbezogen analysierbar. Dieses Problem könnte ebenfalls durch die Verknüpfung der temporalen Aggregatstabellen gelöst werden, weil dann die Abhängigkeit unter den Dimensionsattributen zeitbezogen abgebildet wäre. Für beide Probleme müsste die Struktur der Aggregatstabellen so angepasst werden, dass sich alle zeitabhängigen Attribute aus einer Dimension in Beziehung zueinander setzen lassen.

Temporale Abbildung des Sternschemas

Eine strukturelle Änderung im Sternschema wie etwa das Löschen oder Einfügen eines Dimensionsattributs kann die Neudefinition der temporalen Aggregatstabellen erfordern. Damit den Anwendern die nicht mehr gültigen Datenstrukturen dennoch zur Verfügung stehen, müssen nicht nur die Daten, sondern auch die Datenstrukturen zeitbezogen gespeichert werden. Falls die Quelldaten zeitbezogen abgebildet sind und ihre Strukturen bei einer Anpassung des Zielschemas unverändert bleiben, können die alten Strukturen mit Hilfe zeitbezogener Metadaten rekonstruiert und analysiert werden. Das aggregierte Faktenmodell ist deshalb so zu erweitern, dass es auch zeitbezogene Metadaten verwalten kann. Die physische Speicherung der Objektdaten zu jedem versionierten Schema kann die Speicheranforderungen vervielfachen und ist deshalb nicht zu empfehlen.

Differentielle Aktualisierung

Das OLAP-System kann die temporalen Aggregate nur inkrementell Aktualisieren, falls die Dimensionsdaten in der Quelldatenbank nicht zeitbezogen gespeichert sind. Die inkrementelle Aktualisierung berücksichtigt allerdings nur neue Quelldaten. Ältere Fakten, die im Quellsystem verändert, gelöscht oder neu hinzugefügt wurden, kann das System aufgrund des in der Vergangenheit liegenden Zeitstempels nicht identifizieren. Das aggregierte Faktenmodell muss folglich neben der inkrementellen auch eine differentielle Aktualisierung ermöglichen oder Veränderungen mit einer anderen Methode als dem Zeitstempelvergleich verfolgen.

Verbesserung der Speichereffizienz

Der Speicherverbrauch der Aggregatstabellen kann ein Mehrfaches der ursprünglichen Datenbankgrösse betragen. Eine Möglichkeit zur Verbesserung der Speichereffizienz liegt in der Vermeidung von Redundanzen. Das aggregierte Faktenmodell bildet *alle* zeitbezogenen Dimensionsattribute in Aggregatstabellen ab, ohne dabei zu untersuchen, ob zwischen diesen Attributen funktionelle Abhängigkeiten bestehen. Falls ein zeitabhängiges Attribut ein anderes eindeutig bestimmt und die funktionelle Abhängigkeit im Zeitablauf nicht veränderbar ist, braucht das abhängige Attribut bei der Definition der temporalen Aggregatstabellen nicht berücksichtigt zu werden. Das System muss dazu die Abhängigkeiten bei der Modelldefinition identifizieren und bei der Analyse der temporalen Aggregatstabellen adäquat berücksichtigen.

Optimierung der Laufzeiteffizienz

Die Laufzeiteffizienz ist im aggregierten Faktenmodell gut, weil es Aggregate nicht zur Laufzeit, sondern im Voraus bei der periodischen Datenaktualisierung berechnet. Der Zugriff auf diese vorberechneten Werte steuert das Endbenutzerwerkzeug, indem es mit Hilfe der Metadatenbank die geeignete Aggregatstabelle und die entsprechenden Parameter zur Beantwortung einer Abfrage bestimmt. Das dazu im Prototyp vorgestellte Verfahren bestimmt zunächst alle in Frage kommenden Aggregatstabellen und wählt aus diesen jene Tabelle aus, die nur die in der Abfrage verwendeten, zeitabhängigen Dimensionsattribute enthält. Wie die Ausführungen zu dieser Funktion bei der Beschreibung der Implementation gezeigt haben, bestimmt dieses Vorgehen zwar eine inhaltlich geeignete, jedoch hinsichtlich der Laufzeiteffizienz nicht immer die beste Aggregatstabelle. Die Laufzeiteffizienz kann deshalb durch die Optimierung dieser Funktion noch gesteigert werden.

3.3.3 Ausblick

Das aggregierte Faktenmodell ermöglicht die Darstellung der Abfrageresultate in der aktuellen und der faktbezogenen historischen Sicht. Die Anwender definieren ihre Abfragen wie in einem herkömmlichen OLAP-System, haben aber die Möglichkeit, für jede Dimension die gewünschte temporale Auswertungsform zu wählen.

Zur Abbildung der aktuellen Sicht werden Dimensionen in einem konventionellen Sternschema mit einer überschreibenden Methode aktualisiert, so dass sie ausschliesslich aktuelle Daten enthalten. Die faktbezogene historische Sicht

wird dagegen durch die Verwendung temporaler Aggregatstabellen ermöglicht. Im Gegensatz zur Faktentabelle enthalten diese Tabellen aggregierte Fakten, die abhängig von veränderbaren Dimensionsattributen abgebildet sind. Bei der Aktualisierung der temporalen Aggregate greift das System auf zeitbezogene Dimensionsdaten in der Quelldatenbank zurück oder wendet eine inkrementelle Fortschreibungsmethode an. Diese Vorgehen ermöglichen die Speicherung der aggregierten Werte gemäss der faktbezogenen historischen Sicht.

Die Verknüpfung der Fakten mit Werten aus einer *Dimensionstabelle* definiert zur entsprechenden Dimension die aktuelle Sicht. Verwendet das System in einer Abfrage hingegen Fakten und Dimensionsdaten, die in einer *temporalen Aggregatstabelle* gespeichert sind, dann ist das Ergebnis zu den betroffenen Dimensionen gemäss der faktbezogenen historischen Sicht dargestellt. Die Anwender brauchen dabei keine Kenntnisse der temporalen Aggregatstabellen zu haben, weil das System die geeigneten Tabellen automatisch auswählt, wenn die faktbezogene historische Sicht zu einer Dimension abzubilden ist.

Das aggregierte Faktenmodell hat gegenüber den anderen vorgestellten Methoden Vorteile bei Laufzeiteffizienz und ermöglicht durch die inkrementelle Fortschreibungsmethode eine effiziente Aktualisierung der temporalen Aggregatstabellen. Zu den zentralen Nachteilen gehören die eingeschränkte Verwendung zeitabhängiger Dimensionsattribute bei der Abfragedefinition und die Speicherineffizienz.

Aufgrund der genannten Eigenschaften wird das aggregierte Faktenmodell vorteilhaft zur Abbildung der aktuellen und der faktbezogenen historischen Sicht eingesetzt, wenn nicht alle, sondern nur ausgewählte Dimensionsattribute in beiden temporalen Auswertungsformen abzubilden sind. Im Gegensatz zu den aufgeführten Alternativen differenziert das aggregierte Faktenmodell zwischen zeitabhängigen und zeitunabhängigen Dimensionsattributen. Die explizite Bestimmung der zeitabhängigen Dimensionsattribute erlaubt es, für jedes Datenmodell einen an die vorhandenen Ressourcen angepassten Kompromiss zwischen Speicherverbrauch und Informationsqualität festzulegen.

Die zeitbezogene Speicherung der Dimensionsdaten in der Quelldatenbank ermöglicht die *vollständige* Aktualisierung aller temporalen Aggregate. Eine vollständige Aktualisierung ist zum Beispiel notwendig, wenn das Sternschema verändert wird oder ein Dimensionsattribut im Nachhinein zeitabhängig abzubilden ist. Die Verfügbarkeit zeitbezogener Dimensionsdaten stellt allerdings die Verwendung temporaler Aggregate in Frage, weil ein OLAP-System in diesem Fall bereits mit der Quelldatenbank alle temporalen Abfragen beantworten kann. Dem aggregierten Faktenmodell bleibt der Vorteil der effizienteren Abfrageverarbeitung, weil die Fakten bereits präaggregiert vorliegen und die Di-

mensionsstrukturen nicht erst zur Laufzeit entsprechend der gewünschten Gültigkeitszeit aufzubauen sind.

Falls *alle* Dimensionsdaten in einem Data Warehouse zeitbezogen abgebildet sind, kann das aggregierte Faktenmodell weiter optimiert werden, indem es nicht nur temporale Aggregate, sondern bei Bedarf auch Quelldaten zur Abbildung der faktbezogenen historischen Sicht nutzt. Eine Weiterentwicklung müsste sich dazu insbesondere mit der Anpassung der Analysefunktionen im Endbenutzerwerkzeug befassen. Die Quelldaten würden zur Beantwortung einer Abfrage beigezogen, falls:

- die faktbezogene historische Sicht gewünscht wird, obwohl das entsprechende Dimensionsattribut als zeitunabhängig definiert wurde und deshalb keine temporalen Aggregate zur Beantwortung der Abfrage vorliegen,
- mehrere zeitabhängige Dimensionsattribute derselben Dimension in einer Abfrage vorkommen,
- die Anwender die Abfrage in der ursprünglichen oder dimensionsbezogenen historischen Sicht abbilden möchten,
- die Dimensionsdaten temporal zu analysieren sind.

Eine derartige *Drilling Through*-Funktion, die den Zugriff auf die mit Zeitstempeln versionierten Quelldaten ermöglicht, könnte die Vorteile des aggregierten Faktenmodells und der Zeitstempelmethode verknüpfen. Auf der einen Seite bieten Aggregate eine Möglichkeit zur Verbesserung der Laufzeiteffizienz, und auf der anderen Seite garantieren die mit Zeitstempeln zeitbezogen abgebildeten Quelldaten für die Abbildung aller temporalen Auswertungsformen. Bei knappen Speicherressourcen kann damit die Anzahl der temporalen Aggregate (auf Kosten der Laufzeiteffizienz) reduziert werden, ohne die Informationsqualität zu beeinträchtigen.

Schlussbemerkungen

Obwohl der Zeitbezug der Dimensionsdaten in den meisten Data Warehouses berücksichtigt wird, ist die Unterstützung verschiedener temporaler Auswertungsformen in vielen OLAP-Systemen mangelhaft. Neben bereits bekannten Ansätzen zur Abbildung veränderbarer Dimensionsdaten in OLAP hat diese Arbeit das aggregierte Faktenmodell eingeführt und beurteilt. Die Ausführungen zur Implementation haben gezeigt, wie sich die temporalen Aggregate in ein OLAP-System einbinden lassen. Des Weiteren hat die Entwicklung des Prototyps dazu beigetragen, die Möglichkeiten und Grenzen des entwickelten Modellansatzes besser zu beurteilen.

Der Abschnitt über die weiterführenden Arbeiten hat verdeutlicht, dass bestimmte Eigenschaften des entwickelten Prototyps noch nicht optimiert sind und es einer Weiterentwicklung bedarf, um den vorgestellten Modellansatz nahtlos in eine bestehende OLAP-Umgebung einbinden zu können. Anstelle der *direkten Behebung* der erörterten Probleme, könnte mit der Entwicklung einer adäquaten Drilling Through-Funktion auf die Quelldaten auch eine *Umgehungsstrategie* zur Verbesserung des aggregierten Faktenmodells gewählt werden. Voraussetzung dazu ist allerdings, dass in den Quelldaten nicht nur die Fakten, sondern auch die Dimensionsdaten zeitbezogen abgebildet sind.

Anhang

Glossar

Ad-hoc-Analyse: Analyse, die durch den Endbenutzer zur Laufzeit definiert wird.

Aggregation: Verdichtung hierarchisch untergeordneter Werte zu einem übergeordneten Wert durch die Anwendung einer ▸ Aggregatsfunktion.

Aggregatsfunktion: Funktion, die eine Berechnung über eine Menge von Werten durchführt. Beispiele einer Aggregatsfunktion sind die Summen-, Durchschnitts- oder Maximumsfunktion.

Aktuelle Sicht: ▸ Temporale Auswertungsform, bei der die ▸ Fakten mit den aktuell gültigen Dimensionsdaten verknüpft sind.

Anomalie: Unregelmässigkeit beim Datenentwurf, die das Einfügen, Löschen oder Fortschreiben von Tabelleneinträgen erschwert.

ActiveX: Komponente, die der Common Object Model (COM) Spezifikation genügt. Die physische Verpackungseinheit ist eine Dynamic Link Library (DLL), ein OLE-Control (OCX) oder eine ausführbare Datei.

Balanced Scorecard: Analyseinstrument, das eine umfassende und bereichsübergreifende Kontrolle und Steuerung von Unternehmenszielen ermöglicht.

Chronon: Kleinste in einem bestimmten Zusammenhang relevante Zeiteinheit.

Cube: ▸ Würfel

Data Mart: Lokales Data Warehouse, das auf die Daten eines Funktionsbereichs, einer Abteilung, einer Arbeitsgruppe oder einer einzelnen Person beschränkt bleibt.

Data Mining: Automatische und nichttriviale Suche nach Wissen in Massendaten.

Data Warehouse: Analytische Datenbank, die strategische Entscheidungen unterstützt, indem sie umfangreiche und regelmässige Auszüge aus operativen Datenbanken integriert, zeitbezogen, themenorientiert und dauerhaft

bereit stellt. Im weiteren Sinn werden auch vor- und nachgelagerte wie die das Extrahieren, Transformieren und Laden der Quelldaten oder die Erweiterung in Richtung Analysewerkzeuge zum Begriff *Data Warehouse* gezählt.

Dicing: Wechseln der betrachteten Dimensionsachsen unter Beibehaltung der restlichen Einstellungen.

Dimension: Strukturkomponente eines ▸ Würfels, das die Auswahl, Zusammenfassung und Navigation eines ▸ Fakts erlaubt.

Dimensionsbezogene historische Sicht: ▸ Temporale Auswertungsform, bei der die ▸ Fakten mit der Version der Dimensionsdaten verknüpft sind, die zu einem benutzerdefinierten Zeitpunkt gültig waren.

DOLAP (Desktop OLAP): ▸ MOLAP- oder ▸ ROLAP-System, das Endbenutzern den Zugriff auf mehrdimensionale Daten auf einem serverunabhängigen Client ermöglicht.

Drilling Across: Verzweigen von einem ▸ Würfel in einen logisch verknüpften anderen Würfel.

Drilling Down: Zerlegung eines verdichteten Werts von einer höheren Hierarchiestufe in seine Komponenten niedrigerer Hierarchiestufen innerhalb einer ▸ Dimension.

Drilling Through: Verzweigen von einem ▸ Würfel in eine logisch verknüpfte Tabelle, die detailliertere Daten als der Würfel enthält.

Drilling Up: Umgekehrte Funktion von ▸ Drilling Down.

Entitätsintegrität: Integritätsbedingung, die verlangt, dass jede Tabellenzeile durch einen Primärschlüssel eindeutig bestimmt ist.

Fakt: Aggregierbares, meist numerisches und kontinuierliches Attribut, das die mehrdimensionale Messung eines betrieblichen Erfolgskriteriums erlaubt.

Faktbezogene historische Sicht: ▸ Temporale Auswertungsform, bei der nur solche Dimensionswerte einem ▸ Fakt zugeordnet werden, welche dieselbe Gültigkeitszeit wie der Fakt aufweisen.

Fremdschlüssel: Attribut, das mit einem ▸ Primärschlüssel einer anderen Tabelle verbunden ist.

Granularität: Grad der Detaillierung der Daten. Je geringer die Granularität, desto höher die Verdichtung der Daten und desto kleiner der Speicheraufwand.

HOLAP: Hybrides OLAP. Mischform aus ▸ ROLAP und ▸ MOLAP.

Horizontale temporale Anomalie: Redundanzbildung beim Einfügen einer neuen Objektversion, die durch ein asynchrones Änderungsverhalten der Attribute in einem Tupel verursacht wird.

Indikator: ▸ Fakt

Java-Applet: In Java geschriebene Programme, die meist in Browsern ausgeführt werden.

Konventionelle Aggregate: Vorberechnete, zusammengefasste ▸ Fakten, die zur Optimierung der Laufzeiteffizienz verwaltet werden. Die darstellbare ▸ temporale Auswertungsform der konventionellen Aggregate und der detaillierten Fakten ist identisch.

Materialisierte Sicht: Abfrageresultat, das physisch in der Datenbank gespeichert wird.

Metadaten: Daten, welche die Semantik von Objektdaten und Prozessen beschreiben.

MOLAP: Multidimensionales ▸ OLAP. Speichert die Daten in einem proprietären, multidimensionalen Datenbanksystem.

Multicube: Abbildungskonzept, in dem die Daten in mehreren physischen ▸ Würfeln gespeichert werden und Verknüpfungen zwischen den Würfel bestehen.

OLAP: Online Analytical Processing. Technologie, die es ermöglicht, Daten multidimensional, schnell und interaktiv zu analysieren.

Primärschlüssel: Attribut, das einen Datensatz eindeutig identifiziert.

QBE: Query By Example. Benutzerfreundliche deklarative Abfragesprache für relationale Datenbanken.

Referentielle Integrität: Integritätsbedingung, die verlangt, dass zu jedem Fremdschlüsselwert der einen Tabelle ein passender Primärschlüsselwert der anderen existiert.

ROLAP: Relationales ▸ OLAP. Speichert die Daten in einem relationalen Datenbanksystem.

Schneeflockenschema: Logisches Datenmodell, in dem als Abwandlung zum ▸ Sternschema normalisierte Dimensionstabellen genutzt werden.

Skalierbarkeit: Fähigkeit eines Systems, die Datenkapazität ohne überdurchschnittlichen Aufwand zu vergrössern.

Slicing: Filtern des multidimensionalen Datenbestandes nach einzelnen Dimensionselementen.

SQL: Structured Query Language. Deklarative, standardisierte Abfragesprache in relationalen Datenbanken.

Sternschema: Logisches Datenmodell, in dem die Kennzahlen in einer Faktentabelle und deren Ausprägungen in Dimensionstabellen abgebildet sind.

Temporale Aggregate: Vorberechnete, zusammengefasste ▸ Fakten, die zur Abbildung einer alternativen ▸ temporalen Auswertungsform verwaltet werden. Die darstellbare temporale Auswertungsform der temporalen Aggregate und der detaillierten Fakten können voneinander abweichen.

Temporale Auswertungsform: Darstellung eines multidimensionalen Abfrageresultats, bei welchem die Dimensionsdaten abhängig von einem bestimmten temporalen Kontext ausgewählt werden. Möglich sind die ▸ aktuelle Sicht, die ▸ dimensionsbezogene historische Sicht, die ▸ faktbezogene Sicht und die ▸ ursprüngliche Sicht.

Trigger: Datenbankprozedur, die automatisch vor oder nach einer insert-, update- oder delete-Anweisung auf einer Basistabelle startet.

Tupel: Datensatz im relationalen Modell.

Ursprüngliche Sicht: ▸ Temporale Auswertungsform, bei der die erste Gültigkeitsversion der Dimensionsdaten den ▸ Fakten zugeordnet wird.

Vertikale temporale Anomalie: Aufteilung der Geschichte eines Objekts auf mehrere Tupel.

Virtuelle Sicht: Abfrageresultat, das nicht physisch in der Datenbank gespeichert wird.

Wiederholungsgruppen: Gruppe von Werten des gleichen Attributs im gleichen Datensatz.

Würfel: Mehrdimensionale Datenstruktur, welche die Analyse eines ▸ Fakts nach mehreren ▸ Dimensionen erlaubt.

Literatur

Agarwal et al. 1999
Agarwal, S., Agarwal, R., Deshpande, P., Gupta, A., Naughton, J., Ramakirshnan, R., Sarawagi, S., *On the Computation of Multidimensional Aggregates*. In: Materialized Views, S. 361-380, MIT Press, Cambridge (Massachusetts).

Albrecht 2001
Albrecht, J., *Anfrageoptimierung in Data-Warehouse-Systemen auf Grundlage des multidimensionalen Datenmodells*. Dissertation der Friedrich-Alexander-Universität Erlangen-Nürnberg.

Allen 1983
Allen, J., *Maintaining Knowledge about Temporal Intervals*. In: Communications of the ACM 26, S. 832-843.

Bange, Schnizer 2000
Bange, C., Schnizer, H., *ETL-Werkzeuge für das Data Warehouse: Aufbauhilfe und Prozesssteuerung*. In: it-Fokus 7/2000, S. 10-16, it-Verlag, Höhenkirchen.

Behme, Mucksch 1998
Behme, W., Mucksch, H., *Informationsversorgung als Wettbewerbsfaktor*. In: Das Data Warehouse-Konzept, S. 4-34, Gabler, Wiesbaden.

Berson, Smith 1997
Berson, A., Smith, S., *Data Warehouse, Data Mining & OLAP*. McGraw-Hill, New York.

Bettini et al. 2000
Bettini, C., Jajodia, S., Wang, S., *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer, Berlin.

Bokun, Taglienti 1998
Bokun, M., Taglienti, C., *Incremental Data Warehouse Updates*. In: DM Review 5/98,
http://www.dmreview.com/master.cfm?NavID=55&EdID=609.

Chamoni, Stock 1998
Chamoni, P., Stock, S., *Temporale Daten in Management Support Systemen*. In: Wirtschaftsinformatik 6/98, S. 513-519, Vieweg.

Corey et al. 1998
Corey, M., Abbey, M., Abramson, I., Taub, B., *Oracle 8 Data Warehousing*. McGraw-Hill, Berkeley.

Corr 2001
Corr, L., *Aggregate Improvments*. In: Intelligent Enterprise, Vol. 4, Nr. 15, http://www.intelligententerprise.com/011004/415warehouse1_2.shtml.

Debevoise 1999
Debevoise, N., *The Data Warehouse Method*. Prentice Hall, Upper Saddle River.

Devlin 1997a
Devlin, B., *Data Warehouse: From Architecture to Implementation*. Addison-Wesley, Reading (Massachusetts).

Devlin 1997b
Devlin, B., *Managing Time in the Data Warehouse*. In: InfoDB, Vol. 11, Nr. 1, http://www.dbaint.com/pdf/v11n12.pdf.

Eder, Koncilia 2001
Eder, J., Koncilia, C., *Changes of Dimension Data in Temporal Data Warehouses*, In: Data Warehousing and Knowledge Discovery, S. 284-293, Springer, Berlin.

Engels 1996
Engels, E., *OLAP jenseits der Schlagworte (1): Grundlagen und Datenmodellierung*. In: it-Fokus 7/96, S. 14-24, it-Verlag, Höhenkirchen.

English 1999
English, L., *Improving Data Warehouse and Business Information Quality*. John Wiley, New York.

Firestone 1998
Firestone, J., *Dimensional Modeling and E-R Modeling in the Data Warehouse*. http://www.dkms.com/papers/dmerdw.pdf.

Gluchowski 1997
Gluchowski, P., *Data Warehouse-Datenmodellierung: Weg von der starren Normalform*. In: it-Fokus, 11/97, S. 62-66, it-Verlag, Höhenkirchen.

Giovinazzo 2000
Giovinazzo, W., *Object-Oriented Data Warehouse Design: Building a Star Schema*. Prentice Hall, Upper Saddle River.

Golfarelli et al. 1998
Golfarelli, M., Maio, D., Rizzi, S., *Conceptual Design of Data Warehouses from E/R Schemes*. In: Proceedings of the Hawaii International Conference On System Sciences.

Gupta, Mumick 1999a
Gupta, A., Mumick, I., *Introduction to Views. In: Materialized Views*. In: Materialized Views, S. 3-8, MIT Press, Cambridge (Massachusetts).

Gupta, Mumick 1999b
Gupta, A., Mumick, I., *Challenges in Supporting Materialized Views*. In: Materialized Views, S. 39-48, MIT Press, Cambridge (Massachusetts).

Hahn 2000
Hahn, A., *OLAP hoch im Kurs bei Endanwendern*. In: it-Fokus 8/2000, S. 59-60, it-Verlag, Höhenkirchen.

Hammergren 1998
Hammergren, T., *Official Sybase Data Warehousing on the Internet*. ITP, London.

Herden 1999
Herden, O., *Temporale Daten im Data Warehouse und Temporales OLAP*. http://www.ie.iwi.unibe.ch/zeit/zobis/workshop5/herden/herden.html.

Höhn 2000
Höhn, R., *Der Data Warehouse Spezialist*. Addison-Wesley, München.

Holthuis 1999
Holthuis, J., *Der Aufbau von Data Warehouse-Systemen*. DUV.

Hurtado et al. 1999
Hurtado, C., Mendelzon, A., Vaisman, A., *Updating OLAP Dimensions*. In: Proceedings of the ACM second international workshop on Data Warehousing and OLAP, S. 60-66.

Imhoff et al. 2001
Imhoff, C., Loftis, L., Geiger, J., *Building the Customer-Centric Enterprise*. John Wiley, New York.

Inmon 1992
Inmon, W., *Building the Data Warehouse*. QED Technical Publishing Group.

Inmon et al. 1996
Inmon, W., Welch, J., Glassey, K., *Managing the Data Warehouse*. John Wiley, New York.

Inmon et al. 1999
Inmon, W., Rudin, K., Buss, C., Sousa, R., *Data Warehouse Performance*. John Wiley, New York.

Jarke et al. 2000
Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis P., *Fundamentals of Data Warehouses*. Springer, Berlin.

Jensen et al. 1998
Jensen, C., Dyreson, C., Böhlen, M., Clifford, J., Elmasri, R., Gadia, S., Grandi, F., Hayes, P., Jajodia, S., Käfer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J., Sarda, N., Scalas, M., Tiberio, P., Wiederhold, G., *The Consensus Glossary of Temporal Database Concepts - February 1998 Version*. In: Temporal Databases: Research and Practice, Springer, Berlin.

Jones et al. 1979
Jones, S., Mason, P., Stamper, R., *Legol 2.0: A Relational Specification Language for Complex Rules*. In: Information Systems, Vol. 4, Nr. 4, S. 293-305.

Jung, Winter 2000
Jung, R., Winter, R. (Hrsg.), Data Warehousing 2000, Physica, Heidelberg.

Kaiser 2000
Kaiser, A., *Die Behandlung zeitbezogener Daten in der Wirtschaftsinformatik*. In: Information als Erfolgsfaktor, S. 103-112, Teubner, Stuttgart.

Kaiser, Wurglitsch 2000
Kaiser, A., Wurglitsch, R., *Die Umsetzung zeitbezogener Daten in betrieblichen Informationssystemen mit einer Oracle-Umgebung unter Berücksichtigung des Modells RETTE*. In: Modellierung betrieblicher Informationssysteme, Proceedings der MobIS-Fachtagung, S. 265-272.

Kambayashi et al. 2000
Kambayashi, Y., Mohania, M., Toja, A. (Hrsg.), *Data Warehousing and Knowledge Discovery*. Springer, Berlin.

Kambayashi, et al. 2001
Kambayashi, Y., Winiwarter, W., Arikawa, M. (Hrsg.), *Data Warehousing and Knowledge Discovery*. Springer, Berlin.

Kimball 1996a
Kimball, R., *The Data Warehouse Toolkit*. John Wiley, New York.

Kimball 1996b
Kimball, R., *Aggregate Navigation with (almost) no Metadata*. In: DBMS 8/96, http://www.dbmsmag.com/9608d54.html.

Kimball 1998
Kimball, R., *The Data Warehouse Lifecycle Toolkit*. John Wiley, New York.

Kimball 1999a
Kimball, R., *When a Slowly Changing Dimension Speeds up*. In: Intelligent Enterprise, Vol. 2, Nr. 11, http://www.intelligententerprise.com/db_area/archives/1999/990308/warehouse.shtml.

Kimball 2000a
Kimball, R., *Many Alternate Realities*. In: Intelligent Enterprise, Vol. 3, Nr. 3, http://www.intelligententerprise.com/000209/webhouse.shtml.

Kimball 2000b
Kimball, R., *There are no Guarantees*. In: Intelligent Enterprise, Vol. 3, Nr. 12, http://www.intelligententerprise.com/000801/webhouse.shtml.

Kimball 2000c
Kimball, R., *Backward in Time*. In: Intelligent Enterprise, Vol. 3, Nr. 15, http://www.intelligententerprise.com/000929/webhouse.shtml.

Kimball 2002
Kimball, R., *Tricky Time Spans*. In: Intelligent Enterprise, Vol. 5, Nr. 10, http://www.intelligententerprise.com/020613/510warehouse1_1.shtml.

Kimball, Merz 2000
Kimball, R., Merz, R., *The Data Webhouse*. John Wiley, New York.

Knolmayer, Myrach 1996
Knolmayer, G., Myrach, T., *Zur Abbildung zeitbezogener Daten in betrieblichen Informationssystemen*. In: Wirtschaftsinformatik 1/96, S. 63-74, Vieweg, Wiesbaden.

Lefébure, Venturi 1998
Lefébure, R., Venturi, G., *Le Data Mining*. Eyrolles, Paris.

Luedtke 2000
Luedtke, J., *Implementing Slowly Changing Dimensions*. In: SQL Server Magazine, 2/2000,
http://www.sqlmag.com/Articles/Index.cfm?ArticleID=7835.

Lusti 1996
Lusti, M., *Dateien und Datenbanken*. 3. Auflage, Springer, Berlin.

Lusti 2001
Lusti, M., *Data Warehousing und Data Mining: Eine Einführung in entscheidungsunterstützende Systeme*. 2. Auflage, Springer, Berlin.

Marco 2000
Marco, D., *Building and Managing the Meta Data Repository*. John Wiley, New York.

Martin 1998a
Martin, W., *Data Warehousing, Data Mining*, OLAP. ITP, Bonn.

Martin 1998b
Martin, W., *Data Warehousing und Data Mining: Marktübersicht und Trends*. In: Das Data Warehouse-Konzept, S. 125-139, Gabler, Wiesbaden.

Marx 1999
Marx, V., *OLAP-Varianten: Entscheidungen leicht gemacht*. In: it-Fokus 8/99, S. 71-74, it-Verlag, Höhenkirchen.

Mertens 1998
Mertens, P., *Konzeptionen für ein Data Warehouse: Unternehmensdatenmodell als Basis*. In: it-Fokus 1/98, S. 54-63, it-Verlag, Höhenkirchen.

Mohania, Toja 1999
Mohania, M., Toja, A. (Hrsg.), *Data Warehousing an Knowledge Discovery*. Springer, Berlin.

Mucksch, Behme 1998
Mucksch, H., Behme, W., *Das Data Warehouse-Konzept als Basis einer unternehmensweiten Informationslogistik*. In: Das Data Warehouse-Konzept, S. 33-100, Gabler, Wiesbaden.

Mumick et al. 1999
Mumick, I., Quass, D., Mumick, B., *Maintenance of Data Cubes and Summary Tables in a Warehouse.* In: Materialized Views, S. 387-407, MIT Press, Cambridge (Massachusetts).

Navathe, Ahmed 1993
Navathe, S., Ahmed, R., *Temporal Extensions to the Relational Model and SQL*. In: Temporal Databases, S. 92-109, Redwood City.

Oehler 2000
Oehler, K., OLAP: *Grundlagen, Modellierung und betriebswirtschaftliche Lösungen*. Hanser, München.

Pendse 2001
Pendse, N., *Multidimensional Data Structures*. http://www.olapreport.com/MDStructures.htm.

Poe 1996
Poe, V., *Building a Data Warehouse*. Prentice Hall, Upper Saddle River.

Schnizer et al. 1999
Schnizer, H., Bange, C., Mertens, H., *Data Warehouse und Data Mining: Marktführende Produkte im Vergleich*. Vahlen, München.

Schraml 2000
Schraml, T., *Retyping Slowly Changing Dimensions*. In: DM Direct 8/2000, http://www.dmreview.com/master.cfm?NavID=198&EdID=2263.

Shin, Baik 1998
Shin, Y., Baik, D., *Integratine Temporal Data in a Data Warehouse*. In: Applied Informatics, ACTA Press.

Silverston et al. 1997
Silverston, L., Inmon, W., Graziano, K., *The Data Model Resource Book.* John Wiley, New York.

Sperley 1999
Sperley, E., *The Enterprise Data Warehouse*. Prentice Hall, Upper Saddle River.

Staud et al. 1999
Staud, M., Vaduva, A., Vetterli, T., *The Role of Metadata for Data Warehousing*. Technical Report 6/99, Department of Information Directory, University of Zürich.

Steiner 1997
Steiner, A., *A Generalisation Approach to Temporal Data Models and their Implementations*. Dissertation th12434, ETH Zürich.

Stock 2001
Stock, S., *Modellierung zeitbezogener Daten im Data Warehouse*. Gabler, Wiesbaden.

Tannler 1997
Tannler, R., *The Intranet Data Warehouse*. John Wiley, New York.

Thomsen 1997
Thomsen, E., *OLAP Solutions*. John Wiley, New York.

Thomsen et al. 1999
Thomsen, E., Spofford, G., Chase, D., *Microsoft OLAP Solutions*. John Wiley, New York.

Thurnheer 1999
Thurnheer, A., *Modellierung analytischer Datenbanken anhand eines Controlling-Beispiels*. In: Wirtschaftsinformatik als Mittler zwischen Technik, Ökonomie und Gesellschaft, S. 83-94,Teubner, Stuttgart.

Thurnheer 2000
Thurnheer, A., *Berücksichtigung von Änderungen in analytischen Datenbanken.* In: Information als Erfolgsfaktor, S. 123-132, Teubner, Stuttgart.

Wedekind et al. 1997
Wedekind, H., Lehner, W., Teschke, M., Albrecht, J., *Preaggregation In Multidimensional Data Warehouse Environments*. In: Proceedings of the 4th Conference of the International Society for Decision Support Systems, S. 581-590.

Wieken 2000
Wieken, J., *Der Weg zum Data Warehouse: Wettbewerbsvorteile durch strukturierte Unternehmensinformationen*. Addison-Wesley, München.

Winter 1999
Winter, R., *Be Aggregate Aware*. In: Intelligent Enterprise, Vol. 2, Nr. 13, http://www.intelligententerprise.com/db_area/archives/1999/991409/scalable.shtml.

Yang, Widom 2001
Yang, J., Widom, J., *Incremental Computation And Maintenance Of Temporal Aggregates*. In: Proceedings of the 17th International Conference on Data Engineering, S. 51-60.

Index

A

B

C

D

E

F

G

H

I

K

L

M

O

R

S

T

U

V

W

Z

Lebenslauf

Andreas Thurnheer von Berneck SG wurde am 14. Januar 1971 in Liestal BL als Sohn des Hansueli Thurnheer und der Irène Thurnheer-Steiner geboren. Nach der Primarschule und dem Progymnasium besuchte er das Gymnasium in Liestal. Im Anschluss an die Matura Typus C begann er im Herbst 1992 das Studium der Wirtschaftswissenschaften mit den Schwerpunkten Internationale Wirtschaftsbeziehungen, Bankmanagement und Wirtschaftsinformatik an der Universität Basel. Das Studium schloss er 1998 mit Lizentiat ab. In den Jahren 1998 bis 2002 arbeitete er als Lehr- und Forschungsassistent von Prof. Dr. Markus Lusti in der Abteilung Wirtschaftsinformatik am Wirtschaftswissenschaftlichen Zentrum der Universität Basel. Zum Zeitpunkt des Druckes der vorliegenden Dissertation ist er bei der PostFinance in Bern als Systemspezialist analytischer Datenbanken in der Distribution tätig.