Technische Universität Wien

Institut für Softwaretechnik
Abteilung: Information & Software Engineering

Ein Metadaten-basiertes Sicherheitsmodell für OLAP-Datenbanken

Diplomarbeit zur Erlangung des akademischen Grades eines Diplom Ingenieurs

eingereicht bei o. Univ.-Prof. Dipl.-Ing. Dr. A Min Tjoa

Nevena Katic

Wien, 21.11.1997

Kurzfassung der Diplomarbeit

Datenbankflexibilität und Möglichkeiten befinden sich ständig in einem Evolutionsprozeß. Datenstrukturen haben sich von „*flat file*“ und hierarchischen Dateien zu relationalen und verteilten relationalen Datenbanken entwickelt und sind der menschlichen Art des Denkens angenähert.

Die neueste Entwicklung auf diesem Gebiet sind multidimensionale Datenbanken. Sie stellen eine ideale Grundlage für Analyseprozesse dar, die weltweit unter dem Namen OLAP (On Line Analytical Processing) bekannt sind. Moderne Unternehmen brauchen umfassende Datenbankanwendungen, die für Analyseprozesse optimiert sind. Multidimensionale Datenbanken sind dafür sehr gut geeignet, weil sie einen schnellen und flexiblen Datenzugriff und Datenanalyse sehr großer und sehr komplexer Datenmengen ermöglichen. Der schnelle, unternehmensweite Zugriff auf strategisch relevante Informationen ist von existentieller Bedeutung für jedes Entscheidungssystem.

Die Sicherheit von Datenbanksystemen, insbesondere von Data Warehouses, ist das zentrale Thema dieser Diplomarbeit. In dem theoretischen Teil dieser Diplomarbeit werden verschiedene Sicherheitsmechanismen untersucht. Von dieser Überlegungen ausgehend, wurde ein Mechanismus (Metadaten-Sicherheitsmodell) als besonders geeignet für das WWW-EIS-DWH Projekt befunden und in dem praktischen Teil der Arbeit implementiert.

Der Aufbau dieser Diplomarbeit sieht folgendermaßen aus: Zuerst werden die grundlegenden Begriffe für jedes Data Warehouse eingeführt: multidimensionale Datenbanken und *On-Line Analytical Processing* (OLAP). In erstem Kapitel wird erklärt, was multidimensionale Datenbanken sind, wie sie aus einfachen, relationalen Datenbanken gewonnen werden, welche Eigenschaften sie haben, und wann sie eingesetzt werden sollen. Besonderer Wert wird auf die Sicherheit von multidimensional modellierten Systemen gelegt. In diesem Sinn wird das „Rollenkonzept“ vorgestellt. Anschließend wird eine multidimensionale Datenbank anhand eines Beispiels dargestellt. In zweitem Kapitel werden die OLAP-Charakteristiken und die OLAP-Architektur vorgestellt. Die drei momentan wichtigsten OLAP-Produkte werden beschrieben und ein Vergleich dieser Produkte wird durchgeführt. Kapitel 3 beschäftigt sich mit dem Hauptthema dieser Diplomarbeit, nämlich mit Sicherheit. Zuerst werden die allgemeinen Sicherheitsanforderungen und Sicherheitsbedrohungen beschrieben. Verbunden damit wird der Aufbau eines Sicherheitsplans vorgeschlagen. Besonderer Wert wird aber auf die Datensicherheit (Sicherheit von Datenbanken bzw. Data Warehouse) gelegt. Ausgehend von den wichtigsten, existierenden Mechanismen der Zugriffskontrolle, werden zwei neue Sicherheitsmodelle für das Data Warehousing vorgestellt. Diese Modelle werden jeweils durch Beispiele aus der Praxis unterstützt.

Da auf ein Data Warehouse gewöhnlich vom Unternehmensintranet zugegriffen wird, beschäftigt sich diese Diplomarbeit abschließend mit der Internet- und Intranetsicherheit.

INHALTSVERZEICHNIS

Abbildungsverzeichnis

Tabellenverzeichnis

1 Multidimensionale Datenbanken (MDDB)

Viele Manager denken über Daten, mit denen sie arbeiten, bereits multidimensional: z.B. wie viele Produkte einer bestimmten Art in einem bestimmten Zeitraum, in einer Region, verkauft wurden. Für solche Zwecke bieten die multidimensionale Datenbanken eine benutzerfreundliche Art der Datendarstellung.

Multidimensionale Datenstrukturen sind nichts Neues. Wenn man an eine Tabelle denkt, die z.B. den monatlichen Verkauf von verschiedenen Produkten in verschiedener Geschäften darstellt (in Zeilen Produkte und in Spalten Geschäfte), denkt man schon an eine zweidimensionale Datenstruktur. Wenn man sich nun mehrere solche Tabellen (für jedes Monat eine) auf einem Stapel vorstellt, hat man schon eine 3-dimensionale Datenstruktur vor Augen.

1.1 Was ist eine multidimensionale Datenbank?

Eine multidimensionale Datenbank ist ein Softwaresystem, das dazu bestimmt ist, große Datenmengen effizient und brauchbar zu speichern und aufzurufen. Diese Datenmengen sind eng verwandt und können von verschiedenen Perspektiven aus betrachtet und analysiert werden.[KEN93]

Um die Antworten auf wichtige Managementfragen herauszufinden, müssen Manager die Daten oft von verschiedenen Perspektiven betrachten. Für eine Analyse der Verkaufszahlen wären z.B. folgende Perspektiven nützlich :

- Verkaufszahlen im Bezug auf Produkt
- Verkaufszahlen im Bezug auf Zeit
- Verkaufszahlen im Bezug auf Ort

Wenn man die Informationen, die auf diese Weise erhalten werden, kombiniert , kann man daraus Trends erkennen, neue wichtige Entscheidungen treffen usw. All diese Abfragen könnten natürlich auch aus klassischen relationalen Datenbanken erhalten werden, nur dauert dies viel länger. Da gerade die Zeit bei Entscheidungsprozessen eine sehr wichtige Rolle spielt, werden die multidimensionale Datenbanken von immer mehr Managern bevorzugt.

1.2 Von einfachen zu multidimensionalen Datenbanken

Die Einträge in einer relationalen Datenbank werden „**Rekords**" genannt [KEN93]. Wir betrachten jetzt eine einfache relationale Tabelle (Tabelle 1). In diesem Beispiel

gibt es nur drei Produkte, die in nur drei Orten verkauft werden. Die Tabelle hat aber neun Einträge (Rekords) und wirkt ziemlich unübersichtlich.

Produkt	Ort	Verkauf (öS)
Computer	Wien	50000
Computer	Linz	43000
Computer	Graz	72000
Maus	Wien	8500
Maus	Linz	5000
Maus	Graz	7600
Tastatur	Wien	19000
Tastatur	Linz	9900
Tastatur	Graz	10650

Tabelle 1 : Verkaufszahlen im Bezug auf Produkte und Orte

Eine Möglichkeit, Daten etwas übersichtlicher darzustellen wäre, ein zweidimensionales Array zu benutzen (Abbildung 1).

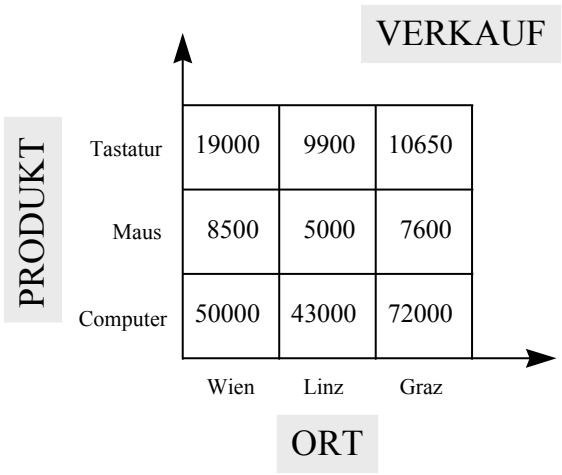

Abbildung 1: Zweidimensionales Array

Jede Achse des Arrays wird **Dimension** genannt. In diesem Beispiel heißen daher die Dimensionen: PRODUKT und ORT. Die Elemente einer Dimension nennt man **Positionen**. Wie der Abbildung 1 zu entnehmen ist, sind die Positionen der ersten Dimension: Computer, Maus und Tastatur, und die der zweiten Dimension : Wien, Linz und Graz. Die Felder, wo sich die Positionen schneiden, heißen **Zellen** und beinhalten die Meßwerte. Die Meßwerte sind meistens numerischen Werte und bilden ein ***Fact*** (VERKAUF).

Darstellung und Verwaltung der Daten in einem zweidimensionalen Array ist viel vorteilhafter als in einer relationalen Tabelle. Erstmal können viele Eigenschaften der präsentierten Daten sofort abgelesen werden (in unserem Beispiel ist es offen-

sichtlich, daß es um zwei Dimensionen mit je drei Positionen geht), außerdem werden bei dieser Art der Darstellung ähnliche Daten in Spalten und Zeilen zusammengefaßt. Bei so gruppierten Daten ist es dann einfach, Vergleiche, Summierungen und verschiedene statistische Berechnungen durchzuführen. Daher stellen die zweidimensionale Arrays eine höhere Stufe der Datenorganisation dar.

Performanzvorteile die diese Art der Datendarstellung mit sich bringt, sind nicht zu unterschätzten. So muß z.B. bei einer SELECT-Abfrage die relationale Tabelle sequentiell durchsucht werden, während bei einem multidimensionalen Array, wo die Daten vororganisiert sind, die Suche erwartungsgemäß schneller abläuft.
Bis jetzt haben wir nur zweidimensionale Arrays betrachtet, aber eine weitere Dimension kann einfach hinzugefügt werden. Wenn wir z.B. wissen möchten, wie der Verkauf im Bezug auf die Zeit abläuft, braucht nur ein neues Feld („Zeit“) in die relationale Datenbank eingefügt zu werden (Tabelle 2).

Produkt	Ort	Zeit	Verkauf (öS)
Computer	Wien	1995	50000
Computer	Wien	1996	65000
Computer	Wien	1997	100000
Computer	Linz	1995	43000
Computer	Linz	1996	50000
Computer	Linz	1997	89000
Computer	Graz	1995	72000
Computer	Graz	1996	88000
Computer	Graz	1997	93200
Maus	Wien	1995	8500
Maus	Wien	1996	10600
Maus	Wien	1997	20430
Maus	Linz	1995	5000
Maus	Linz	1996	6000
Maus	Linz	1997	7000
Maus	Graz	1995	7600
Maus	Graz	1996	8500
Maus	Graz	1997	9200
Tastatur	Wien	1995	19000
Tastatur	Wien	1996	25000
Tastatur	Wien	1997	28000
Tastatur	Linz	1995	9900
Tastatur	Linz	1996	13000
Tastatur	Linz	1997	15000
Tastatur	Graz	1995	10650
Tastatur	Graz	1996	12000
Tastatur	Graz	1997	14000

Tabelle 2: Verkaufszahlen im Bezug auf Produkte, Orte und Zeit

Die Tabelle sieht jetzt noch unübersichtlicher aus, die Arten der Daten sind schwer zu erkennen. Obwohl hier nur ein zusätzliches Feld mit nur drei verschiedenen Einträgen (1995, 1996, 1997) eingefügt wurde, ist es sehr schwer, die Zusammenhänge ausfindig zu machen.

Die Datendarstellung kann mittels eines dreidimensionalen Arrays vereinfacht werden. Mit der dritten Dimension (ZEIT), bilden die Daten ein Würfel (Abbildung 2).

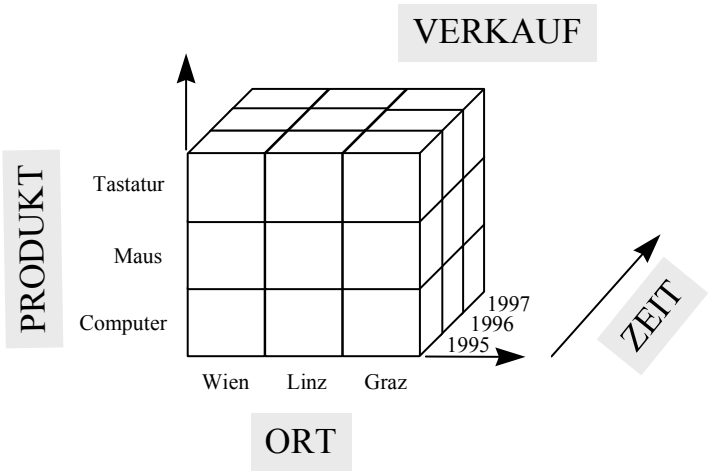

Abbildung 2: Dreidimensionales Array

Hinzufügen von weiteren Dimensionen ist natürlich auch möglich, nur wird die graphische Darstellung von solchen Systemen um einiges komplizierter.

1.3 Charakteristiken einer multidimensionalen Datenbank

1.3.1 Rotation

Betrachten wir eine einfache relationale Tabelle (Tabelle 3).

Diese Tabelle eignet sich gut für eine multidimensionale Datenbank, weil die Daten, die sie beinhaltet, von ihrer Natur her multidimensional sind.

Die Daten können in einem 2-dimensionalen Array übersichtlich dargestellt werden (Abbildung 3). Das *View*, das sich hier bietet, ist „VERKÄUFER“ im Bezug auf „PRODUKT“ und „ORT“. Der Benutzer kann jedoch auch andere *Views* benötigen, wie z.B. „VERKÄUFER“ im Bezug auf „ORT“ UND „PRODUKT“ oder „PRODUKT“ im Bezug auf „VERKÄUFER“ und „ORT“, usw.

Produkt	Ort	Verkäufer
Tastatur	Wien	Mayer
Tastatur	Linz	Schmidt
Tastatur	Graz	Bayer
Computer	Wien	Mayer
Computer	Linz	Bayer
Computer	Graz	Schmidt
Software	Wien	Bayer
Software	Linz	Mayer
Software	Graz	Schmidt

Tabelle 3 : Verkäuferzuordnung

In relationaler Umgebung würde jedes neue *View* ein neues *Query* oder eine neue Sortierung veranlassen. In multidimensionaler Umgebung wird ein neues *View* durch eine „**Rotation**“ aus dem alten *View* gewonnen. In unserem Beispiel (Abbildung 3) wurde das erste *View* um 90° in negativer Richtung rotiert, ohne das die Daten reorganisiert werden mußten. Die Einfachheit und die Schnelligkeit mit der die Rotation durchgeführt werden kann, sind weitere Vorteile der multidimensionalen Arrays im Vergleich mit den relationalen Datenbanken.

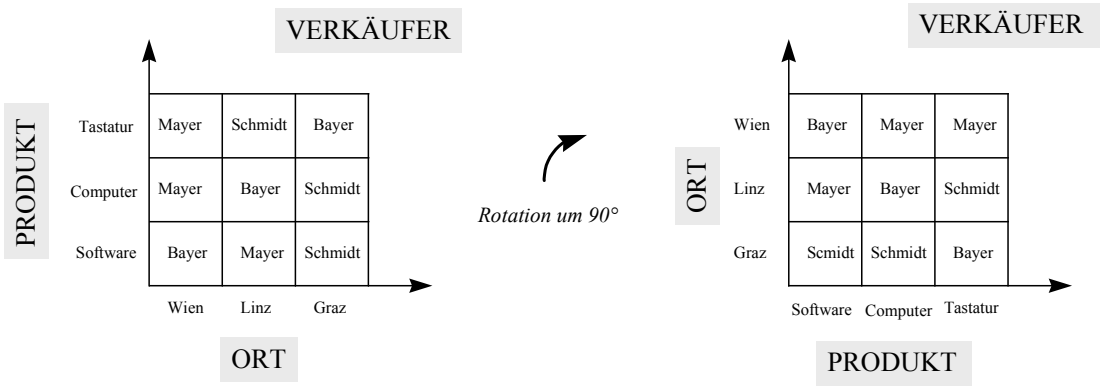

Abbildung 3 : Rotation von einem *View* um 90°, negativer Richtung

Die Anzahl der *Views* wächst exponentiell mit der Anzahl der Dimensionen. Ein zweidimensionales Array hat 2 *Views*, ein dreidimensionales 6 usw. Ein n-dim. Array hat n! *Views*.

1.3.2 *Ranging*[1]

Eine multidimensionale Datenbank erlaubt dem Endbenutzer die Daten zu partitionieren, um das gewünschte *View* zu bekommen. Betrachten wir eine 3-dim. Datenbank (Abbildung 4), die Verkaufszahlen im Bezug auf Produkt, Ort und Zeit erfaßt. Nehmen wir an, daß der Benutzer wünscht, die Verkaufszahlen für die Produkte Tastatur und Maus in Wien und Graz in den Jahren 1995 und 1996 zu sehen. ***Ranging*** ermöglicht es, die gesuchten Positionen für die jeweilige Dimension auszuwählen und damit ein reduziertes Array zu definieren. Dieses 3-dim. Array kann weiter beliebig rotiert oder bearbeitet werden, genauso wie das davor.

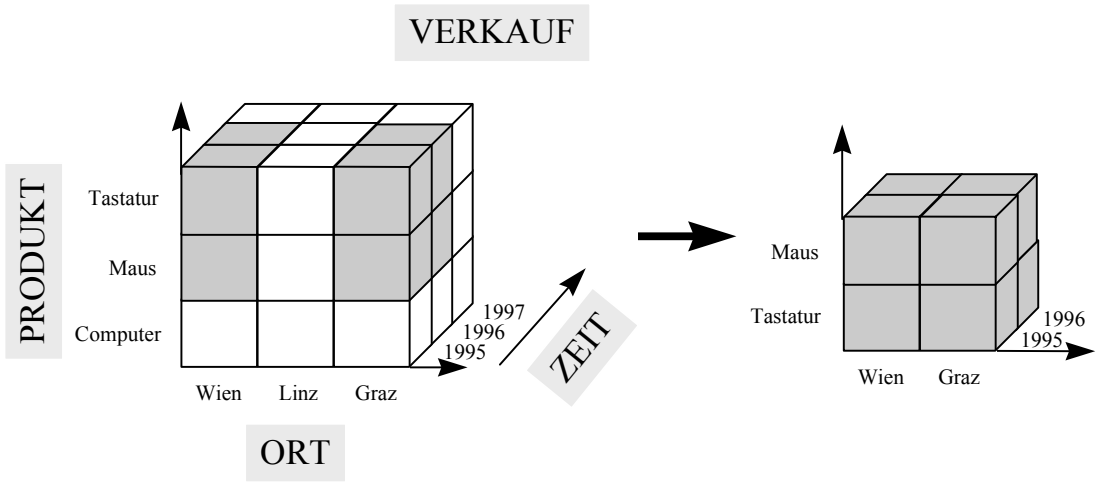

Abbildung 4: *Ranging* (*dicing*) von einem Datenwürfel

Der Vorteil, den man durch das *Ranging* erzielt, ist der, daß man mit kleineren Datenmengen schneller arbeiten kann.
Ranging wird auch „***dicing***“[2] genannt, weil der Datenumfang verringert wird.

1.3.3 Hierarchien, „*Roll-up*“ und „*Drill-down*“

In obigem Beispiel (Abbildung 4) haben wir drei Dimensionen in Betracht gezogen: PRODUKT, ORT und ZEIT. Eigentlich haben wir hier in einer multidimensionalen Datenbank die Verkaufszahlen von verschiedenen Produkten in verschiedenen Städten, in Bezug auf zeitliche Angaben gespeichert. Was passiert aber, wenn der Benutzer am Speichern von Verkaufsdaten in verschiedenen Regionen oder Ländern interessiert ist?

[1] Range = Kollektion
[2] dice = in Würfel schneiden

Wenn man Verkaufsdaten auf regionalem Niveau erfassen will, könnte man dafür eine neue Dimension („REGION“) hinzufügen. Falls man auch am Länderverkauf Interesse hat, würde das noch eine zusätzliche Dimension („LAND“) verlangen. Das Einfügen von neuen Dimensionen für jedes neue Niveau von schon existierenden Daten wäre jedoch zu unübersichtlich und zu umfangreich. Eine viel effizientere Lösung ist es, die verschiedenen Hierarchiestufen von einem Datenfeld in nur einer Dimension aufzubewahren, wobei man immer darauf achten muß, daß die Hierarchiestufen deutlich voneinander zu unterscheiden sind.

Wir wollen in unserem Beispiel neben den Orten (Wien, Graz, Linz) auch die Regionen (Wien, Niederösterreich, Oberösterreich) und das Land (Österreich) hinzufügen. Um das Ganze sinnvoll zu gestalten, verändern wir den Namen der Dimension von „Ort“ auf „Geographie“. Eine Darstellung von einer auf diese Weise organisierten 3-dim. Datenbank wird in der Abbildung 5 dargestellt.

Bei hierarchischen Dimensionen muß man besonders darauf achten, daß die Daten bei verschiedenen mathematischen Operationen nicht mehrmals verwendet werden. Will man, beispielsweise, eine Summe von Einnahmen für „Computer“ im Jahr „1995“ berechnen, wäre es falsch, durch die Dimension „GEOGRAPHIE“ durchzugehen und alle Verkaufszahlen zusammen zu addieren, weil die Verkaufsdaten von z.B. Linz sowohl bei „Linz“ als auch bei „OÖ“ und schließlich auch bei „Österreich“ miteingerechnet wären.

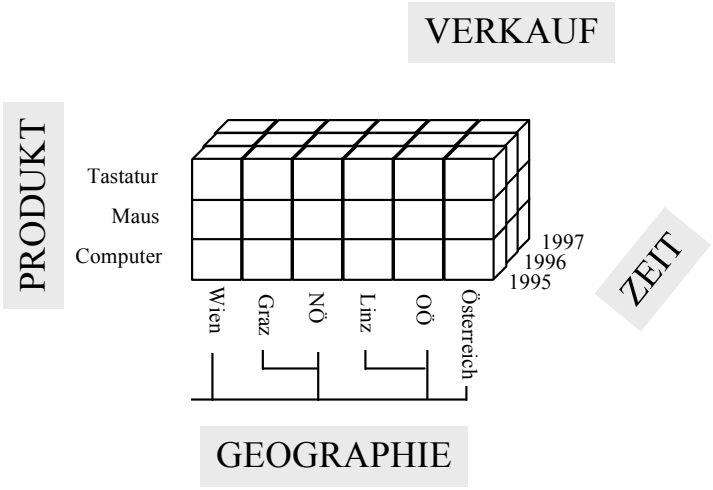

Abbildung 5: Hierarchische Struktur der Dimensionen in einem Datenwürfel

Die höheren Hierarchiestufen dienen eigentlich dafür, aggregierte Daten für spätere Abfragen aufzubewahren. Wenn wir die Verkaufsdaten für „Computer“ im Jahr „1995“ für das ganze Land erfahren möchten, brauchen wir einfach die Daten die in der Zelle (PRODUKT = „Computer“, ZEIT = „1995“ und GEOGRAPHIE = „Österreich“) stehen, abzulesen. (Hier wurden die Hierarchiestufen der Dimension „GEOGRAPHIE“ tiefer analysiert. Ähnlich wäre es bei den anderen Dimensionen. Die Hierarchiestufen der Dimension „Zeit“ wären z.B.: Tag, Monat, Quartal, Jahr, der Dimension „Produkt“: Produkt, Produktgruppe usw.)

Man kann durch die verschiedenen Hierarchiestufen navigieren. Das Navigieren von niedrigeren zu höheren Hierarchiestufen nennt man „***roll-up***“ und in umgekehrter Richtung „***drill-down***“.

1.4 Performanz-Vorteile

Um die Performanzeigenschaften von relationalen mit den von multidimensionalen Strukturen zu vergleichen, betrachten wir ein 10 × 10 × 10 Array. Angenommen, uns interessiert der Verkauf vom Produkt Tastatur in Wien im Jahr 1996. (PRODUKT=„Tastatur“, ORT = „Wien“, ZEIT = „1996“, VERKAUF = ?).
In einem relationalen System müßten alle 1000 Einträge durchgesucht werden, um den gesuchten Wert zu finden. Da die Daten in einem multidimensionalen Array in verschiedenen Dimensionen organisiert sind, ist die Suche hier viel einfacher. Man muß einfach durch die Dimensionen gehen und die gewünschten Einträge finden, was in unserem Beispiel jeweils eine 10-Positionen Suche bedeutet. Das heißt, wir haben maximal 1000 Vergleiche im relationalen System, während es im multidimensionalen nur 30 sind. Da wir annehmen können, daß eine durchschnittliche Suchzeit eine Hälfte der maximalen Suchzeit darstellt, haben wir 500 versus 15 Vergleiche. Daraus läßt sich ausrechnen, daß die multidimensionale Datenstrukturen eine 3300 % Verbesserung von Performanz zur Folge haben.

1.5 Wofür sind MDDB nicht geeignet ?

Multidimensionale Datenbanken eignen sich für multidimensionale Daten, bzw. Datensets, die aufeinander bezogen sind. Betrachten wir nun eine relationale Datenbank, die auf keinen Fall als multidimensionale Datenbank gespeichert werden sollte (Tabelle 4).

Man kann diese Werte als ein 2-dim. Array organisieren. „Produkt“ und „Produktnummer“ könnten Dimensionen darstellen, der Hersteller wäre als Zellenwert gespeichert. Da alle Produkte und ihre Nummer eindeutig sind, würde es 7 Positionen an jeder Dimension geben, was weiter 49 Zellen ergibt. Jedoch wären nur 7 von diesen 49 Zellen nicht leer (Abbildung 6). (Wenn man diese Situation mit dem ersten Beispiel (Tabelle 1, Abbildung 1) vergleicht, merkt man, daß es dort nur drei verschiedene Werte für jedes Feld gab, und daß alle neun Zellen mit einem Meßwert versehen waren.)

Produkt	Produktnummer	Hersteller
Tastatur	234325	Deutschland

Maus	567577	USA
Computer	4256245	Japan
Handbuch	1551234	UK
Software	3456366	Kanada
Diskette	3645002	China
Drucker	4240200	Italien

Tabelle 4: Hersteller geordnet nach Produkten und Produktnummern

In diesem Beispiel entspricht jedem Produkt nur eine Produktnummer. Deswegen sind nur 7 von 49 Zellen mit Werten (Hersteller) ausgefüllt. Man sagt in solchen Fällen, daß das Array spärlich (***spars***) ausgefüllt ist. Aus Performanzgründen ist es ratsam, solche Datensets nicht als multidimensionale Arrays zu speichern. Je größer ein multidimensionales Datenset wird, desto besser ist es zur Speicherung als multidimensionales Array geeignet. Analog: je größer ein nicht multidimensionales Datenset wird, desto weniger geeignet ist es zur Speicherung als multidimensionales Array.

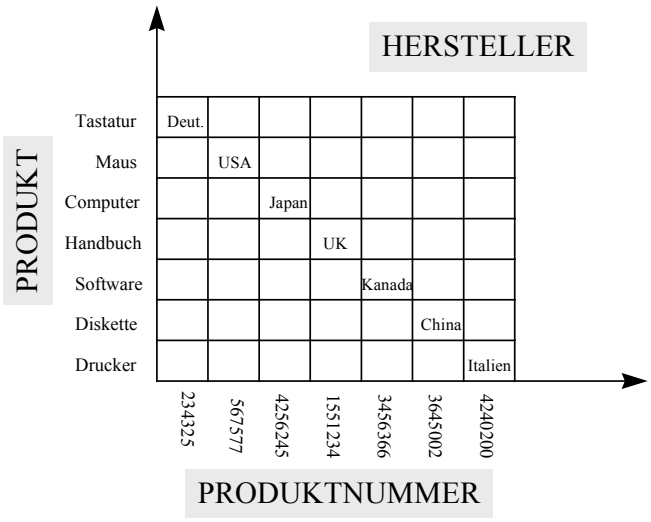

Abbildung 6: Ein spärlich (*spars*) ausgefülltes Array

Multidimensionale Arrays dienen dazu, die schnelle und effiziente Analyse von gespeicherten Daten zu ermöglichen. In unserem „Produkt-Produktnummer - Hersteller“-Beispiel kann man sich sehr schwer vorstellen, daß eine Analyse von vorhandenen Daten gebraucht und relevant werden könnte. Da es hier sehr wenige sinnvolle Relationen zwischen Daten in Datensets gibt, ist diese Datenbank sehr schlecht für die multidimensionale Datenbanktechnologie geeignet.

Zum Unterschied dazu besitzen die Daten aus dem ersten Beispiel (Tabelle 1) viele sinnvolle Relationen. Aus diesen Daten könnte man verschieden Trends ablesen,

z.B. : Wie die Produkte im Bezug auf Orte verkauft werden, welche Produkte am besten verkauft werden, wie viele Produkte insgesamt verkauft werden usw. Die Charakteristik von dieser Datenbank ist, daß die Relationen zwischen Daten wichtiger sind als die Daten selbst. Datenbanken mit dieser Eigenschaft eignen sich sehr gut für multidimensionale Datenaufbewahrung.

1.6 Warum braucht man „Multidimensionale Modellierung“ ?

In den relationalen Systemen liegen meistens die Informationen nicht in der gewünschten Form vor oder sie sind nicht eindeutig definiert, um auf komplexen Fragen effizient eine Antwort zu liefern. Deswegen wird sehr viel Kreativität verlangt um halbwegs gute Ergebnisse zu erzielen. Der Ausweg aus solchen Problemen ist die Abkehr von unstrukturierten Datenmengen hin zu geordneten, multidimensionalmodellierten Datensystemen.

Der Vorteil der „multidimensionalen Modellierung“ (MDM) [RAD96] ergibt sich daraus, daß es sowohl für Summierungen und Anordnen von Daten als auch für komplexe Datenanalysen geeignet ist. Multidimensionale Modelle arbeiten mit numerischen Daten, wie z.B. Werte, Anzahl, Gewichte, Geldbeträge und Ähnliches. Die typische MDM Angabe lautet : „Welcher Profit wird bei einem gewissen Kunden, in einem gewissen Zeitraum, für eine gewisse Organisation erzielt?“ .

Multidimensionales Modellieren ist sehr einfach. Es ist nicht nur für erfahrene Fachleute gedacht, sondern auch für Manager und andere „business“- Leute, die mit den modellierten Daten arbeiten . Sie alle können die Modelle verstehen, weil die Daten hier auf eine für Menschen ganz natürliche Art ausgegeben werden.

1.7 „Multidimensional Model“ versus „Relational Model“

Die Hauptidee, die hinter einem relationalen Modell steht, ist die, daß ein solches System in der Lage ist, Transaktionen zu unterstützen. Im Gegensatz dazu, ist ein multidimensionales Model für die Unterstützung von Berichten und verschiedenen analytischen Bedürfnissen bestimmt.

Die Hauptunterschiede zwischen diesen beiden Modellen sind [RAD96]:

- *MDM betrachtet die Informationen aus der Perspektive eines Zeitintervalls („slice of time“)*

In einem relational modelliertem System werden immer einzelnen Transaktionen durchgeführt. Bei den multidimensionalen Datenmodellen geht es nicht um tatsächliche Ereignisse, sondern um ihre quantitativen Ergebnisse in einem bestimmten Zeitintervall (wie z.B. in einem Tag, in einer Woche usw.).

- *Lokale vs. Globale Konsistenz*

Ein multidimensionales Modell ist für eine ganze Organisation übereinstimmend. Dagegen muß ein relationales Modell nur intern (innerhalb seiner Reichweite) konsistent sein. Wenn z.B. verschiedene Transaktionen in verschiedenen Abteilungen einer Firma jeweils durch ein relationales Modell unterstützt werden, sollen die Daten im Rahmen des jeweiligen relationalen Modells übereinstimmend sein. Falls die Data Warehouse der gesamten Firma ihre Daten aus mehreren Abteilungsystemen kombiniert, wird die Datenübereinstimmung nicht automatisch erfolgen.
Ein multidimensionales Modell geht immer von einer global übereinstimmenden Sicht der Firma aus.

- *Einzelne Vorgänge vs. „Big Picture“*

Relationale Systeme sind für das detaillierte Verfolgen von einzelnen Transaktionen sehr gut geeignet. Wenn man z.B. seinen Kontoauszug betrachtet, will man alle Überweisungen zurückverfolgen können. Will man aber verschiedene analytische Fragen beantwortet haben, gerät man bei solchen Systemen in Schwierigkeiten. Die multidimensionalen Modelle sind für analytische Fragen von derArt: „Werde ich bei diesem Vertrag Gewinn erzielen?“ oder „Wer sind meine beste Kunden und warum?“ sehr gut geeignet.

- *Explizite vs. Implizite Relationen*

Bei einem relationalem Modell werden die expliziten Relationen zwischen einzelnen Daten schon beim Design der Datenbank festgelegt. Bei den multidimensionalmodellierten Systemen werden diese Relationen durch das Existieren von „Facts“ impliziert. Das heißt, wenn bei einem Fact die Verkaufsdaten für den Kunden X und das Produkt A vorhanden sind, ist die Relation zwischen dem Kunden und dem Produkt impliziert.

1.8 Die wichtigsten Punkte beim Aufbau eines Multidimensionalen Modells

Die Aufgabe des Aufbaus eines multidimensionalen Datenmodells kann in 4 Punkten zusammengefaßt werden [AGS97]:

- Ermittlung des Informationsbedarfs
- Informationsbeschaffung und Informationsaufbereitung
- Informationshaltung
- Informationspräsentation

Ermittlung des Informationsbedarfs

Dieser Punkt wird leider oft übergangen, weil man meint, daß die grundlegenden Dinge im eigenen Unternehmen selbstverständlich gegeben sind.

Informationsbeschaffung und Informationsaufbereitung

Ein sehr wichtiger Punkt beim Design eines multidimensionalen Modells ist, den Geschäftsbereich zu erforschen: wöchentliche Verkaufsberichte, monatliche Gewinn- und Verlustrechnungen usw. und zu modellieren. Dieser Vorgang kann in folgenden Schritten zusammengefaßt werden :

1. Welcher Geschäftsprozeß wird modelliert ?
2. Welche Werte werden gemessen („Facts“) ?
3. Auf welchem Detaillierungsgrad wird die Analyse der Meßwerte durchgeführt?
4. Was haben die Meßwerte gemeinsam ?
5. Welche Attribute haben die Dimensionen?
6. Sind die Attribute stabil oder veränderlich im Laufe der Zeit, und ist ihre Kardinalität begrenzt oder unbegrenzt?

(Wenn ein mehrdimensionales Modell in einer relationalen Datenbank gebaut wird, wird unbegrenzte Kardinalität erwartet. Das Gegenteil der Fall in den mehrdimensionalen Datenbanken, wo Änderungen der Kardinalität oft vollständige Umorganisation von Datenbank verlangen, was ein extrem zeitaufwendiger Prozeß ist.)

Wenn ein unternehmensweites multidimensionales Datenmodell existiert, wird die Arbeit erheblich erleichtert und beschleunigt. Um so ein Model zu entwerfen, muß man nicht nur nach den richtigen Datenquellen suchen, sondern auch erforschen, wie die Daten beschafft und aktualisiert werden. (Es kann sich hier um etwas einfaches, wie z.B. Kopieren von Datenbeständen, handeln oder aber auch um kompliziertere Vorgänge, wie z.B. das Sammeln von Änderungsdaten aus OLTP-Anwendungen.)
Ein wichtiger Begriff in diesem Zusammenhang sind die **Metadaten.**

Sie beschreiben die Art und Herkunft, ggf. die Ableitung und Aggregierung sowie Aktualisierung der Datenbestände in einem multidimensional modelliertem System (Data Warehouse). Mit Hilfe der Metadaten kann die Konsistenzproblematik in einem Data Warehouse weitgehend gelöst werden.

Wenn man ein Modell baut, soll man am Anfang vermeiden ‚sofort alles aufzunehmen. Man soll sich statt dessen auf einen bestimmten Bereich, wie Kundenrentabilität oder Verkaufsberichte konzentrieren. Eine Ausführung, die sich als nützlich erwiesen hat, ist, daß die Kunden eine einfache Anweisung formulieren, die das Vorhaben des Modells beschreibt. Z. B. "Nettoverkäufe, in $ und Einheiten, von jedem Produkt, in jedem Geschäft, per Woche, per Firma für die letzten drei Jahre, verglichen mit dem Versand“.
Diese einfache Anweisung gibt uns den Geschäftsprozeß (Einzelhandel), „Facts“ (Umsatz, Anzahl von verkauften Produkten, Versandeinheiten, Preis), Dimensionen (Kunde, Produkt und Zeit) und die niedrigste Granularitätsstufe (Woche, Geschäft, usw.). Die Attribute werden ausgearbeitet, wenn man die umfangreichen Merkmale der Dimensionen auswertet.

Informationshaltung

MDM sieht vor, daß es im Modellzwei (auch physisch getrennte) Datenbestände gibt: einen operativen Datenbestand und einen eigenen Auswertungsbestand. Die Frage, die sich hier stellt, ist die, welcher Art die Datenmodelle sein sollen, die die separaten Bestände beschreiben, und welche Datenbanksysteme beim Umgehen mit den so modellierten Daten eine gute Performanz erzielen können. Eine gute Lösung, die alle diese Anforderungen befriedigen kann, ist das Data Warehouse.

Informationspräsentation

Ein multidimensionales Datensystem wird oft mit den Produkten für seine Visualisierung verwechselt. Ein Grund dafür ist sicherlich die Tatsache, daß das Datensystem auf diese Weise vom Benutzer am ehesten wahrgenommen wird. Die Präsentation der Information ist sehr wichtig, weil der Anwender eines Systems dadurch das ganze System beurteilt. Ist die Präsentation schlecht, kann die hervorragende Qualität der vorgelagerten Komponenten nicht zur Wirkung kommen.

Die Frage, welches System am besten für eine Informationspräsentation geeignet ist, hängt wieder einmal von mehreren Faktoren ab, wie:

- Art der Anwendung (häufiges *Drill-down*, Data Mining, mehrdimensionale Darstellung)
- Host-/Client-Server- oder PC-basiert
- Unterstützung unterschiedlicher DB-Systeme gefordert usw.

1.8.1 Mehrere Fragen

Die sechs beantworteten Fragen (siehe Informationsbeschaffung und Informationsaufbereitung) können helfen, das Denken zu verfeinern und zu einer handlichen Lösung zu führen. Bevor man aber startet, benötigt man viel mehr Informationen über Facts, Dimensionen, Attribute, und Behandlung von „***sparsity***“[3] - dem größten Problem in MDM.
Facts und Dimensionen werden physikalisch in einer relationalen Datenbank als Tabellen dargestellt. Um ein MDM physikalisch, mittels relationaler Datenbanken darzustellen, kann man das grundlegende *Sternschema* (Abbildung 7) benutzen.

In diesem Schema wird jede Dimension durch eine eigene Tabelle dargestellt. Ein Fact wird als eine einzelne große Tabelle dargestellt und mit einem mehrteiligen Schlüssel indexiert (der Schlüssel besteht aus den individuellen Schlüsseln aller Dimensionen).

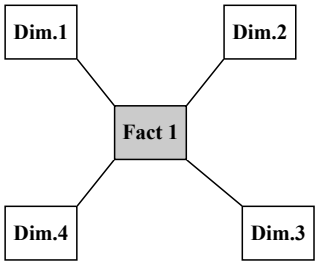

Abbildung 7: Sternschema

Es gibt viele Variationen zu dem einfachen Sternschema, alle haben aber den grundlegenden Begriff von Facttabellen und Dimensionstabellen gemeinsam.
Der Schlüsselpunkt, den man im Auge behalten sollte, ist, daß das mehrdimensionale Modellieren für relationale Datenbanken so designiert werden soll, daß „lange, dünne Facttabellen“ und verhältnismäßig „kleine, kurze und weite Dimensionstabellen“ eingerichtet werden.

Während Facttabellen die numerischen Informationen enthalten, werden die interessanten Informationen eigentlich in Dimensionstabellen aufbewahrt. Wenn Facts benötigt werden, sammelt man die Schlüsselwerte von den Dimensionstabellen und findet dann die passenden Einträge in der Facttabelle, wobei kostspieligen und zeitaufwendigen Tabellendurchsuchungen und komplexe „joins“ vermieden werden.

Attribute sind nicht "Entitäten" und werden deswegen nicht als Tabellen dargestellt. Attribute sind die verlängerten Beschreibungen und Hierarchien von Dimensionen, wie z.B. die Marke, Farbe und Größe in der Dimension „PRODUKT“.

[3] siehe Kapitel 1.5.

1.8.2 Multidimensionalität

Da das multidimensionale Modellieren immer Hierarchien in Dimensionen in sich schließt, ist die Aggregation von Informationen ein Schlüsselelement im Nutzen des Modells. Da die Aggregation meistens ein additiver Prozeß ist, ist es am besten, wenn Facts auf additive, numerische Werte beschränkt werden. Es ist auch möglich, sich mit nichtadditiven Facts (z. B. Text) zu befassen, jedoch gehört der Text, außer wenn er für jeden Satz in der Facttabelle einmalig ist, in Dimensionstabellen.
Manchmal kommt es vor, daß die Facts über einige, aber nicht alle, Dimensionen additiv sind. Manche Facts sind numerisch, aber trotzdem nichtadditiv, wie Preise oder Quoten. Man muß daher bei der Behandlung solcher Facts besondere Sorgfalt anwenden, da sie potentiell zu falschen Antworten auf Suchfragen führen können.

Das Selektieren von Facts für ein Modell ist verhältnismäßig einfach: sobald ein Geschäftsthema selektiert wird, ergibt die Antwort auf die Frage, „Was wird gemessen?“ die Auflistung von Facts.

Dimensionen sind Klassen von Deskriptoren für die Facts. Wenn das Fact ein Verkauf ist, könnte man sich unter den Dimensionen „Zeit“, „Geographie“, „Kunde“ und/oder „Produkt“ vorstellen. Wir sagen, daß die Dimensionen Klassen sind, weil sie in Attribute zerlegt werden können.

Der Begriff der Dimension ist grundsätzlich für das MDM. Der Ausdruck „*gemeinsame Geschäftsdimensionen*“ wird heutzutage sehr oft benutzt. Zum Beispiel ist es selten, daß ein MDM die Zeit nicht als eine grundsätzliche Dimension umfaßt. Andere sogenannte „gemeinsame Geschäftsdimensionen“ sind: Geographie, Organisation (wobei man hiermit die Hierarchiestufen in einer Organisation meint), Kunde und Szenario.

Die drei üblichsten MDMs sind:
- Finanzberichte,
- Finanz-Einzelhandelsberichte und
- Verkauf-Kunden-Produktberichte.

Wenn man diese Dimensionen (gemeinsame Geschäftsdimensionen) betrachtet, kann man sich fragen, ob die Unterteilung wirklich optimal ist. Die Geographie und der Kunde können immerhin eng verwandt sein. Wenn die Attribute in Dimensionen getrennt werden, müssen die Daten für verschiedene detaillierte Stufen noch immer vorhanden sein. Das Gegenteil ist auch der Fall: Wenn vielfache Dimensionen in eine zusammenbrechen, geht das Detail typischerweise verloren.
Ein großes Problem bei dem MDM ist, daß „*drill-down*“ unmöglich wird, wenn die Hierarchie eines Attributs, (wie bei der GEOGRAPHIE), nicht explizit im Design ausgedrückt ist. Falls das Navigationstool die Beziehungen zwischen Attributen nicht verstehen kann, kann er die benötigte Informationen nicht herleiten.

Auf Data Warehouse setzen solche Anwendungen an, die die dort gespeicherten Informationen auswerten. Beispiele solcher Auswertungen sind die Identifizierung von

Geschäftstrends, Analyse von Kundenprofilen, Formulierung von Marketingstrategien, Kostenanalyse etc., welche allgemein die strategischen Entscheidungsprozesse unterstützen. Hier kommen OLAP – Werkzeuge zum Einsatz [BFZ96].

1.9 Rollenkonzept und die Sicherheit von multidimensionalen Modellen

Sicherheit ist ein sehr wichtiger Punkt für jede Datenbank, die von mehreren Anwendern benutzt wird. Zwei der wichtigsten Grundsätze der Datenbanksicherheit sind :

1. Unbefugte Benutzer dürfen auf die Daten nicht zugreifen
2. Der Zugriff auf Teile der Datenbank wird für jeden einzelnen Benutzer kontrolliert

Jedem Benutzer, der den Zugriff auf ein multidimensionales Model hat, wird ein eindeutiges Login und Paßwort zugeteilt. Jedesmal, wenn der Benutzer mit dem System arbeiten will, werden seine Zugriffsrechte überprüft.

Normalerweise, gibt es in einer Organisation mehrere Angestellte, deren Aufgabenbereich die gleichen Datenbedürfnisse aufweist. Falls es z.B. mehrere Angestellte in der Personalabteilung geben sollte, würden sie alle den Zugriff auf die Mitarbeiterdaten brauchen. Oder, alle Manager einer Firma würden eine Übersicht von vorhandenen Produkten brauchen. Man könnte die Zugriffsrechte für jede einzelne Person definieren und diese bei jedem Datenbankzugriff überprüfen; viel effizienter ist es jedoch, die Zugriffsrechte für verschiedene Benutzergruppen, die sich auf eine bestimmte Art bei der Datenverarbeitung verhalten, zu definieren. Diese Art der Zugriffskontrolle ist unter dem Namen „***Role Based Access Control***“ bekannt.

1.9.1 Die Rollen

Das Grundkonzept dieser Methode der Zugriffskontrolle hat zwei Aspekte:

- Benutzern werden Rollen zugeordnet
- Die Privilegien und die Rechte werden den Rollen zugeteilt

Wenn ein Benutzer einer Rolle zugeordnet wird, erwirbt er die Privilegien und die Rechte dieser Rolle. [RAV94]

Um die Rollen (Benutzergruppen) zu erfassen, muß man eine gründliche Analyse des modellierten Systems durchführen. Alle Benutzer, die die gleichen Aufgaben haben,

bilden eine Gruppe. Ein Benutzer kann aber auch mehrere Rollen ausüben. Jede Rolle darf auf gewisse Teile der Datenbank zugreifen. Auf diese Weise werden die restlichen Teile der Datenbank von den dieser Rolle zugeordneten Benutzern geschützt.

Eine wichtige Frage bei der Zuordnung der Benutzern zu den verschiedenen Rollen ist, ob die Zuordnung völlig **zentralisiert** erfolgt (nur auf den „Security Manager" beschränkt), oder ob es eine gewisse **Dezentralisierung** gibt. Bei der dezentralisierten Zuordnung sind bestimmte Benutzer in der Lage, andere Benutzer verschiedenen Rollen zuzuordnen.

Der Vorteil des ersten Verfahrens ist die straffe Kontrolle, die dadurch erzielt wird, und eine zentralisierte Verantwortung. Der Nachteil ist sicherlich der erhöhte administrative Aufwand, sich mit routinemäßigen Vorgängen zu befassen, besonders, wenn das System zu groß wird.
Es gibt zwei verschiedene Aspekte der Benutzerzuordnung zu den Rollen. Beide sind besonders kritisch, wenn es um die dezentralisierte Zuordnung geht.

Der erste Aspekt ist die Frage, ob es gewisse Beschränkungen bezüglich der Rollen, zu denen ein Anwender gehören kann, gibt. In vielen Anwendungen wird angenommen, daß sich einige Rollen gegenseitig ausschließen, zum Zwecke der Trennung von Pflichten. Als Beispiel könnte man sich einen Angestellten vorstellen, der befugt ist, Rechnungen zu unterschreiben und einen anderen Angestellten, der autorisiert ist, das Geld auszuzahlen. Falls nur ein Benutzer beide Rollen gleichzeitig durchführen darf, wird die Anfälligkeit dieser Person für Betrug wegen Mißbrauchs der autorisierten Privilegien erhöht. Um diese Möglichkeit zu vermeiden, kann das gegenseitige Ausschließen dieser zwei Rollen festgesetzt werden, so daß kein Benutzer beide Rollen gleichzeitig ausüben kann.

Der zweite Aspekt beschäftigt sich mit der Beschränkung, die besagt, wann ein Anwender zu einer bestimmten Rolle gehören kann. Zum Beispiel, könnte es folgende Beschränkung geben: ein Benutzer kann der Rolle des Hardwareingenieurs nur dann zugeordnet werden, wenn er die Ingenieurrolle schon hat. In solch einem Fall könnte die Zuordnung von Benutzern zu der Ingenieurrolle zentralisiert kontrolliert werden (vom „Security Manager") während die Zuordnung zu den spezialisierten Ingenieurrollen dezentralisiert erfolgt.

Rollengebrauch

Für die Anwender ist es günstig, gleichzeitig alle ihre Rollen auszuüben zu dürfen. Auf diese Weise können sie gleichzeitig alle ihre Privilegien und Rechte erwerben.
Man kann in Bezug auf Rollen einige zeitliche Beschränkungen einfügen: wie lang ein Benutzer seine Rolle behalten darf und wie oft sie in einem vorgegebenen Zeitintervall ausgeübt werden darf. Die Begründung hier ist die, daß die Schäden wegen Mißbrauchs der Rolle beschränkt werden können.
Nehmen wir an, daß eine Rolle auf Verkaufsdaten für verschiedene Geschäfte in einer Region zugreifen darf, nicht aber auf die Verkaufszahlen für die ganze Region. Dann könnte ein Benutzer, der diese Rolle ausübt, die Geschäftsdaten aller Geschäfte

dieser Region ablesen, die Verkaufszahlen summieren und schon hätte er das Ergebnis vor Augen. Um solche Aktionen zu vermeiden, kann man eine zeitliche Beschränkung für diese Rolle hinzufügen, so daß der Benutzer auf keinen Fall die Gesamtzahlen bekommen könnte.

Rollenevolution

Das Problem der Rollenevolution wird oft vernachlässigt, obwohl es ein sehr wichtiges Thema ist. In großen Unternehmungen gibt es viele Rollen, für die man erwarten kann, daß sie sich mit der Zeit ändern werden. Mit dem Unternehmungswachstum werden die existierenden Rollen zusammengeführt, geteilt, vernichtet und viele neue werden eingerichtet. Deswegen ist es von großer Bedeutung, daß die Sicherheitssysteme diese Änderungen folgen.

Immer mehr Datenbanken werden ans Netz geschlossen. Dies bringt mit sich weitere Aspekte der Sicherheit[4].

1.10 Das „GROCERY“ Beispiel

Der praktische Teil dieser Diplomarbeit wurde auf einem mehrdimensionalen Datenmodel aufgebaut. Es handelt sich hier um ein „Grocery“ Data Warehouse, das die Verkaufsdaten einer Supermarktkette mit 60 Geschäften erfaßt.

Das Data Warehouse hat vier Dimensionen :

- Time
- Product
- Promotion
- Store

und ein Fact: „Sales_fact“.

Die wichtigsten Informationsquellen in einem Geschäft sind die Kassen und die Liefereingänge.

[4] Mehr über Netzsicherheit wird im Kapitel 4 gesagt.

1.10.1 Aufbau des „Grocery“ Data Warehouses

- **Die Dimension „Time“**

Die niedrigste Aggregationsstufe dieser Dimension ist der Tag. Dadurch wird ein detaillierter Blick in die Tagesgeschäfte ermöglicht (z.B. welches Produkt in einem bestimmten Geschäft zu welchem Preis an einem bestimmten Tag verkauft wurde). Es lassen sich auch Tagestrends ablesen: an welchen Tagen der Verkauf am besten war, welche Tage den kleinsten Umsatz erbrachten usw.
Die Daten, die in dieser Dimension aufbewahrt werden, sind:

time_key, day_of_week, week_number_in_year, month, quarter, fiscal_period und *year*.

time_key ist der Primärschlüssel der Tabelle.

Die „Time“ Dimension weist eine einfache Hierarchie auf, die dem alltäglichen menschliche Denken folgt : die niedrigste Hierarchiestufe ist der Tag und die höchste Hierarchiestufe ist das Jahr (Abbildung 8).

- **Die Dimension „Product“**

In jedem Geschäft werden 60 verschiedene Produkte angeboten. Jedes Produkt wird durch eine eindeutige Produktnummer (SKU: „stock keeping units“) spezifiziert. Die Produkte werden in drei Abteilungen („grocery“, „frozen food“ und „meat“) organisiert und durch folgende Attribute detailliert bezeichnet:

product_key, description, full_description, SKU_number, package_size, package_type, diet_type, weight, weigth_unit_of_measure, units_per_retail_case, units_per_shipping_case, cases_per_pallet, shelf_width_cm, shelf_height_cm, shelf_depth_cm, brand, subcategory, category und *department*.

Der Primärschlüssel dieser Tabelle ist *product_key*.

Die hierarchische Darstellung weist 5 Stufen auf. Die höchste Hierarchiestufe ist *department*, dann kommen *category*, *subcategory* und *brand*. Alle andere Attribute dieser Dimension (von *description* bis *shelf_depth_cm*) stellen die niedrigste Aggregationsstufe dar.(Abbildung 8)

- **Die Dimension „Store“**

Die in dieser Dimension gespeicherten Daten beschreiben jede einzelne Filiale. Die hierarchische Darstellung ermöglicht eine geographische Analyse von verkauften Produkten in einem gewissen Zeitraum. Durch das Summieren von Verkaufszahlen kann man darauf schließen, in welchen Gebieten die Nachfrage nach einem bestimmten Produkt groß ist, welche Geschäfte unrentabel sind usw.

Dimension „Store“ hat *store_key* als Primärschlüssel und besteht aus folgenden Daten :

store_key, name, city, sales_district und *sales_region*

Die Hierarchie ist hier mit *sales_region* als oberster und *name* des Geschäftes als niedrigster Hierarchiestufe sehr einfach (Abbildung 8).

- **Die Dimension „Promotion“**

Diese Dimension beinhaltet die Daten der Werbestrategien und Verkaufsbedingungen von verschiedenen Produkten. Die Informationen, die hier gespeichert werden,
sind :

promotion_key, promotion_name, price_reduction_type, ad_type, display_type, coupon_type, ad_media_type, display_provider und *promo_cost.*

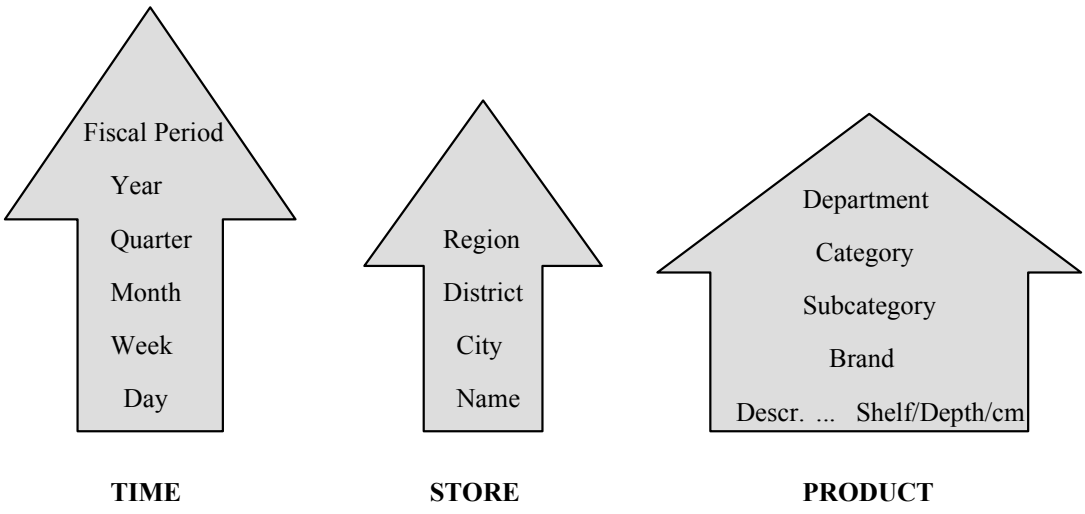

Abbildung 8: Hierarchische Darstellung der Dimensionen

- **Die Fact – Tabelle „Sales_fact“**

Die Fact – Tabelle „Sales fact“ beinhaltet die Verkaufszahlen für die ganze Supermarktkette. Die Werte, die in dieser Tabelle gefaßt werden, sind numerische Werte, die summiert werden können und auf die Weise das Verkaufsverhalten nachvollziehen können. Die Daten aus allen vier Dimensionen werden über ihre Primärschlüssel, über diese Fact-Tabelle, explizit miteinander verbunden. Die „Sales_fact“ Tabelle stellt den Kern des Star – Schemas dar.

Die Fact – Tabelle besteht aus folgenden Attributen :

time_key, store_key, product_key, promotion_key, dollar_sales, unit_sales, dollar_cost, customer_count, avg_price, avg_cost, avg_purchased_dollars, avg_purchased_number, gross_profit und *gross_margin*
Der Primärschlüssel dieser Tabelle setzt sich aus den Primärschlüsseln der vier Dimensionen zusammen.

Einige Attribute der Fact – Tabelle sind nicht editierbar, das heißt, sie werden aus den anderen Attributen berechnet (z.B. *avg_cost =dollar_sales / unit_sales).*

2 On-Line Analytical Processing (OLAP)

2.1 Einleitung

Multidimensional-modellierte Datenbanken, welche bei Data Warehouses oft gefunden werden, speichern große Menge von Daten. Um die Daten optimal nutzen zu können, um schnell die gewünschten Antworten aus solchen Daten ableiten zu können, braucht man ein mächtiges Werkzeug. OLAP ermöglicht sehr effiziente Datenverarbeitung bei großen Datenmengen und wird immer unverzichtbarer für jedes Data Warehouse.

OLAP und Data Warehouse ergänzen sich. Ein Data Warehouse speichert und managt die Daten. OLAP transformiert diese Daten in strategische Informationen. Die Möglichkeiten von OLAP gehen vom einfachen Navigieren und "browsing" zu komplizierteren Kalkulationen und ernsteren Analysen. Man kann dies auch als einfachen „Datenzugriff" und „Wissen" bezeichnen.

OLAP ermöglicht Analytikern, Managern und Datenbearbeitern, durch schnellen, konsequenten, interaktiven Zugriff auf die Informationen, Verständnis für Daten zu erwerben. OLAP transformiert Rohdaten, so daß sie die wirkliche Dimensionalität des Unternehmens widerspiegeln und vom Anwender verstanden werden.

2.2 Die Architektur

Der ganze Prozeß, in dem die Daten aus Data Warehouse geholt, in OLAP-Engine vorbereitet und dem Anwender angeboten werden, kann man in einem 3-Schichten Model darstellen. (Abbildung 9)

Datenschicht

In dieser Schicht wird das Data Warehouse mit Daten versorgt. Das Data Warehouse beinhaltet Daten aus sowohl internen, als auch externen Datenquellen. In dieser Schicht wird eine Datentransformation unternommen, so daß Analyseprozesse schnell durchgeführt werden können. Die Daten aus verteilten, operationalen Datenquellen werden verarbeitet und zusammengeführt, so daß eine verdichtete Datenbank entsteht[5]. Da sich Informationen und Informationsquellen ändern, muß in dieser Schicht auch für das Aktualisieren von Daten gesorgt werden. Dies soll aber so ge-

[5] Für mehr Informationen siehe [STO97]

schehen, daß das Data Warehouse ungestört und ununterbrochen funktionieren kann („on-line“)

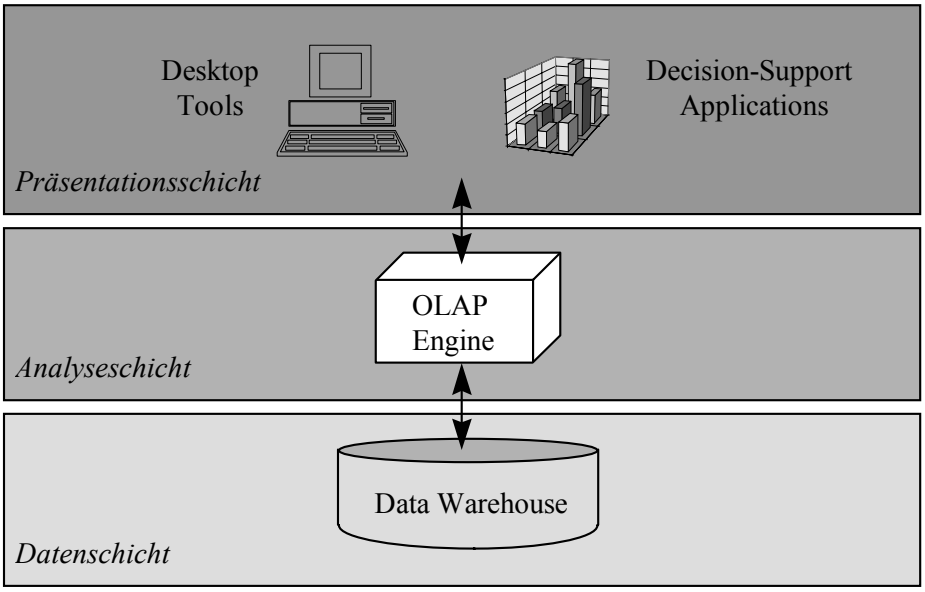

Abbildung 9: Das 3 - Schichten Model

Analyseschicht

Diese mittlere Schicht stellt eine Verbindung zwischen den Rohdaten, die aus der Datenschicht kommen, und den zu präsentierenden Daten, die der Endbenutzer bekommt, dar. An diese Schicht wird jede Benutzerabfrage weitergeleitet mit den Metadaten kombiniert. Aufgrund bestimmter Algorithmen wird dann eine SQL-Abfrage gebildet und zum Data Warehouse geschickt. Die Ergebnisse der Abfrage werden aus dem Data Warehouse wieder an die Analyseschicht geschickt. (Um eine befriedigende Funktion zu gewährleisten, muß die Antwortzeit unter 5 Sekunden liegen). Hier werden die zu präsentierenden Daten verarbeitet und für die Visualisierung vorbereitet.

Präsentationsschicht

Die Präsentationsschicht beinhaltet eine Benutzerschnittstelle. Hier kann der Endbenutzer seine Abfragen formulieren, und hier werden die Ergebnisse präsentiert. Dem Benutzer wird auch angeboten, durch den virtuellen Datenwürfel zu navigieren um detailliertere und gröbere Sichten von Informationen zu bekommen.
Eine vom Benutzer definierte Abfrage wird transformiert und an die Analyseschicht geschickt. Wenn ein Ergebnis aus Anayseschicht kommt, wird es interpretiert und dem Anwender präsentiert.

2.3 OLAP - Charakteristiken

OLAP-Systeme haben nicht nur die Fähigkeit, Fragen wie "wer?"- und "was?" - zu beantworten, sondern auch Fragen wie „was wenn ?" und "warum ?", was sie von Data Warehouse stark unterscheidet.

OLAP ermöglicht Entscheidungsfindung über künftige Aktionen. Eine typische OLAP Kalkulation ist komplexer als das einfache Summieren von Daten, zum Beispiel: "Wie würde sich der Preis von alkoholfreien Getränken ändern, wenn Sirup-preise $.10/gallon hinaufgingen und Transportkosten $.05/mile hinuntergingen?[FORS]

Die wichtigsten OLAP Eigenschaften sind : Geschwindigkeit, Analyse, Datenkapazität, Mehrdimensionalität, Aktualisieren von Daten, Transparenz und Integrität und Zugänglichkeit. Sie werden hier kurz beschrieben:

- **Geschwindigkeit**

Ein OLAP-System ist so gebaut, daß die meisten Ergebnisse für Anwender innerhalb von ungefähr fünf Sekunden zu liefern sind, wobei die einfachsten Analysen nicht mehr als eine Sekunde und sehr wenige mehr als 20 Sekunden dauern. Die jüngsten Forschungen haben gezeigt, daß Endbenutzer annehmen, ein Prozeß habe „versagt", wenn innerhalb von 30 Sekunden keine Ergebnisse geliefert wurden.
Eine solche Geschwindigkeit ist mit großen Datenmengen nicht leicht zu vollbringen, besonders, wenn „on-the-fly" und ad hoc Kalkulationen gebraucht werden. Um dieses Problem möglichst optimal zu lösen, benutzen die Hersteller spezialisierte Formen von Datenspeicher, umfassende Vorkalkulationen und bestimmte Hardware - Voraussetzungen. Trotzdem sind diese Probleme bis jetzt nicht vollständig gelöst und es gibt derzeit kein System, das voll optimiert ist. Im besonderen mißlingt der Vorkalkulationsprinzip mit sehr großen Programmen, wenn auch die Datenbanken zu groß werden. Alles „on-the-fly" zu erstellen wäre für große Datenbanken viel zu langsam, auch wenn man sehr gute Hardware benutzen würde.

- **Analyse**

Das System kann mit jeglicher Geschäftslogik und statistischer Analyse, die für die Anwendung und den Anwender relevant ist, umgehen, und das ganze für den Zielanwender einfach genug halten. Es ist sicherlich notwendig dem Anwender zu erlauben, neue ad hoc Kalkulationen als Teil der Analyse zu definieren und die Berichte über die Daten in jeder gewünschten Weise zu verlangen.

- **Datenkapazität**

Unter der Kapazität von verschiedenen OLAP-Produkten, wird nicht die Anzahl an Gigabyte von Daten, die sie speichern können, verstanden. Die Kapazität bezeichnet die Datenmengen, die von einem OLAP-Produkt behandelt werden können. Die Kapazitäten einzelner Produkten unterscheiden sich sehr: die größten OLAP Produkte können mindestens tausend mal mehr Daten behandeln als die kleinsten.[PEN97]

- **Mehrdimensionalität**

Mehrdimensionalität ist die wichtigste Eigenschaft eines OLAP Systems. Das System muß einen mehrdimensionalen konzeptuellen Blick auf die Daten ermöglichen, wobei volle Unterstützung für Hierarchien und vielfache Hierarchien implementiert werden soll. Die hierarchische Art der Datendarstellung ist sicherlich die natürlichste Weise, die Geschäfte und Organisationen zu analysieren. Ein OLAP System setzt aber nicht voraus, welche zugrunde liegende Datenbanktechnologie benutzt werden soll.

OLAP-Systeme, die momentan vorhanden sind, können in zwei Gruppen geteilt werden: Systeme mit einfacher multidimensionaler Struktur („single-structure multidimensioanlity“) und solche mit vielfachmultidimensionalen Struktur („*multiple multidimensionality*“).

Die Systeme mit der einfachen multidimensionalen Struktur enthalten ein multidimensionales Modell pro Datenwürfel oder Datenbank. Wenn ein neuer Datenwürfel benötigt wird, werden ein neues Modell und eine neue Datenbank eingerichtet.

Vielfach multidimensionale Systeme können pro Modell mehr als nur einen Datenwürfel enthalten, genauso wie die relationalen Datenbanken mehr als nur eine Tabelle enthalten können.

Die Abbildung 10 stellt ein vielfach multidimensionales Systems dar.

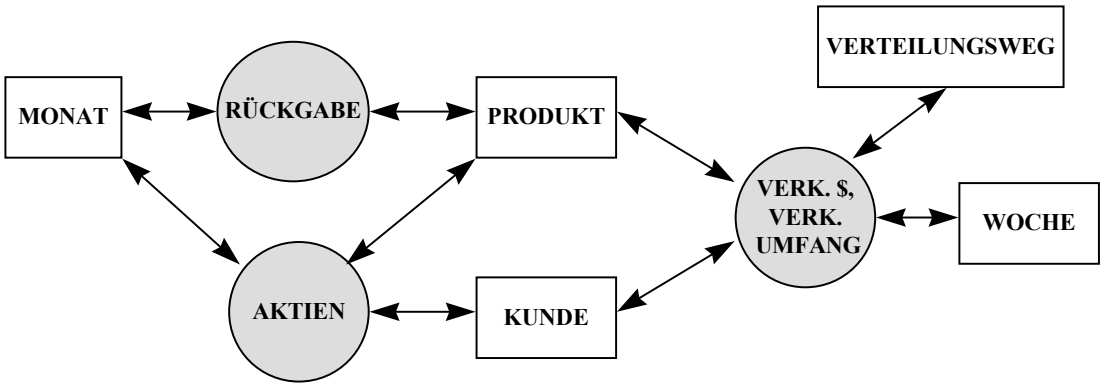

Abbildung 10: **Ein vielfach multidimensionales System [BUY95]**

Das abgebildete Datenmodell zeigt, daß „Verkauf $“ und „Verkaufsumfang“ pro Kunde, Produkt, Woche und Verteilungsweg gemessen werden (4-dimensionales Beispiel). „Rückgabe“ (zurückgegebene Waren) hängen von Kunden, Produkte und Monat ab (3-dim. Beispiel), und „Aktien“ sind von Monat und Produkt abhängig (2-dim. Beispiel). Dieses Beispiel zeigt auch, daß eine Variable, die von einer bestimmten
Dimension nicht abhängt, diese Dimension auch nicht enthält (wie „Aktien“ und Woche), was aber bei einem System mit einfacher multidimensionaler Struktur der Fall wäre.

Die Hauptdifferenz zwischen einfach und mehrfach multidimensionalstrukturierten Systemen ist die Fähigkeit des letzteren, Dimensionen für die verschiedenen Datenwürfel wieder zu verwenden. Dimensionswartung, wie neue Einträge (Kunden, Produkte usw.) einfügen, brauchen nur einmal und nicht für jeden Datenwürfel gemacht zu werden. Die Systeme mit der vielfach multidimensionalen Struktur versichern daher die Datenintegrität.

- **Aktualisieren von Daten**

Wenn die Menge historischer Daten zunimmt, sollte die Geschwindigkeit, mit der die Daten in einer OLAP- Anwendung verarbeitet werden, stabil bleiben. Dies erfordert wachsende Aktualisierungsfähigkeiten. Üblicherweise wird damit die wöchentliche oder monatliche Verarbeitung von neuen Daten gemeint, ohne daß die historischen Daten dabei rekalkuliert werden müssen. Nach den OLAP-Regeln sollte dieses Prinzip auf alle Bereiche angewendet werden. Wenn man eine Abfrage wie „Berechne die Einkommensdifferenz für die zwei bestverkauften Produkte in den letzten 5 Monaten“ formuliert, soll die Analyse nicht auf alle Produkte und alle Monate angewendet werden müssen, um aus dem Gesamtergebnis das benötigte herauszunehmen, sondern es muß möglich sein, das Prozessieren nur auf den betreffenden Bereich zu beschränken.

- **Transparenz und Integrität**

OLAP-Produkte sollten offenen Zugriff für verschiedene Endbenutzeroberflächen zur Verfügung stellen. Obwohl die Daten aus verschiedensten Datenquellen geholt werden können, sollte durch die Präsentationssoftware der Zugriff auf alle ermöglicht werden.

Eine Kette der Datenwerte (von oben nach unten), die auf OLAP angewandt wird, könnte aus folgenden Schritten bestehen:

- ***Dokumentation:*** alle Änderungen der Anwendung werden für Kontrollzwecke automatisch protokolliert
- ***Präsentation****:* interaktive Darstellung am Bildschirm, Papier oder anderen Medien
- ***Analyse:*** “drill down“, „exeption reporting“, Vorhersage, „was wenn“ usw.
- ***Manipulation:*** Konsolidierung, Währungsumwandlung usw.

- ***Kombination:*** die Daten werden in einer mehrdimensionalen Struktur abgespeichert
- ***Import:*** die Daten werden in die OLAP *Tool* geladen
- ***Definition:*** eine konsistente Datendefinitionen muß vereinbart werden, so daß die verschiedenen Daten verglichen werden können.

Wenn alle Schritte innerhalb einer integrierten Umgebung ohne besondere Übersetzungsschritte ausgeführt werden können, ist eine optimale Wartung erreicht worden.

Eine integrierte Umgebung sollte datengesteuerte Wartung anbieten. Dies bedeutet, daß z. B. die neuen Produkte und Kunden sofort richtig bearbeitet werden und automatisch in der Präsentationsschicht der OLAP-Anwendung erscheinen. Das Gegenteil von der datengesteuerten ist die „hart kodierte" Wartung. Datengesteuerte Systeme heben die Datenmodellierung hervor, die hart kodierten Systeme heben das Programmieren hervor.

Ein anderer Aspekt einer integrierten Umgebung ist eine gleiche Behandlung von allen Datentypen. OLAP Systeme sollten ein Datum genauso wie einen Text mehrdimensional behandeln können, ohne daß dadurch zusätzliche Datenkontrolle nötig wird.

Eine erfolgreich implementierte Klient-/Server Architektur für OLAP ist mit einer integrierten Umgebung und Transparenz eng verbunden. Wartung und Kontrolle können minimiert werden, wenn sowohl der Klient als auch der Server die gleiche Datendefinitionssprache und die gleiche Datenbehandlungssprache benutzen. In diesem Fall brauchen die Entwickler nur eine Sprache zu beherrschen, und Anwender wurden potentielle Engpässe nicht merken.

- **Zugänglichkeit**

Zugänglichkeit ist eine Fähigkeit von OLAP Produkten, auf benötigte Daten zugreifen zu können, ohne daß der Anwender merkt, daß die Daten aus verschiedenen Datenquellen stammen.

Es gibt zwei grundlegende Konzepte der Datenspeicherung:
- kontinuierlicher Zugriff auf Transaktionssysteme
- häufiges Aufladen in das MDBMS

Das Abfragen eines Transaktionssystems ist oft schwierig, weil (der relationalen Theorie nach) die Daten normalisiert werden müssen, folglich auch optimiert für das transaktionelle Prozessieren. OLTP-Systeme auf OLAP-Bedarfe anzupassen ist keine gute Lösung, weil das zusätzliche Kontroll- und Wartungsmaßnahmen für die OLTP-Umgebung verursachen würde. Ein MDBMS-Data Warehouse ist wünschenswert.

Das Aufladen von Daten in ein MDBMS kann in einer datengesteuerten Weise erfolgen. Die Relationen zwischen Dimensionen werden ausgenutzt, um die Datenintegrität zu erzielen. [BUY95]

2.4 Die OLAP – Produkte

Die OLAP – Produkte unterteilen sich in drei große Gruppen[MIC95]:

- Ad hoc *Query Tools* and Report Writers
- Multidimensional OLAP
- Relational OLAP

2.4.1 Ad hoc *Query Tools* and Report Writers

Diese Produkte werden in den Systemen für allgemeine Entscheidungszwecke, die mit den relationalen Datenbanken arbeiten, gefunden. Durch ihre graphische Benutzerschnittstelle, die die „point-and-click“ Technik unterstützt, wird der Aufbau der SQL-Abfragen, die für den Datenbankzugriff benötigt werden, automatisiert.

Diese Produkte funktionieren gut für das einfache Abfragen von Daten. Dank der guten graphischen Schnittstelle ist das Benutzerspektrum hier sehr breit.

Der Hauptvorteil der „Ad hoc *Query Tools* and Report Writers“ ist der, daß sie eine Unterstützung für die offene Architektur durch Nutzung von ANSI-Standard SQL bieten. Dadurch werden diese Produkte sehr einfach an die ständig ändernden Anforderungen der Informationssysteme angepaßt.

Während „Report Writers“ viele Vorteile als grundlegende Berichtwerkzeuge anbieten, haben sie Schwierigkeiten, vielen Anforderungen, die Organisationen auf ihren Entscheidungssystemen stellen, zu entsprechen. Ihr größter Nachteil folgt aus der Tatsache, daß Endanwender noch direkt am SQL-Definitionsprozeß beteiligt sind. Um eine vereinfachte ad hoc - Abfragenumgebung zu vollbringen , mußte man hier auf analytische Komplexität verzichten. Demzufolge bieten diese Produkte die komplexen Analysen, die von heutigen Geschäftsanwendern gebraucht werden, nicht.
Ein weiterer Nachteil dieser Produkte bezieht sich auf die Antwortzeit. Datenzugriffszeiten sind, wenn es um kleinen Data Warehouses geht, nicht von großer Bedeutung; wachsen sie aber auf ein paar Gigabyte oder sogar auf zehn und hundert Gigabyte, muß das Data Warehouse Schema optimiert werden, um sinnvolle Antwortzeiten zu vollbringen. Optimierungsverfahren wie Aggregation und/oder Teilung von Fakttabellen, Denormalisierung von Relationstabellen, verbessern die Antwortzeiten, vergrößern aber die Komplexität der SQL - Abfragen. Diese Produkte haben

folglich Schwierigkeiten, sinnvolle Antwortzeiten bei großen Data Warehouses zu schaffen.

Die Personen, die in einer Organisation mit einem Data Warehouse arbeiten, besitzen oft unterschiedliche technische Vorkenntnisse. Da etwas technisches Verständnis von der zugrunde liegenden Datenbankstrukturen immer gebraucht wird, um die ad hoc Abfragenwerkzeuge zu benutzen, sind die „Ad hoc *Query Tools* and Report Writers“ keine realistische Lösungen für den technisch weniger geschickten Anwender einer Organisation.

2.4.2 Multidimensional OLAP (MOLAP)

MOLAP gehört zu der Klasse von Systemen, die mit den mehrdimensionalen Datenbanken arbeiten und raffinierte Analysemöglichkeiten anbieten. MOLAP benutzt den mehrdimensionalen Datenwürfel. Die mehrdimensionalen Datenbanken beruhen auf einer Technologie der „Array-Processing“, wodurch sie fähig sind , detaillierte Analysen kleiner Datenmengen , die aus dem Data Warehouse gewonnen werden, durchzuführen..

Ein Vorteil vom MOLAP ist die Tatsache, daß den Benutzern eine intuitive Sicht der Daten angeboten wird. Die Anwender sehen die Daten multidimensional und denken dabei so, wie sie es im Zusammenhang mit ihren alltäglichen Geschäftstransaktionen tun würden. Außerdem wird hier dem Anwender eine hohe analytische Komplexität zur Verfügung gestellt.
Jedoch gibt es bei dieser Methode gewisse Beschränkungen. Eine davon ist die mangelhafte Skalierbarkeit und Flexibilität. Die mehrdimensionalen Datenwürfel sind inflexibel und unterstützen das Kreieren von ad hoc multidimensionalen *Views* nicht. Noch wichtiger, es können die mehrdimensionale Datenbanken nicht mehr als 20 oder 30 Gigabyte der Daten behandeln. Deswegen ist MOLAP nur für Organisationen mit den kleinen (5-10 Gigabyte) Data Warehouses oder für abteilungsinterne Lösungen eine praktische Technologie.

Ein weiterer Nachteil von MOLAP Technologie ist ihre Unfähigkeit, offene Architektur zu unterstützen. Deswegen werden die Organisationen gezwungen, Hilfsmittel anzuwenden, um ein nicht standardisiertes System zu unterstützen.

Aus diesen Gründen stellen die mehrdimensionalen Datenbanken keine realistische entscheidungsunterstützende Lösung dar.

2.4.3 Relational OLAP (ROLAP)

ROLAP Systeme kombinieren die direkte Datenzugriffsfunktionalität der „Report Writers“ mit den fortgeschrittenen analytischen Möglichkeiten der MOLAP Systemen. Dadurch erreichen sie eine offene, skalierbare „data delivery“ Architektur.

Die ROLAP-Systeme nutzen die dynamische „OLAP-Engine“ , um die SQL - Abfragen zu generieren, die benutzt werden, um auf die relationalen Data Warehouses zuzugreifen. Den Endbenutzer steht eine intuitive mehrdimensionale analytische Schnittstelle zur Verfügung, durch die sie komplexe multidimensionale Analysen definieren können.
Die wichtigsten Merkmale eines ROLAP-Systems sind:

- *Hochentwickelte Analysenfunktionalität:* Zum Unterschied zu den „Report Writers“, beschäftigen sich die Endbenutzer hier nicht mehr mit dem Generieren von SQL-Abfragen.
- *Unterstützung eines großen Benutzerkreises:* ROLAP-Systeme erfüllen die Bedürfnisse sowohl der Benutzer die sich mit der technischen Analyse beschäftigen, als auch der, die wichtige Entscheidungen in einer Organisation zu treffen haben.

- *Hohe Endbenutzerproduktivität:* Die Endbenutzerproduktivität kann durch das Einsetzen von ROLAP-Systemen herrlich vergrößert werden. Die meisten ROLAP Systeme gründen auf einer „object library“ die sowohl einfach zu benutzen ist als auch die Arbeit bedeutend erleichtert und beschleunigt.

- *Unterstützung von sehr großen Data Warehouses:* ROLAP-Systeme sind zu Optimierungsverfahren fähig, die die rechtzeitige OLAP-Analyse sehr großer Datenbanken ermöglichen.

- *Unterstützung von großen Anzahl von Benutzer:* ROLAP-Entscheidungssysteme, sind fähig, Hunderte von Anwendern, die sich überall in der Welt befinden, zu unterstützen. Wenn sie sich an den ROLAP-Server binden, können ihre eigene Systeme Hintergrundverarbeitung durchführen, Abfragen leiten usw.

- *Einfach herzustellen und zu warten:* ROLAP-Systeme benutzen Metadaten - Dateien, die das gesamte System beschreiben. Durch das einfache Verändern dieser Dateien können Organisationen ihre relationalen OLAP-Systeme schnell einrichten, modifizieren und aktualisieren. ROLAP Systeme unterstützen offene Architektur.

Relationale OLAP-Systeme sind eine hybride Technologie, die sich das Beste von „Ad hoc *Query Tools* and Report Writers“ und mehrdimensionalen OLAP-Systemen genommen hat. Diese Technologie erfüllt alle Anforderungen der modernen Data Warehouses und des Decision Supports. Sie stellt eine flexible und robuste Plattform, die weiter wachsen kann, zur Verfügung.

	Ad hoc *Query Tools* and Report Writers	**MOLAP**	**ROLAP**
Raffinierte analytische Funktionalität	○	●	●
Unterstützung großer Benutzerkreise	○	◐	●
Hohe Produktivität der Endbenutzer	◐	●	●
Unterstützung großer Data Warehouses	◐	○	●
Unterstützung einer großen Anzahl von Benutzern	◐	◐	●
Einfach einzurichten und zu Warten	◐	○	●
Unterstützung von offener Architektur	●	○	●

Tabelle 5: Vergleich von Eigenschaften verschiedenen OLAP-Systemen, [MIC95]

Legende : Keine ○ Teilweise ◐ Voll ●

2.5 Welche OLAP - Architektur ist die Passende ?

Bei der Auswahl der OLAP-Werkzeuge muß man gewisse Dinge beachten, wie das Laden , Managen und Umgehen mit den Daten. Da viele Leute schon Erfahrungen mit dem Prozessieren, Speichern, Managen, Warten und Erweitern von Daten die in einer relationalen Datenbank stehen, haben, ziehen sie es oft vor, ihre Daten in solchen Datenbanken auch zu lassen, statt wegen des Entscheidungsprozesses die Daten in andere (multidimensionale Datenbanken) überzuladen. ROLAP bietet die Möglichkeit, sowohl mit den relationalen als auch mit den multidimensionalen Datenbanken zu arbeiten. Dabei muß die Architektur, die dahinter steht, nicht streng vordefiniert werden.

Eine Möglichkeit ist, die SQL Abfragen, die für die relationale Datenbanken benötigt werden, bei dem Klienten zu definieren, sie dann an die Datenbank zu schicken und auf die Antwort zu warten, wobei das Ergebnis in multidimensionalen Format dargestellt wird.

Eine andere Lösung ist die „three-tier architecture“, wo der Server die mittlere Schicht repräsentiert. Der Server generiert in diesem Fall die Abfragen, speichert die Daten im Cache, so daß sie nicht an den Kunden gehen. Da die Daten durch das Ca-

che gehen müssen, kann man mit viel größeren Datenmengen arbeiten (ein Server kann viel mehr Daten bearbeiten , als auf einen PC geholt werden kann) und die Berechnungen bei dem Server durchführen.

Einige OLAP-Produkte (wie z.B. „Oracle Express“) die die „three-tier architecture“ benutzen, ermöglichen das „on the fly“ Erstellen von SQL-Abfragen. Der Server (mittlere Stufe der Architektur) formuliert die multidimensionalen Arrays, um sie dem Klienten anzubieten. Eine zusätzliche Leistung ist auch das Speichern von solchen Arrays in der ROLAP Engine, so daß man nicht jedesmal , wenn sie benötigt werden, auf die relationale Datenbank zugreifen muß. Sogar die Daten, die zu einer gemeinsamen Analyse gehören, können unterschiedlich aufbewahrt werden (relationale DB oder multidimensionale Arrays). Das Ganze ist für den Benutzer transparent.

Die Frage die sich stellt, ist, welche Daten in den relationalen Datenbanken bleiben und welche multidimensional gespeichert werden sollten. Die Antwort hängt von den Aufgaben ab, die mit den Daten durchgeführt werden. Wenn man komplexe multidimensionale Modelle baut oder Simulationen durchführt („what-if“ Analysen), dann kann man die komplizierten Kalkulationen nicht so einfach im SQL ausdrucken. Das heißt weiter, daß es ungeschickt wäre, in einem solchen Fall mit einer relationalen Datenbank zu arbeiten. Viel gängiger wäre es, mit einer Art multidimensionalen Cache zu arbeiten. Dabei muß man nicht sofort alle Daten so einer multidimensionalen Speicherung unterziehen. Fast 80% der Daten können in der relationalen Datenbank bleiben, und werden nach Bedarf daraus gelesen. Die restlichen 20%, die bei den multidimensionalen Modellen oder Analysen sehr oft gebraucht werden, gehören eigentlich fast ausschließlich den höheren Aggregationsstufen an. Statt jedesmal, wenn die Daten gebraucht werden, in die relationale Datenbank zu gehen und sie dynamisch zu erzeugen, ist es viel effizienter, die Daten in dem multidimensionalen Cache aufzubewahren.

Einer der größten Vorteile, die eine solche Mischung aus der relationalen und multidimensionalen Aufbewahrung von Data mit sich bringt, ist auch die Möglichkeit, die Aggregationen dynamisch zu erzeugen. Beispielsweise, würde das heißen: wenn man nur die relationalen Datenbanken benutzt, um die Verkäufe einer Region im Bezug auf die Zeit zu erfassen, würde man wahrscheinlich eine Tabelle für die jährliche, eine für die monatliche und schließlich eine für die täglichen Verkäufe führen. Falls eine Abfrage kommt, in der man die Verkäufe im Mai 1997 ansehen will, und falls es diesen Eintrag in der Monatstabelle nicht gibt, wird die Abfrage nicht beantwortet. Bei einer Kombination von Datenspeicherungen würde man zuerst überprüfen, ob die Abfrage im multidimensionalen Cache direkt beantwortet werden kann. Falls das auf Grund der dort gespeicherten Daten nicht möglich sein sollte, würde die Abfrage an die relationale Datenbank weitergeleitet werden, wo man alle Tagesverkäufe des Monats Mai dynamisch aggregieren könnte und auf die Weise das Ergebnis bekommen würde.

Wenn man sich zwischen verschiedenen OLAP Produkten entscheiden muß, sollte man drei Dinge in Betracht ziehen: welche Anwendungen sollen durchgeführt werden, wer wird das System benutzen und was werden die Benutzer brauchen. Sehr oft stellt sich heraus, daß viele Anwender Analysen brauchen, diese aber nicht selbst

durchführen müssen. In dem Fall werden die Analysen von Experten durchgeführt; alle andere Anwender benutzen sie nur.

2.5.1 ROLAP vs. MOLAP

Es besteht ein signifikanter Designunterschied zwischen ROLAP und MOLAP Architekturen. Die ROLAP Architektur verwendet relationale Datenbanken, um Daten zu speichern und zu verwalten, während die Architektur von MOLAP auf der multidimensionalen Datenbank basiert. MDDBs sind optimiert für kurze Antwortzeiten der Abfragen - deshalb besitzen die MOLAP Systeme ein besseres Hashing als die ROLAP Systeme.

Die Philosophie von MDDB beruht auf dem Erzielen der maximalen Abfragenperformanz. Diese wird durch starke Voraggregation der Daten erreicht. MDDBs sind besonders empfehlenswert wenn es um die Daten bis 5GB geht [MIC95]. ROLAP zeichnet sich anderseits durch vorteilhafte dynamische Konsolidierung aus und zeigt keine Schwächen gegenüber der Datenkompilation . Außerdem bietet ROLAP die Möglichkeit der Datenpartitionierung und parallelen Datenabfragen.

Ein sehr wichtiger Aspekt der Datenorganisation und Datenstrukturierung ist die Dimensionierung. Betrachten wir diese Problematik im Bezug auf anwachsende Dimensionszahl. Bei einem System, das eine Struktur von 3 Dimensionen besitzt, können die gesamten Durchführungsanforderungen leicht von beiden Architekturen erfüllt werden. Hat das betrachtete System die Struktur von 10 Datendimensionen, entscheidet dann die Kompilation der Daten bei der Auswahl der einzusetzenden OLAP-Architektur. Die volle 100% Datenkompilation ist realistisch nicht erreichbar. Könnte man bei einem gut organisierten Konzept mit etwa 85-90% rechnen, wäre es noch möglich, die MD-OLAP einzusetzen. Andernfalls und für weiter anwachsende Datendimensionen wäre eindeutig ROLAP zu bevorzugen.

3 Sicherheit

3.1 Sicherheitsanforderungen

Anforderungen der Computer- und Netzwerksicherheit sind:

1. ***Geheimhaltung*** (*Secrecy*) : Besagt, daß auf die Informationen in einem Computersystem nur von den berechtigten Personen zugegriffen werden darf. Unter „Zugreifen“ wird das Lesen, Ausdrucken oder ähnliche Arten der Datenenthüllung verstanden, oder einfach das Herausfinden, daß ein Objekt überhaupt existiert.

2. ***Integrität***: sieht vor, daß die Systemkomponenten, insbesondere Informationen, die in einem Computersystem aufbewahrt werden, nur von den berechtigten Personen modifiziert werden dürfen. Unter Modifikationen versteht man das Schreiben, Ändern, Löschen oder Kreieren von Daten, oder das Ändern ihres Status.

3. ***Verfügbarkeit***: fordert, daß die Informationen den autorisierten Personen zur Verfügung stehen.

3.2 Sicherheitsbedrohungen von Computersystemen

Im allgemeinen, geht der Datenfluß von der Informationsquelle (Datei, Hauptspeicher) zum Ziel (andere Datei oder Benutzer) (Abbildung 11).

Dabei existieren vier Arten von Sicherheitsbedrohungen: Unterbrechen, Abfangen, Modifizieren, Herstellen

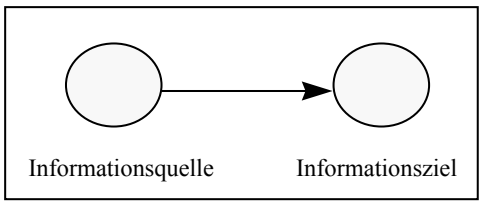

Abbildung 11: Normalfluß [STA92]

- ***Unterbrechen***

  Ein Systemteil wird vernichtet oder wird unbrauchbar. Durch die Unterbrechung wird die Verfügbarkeit bedroht. Dabei können die Hardwarekomponenten gefährdet werden (z.B. Festplattendefekt), es können die Kommunikationslinien unterbrochen werden oder das Filemanagementsystem kann außerstand gesetzt werden (Abbildung 12).

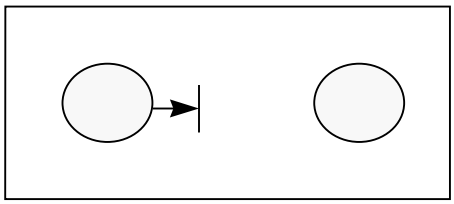

Abbildung 12: Unterbrechung [STA92]

- ***Abfangen***

  Eine unberechtigte Partei (eine Person, ein Rechner oder ein Programm) erwirbt den Zugang zu einem Systemteil. Dies ist eine Bedrohung der Geheimhaltung. Die Beispiele solcher Attacken sind das Abhören von Kommunikationslinien oder das unerlaubte Kopieren von Dateien oder Programmen (Abbildung 13).

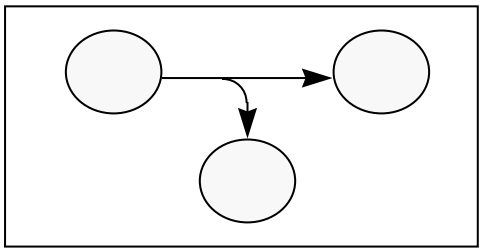

Abbildung 13: Abfangen [STA92]

- ***Modifizieren***

  Eine unberechtigte Partei erwirbt den Zugang zu einem bestimmten Systemteil und modifiziert ihn. Dies schließt das Verändern von Dateiinhalten, Verändern von Programmen, so daß sie anders als vorgesehen funktionieren, oder das Modifizieren von übertragenen Nachrichten am Netzwerk ein. Durch diese Attacke wird die Integrität gefährdet (Abbildung 14).

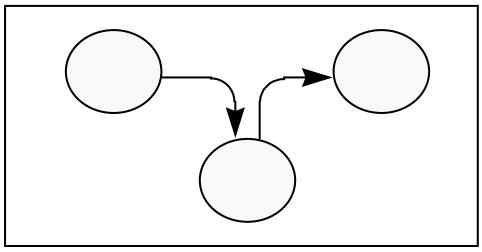

Abbildung 14: Modifizieren [STA92]

- ***Herstellen***

  Dies ist eine zusätzliche Bedrohung der Integrität. Die unberechtigte Parteien fügen die gefälschten Objekte in das System ein. Z.B. das Einsetzen von falschen Nachrichten in das Netzwerk oder das Hinzufügen von den Zusatzdaten in die Dateien (Abbildung 15).

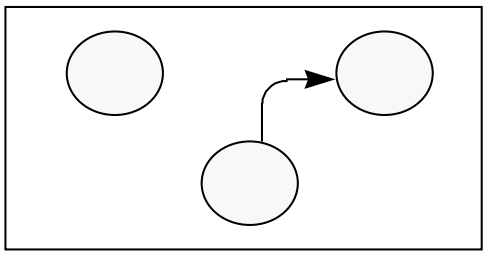

Abbildung 15: Herstellen [STA92]

Die Komponenten eines Computersystems können in 4 Gruppen unterteilt werden [STA92]:

- Hardware
- Software
- Daten
- Kommunikationslinien

und jede Gruppe wird den für sie spezifischen Bedrohungen ausgesetzt.

Hardware

Die größte Bedrohung für die Hardwareteile eines Computersystems liegt im Bereich der Verfügbarkeit. Die Hardware ist die den Attacken am meisten ausgesetzte Komponente eines Computersystems. Diese Komponente kann am schwersten automatisch kontrolliert werden.

Unter Bedrohungen von Hardware versteht man die absichtliche und die unabsichtliche Beschädigung der Hardwarekomponenten.

Um diese Bedrohungen zu vermeiden, müssen sowohl die physikalischen als auch die administrativen Maßnahmen eingesetzt werden.

Software

Die größte Bedrohung von Betriebssystemen, Anwendungen und Werkzeugen (von Software) bezieht sich auf die Verfügbarkeit, obwohl auch die Integrität und Secrecy hier eine große Rolle spielen. Software, insbesondere die Anwendungsprogramme, werden einfach gelöscht. Außerdem kann die Software so geändert oder beschädigt werden, daß sie, obwohl sie nicht gelöscht wurde, unbrauchbar wird. Um die hohe Verfügbarkeit zu sichern, sollen ständig Sicherheitskopien (backups) von den neuesten Versionen der Software erzeugt werden.

Ein größeres Problem tritt auf, wenn Programme, die noch immer laufen, anders funktionieren als sie funktionieren sollten. Diese unerwünschten Erscheinungen werden durch Viren und ähnliche Attacken verursacht.

Letztendlich ist die Secrecy von Software ein sehr wichtiger Punkt. Obwohl es gewisse Sicherheitsmaßnahmen schon gibt, kann das illegale Kopieren von Software nicht völlig kontrolliert und gestoppt werden.

Daten

Ein viel größeres Problem als die Sicherheit von Hardware und Software ist Sicherheit von Daten[6], die in einem Computersystem gespeichert werden. Dabei geht es vor allem um das Geheimhalten von Daten, aber auch um die Datenintegrität und Datenverfügbarkeit. Wenn die Daten eines Computersystems so modifiziert werden, daß sie die Integritätsanforderungen des Systems verletzen, können sehr große Schaden entstehen. Falls die Daten (absichtlich oder zufällig) vernichtet werden, wird die Anforderung der Verfügbarkeit verletzt.

Die Bedrohungen, mit denen die Sicherheitsmanager vieler Computersysteme am meisten Probleme haben, beziehen sich auf die Verletzung der Geheimhaltung. Dabei geht es am häufigsten um das unbefugte Lesen von Daten. Diese Bedrohung ist deswegen sehr gefährlich, weil man sehr schwer feststellen kann, ob die Daten von den unberechtigten Parteien gelesen wurden oder nicht. Die Daten müssen nicht unbedingt gelöscht oder geändert werden, es genügt, daß sie gelesen werden, die Schäden die manchmal dadurch entstehen, können unermeßlich sein.

Besonders kompliziert wird die Sache, wenn die Geheimhaltung von Datenbanken, die statistische, bzw. aggregierte Daten enthalten, verletzt wird. Durch das unberechtigte Lesen von aggregierten Daten wird die Sicherheit der einzelnen Daten stark gefährdet. Ein Beispiel dafür: Ein Datenbankeintrag beinhaltet die Verkaufszahlen für die Verkaufsregion „Österreich - Süd“ in Monaten Januar bis Juni. Ein anderer

[6] Files und andere Formen von Daten die von Privatpersonen, Gruppen oder Unternehmen kontrolliert werden.

Eintrag repräsentiert die Verkaufszahlen für die selbe Verkaufsregion in den Monaten Januar bis Juli. Falls man diese beiden Einträge liest und die Verkaufszahlen voneinander subtrahiert, bekommt man die Verkaufszahlen der Region „Österreich - Süd“ im Monat Juli. Dieses Problem wächst mit dem steigenden Wunsch, Daten zu kombinieren.

Kommunikationslinien und Netzwerke

Kommunikationslinien werden benutzt, um Daten zu transferieren. Man muß sich hier um die Geheimhaltung, Verfügbarkeit und um die Integrität genauso sehr wie um die Netzwerksicherheit kümmern. Die Bedrohungen, die hier vorkommen, werden in zwei Gruppen geteilt: passive und aktive Bedrohungen.

Passive Bedrohungen (Abbildung 16) beziehen sich auf das Herausfinden von Informationen, die durch die Kommunikationslinien übertragen werden. Dabei kann sich um zwei Arten von Bedrohungen handeln: Herausfinden von den Dateninhalten und Analyse des Netzverkehrs.

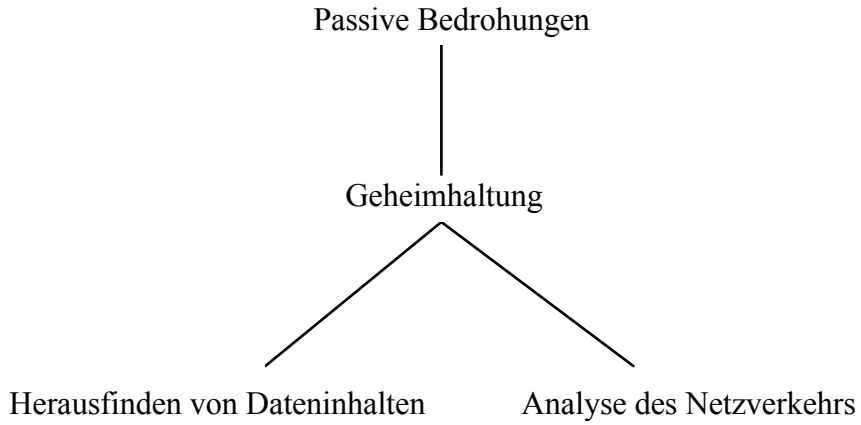

Abbildung 16: Passive Netzwerk Bedrohungen

- *Herausfinden vom Dateninhalt*: die transferierten Dateien und die Email - Messages können vertrauliche Daten einschließen. Die potentiellen Angreifer müssen am Entdecken solcher Informationen gehindert werden.

- *Analyse des Netzverkehrs:* Wenn es um das Transferieren von vertraulichen Informationen geht, wird oft irgend eine Art der Datenverdeckung, wie z.B. Verschlüsselung angewendet. Dadurch soll für den Fall, daß ein Angreifer die transferierten Daten empfängt, erzielt werden, daß die Daten für ihn unbrauchbar sind. Wenn aber der Angreifer die Kommunikationslinien lang genug analysiert, kann er den Sender und den Empfänger ermitteln und die Frequenz und die Länge der ausgetauschten Datenportionen verfolgen. Durch all diese Informationen kann er

dann feststellen,um welche Datentypen es sich handelt und bei den detaillierten Analysen, welche Daten übertragen werden.

Diese Art der Bedrohung ist sehr gefährlich, weil die enthüllten Daten nicht geändert werden müssen, wodurch die Angriffe schwierig nachzuvollziehen sind. Da die Entdeckung der Angriffe sehr mühsam ist, bleibt die Prävention als einzige Möglichkeit, diese Sicherheitsanforderungen aufrechtzuerhalten.

Unter ***aktiven Bedrohungen*** wird das Modifizieren von vorhandenen, bzw. das Kreieren von neuen Dateninhalten verstanden. Diese Bedrohungen können in drei Kategorien unterteilt werden: die Dienstablehnung, das Modifizieren von Dateninhalten und die Maskerade (Abbildung 17).

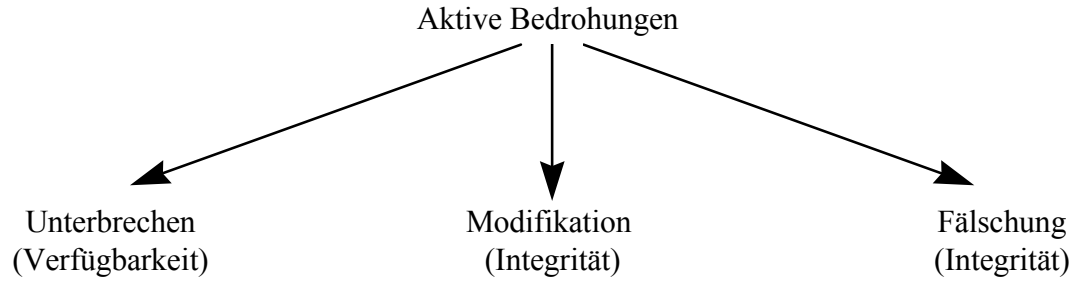

Abbildung 17: Aktive Netzwerk Bedrohungen[STA92]

- *Die Dienstablehnung*: verhindert oder verbietet die normale Nutzung oder Management von Kommunikationseinrichtungen. Diese Attacken können spezifische Ziele haben, z.B. alle Nachrichten, die für einen gewissen Empfänger bestimmt sind, zu unterdrücken u.ä.. Eine andere Form von Dienstablehnung ist das Zerbrechen des gesamten Netzwerkes, entweder durch unbrauchbar machen oder durch Belasten der Kommunikationslinien, wodurch die Netzwerkperformanz sinkt.

- *Das Modifizieren von Dateninhalten*: einige Teile der gültigen Datennachricht werden geändert, die Nachricht wird verzögert, oder umgeordnet u.ä., wodurch die ungültigen Effekte verursacht werden.

- *Die Maskerade* passiert, wenn eine Einheit vorgibt eine andere zu sein. Diese Attacke schließt oft eine der beiden anderen aktiven Attacken ein.

Es ist ziemlich schwierig, die aktive Attacke völlig zu verhindern, weil das physischen Schutz von allen Kommunikationswegen und jederzeit bedeuten würde. Deswegen ist es bei dieser Art der Sicherheitsbedrohungen das Wichtigste, diese Attacke rechtzeitig herauszufinden und möglichst rasch die normalen Funktionalitäten des Systems wiederherzustellen.

3.3 Sicherheitsplanung

Die beste Möglichkeit, alle potentiellen Sicherheitsrisiken auf ein annehmbares Niveau zu reduzieren, ist, Sicherheitsplanung schon beim Design vom Data Warehouse zuberücksichtigen. Der Zweck eines Sicherheitsplans ist, den Wert von Informationen zu schützen.

Einen Sicherheitsplan vorzubereiten bedeutet, sich vier einfache Fragen zu stellen [SILV96]:

1. Was muß geschützt werden?
2. Wer kann in das System einbrechen?
3. Welche Schwächen hat das System?
4. Was kann gemacht werden, um das System zu schützen?

Was muß geschützt werden?

Zu den Vermögen eines Computersystems zählen: Hardwarekomponenten, Software und Dateien. Der wichtigste Data Warehouse-Besitz sind die Daten, die das Warehouse enthält, seien es persönliche Informationen über Mitarbeiter, Rentabilitätstrends, regionale Abschlußanalysendaten oder z.B. Kundenlisten. Jedes hat einen Wert im Sinne seiner Verfügbarkeit, Integrität und Vertraulichkeit. Jedes erfordert einen Schutz im Verhältnis zu seinem Wert (für seinen Besitzer) und seinen potentiellen Wert (für den Konkurrenten seines Besitzers).

Es ist ziemlich einfach, das Informationsvermögen zu identifizieren, aber das Bestimmen dessen Wertes ist viel schwieriger. Eine Möglichkeit, den Wert der Informationen zu ermitteln, ist, festzustellen, was das Schaffen der Daten kosten würde. Eine andere Möglichkeit ist, den Wert im Sinne der verlorenen Produktivität, die wegen nicht vorhandener Daten verursacht wurde, zu definieren. Letztendlich kann man den Informationswert so bestimmen, in dem man abschätzt, wieviel verloren gehen würde, wenn die Informationen in falsche Hände fielen.

Sicherheit fängt mit grundlegender Kontrolle an. Darunter sind die mindest notwendigen Aktionen für den Schutz von Daten im Data Warehouse, wenn alle Computerfunktionen fehlerfrei durchgeführt werden könnten und wenn alle Menschen ehrlich sein würden, zu verstehen.

Die grundlegende Kontrolle besteht aus zwei wesentlichen Gesichtspunkten. Der erste ist Identifizierung und Authentifizierung (oft in der Sicherheitsliteratur als I&A bezeichnet). I&A bedeutet üblicherweise das Fordern des Benutzernamens und des Kennwortes, bevor einem Benutzer der Zugang zum System erlaubt wird. Der zweite Gesichtspunkt ist die Zugriffskontrolle, die (im Fall des Data Warehouse) definiert, welche Benutzergruppen in der Lage sein sollten,bestimmte Datengegenstände zu lesen.

Es ist sehr wichtig, daß ein Teil des Sicherheitsplans die individuellen Benutzer, ihre Benutzernamen und ihre Zugangsanforderungen zu den Daten dokumentiert. Es muß auch einen Weg geben, die Liste zu aktualisieren. Der Sicherheitsmanager muß benachrichtigt werden, wenn sich neue Angestellte dem Unternehmen anschließen, wenn existierende Angestellte das Unternehmen verlassen oder wenn sich Tätigkeitsbeschreibungen und Zugangsanforderungen der Angestellten verändern. Dieser Abschnitt des Sicherheitsplans sorgt dafür, daß die Zugriffsrechte (auf welche Daten ein Benutzer zugreifen kann) mit der Autorität (zu welchen Daten der Benutzer den Zugang haben soll) übereinstimmen.

Wer kann in das System einbrechen?

Als Teil einer Sicherheitsplanung, muß man voraussehen können, wer in das System einer Firma einbrechen wollte, welche Motivation er haben könnte und was seine Kräfte und Schwächen sein könnten. Die einfachste Unterteilung der potentiellen Einbrecher ist in „insiders“ und „outsiders“.

- *Profil:* ***insiders***

  Studien haben gezeigt, daß die bedeutendste Bedrohung in jeglicher Organisation von firmeneigenen Angestellten kommt. Nach der „American Society of Industrial Security“(ASIS) sind für beinah drei Viertel von den gemeldeten Sicherheitszwischenfällen Angestellte, ex-Angestellte oder andere vertrauenswürdige Parteien verantwortlich. Die Begründung ist manchmal Gier, oder Rache. Häufig gibt es aber keinen böswilligen Grund, der Angestellte ist einfach neugierig oder hält die Sicherheitsbeschränkungen für schlecht.

  Kraft:
  Die bedeutendste Kraft, die die Insiders haben ist ihr bestehender legitimer Systemzugang und ihr Wissen von inneren Verfahren.

  Schwäche:
  Ihre bedeutendste Schwäche ist die Furcht von der Disziplinarmaßnahme und dem Einkommensausfall.

- *Profil:* ***Hackers (outsiders)***

  Hackers sind vom Prestige genug motiviert. Sie möchten beweisen, daß sie intelligent und geschickt genug sind, um in ein geschütztes System einzubrechen.

  Kraft:
  Ihre bedeutendste Kräfte sind ihre technischen Sachkenntnisse und viel freie Zeit.

  Schwäche:

Ihre bedeutendste Schwäche ist die Tatsache, daß sie ihre Leistungen loben, häufig schriftlich.

- *Profil:* ***Industriespionage (outsiders)***

  Bei der Industriespionage sind die Leute durch den finanziellen Gewinn motiviert.

  Kraft:
  Ihre größte Kraft ist ihre Finanzrückendeckung.

  Schwäche:
  Ihre größte Schwäche ist die Furcht vor krimineller Verfolgung, die sie erwartet falls sie verhaftet werden sollten.

- *Profil:* ***politische Aktivisten (outsiders)***

  Eine mögliche Gruppe äußerer Angreifer sind die politischen Aktivisten. Ihr Ziel ist üblicherweise das Unterbinden des betreffenden Geschäfts oder das Erwerben potentiell peinlicher Informationen über das Zielunternehmen.

  Kraft und Schwäche:

  Ihr Eifer und ihre Hingabe sind ihre größte Kraft und ihre größte Schwäche, gleichzeitig: sie nehmen sehr große Risiken auf sich, aber die Wahrscheinlichkeit, daß sie gefaßt werden, ist auch größer als bei allen andern.

Welche Schwächen hat das System?

Die Geräte, die man in ein System einbaut, funktionieren nicht immer wie sie funktionieren sollen. Die neuen Methoden, Sicherheitsmechanismen zu umgehen oder zu vereiteln, wachsen sehr rasch. Das heißt, daß unabhängig davon wie hart man gearbeitet hat, man nie 100% sicher sein kann, daß man alle Schwachpunkte des Systems identifiziert hat.

Man soll die mögliche Angriffspunkte eines Systems dokumentieren. Angefangen von den grundlegenden Kontrollmaßnahmen, kann man drei verletzbare Stellen definieren:

- *Erweiterung von Zugriffsrechten* - erwerben von Zugriffsrechten, die nicht vom Datenmanager bewilligt wurden.

- *Umgehen von Authentifizierung* - der Angreifer gibt vor, jemand zu sein, der Zugriffsrechte auf Daten hat, auf die der Angreifer nicht zugreifen dürfte.(Dies kann

kann als Ergebnis eines gestohlenenen Kennwortes, „Trojan Horse“ Programmen oder ähnlichem geschehen.)
- *Umgehen von Identifizierung* - Erhalten von geheimen Informationen, ohne eigentlich auf dem System angemeldet zu sein. Kann z.B. durch das Abhören vom Netzwerk geschehen.

Wenn man die Data Warehouse Schwächen abschätzt, sollte man darauf aufpassen, was sie besonders anfällig macht:

- Es existieren weit verteilte, vielfache Kopien von empfindlichen Daten.
- Empfindliche Daten werden oft über unsichere Kommunikationsnetze übertragen.
- Empfindliche Daten werden auf einer Vielfalt an Hardware und Softwarearchitekturen gelagert - wobei einige nicht unbedingt stark sind.
- Das Extrahieren erzeugt und ladet zwischenliegende Dateien, die manchmal nicht so gut geschützt werden wie das Data Warehouse selbst.

Was kann gemacht werden, um das System zu schützen?

Individuelle Risiken sind die einmalige Mischung aus Bedrohungen und Schwächen, die vorher definiert wurden. Für jedes Risiko (oder Gruppe von Risiken) sollen Gegenmaßnahmen erarbeitet werden. Folgendes soll berücksichtigt werden:

1. *Prävention*
2. *Aufdeckung*
3. *Datenbeschränkungen*
4. *Recovery*
5. *Untersuchung*

Prävention *- können die Schwächen gestärkt werden? Wie kann der Einbruch gestoppt werden?*

Präventionsmaßnahmen bestehen aus solchen Dingen wie Aufbau von Kennwortkonstruktionsregeln, definieren der Schwelle, von der aus erfolglose Anmeldungsversuche unterbrochen werden, Time-outs, das Einrichten von Firewalls, das Sperren von inaktiven Accounts, chiffrieren von empfindlichen Daten. (Zum Beispiel können die Daten während der Extraktion entschlüsselt und verschlüsselt werden (für berechtigte Benutzer) als Teil des Abfrageprozesses. Der Vorteil davon ist, daß, auch wenn ein Hacker den Identifikations- und Authentifikationsmechanismus austrickst und an den Datenbankzugriffssicherungen vorbeikommt, die Daten noch nicht lesen kann.

Aufdeckung *- was kann getan werden, um den Sicherheitsmanager zu informieren, daß ein Einbruch vorgekommen ist? Können die Mechanismen zur Aufdeckung von Eindringlingen angewendet werden, um eine Warnung in Falle eines Angriffs auszulösen?*

Aufdeckung wird in erster Linie durch die Überwachung und durch das Protokollieren von Anmeldungen am Server durchgeführt. Da die Aufzeichnungsdatei schnell zu groß wird, um gemanagt werden zu können, soll man im voraus definieren, wie eine verdächtige Folge von Anmeldungssequenzen aussieht (z. B. Anmeldung nach der gewöhnlichen Arbeitszeit, mehrmalige Verweigerungen vom Zugriff auf das gleiche Ressource etc.).

Datenbeschränkungen *- wie können die Ressourcen für den Fall eines erfolgreichen Angriffes geteilt und konfiguriert werden, so daß die Schäden minimiert werden?*

Beim Beschränken von Daten muß man vernünftig über die Daten, die in einem Data Warehouse gelagert werden, entscheiden. Man muß sich im Klaren sein, zumindest für die wirklich empfindlichen Daten, wie weit man das Ausbreiten von Daten betreiben will. Falls man den Vorteil der Lagerung der wichtigen (und empfindlichen) Daten ausnutzen will, muß man das Risiko eingehen, daß die Daten vielleicht enthüllt werden, was negative finanzielle Auswirkungen zur Folge haben könnte.

Recovery *- wie kann das System rasch zu einem normalen betrieblichen Zustand wieder hergestellt werden, wenn der Einbrecher ein oder mehrere Ressourcen beschädigt oder verändert hat?*

Recovery bezieht sich darauf, beschädigte oder gelöschte Dateien wieder herzustellen, damit das System weiter benutzt werden kann. Dies wird vollbracht, in dem Sicherheitskopien von den systemrelevanten Dateien erstellt werden. Diese werden benötigt, um Recovery innerhalb der kürzesten Zeit durchführen zu können. Für die enthüllten Geheimnisse gibt es natürlich keine Recovery. Sobald die Datenvertraulichkeit verloren geht, gibt es keine Möglichkeit sie zurückzubekommen.

Untersuchung *- wie können die verantwortlichen Parteien identifiziert und verfolgt werden?*

Die letzte Schutzschicht besteht aus all den Aktivitäten, die dazu beitragen, Personen, die für die Sicherheitsbrüche verantwortlich sind, zu identifizieren und zu verfolgen. Zwei Kernbestandteile davon sind individuelle Benutzernummern, für die die Kennworte unter keinen Umständen geteilt werden dürfen und Benutzerübereinstimmungen, die die Verhaltungsregeln im System bestimmen und die Strafen für den Verstoß erklären. Dies bezieht sich auf die Insiders. Die Einbrecher die von außerhalb des Unternehmens kommen (outsiders) sind schwieriger zu verfolgen, aber es ist möglich. [SILV96]

3.4 Kontrollmaßnahmen für den sicheren Zugriff

Die Kontrollmaßnahmen, mit denen den Zugriff auf Datenverarbeitungssysteme kontrolliert wird, werden in zwei Kategorien unterteilt: die Benutzerorientierte und die Datenorientierte [STA92]:

3.4.1 Benutzerorientierte Zugriffskontrolle

Die gewöhnlichste Art der benutzerorientierten Zugriffskontrolle berührt auf dem "Benutzername-Paßwort" - Prinzip. Ein Benutzer, der sich an dem System anmeldet, wird nur dann zugelassen, wenn sein Benutzername (User ID) dem System bekannt ist, und wenn der Benutzer das diesem Namen zugeordnete Paßwort kennt. Diese Art der Zugriffskontrolle ist sehr unverläßlich, weil die Benutzer ihre Paßwörter oft vergessen oder sie absichtlich oder unabsichtlich weitergeben. Die Hackers werden immer fähiger, wenn es um das Erraten der Benutzernamen und Paßwörter geht. Außerdem sind die Dateien, in denen diese vertrauliche Daten gespeichert werden, sehr oft das Ziel der Attacke.

Verschiedene Sicherheitsmaßnahmen können angewendet werden, um die Sicherheit des Paßwortkonzeptes zu verbessern:

1. Es muß eine hohe Anzahl von Paßwortkombinationen möglich sein. Dies soll es dem Angreifer schwer machen, die Paßwörter durch Ausprobieren oder durch das Einsetzen von Computerprogrammen zu erraten. Es hat sich als nützlich erwiesen, die Paßwörter auf Buchstaben zu reduzieren und aussprechbare Kombinationen zu benutzen, so daß sie sich Benutzer leicht merken können und sie nirgendwo aufschreiben müssen. (Mit 5 Buchstaben kann man 6 Millionen Paßwörter erstellen, was das Erraten sehr schwer macht.)

2. Eine automatische Unterbrechung der Verbindung muß gesichert werden, wenn man sich mehrmals hintereinander mit einem falschen Paßwort an das System anmelden will. Gewöhnlich werden 3 bis 5 Versuche zugelassen. Dadurch wird ein Angreifer gezwungen, die Verbindung zum System neu aufzubauen und wieder zu versuchen, wodurch das Einbrechen ins System verzögert wird.

3. Die Betriebssysteme sollten die fehlerhaften Login-Versuche oder ähnliche sicherheitsbedrohliche Aktivitäten protokollieren und melden. Dies würde die Versuche unberechtigter Personen, vertrauliche Anwendungen auszuführen, einschließen.

Ein weiterer, sehr wichtiger Aspekt ist die Frage , wer berechtigt sein soll, die Paßwörter auszustellen. Falls die Benutzer selbst die Paßwörter wählen dürfen, ist es einfacher, sie zu „knacken", weil die Leute dazu neigen, persönliche Daten wie Namen, Datum usw. zu benutzen. Falls die Paßwörter aber vom Systemadministrator vorgegeben werden, ist die Wahrscheinlichkeit größer, daß die Benutzer sie (aus Angst sie vergessen zu können) aufschreiben. Dies ist aber viel sicherheitsgefährdender, als wenn die Benutzer die Freiheit haben, ihre Paßwörter selber auszusuchen.

Bei den verteilten Systemen kann die benutzerorientierte Zugriffskontrolle entweder **zentralisiert** oder **dezentralisiert** erfolgen.

In erstem Fall wird ein Anmeldungsdienst vom Netzwerk angeboten, um zu überprüfen, wer das Netz benutzen darf und mit wem er kommunizieren darf. Im zweiten Fall ist das Netz für den Benutzer transparent und die Anmeldungsprozedur wird am Zielserver durchgeführt. Die Sicherheitsmaßnahmen bei der Paßwortübertragung müssen getroffen werden.

Ein weiteres Ziel der Angriffe sind die **Paßwortdateien**. In diesen Dateien werden alle Benutzernamen eines Computersystems und deren Paßwörter gespeichert. Falls ein Einbrecher diese Datei findet, kann er alle Informationen ablesen. Eine Möglichkeit dies zu verhindern ist, die Paßwörter zu verschlüsseln (Abbildung 18).

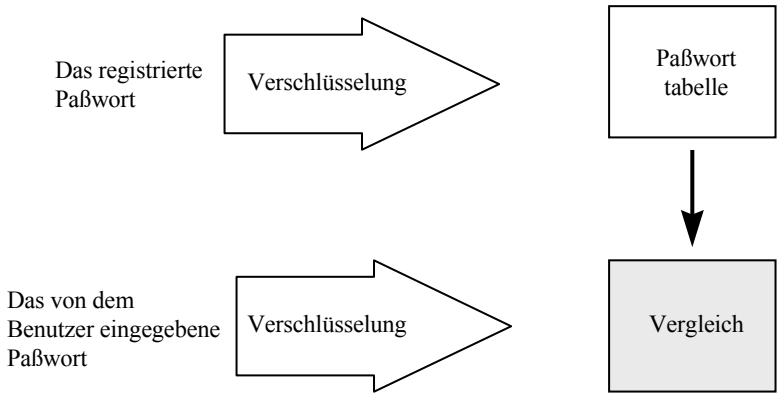

Abbildung 18: One-way Verschlüsselung von Paßwörter [STA92]

Sollte der Angreifer die Paßwortdatei finden und lesen, könnte er in diesem Fall nicht viel mit den verschlüsselten Daten anfangen.

Außer dem „Benutzername-Paßwort" - Prinzip werden für die Identifikation von Anwendern auch weitere Konzepte vorgeschlagen. Einige davon sind „Fingerabdruck", „Stimmenabdruck" *(voiceprints)* und „Analyse der Handgeometrie". Diese Methoden sind noch effizienter und sicherer, kosten aber viel mehr und werden deswegen sehr selten eingesetzt.

3.4.2 Datenorientierte Zugriffskontrolle

Systeme, die in ihren Datenbanken vertrauliche Daten einschließen, können sich auf die benutzerorientierte Zugriffskontrolle nicht verlassen. Wenn ein Benutzer von diesen an das System zugelassen wird, ist es noch nicht sicher, daß er nur auf die für

ihn zugelassene Daten zugreifen wird. DBMS muß für den sicheren Zugriff auf Daten bzw. Datenblöcke in einer Datenbank sorgen[7].

Das allgemeine Modell der datenorientierten Zugriffskontrolle basiert auf der „**access matrix**“ (Zugriffsmatrix) (Tabelle 6).

	Program 1	. . .	Segment A	Segment B
Process 1	Read Execute		Read Write	
Process 2				Read
. . .				

Tabelle 6 : Access matrix [STA92]

Die wichtigste Elemente dieses Models sind Subjekt, Objekt und Zugriffsrechte.

- ***Subjekt*** ist eine Einheit (ein Benutzer oder ein Prozeß), die in der Lage ist auf Objekte zuzugreifen.
- ***Objekt*** ist alles (Dateien, Teile der Dateien, Programme, Speicherplatzsegmente), worauf der Zugriff kontrolliert wird.
- ***Zugriffsrechte*** beschreiben, wie ein Subjekt auf ein Objekt zugreifen darf (z.B. Lesen, Schreiben, Ausführen).

Eine Achse der access matrix beinhaltet alle Subjekte die auf Daten zugreifen dürfen, die andere listet die Objekte auf. Die niedrigste Detaillierungsstufe bei Objekten sind die einzelne Datenfelder, die höchste sind die Dateiteile, Dateien oder sogar die ganze Datenbank. Die Zellen des Matrix besagen, wie die Zugriffsrechte eines Subjekts auf ein Objekt ausschauen.

Die access matrix ist normalerweise spärlich und wird implementiert, nachdem sie auf eine der zwei folgenden Arten zerlegt worden ist.

Falls sie in Spalten geteilt wird, spricht man von einer Liste der Zugriffskontrolle *(access control lists*) (Tabelle 7). Hier wird für jeden Benutzer eine Liste seiner Zugriffsrechte erstellt. Falls ein Benutzer in der Liste nicht erscheint, werden im die Grundrechte des Systems zugeteilt.

Access Control List for Program 1
Process1 (Read, Execute)

[7] Z.B. kann das Computersystem eines Unternehmens so organisiert werden, daß alle Angestellte der Personalabteilung auf die Personaldaten zugreifen dürfen, aber nur einige, berechtigte die Gehaltsdaten lesen dürfen.

Access Control List for Segment A Process1 (Read, Write)
Access Control List for Segment B Process2 (Read)

Tabelle 7: Access control List [STA92]

Falls das Matrix in Zeilen geteilt wird, geht es um die Fähigkeitstickets (*capability tickets*) (Tabelle 8).

Capability List for Process1: Program1 (Read, Execute) SegmentA (Read, Write)
Capability List for Process2: SegmentB (Read)

Tabelle 8: Capability list [STA92]

Wenn man das „Access matrix“ - Modell erweitert, um es inhaltsabhängig zu gestalten, sollte man die Zugriffsrechte um die Prädikate[8] erweitern.
Ein Eintrag (eine Zugriffsregel) bei so einem Modell würde beispielsweise folgendermaßen aussehen:

Subjekt: Sekretärin
Objekt: Mitarbeiter
Zugriffsrecht: Lesen
Prädikat: Gehalt < 100

Um die Zugriffsrechte zu kontrollieren, muß eine Richtigkeitsüberprüfung durchgeführt werden. Ein mögliches Überprüfungsmodell ist in Abbildung 19 gezeigt.

Falls gewisse Dateneinheiten vertrauliche Informationen beinhalten und nur von bestimmten Personen gelesen werden dürfen, wird bei der Netzwerkübertragung von solchen Daten ein Kodierungsverfahren eingesetzt. Normalerweise ist diese Art der Zugriffskontrolle dezentralisiert. Das heißt, daß die Zugriffskontrolle bei dem Host-DBMS durchgeführt wird.

[8] Ein Prädikat ist eine Aussage die die Angehörige einer Menge definiert, in dem sie die Angehörigkeitsbedingungen beschreiben oder in dem sie die Angehörige aufzählen.

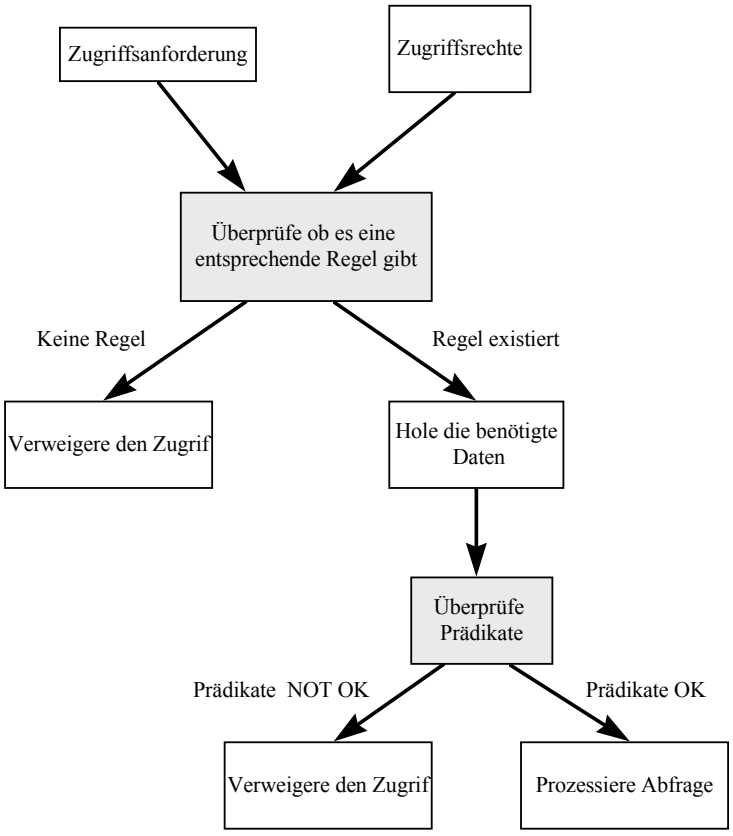

Abbildung 19: Das Zugriffsrechte-Überprüfungsmodel [FERN80]

3.4.3 Datenbanksicherheit

Die Information wird zu einer immer wichtigeren Ressource jeden Unternehmens, sei es industrielle, wirtschaftliche oder zivile Information. Die Prozesse werden automatisiert, sowohl grundlegende operationelle Funktionalitäten als auch wichtige Analyseprozesse wie z.B. Trenderkennung, finanzielle Analysen, Überprüfungen von Verkaufsstrategien usw. Um solche komplizierte Vorgänge durchzuführen und um die eigene Produktivität immer wieder zu untersuchen und zu verbessern, brauchen die Unternehmen immer mehr Informationen. Die Datenmengen werden sehr groß und schwer zu verarbeiten. Zunehmend werden die Data Warehouse-Konzepte implementiert. Da ein Data Warehouse große Datenmenge einschließt, wird ihre Sicherheit für jedes Unternehmen von essentieller Bedeutung. Das erfolgreiche Funktionieren eines Unternehmens wird durch folgende Anforderungen unterstützt:

1. Vertrauliche Informationen stehen nur den berechtigten Personen zur Verfügung. Dadurch werden den Anforderungen der Geheimhaltung Genüge getan und die Unternehmensgeheimnisse beschützt.

2. Die Informationen werden vor versehentlicher oder vorsätzlicher Zerstörung oder Korruption geschützt.

Diese Anforderungen sind Angelegenheit der **Datenbanksicherheit**.

Die Informationen, die in einem Data Warehouse gespeichert werden, sind von entscheidender Bedeutung für das Funktionieren einer Organisation. Das Zerstören oder die Beschädigung von aufbewahrten Daten kann die Organisation paralysieren und unermeßliche Schäden anrichten. Z.B., wenn eine Krankenhausdatenbank angegriffen wird, und die Daten beschädigt werden, können als Ergebnis die Patienten falsche Medikamente bekommen. Werden die Militärdaten korrumpiert, kann die nationale Sicherheit gefährdet werden. Fehlerhafte Daten in einer Fluggesellschaftsdatenbank können die Sicherheit der Fluggäste bedrohen.
Die kritischen Informationen können aber identifiziert werden und spezielle Sicherheitsmaßnahmen können angewendet werden, um sie vor unberechtigtem Zugriff zu schützen und ihre Genauigkeit und Verfügbarkeit zu garantieren.

Normalerweise beinhaltet eine Datenbank Daten verschiedener Wichtigkeitsgraden und verschiedener Empfindlichkeitsstufen. Diese Daten werden zwischen Benutzern mit verschiedenen Zugriffsrechten und verschiedenen Verantwortungen geteilt. Deswegen ist es sehr wichtig, die Benutzer einer Datenbank nur auf die, für Durchführung ihrer Tätigkeit notwendigen Datenbankteile zu beschränken. Die Kontrollmaßnahmen bei den Datenveränderungen müssen verschärft werden, so daß die Integritätsanforderungen aufrecht erhalten bleiben.

Wenn die Werte der Datenbankeinträge geändert werden, werden meist die alten Datenwerte vernichtet. Deswegen werden intelligente Verfahren für das Wiederherstellen der Datenbankinhalte notwendig, wenn das System aus irgend einem Grund versagen sollte.

Viele der heutigen Datenbanken sind „on-line“, das heißt, daß auf sie von mehreren Benutzern und von mehreren Stellen gleichzeitig zugegriffen wird. Um die Integrität und die Konsistenz der Datenbankeinträge zu garantieren, muß das Sicherheitsmodell einer Datenbank sorgfältig geplant werden.

3.5 Datenbanksicherheitsmodelle

Es existieren verschiedene Sicherheitsmodelle und -techniken, die die Datenbanksicherheit garantieren sollen. Die wichtigsten sind:

- Discretionary Security Models
- Mandatory Security Models
- Adapted Mandatory Security Models

Discretionary Security Models werden aus vier wichtigen Teilen aufgebaut:
- Sicherheitsobjekte O
- Sicherheitssubjekte S

- Menge der Zugriffsrechte T, die beschreiben, ob und auf welche Art ein Subjekt auf ein Objekt zugreifen darf
- Menge von Prädikaten P, mit denen inhaltsabhängige Regeln definiert werden können

In diesem Modell wird die „access matix“ eingesetzt, um die Beziehungen zwischen Subjekten und Objekten zu definieren.

Mandatory Security Models kontrollieren sowohl den Datenzugriff als auch den Datenfluß innerhalb eines Systems. Hier werden den Sicherheitssubjekten und den Sicherheitsobjekten gewisse Sicherheitsstufen (Labels) zugeteilt. Durch das Vergleichen von Labels für Subjekte und Objekte wird festgestellt, ob ein Subjekt den Zugriff auf ein Objekt hat und wenn er ihn hat, welche Rechte er besitzt.

Adapted Mandatory Security Models gehören zu der Gruppe der rollenorientierten Sicherheitsmodelle. Diese Modelle setzen voraus, daß jeder Benutzer eine gewisse Rolle in einer Organisation spielt. Aufgrund ihrer Rolle sind die Benutzer berechtigt, bestimmte Datenbankoperationen auf bestimmten Daten durchzuführen.

3.5.1 OLAP / Data Warehouse Sicherheitsmodell

Das Sicherheitsproblem eines Data Warehouses kann als ein Problem von DBMS, mit dem das Data Warehouse gemanagt wird, gesehen werden. Die DBMSs haben ihre eigenenSicherheitsmaßnahmen; diese sind aber *View*- orientiert und nicht für die Entscheidungssysteme (decision support systems - DSS) gedacht. In viewbasierten Systemen definiert der Data Warehouse-Administrator, welche Daten von einem vordefinierten *View* gesehen werden dürfen. Die Annahme von diesem Konzept ist, daß der Datenbankadministrator weiß, was mit den Daten gemacht wird, nachdem auf sie zugegriffen wurde.
In DSS oder OLAP Umgebung kann man kontrollieren, was der Benutzer mit den Daten, die er vom Data Warehouse geholt hat, unternehmen kann.

Ein weiteres Problem bei *View*-orientierter DBMS-Sicherheit in der Data Warehouse-Umgebung ist, daß sie leicht umgangen werden kann. Ein Endbenutzer muß kaum eine direkte “Ausladung” des Disks machen und die Daten umformatieren, um herauszufinden, wie die Inhalte aussehen. *Views* können eine Interpretation oder Aggregation von Daten, sobald sie außerhalb von DBMS - Kontrolle passiert, nicht verhindern.

Das Sicherheitsmodell für Data Warehouse [SICH97] sollte alle Merkmale des OLAP-Konzepts unterstützen. Die Definition von Sicherheitsbeschränkungen, die auf der Granularitätsstufen basieren, ist für das Data Warehouse nicht ausreichend. Es muß so erweitert werden, daß es auch die Hierarchien unterstützt.
Betrachten wir Beispielsweise die Dimension „Zeit“ und ihre Hierarchien : *Tag* → *Woche* → *Monat* → *Jahr*. Nur das Definieren von Zugriffsrechten für die

Granularitätsniveaus werden (wegen Aggregationen) für die Zugriffskontrolle nicht ausreichen. Sollte der Benutzer die Zugriffsrechte für die monatlichen Daten haben und diese nicht vorkalkuliert sein, wird der Benutzer nicht in der Lage sein die gesuchten Daten zu bekommen. Dies passiert deswegen, weil er nicht authorisiert ist, auf die Tagesdaten zuzugreifen, um diese zu summieren. In OLAP Umgebung sollte auf die aggregierte Daten zugegriffen werden dürfen, auch wenn man keinen Zugang zu den Basisdaten hat.

In diesem OLAP / DWH Sicherheitsmodell werden die Sicherheitssubjekte durch Rollen repräsentiert. Deswegen können die Zugriffsrechte nur den Rollen zugeordnet werden. Eine Rolle beschreibt einen Aufgabenbereich (unabhängig davon wer die Aufgabe konkret ausführt) und soll nur die Befugnisse haben, die für das Durchführen dieser Rolle notwendig sind.
Jedem Benutzer, jedem Mitarbeiter einer Organisation wird mindestens eine Rolle zugeordnet. Ein Benutzer kann auch mehrerenRollen spielen, aber immer nur eine zu einem Zeitpunkt, wodurch verschiedene Befugniskonflikte verhindert werden können.

Sicherheitsobjekte sind die passiven Gegenstände eines Sicherheitssystems, die die zu schützenden Informationen enthalten. In einer OLAP-Umgebung sind diese die Dimensionen und Facts und ihre Attribute. Jedes Sicherheitssubjekt ist befugt, auf einem Sicherheitsobjekt gewisse Aktionen durchzuführen. Diese Aktionen nennt man Zugriffstypen. Bei meisten OLAP-Daten haben die Endbenutzer den Lesezugriff.

Die Rechte werden beim OLAP als Regeln dargestellt. In diesem Sicherheitsmodell wird das „Discretionary access controls“ (DAC) Modell benutzt. Wie schon erwähnt, basiert dieses Modell auf einer Sammlung von Begriffen, einschließlich einer Gruppe von Sicherheitssubjekten (S), einer Gruppe von Zugriffstypen (A) und eine Gruppe von Sicherheitsobjekten (O). In allgemeinen, ist eine ***Sicherheitsregel*** ein Quadrupel (s,a,o,p), wo Subjekt *s* für den Zugriff auf das Objekt *o* den Zugriffstyp *a* hat und das Ganze in den von Prädikat *p* definierten Rahmen passiert. Die Prädikate werden benutzt, um den Zugang auf Sicherheitsobjekte zusätzlich zu beschränken.

In diesem Model werden die Sicherheitsbeschränkungen mittels dreier verschiedener Prädikate ausgedrückt [SICH97]:

1. **Einfaches Prädikat (SP)**[9]**:** SP(S,A,O), wo S das Sicherheitssubjekt, A der Zugriffstyp und O das Sicherheitsobjekt ist.

2. **Einfaches Attributprädikat (SaP)**[10]**:** SaP(S, A, O, Attr.), wo S das Sicherheitssubjekt, A der Zugriffstyp, O das Sicherheitsobjekt und Attr. ein Attribut des Sicherheitsobjektes ist.

[9] SP steht für „Simple predicate“
[10] SaP steht für „Simple attribute predicate“

3. **Wertbasiertes Attributprädikat (VBaP)**[11]**:** VBaP(S, A, O, Attr., Theta, V), wo S das Sicherheitssubjekt, A der Zugriffstyp, O das Sicherheitsobjekt, Attr. ein Attribut des Sicherheitsobjektes, Theta der Vergleichsoperator, und V der Vergleichswert ist.

Um die Zugriffsrechte einer Rolle zuzuordnen, kann man jede Kombination von oben beschriebenen Regeldefinitionen benutzen. Die Liste dieser Regeln definiert eine Untermenge des OLAP-Datenwürfels auf welche eine Rolle (Sicherheitssubjekt) Zugriffsrechte hat. Die Regeldefinition kann vom Systemadministrator oder vom Besitzer des Sicherheitsobjektes durchgeführt werden. Der abgeleitete Datenwürfel dieser Rolle hat seine eigenen Dimensionen und Dimensionshierarchien. Z.B., falls wir eine Rolle so begrenzen, daß sie in der Dimension „Zeit" nur auf Tages- und Jahresdaten zugreifen darf, dann sieht für diese Rolle die „Zeit" Dimension nicht mehr *Tag → Woche → Monat → Jahr* sondern *Tag → Jahr.*

Um die Sicherheitsanforderungen, die mittels der Regeln definiert wurden, zu implementieren, wird eine strukturierte Dekomposition von Sicherheitsobjekten in einzelnen Fragmenten durchgeführt. Die Dekomposition basiert auf der Regelstruktur und resultiert in einer Menge von Fragmenten. Das Zerlegen passiert, indem die vertikale, horizontale oder abgeleitete horizontale Fragmentierung benutzt wird.

- ***Vertikale Fragmentierung (vf)*** ist die Projektion von einem relationalen Schema (RS) in Untermengen seiner Attribute. Um die Fragmentierung verlustfrei durchzuführen, bleibt der Schlüssel in jedem vertikalen Fragment vorhanden.

- ***Horizontale Fragmentierung (hf)*** teilt ein relationales Schema in getrennte Fragmente, basierend auf den Prädikaten, die auf dem RS definiert sind. Die Prädikate werden als eine boolesche Kombination von Bedingungen ausgedrückt, wobei jede Bedingung ein einfacher Vergleich ist, der als wahr oder falsch bewertet werden kann. Ein Attribut, anhand dem die *hf* durchgeführt wird, wird *Selektionsattribut* genannt.

- ***Abgeleitete horizontale Fragmentierung (dhf)***[12] teilt ein relationales Schema RSi, in dem sie auf RSi das gleiche Teilungskriterium anwendet, als schon bei RSj (i != j) angewandt war. Im Fall von *dhf* bilden die ausgewählte Attribute keine Untermenge von der RSi-Attributmenge. Um die *dhf* auszuführen, muß eine Beziehung zwischen RSi und RSj existieren [PERN94], [CAS92].

Ein an OLAP Daten definierter Unterwürfel Ci (Ci $\in$ V) stellt ein Gebiet des Datenwürfels dar, zu dem eine entsprechende Rolle den Zugang hat.
Sei F ein Fragment, für das gilt: F = Ci $\cap$ Cj. Dann repräsentiert F einen Teil des Datenwürfels, zu dem zwei Rollen gemeinsam Zugang haben. Falls F = Ci \ Cj sein sollte, dann darf auf F nur die Rolle, die Datenwürfel Ci als ihre Datenwürfelschnittstelle hat, zugreifen. F repräsentiert die Daten, die im Datenwürfel Cj nicht enthalten sind. Deswegen darf der entsprechenden Rolle kein Zugang zu F gewährleistet werden.

[11] VBaP steht für „Value based attribute predicate"
[12] dhf steht für „derived horizontal fragmentation"

Das vorgeschlagene Sicherheitskonzept wird an einem Beispiel illustriert. Zuerst werden die Rollen definiert. Die Zugriffsrechte jeder Rolle werden durch Regeln ausgedrückt. Anschließend wird das Fragmentierungsverfahren durchgeführt.

Betrachten wir ein Unternehmen mit einem Data Warehouse, das die Verkaufszahlen für verschiedene Produkte, in verschiedenen Regionen und Zeitintervallen erfaßt. (Dies ist eine etwas vereinfachte Version von dem im Kapitel 1.11.1 vorgestellten Data Warehouse). Dieses Data Warehouse hat vier Dimensionen und eine Fact-Tabelle. Die Dimensionen und ihre Hierarchien sehen folgendermaßen aus (die niedrigste Granularitätsstufe wird zu erst dargestellt):

Time : *day → week → month → year*
Store *: store → city → district → region*
Product *: product → subcategory → category → department*
Promotion *: promotion → promotion_group*

Die Fact-Tabelle heißt ***Sales*** und hat folgende Attribute: *TimeID, StoreID, ProductID, PromotionID, dollar_salles, dollar_costs* und *customer_count.*

Rollendefinition

- ***Rolle 1****:* Zu dieser Rolle gehören die Mitarbeiter, die jährliche Abrechnungen durchführen. Da es aber um eine große Firma geht, werden die Abrechnungen nach Regionen und Produktabteilungen geteilt ausgeführt. Die Mitarbeiter, die die Rolle1 ausüben, sind für die Region Ost verantwortlich.
  (Zusammengefaßt, dürfen diese Mitarbeiter in der Dimension „Time“ auf jährliche Daten, in Dimension „Store“ auf Region=”Ost” und in Dimension „Produkt“ auf department-Daten zugreifen. Der Zugriff auf die Dimension „Promotion“ wird nicht begrenzt.)

- ***Rolle 2:*** Die Mitarbeiter, die diese Rolle ausüben, verarbeiten die Jahresverkaufsdaten für die westliche Region. Dabei soll ihnen der Zugriff auf Daten verweigert werden, falls die dollar_sales > 1000$ sind.

- ***Rolle 3:*** Diese Rolle ist für die Werbeabteilung vorgesehen. Die Mitarbeiter die sie ausführen, brauchen den Zugriff auf die Produkte der Kategorie „Getränke“. Sie interessieren sich außerdem für den Typ der Werbeaktion (promotion_group). Dabei benötigen sie nur die Daten der Werbeaktionen die Montags gültig sind. Eine weitere Beschränkung bezüglich der Daten in der Fact-Tabelle ist vorhanden: diese Rolle darf nur die Daten sehen, für die die dollar_costs nicht mehr als 500 $ betragen.

- ***Rolle 4:*** Zu dieser Rolle gehören die Mitarbeiter der Managementabteilung. Für sie soll es keine Beschränkungen geben, das heißt, daß die Benutzer, die diese Rolle ausführen, den Zugang zu allen Daten im Data Warehouse haben.

Die Regeln für die Rollen sehen wie folgt aus:

Rolle 1:

⇒ SP(Rolle1, Read, Sales)
⇒ SaP(Rolle1, Read, Time, year)
⇒ VBaP(Rolle1, Read, Store, region, „=“, „Ost“)
⇒ SaP(Rolle1, Read, Product, department)
⇒ SP(Rolle1, Read, Promotion)

Rolle 2:

⇒ VBaP(Rolle2, Read, Sales, dollar_sales[13], „<“, 1000)
⇒ SaP(Rolle2, Read, Time, year)
⇒ VBaP(Rolle2, Read, Store, region, „=“, „West“)
⇒ SP(Rolle2, Read, Product)
⇒ SP(Rolle2, Read, Promotion)

Rolle 3:

⇒ VBaP(Rolle3, Read, Sales, dollar_costs[14], „<“, 500)
⇒ VBaP(Rolle3, Read, Time, day, „=“, „Monday“)
⇒ SP(Rolle3, Read, Store)
⇒ VBaP(Rolle3, Read, Product, category, „=“, „Drinks“)
⇒ SaP(Rolle3, Read, Promotion, promotion_group)

Rolle 4:

⇒ SP(Rolle4, Read, Sales)
⇒ SP(Rolle4, Read, Time)
⇒ SP(Rolle4, Read, Store)
⇒ SP(Rolle4, Read, Product)
⇒ SP(Rolle4, Read, Promotion)

Fragmentierung

Nachdem diese Regeln definiert sind, wird die Dekompositionsprozedur durchgeführt. Die Anzahl der resultierenden Fragmente hängt von der Anzahl der horizontalen und vertikalen Fragmentierungen ab. Um die Anzahl der Fragmente zu minimieren, soll die vertikale Fragmentierung zuerst durchgeführt werden.
Durch die Fragmentierung entsteht ein Fragmentierungsbaum. Die Wurzel dieses Baumes repräsentiert das Data Warehouse. Die Zwischenknoten stellen die Zwi-

[13] Wegen des Platzmangels werden dollar_sales im Baum als $sales geschrieben
[14] Wegen des Platzmangels werden dollar_costs im Baum als $costs geschrieben

schenfragmente dar, die weiter geteilt werden und die Blätter repräsentieren die Endfragmente.

Erklärung des Fragmentationsschemas (Abbildung 20):
Im ersten Schritt wird das Data Warehouse vertikal fragmentiert. Dadurch entstehen 5 Zwischenknoten: TIME, SALES, STORE, PRODUCT und PROMOTION. (Die vertikale Fragmentierung wird mit der dicken Linie dargestellt). In zweitem Schritt wird mit der Fragmentierung für die Rolle1 begonnen. Zuerst wird der Zwischenknoten TIME horizontal fragmentiert in zwei Teile (dünne ausgefüllte Linie): IF1[15] und IF2. IF1 beinhaltet die TIME - Daten, die sich auf Jahre beziehen, und IF2 alle restliche Daten. Da die Rolle1 nur auf die jährlichen Daten zugreifen darf, wird für die Rolle nur der Zwischenknoten IF1 von weiterem Interesse sein. Die nächste Fragmentierung, die gebraucht wird, ist die abgeleitete horizontale (durchbrochene Linie). Dies wird anhand des Attributs „region“ durchgeführt. (Da das Attribut region der Dimension Store und nicht der Dimension TIME angehört, geht es hier um die abgeleitete horizontale Fragmentierung.). Alle TIME - Daten die sich auf die Region „Ost“ beziehen, werden durch den Zwischenknoten IF11 präsentiert. Alle anderen werden durch den Zwischenknoten IF12 dargestellt. Der letzte Fragmentierungsschritt für die Rolle1 und die Dimension TIME stellt wieder eine abgeleitete horizontale Fragmentierung dar. Hier hängt das Fragmentierungsergebnis von dem Attribut „department“ der Dimension PRODUCT ab. Diese Fragmentierung resultiert in zwei Fragmenten, F1 und F2, die die Blätter des Baumes darstellen. Im Fragment F1 der Dimension TIME des Data Warehouses stehen dann die jährlichen Daten, die sich auf die Region Ost und auf die Aggregationsstufe department der Dimension PRODUCT beziehen. Genau diese Anforderungen hat die Rolle1 ja gestellt. Das heißt, daß die Rolle1 den Zugang zum Fragment F1 haben muß.

Man wendet jetzt dieses Fragmentierungsverfahren für die Rolle1 auch auf Zwischenfragmente STORE und PRODUCT an. Das ganze Verfahren für die Rolle1 resultiert in 3 Fragmenten (F1, F19 und F25), auf die die Rolle1 zugreifen darf. Da es keine Beschränkungen im Bezug auf Dimension PROMOTION und Fact Sales gegeben hat, darf die Rolle1 auch auf alle aus PROMOTION und SALES abgeleitete Fragmente zugreifen.

Wenn man den gleichen Vorgang jetzt für die Rolle2 macht, und wieder bei dem Zwischenknoten TIME beginnt, merkt man, daß diese beiden Rollen einiges gemeinsam haben. Zum Beispiel, beiden auf jährliche Daten zugreifen dürfen. Da

[15] IF steht für intermediate fragment

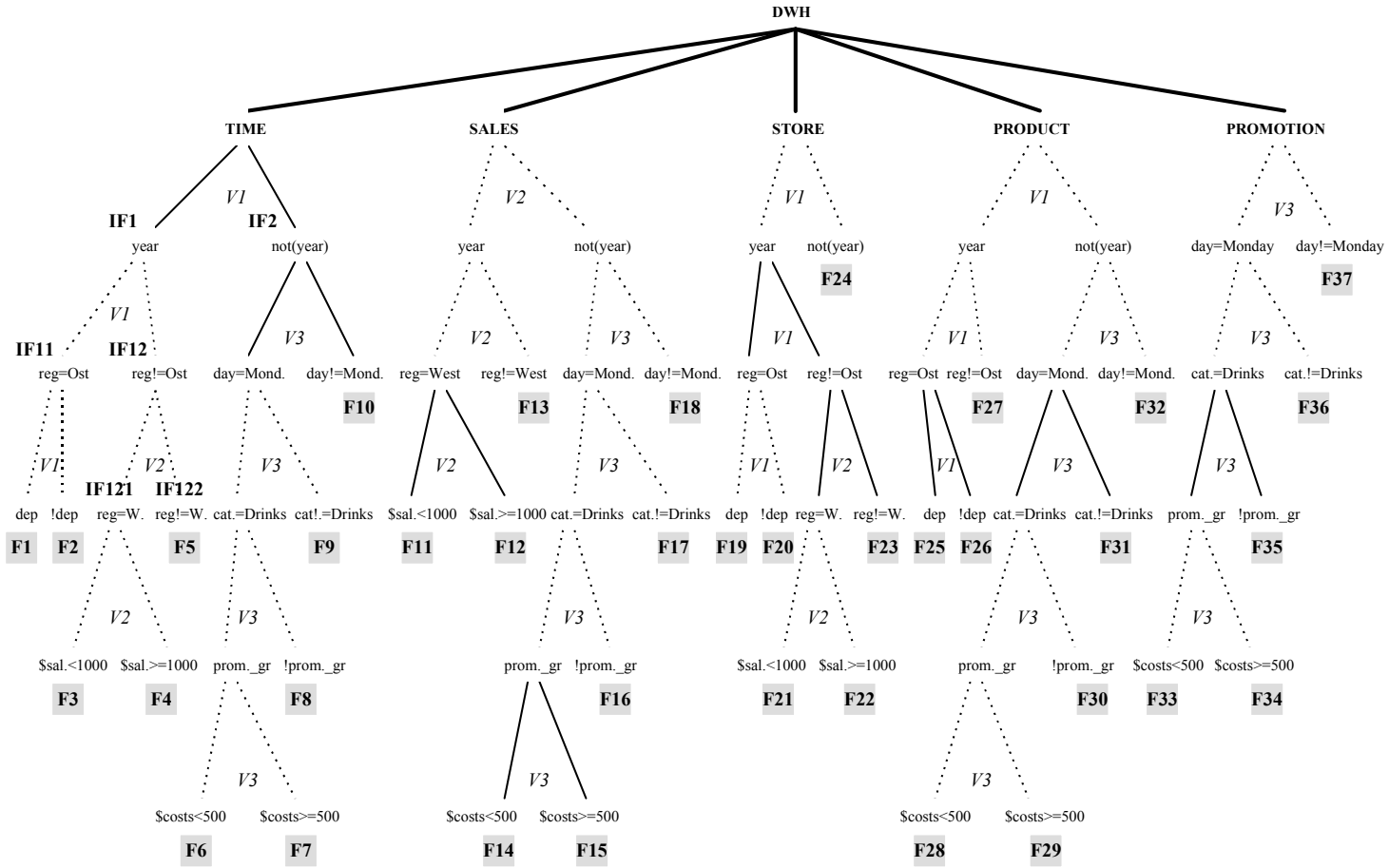

Abbildung 20: Der Fragmentierungsbaum

die horizontale Fragmentierung von TIME schon durchgeführt wurde und in zwei Zwischenknoten IF1 und IF2 resultiert, wird auch für die Rolle2 nur der linke Unterbaum (der, der von dem Knoten IF1 abgeleitet wurde) von Interesse sein. Man hat für die erste Rolle den Knoten IF1 geteilt und zwar in den IF11 Unterbaum, in dem die Daten der Region „Ost“ aufbewahrt werden und in den Unterbaum mit dem Wurzel IF12, in dem alle restliche Daten gespeichert werden. Die Rolle2 darf auf die Daten der Region „West“ zugreifen. Deswegen kommen für diese Rolle nur noch die Daten des rechten (IF12) Unterbaums in Frage. Der Knoten IF12 wird an dieser Stelle durch die abgeleitete horizontale Fragmentierung in IF121 und IF122 geteilt. Diese Fragmentierung wurde auf Grund des Wertes des Attributs „region“ der Dimension STORE durchgeführt. Alle jährlichen Daten der Dimension TIME, die sich auf die Region West beziehen, werden in dem Unterbaum mit dem Wurzel IF121 gefunden. (Der linke Unterbaum, der vom Knoten IF122 ausgeht, beinhaltet jährliche Daten die sich weder auf die Region Ost noch auf die Region West beziehen.) Der letzte Fragmentierungsschritt für die zweite Rolle und Dimension TIME stellt wieder eine abgeleitete horizontale Fragmentierung dar. Die wird im Bezug auf den Wert des Attributs „dollar_sales“ der SALES-Fact-Tabelle durchgeführt. Das Fragment F3 beinhaltet letztendlich die jährlichen Daten der Region West, wo die „dollar_sales“ kleiner als 1000 $ ist. Dieses Fragment erfüllt alle Anforderungen der Rolle2 und wird deswegen der Rolle zur Verfügung gestellt.

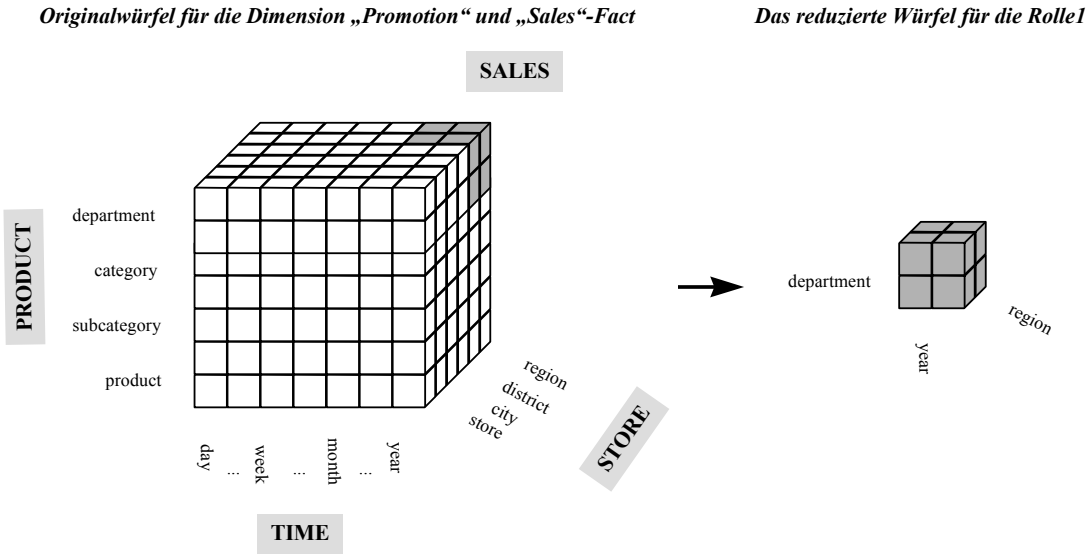

Abbildung 21: Das original Datenwürfel und das reduzierte Würfel für die Rolle1

Dieses Fragmentierungsverfahren erfolgt ähnlich für alle andere Rollen und resultiert in 37 Fragmenten.

Die Zuordnung von Fragmenten zu den Rollen sieht folgendermaßen aus:

- Die Rolle1 hat im Bezug auf die Dimensionen: TIME, STORE und PRODUCT gewisse Zugriffsbeschränkungen. Wie dem Fragmentierungsbaum zu entnehmen ist, darf die Rolle auf die Fragmente F1, F19 und F25 zugreifen. Da der Zugang zu der Dimension PROMOTION und zum Fact SALES nicht begrenzt ist, darf die Rolle alle von ihnen abgeleiteten Fragmente (F11 bis F18 und F33 bis F37) sehen.

- Die Zugriffsrechte der Rolle2 werden in Bezug auf die Dimensionen TIME und STORE und auf das Fact SALES beschränkt. Diese Rolle darf deswegen auf die Fragmente F3, F11 und F21 zugreifen. Da sie alle Daten der Dimensionen PRODUCT und PROMOTION sehen darf, hat sie den Zugang zu allen ihren Fragmenten (F25 bis F37).

- Die Rolle3 ist am meisten beschränkt. Sie darf ungestört alle Daten der Dimension STORE sehen (Fragmente F19 bis F24), bei allen anderen Dimensionen und beim Fact werden ihre Zugriffsrechte begrenzt. Als Ergebnis darf sie noch auf die Fragmente F6, F14, F28 und F33 zugreifen.

- Die Rolle4 (Management) sieht alle Daten, das heißt sie hat den Zugang zu allen Fragmenten.

Die Zuordnung von Rollen zu den Fragmenten sieht so aus:

- Auf die Fragmente F1, F19, F25 darf von der Rolle1 zugegriffen werden, weil sie genau die Anforderungen dieser Rolle beinhalten. Auch die Rolle4 hat den Zugang zu diesen Fragmenten, da sie alle Daten des Data Warehouses sehen darf.
- Fragmente F3, F11 und F21 stehen den Rollen 2 und 4 zur Verfügung
- Fragmente F6, F14, F28 und F33 werden den Rollen 3 und 4 zugeordnet
- Auf alle restlichen Fragmente darf nur von der Rolle 4 aus zugegriffen werden.

Das Überprüfen der Zugriffsrechte

Wie funktioniert jetzt das Überprüfen der Zugriffsrechte? Nehmen wir an, daß ein Benutzer, der die Rolle2 spielt, eine Abfrage an das Data Warehouse schickt, in der er die jährlichen Verkaufszahlen der Region „West“ anfordert. Das Sicherheitssystem generiert eine Liste der Fragmente die notwendig sind, um diese Abfrage zu beantworten. Diese Liste wird dann mit der, zu der Rolle2 zugeordneten Liste der Fragmenten verglichen. Falls sich herausstellt, daß die Fragmente, die die Abfrage beantworten, der Liste der Rolle2 - Fragmenten angehören, wird die Abfrage beantwortet und der Benutzer bekommt die gesuchten Informationen. Falls das aber nicht der Fall sein sollte, heißt das, daß die Rolle zu einigen Daten, die zu Beantwortung der Abfrage notwendig sind, keinen Zugang hat.

Zuordnungsschema

Eine sehr wichtige Voraussetzung für die Sicherheit eines Informationssystems ist eine klare Festlegung der Datensensibilität und ebenso die Erstellung der Sichtbarkeitslabel („*clearance labels*“) für *Views*[16].
Betrachten wir die definierten *Views* und die Fragmente auf die diese *Views* zugreifen. Die *Sicherheitslabel* setzen sich aus zwei Komponenten zusammen:

1. Die ***Sensibilitätsstufe***, (z.B. Top-Secret, Secret,Company-Confidetail, Calssified, ..., Public) die ihrer Natur nach eigentlich eine hierarchische Komponente ist, spielt im Sicherheitsmodell eine bedeutende Rolle.

2. Die ***Kategorie*** bzw. die Relationen oder Objekttypen, auf die sich die ermittelte Sensibilität bezieht.

Diese Komponente bilden ein Paar: *<Sensibilitätsstufe, Kategorie>*.
Alle Fragmente sind geordnet nach der Anzahl der *Views,* die auf sie zugreifen können. Dieses ist wichtig, um die Sensibilitätsstufe der Fragmente bestimmen zu können. Je mehr *Views* auf ein Fragment zugreifen dürfen, desto niedriger wird seine Sensibilitätsstufe und somit auch seine Klassifikationsstufe. Die Anordnung der Fragmente repräsentiert die Sensibilität der gespeicherten Daten in Fragmenten.

Bei den *Views* ist es umgekehrt. Je mehr Fragmente ein *View* sehen kann, desto höher wird seine *„Clearance“*. Das heißt, daß ein *View* mit der höheren Clearance mehr Daten sehen kann als eins mit der niedrigeren Clearance (der auf weniger Fragmente zugreifen kann).

Um dieses mathematisch darzustellen, muß die bekannte mathematische Kardinalitätsfunktion so redefiniert werden, daß sie für dieses Sicherheitsmodell ausgenützt werden kann.

Definition:
Es sei F die Menge der Fragmente eines relationalen Schemas, V die Menge der *Views,* die auf diese Fragmente zugreifen, und card(a: F->V) die Kardinalität der Relation, die jedes Fragment zu einer Menge der *Views* zuordnet. Dann stellt die Relation card(a: F->V) die Klassifikationsstufe ***class(F)*** des Fragments F dar.

Dem Fragmentierungsbaum nach, gilt es:

a(F1)={V1,V4}	a(F8)={V4}	a(F15)={V4}
a(F2)={V4}	a(F9)={V4}	a(F16)={V4}
a(F3)={V2,V4}	a(F10)={V4}	a(F17)={V4}
a(F4)={V4}	a(F11)={V2,V4}	a(F18)={V4}
a(F5)={V4}	a(F12)={V4}	a(F19)={V1,V4}
a(F6)={V3,V4}	a(F13)={V4}	a(F20)={V4}
a(F7)={V4}	a(F14)={V3,V4}	a(F21)={V2,V4}

[16] Ein *View* stellt die Sicht, die eine Rolle auf die Data Warehouse Daten hat, dar

a(F22)={V4}
a(F23)={V4}
a(F24)={V4}
a(F25)={V1,V4}
a(F26)={V4}
a(F27)={V4}
a(F28)={V3,V4}
a(F29)={V4}
a(F30)={V4}
a(F31)={V4}
a(F32)={V4}
a(F33)={V3,V4}
a(F34)={V4}
a(F35)={V4}
a(F36)={V4}
a(F37)={V4}

Daraus folgt:

class(F1) = class(F3) = class(F6)= class(F11) = class(F14) = class(F19) = class(F21) = class(F25) = class(F28) = class(F33)

und

class(F2) = class(F4) = class(F5) = class(F7) = class(F8) = class(F9) = class(F10) = class(F12) = class(F13) = class(F15) = class(F16) = class(F17) = class(F18) = class(F20) = class(F22) = class(F23) = class(F24) = class(F26) = class(F27) = class(F29) = class(F30) = class(F31) = class(F32) = class(F34) = class(F35) = class(F36) = class(F37)

Jetzt bestimmt man die Datensensibilität unserer Fragmente folgenderweise :

class(F1, F3, F6, F11, F14, F19, F21, F25, F28, F33) < class(F2, F4, F5, F7, F8, F9, F10, F12, F13, F15, F16,F17, F18, F20, F22, F23, F24, F26, F27, F29, F30, F31, F32, F34, F35, F36, F37)

Weiters sei „d:V->F“ die Relation, die den verschiedenen *Views* die ihnen zugeordneten Fragmente zuweist. Dann ist card(d:F->V) die Anzahl der ungleichen Fragmente F, auf die ein Anwender mit dem *View* V zugreifen darf. Sie stellt die „Clearance“ (***clear(V***)) des *Views* V dar.

Der Fragmentierung unseres Beispieles nach gilt:

d(V1) = { F1, F11, F12, F13, F14, F15, F16, F17, F18, F19, F25, F33, F34, F35, F36, F37 }
d(V2) = { F3, F11, F21, F25, F26, F27, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37 }
d(V3) = { F6, F14, F19, F20, F21, F22, F23, F24, F28, F33 }
d(V4) = { F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37 }

Daraus schließen wir Folgendes:

{V4} > {V1, V2} > {V3}

Also, *View* V4 ist das „mächtigste“ *View*, das am meisten Fragmente sehen kann, während das *View* V3 auf die wenigsten Fragmente zugreifen kann.

Die Liste der Fragmente (und *Views*) ist keine total-geordnete Liste. Zum Beispiel gehören die Fragmente F1 und F3 gehören zur gleichen Sensibilitätsstufe, werden aber von verschiedenen *Views* gesehen .

3.5.2 Metadaten - Sicherheitsmodell

Einer der wichtigsten Teile eines Data Warehouses sind seine **Metadaten** - oder Daten über Daten. Metadaten beeinflussen alle Niveaus eines Data Warehouse , existieren und fungieren aber in einer anderen Dimension als andere Warehousedaten. Metadaten, die von Data Warehouse - Entwicklern benutzt werden, um das Kreieren und Warten vom Data Warehouse zu managen und zu kontrollieren, werden außerhalb des Data Warehouse aufbewahrt. Die Metadaten, die sich auf die Data Warehouse - Benutzer beziehen, sind dagegen ein Teil des Data Warehouse. Diese Daten werden benutzt, um den Zugriff auf und die Analyse von Daten zu kontrollieren. [PERK96]

Es gibt zwei Typen von Data Warehouse Metadaten:
- Strukturelle Metadaten
- Zugriffsmetadaten

Strukturelle Metadaten

Strukturelle Metadaten werden für die Schöpfung und Wartung vom Data Warehouse benutzt. Sie beschreiben völlig die Data Warehouse - Struktur und seinen Inhalt. Der grundlegende Baustein der strukturellen Metadaten ist ein Modell, das ihre Datengegenstände, ihre Merkmale und ihre Zwischenbeziehungen beschreibt. Die Information, wie die Benutzer ein Data Warehouse benutzen oder benutzen möchten, hilft bei der Entscheidung, wie sie am besten vom Data Warehouse bedient werden. Daraus kann entschieden werden, welche Datengegenstände gebraucht werden und wie sie zu organisieren (aggregieren) sind.

Die Zahl und Besonderheiten von Datenaggregationskategorien in einem Data Warehouse hängen direkt von den Arten von Individuen , die das Data Warehouse benutzen, ab.

Strategische Denker suchen globale Antworten und benötigen sehr wenige Aggregationskategorien. Diese Aggregationen können aber sehr komplex werden.
Betriebliche Denker neigen dazu, jedes Maß durch jede Kategorie, die in ihrem Teil des Unternehmens benutzt wird, anzusehen. Deswegen verlangen sie eine Vielzahl von weniger komplexen Aggregationskategorien.

Wenn sich die Data Warehouse - Struktur verändert, ändern sich seine Metadaten dementsprechend. Alte Versionen der strukturellen Metadaten werden aufbewahrt, um die veränderliche Natur des Data Warehouse zu dokumentieren und den Zugang zu archivierten Daten zu ermöglichen.

Zugriffsmetadaten

Zugriffsmetadaten stellen eine dynamische Bindung zwischen dem Data Warehouse und den Endbenutzer - Anwendungen dar. Sie enthalten die Unternehmensmeßwerte die vom Data Warehouse unterstützt werden, Benutzer-definierte Namen und Aliases. Diese Daten enthalten noch die Beschreibung des Data Warehouse Servers, Datenbanken, Tabellen, detaillierte Daten und summierte Daten zusammen mit einer Beschreibung der ursprünglichen Datenquellen und der durchgeführten Transformationen.

Zugriffsmetadaten stellen „*drill-down*“ und „*roll-up*“ Regeln, sowie die *Views* über die Dimensionen und Hierarchien (wie Produkte, Märkte und Kunden) zur Verfügung. Diese Daten können auch die Regeln für die benutzerdefiniert Kalkulationen und Abfragen enthalten. Diese Metadaten ermöglichen die Sicherheit für den einzelnen Benutzer, Benutzergruppen oder das ganze Unternehmen im Bezug aufs Lesen, Verändern u. Ä. von Kalkulationen, summierten Daten oder Analysen.

Ein Beispiel

In Rahmen des praktischen Teils dieser Diplomarbeit wurde ein Metadaten - Sicherheitsmodel implementiert. Das grundlegende Data Warehouse, für das das Sicherheitssystem erstellt werden sollte, wurde schon im Kapitel 1.11.1 vorgestellt. Es handelt sich um ein Data Warehouse mit 4 Dimensionen *(Time, Product, Promotion* und *Store*) und einem Fact (*Sales*).

Das Sicherheitsmodell funktioniert auf folgende Weise:

1. Es gibt eine Metadaten-Datei, die das ganze Data Warehouse beschreibt.
2. Verschiedene Benutzer können sich an das System anmelden und brauchen für die Ausführung ihrer Aufgaben verschiedene Daten aus dem Data Warehouse. Benutzer, die gemeinsame Bedürfnisse und Befugnisse äußern, werden in Gruppen zusammengefaßt.
3. Für jede Benutzergruppe wird definiert, was diese Gruppe von den Data Warehouse-Daten sehen darf.
4. Wenn sich ein Benutzer am System meldet, wird überprüft, zu welcher Gruppe er gehört. Seiner Gruppenangehörigkeit nach werden ihm die für ihn zugelassenen Daten präsentiert.

Metadaten

Der Inhalt des Data Warehouse wird durch diese Datei beschrieben.
Zuerst werden die allgemeinen Daten angeführt (Programmstück 1), wie z.B. die Währung, die im Data Warehouse benutzt wird (*currency*), verschiedene Maße (*weight*), das letzte Veränderungsdatum (*lastupdate_time*) , der Name des Data Warehouse (*database*) usw.

```
dwh = (
        description = (
                lastupdate_time = "Fri, 17-Apr-1997 10:54:00 GMT"
                currency = (
                        USD = ("Us-Dollar" "$")
                        1
                )
                weight = (
                         ("Once" "oz")
                        1
                ) // Eof Weight unit
                ini = (
                        user = test
                        pass    =       test
                        database        = grocery
                )
        ) // EOF description
```

Programmstück 1

Nach dieser allgemeinen Beschreibung, werden die Facts mit zu ihnen gehörenden Dimensionen und Primärschlüsseln der Dimensionen aufgezählt. (In „grocery“ gibt es nur ein Fact-Sales und vier dazugehörende Dimensionen: Time, Product, Promotion und Store.) (Programmstück 2)

```
facts = (
   (
          id  = "Sales_Fact"
     sql = "Sales_Fact"
     des = "Sales Data"
     dim = (
       (
         sql = "Time"
         key = "time_key"
       )
       (
         sql = "Product"
         key = "product_key"
       )
       (
         sql = "Promotion"
         key = "promotion_key"
       )
       (
         sql = "Store"
         key = "store_key"
       )
     )
```

Programmstück 2

Weiter werden alle Attribute (*attr*) des Facts aufgezählt und beschrieben. Zu der Beschreibung eines Attributs gehören:

- sein Identifikator (*id*),
- seine sql-Beschreibung (*sql*),

- sein Format (*format*) (da es sich bei einem Fact um numerische Daten handelt, wird dieses Feld gebraucht, um die Zahlenformatierung festzulegen), sein Typ (*type*) - dieses Feld hat üblicherweise die Werte „additive“ oder „derived“, was darauf hinweist, daß das Attribut summierbar ist, oder daß sein Wert aus den Werten anderer Attribute abgeleitet werden kann. Falls es sich um ein abgeleitetes Attribut handelt, wird noch ein zusätzliches Feld - Kalkulation (*calculation*) gebraucht, um anzuzeigen, wie der Attributwert berechnet wird.)
- Zuletzt gibt es noch ein Feld für die Beschreibung des Attributs (*des*), das aus zwei Einträgen besteht: einer langen und einer kurzen Beschreibung. (Programmstück 3)

```
attr = (
      (
                  id = "dollar_sales"
                  sql = "dollar_sales"
                  format = "$#,##0.00"
                  type = additive
                  des =  ("Dollar Sales" "Dollar Sales")
     ) // dollar_sales
```

Programmstück 3

Nach der Fact-Beschreibung werden alle Dimensionen und ihre Aggregationen beschrieben. Zuerst wird der sql-Wert der Dimension angeführt(*sql*). Danach kommt ihr Primärschlüssel (*key*) und die lange und die kurze Beschreibung (*des*) (Programmstück 4).

```
dim = (
  // Dim 1. Product
   (
    sql = "Product"
    key = "product_key"
         des = ("Product" "Product" )
```

Programmstück 4

Für jede Dimension werden ihre Aggregationen (*agg*) beschrieben. Um aus dieser Beschreibung die dazugehörige Hierarchie ablesen zu können, besitzt jede Aggregation ein Vorgänger- und ein Nachfolgerfeld (*prev*, *next*). Das Programmstück 5 zeigt die Beschreibung der Aggregation „Sales District“ (Dimension „Store“).Da diese Aggregation in der hierarchischen Darstellung zwischen „Sales region“ und „City“ steht, werden die zwei Aggregationen als ihre Vorgänger bzw. Nachfolger (*prev*, *next*) bezeichnet. Der Spezifikation einer Aggregation gehören noch ihre Sql - Bezeichnung (*sql*) und die lange und die kurze Beschreibung (*des*) an.

```
(
       prev = (
                        "sales_region"
       )
       next = (
                        "city"
       )
```

```
sql = "sales_district"
        des = (
                "Sales District"
                "Sales District"
        )
) // sales_district
```

Programmstück 5

Die vollständige Metadaten-Datei des „grocery“ Data Warehouse finden Sie im Anhang A.

Verschiedene Benutzergruppen sollen verschiedene Daten des Data Warehouse sehen können. Eine Art, sie auf diese Daten zu beschränken, ist die, die wir in diesem Sicherheitsmodell angewandt haben: Für jede Benutzergruppe wird aus der Original-Metadaten-Datei eine eigene Datei abgeleitet. Diese Gruppen-Metadaten-Datei beschreibt das reduzierte Data Warehouse, das diese Gruppe sehen darf. Die Benutzer dieser Gruppe haben dann den Eindruck, das sie nicht beschränkt sind, denn sie können ungestört durch ihr Data Warehouse „*drill-down*“ und „*roll-up*“ machen, verschiedenste Abfragen stellen, Daten summieren, rotieren, usw. Die auf diese Weise implementierte Sicherheit erspart dem Entwickler weitere Überprüfungsmaßnahmen.

Ein reduziertes Data Warehouse sieht anders aus als das Original. Es hat seine eigenen Hierarchien. Wenn eine Benutzergruppe z.B. in der Dimension „Store“ statt allen vier Hierarchiestufen (Sales Region → Sales District → City → Store Name) nur Sales Region und City sehen darf, sieht die Hierarchie der „Store“-Dimension in ihrem Data Warehouse so aus: Sales Region → City. Die Benutzer aus dieser Gruppe können dann auf die Sales District-Ebene und auf die niedrigste Granularitätsstufe (Sotre Name) nicht mehr zugreifen, da sie überhaupt nicht wissen, daß es diese Ebenen gibt.

Die Gruppen-Metadaten-Dateien werden für jede neue Gruppe definiert. Dafür gibt es spezielle Eingabemasken, wo der Sicherheitsmanager diese Dateien durch einfache graphische Schnittstellen definieren kann. Alle Metadaten-Dateien werden dann auf dem Informationsserver aufbewahrt. Wenn sich ein Benutzer an dem System anmeldet, wird seine Gruppenangehörigkeit untersucht und das für ihn vorgesehene reduzierte Data Warehouse wird im zur Verfügung gestellt[17].

[17] Die Benutzerdefinition und die Überprüfung ihrer Benutzernamne/Passwort Daten bei der Anmeldung wird durch dieses System auch unterstützt. Mehr davon finden Sie in der Diplomarbeit [STO97].

4 Intranet

Die Lieferung von Data Warehouse Informationen an Mitarbeiter eines Unternehmens und weiter, in die Welt, ist eine teure Angelegenheit. Das World Wide Web bietet eine Lösung an: private Firmenintranets sind die am schnellsten wachsenden Gebiete des Webs. Intranets stellen ein neues Niveau der Informationsverteilung zwischen den Angestellten eines Unternehmens dar. Diese private Netze unterstützen derzeit Text, Bild und Ton-Datentypen als "statische" HTML-Dokumente.

Mit den richtigen Werkzeugen und der richtigen Architektur kann das Data Warehouse über das Intranet unternehmensweit zugänglich gemacht werden. Dieses Netz bildet die Grundlage für eine umfassende Informationsinfrastruktur des Unternehmens. Es gibt drei wichtige Vorteile solch einer Infrastruktur[INFA96]: Intranet-Ökonomie, Datenintegration und Benutzerkooperation.

Intranet Ökonomie

Die Gesamtkosten der Klient/Server Datenverarbeitung sind hoch, wenn man Kommunikations- und andere Kosten berücksichtigt. Der PC ist sicherlich mit der Verarbeitungsleistung, dem Speicher, Software und mit File-Management sehr stark aufgerüstet und belastet geworden. Das Ergebnis ist eine "fette" Klientenarchitektur.

Das Intranet kann ökonomische Aspekte der Unterstützung einer breiten Schicht von Nutzern verbessern. Das Intranet ist eine "dünne" Kundenarchitektur. Es reduziert nicht nur die
Kommunikationskosten, sondern es kann auch der PC sehr gut durch ein preiswertes Intranet Gerät ersetzt werden.

Datenintegration

Das wertvollste Vermögen eines Unternehmens sind die Betriebsdaten, die bei der täglichen Aufgabenlösung benutzt werden. Wenn ein Data Warehouse entwickelt wird, organisieren die Firmen die Daten auf eine Art, die die Daten für die Entscheidungsträger nützlich macht. Wann das Data Warehouse auf dem Intranet basiert, können die Benutzer zwischen strukturierten Datenanalysen (erzeugt Berichte in Spalten und Reihen) und unstrukturierten Daten auswählen.

Benutzerkooperation

Das Teilen von Ideen und Erfahrungen zwischen den Arbeitsgruppen durch das Intranet ist ein Teil des Wissensaustauschs.
Informationsarten ,die auf dem Intranet existieren, sind:

- Numerische Daten

- Dokumente
- Erfahrungen
- Ideen

Wahre Kooperation verlangt eine interaktive Informationsteilung so, daß der Empfänger ohne Hilfe mit der Analyse weitermachen kann oder sich in eine ganz neue Richtung verzweigen kann. Wenn man z.B. einen Bericht von einem Intranet Benutzer bekommt, sollte man sofort in der Lage sein„„*drill-down*“ und „*roll-up*“ in jeder Dimension zu machen, die Ergebnisse zu drehen, zusätzliche Kalkulationen als Teil seiner Analyse hinzuzufügen und dann seine Arbeit für die anderen in der Organisation freizugeben. Dies erfordert dynamische Berichterstattung, basierend auf den Data Warehouse-Daten.

Die Technologien, die das Intranet stützen, stellen folgenden Nutzen zur Verfügung:
- Vernetzte Rechner können untereinander Informationen teilen oder übertragen.
- Vernetzte Rechner sind heterogen - das heißt, sie können auf einer Vielfalt von Betriebssystemen und Hardware von verschiedenen Verkäufern laufen.
- Allgemeine Benutzeranwendungen wie Email, Netzbrowser, etc. sind auf den allgemein benutzten Plattformen vorhanden.
- Hypertext-Links vereinfachen die Navigation und das Abrufen von Informationen.

Die Untersuchungen im Jahr 1996 haben gezeigt, daß zwei Drittel der Unternehmen das Intranet schon benutzt haben, oder zumindest überlegen, es zu nutzen (Abbildung 22). Diese Unternehmen haben das Intranet als einen mächtigen Mechanismus identifiziert, der die Informationen schnell und einfach zur Verfügung stellt.

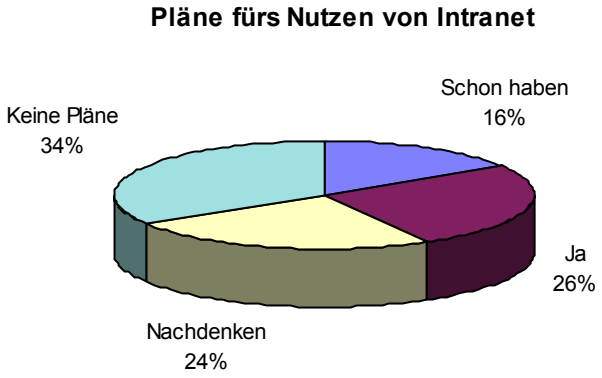

Abbildung 22: Pläne fürs Nutzen vom Intranet [INET96]

Intranets beschaffen eine sehr wirksame Kommunikationsplattform, die sowohl effizient als auch ausdehnbar ist.

4.1 Intranetsicherheit

Sicherheit bedeutet, daß einigen Parteien der Zugang zu gewissen Informationen ermöglicht wird, während dies den anderen beschränkt oder verboten wird. Die meisten gängigen Webserver bieten heute Zugriffskonfigurationen auf einer Benutzer/Gruppen/Bereichsgrundlage. Einige gehen noch weiter und erlauben dem Systemadministrator, die Zugriffsrechte auf bestimmte Seiten aufgrund von IP-Adressen zu vergeben. (Dies würde z.B. bedeuten, daß auf die Dateien mit Personalinformationen eines Unternehmens nur von dem Computer aus der Personalabteilung dieses Unternehmens zugegriffen werden kann.)

Data Warehouse und die Sicherheit im Intranet

Das liberale Teilen von Informationen und Kooperation unter Benutzern auf dem Intranet hebt die Datensicherheitsanforderungen. Das Data Warehouse soll als eines der wertvollsten Vermögen des Unternehmens betrachtet werden. Daten müssen gesichert werden, aber wenn zu dicht kontrolliert wird, wird der Wert des Data Warehouses nie völlig verwirklicht werden .

Nehmen wir an, daß ein Benutzer einen Bericht erstellt und entscheidet, ihn auch anderen zugänglich zu machen. Wenn ein anderer Benutzer nicht authorisiert wird, den Bericht zu empfangen und auf ihn zuzugreifen, dann darf dieser Benutzer den Bericht nicht ansehen und kein „drill-down“ in den Gebieten, wo er keine Zugriffsrechte hat, durchführen. Berichte, die von Daten, die im Data Warehouse gelagert werden, geschaffen worden sind, sollten nicht einfach als Textdateien geteilt werden. Alle Berichte sollten die zugrundeliegende Logik umfassen, die dem Empfänger die Fähigkeit gibt, Berichte sofort zu analysieren und zu modifizieren.

Verschlüsselung im Intranet

Verschlüsselung von Daten stellt ein höheres Sicherheitsniveau zur Verfügung. Z.B. durch das Nutzen von Netscape „Secure Socket Layer“ (SSL), können die Daten, die zwischen Klient und Server ausgetauscht werden, verschlüsselt werden. Dies ermöglicht den Benutzern, wichtige Anwendungen über unsichere Kommunikationskanäle laufen zu lassen, ohne sich Sorgen um einen Eindringling zu machen, der das Netz belauscht und die übertragene Informationen ansieht.
Verschlüsselung kann auch dann eine Rolle spielen, wenn die Intranet - Anwendungen vielfache Organisationen oder Stellen überspannen - ein wirksames virtuelles Privatnetz, das über das öffentliche Internet läuft. Eine zunehmende Anzahl von Organisationen benutzt ihre öffentlichen Webserver in dieser Weise - wobei sie Daten nur den Partnern oder Kunden zur Verfügung stellt. Intelligente „Firewall“- Lösungen verwenden „Tunnels“, um eine sichere Verbindung zwischen Stellen zu schaffen und offen zu halten.
Sehr wichtig für eine gelungene Verschlüsselung ist auch die richtige Auswahl von Schlüsseln und die Vertraulichkeit, wenn der Schlüssel schon ausgewählt worden ist.
„Purveyor IntraServer“ benutzt eine sehr sichere und unknackbare Methode, die Schlüsselpaare zu erzeugen: eine Mischung aus Mausbewegungen und anderen Parametern [INET96].

Sicherheit der Anwendungsarchitektur

„The Common Gateway Interface“ (CGI)-Einrichtung vom Web ermöglicht, daß die Klienten die Software, die bei dem Server liegt, ausführen. Um die Intranet-sichere Anwendungen zu erstellen, muß man sowohl über eine durchdachte Sicherheitsstrategie als auch über entsprechende Anwendungsarchitektur verfügen. Die meisten Web- Anwendungen bieten allen Benutzern den gleichen Zugang. Die vorhandenen Informationen sind entweder von geringer oder gar keiner Vertraulichkeit [INFA96].

4.2 Firewalls

Nach der Internet-Explosion in den Neunzigern sind es mehr als 100 Millionen Rechner, die ans Netz geschlossen sind, und die Zahl wächst exponentiell weiter. Während das Internet einem Unternehmen unbegrenzte Ressourcen und eine globale Sicht zur Verfügung stellen kann, kann es auch den Konkurrenten und den Hackern Zugang zu wertvollsten Unternehmensvermögen geben. Es ist deswegen im globalen Markt von heute eine Verpflichtung, das eigene Netz mit einem Firewall und einem Sicherheitsplan zu schützen.

Eine bekannte Vorgangsweise der Internet-Sicherheit ist, den Firewall an der Verbindungsstelle zum Internet zu errichten. **Firewall** ist ein System, das den Verkehrsfluß zwischen dem Internet und Intranet kontrolliert.

Die wichtigsten Firewall-Lösungen sind[INTSC]: Paket-Filternde Firewalls, Firewalls der Anwendungsebene, Verschlüsselungs-Firewalls und die Pseudo -Firewalls. Hier werden die ersten zwei, die am meisten verwendet werden, beschrieben.

- ***Paket-Filternde Firewalls***

  Ein Router kann als ein Firewall benutzt werden. Netze bieten das Datenpaketfiltern auf ihren Routers. Da der Router schon mit der Führung von Datenpaketen befaßt ist, ist dies eine gute Weise, ein Firewall anzulegen. Datenpaketfilternde Routers erzeugen Firewall-Sicherheit, in dem sie gewisse Pakete („dropping packets“) nicht liefern. IP ist die niedrigste Protokollstufe des TCP/IP - Protokolls und jedes IP-Datenpaket enthält eine Quellen-IP-Adresse, eine Ziel-IP-Adresse und eine IP-Portnummer.

  Das Paketfiltern funktioniert so, daß der Router die Datenpakete auf Grund ihrer Quellen- , Zieladresse oder ihrer Port-Nummer fallen läßt. Die Routers erlauben den Benutzern eine „Annehmen /Ablehnen“ Tabelle zu definieren. Jede Zeile dieser Tabelle enthält sowohl Quellen und Ziel- IP-Adresse als auch IP Portnummer.

Zusätzlich wird noch angezeigt ob das Datenpaket, das der Beschreibung entspricht, anzunehmen oder abzulehnen ist.

Ein Problem beim Datenpaketfiltern-Verfahren ist, daß manche Anwender sehr schwer mit Begriffen wie IP-Adressen oder Portnummer zurecht kommen. Viel natürlicher ist es, auf dem höheren Niveau von Anwendungen, wie Telnet, FTP und HTTP zu denken. Auf dem Paketniveau kann man aber nicht unterscheiden, um welche Anwendung es sich handelt, deswegen kann man auch nicht herausfinden welche Effekte die Tabelleneinträge auf aktuelle Anwendungen haben werden. Ein zweites Problem bei Datenpaketfiltern ist es, daß sie leicht fehlfunktionieren. Nämlich dann, wenn man beim Erstellen von der Tabelle nicht vorsichtig genug ist und ein bekanntes Internet-Site zuläßt zum Intranet zuläßt, kann es passieren, daß das einen weit offenen Zugang zum Intranet zur Folge hat.

- ***Firewalls der Anwendungsebene***

Ein Rechnersystem zwischen dem Internet und einem Privatnetz anzulegen, ist ebenfalls ein Weg, ein Firewall zu bauen. Solche Firewalls nennt man Bastion-Server (Anwendungs-Proxy-System).

Jedes System, das ein Privatnetz mit dem Internet verbindet, hat zwei Netzeingänge. Proxy wird mit zwei IP-Adressen konfiguriert, eine für jede Schnittstelle. Alle Pakete die zwischen Internet und Intranet fließen sollen, müssen zuerst durch das Proxy-System durchgehen (Abbildung 23).

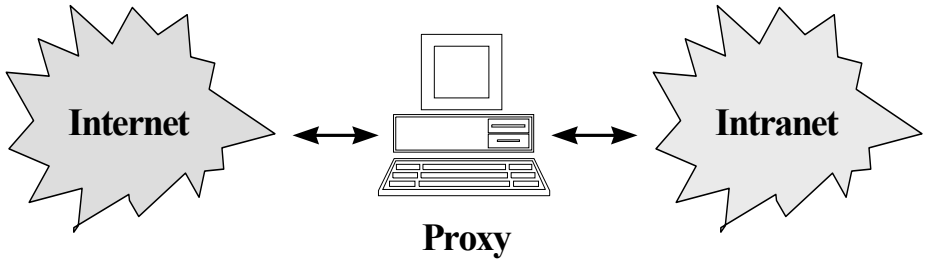

Abbildung 23: Ein Firewall der Anwendungsebene

Ein Anwendungsproxy ist einfach eine Miniaturausführung einer bestehenden Internet Dienstleistung, die man durch das Firewall durchlassen will. Will z.B. ein Benutzer auf dem Privatnetz das Telnet benutzen, um eine Verbindung mit einem Rechner aufzunehmen, der im Internet (außerhalb des Privatnetzes) steht, mußer sich zu erst mit dem Anwendungsproxy, das auf dem Bastion-Server läuft, verbinden. Proxy entscheidet dann, ob sich der Benutzer mit dem Zielrechner verbinden darf. Seine Entscheidung hängt von vielen Parametern ab (Zielsystem, Tageszeit, Benutzername)[18], die vom Systemadministrator definiert werden.

[18] Das sind die Parameter, zu welchen die Paketfiltern keinen Zugang haben

Falls alle Anforderungen erfüllt werden, werden die Datenpakete auf die andere Netzwerkschnittstelle des Proxy und dann weiter ins Internet weitergeleitet. Durch die Firewalls der Anwendungsebene kann man also besser kontrollieren, welche Datenpakete über das Firewall übertragen werden können und welche nicht.

5 Zusammenfassung

Die Personal-Computerära transformierte eine Generation von Benutzern in elektronische "knowledge seekers“. Bei der Suche nach dem Wissen haben die Benutzer ihre Personal Computers mit genau so viel Daten geladen, wie es möglich war. Benutzer wollten selbständige Analytiker werden.

Die Netzwerk-Ära erlaubt Benutzern, sich in "Wissensteiler" zu entwickeln und hebt den mächtigen Vorteil vom Gemeinschaftsproblem - Einsatz hervor. Die Anwendung von Internettechnologien in einem Intranetsystem kann den Informationsfluß und -Wert innerhalb einer Organisation dramatisch vergrößern. Benutzer können schnellen und rechtzeitigen Zugang zu einer viel weiteren Vielfalt an bestehenden Informationen, die aus verschiedenen Quellen gewonnen werden, erwerben. Traditionelle, auf Papier basierende Informationsverteilung kann durch Intranet-Anwendungen ersetzt werden, was die Kosten senkt und die Promptheit des Informationsflusses vergrößert.

Das Intranet stellt eine Grundlage für den Umbau der Unternehmensdateninfrastruktur. Durch den Einsatz eines Data Warehouses wird die Effizienz und die Produktivität eines Unternehmens deutlich erhöht. Mit dem Wachstum des Inter- und Intranets steigen die Bedrohungen, denen ein Data Warehouse ausgesetzt ist. Die Anwendung von Intranet-Firewalls bietet einen Schutz vor Attacken, die aus dem Intranet kommen können.

Das Recherchieren, das ich im Gebiet der Data Warehouse-Sicherheit betrieben habe, hat mir gezeigt, wie wichtig dieses Thema und allgemein das Thema Computersicherheit ist. Datenbanksicherheit ist eine der wichtigsten Aufgaben der Computersicherheit. Die Informationen, die ein Unternehmen besitzt, sind sein größtes Kapital und müssen vor den Konkurrenten streng geschützt werden. Es sind aber nicht nur die wirtschaftlichen Interessen, die die Datensicherheit erfordern. Wie das Beispiel der Krankenhäuser und der Flugzeuggesellschaften zeigte, kann das unberechtigte Verfügen über Daten große gesundheitliche und menschliche Schäden zur Folge haben. Aus allen diesen Gründen ist die Sicherheit heute eigentlich eine Verpflichtung jedes Datenbankbesitzers.

6 Index

L

M

N

O

P

Q

R

S

T

U

V

W

Z

7 Literaturverweise

[AGS97] *"Daten Bazar ", A*gens Consulting GmbH 1997 http://www.agens.de/

[BFZ96] *Hintergrundinformation*, Bayerisches Forschungszentrum für Wissensbasierte Systeme, 1996, http://www-forwiss.tu-muenchen.de/

[BUY95] *OLAP: playing for keeps*, Frank A. Buytendijk, July 1995

[CAS92] *Database Security*, S. Castano, M. Fugini, M.Mertella, P. Samarati, Addison-Wesley, 1992

[FERN80] *Database Security and Integrity*, E.B:Fernandez, R.C. Summers, C.Wood, Addison-Wesley Publishing Company

[FORS] *OLAP Council White Paper*, Sarah Forsman

[INET96] *Intranet: Intranet Technologies Deployes Behind the Firewall for Corporate Productivity*, Prepared for Interet Society INET'96 Annual Meeting, http://www.process.com/intranets/wp2.htp.htm

[INFA96] Putting the Data Warehouse on the Intranet, An INFORMATION ADVANTAGE White Paper, 1996, http://www.briefingbook.co.za/pubs/9611/dwintra.htm

[INTSC] *Internet and Intranet Security*, http://www.cs.purdue.edu/homes/holtsm/Sindex.html

[KEN93] *Introduction To Multidimensional Database Technology,* Kenan System Corporation 1993-1995

[MIC95] MicroStrategy 95- Comparsion of Approaches, Whitepaper

[MIC95] *True Relational OLAP: The Future of Decision Support*, MicroStrategy 1995, http://www.strategy.com/tro_dbj.htm

[PEN97] *The OLAP Report: What is OLAP?*, Nigel Pendse, Business Intelligence Ltd, July 18th, 1997

[PERK96] *Developing a Data Warehouse, The Enterprise Engineering Approach*, Alan Perkins, Visible Systems Corporation, 1995-1996, http://www.ozemail.com.au/~ieinfo/dw.htm

[PERN94] *Database Security,* G. Pernul, Advances in Computers, Vol. 38, p. 1-72, 1994

[RAD96] *Modeling the Data Warehouse,* Neil Raden

http://techweb.cmp.com/iw

[RAV94] *Role-Based Access Control: A Multi-Dimensional View*, Ravi S. Sandhu, Edwad J. Coyne, Hal L. Feinstein and Charles E. Youman, Proc. of 10th Annual Computer Security Applications Conf., Orlando, Florida, December 5-9, 1994, pages 54-62.

[SICH97] *A Security Concept for OLAP*, Remzi Kirkgöze, Nevena Katic, Mladen Stolba, A Min Tjoa, Eighth International Workshop on Database and Expert Systems Applications, pages 619-627, September 1997

[SILV96] *Mind If I Have a Look at Your Sensitive Warehouse Data?* , Len Silverstoen, Larry Smith, October 1996, RMOUG Newsletter, Volume13

[STA92] *Operating Systems*, William Stallings, Macmillan Publishing, Singapore 1992

[STO97] „Sicherheitsmodellierung in Data Warehouse Systemen“ - Diplomarbeit von Mladen Stolba, Institut für Softwaretechnik, TU Wien 1997

[WID95] *Research Problems in Data Warehousing*, Jennifer Widom, Proc. of 4th Int'l Conference on Information and Knowledge Management

8 Anhang A

Das komplette Kod der Metadaten-Datei, die das "Grocery"- Data Warehouse beschreibt.

```
dwh = (
        description = (
                lastupdate_time = "Fri, 17-Apr-1997 10:54:00 GMT"
                currency = (
                        USD = ("Us-Dollar" "$")
                        1
                )
                weight = (
                         ("Once" "oz")
                        1
                ) // Eof Weight unit
                ini = (
                        user = test
                        pass     =        test
                        database         = grocery
                )
        ) // EOF description
 facts = (
  (
          id  = "Sales_Fact"

    sql = "Sales_Fact"
    des = "Sales Data"
    dim = (
      (
       sql = "Time"
       key = "time_key"
      )
      (
       sql = "Product"
       key = "product_key"
      )
      (
       sql = "Promotion"
       key = "promotion_key"
      )
      (
       sql = "Store"
       key = "store_key"
      )
    )
    attr = (
      (
                        id = "dollar_sales"

                        sql = "dollar_sales"
                        format = "$#,##0.00"
                        type = additive
```

```
                    des =  ("Dollar Sales" "Dollar Sales")
) // dollar_sales
(
  id = "unit_sales"

                sql = "unit_sales"
  format = "#,##0"
                type = additive
  des =  ("Unit Sales" "Unit Sales")
) // unit_sales
(
  id = "dollar_cost"

                sql = "dollar_cost"
  format = "$#,##0.00"
                type = additive
                des =  ("Dollar Cost" "Dollar Cost")
) // dollar_cost
(
  id = "customer_count"

                sql = "customer_count"
  format = "#,##0"
                type = additive
  des =  ("Customer Count" "Customer Count")
) // customer_count
            (
                    id = "avg_price"

                    sql = "avg_price"
                    type = derived
                    calculation = "/ f_3 f_1"
                    format = "$#,##0.00"
                    des = ("Avg. Price" "Avg. Price")
            ) // avg_price
            (
                    id = "avg_cost"

                    sql = "avg_cost"
                    type = derived
                    calculation = "/ f_0 f_1"
                    format = "$#,##0.00"
                    des = ("Avg. Cost" "Avg. Cost")
            ) // avg_cost
            (
                    id = "avg_purchased_dollars"

                    sql = "acg_purchased_dollars"
                    type = derived
                    calculation = "/ f_0 f_4"
                    format = "#,##0.00"
                    des = ("Avg. Purchased Dollars" "Purchased Dollars")
            ) // avg_purchased_dollars
            (
                    id = "avg_purchased_number"

                    sql = "avg_purchased_number"
                    type = derived
                    calculation = "/ f_0 f_3"
```

```
                format = "#,##0.00"
                des = ("Avg. Purchased Number" "Purchased Number")
            ) // avg_purchased_number
            (
                id = "gross_profit"

                sql = "gross_profit"
                type = derived
                calculation = "- f_0 f_3"
                format = "$#,##0.00"
                des = ("Gross Profit" "Gross Profit")
            ) // gross_profit
            (
                id = "gross_margin"

                sql = "gross_margin"
                type = derived
                calculation = " / * 100.0 - f_0 f_3 f_0"
                format = "$#,##0.00"
                des = ("Gross Margin" "Gross Margin")
            ) // gross_margin
    ) // attr
  ) // Sales Fact
) // facts
dim = (
 // Dim 1. Product
  (
    sql = "Product"
    key = "product_key"
          des = (
                "Product"
                "Product"
           )
    agg = (
      (
        prev = (
          "brand"
        )
        next = (
        )
        sql = "description"
                  des = (
                        "Description"
                        "Desc."
                   )
      ) // attr. description
      (
        prev = (
          "brand"
        )
        next = (
        )
        sql = "full_description"
                  des = (
                        "Full Description"
                        "F. Desc."
                   )
      ) // attr. full_description
      (
```

```
  prev = (
    "brand"
  )
  next = (
  )
  sql = "SKU_number"
            des = (
                        "SKU_number"
                        "SKU"
              )
) // attr. SKU_number
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "package_size"
            des = (
                        "package_size"
                        "Pack. Size"
              )
) // attr. package_size
  (
  prev = (
    "subcategory"
  )
  next = (
     "description"
     "full_description"
     "SKU_number"
     "package_size"
     "package_type"
     "diet_type"
     "weight"
     "weight_unit_of_measure"
     "units_per_retail_case"
     "units_per_shipping_case"
     "cases_per_pallet"
     "shelf_width_cm"
     "shelf_height_cm"
     "shelf_depth_cm"
  )
  sql = "brand"
            des = (
                        "Brand"
                        "Brand"
              )
) // attr. brand
  (
  prev = (
    "category"
  )
  next = (
     "brand"
  )
  sql = "subcategory"
            des = (
                        "Subcategory"
```

```
                    "Subcat."
                )
) // attr. subcategory
  (
  prev = (
    "department"
  )
  next = (
     "subcategory"
  )
  sql = "category"
            des = (
                    "Category"
                    "Cat."
                )
) // attr. category
  (
  prev = (
  )
  next = (
     "category"
  )
  sql = "department"
            des = (
                    "Department"
                    "Depart."
                )
) // attr. department
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "package_type"
            des = (
                    "Package Type"
                    "PA Type"
                )
) // attr. package_type
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "diet_type"
            des = (
                    "Diet Type"
                    "Diet Type"
                )
) // attr. diet_type
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "weight"
```

```
            des = (
                        "Weight"
                        "W"
             )
) // attr. weight
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "weight_unit_of_measure"
            des = (
                        "Weight Unit of Measure"
                        "W UoM"
             )
) // attr. weight_unit_of_measure
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "units_per_retail_case"
            des = (
                        "Units per Retail Case"
                        "Units per Retail Case"
             )
) // attr. units_per_retail_case
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "units_per_shipping_case"
            des = (
                        "Units per Shipping Case"
                        "Units per Shipping Case"
             )
) // attr. units_per_shipping_case
  (
  prev = (
    "brand"
  )
  next = (
  )
  sql = "cases_per_pallet"
            des = (
                        "Cases per Pallet"
                        "Cases per Pallet"
             )
) // attr. cases_per_pallet
  (
  prev = (
    "brand"
  )
  next = (
  )
```

```
        sql = "shelf_width_cm"
                des = (
                        "Shelf Width [in cm]"
                        "Shelf/Width/cm"
                 )
      ) // attr. shelf_width_cm
        (
        prev = (
          "brand"
        )
        next = (
        )
        sql = "shelf_height_cm"
                des = (
                        "Shelf Height [in cm]"
                        "Shelf/Height/cm"
                 )
      ) // attr. shelf_height_cm
        (
        prev = (
          "brand"
        )
        next = (
        )
        sql = "shelf_depth_cm"
                des = (
                        "Shelf Depth [in cm]"
                        "Shelf/Depth/cm"
                 )
      ) // attr. shelf_depth_cm
    ) // eof Agg Product
  ) //   Eof  Dim 1. Product
 // Dim 2. Promotion
  (
    sql = "Promotion"
    key = "promotion_key"
          des = (
                "Promotion"
                "Promotion"
           )
    agg = (
      (
        prev = (
        )
        next = (
          "price_reduction_type"
        )
        sql = "promotion_name"
                des = (
                        "Promotion_Name"
                        "Name"
                 )
      ) // promotion_name
      (
        prev = (
          "promotion_name"
        )
        next = (
          "ad_type"
```

```
  )
  sql = "price_reduction_type"
            des = (
                        "Price Reduction-Type"
                        "PR Type"
             )
) // price_reduction_type
(
  prev = (
    "price_reduction_type"
  )
  next = (
    "display_type"
  )
  sql = "ad_type"
            des = (
                        "AD Type"
                        "AD Type"
             )
) // ad_type
(
  prev = (
    "ad_type"
  )
  next = (
    "coupon_type"
  )
  sql = "display_type"
            des = (
                        "Display-Type"
                        "Disp.-Type"
             )
) // display_type
(
  prev = (
    "display_type"
  )
  next = (
    "ad_media_type"
  )
  sql = "coupon_type"
            des = (
                        "Coupon-Type"
                        "Coupon-Type"
             )
) // coupon_type
(
  prev = (
    "coupon_type"
  )
  next = (
    "display_provider"
  )
  sql = "ad_media_type"
            des = (
                        "AD Media-Type"
                        "ADM-Type"
             )
) // ad_media_type
```

```
      (
        prev = (
          "ad_media_type"
        )
        next = (
          "promo_cost"
        )
        sql = "display_provider"
                  des = (
                              "Display Provider"
                              "Disp. Provider"
                   )
      ) // display_provider
      (
        prev = (
          "display_provider"
        )
        next = (
        )
        sql = "promo_cost"
                  des = (
                              "Promotion Cost"
                              "Promo. Cost"
                   )
      ) // promo_cost
    ) // eof agg Promotion
  ) // Eof Dim 2. Promotion
  // Dim 3. Store
   (
     sql = "Store"
     key = "store_key"
           des = (
                 "Store"
                 "Store"
            )
     agg = (
       (
         prev = (
         )
         next = (
                               "sales_district"
         )
         sql = "sales_region"
                   des = (
                              "Sales Region"
                              "Sales Region"
                    )
       ) // sales_region
       (
         prev = (
                               "sales_region"
         )
         next = (
                               "city"
         )
         sql = "sales_district"
                   des = (
                              "Sales District"
                              "Sales District"
```

```
                )
        ) // sales_district
        (
          prev = (
                                "sales_district"
          )
          next = (
                                "name"
          )
          sql = "city"
                    des = (
                                "City"
                                "City"
                    )
        ) // city
        (
          prev = (
                                "city"
          )
          next = (
          )
          sql = "name"
                    des = (
                                "Store Name"
                                "Store Name"
                    )
        ) // name
    ) // eof Agg Store Dimension
  ) // EOF Dim Store
    // Dim 4. Time
    (
      sql = "Time"
      key = "time_key"
            des = (
                  "Time"
                  "Time"
             )
      agg = (
        (
          prev = (
          )
          next = (
            "year"
          )
          sql = "fiscal_period"
                    des = (
                                "Fiscal Period"
                                "Fisc. Perid."
                    )
        ) // fiscal_period
        (
          prev = (
            "fiscal_period"
          )
          next = (
            "quarter"
          )
          sql = "year"
                    des = (
```

```
                    "Year"
                    "Year"
                )
        ) // year
        (
          prev = (
            "year"
          )
          next = (
            "month"
          )
          sql = "quarter"
                des = (
                    "Quarter"
                    "Quart."
                )
        ) // quarter
        (
          prev = (
            "quarter"
          )
          next = (
            "week_number_in_year"
          )
          sql = "month"
                des = (
                    "Month"
                    "Mon."
                )
        ) // month
        (
          prev = (
            "month"
          )
          next = (
            "day_of_week"
          )
          sql = "week_number_in_year"
                des = (
                    "Week / Year"
                    "Week"
                )
        ) // week_number_in_year
       (
          prev = (
            "week_number_in_year"
          )
          next = (
          )
          sql = "day_of_week"
                des = (
                    "Day / Week"
                    "Day"
                )
        ) // day_of_week
    ) // agg Dims. Time
) // EOF Dim. Time
) // EOF Dims
```

```
) // EOF DWH - grocery
```