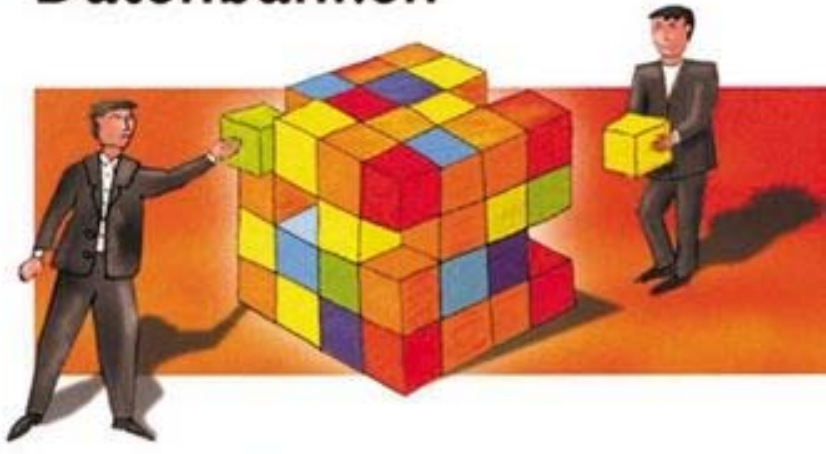
PEARSON Education

OLAP

Multidimensionale Datenbanken

Nils Clausen



Produkte, Markt, Funktionsweisen und Implementierung

Auch zum OLAP-Server von Microsoft

ADDISON-WESLEY

2 Schritte zum Aufbau einer OLAP-Datenbank

2.1 Unterscheidung zwischen Struktur- und Bewegungsdaten

Wir beginnen dieses Kapitel mit einem praktischen Beispiel. Sie sind verantwortlicher Projektleiter eines Direktversenders für Möbel und Einrichtungsgegenstände und wollen eine OLAP-Datenbank für Ihre Vertriebsabteilung aufbauen, um die Vorteile der multidimensionalen Auswertungen zu nutzen und auf sich häufig ändernde Marktsituationen reagieren zu können.

Die erste Frage, die Sie sich und Ihren zukünftigen Anwendern stellen sollten, ist: Was wollen wir sehen? Welche Informationen müssen mir die Datenbank und darauf aufsetzende Benutzeroberflächen zur Verfügung stellen, und welchen Nutzen erwarte ich hieraus? Erst dann sollten Sie versuchen herauszufinden, wo was zur Verfügung steht: sowohl intern wie auch extern. Nachdem diese Untersuchung abgeschlossen ist, beginnt die eigentliche Arbeit: der Aufbau der OLAP-Datenbank.

Um sich dem Aufbau einer OLAP-Datenbank zu nähern, ist es notwendig, die Datenbestände im wesentlichen in zwei Kategorien zu unterscheiden: die Struktur- und die Bewegungsdaten.

2.1.1 Was sind Strukturdaten?

Sicher haben Sie in Ihrem Unternehmen im Data Warehouse, auf dem Host oder in PC-basierter Software Daten vorliegen, welche Kunden in welchem Ort wohnen und wer diese betreut, welche Produkte sich in Ihrem Sortiment befinden und wie diese in Produktgruppen zusammengefaßt werden. Ebenso werden im Vertrieb Kennzahlen oder Variablen wie Umsatz, Absatz und Stornoquote gebraucht, um die Geschäftsvorfälle zu beschreiben. Trivial, aber notwendig zu erwähnen ist, daß auch die Zeit zu den Strukturinformationen gezählt werden kann: Tage und Wochen summieren sich zu Monaten, Monate summieren sich zu Quartalen und

so weiter. Allgemeingültig läßt sich sagen: Daten, die Ihr Unternehmen oder Ihre Anwendung beschreiben, sind Strukturdaten.

Nachfolgend zwei beispielhafte Tabellen:

Merkm	KundNr	Name	Vor-name	Adresse	PLZ	Ort	Kund-Grp
Gesperrt	1102993	Klefisch	Peter	Vogelius-weg 47	33100	Pader-born	Ange-stellte
	1102994	Müller	Hans	Kaffweg 1	35039	Marburg	Selb-ständg.
Gesperrt	1102995	Fischer	Heide	Adickes-allee 14	60322	Frankfurt	Selb-ständg.
	1102996	Knapp	Dieter	Dianastr. 7	14482	Potsdam	Ange-stellte
	1102997	Schuster	Klaus	F1, 20	68159	Mann-heim	Student

Tabelle 2.1: Kundentabelle

Merkm	ArtNr	ArtText	Farbe	Sparte	BxHxT	Prod-Werk
inaktiv	BT11029	Beistelltisch	Grau	Wohnen	$90 \times 98 \times 30$	Bremen
	CT11030	Couchtisch	Erle grau	Wohnen	$100 \times 50 \times 100$	Bremen
	GA11031	Garderobe	Kiefer	Wohnen	$120 \times 120 \times 10$	Neuwied
inaktiv	KS11032	Küchen-schrank	Weiß	Küche	$60 \times 180 \times 60$	Bremen

Tabelle 2.2: Produkttabelle

Die Anordnung der einzelnen Datensätze geschieht in der OLAP-Datenbank innerhalb der Dimensionen. Beachten Sie, daß nicht alle Informationen aus den Tabellen verarbeitet wurden – die oben beschriebenen Tabellen und die notwendigen Kennzahlen und Zeit sehen in der multidimensionalen Struktur dann folgendermaßen aus:

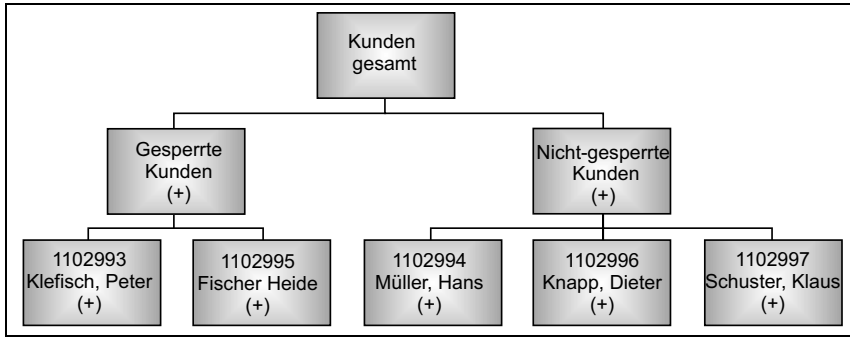


Abbildung 2.1: Kundendimension

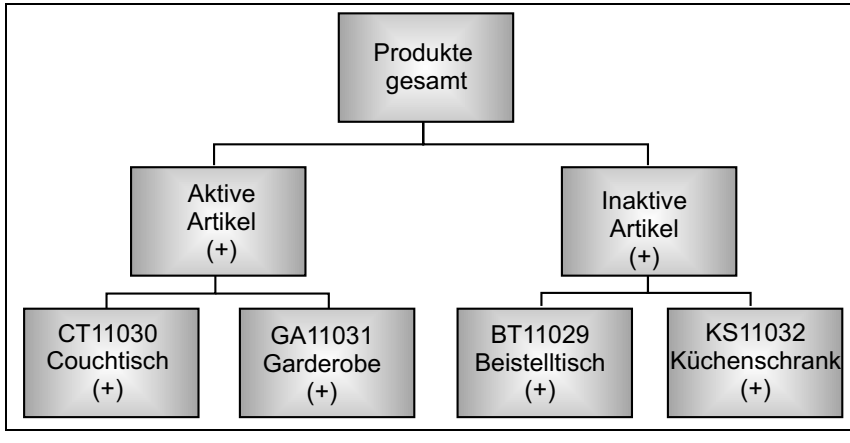


Abbildung 2.2: Produktdimension

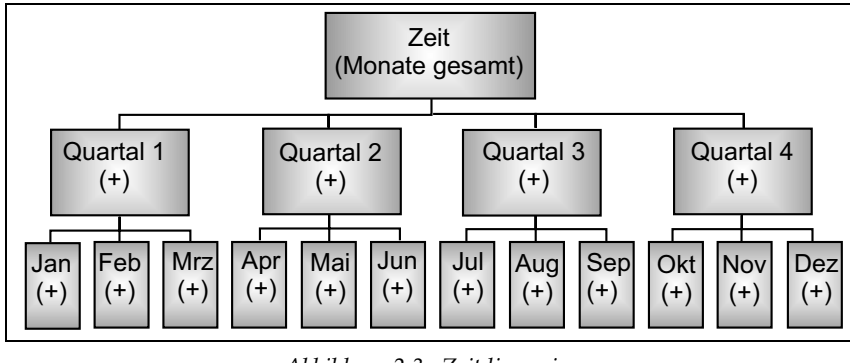


Abbildung 2.3: Zeitdimension

Diese Dimensionen sind es, anhand derer der Anwender nachher in der Lage ist, zu definieren, was er als Ergebnis seiner Abfrage sehen möchte. In dem obigen Datenmodell sind aus Vereinfachungsgründen einige Informationen bisher nicht abgebildet worden:

- Adresse, PLZ, Ort und Kundengruppe der Kunden
- Farbe, Sparte, Maße und Produktionswerk der Produkte

Diese Informationen lassen sich als zusätzliche beschreibende Dimensionen aufnehmen, sofern dies von den Anwendern gewünscht wird. Hierzu sollten Sie folgende Überlegung anstellen: Treten die Informationen häufig auf (PLZ, Kundengruppe, Ort, Farbe, Sparte, Produktionswerk) oder weniger häufig (Adresse und Maße). Treten diese weniger häufig auf, sollten Sie diese nicht in eine zusätzliche Dimension aufnehmen, sondern als Attribute, da der Anwender sonst mit einer Vielzahl an Ausprägungen konfrontiert wird. Nach genauer Überlegung entscheiden wir in diesem Fall, daß die ersten drei Stellen der PLZ mit dem Ort, die Kundengruppe, die Farbe, Sparte und das Produktionswerk als Dimension in das Datenmodell aufgenommen werden sollen.

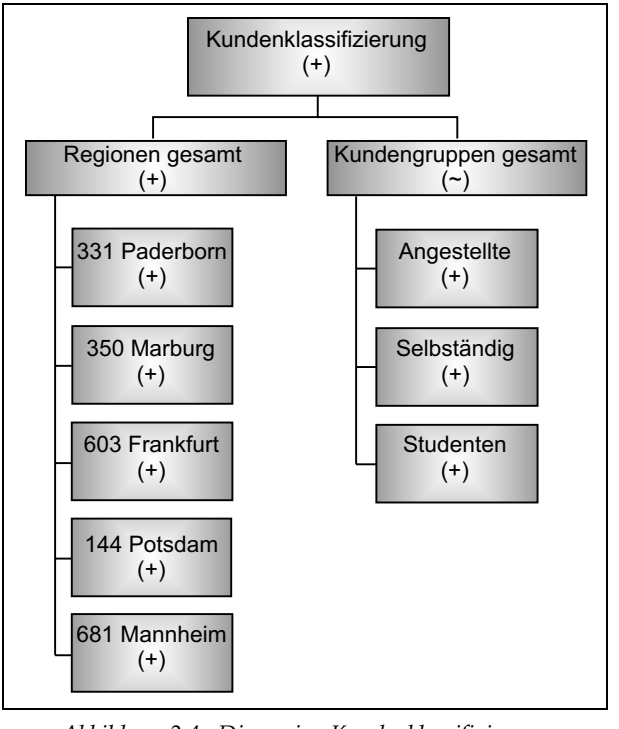


Abbildung 2.4: Dimension Kundenklassifizierung

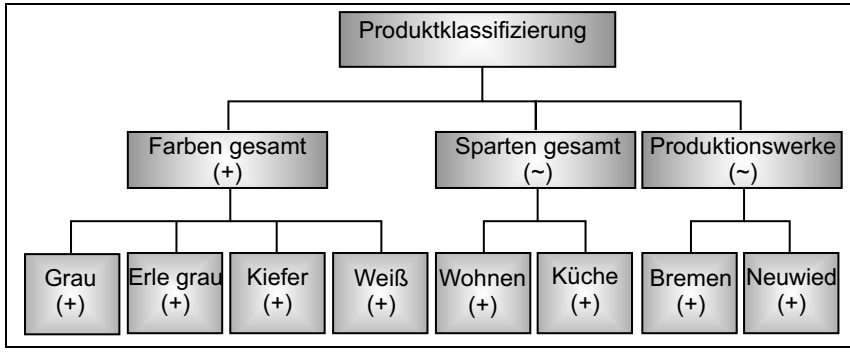


Abbildung 2.5: Dimension Produktklassifizierung

Folgendes gilt:

- Die Dimension Kundenklassifizierung bezieht sich ausschließlich auf Kunden und die Dimension Produktklassifizierung ausschließlich auf Produkte.
- Die Aggregation „Kundengruppen gesamt“ fließt nicht additiv in die Aggregation Kundenklassifizierung ein, da sonst eine doppelte Summe vorhanden wäre. Es handelt sich hierbei vielmehr um eine alternative Sichtweise.

Die Maße der Produkte und die Kundenadressen nehmen wir als Attribute zu den Produkten und Kunden auf. Das bedeutet, daß der Anwender in seiner Auswertung auf einzelne Produkte oder Kunden durch einfachen Mausklick diese Informationen angezeigt bekommt, hierüber allerdings keine hierarchischen Sichten aufgebaut werden können.

Hiermit haben wir nun die Struktur einer OLAP-Datenbank mit fünf Dimensionen aus den vorliegenden, relationalen Tabellen aufgebaut. Der Anwender ist jetzt zwar bereits in der Lage, Auswertungen zu entwerfen, hierfür fehlt allerdings ein wesentlicher Bestandteil der Datenbank: die Bewegungsdaten.

2.1.2 Was sind Bewegungsdaten?

Bewegungsdaten, auch Transaktionsdaten genannt, entstehen im normalen Geschäftsalltag. Unsere Möbelfirma stellt Möbel her und verkauft diese. Bei jedem Vorgang entstehen Daten vorher definierter Form, die sich zu den Strukturdaten in Beziehung bringen lassen.

Der Kunde Schuster bestellt am 4. August telefonisch den Couchtisch CT11030 zum aktuellen Preis von 295,30 DM. Die Auftragsbearbeitung gibt diese Informationen in die Auftragsbearbeitung ein. Diese Bestellung führt am 10. August zu einer Auslieferung des Artikels und zur Rechnungsstellung. Diese Daten reichen aus, um anhand der bereits vorliegenden Strukturdaten alle weiteren benötigten Daten zu erhalten. Daraus wird ein Datensatz in der Tabelle Transaktionsdaten erzeugt, der folgenden Aufbau hat:

Datum	ArtNr	KundNr	Kennzahl	*Wert*
0810	CT11030	1102997	Umsatz	295.30

Tabelle 2.3: Transaktionsdaten

Transaktionen werden durch sog. OLTP-Systeme (Online Transaction Processing Systems) wie beispielsweise SAP, Baan, Peoplesoft oder anderen Applikationen erzeugt. In anderen Bereichen entstehen diese auch durch Systeme wie Scannerkassen, Kreditkartenterminals oder Fahrkartenautomaten.

Diese Tabelle kann nun in die OLAP-Datenbank eingelesen werden. Hierzu ist es notwendig, daß die Transaktionstabelle keine Datensätze enthält, die nicht auch in den Strukturdaten vorhanden sind. Andernfalls werden diese nicht mit eingelesen, und die Auswertungen liefern falsche Ergebnisse. In Abschnitt 2.3 können Sie Einzelheiten dazu erfahren und lesen, wie diese Fehler vermieden werden können.

Der Umfang der vorhandenen Transaktionsdaten hängt von verschiedenen Faktoren ab:

- Aktualisierungshäufigkeit der Daten (täglich, monatlich, jährlich)
- Anzahl der Dimensionen und Attribute
- Anzahl der Datensätze
- evtl. vorhandene redundante Daten

Nachdem die Strukturen aufgebaut und Daten eingelesen wurden, kann die OLAP-Datenbank kalkuliert werden. D.h. in unserem Beispiel, die in der Struktur hinterlegten Regeln zur Aggregation, in diesem Fall ausschließlich Additionen, werden für alle Dimensionskombinationen oder Zellen durchgeführt. Somit erhalten wir eine konsistent berechnete Da-

tenbank, die auf allen definierten Ebenen entsprechende Ergebnisse liefert. Auf dieser Datenbank basierend können Benutzeroberflächen aufgesetzt (siehe auch Kapitel 4) und Analysen betrieben werden.

2.2 Transformierung relationaler Daten in das multidimensionale Modell

Auf die Vorgänge des Einlesens und Kalkulierens soll an dieser Stelle genauer eingegangen werden. Es wird nachfolgend beschrieben, welche Schritte in der Datenmodellierung vorgenommen werden müssen, um eine OLAP-Datenbank zu „füllen".

Wie bereits beschrieben, gehen wir davon aus, daß die Strukturdaten eingelesen und verarbeitet wurden, folglich in ein multidimensionales Datenmodell resultieren, welches keine gefüllten Zellen enthält.

Um Bewegungsdaten aus Datenbanken oder Textdateien einlesen zu können, sollte eine Zugriffsmöglichkeit zu den originären Datenbeständen aufgebaut werden, um möglichst wenig Zwischenschritte zu generieren. Daher ist es nicht ratsam, Datenbestände beispielsweise aus dem Data Warehouse auf den Arbeitsplatz PC im Batchbetrieb zu laden, diese im Access zu verarbeiten, manuelle Eingaben hinzuzufügen und den generierten Datenbestand zum Laden in die OLAP-Datenbank bereitzustellen.

Vielmehr ist es ratsam, einen möglichst automatisierten Ladevorgang anzustreben und damit Fehlerquellen bei der Transformation zu umgehen. Um dies bewerkstelligen zu können, wird das ideale Szenario durch eine dreistufige Architektur im Client/Server-Umfeld ermöglicht (siehe Abbildung 2.6).

Architekturen wie ODBC von Intersolv (*http://www.intersolv.com/*) leisten für einen konsistenten Durchgriff auf relationale Datenbanken auf vielen bekannten Hardwareplattformen gute Dienste. Nachteilig wirkt sich diese universelle Verfügbarkeit in der Verarbeitungsgeschwindigkeit aus. Bei sehr großen Datenmengen ist der vom jeweiligen Hersteller mitgelieferte, sog. native Treiber dem Datenbankzugriff über ODBC vorzuziehen.

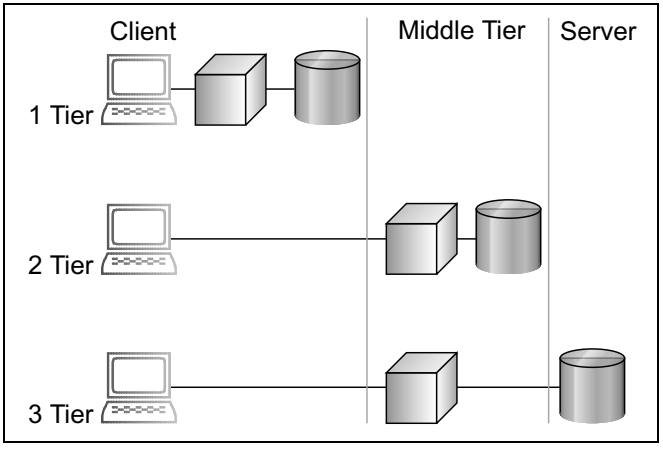


Abbildung 2.6: Dreistufige Architektur im Client/Server-Umfeld

Um den Weg der Daten vom relationalen zum multidimensionalen Medium unabhängig von der physikalischen Zugriffstechnik zu beschreiben, nehmen wir eine, zugegeben sehr einfache und fachlich unvollständige, OLAP-Datenbank mit folgender Struktur:

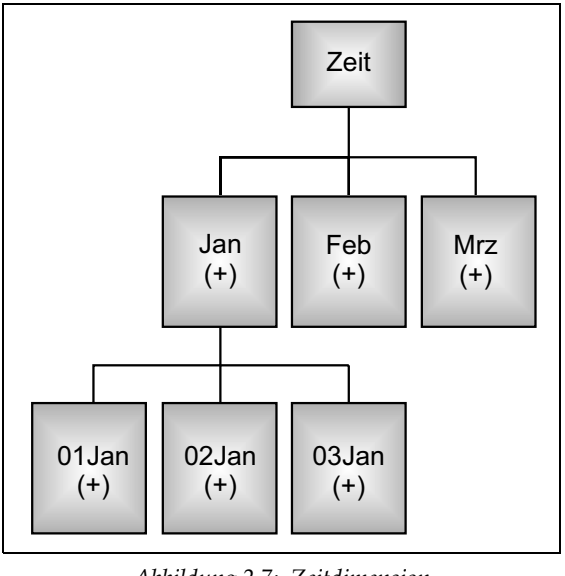


Abbildung 2.7: Zeitdimension

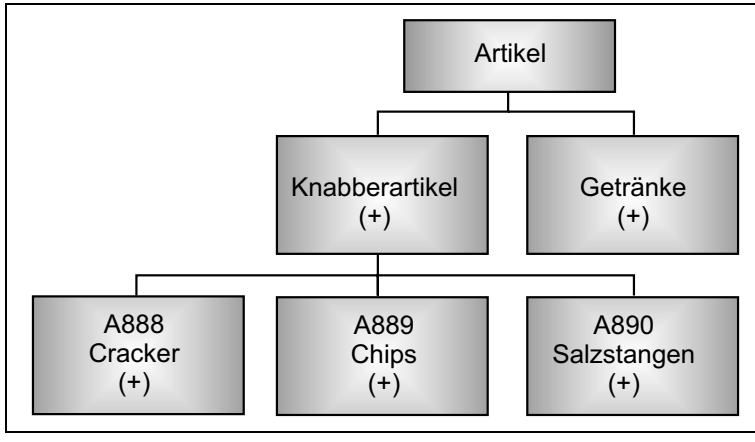


Abbildung 2.8: Artikeldimension

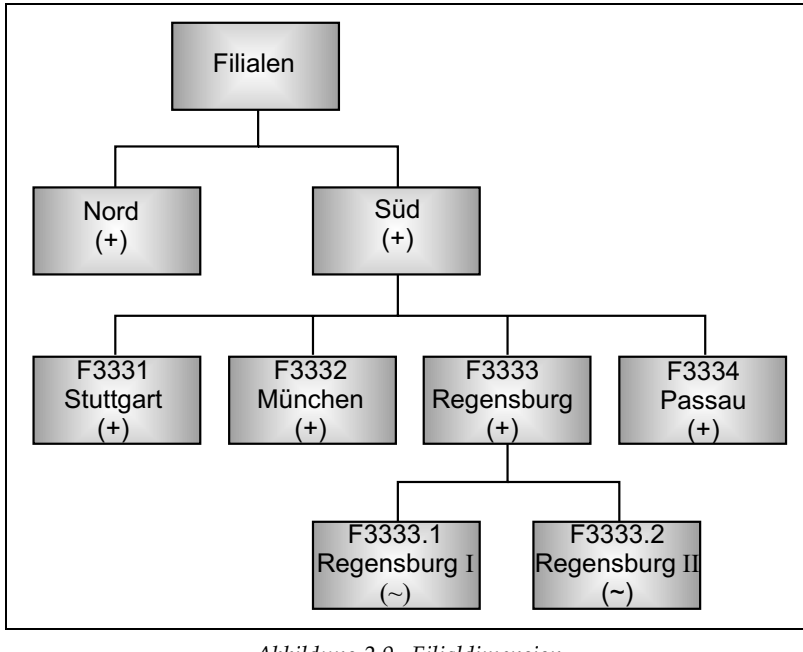


Abbildung 2.9: Filialdimension

Der zu diesem Modell passende Transaktionsdatensatz sieht folgendermaßen aus:

Datum	ArtNr	FilialNr	*Wert*
01Jan	A8888	F3333	433.65

Dieser Datensatz kann nun anhand seiner Spalten `Datum`, `ArtNr` und `FilialNr` im multidimensionalen Modell genau einer Zelle zugeordnet werden. Die Spalte `*Wert*` enthält den Wert dieser Zelle.

In dem dreidimensionalen Modell des Würfels (wir gehen idealerweise davon aus, daß alle Kantenlängen äquivalent sind) könnte das Positionieren und Einfügen des Transaktionsdatensatzes folgendermaßen aussehen:

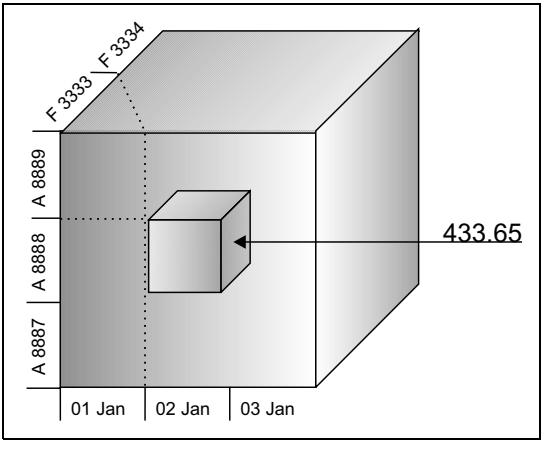


Abbildung 2.10: Einfügen einer Transaktion in die dreidimensionale OLAP-Datenbank

Durch die Informationen im Transaktionsdatensatz lassen sich, obwohl nicht mehr räumlich vorstellbar, auch mehr als drei Dimensionen schlüsseln und so einem entsprechenden multidimensionalen Modell zuordnen. Aufgrund der leichteren Vorstellbarkeit wird in diesen Fällen trotzdem von Würfeln, oder auch von sog. OLAP-Cubes, gesprochen.

Allgemein gelten folgende Grundsätze:

- Die relationalen Transaktionsdaten müssen mindestens so viele Spalten enthalten, wie Dimensionen in der zu ladenden OLAP-Datenbank vorhanden sind, da sonst keine Zelle bzw. Dimensionskombination genau adressiert werden kann. Die benötigte Spaltenanzahl kann, sofern notwendig, natürlich auch über einen SQL-Join mehrerer Tabellen hergestellt werden.
- Nur Transaktionsdaten, für die eine Entsprechung in der OLAP-Datenbank existiert, können geladen werden. Das bedeutet, daß in Fällen, in denen dies nicht zutrifft, eine Methode zum Abfangen angewandt werden muß, um die Strukturen in der OLAP-Datenbank zu erweitern, bevor die Transaktionsdaten geladen werden.

Nachdem die Daten in die OLAP-Datenbank geladen wurden, steht aber noch keine durchweg konsistente Datenbasis zur Verfügung, denn die Datenbank muß noch kalkuliert werden, d.h. alle nicht vorhandenen Werte werden gemäß den vorgegebenen Regeln errechnet und danach zurückgeschrieben. Hier spielt die Performance und die interne Berechnungssystematik der OLAP-Datenbank eine gewichtige Rolle: die Anzahl der Rechenschritte pro Zeiteinheit und die Notwendigkeit, nur wirklich vorhandene Werte in die Berechnung einzubeziehen.

In unserem Beispiel steckt allerdings eine Falle: unter der Ausprägung F3333 befinden sich zwei weitere Ausprägungen F3333.1 und F3333.2, deren Zellen für diese Methodik als fehlend markiert wären, da in dem Transaktionsdatensatz keine Daten hierfür vorhanden sind. Da eine Kalkulation standardmäßig von unten nach oben (Bottom-Up-Szenario) durchgeführt wird, greift auch keine Kalkulation. Für diesen Zweck muß, dies ist produktabhängig, ein Skript oder eine Funktion vom Datenbankadministrator umgesetzt werden, die eine gewünschte Verteilung (prozentual/absolut) der F3333-Ausprägung vornimmt (Top-Down-Szenario). Diese Vorgehensweise gewinnt bei der Planung von Unternehmenskennzahlen, z.B. im Controlling, starke Bedeutung. In diesem Zusammenhang sei auf die Nichtaggregation der Ausprägungen F3333.1 und F3333.2 hingewiesen, damit F3333 bei der Kalkulation der OLAP-Datenbank nicht überschrieben wird.

2.3 Fehlerquellen

Beim Aufbau von OLAP-Datenbanken können Fehler auftreten, die das Gesamtergebnis erheblich beeinflussen können. Zudem existieren eine Anzahl von Beschränkungen, die bei der Bedarfsanalyse oder Konzeption bedacht werden müssen. So ist eine OLAP-Datenbank definitiv nicht geeignet, transaktionsorientierte oder Real-Time-Anwendungen abzubilden.

Nachfolgend soll aufgrund praktischer Erfahrungen auf die häufigsten Fehler und ihre Ursachen eingegangen werden, ohne auf einzelne Produkteigenheiten spezifisch Bezug zu nehmen.

2.3.1 Nicht vorhandene Ausprägungen

Bedingt durch die Notwendigkeit, vor dem Laden der Transaktionsdaten multidimensionale Strukturen aufzubauen, kann es zu einem unterschiedlichen Stand zwischen diesen kommen. Konkret bedeutet dies, daß für vorhandene Transaktionsdatensätze, auch wenn alle notwendigen Zellinformationen geliefert werden, keine Entsprechung in der OLAP-Datenbank vorhanden ist. Dies läßt sich mit dem erfolglosen Einsortieren eines Artikels in ein Supermarktregal vergleichen, für den kein Regalplatz reserviert wurde.

Sollte versucht werden, einen solchen Datensatz zu laden, wird dieser zurückgewiesen und bestenfalls durch einen Eintrag in der Logdatei durch den Datenbankadministrator erkannt. Da ein erfolgreiches Laden die Ergänzung der multidimensionalen Struktur bedingt, wird ein aufwendiges, iteratives Vorgehen eingeschlagen. Vielmehr sollte, bevor das Einlesen der Transaktionsdaten begonnen wird, ein Abgleich der Datenbestände untereinander vorgenommen werden. Dies kann sowohl auf der relationalen Basis anhand eines SQL-Joins und daraus folgender Hinweismeldungen als auch durch das automatisierte Ergänzen nicht vorhandener Ausprägungen in der OLAP-Datenbank erfolgen. Da in den meisten Fällen allerdings nicht bekannt ist, zu welchen Aggregationsstufen die Ausprägung gehört, kann diese Ausprägung in der Regel nur unterhalb der Gesamtsumme angeordnet oder beispielsweise unter ‚Nicht vorhanden' zusammengefaßt werden.

2.3.2 Schlüsselkonflikte

Wenn aus heterogenen Datenbeständen Strukturen und Transaktionen gelesen werden sollen, so kommt es häufig vor, daß Schlüssel als nicht eindeutig vergeben wurden.

Wurde beispielsweise in der Vergangenheit der Artikel 4711 eingestellt und durch einen neuen mit der gleichen Artikelnummer ersetzt, so handelt es sich hierbei um einen Schlüsselkonflikt. Aus diesem Grund muß bei der Neuaufnahme von Objekten jeglicher Art ein eindeutiger, besser noch ein eineindeutiger Schlüssel vergeben werden. Eineindeutig bedeutet in diesem Zusammenhang, daß sowohl aufgrund des Schlüssels auf den Artikeltext als auch vom Artikeltext auf den Schlüssel geschlossen werden kann. Idealerweise werden Schlüssel auch über das gesamte Datenmodell eineindeutig vergeben, d.h., bei gleichen Schlüsseln von Kunden und Artikeln wird ein entsprechendes Präfix wie ‚Kd' bzw. ‚Artikel' vorangestellt, um Probleme bei der OLAP-Datenbank einiger Hersteller auszuschließen.

Schlüsselkonflikte sollten bei der Übernahme von konsistenten und bereinigten Data Warehouse-Beständen in die OLAP-Datenbanken generell nicht auftreten.

2.3.3 Nullwerte

Das Einlesen von Nullwerten stellt in logischer Hinsicht keine Probleme dar, da „Nullen" gewöhnliche numerische Werte darstellen. Problematisch wird es erst dann, wenn aufgrund fehlerhafter Transaktionsdaten fehlende Werte, das heißt technisch sog. Null- oder Nil-Werte durch Nullen ersetzt wurden oder beim Einlesen als Nullwerte interpretiert werden. Dadurch steigt dann die Anzahl der abzuspeichernden Zellen in der OLAP-Datenbank erheblich, so daß auf Sparsity ausgerichtete oder generell verfügbare Kompressionsalgorithmen nicht mehr effizient greifen können. Das resultierende Datenvolumen der OLAP-Datenbank wäre geradezu exorbitant, da möglicherweise alle theoretisch vorhanden Zellen mit Werten gefüllt wären.

Es empfiehlt sich daher, die Datenbestände vor dem Ladevorgang genau zu prüfen, um eine derartige Überraschung auszuschließen.

2.3.4 Überschreiben bestehender Zellen und falsche Arithmetik auf vorhandene Zellen

Sofern die produktspezifische Architektur der OLAP-Datenbank es gestattet, Transaktionsdaten auf einer höheren und nicht nur auf der untersten Ebene einzulesen, können Probleme auftreten, wenn die Datenbank Bottom-Up kalkuliert wird und damit unter Umständen bereits bestehende Zellen auf höheren Ebenen überschrieben werden. Dies kann natürlich auch bei der Aggregation leerer Zellen auftreten. Da das Prüfen auf Zellinhalte oder der Vergleich mit einem zweiten Wert zeitaufwendig und kompliziert ist und, wenn überhaupt, nur über externe Programmierung von Skripten möglich gemacht werden kann, sollte bereits bei der Datenmodellierung auf diese Problematik Rücksicht genommen werden und eine korrekte Reihenfolge der Lade- und Kalkulationsvorgänge festgelegt werden.

Andererseits verbirgt sich hinter diesem Tatumstand auch eine interessante Alternative, wenn es um den Abgleich nachträglich geänderter Daten aus dem OLTP geht. So können z. B. Korrekturdaten einfach auf Zellbasis miteinander verrechnet oder überschrieben werden, so daß immer der letzte aktuelle Stand ohne weiteren Programmieraufwand und vor allen Dingen Performanceeinbußen realisiert werden kann.

2.3.5 Falsche Berechnungsregeln

Bei Anwendern und Datenbankadministratoren, die dem Fachbereich relationaler Datenbanktechnologien entstammen und möglicherweise die Multidimensionalität noch nicht komplett durchdrungen haben, können allgemeine Fehler bei der Datenmodellierung auftreten. Diese sind leider, in Bezug auf das Gesamtergebnis, komplett durchschlagend. Insofern sollte nach Fertigstellung eines Prototypen oder der Designphase das Approval durch den Fachbereich erfolgen, um zu prüfen, ob alle Regeln, insbesondere die der unternehmensspezifischen Kennzahlen oder Variablen, korrekt umgesetzt wurden und ob das Gesamtergebnis den Tatsachen entspricht.

2.3.6 Anteilsberechnungen ohne zweiten Kalkulationslauf

Bei der Berechnung von Prozentwerten und ganz besonders bei Anteilsberechnungen läßt sich folgende Regel anwenden: immer einen zweiten Kalkulationslauf folgen lassen. Der Grund: Da zuerst die absoluten Werte in der OLAP-Datenbank vorliegen bzw. errechnet werden müssen, sollte im zweiten Durchlauf die Zahlenbasis der Prozente auf einen konsistenten Stand gebracht werden. Ansonsten werden die Prozent- wie Absolutwerte behandelt und damit gemäß den allgemeingültigen Berechnungsregeln aggregiert. Diese Werte enthalten dann auf den höheren Ebenen nicht die erforderlichen Zahlen.

2.3.7 Falsche Konfiguration der internen Kompressionslogik

Um das theoretisch gigantische Datenvolumen von multidimensionalen Datenbeständen aufnehmen zu können, wird bei nahezu jedem OLAP-Produkt auf Kompressionsverfahren zurückgegriffen, die sich verschiedene Charakteristika des aufzunehmenden Datenbestandes zunutze machen. Dies können u.a. die sog. Sparsity sein, bei der es sich um nicht komplett mit Daten gefüllte Kombinationen einzelner Dimensionen handelt. Die Information der daraus resultierenden Einstellung wird in der Regel beim Einrichten vom Datenbankadministrator vorgenommen, kann sich aber mit der dynamischen Übernahme neuer Dimensionen oder deren Ausprägungen nachhaltig ändern. Sofern die Einstellungen nicht korrekt vorgenommen wurden und auch nicht in regelmäßigen Abständen überprüft werden, kann es zu negativen Effekten kommen, die das System erheblich verlangsamen, immense Datenmengen verursachen und den Netzwerkverkehr lahmlegen können, da mit jeder Abfrage sehr große Datenblöcke transferiert werden.

Da die verwendeten Kompressionsmöglichkeiten von Produkt zu Produkt sehr unterschiedlich sind und ein Standard derzeit nicht existiert, empfiehlt sich in jedem Fall die genaue Prüfung der Einstellungen durch das Handbuch oder die Online-Hilfe.

2.4 Größenbetrachtungen von OLAP-Datenbeständen

2.4.1 Allgemeines

OLAP-Datenbanken können aus kleinen Datenbeständen sehr große erzeugen. Dies spaltet die EDV-Gemeinde in die Sparer und die Strategen. Die zweite Gruppe erkennt die wahren Vorteile einer OLAP-Datenbank und kann mit einer guten Analyse und der daraus folgenden Architektur der Daten optimale Ergebnisse erzielen. Um diese Analyse durchführen zu können, bedarf es des Verständnisses des Datenbestandes und des daraus resultierenden Volumens, möglicher Kniffe und allgegenwärtiger Fehler.

Nehmen wir einmal an, daß Sie eine Datenbank aufbauen wollen mit folgenden Dimensionen und, der Vereinfachung wegen, nur deren Ausprägungen auf unterster Stufe: 100.000 Kunden, 5.000 Produkte, 156 Wochen, 20 Kennzahlen jeweils mit Ist, Plan, absoluten und prozentualen Abweichungen, 35 Abteilungen, 8 Länder und 19 Verkäuferteams. Wie berechnet sich das daraus resultierende Datenvolumen, wenn jede Zelle 8 Byte beinhaltet? Ganz einfach: Sie bilden das kartesische Produkt der einzelnen Ausprägungen, d.h., Sie multiplizieren alles miteinander. Das gigantische Ergebnis lautet dann 265.574.400.000.000.000 Byte oder 241.539 Terabyte! Wahrlich zu groß, um auf irgendeinem momentan verfügbaren Massenspeicher abgelegt werden zu können. Dabei ist eine solche Datenbank für OLAP-Anwender nicht ungewöhnlich und stellt eher den mittleren Bereich des Möglichen dar. Da dieses gigantische Volumen physikalisch allerdings nicht speicherbar ist, wird die geringe Befüllung der Zellen mit Daten berücksichtigt – denn beispielsweise wird nicht jede Woche jedes Produkt durch jedes Verkäuferteam an alle Kunden verkauft und stellt insofern eine sog. Sparsity, eine dünne Besiedlung mit Daten, dar.

Allerdings läßt sich die Dichte der Daten nicht bestimmen oder einfach voraussagen und entwickelt sich auch nicht linear, sondern eher exponentiell. Um daher verläßliche Aussagen zum resultierenden Datenvolumen einer zu kalkulierenden OLAP-Datenbank treffen zu können, bedarf es der Analyse des Datenbestandes hinsichtlich seiner Beschaffenheit.

Die Datendichte kann sehr gut mittels der folgenden Vorgehensweise bestimmt werden: Bilden Sie jeweils Matrizen aus den einzelnen Dimensionen und schauen Sie sich die Verteilung der Daten an.

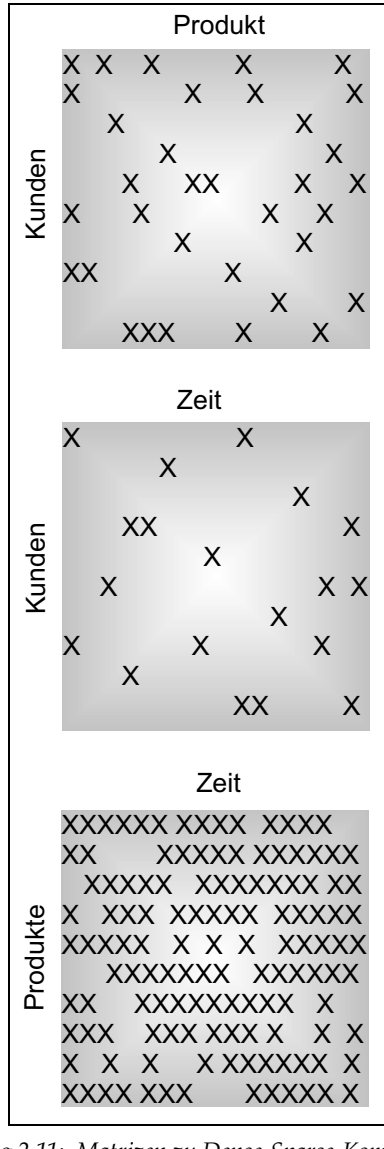


Abbildung 2.11: Matrizen zu Dense-Sparse-Kombinationen

Verschiedene Produkte, wie beispielsweise Arbor Essbase oder der IBM DB2 OLAP-Server, legen mittels dieses Verfahrens auch die internen Datenstrukturen an (Dense-Sparse-Settings), berücksichtigen dabei allerdings nicht, daß es ebenso Kombinationen gibt, die keine Aussagekraft für den Anwender besitzen: Die Frage, welche Kunden der Verpakkungsgröße 500 gr. in diesem Jahr ihren Umsatz um mehr als 20% steigern konnten, ist insofern irrelevant, als Kunden nicht in Verpackungsgrößen unterteilt werden und die, zumindest theoretisch, vorhanden Zellen Kunden/Verpackungsgrößen, keinen Sinn ergeben und somit auch gar nicht reserviert werden sollten.

Es ist allerdings möglich und sicher auch notwendig, bei größeren Datenbeständen auf sog. dynamische Kalkulationen dieser Produkte zurückzugreifen. Dynamische Kalkulationen werden zur Laufzeit durchgeführt und können optional eine Speicherung der kalkulierten Werte herbeiführen – diese Option bewirkt das dynamische Wachsen der OLAP-Datenbank je nach den Abfragegewohnheiten seiner Nutzer. Der Vorteil dynamischer Kalkulationen liegt darin, daß nicht alle Werte vorberechnet werden müssen und insofern auch keine Plattenspeicher und Performance des Servers benötigen. Dies wird zur Laufzeit durchgeführt und führt daher in diesem Punkt zu längeren Abfragezeiten, sofern allzu viele Werte „on-the-fly“ berechnet werden müssen. Daher empfiehlt sich der Einsatz dynamischer Kalkulationen von Ausprägungen nur bei kleinen Dimensionen wie Zeit, Kennzahlen und Soll/Ist. Bei einem großen „Fan-In“ (viele Ausprägungen auf unteren Ebenen) ergibt sich sodann ein nachteiliger Effekt bei der Abfrage-Performance.

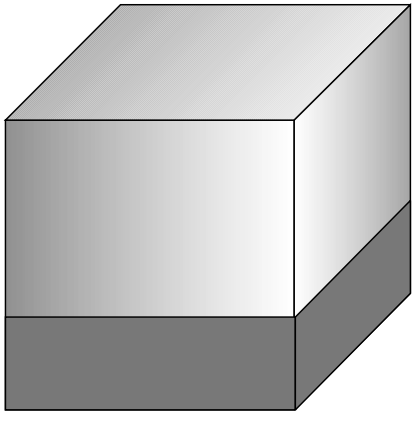


Abbildung 2.12: Nicht komplett vorberechneter Cube

2.4.2 Tuning

Jede Datenbank kann in Ihrer Performance optimiert werden. Die dafür möglichen Mittel differieren allerdings stark, da die Architekturen der in diesem Buch untersuchten Produkte Arbor Essbase, IBM DB2 OLAP-Server, Microsoft OLAP-Server und Oracle Express sehr unterschiedlich sind.

Produktübergreifend lassen sich folgende Faktoren der Minimierung der Lade- und Kalkulationszeiten feststellen:

1. Löschung nicht notwendiger Ausprägungen – nur in Ausnahmefällen empfehlenswert, da eine unvollständige Datenbankstruktur den Nutzer nicht unbedingt zufriedenstellt und der Anpassungsaufwand des eingesetzten Frontends bei jeder Erweiterung der Datenbankstruktur zusätzlich Arbeit verursachen kann.
2. Aufteilung der OLAP-Datenbank in mehrere Teildatenbanken, die sich durch Anpassung der Benutzeroberfläche oder durch die Möglichkeiten der datenbankinternen Verknüpfung dem Anwender transparent wie eine Datenbank darstellen können.

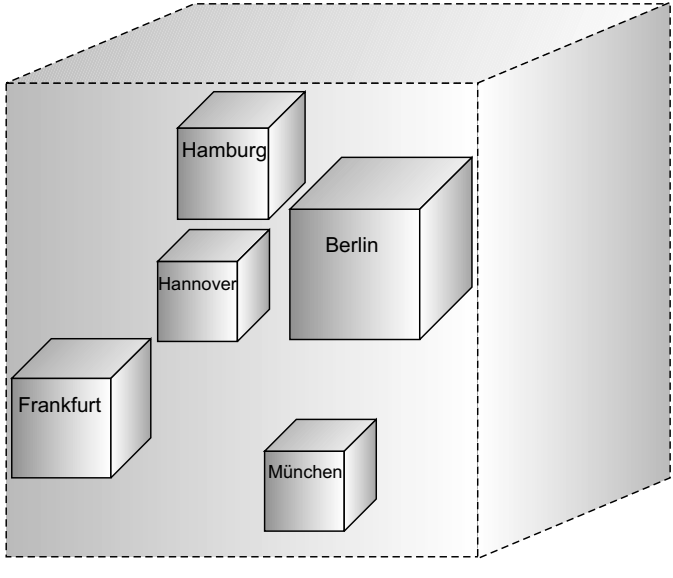


Abbildung 2.13: Aufteilung einer OLAP-Datenbank in Teildatenbanken

3. Sortierung der zu ladenden Datenbestände nach der produktspezifischen, optimalen Methode. Bei einigen Produkten kann eine Sortierung z.B. nach der Dimension mit den meisten Ausprägungen oder auch anderen Richtlinien aufgrund der internen Datenstrukturen eine erhebliche Steigerung bewirken.
4. Einschränkung der theoretisch möglichen Kombinationen/Zellen des Würfels – dies wird nicht von allen Produkten unterstützt. Hierbei werden, entgegen der Vorgehensweise bei Arbor Essbase oder dem IBM DB2 OLAP-Server durch Dense-Sparse-Settings, nicht vorhandene, sachlich falsche oder unbesetzte Zellen auch nicht theoretisch angelegt (Kunden pro Verpackungsgröße). Diese Einschränkungen können allerdings nicht mit allen OLAP-Datenbanken vorgenommen werden und bedeuten in den meisten Fällen eine zeitraubende, manuelle Vorgehensweise, um die Datenbank den aktuellen Anforderungen anzupassen.
5. Änderung des Datenmodells durch Ersetzen von Dimensionen durch Attributierungen auf Ausprägungen.

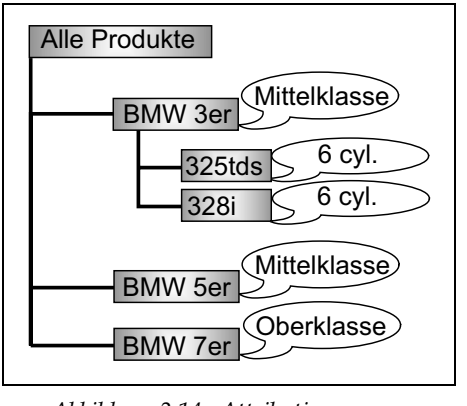


Abbildung 2.14: Attributierung von Ausprägungen einer Dimension

6. Erhöhung der Puffergrößen (Caches), sofern hierdurch keine Überschreitung des physikalisch vorhandenen Arbeitsspeichers eintritt, denn in diesem Fall wird auf den virtuellen Arbeitsspeicher der Festplatte zugegriffen. Dies führt zu sehr negativen Erscheinungen des Laufzeitverhaltens. Kontrollieren Sie regelmäßig den physikalisch vorhandenen Hauptspeicher über mitgelieferte Programme des Be-

triebssystems (z.B. perfmon.exe, der Task Manager unter Windows NT oder dem Dienstprogramm lsps unter IBM AIX zur Feststellung der Speichernutzung).

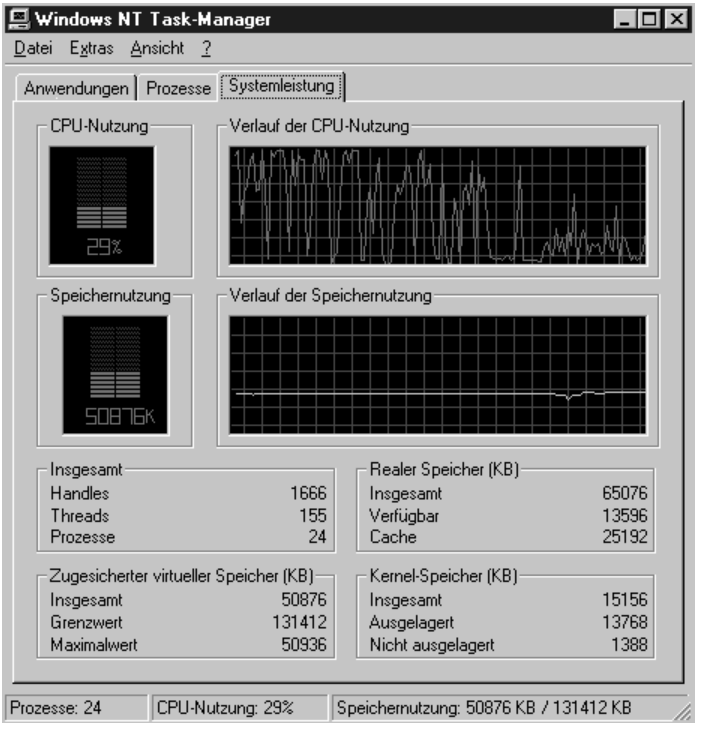


Abbildung 2.15: Windows NT Task Manager

7. Optimierung der Betriebssystem-Einstellungen. Hier wird häufig nicht genau genug auf optimale Einstellungen geachtet:
   - Windows NT Server erlaubt den für Fileserver- oder Netzwerk-Anwendungen optimierten Modus. Bei reinen Client/Server-Anwendungen, wie dem Betrieb einer OLAP-Datenbank, sollte die standardmäßige Einstellung „Durchsatz für Netzwerkanwendungen maximieren“ gewählt werden, da sonst bei jeder Abfrage weiterer Hauptspeicher allokiert und möglicherweise nicht wieder freigegeben wird (siehe Abbildung 2.16).
   - IBM AIX stellt betriebssystemseitig für Anwendungen nur 64 MB zur Verfügung. Um den verfügbaren RAM-Bereich zu erhöhen, sollte das Kommando „makelarge“ ausgeführt werden.

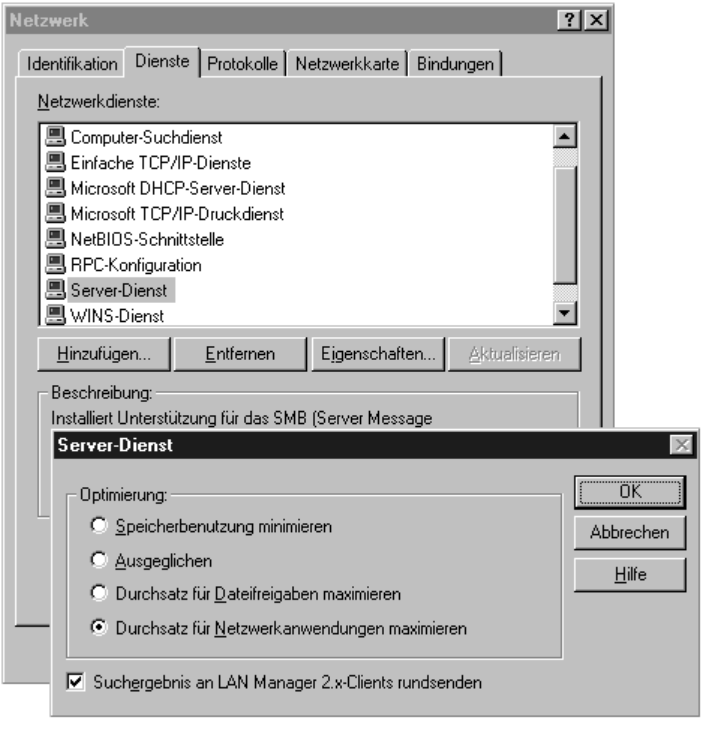


Abbildung 2.16: Optimale Einstellung des NT-Servers für Client/Server-Anwendungen

8. Prüfung der Hardware-Nutzung: durch Einsatz von RAID-Baugruppen (Redundant Array of Inexpensive Disks) läßt sich eine Steigerung der Verarbeitungsgeschwindigkeiten, speziell des Kalkulationsprozesses und der notwendigen Schreibzugriffe auf den Festplattenspeicher, erreichen. RAID bedeutet: mehrere (preisgünstige) Festplatten werden miteinander verschaltet, so daß zum einen eine höhere Zugriffsgeschwindigkeit und zum anderen eine niedrigere Ausfallrate des Gesamtsystems erreicht werden kann. Bei RAID-Arrays wird im wesentlichen zwischen 5 verschiedenen Varianten unterschieden, deren Numerierung keine Klassifizierung darstellt[1]:
   - RAID 0 (Striping) – hierbei wird ein sog. Stripe-Set, Einzelteile des zu speichernden Datenbestandes, aufgebaut, welches über die einzelnen Elemente des Arrays gleichmäßig verteilt wird. Diese Me-

1. Weitere Informationen zu RAID und den RAID-Varianten finden Sie unter *http://www.adtx.co.jp/english/raid/raid0.html*.

thode baut keine Redundanz des Datenbestandes auf, daher kann der Ausfall eines einzelnen Elements des Sets einen vollständigen Verlust der Daten bewirken. Durch die Parallelität werden dafür die Lese- und Schreibzugriffe mit dieser Methode minimiert und bewirken daher eine Steigerung des Datendurchsatzes.

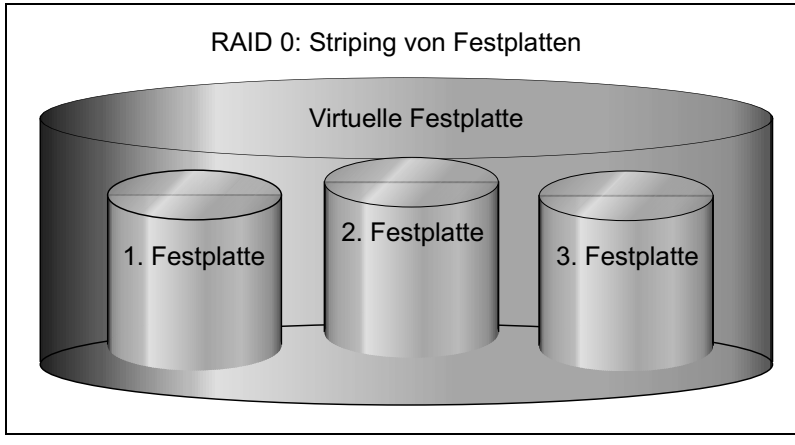


Abbildung 2.17: RAID 0: Erscheinung der Einzelplatten als gesamte, physikalisch nicht vorhandene („virtuelle") Festplatte

- RAID 1 (Mirroring) – mindestens 2 der Elemente eines Arrays werden gespiegelt beschrieben und gelesen. Diese Methode ermöglicht eine hohe Verfügbarkeit und in einigen Fällen eine erhöhte Lese- und Schreibgeschwindigkeit.

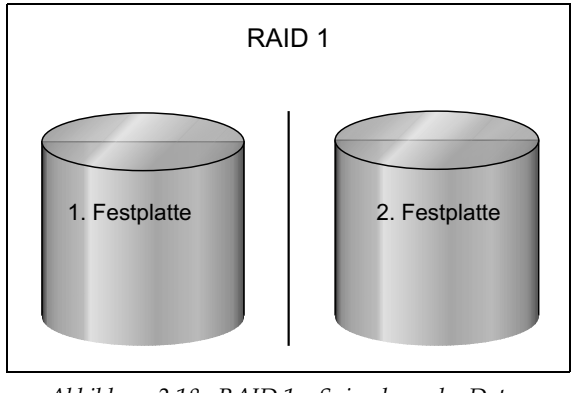


Abbildung 2.18: RAID 1 – Spiegelung der Daten auf mehrere Festplatten

- RAID 4 (Parity auf zusätzliche Festplatte) – mittels der Parity-Festplatte kann bei einem Versagen eines der RAID-Elemente eine Reorganisation der Daten vorgenommen werden und somit eine Redundanz aller eingesetzten RAID-Elemente vermieden werden. RAID 4 stellt eine nicht-optimale Methode hinsichtlich der Zugriffsgeschwindigkeit aufgrund des einfach vorhandenen Parity-Bereichs auf einer physikalischen Festplatte dar.
- RAID 5 (Parity auf verschiedenen Festplatten) – wie bei RAID 4 wird ein Parity-Bereich beschrieben, der für eine sichere Datenhaltung sorgt. Bei RAID 5 wird dieser Bereich allerdings auf mehrere Festplatten des RAID verteilt, so daß eine erhöhte Geschwindigkeit, insbesondere beim Schreibzugriff, erreicht werden kann.

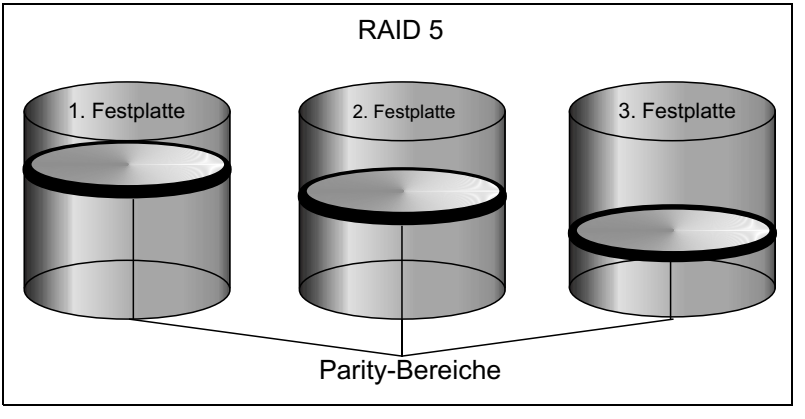


Abbildung 2.19: Bei RAID 5 werden die Parity-Bereiche auf die einzelnen Festplatten verteilt

- RAID 10 ist eine kombinierte Methode aus RAID 1 (Mirroring) und RAID 0 (Striping), womit die Vorteile beider Methoden kombiniert werden: Redundanz und Zugriffsgeschwindigkeit. RAID 10 ist die für zugriffsintensive Anwendungen, wie OLAP-Datenbanken, optimale Methode.

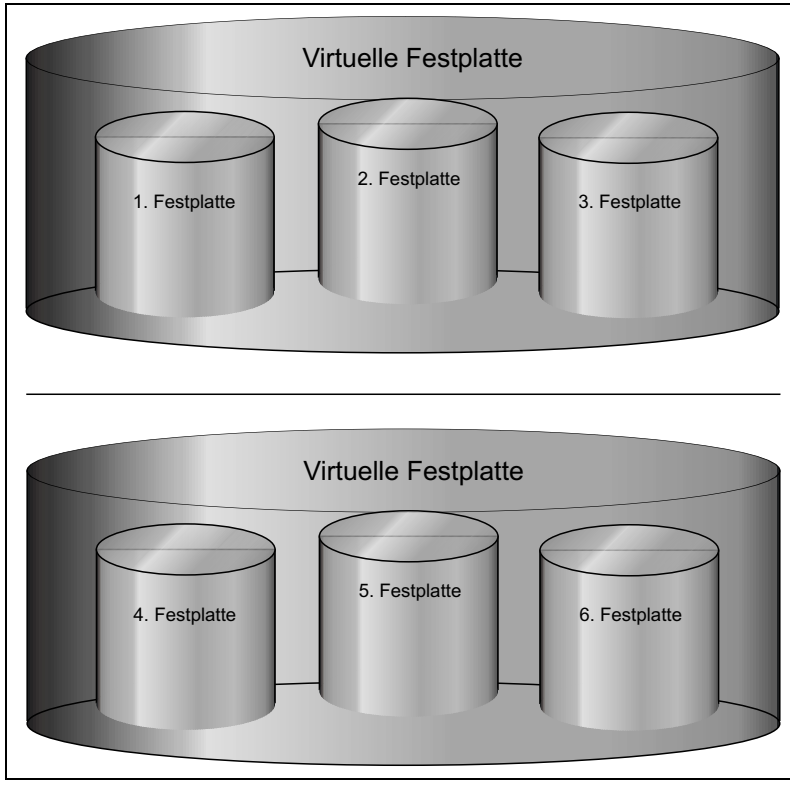


Abbildung 2.20: RAID 10 (gespiegeltes Stripe-Set)

Weitere Informationen und produktspezifische Besonderheiten zu RAID-Systemen finden Sie beim „RAID Advisory Board“, dem Zusammenschluß wichtiger Hersteller von RAID-Systemen, unter *http://www.raid-advisory.com/*.

2.5 Integration multimedialer Elemente (Audio, Video, Internet)

Die Vernetzung weiterer Informationsquellen mit Datenbanken ist im heutigen Unternehmensumfeld notwendig und sinnvoll. Häufig wird in diesem Zusammenhang von sogenannten multimedialen Elementen, wie Bild-, Ton-, Video- und sonstigen Binärdateien ausgegangen, die eine Datenbank-Anwendung in Ihrer Funktionsweise wesentlich erweitern können. So können in einer Produkt-Datenbank Bilder und Zeichnungen

zu den einzelnen Produkten abgelegt werden, so daß eine verbesserte Bedienerführung auch für unerfahrene Anwender möglich ist. Ebenso ist es vorstellbar, in einer Controlling-Datenbank den Planungsverlauf mit den erforderlichen Notizen, Word-Dateien etc. festzuhalten und nachvollziehbar zu machen.

Auch OLAP-Datenbanken lassen sich um die Möglichkeiten der Multimedia-Integration erweitern, dies ist allerdings sehr stark vom jeweiligen Produkt abhängig. Auch ändert sich versionsbedingt an den Eigenschaften der jeweiligen Produkte häufig etwas, weshalb in diesem Zusammenhang auf eine produktspezifische Gegenüberstellung verzichtet wird. Wie in der Software-Industrie üblich besteht zwischen Aussage und Wirklichkeit in vielen Fällen eine Lücke, die Sie durch dritte Quellen wie den OLAP-Report, der allerdings nur über ein entsprechend zu bezahlendes Abonnement zu beziehen ist, das OLAP-Council, den Zusammenschluß wichtiger OLAP-Anbieter oder die OLAP-Newsgroup überprüfen können[1].

2.5.1 Frontend-basierte Integration multimedialer Daten

Die Einbeziehung von Internet-Ressourcen, wie z.B. Web-Pages, spielt natürlich auch eine Rolle in der stetigen Vernetzung der Informationsquellen im Unternehmen. In der Regel gestatten mögliche Frontends zu OLAP-Datenbanken, wie Microsoft Excel, Cognos Powerplay oder arcplan inSight allerdings eine bessere Integration als die zugrundeliegende Datenbank-Technologien. Der wesentliche Nachteil bei der Frontend-orientierten Integration liegt bei den dann fehlenden Client/Server-Fähigkeiten, wie z.B. erhöhtem Netzwerkverkehr und der schlechteren Wartbarkeit des Gesamtsystems, da jeder Anwender „seine“ Anwendung auf den Desktop distribuiert bekommen muß. Weitere Informationen zu Cognos Powerplay und arcplan inSight finden Sie unter *http://www. cognos.com* und *http://www.arcplan.com*.

1. *http://www.olapreport.com/* sowie *news://comp.databases.olap/* und *http://www.olapcouncil.org/*

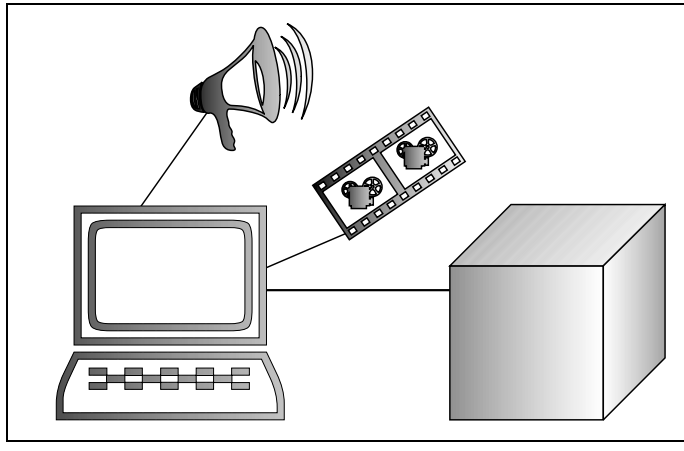


Abbildung 2.21: Frontend-basierte Integration multimedialer Daten

2.5.2 Server-basierte Integration multimedialer Daten

Die Server-basierte Integration erlaubt eine bessere Kontrolle der Datenbestände hinsichtlich

- **Größe**

  Die integrierten multimedialen Daten, besonders Videos, benötigen erheblichen Speicherplatz, so daß sinnvolle Beschränkungen pro Anwender und Datenbank aufgelegt werden sollten.

- **Zugriffsvergabe**

  Die Server-basierte Anwendung erlaubt eine bessere Kontrolle der Rechte auf Elemente wie Lesen, Schreiben und Ausführbarkeit.

- **Versionierung**

  Die Versionierung behandelt die Aktualität der abgespeicherten Informationen und sollte wesentlicher Bestandteil eines Implementierunsgkonzeptes multimedialer Datenquellen sein. Ein entsprechendes Konzept sichert die höchstmögliche Aktualität und eine evtl. notwendige historische Sichtweise der abgespeicherten Informationen.

- **Redundanz**

  Redundanzen (mehrfache Speicherung gleicher Informationen) lassen sich aufgrund teilweise notwendiger Performance-Betrachtungen (z.B. sind Niederlassungen eines Unternehmens nur über eine Leitung geringerer Bandbreite mit der Zentrale verbunden, benötigen

aber dieselben Informationen) zwar nicht gänzlich ausschließen, können aber durch die Server-basierte gegenüber der Client-basierten Speicherung auf ein Minimum reduziert werden.

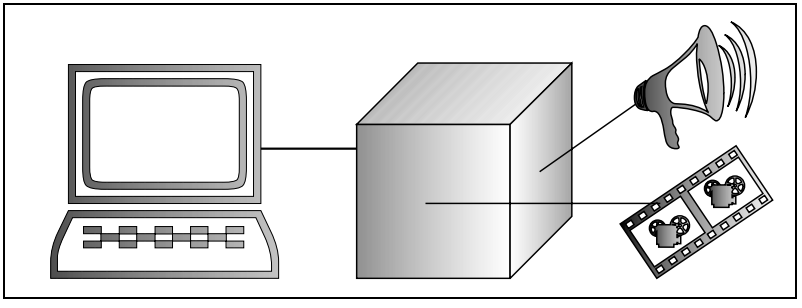


Abbildung 2.22: Server-basierte Integration multimedialer Daten