Logische Modellierung multidimensionaler Datenstrukturen

Matthias Ansorg

5. Juni 2004

Inhaltsverzeichnis

1 Einführung **5**

1.1 Zusammenfassung 5

1.2 Multidimensionale Datenstrukturen als semantisches Konzept 5

1.3 Multidimensionale Datenstrukturen als logisches Konzept 9

1.4 Die Rolle von Modellen im Data Warehousing 10

2 Vergleichende Darstellung der logischen Modelle **13**

2.1 Kriterien zur Bewertung logischer Modelle 13

2.2 Materialisierte Sichten 16

2.3 Flat Schema 17

2.4 Terraced Schema 17

2.5 Star Schema 18

2.6 Developed Star Schema 22

2.7 Snowflake Schema 23

2.8 Constellation Schema und Galaxy Schema 24

2.9 Zusammenfassende Bewertung 25

3 Kritik am Einsatz der logischen Modelle **29**

A Glossar **33**

B Weiterführende Literatur **37**

Kapitel 1

Einführung

1.1 Zusammenfassung

Ausarbeitung zum Thema »Logische Modellierung multidimensionaler Datenstrukturen«. Sie entstand im Informatik-Studium an der FH Gießen-Friedberg im Wirtschaftsinformatik-Seminar bei Prof. Dr. Harald Ritz und Prof. Dr. Manfred Scheer (Sommersemester 2004). Dabei wurde das Thema »Multidimensionale Datenstrukturen - semantische und logische Modellierung« durch diesen und einen weiteren Teilvortrag bearbeitet.

 Verweise auf dieses Dokument können sich beziehen auf: `http://matthias.ansorgs.de/InformatikAusbldg/Modul.AllgSemWI.Ritz_Scheer/MdDst.Ausarbeitung/MdDst.pdf`.

Dieses Dokument setzt grundlegendes Wissen zum Thema Data Warehouse voraus. Wo es noch fehlt, wird oft das Glossar mit seinen kurzen Erklärungen zu Fachbegriffen weiterhelfen können (siehe Anhang A). Dieses Dokument nutzt eine besondere typografische Konvention für Literaturangaben aus PDF-Dokumenten, bei denen logische Seitenzahlen nicht Bestandteil des Dateiformats sind[1]. Die Literaturangabe nennt dann zuerst die logische Seitenzahl und anschließend ein Offset, etwa »(+16)«, mit dem die physikalische Seitenzahl zur automatischen Navigation im PDF-Dokument ermittelt werden kann. Das erleichtert dem Leser den Umgang mit solchen Dokumenten.

1.2 Multidimensionale Datenstrukturen als semantisches Konzept

Wir trennen das Thema »multidimensionale Datenstrukturen« strikt in einen semantischen und einen logischen Bereich. Das entspricht nicht der Praxis: beide Bereiche entstanden gemeinsam und ver-

[1] Logische Seitenzahlen sind solche, die auch auf den Seiten selbst abgedruckt sind; dabei können in einem Dokument unterschiedliche Schemata verwendet werden, meist klein römisch für das Inhaltsverzeichnis und arabisch für den Rest des Dokuments. Physikalische Seitenzahlen sind eine einfache Zählung aller Vorder- und Rückseiten des ausgedruckten Dokuments. Eine automatische Navigation mit logischen Seitenzahlen ist in PDF-Dokumenten nur möglich, wenn dies bei der Erstellung vorgesehen wurde.

schränkt. So werden jedoch die unterschiedlichen Anforderungen getrennt: Anforderungen der Benutzer im semantischen Bereich, technische Anforderungen im logischen und physikalischen Bereich. Außerdem simulieren wir so idealtypische Softwareentwicklung: getrieben von den Wünschen der Anwender und umgesetzt mit den technischen Möglichkeiten, nicht andersherum.

Multidimensionale Datenstrukturen werden üblicherweise im Data Warehousing eingesetzt: »Es wird sich dabei zeigen, dass allen betrachteten Datenmodellen [für Data Warehouses] eine multidimensionale Sichtweise zu Grunde liegt und sie sich in erster Linie an Analysen aus dem Bereich OLAP orientieren.« [Schw03, S. 35 (+16)]. Daher versuchen wir, über die Anforderungen an Data Warehouses zu begründen, warum dort bisher gewöhnlich multidimensionale Datenstrukturen verwendet werden. Schließlich ist ein Data Warehouse selbst nicht über multidimensionale Datenstrukturen definiert (vgl. Kap. A (Glossar)).

Was ist eine multidimensionale Datenstruktur als semantisches Konzept? Das Entity-Relationship-Modell stellt für die semantische Modellierung keine Möglichkeit bereit, den Daten immanente Ordnungen explizit darzustellen; multidimensionale Datenstrukturen als semantisches Konzept bieten gerade das [Ma00, S. 36]. Multidimensional bedeutet gerade: Daten können gleichzeitig nach verschiedenen Anordnungen (»Dimensionen«) geordnet sein.

Semantische Konzepte sind in der Hauptsache Denkmuster für Menschen, Modelle um die Problemwelt zu strukturieren. Deshalb ist es wertvoll, eine anschauliche Vorstellung dieser Konzepte zu haben. Bei multidimensionalen Datenstrukturen erweist sich das als etwas schwierig; wir stellen einige Denkhilfen vor, die einander ergänzen:

Datenwürfel Er leistet bei bis zu drei Dimensionen gute Dienste und ist die gebräuchlichste Veranschaulichung. Ein Beispiel zeigt Abbildung 1.1. Dabei wird jede Dimension durch eine Kante des Würfels dargestellt: Region, Artikel und Zeit. Jede Dimension hat eine eigene Skala entsprechend den Dimensionspositionen. Am Schnittpunkt jedes Tripels der Dimensionspositionen befindet sich eine Zelle in dieser dreidimensionalen Matrix, die eine oder mehrere Kennzahlen enthält. Im Beispiel ist das die Absatzmenge.

Analogie zum Relationenmodell Um Anordnungen zu realisieren, unterscheidet man bei multidimensionalen Datenstrukturen gewöhnlich zwischen quantitativen und qualitativen Daten: »Quantitative Größen (Maßzahlen, Kennzahlen, measures, facts bzw. measured-facts) werden nach verschiedenen qualitativen Aspekten (Blickwinkeln, Dimensionen) aufgeschlüsselt. [...] Beispielsweise ist eine Auswertung der Kennzahl Anzahl der Studierenden nach den Dimensionen Zeit, Studienabschnitt und Studienausrichtung möglich.« [BoeUl01, S. 2].

Dieser Unterschied ist jedoch nicht zwingend: manche Dimensionen sind gleichzeitig quantitative Größen, im vorigen Zitat etwa »Zeit« und »Studienabschnitt«. Sie könnten auch als Kennzahlen und die Kennzahlen als Dimensionen aufgefasst werden. Eine solche Verallgemeinerung, die Dimensionen und Kennzahlen gleichartig behandelt, wird z.B. in ADAPT[2], einem semantischen Modell für multidimensionale Datenstrukturen, verwendet [HePrNi03, S. 99.101.104].

[2] Application Design for Analytical Processing Technologies

Diese Verallgemeinerung lässt auch im Relationenmodell Multidimensionalität erkennen, jedoch nicht explizit modelliert wie für ein semantisches Konzept gewünscht: Jede Relation definiert einen multidimensionalen Raum, aufgespannt durch die Attribute der Relation. Jedes Tupel repräsentiert einen Punkt in diesem multidimensionalen Raum. Oder allgemeiner formuliert: jede nicht leere Teilmenge der Attribute einer Relation spannt einen multidimensionalen Raum auf, dessen Punkte mit Informationen aus den restlichen Attributen verbunden sind. [Ra02, S. 3 (+20)]

magnetische Messlatten [3] Bei mehr als drei Dimensionen kann man vom Datenwürfel zu »magnetischen Messlatten« als Veranschaulichung übergehen. Man denke sich eine Messlatte für jede Dimension, beschriftet mit den Dimensionspositionen in ihrer dimensionseigenen Ordnung. Die Daten denke man sich als magnetische Partikel mit der Eigenschaft, sich entsprechend ihrer Dimensionsposition an jeder dieser Messlatten selbständig anzuordnen. Nur eine Messlatte wird zur gleichen Zeit angelegt; dieses Konzept von »Dimensionen im Zeitmultiplex« macht auch mehr als drei Dimensionen vorstellbar. Ein Beispiel zeigt Abbildung 1.2. Darin wurden Daten mit ihren Kennzahlen als kleine Kreise mit Nummern dargestellt. Nur der Übersichtlichkeit halber sind diese Nummern eindeutig und fortlaufend, im Unterschied zu den üblichen Eigenschaften von Kennzahlen.

Anforderungen der Benutzer an ein Data Warehouse Betriebliche Fach- und Führungskräfte wünschen sich, die im Unternehmen verfügbaren quantitativen Daten einfach und intuitiv analysieren zu können. Eine operative Datenbank reicht dazu nicht aus, da sie keine Analysen über Zeiträume ermöglicht. Und auch das bei operativen Datenbanken übliche Verfahren, stets einen Bericht von der IT-Abteilung anfordern zu müssen um Informationen zu erhalten, die mit Standardformularen nicht erreichbar sind, steht der intuitiven und einfachen Analyse der Daten im Weg. Ein Data Warehouse sollte Fach- und Führungskräften idealerweise »Selbstbedienung« ermöglichen. Dazu müssen die Daten jedoch entsprechend dem natürlichen Geschäftsverständnis dieser Nutzer aufbereitet werden, ihre Nutzung soll keine besonderen technischen Qualifikationen mehr erfordern. Nach [GaGl98, S. 493] und [Ec99].

Wie multidimensionale Datenstrukturen diese Anforderungen erfüllen Die gängigen Methoden für den Data Warehouse-Entwurf sind Ansätze, bei denen sich die logische Struktur des Data Warehouse direkt aus den Abfrageanforderungen der Benutzer ableitet (»first principles approach«) [MoKo00, S. 1]. Ein solches Data Warehouse entspricht dann recht gut dem »natürlichen Geschäftsverständnis des Managers« [GaGl98, S. 493], der Art wie Manager ihre Daten sehen wollen. Diese derzeit gängigen Entwurfsmethoden verwenden nun gerade multidimensionale Datenstrukturen - Multidimensionalität scheint damit der Denkweise der Anwender recht ähnlich zu sein.

[3] Es sei dem Autor erlaubt, hiermit eine eigene Veranschaulichung multidimensionaler Datenstrukturen vorzustellen.

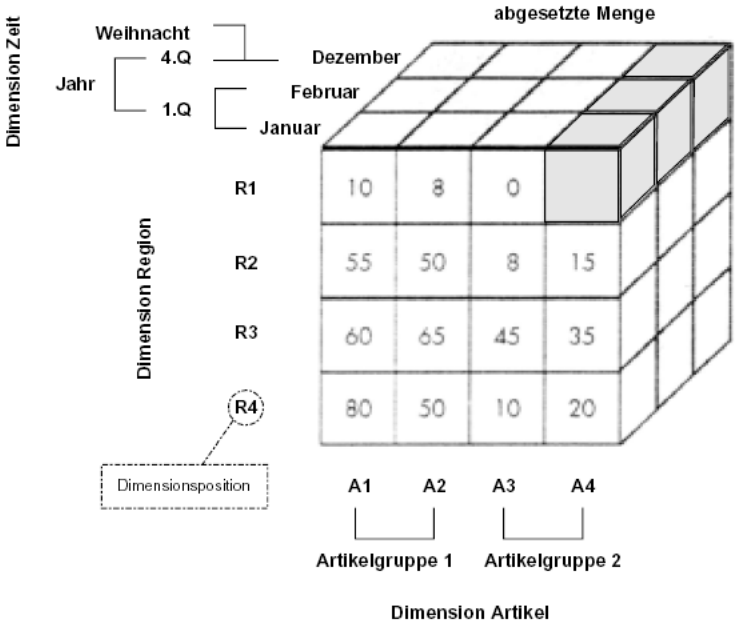

Abbildung 1.1: Der Datenwürfel (nach [Ma00, S. 21] und [ChSt98])

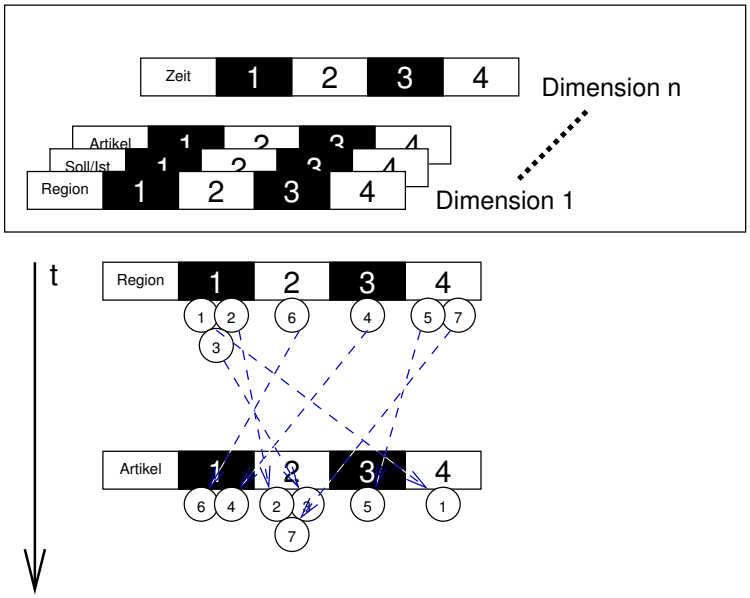

Abbildung 1.2: Dimensional skalierte magnetische Messlatten als Veranschaulichung multidimensionaler Datenstrukturen

1.3 Multidimensionale Datenstrukturen als logisches Konzept

Was ist eine multidimensionale Datenstruktur als logisches Konzept? Beim logischen Entwurf soll das im semantischen Entwurf erstellte Schema derart abgebildet werden, dass sowohl die Bedeutung erhalten bleibt als auch das gegebene Ziel-Datenbanksystem berücksichtigt wird [Schw03, S. 47 (+16)]. Verwendete der semantische Entwurf die multidimensionale Datenstruktur als semantisches Konzept, muss sie sich also auch im logischen Entwurf wiederfinden. Das nennen wir dann »multidimensionale Datenstruktur als logisches Konzept«, völlig unabhängig davon wie die Abbildung von Multidimensionalität auf die logische Ebene konkret aussieht.

Technische Anforderungen an ein Data Warehouse In der bisherigen Praxis übliche technische Anforderungen an ein Data Warehouse:

Korrektheit Das verwendete Datenbanksystem soll eine korrekte Abbildung des semantischen Entwurfs enthalten.

Wartbarkeit Das betrifft insbesondere den Aufwand, das semantische und logische Modell entsprechend geänderter Ausgangsdaten anzupassen. Für typische Anpassungen wie zusätzliche Dimensionen und Änderungen an Dimensionsattributen und -hierarchien soll er gering sein.

Effizienz Typische OLAP-Analysen sollen effizient möglich sein.

Nach [Schw03, S. 47 (+16)]. Darüber hinaus stellen Data Warehouses an das logische Schema eine Anforderung, die sich bei OLTP nicht findet: Verständlichkeit für Endbenutzer. Bei OLTP wird die logische Datenbankstruktur durch eine Softwareschicht verdeckt wird: der Endbenutzer sieht nur Formulare zur Eingabe und muss sich bei spezielleren Anforderungen an die IT-Abteilung wenden. Im Data Warehouse dagegen soll völlige »Selbstbedienung« herrschen; deshalb gewährt man dem Endbenutzer gewöhnlich den direkten Zugang zur logischen Datenbankstruktur. Sie muss deshalb für technisch nicht geschulte Nutzer verständlich sein.

Wie multidimensionale Datenstrukturen diese Anforderungen erfüllen Multidimensionalität kann recht unterschiedlich auf die logische Ebene abgebildet werden, wie wir in Kapitel 2 noch sehen werden. Nicht immer werden die oben genannten technischen Anforderungen an ein Data Warehouse dabei gleich gut erfüllt. Um den gängigen Einsatz multidimensionaler Datenstrukturen auf der logischen Ebene im Data Warehouse nachzuvollziehen, skizzieren wir deshalb nur am Beispiel des »Dimensional Modeling«, wie sie die technischen Anforderungen prinzipiell erfüllen können.

Korrektheit Beim Dimensional Modeling mit dem Flat Schema etwa erreicht man eine korrekte Abbildung der Multidimensionalität, indem man alle Dimensionen inkl. Attributen zusammen mit den Kennzahlen in einer einzigen Tabelle speichert [Schw03, S. 47 (+16)]. Mehr dazu in Kapitel 2.

Wartbarkeit Das Flat Schema z.B. kann um Dimensionen oder Dimensionsattribute erweitert werden, indem man wie eben beschrieben weitere Attribute zur einzigen Tabelle des Schemas hinzufügt.

Effizienz Effizente Analysen zumindest für typische OLAP-Analysen waren ein besonders wichtiges Ziel bei der Entwicklung multidimensionaler Datenstrukturen auf der logischen Ebene [Schw03, S. 47 (+16)]; vgl. auch [Qu03, S. 10-11]. Effizienzsteigernde Maßnahmen sind etwa:

Explizite Dimensionsstruktur Auch die operativen Datenbanken enthalten eine Dimensionsstruktur - sonst könnte sie nicht zur Verwendung im Data Warehouse extrahiert werden. Im Data Warehouse ist sie jedoch erstmals explizit in der Datenstruktur abgelegt, d.h. es werden multidimensionale Datenstrukturen eingesetzt [Qu03, S. 10-11]. Damit steht die Dimensionsstruktur effizient zur Verfügung, ohne stets neu extrahiert werden zu müssen, und ist für Benutzer unmittelbar sichtbar. Die Dimensionsstruktur ist grundlegend für OLAP-Analysen.

Vorverdichtung Mehrere Datensätze werden aggregiert, um eine für die geplanten Analyseanwendungen geeignete feinste Granularität nicht zu unterschreiten. Das nämlich würde den Analyseaufwand durch zusätzliche Aggregationen zur Laufzeit unnötig erhöhen. Vorverdichtung bedeutet andererseits Informationsverlust, was die möglichen Analysen und Anwendungen im Data Warehouse einschränkt.

Dimensionen vernachlässigen sofern sie für die geplanten Analysen uninteressant sind.

Redundanz Sie verbessert die Performanz komplexer Abfragen; konkret in SQL-Termini: sie spart Joins. Data Warehouses werden üblicherweise durch Stapelverarbeitungsprozesse erstellt und danach nicht verändert - Redundanz führt also theoretisch nicht zu Inkonsistenzen wie bei operativen Datenbanken. Siehe [GuMa00, S. 2].

1.4 Die Rolle von Modellen im Data Warehousing

Drei Abstraktionsebenen beim Datenbankentwurf Datenabstraktion ist eine Standardtechnik beim Entwurf von Datenbanksystemen für OLTP. Dabei unterscheidet man in erster Näherung drei Abstraktionsebenen in einem Datenbanksystem (u.a. nach [Zi01, S. 23 (+1)]):

Sichten (view level) Eine Sicht stellt jeweils eine Teilmenge der Daten dar, angepasst auf die Bedürfnisse der jeweiligen Benutzer.

logische / konzeptuelle Ebene (conceptual level) Hier wird durch Schemata definiert, welche Daten in der Datenbank abgespeichert werden. Es wird weiter differenziert:

konzeptuelle Ebene Auf dieser Ebene wird ein semantisches Datenmodell erstellt. »Dieses Modell stellt die Realität aus der Sicht der Anwender dar und benutzt hierzu natürliche, möglichst vielfältige Beschreibungselemente [LoRa90, S. 5]. Das Datenmodell kann daher als Schnittstelle zwischen Anwendern und Entwicklern sowie als Diskussionsgrundlage zwischen diesen Gruppen eingesetzt werden.« [HePrNi03, S. 99]. Es »[. . .] soll vom Typ des zu verwendenden Datenbanksystems und der physischen Struktur, in der die Daten abgelegt werden, unabhängig sein [BaCeNa92]. Im Data-Warehouse-Bereich bedeutet dies

eine Modellierung unabhängig davon, ob die Daten in einem relationalen oder multidimensionalen Datenbanksystem verwaltet werden [Wu97] [HueLeVo00].« [Schw03, S. 36 (+16)]

logische Ebene »Das Ziel des logischen Entwurfs ist es, das im konzeptuellen Entwurf erstellte Schema auf ein Schema abzubilden, das das gegebene Ziel-Datenbanksystem berücksichtigt. Beim logischen Entwurf für ein Data Warehouse muss also berücksichtigt werden, ob für ein relationales oder ein multidimensionales Datenbanksystem modelliert werden soll [GoRi99] [HueLeVo00].« [Schw03, S. 47 (+16)]. »Logische Datenmodelle wie das Netzwerk- oder das Relationenmodell bilden die Nutzersicht bestimmter Datenbankverwaltungssysteme und sind, sofern kein Produktstandard existiert, proprietär.« [vM00, S. 54 (+24)].

physische Ebene (physical level) Hier ist definiert, wie die Daten auf dem Massenspeicher gespeichert werden.

Aus den Literaturhinweisen, die z.T. aus Werken über multidimensionale Datenstrukturen stammen, erkennt man bereits dass die bei OLTP bewährte Datenabstraktion auf den Bereich Data Warehousing übertragen wurde. Was nicht sagt dass sie auch konsequent eingesetzt würde: »Die Modellierung multidimensionaler Datenstrukturen, wie sie in managementunterstützenden Systemen häufig anzutreffen sind, beschränkt sich heute zumeist auf den Aufbau von Star- bzw. Snowflake- Schemata und damit auf die logische Datenmodellierung.« [GaGl98, S. 10]

Drei Entwurfsphasen beim Datenbankentwurf Der konzeptuellen, logischen und physischen Ebene entsprechen die drei Phasen des klassischen Datenbankentwurfs: der konzeptuelle (auch: semantische), logische und physische Entwurf. Sie können als Phasen des Entwurfs von Data Warehouses übernommen werden [Schw03, S. 36 (+16)]. Multidimensionale Datenstrukturen finden sich dann als semantisches Konzept (vgl. Kap. 1.2) im konzeptuellen Entwurf, als logisches Konzept (vgl. Kap. 1.3) im logischen Entwurf.

Im konzeptuellen Entwurf verwendet man konzeptuelle Modelle und erstellt konzeptuelle Schemata. Im logischen Entwurf verwendet man logische Modelle und erstellt logische Schemata[4].

Ist die Modellierung von Data Warehouses etwas Besonderes? *De facto* unterscheidet sich die konzeptionelle Modellierung von Datenbanken für OLTP und Data Warehouses. Diese Unterschiede lassen sich wie folgt verstehen: die beiden Datenmodelle folgen verschiedenen Modellierungs-Intentionen, weil ihnen verschiedene Interessen zugrunde liegen. Während Datenmodellierung für OLTP das Ziel hat, das operative Geschäft eines Unternehmens als Ergebnis einer Organisationsanalyse abzubilden, hat Datenmodellierung für Data Warehouses das Ziel, Analysen im Sinne von OLAP und Data Mining zu ermöglichen. Während deshalb z.B. für Data Warehouses Daten üblicherweise auch zeitlich strukturiert werden, sind alte Zustände für OLTP weitgehend uninteressant.

[4] Modelle, Datenmodelle, Schemata, Schematypen: In der Literatur werden diese Begriffe nicht einheitlich verwendet, vgl. etwa die vorhergehenden Zitate. Für den Rest dieses Dokuments beschränken wir uns deshalb auf die im Glossar (Kap. A) gegebenen Verwendungen.

Die erstellten Schemata unterscheiden sich so in den abgebildeten Daten und Strukturen. Datenmodellierung für Data Warehouses ist nichts Besonderes, sondern etwas Anderes. Siehe [vM00, S. 55 (+24)].

Auch die logische Modellierung für Data Warehouses ist *de facto* anders. Betrachten wir etwa das Dimensional Modeling nach Kimball, eine wesentliche Technik zum logischen Entwurf von Data Warehouses. Sie unterscheidet sich vom logischen Entwurf für OLTP mit der Begründung der unterschiedlichen Anforderungen von OLTP und OLAP [MoKo00, S. 2]. Die Unterschiede im semantischen Entwurf werden also im logischen Entwurf reflektiert. Bei Kimball geschieht das u.a. deshalb, weil die Struktur traditionell modellierter relationaler Datenbanken für Endbenutzer zu komplex ist, sie dagegen multidimensionale Datenstrukturen fast intuitiv verstehen können [MoKo00, S. 2][5]. Die bisher vorgeschlagenen Möglichkeiten, Multidimensionalität auf logischer Ebene in einer relationalen Datenbank abzubilden, werden nun in Kapitel 2 diskutiert.

[5] Es sei schon hier angemerkt, dass der Autor es für wenig sinnvoll hält, die logische und physikalische Struktur des Data Warehouse auf eine möglichst intuitive Bedienung durch Endbenutzer zu optimieren oder sie dem Endbenutzer überhaupt direkt sichtbar zu machen. In Kapitel 3 wird dann eine alternative Architektur vorgestellt, in der das zentrale Data Warehouse in traditioneller Weise logisch modelliert ist.

Kapitel 2

Vergleichende Darstellung der logischen Modelle

Es wird ein Kriterienkatalog erarbeitet, mit dem logische Modelle[1] für multidimensionale Datenstrukturen bewertet werden können. Er wird dann der Reihe nach auf die wesentlichen Modelle angewandt, wobei hier nur Modelle für relationale Datenbanken betrachtet werden. Gleichzeitig werden die Verwandtschaftsbeziehungen der Modelle untersucht - Abbildung 2.1 zeigt die Ergebnisse und ermöglicht die Einordnung der Modelle schon beim ersten Lesen. Die Abbildung zeigt, dass die logischen Modelle hauptsächlich Star- und Snowflake-Schema-Varianten sind. Sie unterscheiden sich voneinander vor allem durch folgende Merkmale (aus [BoeUl00, S. 9]):

- Grad der Normalisierung
- Berücksichtigung von Aggregationen
- Verwendung von künstlichen Primärschlüsseln
- Anzahl der berücksichtigten Datenwürfel

Am Ende des Kapitels wird zusammenfassend und vergleichend anhand des Kriterienkatalogs bewertet, ob und wofür sich die einzelnen logischen Modelle eignen.

2.1 Kriterien zur Bewertung logischer Modelle

Verständlichkeit für Endbenutzer Bei operativen Datenbanken greifen Endbenutzer gewöhnlich auf die Datenbank nur über formularbasierte Anwendungen zu, die Tabellenstruktur der Datenbank bleibt ihnen dadurch verborgen. Das ist ein hauptsächlicher Unterschied zu Data Warehouses, wo auch Endbenutzer ihre Anfragen direkt an die Datenbank richten. [MoKo00, S. 2]. Deshalb ist eine

[1] »Modell« wird hier i.S.d. Dokumententitels verwendet, nicht i.S.v. »Datenmodell«. Das logische Datenmodell aller hier betrachteten »Modelle« ist das Relationenmodell, die »Modelle« sind damit eigentlich Schematypen dieses Datenmodells. Vgl. Kap. A (Glossar).

Anforderung, die *de facto* an logische Schemata gestellt wird: sie sollen für Endbenutzer verständlich sein.

Dabei ist nicht die Verständlichkeit der grafischen Darstellung des logischen Schemas, sondern repräsentationsunabhängig die Verständlichkeit der logischen Datenbankstruktur selbst gemeint - nur diese ist für Endbenutzer ja sichtbar. Repräsentationsunabhängige Verständlichkeit bedeutet hauptsächlich geringe Komplexität: in traditioneller Manier entworfene operative relationale Datenbanken enthalten oft einige hundert Tabellen und ihre Beziehungen; eine solche Struktur erfordert selbst bei recht einfachen Abfragen Joins über mehrere Tabellen, was für Endbenutzer schon wesentlich zu komplex ist. Schon aus diesem Grund fordert man für den Entwurf von Data Warehouses andere Modelle. [MoKo00, S. 3]

Effizienz typischer Abfragen Für interaktive Analyseformen wie OLAP ist eine schnelle Abarbeitung von Anfragen wünschenswert. Es wurden sogar Anstrengungen unternommen, das Benutzerverhalten zu modellieren und vorherzusagen, um damit seine wahrscheinlich nächste Anfrage schon vorauszuberechnen und die Antwortzeit zu minimieren ([Sa99]; so nach [Jue99, S. 10]). Daran erkennt man, wieviel Wert in der Praxis auf geringe Antwortzeiten gelegt wird. Auch in [MoKo99] wird als Ziel des Dimensional Modeling, einer Technik der logischen Modellierung für Data Warehouses, die Effizienzmaximierung der Abfragen angegeben (nach [GuMa00, S. 5]). Vgl. auch [Schw03, S. 47 (+16)]. Ohne besondere Maßnahmen sind die gewünschten Antwortzeiten aber nicht erreichbar; verantwortlich dafür sind die bei OLAP häufigen rechenintensiven Aufgaben wie die Aggregation über sehr viele Tupel [Jue99, S. 9-10].

Abbildung für reichhaltige Semantik Konzeptionelle Modelle sollten im Idealfall völlig unabhängig von den anschließend verwendeten logischen Modellen sein. Dabei ist einzuschränken: »Da eine konzeptionelle Datenmodellierungs-Methode allerdings nicht sinnvoll für ein logisches Datenmodell zu verwenden ist, das weniger Semantik beinhaltet, ist diese Unabhängigkeit nicht wirklich gegeben. In der Praxis werden deshalb auch z. B. für hierarchische Datenmodelle Bachmann-Diagramme, für Relationenmodelle ERM verwendet.« [vM00, S. 54 (+24)]. Um trotzdem dem Ideal unabhängiger konzeptioneller Modellierung möglichst nahe zu kommen, ist für logische Modelle zu fordern, dass sie die in gängigen konzeptionellen Modellen verwendete Semantik verlustfrei abbilden können. Konkret bedeutet das die Abbildung von:

- quantitativen Daten (Kennzahlen) getrennt von qualitativen Daten (Dimensionen)
- einfache Hierarchie als Dimensionsstruktur. Bei einer einfachen Hierarchie besitzen alle Blätter dieselbe Pfadlänge.
- Hierarchie mit unterschiedlichen Pfadlängen als Dimensionsstruktur. Hier haben Blätter (d.h. die atomaren Dimensionspositionen) unterschiedlich lange Wege zur Wurzel.
- parallele Hierarchie als Dimensionsstruktur. Solch eine Struktur liegt vor, wenn innerhalb einer Dimensionshierarchie unterschiedliche, parallele Konsolidierungen vorgenommen werden.

- Heterarchien als Dimensionsstruktur. Heterarchien werden auch »anteilige Verrechnung« genannt. Die Dimensionsstruktur ist dabei kein Baum, sondern eine Dimensionsposition wird anteilig auf zwei Dimensionspositionen der nächsthöheren Ebene verrechnet [Ho99, S. 131].
- Dimensionsattribute als zusätzliche Beschreibungen von Dimensionspositionen und -ebenen
- Formeln, mit denen abgeleitete Kennzahlen aus Basiskennzahlen hergeleitet werden können. [GaGl97]
- Aggregationen überhaupt
- Aggregationsverhalten von Kennzahlen bezüglich einer Dimension

Nach [HePrNi03, S. 98].

Orientierung am Zieldatenbanksystem Logische Modelle sind per definitionem vom Zieldatenbanksystem abhängig; deshalb gibt es unterschiedliche logische Schemata für den Entwurf eines Data Warehouse für ein objektorientiertes, multidimensionales oder relationales Datenbanksystem [Schw03, S. 47 (+16)]. Geeignet ist ein logisches Modell dann, wenn es auf dem Datenmodell des Zieldatenbanksystems (etwa dem Relationenmodell) basiert [Schw03, S. 35 (+16)]. Eine Anforderung an ein logisches Modell ist damit: es darf nur Beschreibungselemente verwenden, die entweder direkt aus dem Datenmodell des Ziel-DBMS stammen oder durch solche darstellbar sind. So ist garantiert dass die erstellten Schemata quasimechanisch in eine Datenbankdefinition für das Zieldatenbanksystem umgesetzt werden können. Semantische Modelle dagegen sollen sich am Problembereich orientieren und natürliche, möglichst vielfältige Beschreibungselemente verwenden [LoRa90, S. 5] (aus [HePrNi03, S. 97]).

Wartbarkeit Auch für das logische Modell gilt: »Typische Änderungen der Ausgangsdaten sollen in den Modellen einfach nachvollzogen werden können. Dies betrifft einerseits das Hinzufügen zusätzlicher Dimensionen und andererseits Änderungen in den Attributen und Hierarchien einer Dimension.« [Schw03, S. 47 (+16)]. Wartbarkeit logischer Modelle bedeutet insbesondere auch: es darf keinen besonderen Aufwand bedeuten, diese Änderungen derart durchzuführen, dass das Data Warehouse sich danach noch in einem konsistenten Zustand befindet. Gerade das erweist sich bei logischen Modellen, die gezielt redundante Daten enthalten, oft als schwierig (vgl. [MoKo00, S. 9]).

Werkzeugunterstützung Sie soll ein effizientes Erstellen und Verwalten der erstellten Schemata ermöglichen [HePrNi03, S. 98] und ist deshalb für alle Modelle gleichermaßen wünschenswert.

Weniger wesentliche Kriterien Anforderungen wie leichte Erlernbarkeit, übersichtliche grafische Darstellung und eine beschränkte Anzahl an Notationselementen sind wichtig für konzeptuelle Modelle [HePrNi03, S. 98]. Weniger aber für die logischen Modelle, weil sie üblicherweise von informationstechnisch versierten Personen erstellt werden.

2.2 Materialisierte Sichten

Dieser Abschnitt nimmt eine Sonderstellung ein - er begründet lediglich warum materialisierte Sichten *nicht* unter die anderen logischen Modelle für multidimensionale Datenstrukturen zu zählen sind, trotz dass sie in der Praxis alternativ zu diesen Modellen eingesetzt werden können.

Was sind materialisierte Sichten? Es sind in vorberechneter Form bereitgehaltene Ansichten der Daten in relationalen Datenbanken. Vorberechnet bedeutet, dass sie Ergebnisse von Join-Operationen zwischen Relationen sind.

Materialisierte Sichten zur Zugriffsbeschleunigung Zum einen sind materialisierte Sichten ein breit eingesetztes Mittel, um den Zugriff auf multidimensionale Datenstrukturen in relationalen Datenbanken zu beschleunigen [GuHaRaUl97] (aus [Jue99, S. 11]). Die Basistabellen eines Data Warehouse können sehr groß werden, weshalb es mit etwas Vorauswissen über die zukünftigen Analysen sinnvoll ist, die aufwändigen Joins dieser Basistabellen vorauszuberechnen und für die erwarteten Analysen vorrätig zu halten. [Jue99, S. 11]

Für materialisierte Sichten zur Zugriffsbeschleunigung ist es irrelevant, mit welchem logischen Modell die Basistabellen erstellt wurden. In dieser Anwendung sind sie also kein logisches Modell und übernehmen auch nicht seine Funktion. Es ist denkbar, auch Zugriffe in operativen Datenbanken so zu beschleunigen. Die Informatik hat sich der materialisierten Sichten als Forschungsgegenstand angenommen, dabei »stehen v.a. die Aspekte Auswahl, Aktualisierung und Verwendung von materialisierten Sichten im Mittelpunkt.« [BoeUl00, S. 32]

Materialisierte Sichten alternativ zu logischen Modellen Die verschiedenen Methoden dimensionaler Modellierung, wie sie in den folgenden Kapiteln behandelt werden, sind deutlich als logische Modelle einzuordnen, weil sie ein konzeptionelles Modell angepasst auf ein Zieldatenbanksystem transformieren können. Neben der dimensionalen Modellierung ist der Entwurf von Data Warehouses mit materialisierten Sichten eine Alternative, die zumindest genannt werden sollte [GuMa00, S. 5].

Dabei werden dann weder konzeptuelle noch logische Modelle eingesetzt. Stattdessen werden materialisierte Sichten direkt aus den operativen Daten per SQL-Abfragen generiert. Welche Sichten generiert werden hängt dann z.B. ab vom Platzbedarf dieser Sichten, ihrer Eignung und dem Aufwand für die Anfragebeantwortung und dem Wartungsaufwand für diese Sichten. Zur Verwaltung der Sichten wird u.a. vorgeschlagen, eine Sammlung von Sichten zusammen mit Anfragen als Zustand und die Änderung von Sichten oder Anfragen als Zustandsübergang zu sehen. Aus [TheSe99] (nach [GuMa00, S. 7]).

Nachteilig daran, das Schema eines Data Warehouse aus materialisierte Sichten aufzubauen, ist: jede Sicht muss mit einem einzigen SQL-Statement aus den operativen Daten erzeugt werden. Das logische Schema des Data Warehouse wird damit abhängig von der Mächtigkeit einer Datenbankabfragesprache, der Struktur der operativen Datenquellen. Der logische Entwurf ist nicht vom semantischen Entwurf allein abhängig, sondern im Gegenteil von den physischen Gegebenheiten. Siehe [GuMa00,

S. 8]. Weil ein logisches Modell im Sinne dieses Dokuments aber die Umsetzung eines konzeptuellen Modells für ein Zieldatenbanksystem sein soll, zählen wir materialisierte Sichten nicht dazu.

2.3 Flat Schema

Aufbau und Darstellung Die logische Grundlage aller Abbildungen multidimensionaler Strukturen auf relationale Datenbanken ist bei materialisierten Sichten und allen Möglichkeiten dimensionaler Modellierung identisch. Sie wurde in Kapitel 1.2 als »Analogie zum Relationenmodell« vorgestellt: multidimensionale Daten werden in einer Faktentabelle $R(a_1, \ldots, a_n, s)$ abgelegt. Dabei stellen die Attribute $a_1, \ldots, a_n$ die n Dimensionen dar, sie spannen einen n-dimensionalen Raum auf. s ist die Kennzahl, die jedem Punkt dieses Raums zugeordnet werden kann. Nach [Jue99, S. 12].

Die Faktentabelle enthält Fakten; darunter versteht man beim Dimensional Modeling die Zusammenfassung von Kennzahlen und ihrem Kontext (Verweise auf die Dimensionspositionen der Dimensionen, unter denen die Kennzahlen betrachtet werden) [GuMa00, S. 5]. Zusätzlich zu den Fakten gibt es weitere Daten wie etwa Dimensionsattribute und die Dimensionspositionen selbst. Ein Flat Schema entsteht nun, indem diese weiteren Daten durch Denormalisierung der Tabellen mit in der Faktentabelle gespeichert werden. Star Schema und Snowflake Schema sind unterschiedlich weit normalisierte Varianten des Flat Schema. Flat Schemata haben außerdem auch nur eine minimale Zahl an Fakttabellen: manche Fakttabellen können zu detaillierteren kombiniert werden, etwa wenn die Dimensionen einer Fakttabelle eine Untermenge der Dimensionen einer anderen bilden. Insgesamt halten die Tabellen im Flat Schema also alle möglichen Joins in vorberechneter Form vorrätig. Nach [AbSaSa01, S. 15].

Probleme beim Flat Schema Bei Flat Schemata wurde auch die Zahl der Fakttabellen minimiert, indem übergeordnete Fakten unter Wiederholung in die Tabellen der untergeordneten Fakten eingeordnet wurden. [MoKo00, S. 7] führt ein Beispiel an, in dem es Verkäufe gibt die jeweils mehrere Verkaufs-Positionen und einen Rabatt als Geldbetrag umfassen können. Im Flat Schema werden alle Informationen der Verkäufe inkl. dem Rabatt in die Tabelle der Verkaufs-Positionen eingefügt. Das kann zu fehlerhaften Aggregationen: der Rabattbetrag wird für einen Verkauf dann mehrfach gezählt, nämlich für jeden Artikel statt nur für den ganzen Verkauf! [MoKo00, S. 7]

Weiterhin führen Flat Schemata oft zu Tabellen mit sehr vielen Attributen. Während durch die minimierte Zahl an Tabellen die Systemkomplexität sinkt, steigt die Elementkomplexität an und der Gewinn an Verständlichkeit für Endbenutzer wird wieder zunichte gemacht [MoKo00, S. 7].

2.4 Terraced Schema

Wie beim Flat Schema enthalten die Fakttabellen zusätzlich alle Informationen über Dimensionen (etwa Dimensionsattribute, Dimensionshierarchien) in denormalisierter Form. Damit werden die sogenannten »Star Joins[2]« vorausberechnet bevorratet. Im Unterschied zum Flat Schema werden Joins

[2]Die Bezeichnung leitet sich vom Star Schema her, wo eine zentrale Fakttabelle von Dimensionstabellen sternartig umgeben ist. Den Join zwischen Fakt- und Dimensionstabellen nennt man daher auch »Star Join«.

mehrerer Fakttabellen zu einer detaillierteren Fakttabelle nicht mehr bevorratet. Nach [AbSaSa01, S. 15]. Vorausberechnung von Joins kostet Speicherplatz durch Redundanz, macht die Datenbankstruktur evtl. besser verständlich, beschleunigt aber v.a. die Anfragen; damit sind die qualitativen Unterschiede zum Flat Schema genannt.

2.5 Star Schema

Die Darstellung in diesem Abschnitt lehnt sich an [BoeUl00, S. 12 ff.] an.

Grundsätzliches Das klassische Star Schema ist ein Schematyp zur logischen Modellierung von Data Warehouses für relationale Datenbanken. Es wird als das beliebteste denormalisierte Schema zum Entwurf von Data Warehouses bezeichnet [Jue99, S. 10]. Anders als beim Flat Schema und Terraced Schema gibt es zwei Arten von Tabellen: zusätzlich zu den Fakttabellen gibt es Dimensionstabellen. Anders formuliert: die Star Joins werden nicht mehr vorausberechnet vorgehalten, sondern werden bei Bearbeitung der Abfrage berechnet. Vorteilig sind Speicherplatzersparnis durch weniger Redundanz und eine bessere Verständlichkeit der Datenbank, weil die Basiskonzepte multidimensionaler Datenstrukturen (Kennzahlen und Dimensionen) nun getrennt und explizit abgebildet werden. Nachteilig ist natürlich der größere Aufwand zur Bearbeitung von Abfragen, weil im Unterschied zum Flat Schema die Star Joins nun notwendig sind.

Aufbau und Darstellung Sowohl Fakt- als auch Dimensionstabellen werden durch Rechtecke symbolisiert. Die Fakttabelle enthält die Kennzahlen und ihre Position im multidimensionalen Raum, d.i. ihren Kontext. Diese Position wird durch Fremdschlüsselbeziehungen zu den Primärschlüsseln aller Dimensionstabellen angegeben. Die Dimensionstabellen nun beinhalten die qualitativen Daten der Dimension, die Beschreibungen der Dimensionspositionen, wie sie zur Visualisierung eingesetzt werden. Die Fremdschlüsselbeziehung zwischen Fakttabelle und Dimensionstabellen wird durch eine Kante mit oder ohne Pfeilspitze dargestellt. Insgesamt ergibt sich ein Schema mit einer zentralen Fakttabelle, sternförmig umgeben von beliebig vielen Dimensionstabellen (daher »Star Schema«). Um die Fremdschlüsselbeziehung zu realisieren werden sowohl künstliche Primärschlüssel verwendet (etwa in [Jue99, S. 11]) als auch Attributkombinationen zugelassen, deren Werte dann aus der Realwelt abgeleitet wurden (etwa in [BoeUl00, S. 13]).

Abbildung 2.2 zeigt den prinzipiellen Aufbau eines Star Schema, die Abbildungen 2.3 und 2.4 zeigen ein Beispiel aus dem Hochschulbereich, modelliert mit einem Star Schema.

Einige wesentliche Nachteile des Star Schema sind nach [Schw03, S. 50 (+16)]:

- Dimensionshierarchien sind unvollständig modelliert
  - Es wird externe Information benötigt, um zu wissen, welches Attribut einer Dimensionstabelle welche Hierarchiestufe repräsentiert. Das Schema informiert damit nicht über die möglichen Aggregationspfade innerhalb einer Dimension.

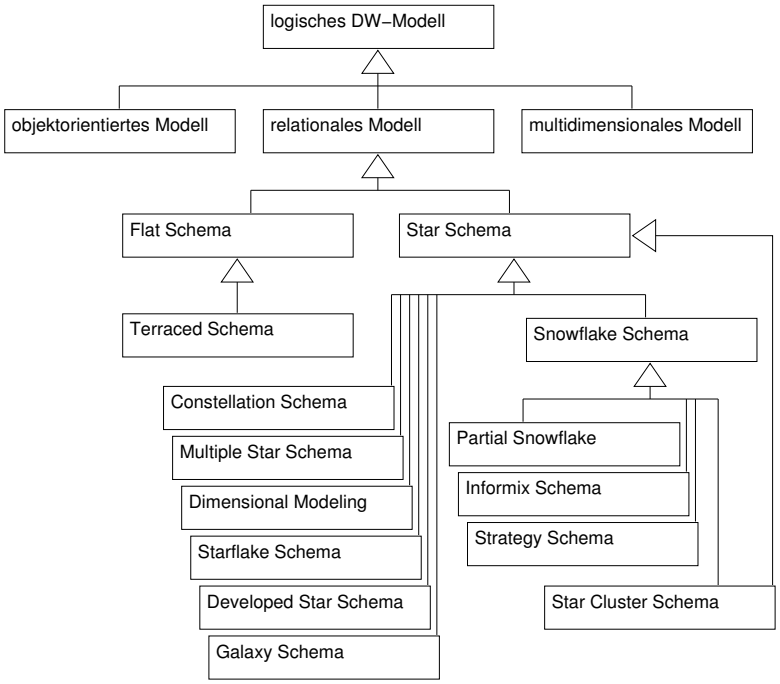

Abbildung 2.1: Beziehungen zwischen logischen Modellen für multidimensionale Datenstrukturen. Nach [AbSaSa01, S. 15], [BoeUl00, S. 9], [Schw03, S. 67-69].

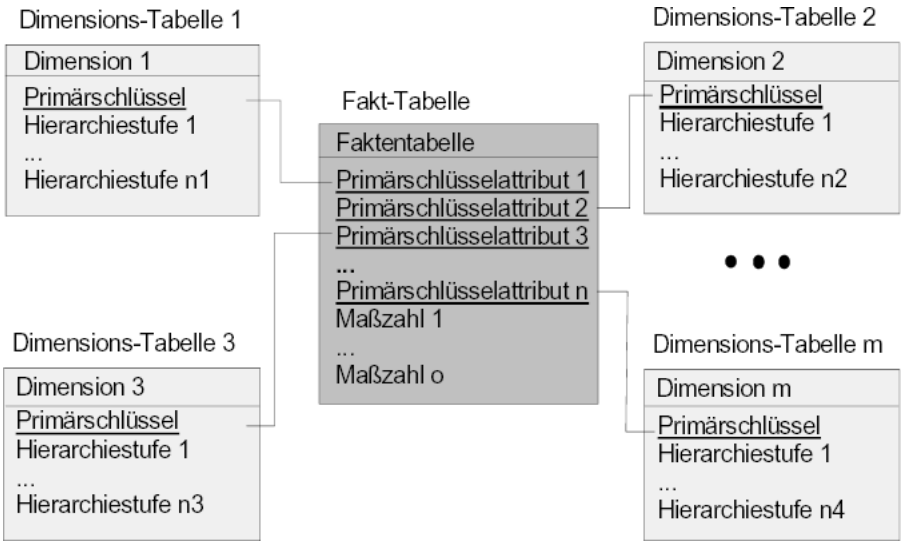

Abbildung 2.2: Prinzipieller Aufbau des Star Schemas (aus [BoeUl00, S. 12])

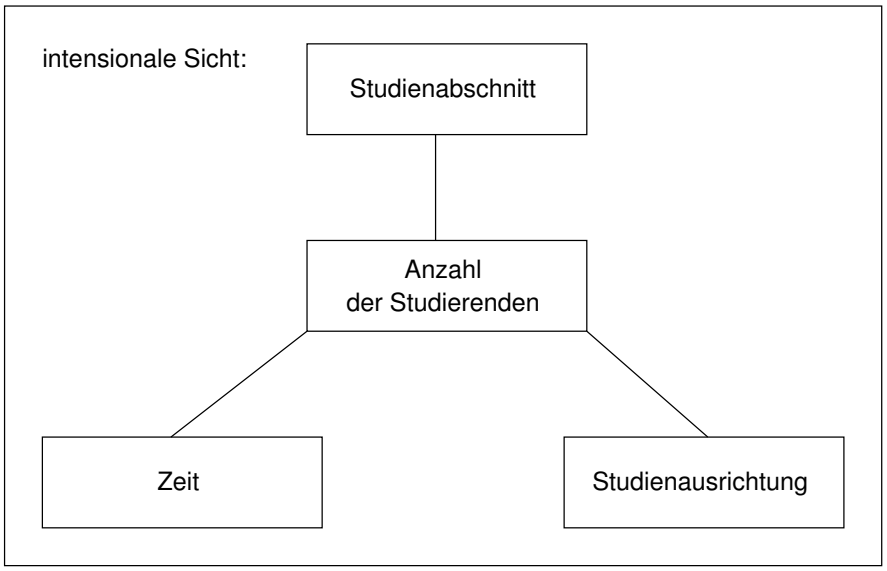

Abbildung 2.3: Star Schema an einem Beispiel aus dem Hochschulbereich, intensionale Sicht (angelehnt an [BoeUl00, S. 10])

Faktentabelle

Zeit	Studienabschnitt	Studienausrichtung Universität	Studienausrichtung Studiengang	Anzahl der Studierenden
2003–SS	Grundstudium	FHGI	MI	86
2004–SS	Hauptstudium	FHGI	I	78
2000–WS	Hauptstudium	FHGI	BMT	16
1999–SS	Grundstudium	UNIGI	L5	125
. . .	. . .	. . .	. . .	. . .

Dimensionstabelle Studienausrichtung

Hierarchiestufe Universität	Attribut Studienort	Hierarchiestufe Fakultät	Hierarchiestufe Studiengang
FHGI	Friedberg	MNI	MI
FHGI	Gießen	MNI	I
FHGI	Gießen	KMUB	BMT
UNIGI	Gießen	EW	L5
. . .	. . .	. . .	. . .

Dimensionstabelle Studienabschnitt

. . . . . .

Dimensionstabelle Zeit

. . . . . .

Abbildung 2.4: Star Schema an einem Beispiel aus dem Hochschulbereich, extensionale Sicht (angelehnt an [BoeUl00, S. 10])

  - Eine Hierarchie ist nicht durch dedizierte Schemaelemente repräsentiert, also nur implizit statt explizit enthalten.

- Fakttabellenhierarchien i.S.d. Constellation-Schemas sind nicht modellierbar.

Wesentliche Unterschiede zu anderen Schematypen zur logischen Modellierung Der wesentliche Unterschied von Star Schema und Snowflake Schema ist: im Star Schema sind die Dimensionstabellen noch vollständig denormalisiert, wie in Abbildung 2.4 ersichtlich. Sie entsprechen damit dem Ergebnis von Joins über die Tabellen aller Hierarchiestufen im Snowflake Schema [AbSaSa01, S. 15]. Mit einem Star Schema kann darüber hinaus nur ein einziger Datenwürfel auf Tabellen abgebildet werden. Natürlich kann diese Abbildung für jeden Datenwürfel wiederholt werden, es entsteht dann eine unverbundene Menge von Star Schemata wie in Abbildung 2.5. Werden diese anschließend entsprechend den gemeinsamen Dimensionen und Beziehungen zwischen den Fakten zusammengefasst, entsteht ein Constellation Schema oder Galaxy Schema wie in Abbildung 2.8 [Schw03, S. 50].

Das temporale Star Schema Ein bekanntes Problem im Data Warehousing ist, dass die Struktur von Dimensionen selbst oft zeitabhängig ist. Im Beispiel aus dem Hochschulbereich (Abbildung 2.3) etwa dann, wenn ein Studiengang in einen anderen Fachbereich umklassifiziert wird. Das Data Warehouse muss nun verschiedene Versionen der Dimensionen verwalten, um die Studenten vor und nach solch einer Umklassifizierung dem richtigen Fachbereich zuordnen zu können. Eine Möglichkeit zur Historisierung ist es, das Star Schema (oder auch Snowflake Schema) zu einer temporalen Version zu erweitern; dabei werden alle Tupel aller Relationen mit Zeitstempeln versehen und also auch alle anderen Dimensionen unter der Dimension »Zeit« betrachtet. Eine ausführliche Darstellung bietet [Ma00, S. 42-44].

Dimensional Modeling nach Kimball Dimensional Modeling ist eine Entwurfsmethodik für Data Warehouses, die zu unverbundenen Star Schemata führt [MoKo00, S. 4]. Das Dimensional Modeling wurde allein aufgrund praktischer Erfahrungen entwickelt. Weder basiert es auf einer mathematischen oder formalen Theorie (wie etwa das Relationenmodell als solches), noch wurde es jemals empirisch getestet [MoKo00, S. 2]. Das Dimensional Modeling hat einerseits das Ziel, die logische Struktur der

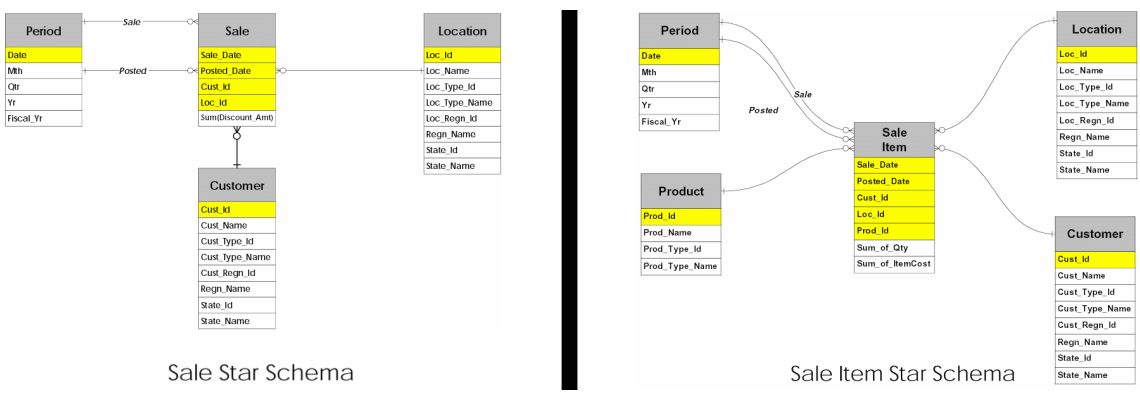

Abbildung 2.5: Unverbundene Star Schemata als Vorbereitung zum Constellation Schema [MoKo00, S. 8]

Datenbank einfach zu halten, damit Endbenutzer sie verstehen können und in der Lage sind, Abfragen an die Datenbank zu formulieren. Andererseits sollen die Abfragen möglichst effizient ausgeführt werden. Das Star Schema eignet sich dazu in besonderer Weise: die Anzahl der Tabellen und besonders der Beziehungen zwischen Tabellen ist minimal. Wenige Beziehungen bedeuten wenige Joins und steigern damit die Effizienz der Abfragen [GuMa00, S. 5]. Kimball verzichtet beim »Dimensional Modeling« aus solchen Effizienzgründen darauf, das Star Schema durch Normalisierung der Dimensionstabellen zu einem Snowflake Schema weiterzuentwickeln: so wären zwar Dimensionshierarchien explizit modellierbar, der gesparte Speicherplatz jedoch sei irrelevant während die Abfrageperformanz wesentlich verschlechtert würde; letzteres geschieht, weil eine Anzahl von Joins notwendig wird. [AbSaSa01, S. 13]

2.6 Developed Star Schema

Dieser Abschnitt orientiert sich an [BoeUl00, S. 13-14]. Das Developed Star Schema ist eine von SAP entwickelte Variante des in Abschnitt 2.5 vorgestellten einfachen Star Schemas. Im Developed Star Schema heißt der bekannte Datenwürfel »*InfoCube*«. Das *InfoCube* Star Schema ist der Teil des Developed Star Schema, der die Funktionalität des einfachen Star Schema übernimmt. Ein weiterer Teil ist von diesen InfoCubes unabhängig; er enthält beliebig viele sog. *Master Data*-, *Text*- und *Hierarchy-Tables*. Diese können mit beliebigen *InfoCubes* über Zwischentabellen mit speziellen systemvergeben Attributen (*Set IDs*) verbunden werden. Abbildung 2.6 zeigt ein mit dem Developed Star Schema modelliertes Beispiel, das noch einige erklärungswürdige Eigenschaften hat:

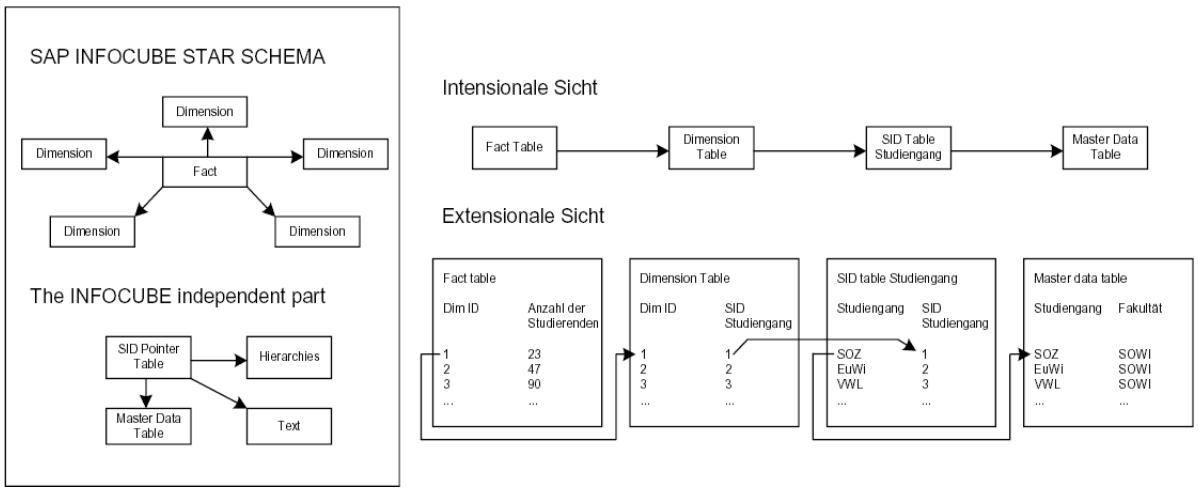

Abbildung 2.6: Beispiel zum Developed Star Schema aus dem Hochschulbereich [BoeUl00, S. 14]

- Beim Developed Star Schema werden ausschließlich künstlich generierte Identifikationsschlüssel als Primärschlüssel verwendet. Jeder dieser Identifikationsschlüssel, der dazu dient eine Fremdschlüsselbeziehung zwischen der Fakttabelle und einer Dimensionstabelle zu realisieren, heißt *Dim ID*. Eine *Dim ID* bezeichnet eine Dimensionsposition. Die *Dim IDs* aller Dimensionstabellen zusammengenommen bilden den Primärschlüssel der Fakttabelle. Zu beachten ist, dass in Abbildung 2.6 verkürzend nur eine *Dim ID* in die Fakttabelle aufgenommen wurde (diejenige für die näher betrachtete Dimension »Studienausrichtung«).

- »Es lassen sich in *Master Data Tables* zwei verschiedene Arten von Attributen unterscheiden: *Navigational Attributes*, mit denen Navigationsoperationen, wie z.B. *Drill Down* oder *Roll Up*, in den InfoCube-Datenbeständen möglich sind und *Reporting Attributes*, die in Berichten Verwendung finden, aber keine Navigationsmöglichkeiten bieten.« [BoeUl00, S. 14]. In Abbildung 2.6 finden wir das *Navigational Attribute* »Fakultät« vor, mit dem verschiedene Studiengänge zusammengefasst behandelt werden können (sog. *Roll Up*).

- Eine Dimensionstabelle kann die vollständigen Informationen über die Hierarchie in Form von Attributen enthalten, ähnlich wie beim einfachen Star Schema (vgl. Abbildung 2.4). Alle Attribute einer Dimensionstabelle außer der *Dim ID* heißen *Characteristics*.

- Hierarchien müssen jedoch nicht innerhalb der Dimensionstabelle gespeichert werden. In Abbildung 2.6 wird gezeigt, wie eine Dimensionstabelle über eine Zwischentabelle »*SID Table Studiengang*« mit einer *Master Data Table* verbunden werden kann. Eine solche Zwischentabelle kann ebenso eine Dimensionstabelle und eine *Hierarchy Table* verbinden, in die die Dimensionshierarchie ausgelagert werden kann. Es ist möglich, mehrere parallele externe Hierarchien für eine *Characteristic* (hier die *SID Studiengang*) zu definieren, indem man mehrere Zwischentabellen und so auch mehrere *Hierarchy Tables* verwendet.

- Eine *Characteristic* kann mit einer korrespondierenden *Master Data Table* verbunden sein (wie in Abbildung 2.6 gezeigt), muss es aber nicht.

- Das Developed Star Schema bietet sogar Internationalisierung: eine *Characteristic* kann mit textlichen Beschreibungen in sprachspezifischen *Text Tables* verbunden sein.

- »Der Einsatz künstlicher Schlüssel erlaubt die Vergabe von Nullwerten bei einem Teil der Primärschlüsselattribute und ermöglicht dadurch unausgeglichene Hierarchien (Unbalanced Hierarchies) und N:M-Beziehungen innerhalb einer Dimension.« [BoeUl00, S. 13-14]. Mit dieser Erweiterung kann im Vergleich zum einfachen Star Schema bedeutend mehr Semantik abgebildet werden[3].

Die genannten Erweiterungen bewirken: »Beim Developed Star Schema der SAP AG handelt es sich um eine vielversprechende Variante des klassischen Star Schemas« [BoeUl00, S. 14].

2.7 Snowflake Schema

»Snowflake Schemata sind Star Schemata mit expliziten Dimensionshierarchien, wie man sie aus der Normalisierung der Dimensionstabellen erhält.« (nach [AbSaSa01, S. 15])

Vom Star Schema zum Snowflake Schema Im Star Schema sind die Hierarchiestufen der Dimensionen nur implizit enthalten, denn die Dimensionstabelle ist vollständig denormalisiert (siehe dazu Abschnitt 2.5). Ausgehend vom Star Schema (Abbildung 2.5 links) erhält man das Snowflake

[3] Sofern beim einfachen Star Schema eben keine künstlichen Identifikationsschlüssel als Primärschlüssel eingesetzt werden.

Schema (Abbildung 2.7), indem man alle Dimensionstabellen vollständig normalisiert. Es entstehen hierarchische Tabellenbeziehungen für jede Dimension, die die hierarchische Dimensionsstruktur explizit wiedergeben. Weil eine Dimensionstabelle im Star Schema mehrere unabhängige Hierarchien enthalten kann, können sich auch mehrere Hierarchien zwischen den Tabellen für eine Dimension im Snowflake Schema ergeben. [MoKo00, S. 8]

Als Vorteil des Snowflake Schema halten wir damit fest, dass ein weiteres Konzept multidimensionaler Datenstrukturen nun explizit modelliert werden kann, nämlich Dimensionshierarchien. Sie sind im logischen Schema sofort ersichtlich, was seine Lesbarkeit und damit Brauchbarkeit zumindest in der Phase der logischen Modellierung verbessert (nach [Ma00, S. 39]). Um solch eine Dimensionshierarchie zu nutzen müssen nun aber die zugehörigen Tabellen per Join verbunden werden - das Ergebnis ist eine denormalisierte Dimensionstabelle, genau wie sie beim Star Schema von Anfang an vorliegt. Der zusätzliche Aufwand dieser »unnötigen« Joins und die zusätzlichen Tabellen für Hierarchiestufen, die die Systemkomplexität »unnötig« erhöhen, führen z.B. Kimball dazu, das Snowflake Schema als unnötig abzulehnen [MoKo00, S. 9]. Dieser Nachteil im Einsatz wird von Kimball also gravierender eingeschätzt als der Vorteil expliziter Hierarchien und der so auch minimierten Redundanz.

Das Star Cluster Schema Eine Variante des Snowflake Schemas sei genannt: das Star Cluster Schema nach Moody und Kortink, eingeführt und genau dargestellt in [MoKo00]. Dabei werden Dimensionstabellen teilweise normalisiert wie im Snowflake Schema, teilweise denormalisiert wie im Star Schema dargestellt. Und zwar bleiben nur diejenigen Teile von Dimensionshierarchien normalisiert, die gleichzeitig Bestandteil mehrerer Dimensionshierarchien sind (sog. »Subdimensionen«, die größten gemeinsamen Elemente mehrerer Dimensionen). [MoKo00, S. 9-10] [AbSaSa01, S. 15]

2.8 Constellation Schema und Galaxy Schema

Constellation Schema und seine allgemeinere Form, das Galaxy Schema, sind Möglichkeiten, mehrere separate Schemata zu konsolidieren, die jeweils einen Datenwürfel darstellen. Dabei ist es gleich, welchem der bisher dargestellten Schematypen die solcherart integrierten Schemata angehören: es können z.B. Star Schemata oder Snowflake Schemata in ihren verschiedenen Varianten sein [MoKo00, S. 1.10].

Warum mehrere Fakttabellen modellieren? Es ist zwar möglich, ein Data Warehouse durch einen einzigen Datenwürfel zu modellieren (sog. Hypercube-Ansatz; [Ma00, S. 37]). Dann wird jede Kennzahl unter allen Dimensionen betrachtet, auch unter solchen von denen sie eigentlich unabhängig ist, von der nur andere Kennzahlen abhängig sind. Gibt es viele solcher unnötigen Tupel »$(\mathrm{Kennzahl}, \mathrm{Dimension})$«, so steigt der Speicherplatzbedarf bei der üblichen Abbildung auf das relationale Datenmodell enorm an [Ma00, S. 36-37]. Weiterhin haben Fakttabellen je mehr Datensätze, je mehr Dimensionen sie haben und je mehr Dimensionspositionen diese Dimensionen haben. So entstehen bei einem Hypercube-Ansatz meist große Fakttabellen und entsprechend schlechte Antwortzeiten [Ma00, S. 39]. Außerdem sind Hypercubes oft unübersichtlich - als Richtwert sollten höchstens 10 Dimensionen für den Endbenutzer sichtbar sein.

Diese Gründe machen es sinnvoll, mehrere Fakttabellen zu modellieren (vgl. [Ma00, S. 39]). Das kann natürlich durch mehrere, unverbundene Star Schemata geschehen (siehe Abschnitt 2.5) oder auch durch mehrere Snowflake Schemata in seinen verschiedenen Varianten. Dabei sind dann jedoch gemeinsame Dimensionen mehrfach vorhanden. Solche Redundanz wird durch das Constellation Schema und das Galaxy Schema vermieden. Das Constellation Schema ist auch als »Fact Constellation Schema« bzw. »Fact/Constellation Schema« (nach [Ra95]) bekannt.

Ein zusätzlicher spezieller Grund für die Entwicklung des Constellation Schema Betrachten wir ein Problem des Star Schema am Beispiel von Verkäufen (»*Sale*«), die jeweils mehrere Positionen (»*Sale Item*«) umfassen können. Diese $1 : n$-Beziehung zwischen *Sale* und *Sale Item* lässt vermuten, dass *Sale* als Dimension zu *Sale Item* modelliert werden kann. In diesem Fall ist das unmöglich, weil *Sale* selbst eine Kennzahl *Discount_Amt* besitzt: Rabatt zu einem Verkauf als Geldmenge. Weil der Rabatt sich z.B. aus einem pauschalen und einem prozentualen Anteil zusammensetzen kann, kann diese Kennzahl nicht wie bei einem Rabatt-Prozentsatz auf die *Sale Items* aufgeteilt werden. Für *Sale* muss daher eine eigene Fakttabelle erstellt werden. Verwendet man Star Schemata, so führt das zu zwei unverbundenen Schemata (Abbildung 2.5), d.h. die $1 : n$-Beziehung zwischen *Sale* und *Sale Item* bleibt unberücksichtigt, eine automatische drill-down Operation (Abstieg in die detaillierter Ebene, hier »*Sale Item*«) ist unmöglich.

Ein Constellation Schema wie in Abbildung 2.8 stellt die benötigte hierarchische Verbindung von Fakttabellen zur Verfügung [AbSaSa01, S. 15], wobei die Fakttabellen der detaillierteren Ebene alle Dimensionen der übergeordneten Ebene »erben«. Dadurch wird ein automatisches drill-down möglich [MoKo00, S. 7]. Es ist nicht korrekt, dass das Constellation Schema dazu dient, »die aggregierten Werte der verschiedenen Verdichtungsstufen der Dimensionshierarchie zu integrieren« [Ma00, S. 39], sie also in vorberechneter Form vorrätig zu halten. Das Constellation Schema integriert Verdichtungsstufen nichtdimensionaler Strukturen!

Vom Constellation Schema zum Galaxy Schema Zwar werden Galaxy Schema und Constellation Schema in der Literatur manchmal als synonym dargestellt (etwa [Ez04, S. 116], [Schw03, S. 50 (+16)]). Wir unterscheiden beide jedoch im Sinne von [MoKo00, S. 7]: Sowohl Constellation Schema als auch Galaxy Schema sind konsolidierte Schemata [Schw03, S. 50 (+16)]. Auch beim Galaxy Schema werden Dimensionstabellen gemeinsam genutzt [AbSaSa01, S. 15]. Der Unterschied zum Constellation Schema besteht darin, dass die Fakttabellen nicht hierarchisch oder in anderer Form direkt verbunden sein *müssen* - aber sehr wohl sein *können* [MoKo00, S. 8].

2.9 Zusammenfassende Bewertung

Wie zu Beginn dieses Kapitels versprochen, folgt nun der zusammenfassende Vergleich der logischen Schemata entsprechend dem erarbeiteten Kriterienkatalog. Dabei liefert die jeweilige Darstellung des Schemas in diesem Kapitel leider noch nicht alle Begründungen für seine Bewertung in diesem Abschnitt. Wo solche fehlen, konsultiere man der Reihe nach folgende Werke:

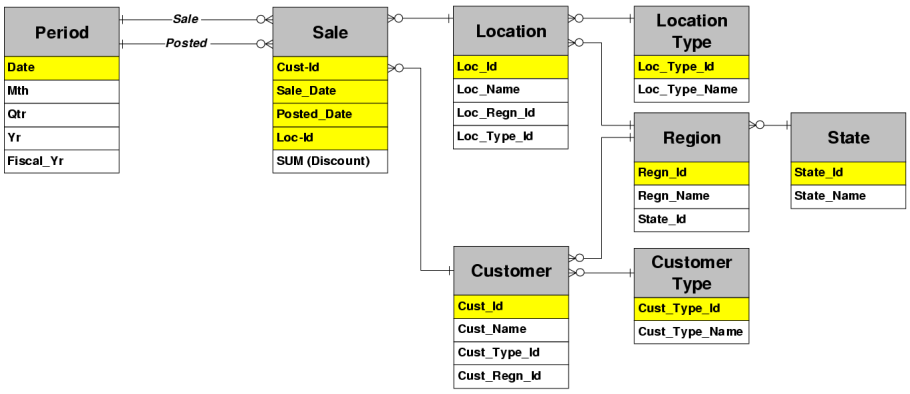

Abbildung 2.7: Snowflake Schema an einem betriebswirtschaftlichen Beispiel [MoKo00, S. 9]

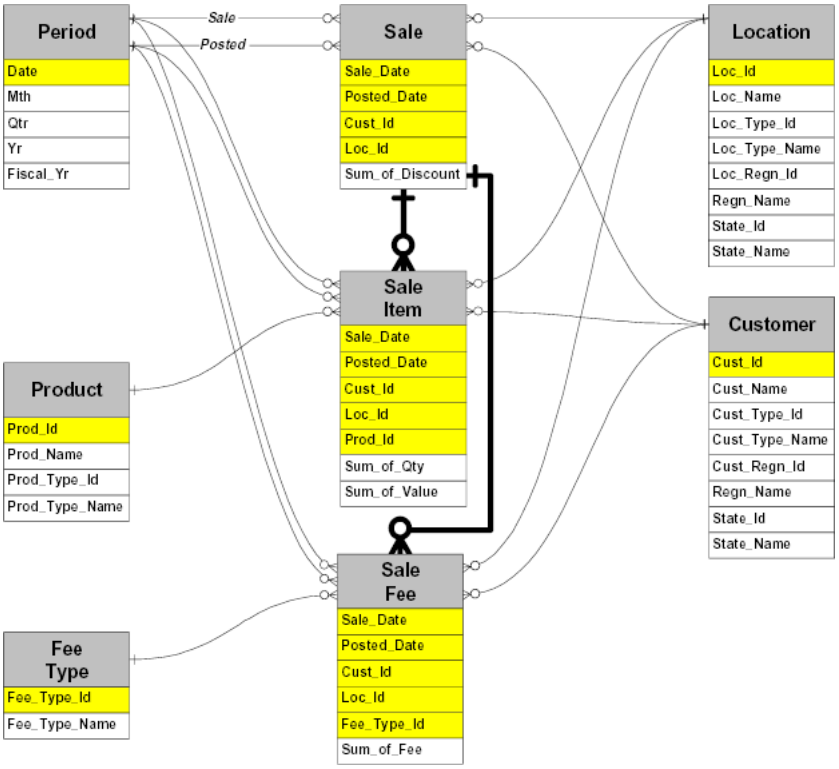

Abbildung 2.8: Beispiel zum Constellation Schema (aus [MoKo00, S. 8])

- [MoKo00, S. 3ff.]
- [BoeUl00, Kap. 1-3.1; 4-4.1; 4.3; 5; 7]
- [Schw03, Kap. 4.2 S. 47 (+16)]
- [AbSaSa01, S. 12ff.]
- [vM00, Kap. 2 S. 13 (+24)]
- [Ma00, Kap. 4.3.2 ff. S. 38]

Constellation Schema und Galaxy Schema können sowohl auf Star Schema als auch auf Snowflake Schema in den jeweiligen Varianten angewandt werden (Abschnitt 2.8). Der Schemavergleich in Tabelle 2.9 berücksichtigt diese Kombinationen in angemessenem Umfang.

	Flat Schema	Terraced Schema	Star Schema	Developed Star Schema	Snowflake Schema	Star Cluster Schema	Constellation & Star Schema	Galaxy & Star Cluster Schema
Verständlichkeit für Endbenutzer	◎	⊙	●	⊙	○	⊙	⊙	◎
Effizienz typischer Abfragen	●	●	⊙	⊙	○	⊙	⊙	⊙
Abbildung für reichhaltige Semantik	◎	◎	⊙	●	●	⊙	●	●
Orientierung am Zieldatenbanksystem	●	●	●	●	●	●	●	●
Wartbarkeit	○	○	◎	⊙	⊙	⊙	⊙	●
Werkzeugunterstützung	●	●	●	●	●	●	●	●

Die Anforderung wird ● gut erfüllt, ⊙ zufriedenstellend erfüllt, ◎ weniger erfüllt, ○ nicht erfüllt.

Abbildung 2.9: Vergleich von Schematypen zur logischen Modellierung multidimensionaler Datenstrukturen

Kapitel 3

Kritik am Einsatz der logischen Modelle

Einleitung Dieses Kapitel spricht multidimensionalen Datenstrukturen keineswegs ihre Existenzberechtigung ab. Ihr Einsatz im Bereich OLAP kann durchaus als erfolgreich bezeichnet werden: Kundenzufriedenheit und Amortisation der getätigten Investitionen werden als gut bezeichnet [GrCoOl96] (nach [MoKo00, S. 1]). Auch OLAP selbst ist damit eine sinnvolle, berechtigte Anwendung elektronischer Datenverarbeitung.

Dieses Kapitel kritisiert nicht Modellvielfalt und Stückwerk in der Modellierung multidimensionaler Datenstrukturen, obwohl das durchaus berechtigt ist (vgl. [Bl00, S. 11 (+14)], [BoeUl00, S. 32]). Es wird auch kein kritischer Vergleich der logischen Modelle aufgestellt, um das geeignetste zu ermitteln. Vergleiche dazu Kapitel 2.

Dieses Kapitel kritisiert die wartungsintensive Art des Umgangs mit den logischen Schemata in der Praxis. Das Problem der Wartung logischer Schemata wird zuerst in verschiedenen Data Warehouse-Architekturen aufgezeigt. Anschließend wird eine Alternative vorgeschlagen, eingebettet in eine idealtypische Data Warehouse-Architektur.

Wartungsintensive logische Schemata in bisherigen Data Warehouse-Architekturen Ein bedeutendes Problem in der Praxis des Data Warehousing ist der große manuelle Wartungsaufwand für die Anpassung der logischen Schemata. Diese Schemata verwenden multidimensionale Datenstrukturen als eine für Endbenutzer oder Werkzeuge aufbereitete Darstellungsform. Diese Aufbereitung erzeugt den Wartungsaufwand:

- Ändern sich die Datenstrukturen der Quellschichten, entsteht Wartungsaufwand an den multidimensionalen Datenstrukturen im logischen Schema.
- Die logischen Schemata sind auf die aktuellen, kurzlebigen Analyseanforderungen der Benutzer abgestimmt. Ändern sie sich, entsteht Wartungsaufwand am logischen Schema. Es macht dabei keinen Sinn, ein logisches Schema zu erstellen, das alle möglichen Analysen abdeckt, weil seine Komplexität für Endbenutzer zu groß wäre.

Dieses Problem tritt in allen bisher bekannten Data-Warehouse-Ansätzen auf, stets dort wo Daten für den Endbenutzer aufbereitet werden:

- Beim Dimensional Modeling nach Kimball verwendet das ganze Data Warehouse multidimensionale Datenstrukturen auf logischer Ebene und richtet sich nach den aktuellen Analyseanforderungen seiner Nutzer [MoKo00, S. 4]. In oben genannten Fällen ist dann eine Schema-Evolution für das ganze Data Warehouse nötig. Der benötige Wartungsaufwand steigt noch, weil meist keine rekonziliierte Datenbasis (Basisdatenbank) zur Verfügung steht.
- In einer anderen Architektur verwendet das ganze Data Warehouse multidimensionale Datenstrukturen, aber nur die Data Marts[1] richten sich nach den aktuellen Analyseanforderungen der Benutzer [Qu03, S. 10ff.]. Änderungen in den Quellschichten führen zur Schema-Evolution im ganzen Data Warehouse, Änderungen der Analyseanforderungen aber nur zur Schema-Evolution in den Data Marts. Weil in dieser Architektur eine Basisdatenbank zur Verfügung steht, hält sich der Wartungsaufwand hier in Grenzen.
- In fortgeschrittenen Data Warehouse-Architekturen werden im Data Warehouse selbst auf logischer Ebene keine multidimensionalen Datenstrukturen verwendet, sondern erst in den Data Marts (vgl. etwa [MoKo00], [vM00, S. 58-62 (+24)]). Die Data Marts richten sich nach den aktuellen Analyseanforderungen der Benutzer. In einer solchen Architektur, die bereits deutlich von der gängigen Architektur abweicht [vM00, Fußnote 49], ist ebenfalls manuelle Schema-Evolution nötig; sie beschränkt sich allerdings auf die Data Marts.

Wartungsfreie logische Schemata durch Data Marts mit Cache-Struktur Ein Data Warehouse ist eine Datenbank, die als eine einzige, konsistente, organisationsweite Informationsquelle für entscheidungsunterstützende Informationssysteme fungiert [MoKo00, S. 1] (nach [In96], [Lo94]). Konkret ergibt sich die genannte Zweckbindung aus Maßnahmen, die Abfragen aus dem Bereich Data Mining und OLAP effizienter machen, wodurch aber gleichzeitig andere Anwendungen des Data Warehouse eingeschränkt oder verunmöglicht werden [Qu03, S. 10-11] [MoKo00, S. 4]. Bei den fortschrittlichen Data Warehouse-Architekturen, von denen wir im folgenden ausgehen, werden diese Maßnahmen nur auf die Data Marts angewandt (vgl. etwa [MoKo00], [vM00, S. 58ff. (+24)]). Die wichtigsten dieser Maßnahmen wurden in Kap. 1.3 genannt.

Alle Daten in derart behandelten Data Marts sind oder waren auch in den operativen Datenbanken enthalten. Die Data Marts enthalten eine Auswahl der Daten aus den operativen Datenbanken in einer auf Analyseanwendungen optimierten Form. Das macht eine Cache-Architektur denkbar: Data Marts als effizienter Zwischenspeicher für Analyseanwendungen. Konsequent umgesetzt, wird manuelle Schema-Evolution der logischen Schemata unnötig. Ein Cache ist schließlich für seine Benutzer transparent: er definiert keine eigenen Datenstrukturen, an denen Änderungen anderer Datenstrukturen nachvollzogen werden müssen. Alle denkbaren Strukturen von Data Marts müssen dann aus Metadaten abgeleitet werden können, die im Data Warehouse bereitstehen. Sie definieren z.B., welche Attribute Fakten sein können, welche Tabellen Dimensionen sein können und in welchen Kombinationen Fakten und Dimensionen auftreten können.

[1] Ein System, das eine Teilmenge der Daten des Data Warehouse verwaltet.

Man könnte nun einwenden, wartungsfreie Data Marts seien hier nur möglich indem der Wartungsaufwand auf das Data Warehouse und die operativen Datenbanken abgewälzt wurde. Tatsächlich ergeben sich dort einige zusätzliche Aufgaben. Der Gesamtaufwand sinkt jedoch. Wartungsaufgaben haben schließlich die Eigenschaft, sich gegenseitig aufheben zu können; etwa in Fällen, in denen Benutzer Analyseanforderungen haben, die zumindest teilweise schon einmal vorkamen. Effizienter ist es also, manuelle Wartung durch eine allgemeine Lösung zu ersetzen, die dann programmgestützt in konkreten Situationen eingesetzt wird. Etwa so:

- Die Benutzer definieren ihre Data Marts selbst. Geänderte Analyseanforderungen bedeuten nun keine Wartungsaufgaben für EDV-Mitarbeiter an den Data Marts mehr. Stattdessen bedienen sich die Benutzer aus dem Data Warehouse, was den Zielen des Data Warehousing ohnehin besser entspricht. Damit es nicht ähnlich komplex ist, einen eigenen Data Mart zu definieren wie die für Endbenutzer unzumutbare Aufgabe, einen Hypercube des gesamten Data Warehouse zu benutzen, sollte ein Programm die Komplexität verbergen. Beispielsweise können in einer grafischen Oberfläche Datenwürfel definiert werden, wobei für jeden Würfel aus Listen Untermengen der verfügbaren Fakten, Dimensionen und Aggregationsstufen gewählt werden können. Das Programm übernimmt es dann, den Data Mart derart zu erzeugen, dass er z.B. auch den speziellen Anforderungen eines verwendeten OLAP-Tools genügt.
- Die Administratoren der operativen Datenbank definieren die möglichen multidimensionalen Sichten. Sie verwalten also die erwähnten Metadaten. Der Aufwand ist vergleichsweise gering: [MoKo00] beschreibt ein Verfahren, mit dem Entity-Relationship-Modelle in multidimensionale Sichten auf Daten transformiert werden können. Es ist zum guten Teil mechanisch, so dass die meisten Änderungen an operativen Datenbanken automatisiert an den Metadaten und Data Marts nachvollzogen werden können. Menschliche Beteiligung ist hauptsächlich bei der initialen Erstellung der Metadaten in Form von Modellbildung nötig, und nicht wie bisher bei jeder Änderung an Data Marts.

Als weiterer Vorteil dieses Verfahrens ist zu nennen, dass Data Marts nun kohärent zum Data Warehouse gehalten werden können, vergleichbar mit Cache-Kohärenz. Dazu wird anhand der ja stets aktuellen Metadaten geprüft, ob ein Data Mart eine gültige Sicht auf ein Data Warehouse darstellt. Eine automatische Anpassung an Änderungen in den operativen Datenbanken (und damit auch im Data Warehouse) ist dann ebenfalls möglich, etwa das Hinzufügen von Dimensionspositionen wie einer Artikelnummer.

Anhang A

Glossar

Aggregation Der Vorgang, mehrere Kennzahlen entlang aufeinanderfolgender Dimensionspositionen einer Dimension zu einer Kennzahl höherer Granularität zusammenzufassen. Dazu wird eine Aggregationsfunktion wie etwa Summenbildung oder Mittelwertbildung verwendet.

Basisdatenbank Auch Operational Data Store (ODS). Eine physische Datenbank, die eine integrierte Sicht auf beliebige Daten bietet. Die in ihr enthaltenen Daten sind noch nicht auf einen bestimmten Zweck (etwa OLAP-Analyse) festgelegt, weshalb die Schemata dieser Datenbank eine verlustfreie Transformation der Daten aus den integrierten Datenbeständen bieten sollen, d.h. auch ohne Vorverdichtung. Das Data Warehouse sollte seine integrierten Daten aus der Basisdatenbank erhalten. Vgl. [BaGue01, S. 515], [Qu03, S. 10].

Business Intelligence Der Prozess, Daten in Information zu transformieren und danach durch einen Entdeckungsprozess in Wissen [MuBe98]. Nach [Ma00, S. 7].

Data Mart Ein System, das eine Teilmenge der Daten des Data Warehouse verwaltet. Er ist gegenüber dem Data Warehouse i.d.R. redundant [BaGue01, S. 515-516].

Data Warehouse »A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions.« [Qu03, S. 8] Strenggenommen bezeichnet Data Warehouse tatsächlich nur diese zentrale Datenbank (im Unterschied zum Data Warehouse-System) [Qu03, S. 7-8].

Data Warehouse-System Ein Data Warehouse zusammen mit allen Zusatzprogrammen und weiteren, unterstützenden Datenbanken. In der Literatur oft nicht vom Data Warehouse getrennt [Qu03, S. 7-8].

Datenmodell Formalsprachlicher Beschreibungsrahmen für Schemata. Er stellt Mittel zur Verfügung, mit dem solche Schemata und auf ihren Instanzen mögliche Operationen definiert werden können. Nach [Bl00, S. 25 (+14)].

Datenmodellierung Die Erfassung und Beschreibung von Datenstrukturen [GaGl98, S. 497].

Datenwürfel Eine multidimensionale Matrix. Aufgespannt von beliebig vielen Dimensionen, enthält er soviele Zellen wie das kartesische Produkt aller Dimensionen Elemente hat. Jede Zelle kann eine oder mehrere Kennzahlen enthalten. Nach [BaGue01, S. 520].

Dimension Im multidimensionalen Datenmodell ist eine Dimension eine Entität, die eine Analysesicht auf Daten definiert (etwa Zeit, Artikel, Region) [BaGue01, S. 517]. Dimensionen spannen einen Datenwürfel auf und stellen den Kontext dar, in dem jede Kennzahl des Datenwürfels zu interpretieren sind. Eine Dimension besteht aus Dimensionspositionen.

Dimensional Modeling Eine von Ralph Kimball begründete Entwurfsmethodik für Data Warehouses, die zu unverbundenen Star Schemata führt [MoKo00, S. 4]. Wesentliche Ziele sind: Verständlichkeit für Endbenutzer und Effizienz typischer Analysen [GuMa00, S. 5].

Dimensionsstruktur Die Menge der Beziehungen zwischen den Dimensionselementen einer Dimension. Insbesondere sind das hierarchische Beziehungen, die eine oder mehrere Dimensionshierarchien bilden können. Nach [GaGl98, S. 495].

Dimensionsposition Auch Dimensionselement. Basisgranular einer Dimension [BaGue01, S. 517]. Beispiel: die Dimension »Artikel« enthält alle Produkte als Dimensionspositionen.

Fakt Im Dimensional Modeling der Inhalt einer Würfelzelle (also eine oder mehrere Kennzahlen) zusammen mit ihrem Kontext (d.h. den Dimensionspositionen, die diese Würfelzelle identifizieren). [GuMa00, S. 5]

Fakttabelle In der logischen Modellierung multidimensionaler Datenstrukturen für relationale Datenbanken (etwa durch Star Schema oder Snowflake Schema) gibt es zwei Arten von Tabellen: Fakt- und Dimensionstabellen. Eine Fakttabelle enthält alle Fakten, die zu einem Datenwürfel gehören. Vgl. [GuMa00, S. 5].

Kennzahl Kennzahlen repräsentieren quantitative Größen im multidimensionalen Datenmodell, die nach qualitativen Größen, den Dimensionen, geordnet sind. Die Kennzahl »Anzahl der Studierenden« kann z.B. nach den Dimensionen Zeit, Studienabschnitt und Studienausrichtung geordnet sein. [BoeUl01, S. 2]. Eine oder mehrere Kennzahlen bilden den Inhalt einer Würfelzelle.

konzeptionelle Ebene Die Ebene höchster Abstraktion beim Entwurf eines Datenbanksystems. Die gefundenen Datenstrukturen werden in einem semantischen Datenmodell formuliert und sind allein am Problembereich orientiert, unabhängig vom Zieldatenbankverwaltungssystem und den späteren physischen Datenstrukturen.

logische Datenmodellierung Abbildung der semantischen Datenmodellierung auf das Datenmodell eines konkreten Datenbankverwaltungssystems [GaGl98, S. 497].

logische Ebene Abstraktionsebene beim Entwurf eines Datenbanksystems, die unter der konzeptionellen Ebene angesiedelt ist. Die erstellten logischen Schemata sind vom Zieldatenbankverwaltungssystem abhängig, aber unabhängig von den späteren physischen Datenstrukturen. Logische Schemata enthalten als Abbildung semantischer Schemata die Beschreibung des Problembereichs in einer oft noch gut verständlichen Form. Nach [AbSaSa01, S. 12].

Modell »In jeder Phase des Entwurfsprozesses werden spezifische Modelle verwendet, um das Schema, welches das Ergebnis der jeweiligen Entwurfsphase ist, zu beschreiben. Das Schema gibt jeweils die Struktur einer konkreten Datenbank an, während das Modell die Sprache darstellt, in der das Schema beschrieben wird.« [Schw03, S. 35]

Multidimensionalität Im Bereich der Datenstrukturen ist damit die logische Anordnung quantitativer Größen anhand mehrerer sachlicher Kriterien, den Dimensionen, gemeint [GaGl98, S. 494]. Zu betonen für *Multi*dimensionalität ist: ein Datum kann *gleichzeitig* nach *mehreren* dieser Dimensionen (»Anordnungen«) angeordnet sein.

multidimensionale Datenstruktur Eine Datenstruktur, die geeignet ist, multidimensionale Daten zu repräsentieren.

Schema Die Struktur einer konkreten Datenbank, beschrieben in einem Modell als einer formalen Sprache [Schw03, S. 35]. Es gibt Schemata und Modelle für die konzeptionelle, logische und physische Ebene.

Schema-Evolution Die Tätigkeit, ein Schema anhand geänderter äußerer Bedingungen weiterzuentwickeln. Ändert sich z.B. das konzeptionelle Schema, so ist eine Schema Evolution für das logische und physikalische Schema nötig.

Schematyp Ein Schema, das sich durch spezifische Strukturierungsmerkmale auszeichnet, zeigt dadurch seine Zugehörigkeit zu einem Schematyp. In diesem Sinn sind »Snowflake-Schema« usw. nicht Schemata, sondern Schematypen [Schw03, S. 47].

semantische Datenmodellierung Möglichst verständliche Datenmodellierung auf einer abstrakten, datenbankunabhängigen Ebene [GaGl98, S. 497]. Semantische Datenmodellierung ist am Problembereich orientiert.

Star Join Join von Fakttabelle und Dimensionstabelle(n).

Würfelstruktur Die Menge der Würfel zusammen mit der Art, wie Dimensionen diesen Würfeln zugeordnet werden, betrachtet für eine spezifische multidimensionale Anwendung [GaGl98, S. 495].

Anhang B

Weiterführende Literatur

Die folgenden Werke trugen nicht zu diesem Dokument bei, sind aber thematisch so nah verwandt dass sie als Quellen bei zukünftigen Erweiterungen dieses Dokuments prinzipiell geeignet sind. Damit sind es auch gute Tipps für eine weitergehende Beschäftigung mit dem Thema. Alle diese Werke sind online erhältlich.

[Ab02] Abello, Alberto: YAM A Multidimensional Conceptual Model, Catalunya 2002, online im Internet: URL: `http://www.tdx.cesca.es/TESIS_UPC/AVAILABLE/TDX-1004102-091640/THESIS.pdf` [Stand 2004-05-13]

[Ca02] Carpani, Fernando: Multidimensional Models: A State of Art., o.O. 2002, online im Internet: URL: `http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0012.pdf` [Stand 2004-05-13]

[He01] Herden, Olaf: Eine Entwurfsmethodik für data warehouses (Diss.), Oldenburg 2001, online im Internet: URL: Archivserver der Deutschen Bibliothek `http://deposit.ddb.de/cgi-bin/dokserv?idn=964097281` [Stand 2004-05-13]

[Hi02] Hinrichs, Holger: Datenqualitätsmanagement in Data-warehouse-Systemen (Diss.), Oldenburg 2002, online im Internet: URL: Archivserver der Deutschen Bibliothek `http://deposit.ddb.de/cgi-bin/dokserv?idn=964461552` [Stand 2004-05-13]

[Ki0004] Kimball, Ralph et al.: Design Tips on Data Warehousing, 2000-2004, online im Internet: URL: `http://www.rkimball.com/html/designtips.html` [Stand 2004-05-13]. 50 im Volltext zugängliche Tipps.

[Ki9504] Kimball, Ralph et al.: Articles on Data Warehousing, 1995-2004, online im Internet: URL: `http://www.rkimball.com/html/articles.html` [Stand 2004-05-13]. Über 100 im Volltext zugängliche Fachartikel.

[Le03a] Lehmann, Peter: Prozesse im Data Warehousing - Gestaltungsprozesse im Überblick, o.O. 2003, online im Internet: URL: `http://www.prof-lehmann.de/extdoc/SS2003-DBS-02.pdf` [Stand 2004-05-13]

[Pi03] Pieringer, Roland: Modeling and implementing multidimensional hierarchically structured data for data warehouses in relational database management systems and the implementation into transbase (Diss.), München 2003, online im Internet: URL: Archivserver der Deutschen Bibliothek http://deposit.ddb.de/cgi-bin/dokserv?idn=969373791 [Stand 2004-05-13]

Die folgenden Werke trugen nicht zu diesem Dokument bei und sind thematisch weniger nah verwandt. Sie mögen aber wertvoll sein für eine eingehendere Beschäftigung mit multidimensionalen Datenstrukturen und dem Data Warehouse.

[FrKa03] Franconi, Enrico; Kamble, Anand: The GMD Data Model for Multidimensional Information: a brief introduction, Bozen-Bolzano 2003, online im Internet: URL: http://www.inf.unibz.it/%7Efranconi/papers/dawak-03.pdf [Stand 2004-05-13]

[Ho99] Holthuis, Jan: Der Aufbau von Data Warehouse-Systemen, 2. Aufl., Wiesbaden 1999

[KiRo02] Kimball, Ralph; Ross, Margy: The data warehouse toolkit : the complete guide to dimensional modeling, 2. Aufl., New York [u.a.] 2002

[Ki96] Kimball, Ralph: The data warehouse toolkit : practical techniques for building dimensional data warehouses, New York [u.a.] 1996

[Ki98] Kimball, Ralph: The data warehouse lifecycle toolkit : expert methods for designing, developing, and deploying data warehouses, New York [u.a.] 1998

[Ku99] Kurz, Andreas: Data Warehousing - Enabling Technology, Bonn 1999

[KuoJ] Kurz, Anke: Modellierung für Data Warehouse, Siegen 2004, online im Internet: URL: http://www-winfo.uni-siegen.de/winfo/deutsch/lehre/WS03-04/Mobis_Dateien/Mobis12.pdf [Stand 2004-05-13]

[Le03b] Lehmann, Peter: Data-Warehouse-Systeme, o.O. 2003, online im Internet: URL: http://www.prof-lehmann.de/extdoc/SS2003-DWS-01-Skript-Teil1.pdf [Stand 2004-05-12]. Zu diesem Skript gehört die Präsentation [Le03a].

[Ma96] Mattison, Rob: Data warehousing : strategies, technologies, and techniques, New York [u.a.] 1996

[Me04] Mehrwald, Christian: SAP Business Information Warehouse 3 : Architektur, Konzeption, Implementierung, 2. korr. Aufl., Heidelberg 2004

[Mu00] Mucksch, Harry; Behme, Wolfgang (Hrsg.): Das Data Warehouse-Konzept, 2. vollst. überarb. und erw. Aufl., Wiesbaden 2000

[Oehl00] Oehler, Karsten: OLAP, München/Wien 2000

[St03] Stührenberg, Insa: Seminarausarbeitung Data Warehousing, Oldenburg 2003, online im Internet: URL: http://www.diko-project.de/dokumente/ausarbeitungen/stuehrenberg.pdf [Stand 2004-05-12]

[ToJa98] Totok, Andreas; Jaworski, Ramon: Modellierung von multidimensionalen Datenstrukturen mit ADAPT - Ein Fallbeispiel, Braunschweig 1998, ISBN 3-930166-92-5, online im Internet: URL: http://www.wiwi.tu-bs.de/controlling/totok/Arbeitsbericht_ADAPT.pdf [Stand 2004-05-13]

[TrPaMaGo01] Trujillo, Juan; Palomar, Manuel; Gomez, Jaime; Song, Il-Jeol: Designing Data Warehouses with OO Conceptual Models, o.O. 2001, online im Internet: URL: http://www.dlsi.ua.es/~mpalomar/gold.pdf [Stand 2004-05-13]

Abbildungsverzeichnis

1.1 Der Datenwürfel (nach [Ma00, S. 21] und [ChSt98]) 8

1.2 Dimensional skalierte magnetische Messlatten als Veranschaulichung multidimensionaler Datenstrukturen 8

2.1 Beziehungen zwischen logischen Modellen für multidimensionale Datenstrukturen. Nach [AbSaSa01, S. 15], [BoeUl00, S. 9], [Schw03, S. 67-69]. 19

2.2 Prinzipieller Aufbau des Star Schemas (aus [BoeUl00, S. 12]) 19

2.3 Star Schema an einem Beispiel aus dem Hochschulbereich, intensionale Sicht (angelehnt an [BoeUl00, S. 10]) 20

2.4 Star Schema an einem Beispiel aus dem Hochschulbereich, extensionale Sicht (angelehnt an [BoeUl00, S. 10]) 20

2.5 Unverbundene Star Schemata als Vorbereitung zum Constellation Schema [MoKo00, S. 8] 21

2.6 Beispiel zum Developed Star Schema aus dem Hochschulbereich [BoeUl00, S. 14] . . 22

2.7 Snowflake Schema an einem betriebswirtschaftlichen Beispiel [MoKo00, S. 9] 26

2.8 Beispiel zum Constellation Schema (aus [MoKo00, S. 8]) 26

2.9 Vergleich von Schematypen zur logischen Modellierung multidimensionaler Datenstrukturen 27

Literaturverzeichnis

[AbSaSa01] Abello, Alberto; Samos, Jose; Saltor, Felix: A Data Warehouse Multidimensional Data Models Classification, Catalunya [u.a.] 2001, online im Internet: URL: http://www-lsi.ugr.es/%7Ebdf/Trabajos/lsi0006ugr.pdf [Stand 2004-04-01]

[BaGue01] Bauer, Andreas; Günzel, Holger (Hrsg.): Data Warehouse-Systeme - Architektur, Entwicklung, Anwendung, Heidelberg 2001, online im Internet: URL: http://www.data-warehouse-systeme.de/ [Stand 2004-05-12]. Im Internet finden sich Vorwort (S. v-vi; http://www.dpunkt.de/leseproben/3-932588-76-2/Vorwort.pdf), das erste Kapitel (S. 5-28; http://www.dpunkt.de/leseproben/3-932588-76-2/Kapitel%201.pdf) und das Glossar (S. 515-520; http://www.dpunkt.de/leseproben/3-932588-76-2/Glossar.pdf).

[Bl00] Blaschka, Markus: FIESTA: a framework for schema evolution in multidimensional databases (Diss.), München 2000, online im Internet: URL: Archivserver der Deutschen Bibliothek http://deposit.ddb.de/cgi-bin/dokserv?idn=962067342 [Stand 2004-04-01]

[BaCeNa92] Batini, C.; Ceri, S.; Navathe, S. B.: Conceptual Database Design: An Entity-Relationship Approach. Redwood City [u.a.] 1992

[BoeUl00] Böhnlein, M.; Ulbrich-vom Ende, A.: Grundlagen des Data Warehousing - Modellierung und Architektur, Bamberg 2000, online im Internet: URL: http://www.seda.wiai.uni-bamberg.de/ceus/publikationen/downloads/BoUl2000.pdf [Stand 2004-04-01]

[BoeUl01] Böhnlein, M.; Ulbrich-vom Ende, A.: Semantisches Data Warehouse Modell (SDWM), Bamberg 2001, online im Internet: URL: http://pda15.seda.sowi.uni-bamberg.de/CEUS/papers/SDWM_Rundbrief.pdf [Stand 2004-04-01]

[ChSt98] Chamoni, Peter; Stock, Steffen: Temporale Daten in Management Support Systemen, in: Wirtschaftsinformatik, 40, 1998, S. 513-519

[Ec99] Eckerson, Wayne: The Fallacy of Self-Service Decision Support, in: DM Review Magazine, Januar 1999, 1999, online im Internet, URL: http://www.dmreview.com/article_sub.cfm?articleId=228 [Stand 2004-04-29]

[Ez04] Ezeife, C. I.: Emerging non-traditional database systems (Data Warehousing and Mining), Windsor 2004, online im Internet: URL: http://davinci.newcs.uwindsor.ca/~cezeife/courses/60-539/04note.pdf [Stand 2004-04-01]

[GaGl97] Gabriel, Roland; Gluchowski, Peter: Semantische Modellierungstechniken für multidimensionale Datenstrukturen, in: HMD Theorie und Praxis der Wirtschaftsinformatik 195, 34, 1997, S. 18-37

[GaGl98] Gabriel, Roland; Gluchowski, Peter: Grafische Notationen für die semantische Modellierung multidimensionaler Datenstrukturen in Management Support Systemen , in: Wirtschaftsinformatik, 6, 1998, S.493-502

[GoRi99] Golfarelli, M.; Rizzi, S.: Designing the Data Warehouse: Key Steps and Crucial Issues, in: Journal of Computer Science and Information Management 1/1999, 2, 1999

[GrCoOl96] Graham, S.; Coburn, D.; Oleson, C.: The Foundations of Wisdom: A Study of the Financial Impact of Data Warehousing, International Data Corporation (IDC) Ltd. Canada 1996

[GuHaRaUl97] Gupta, H.; Harinaryan, V.; Rajaraman, A.; Ullman, J. D.: Index selection for OLAP, in Proceedings of the International Conference on Data Engineering (ICDE), 1997, S. 208-219

[GuMa00] Gutiérrez, Alejandro; Marotta, Adriana: An Overview of Data Warehouse Design Approaches and Techniques, Montevideo 2000, online im Internet: URL: http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0109.pdf [Stand 2004-04-01]

[HePrNi03] Hettler, Dora; Preuss, Peter; Niedereichholz, Joachim: Vergleich ausgewählter Ansätze zur semantischen Modellierung von Data-Warehouse-Systemen, in: Heilmann, Heidi; Strahringer, Susanne (Hrsg.): Neue Konzepte in der Software-Entwicklung, HMD 231, 40, 2003, S. 97-107

[Ho99] Holthuis, Jan: Der Aufbau von Data Warehouse-Systemen: Konzeption Datenmodellierung Vorgehen; 2. Aufl., Wiesbaden 1999

[HueLeVo00] Hüsemann, B.; Lechtenbörger, J.; Vossen, G.: Conceptual Data Warehouse Design, in: Proceedings of the International Workshop on Design and Management of Data Warehouses, (DMDW'2000) Stockholm, Sweden, June 5-6, 2000, Stockholm 2000, online im Internet: URL: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/paper6.pdf [Stand 2004-04-01]

[In96] Inmon, W.: Building the Data Warehouse; 2. Aufl., New York [u.a.] 1996

[Jue99] Marcus Jürgens: Index structures for data warehouses (Diss.) Berlin 2000, online im Internet: URL: Archivserver der Deutschen Bibliothek http://deposit.ddb.de/cgi-bin/dokserv?idn=961845872 [Stand 2004-04-01]

[Lo94] Love, B.: Enterprise Information Technologies, Van Nostrum Reinhold, New York 1994

[LoRa90] Lockemann, P. C.; Rademacher, K.: Konzepte, Methoden und Modelle zur Datenmodellierung, in: HMD Theorie und Praxis der Wirtschaftsinformatik 152, 27, 1990, März 1990, S. 3-16.

[Ma00] Maumenee, Roger: Entwurf und Implementierung von Abbildungen für unterschiedliche Data-Warehouse-Speicherungsschemata in SIRIUS, Zürich 2000, online im Internet: URL: `http://www.ifi.unizh.ch/ifiadmin/staff/rofrei/DA/DA_Arbeiten_2000/Maumenee_Roger.pdf` [Stand 2004-04-01]

[MoKo00] Moody, Daniel; Kortink, Mark: From enterprise models to dimensional models: a methodology for data warehouse and data mart design, in: Proceedings of the International Workshop on Design and Management of Data Warehouses, (DMDW'2000) Stockholm, Sweden, June 5-6, 2000, Stockholm 2000, online im Internet: URL:`http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/paper5.pdf` [Stand 2004-04-01]

[MoKo99] Moody, Daniel; Kortink, Mark: From Entities to Stars, Snowflakes, Clusters, Constellations and Galaxies: A Methodology for Data Warehouse Design, in: 18th International Conference on Conceptual Modeling, Industrial Track Proceedings, ER' 99.

[MuBe98] Mucksch, Harry; Behme, Wolfgang: Das Data Warehouse-Konzept - Architektur, Datenmodelle, Anwendungen, 3. Aufl., Wiesbaden 1998

[Qu03] Quix, Christof Josef: Metadatenverwaltung zur qualitätsorientierten Informationslogistik in Data-Warehouse-Systemen (Diss.), Aachen 2003, online im Internet: URL: Archivserver der Deutschen Bibliothek `http://deposit.ddb.de/cgi-bin/dokserv?idn=969979800` [Stand 2004-04-01]

[Ra02] Ramsak, Frank: Towards a general purpose, multidimensional index : integration, optimization, and enhancement of UB-trees (Diss.), München 2002, online im Internet: URL: Archivserver der Deutschen Bibliothek `http://deposit.ddb.de/cgi-bin/dokserv?idn=965205096` [Stand 2004-04-01]

[Ra95] Raden, N.: Star Schema 101, o.O. o.J., online im Internet: URL: `http://members.aol.com/nraden.str.htm` [Stand nicht mehr aktuell]

[Sa99] Sapia, C.: On modeling and predicting user behaviour in OLAP systems, in: Proceedings of the International Workshop on Design and Management of Data Warehouses, (DMDW'1999) Heidelberg, Germany, June 14-15,1999, Heidelberg 1999, online im Internet: URL: `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/paper2.pdf` [Stand 2004-05-13]

[Schw03] Schwarz, Holger: Integration von Data Mining und Online Analytical Processing: Eine Analyse von Datenschemata, Systemarchitekturen und Optimierungsstrategien (Diss.), Stuttgart 2003, online im Internet: URL:Archivserver der Deutschen Bibliothek `http://deposit.ddb.de/cgi-bin/dokserv?idn=968816657` [Stand 2004-04-01]

[TheSe99] Theodoratos, D., Sellis, T.: Designing Data Warehouses, DWQ project, o.O. 1999

[vM00] von Maur, Eitel: Object Warehouse : Konzeption der Basis objektorientierter Management Support Systems am Beispiel von Smalltalk und dem ERP Baan (Diss.), Osnabrück 2000, online im Internet: URL: Archivserver der Deutschen Bibliothek http://deposit.ddb.de/cgi-bin/dokserv?idn=961697555 [Stand 2004-04-01]

[Wu97] Wu, M.; Buchmann, A. P.: Research Issues in Data Warehousing, in: K. R. Dittrich, A. Geppert (Hrsg.): Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Ulm , 5.-7. März 1997, Informatik Aktuell, Berlin [u.a.] 1997

[Zi01] Zicari, Roberto: Grundlagen der Datenbanksysteme I; Vorlesungsskript aus dem Sommersemester 2001, Johann Wolfgang Goethe-Universität, Frankfurt am Main 2001, online im Internet: URL: http://www.dbis.informatik.uni-frankfurt.de/TEACHING/DB-Vorlesung/2001_SS/Einfuehrung_DBI.pdf [Stand 2004-04-30]