



GMD Research Series

GMD –
Forschungszentrum
Informationstechnik
GmbH

Thorsten Gajewski
Thomas Klement

Generierung eines multidimensionalen Katalogsystems mit Datawarehouse- Architektur

© GMD 2000

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin, Germany
Telefon +49 -2241 -14 -0, Telefax +49 -2241 -14 -2618
<http://www.gmd.de>

In der Reihe GMD Research Series werden Forschungs- und Entwicklungsergebnisse aus der GMD zum wissenschaftlichen, nichtkommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Research Series is the dissemination of research work for scientific non-commercial use. The commercial distribution of this document is prohibited, as is any modification of its content.

Anschriften der Verfasser/Addresses of the authors:

Thorsten Gajewski
Im Busch 13
D-64297 Darmstadt
E-mail: Gajewski@danet.de

Thomas Klement
Institut für Integrierte Publikations- und Informationssysteme
GMD – Forschungszentrum Informationstechnik GmbH
Dolivostraße 15
D-64293 Darmstadt
E-mail: Thomas.Klement@gmd.de

Die Deutsche Bibliothek - CIP-Einheitsaufnahme:

Gajewski, Thorsten:

Generierung eines multidimensionalen Katalogsystems mit
Datawarehouse-Architektur / Thorsten Gajewski ; Thomas Klement.

GMD – Forschungszentrum Informationstechnik GmbH. - Sankt Augustin :

GMD – Forschungszentrum Informationstechnik, 2000

(GMD Research Series; 2000, No. 2)

Zugl.: Darmstadt, Techn. Univ., Diplomarbeit T. Gajewski, 1999

ISBN 3-88457-373-X

ISSN 1435-2699

ISBN 3-88457-373-X

Zusammenfassung

Die Bedeutung elektronischer Katalogsysteme im Internet wächst. Gleichzeitig besteht der Bedarf komplexere Analysemöglichkeiten für Produktpräsentationen zu realisieren. Das Projekt zur Erstellung eines multidimensionalen Katalogsystems kombiniert die Eigenschaften elektronischer Kataloge mit den Vorteilen einer Data Warehouse-Architektur. Im Rahmen der Diplomarbeit wird die Generierung einer multidimensionalen Datenbank anhand der im XML-Format spezifizierten Katalog-Metadaten implementiert.

Eine Katalog mit Aggregatwerten für Last Minute-Reisen soll als Anwendungsszenario die Vorteile eines solchen multidimensionalen Katalogsystems aufzeigen.

Schlagwörter: Katalogsystem, Data Warehouse, OLAP, multidimensional, XML

Abstract

The importance of electronic catalogs in the internet is growing. At the same time there is the requirement to realize complex analysis for product presentation. The project of creating a multidimensional catalog combines the properties of electronic catalogs with the advantages of a Data Warehouse architecture. Within the context of this degree dissertation there is the implementation to create a multidimensional database with the help of the catalog's metadata in the form of XML.

A catalog with aggregation values for last minute travel presents the advantages of such a multidimensional catalog.

Keywords: Catalog, Data Warehouse, OLAP, multidimensional, XML

Inhaltsverzeichnis

1 Einleitung	7
1.1 Motivation und Ziel	7
1.2 Anwendungsszenario	9
2 Theoretische Grundlagen	11
2.1 Multidimensionales Katalogsystem.....	11
2.2 OLAP	12
2.2.1 Relationale und multidimensionale OLAP Systeme	12
2.2.2 Konzept und Begriffe	13
2.2.3 Metadaten	15
2.2.4 MDX.....	16
3 Verwendete Technologien	17
3.1 Entwicklungsumgebung.....	17
3.1.1 XML	17
3.1.2 XML for Java Parser.....	18
3.1.3 DOM.....	19
3.1.4 MS Developer Studio	20
3.2 Datenbankserver	21
3.2.1 MS SQL-Server	21
3.2.2 MS OLAP-Server	21
3.3 Schnittstellen.....	25
3.3.1 MS COM	25
3.3.2 MS UDA.....	26
4 Entwurf	28
4.1 Anforderungen	28
4.2 Datenmodell.....	28
4.3 Ablaufbeschreibung	29
4.4 System-Architektur	32
5 Beispiel	34
5.1 Motivation.....	34
5.2 Katalog-Struktur	35
6 Ausblick	37
Literaturverzeichnis	I

Anhang A: XML DTD..... IV

Anhang B: XML-Dokument..... VII

Abbildungsverzeichnis

Abbildung 1: Architektur des multidimensionalen Katalogsystems	8
Abbildung 2: Begriffe des OLAP-Modells	13
Abbildung 3: Beispiel für Aggregation	14
Abbildung 4: MS OLAP-Server	23
Abbildung 5: DSO-Objektmodell.....	24
Abbildung 6: Objektmodell der Metadaten	29
Abbildung 7: Transformation vom relationalen in das multidimensionale Modell	30
Abbildung 8: Architektur des Systems	32
Abbildung 9: Struktur des Last Minute-Cube	36

1 Einleitung

Das Thema der Diplomarbeit ist Teil eines übergeordneten Projektes, dessen Ziel es ist, ein multidimensionales Katalogsystem zu erstellen. Als Anwendungsszenario für ein solches Katalogsystem wurde ein Katalog für Last Minute Reisen gewählt. Der Schwerpunkt der Arbeit liegt in der Generierung eines Data Warehouse mit Hilfe der XML-Technologie. Auf der Basis des Data Warehouse soll ein multidimensionales Katalogsystem betrieben werden.

Im Anschluß an das einleitende Kapitel, welches eine Beschreibung des Gesamtprojektes beinhaltet, wird ein Überblick über die theoretischen Grundlagen gegeben. Das 3. Kapitel stellt die verwendete Technologie zusammen, die im Rahmen dieser Arbeit eingesetzt wurde. Anschließend wird der realisierte Entwurf des Systems erläutert, während detailliertere Informationen zur XML-Implementierung im Anhang zu finden sind. Im 5. Kapitel wird als Beispiel das Szenario Last Minute-Reisen näher vorgestellt, und das letzte Kapitel gibt einen Ausblick auf die weitere Entwicklung des Projektes.

1.1 Motivation und Ziel

Das Internet stellt als dynamisch wachsendes Medium für viele Bereiche riesige Informationsmengen zur Verfügung. Diese Informationen sind aufgrund ihrer Verteilung, der heterogenen Speicherform und der unterschiedlichen Präsentation schwer systematisch zu nutzen.

Das hier vorgestellte System eines multidimensionalen Katalogs bietet die Möglichkeit der Visualisierung und Analyse der Informationen auf Basis eines Informationsanbieter übergreifenden Datenbestandes. Durch die Sammlung und Filterung der Daten aus dem World Wide Web ist es mit Hilfe der XML-Technologie möglich, das Angebot an Informationen automatisch zu konsolidieren und zu integrieren. Die Speicherung der Daten in einer multidimensionalen Datenbank kombiniert die Eigenschaften eines elektronischen Katalogsystems mit den Vorteilen eines Data Warehouse.

Zu den Leistungen eines elektronischen Katalogsystems gehört vor allem der einfache, menugesteuerte Zugriff auf Produktinformationen mit beliebigen Kriterien [CDM97].

Aufgrund der wachsenden Datenmengen und steigenden Anwenderzahlen bei elektronischen Katalogsystemen bietet sich der Einsatz von Data Warehouse Technologie an. [MER99] erkennt einen Trend beim Einsatz von Data Warehouses in der Speicherung von Benutzungsprofilen und Kundenprofilen. Die multidimensionale Speicherung in einem Data Warehouse eignet sich aber auch zur dynamischen Generierung von Katalogen.

In der Implementierung der Diplomarbeit erfolgt die Generierung des Data Warehouse durch die als XML-Dokument definierten Metadaten des Katalogs. Die Struktur des Katalog-Schemas ist entscheidend für die Analysemöglichkeiten und sollte mit geringem Aufwand anzupassen sein [BSH99]. Daher ist die XML-Schnittstelle aufgrund der Standardisierung in diesem Bereich und der gebotenen Flexibilität gut geeignet.

Eine Kombination der OLAP-Technologie mit den Möglichkeiten des Internet wird allgemein zukünftig die Basis zahlreicher Anwendungen darstellen [BS97].

Das Projekt CataCube, dessen Architektur in Abbildung 1 vereinfacht dargestellt ist, verfolgt das Ziel, ein solches multidimensionales Katalogsystem aufzubauen.¹

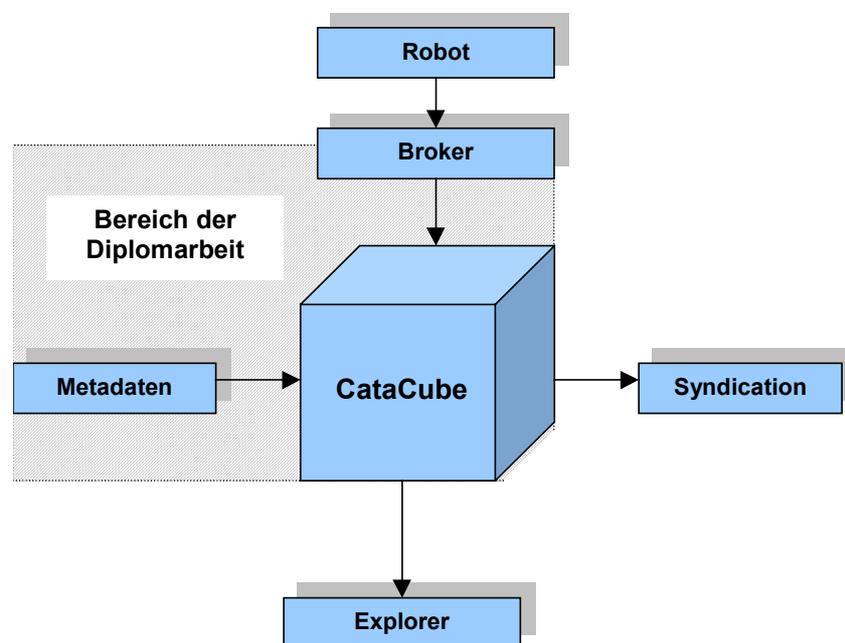


Abbildung 1: Architektur des multidimensionalen Katalogsystems

Es ist geplant, mit der Komponente „Robot“ die gewünschten Informationen von Webseiten zu parsen, zu sammeln und zu filtern. Die integrierten Ergebnisse sollen in XML-Form über die Komponente „Broker“ als Datenquelle in das System eingehen. Die Speicherung erfolgt mit Hilfe einer multidimensionalen Datenbank, die auf Basis der ebenfalls in XML spezifizierten Metadaten gebildet wird. Anschließend soll eine Komponente „Explorer“ die Visualisierung der Informationen und die Navigation im multidimensionalen Datenraum ermögli-

¹ Eine Beschreibung des Projektes ist unter der Adresse <http://www.darmstadt.gmd.de/~klement> verfügbar.

chen. Eine weitere Komponente („Syndication“) verfolgt das Ziel zusätzliche Dienste für verteilte Kataloge zur Verfügung zu stellen. Als Beispiel hierfür können Notifikationsdienste und das Abonnement von Katalogausschnitten auf Basis der Push-Technologie genannt werden.

Im Rahmen dieser Arbeit, die als Einstieg in das Projekt zu sehen ist, liegt der Schwerpunkt auf der Definition der Metadaten und der damit verbundenen Generierung der multidimensionalen Datenbank. Der zweite Bereich realisiert durch die Integration der Rohdaten und ihrer Speicherung in der Datenbank einen Teil der „Broker“ Komponente.

1.2 Anwendungsszenario

Um eine praxisorientierte Entwicklung des Projekts zu gewährleisten, wurde das folgende Anwendungsszenario entworfen, das die Vorteile eines solchen Katalogsystems aufzeigen soll.

Das Produkt Last Minute-Reise gehört zu den führenden Angeboten bei der Nutzung des Mediums Internet als Vertriebsweg. Zahlreiche Reiseveranstalter und Agenturen bieten dem Kunden zur Reiseplanung den Zugriff auf umfangreiche Datenbestände. Die angebotenen Systeme ermöglichen zwar meist die Angabe verschiedener Suchkriterien, jedoch ist dieses Angebot in der Regel sehr beschränkt und die Reihenfolge der Eingrenzung der Attribute statisch vorgegeben.

Zum Beispiel ist häufig eine Auswahl möglich, ob die Suche nach dem Reiseziel, dem Abflugdatum oder dem Abflughafen erfolgen soll. Anschließend jedoch wird das nächste Kriterium (in Abhängigkeit vom ersten Schritt) zur weiteren Eingrenzung fest vorgegeben. Wenn dieser Weg zu keinem gewünschtem Ergebnis führt, bleibt dem Kunden nur der Start einer neuen Suche über ein anderes Kriterium.

Das hier vorgestellte multidimensionale Last Minute-Katalogsystem bietet dem Anwender dagegen die flexible Suche über sämtliche Produkteigenschaften im Datenbestand. Die Navigation wird zusätzlich durch angegebene Aggregationswerte wie zum Beispiel den Durchschnittspreis der aktuell ausgewählten Angebote, unterstützt. Des Weiteren ist die Suche nicht auf einen Anbieter beschränkt, so dass Vergleiche möglich werden.

Die Datenstruktur der Angebote besitzt aufgrund ihrer voneinander unabhängigen Attribute einen multidimensionalen Charakter, was eine Kategorisierung des Produkts erleichtert und die Speicherung in Form eines Data Warehouse anbietet. Die Data Warehouse Architektur ermöglicht durch vorberechnete Aggregationswerte die flexible und komplexe Analyse der vorhandenen Informationen. Während einfache Anfragen der Art „Welche Angebote gibt es

für den 01.12.1999 für 2 Wochen ab Frankfurt?“ auch von bestehenden Systemen beantwortet werden können, lassen sich folgende Beispiele dort nicht mehr umsetzen:

- “Ist es günstiger im Dezember in die Südsee oder in die Karibik zu reisen?”
- “Welches sind die 10 günstigsten Fernreisen für 3 Wochen in den nächsten Tagen?”
- “Für welches Reiseziel gibt es im Moment die meisten Angebote mit einer 4 Sterne Unterkunft?”
- “Wie hoch ist der durchschnittliche Preis für 2 Personen mit Halbpension nach Mexiko im Winter?”

Für solche komplexen Anfragen sind momentan keine Systeme im Internet verfügbar. Die auf relationalen Datenbanken basierenden Katalogsysteme stoßen generell hier an ihre Grenzen. Die Umsetzung solcher Anfragen mit SQL hat wegen der Bildung von komplexen Joins einen klaren Nachteil gegenüber der Realisierung in Form einer multidimensionalen Speicherung in Verbindung mit einer entsprechenden Abfragesprache wie MDX (Multidimensional Expressions)².

Durch die Integration zusätzlicher Informationen und Dienste werden weitere Vorteile für das Katalogsystem erreicht. Ein Nutzer kann sich beispielsweise per E-Mail informieren lassen, wenn neue Angebote verfügbar sind oder ein bestimmtes Angebot eine gewünschte Preisgrenze erreicht hat.

² Vgl. Kapitel 2.2.4.

2 Theoretische Grundlagen

Dieses Kapitel beschreibt die grundlegenden Begriffe und Konzepte der Bereiche Katalogsysteme und OLAP-Technologie.

2.1 Multidimensionales Katalogsystem

Ein elektronischer Katalog stellt ein System dar, das eine effiziente und strukturierte Informationsspeicherung mit einer komfortablen und flexiblen Visualisierung der Informationen verbindet. Diese elektronischen Katalogsysteme beruhen auf relationalen Datenbanken und haben das Ziel, die Informationen in einer konsistenten Form und hierarchisch nach Kategorien strukturiert, zugänglich zu machen. Hauptanwendungsgebiete solcher Systeme sind Produktkataloge, die auch als Online-Kataloge bezeichnet werden und durch den wachsenden Markt des Electronic Commerce einen großen Stellenwert besitzen. Besonders im Bereich des Business-to-Consumer-Commerce (B2C) dient der elektronische Katalog als Schnittstelle zwischen Unternehmen und Kunde, um Produkte zu präsentieren. Dabei werden verschiedene Attribute der Produkte wie Beschreibung, Merkmale und Preis als Suchkriterien bereitgestellt [CDM97].

Das Hauptmerkmal der Kataloge ist die Möglichkeit für den Anwender, den Katalog mit Hilfe entsprechender Werkzeuge selbst zu konfigurieren. Neben Inhalt und Layout läßt sich auch die Struktur des Katalogs durch Definition der Produktattribute individuell und flexibel festlegen [MER99].

Die Marktsituation der elektronischen Kataloge ist nach [MER99] durch die Vielzahl herstellerepezifischer, heterogener Kataloge gekennzeichnet. Vor allem im Interesse der Kunden wäre es jedoch wünschenswert, einen einheitlichen Standard zu schaffen, der eine schnelle, einfache und universell gültige Zugriffsform gewährleistet. Neben der Verbesserung der Bedienungskriterien würde hiermit ein Vergleich zwischen Produkten verschiedener Anbieter erleichtert [CDM97].

Ein multidimensionales Katalogsystem erweitert konventionelle elektronische Katalogsysteme konzeptuell um die Möglichkeit multiple Kataloghierarchien über den Produktdaten zu modellieren. Die Kataloghierarchien (Dimensionen) leiten sich aus den Produktattributen ab, die als assoziative Metadaten genutzt werden. Auf Basis eines Data Warehouse wird durch zusätzliche Werkzeuge dem Nutzer die Navigation in einem multidimensionalen Datenraum ermöglicht.

2.2 OLAP

Die Bezeichnung OLAP (Online Analytical Processing) hat sich auf dem Gebiet der multidimensionalen Datenanalyse durchgesetzt. OLAP-Datenbanken unterscheiden sich von operativen, transaktionsbestimmten Datenbankanwendungen, die mit OLTP (Online Transaction Processing) bezeichnet werden, durch die Aktualität des Datenbestandes. OLAP Anwendungen basieren auf historischen Daten, die zur Analyse konsolidiert werden [KE97]. Es hat sich gezeigt, dass die relationalen Datenbanken in Verbindung mit der Abfragesprache SQL nicht ohne weiteres für komplexere Analysen geeignet sind und ein Bedarf für entsprechende Werkzeuge besteht [BS97]. Vor allem fehlende Funktionalität und mangelnde Performance sind die Hauptargumente für die neuen Ansätze.

Typische Anwendungsgebiete für OLAP sind Management-Informationssysteme oder Decision-Support-Systeme. Hierbei werden alle relevanten Daten in eine separate Datenbank integriert, um den speziellen Bedürfnissen der Analyse gerecht zu werden. Diese physischen Datenbanken werden als Data Warehouse bezeichnet, während der Begriff OLAP häufig nur für das Analyse-Werkzeug für den Benutzer oder den Client steht.³ Hauptaufgaben eines OLAP-Systems sind die einfache Suche, Darstellung und Navigation innerhalb eines bestimmten Datenraumes. Weitere Leistungen stehen im Zusammenhang mit der Abfrage bestimmter Teilmengen und Beziehungen der Daten, sowie der automatischen Berechnung komplexer, häufig statistischer Funktionen auf Basis des Datenbestandes [MSO].

Der Markt für OLAP-Systeme verzeichnet hohe Wachstumsraten. Die Funktionalität der Produkte wird stetig erweitert, und immer mehr Unternehmen drängen mit neuen Lösungen auf den Markt [DSH98].

2.2.1 Relationale und multidimensionale OLAP Systeme

Bei der Architektur von OLAP Systemen läßt sich eine Unterteilung nach der Art der Speicherung der Daten vornehmen. Für den Fall der relationalen Datenbank als Speicherform, bei dem die Daten in Tabellen gehalten werden, wird der Begriff ROLAP (Relational OLAP) verwendet. Die Anfragen an das OLAP-System werden hierbei als SQL-Anfragen an das zugrundeliegende relationale Datenbankmanagementsystem übertragen. Werden die Daten hingegen analog zum Datenmodell in mehrdimensionalen Datenstrukturen gespeichert, so handelt es sich um MOLAP-Systeme (Multidimensional OLAP).

Die bessere Performance bezüglich der Antwortzeiten wird bei MOLAP durch den Nachteil erkauft, dass der Speicherbedarf bei wachsenden Datenbeständen schneller wächst als bei

³ Die Terminologie in diesem Bereich ist in der Literatur nicht einheitlich. Vgl hierzu u.a. [BRO99] oder [THO97].

ROLAP-Datenbanken. Eine Ursache für den relativ hohen Speicherbedarf ist der mehrdimensionale Informationsraum, der in der Regel nur sehr gering gefüllt ist (je nach Anwendung ca. 1% bis 5%).⁴

Die dritte Variante besteht aus einem hybriden Konzept (HOLAP), das eine Zwischenform von ROLAP und MOLAP verwendet. Hierbei werden die Detaildaten zwar in relationaler Form gespeichert, vorberechnete Aggregationen jedoch als Datacube gespeichert [BRO99].

2.2.2 Konzept und Begriffe

Die Basis des OLAP-Konzeptes und der multidimensionalen Datenanalyse bildet der „Datacube“. Die für den dreidimensionalen Fall noch sehr anschauliche Darstellung der Datenhaltung läßt sich um eine beliebige Anzahl von Dimensionen erweitern, die einen Datenraum definieren.

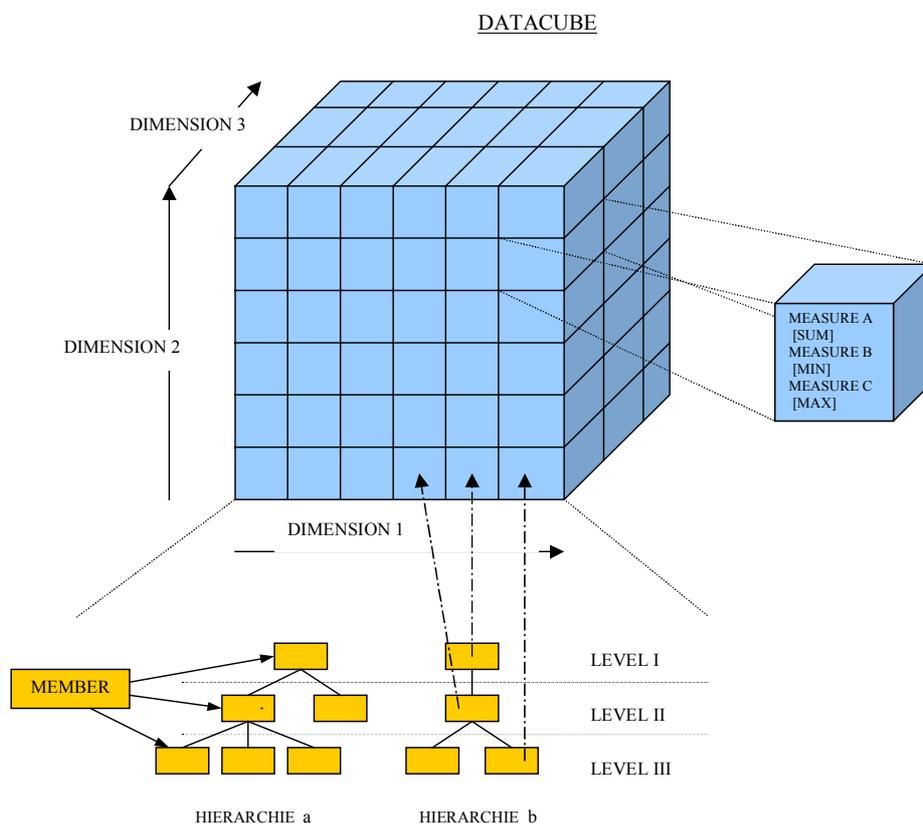


Abbildung 2: Begriffe des OLAP-Modells

⁴ Es sind jedoch bereits OLAP-Server verfügbar, die für leere Zellen keinen Speicherplatz benötigen. Auch der im Rahmen dieser Arbeit eingesetzte MS-OLAP-Server erfüllt dieses Kriterium.

2. Theoretische Grundlagen

Jede dieser Dimensionen kann über mehrere Hierarchien verfügen, die jeweils die Dimension unterschiedlich abbilden. Ein Beispiel für eine Dimension, die über mehrere Hierarchien verfügt, ist die Zeitdimension. Hier ist es häufig sinnvoll die Zeit nach Jahren, Monaten und Tagen aufzuteilen und parallel dazu eine Gliederung nach Jahren, Wochen und Tagen vorzunehmen. Durch die fehlende Möglichkeit Wochen eindeutig einem Monat zuzuordnen ist die Abbildung innerhalb einer Hierarchie nicht möglich.

Die Untergliederung der Dimension erfolgt für jede Hierarchie mit Hilfe mehrerer Ebenen (Level). Die einzelnen Elemente der Dimension, die exakt einer Ebene angehören, werden als „Member“ bezeichnet. Charakteristisch für eine OLAP-Modellierung ist eine Dimension, die hierarchisch über mehrere Levels abgebildet wird. Die zugehörige Struktur der Members wird in diesem Fall als Baum modelliert.

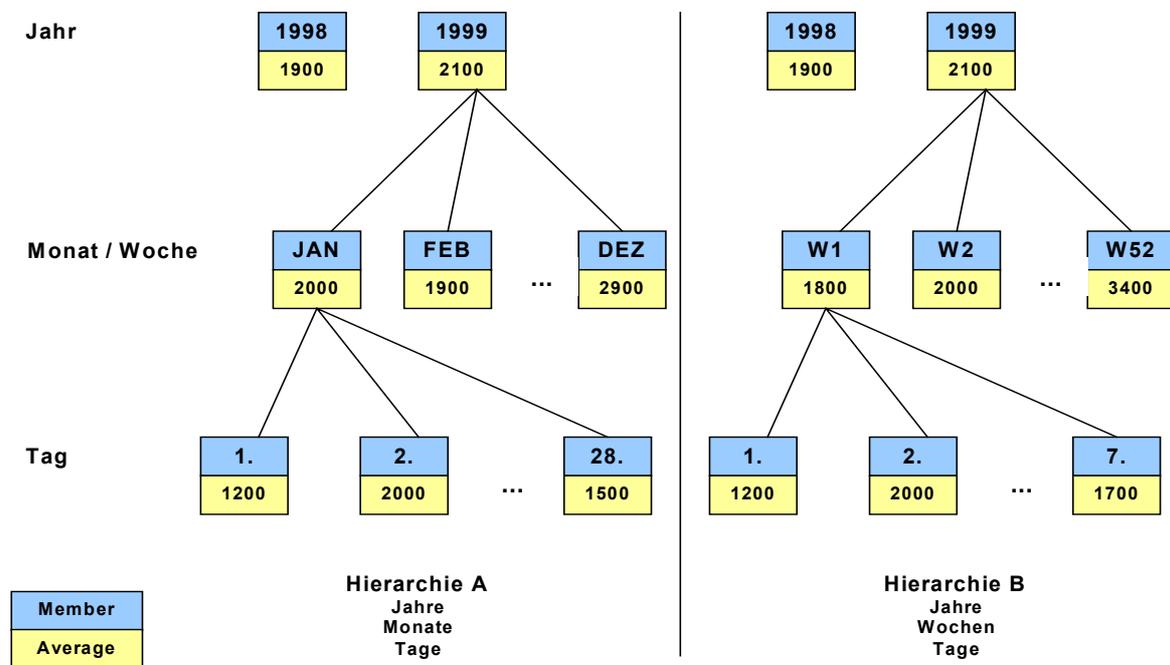


Abbildung 3: Beispiel für Aggregation

Die möglichen Kombinationen der Members aus den vorhandenen Dimensionen referenzieren Zellen in dem durch die Dimensionen aufgespannten Datenraum. Diese Zellen beinhalten die „Measures“, also die Werte, die entweder als Detaildaten angezeigt, oder bei der Navigation entsprechend aggregiert werden. Typische Aggregationsarten sind die Summenbildung, die Bildung eines Maximums oder Minimums, sowie das Aufzählen der Werte oder die Berech-

nung des Durchschnitts. Abbildung 3 zeigt eine Beispieldarstellung für die Aggregation der Durchschnittspreise für zwei Hierarchien der Zeitdimension.

Die Definition einer Menge von Dimensionen und einer Menge von Measures bilden zusammen den Datacube.

Das OLAP-Konzept bietet auch die Möglichkeit „Virtual Cube“ zu verwenden. Als Äquivalent zu einer View im relationalen Modell kann ein Virtual Cube durch die Definition von Join-Verbindungen über reale Datacubes gebildet werden. Während für den Nutzer beim Zugriff auf den Datacube kein Unterschied besteht, enthält der Virtual Cube tatsächlich keine materialisierten Daten, sondern wird durch Anfragen auf Datacubes gebildet.

Bei den Analyseanfragen werden folgende Operationen unterschieden:

- Roll-up Aggregation von Dimensionen zur Verdichtung der Daten
- Drill-down Gegensätzliche Bewegung zur Gewinnung von Details
- Slicing Bildung von Teilmengen durch Reduzierung der Dimensionen
- Dicing Bildung von Teilmengen allgemein (Sub-Cube)
- Pivot Reorganisation der Ergebnisdaten durch Vertauschen von Dimensionen

2.2.3 Metadaten

Die Grundlage eines OLAP-Systems bildet die Definition seiner Metadaten. Die schnelle und weit verzweigte Entwicklung des OLAP-Marktes hat jedoch dazu geführt, dass sich zahlreiche verschiedene Formen und Ausprägungen im Umgang mit Metadaten gebildet haben [BS97]. Da die Arbeit mit OLAP-Systemen den Zugriff und die Manipulation dieser Metadaten erfordert, wäre eine Standardisierung für die weitere Entwicklung in diesem Bereich wünschenswert. Ein Ansatz der Metadata Interchange Specification Initiative (MDIS) versucht dieses Ziel durch die Definition eines Standardschemas zum Austausch von Metadaten zu erreichen [MDC].

Grundsätzlich sind bei den OLAP-Metadaten die technischen Metadaten und die anwendungsbezogenen Metadaten zu unterscheiden [BS97].

Die technischen Metadaten, die auch als Metadaten der Metadaten bezeichnet werden, beschreiben den Aufbau und die Wartung der Datenbank und damit das Schema des Systems. Diese für den Administrator relevante Ebene enthält an erster Stelle die Informationen über die Datenquellen des OLAP-Systems. Hierzu gehören beispielsweise der Speicherort, die

Form der Datenhaltung sowie benötigte Schnittstelleninformationen über die Werte der Datenbank, sowie über die anwendungsspezifischen Metadaten.

Die Anwendungs-Metadaten beschreiben als eine Menge von Objekten das Schema der zugrunde liegenden Anwendung. Diese Informationen zum Aufbau des Data Warehouse werden in Tabellen einer relationalen Datenbank gehalten. Selbst bei der Verwendung des MOLAP-Prinzips werden diese relationalen Tabellen für die Erzeugung und Aktualisierung des Datacube benötigt. Das Schema sieht eine „fact-table“ und mehrere Dimensionstabellen vor, die dem OLAP-System alle relevanten Daten zur Verfügung stellen. Die fact-table beschreibt die Zellen eines Datacube in der Form, dass in jeder Zeile die Werte der Measures und ein Fremdschlüssel für jede vorhandene Dimension existiert. Diese Einträge verweisen auf die Dimensionstabellen, in denen die Ebenen und Elemente der Dimensionen bestimmt sind.

Bei der Gestaltung des Datenmodells gibt es zwei grundsätzliche Entwurfsarten. Am häufigsten wird das „star schema“ verwendet. Dort sieht das Schema exakt eine nicht normalisierte Tabelle pro Dimension vor, die einen Eintrag für jedes Element auf der untersten Ebene besitzt. Die Alternative des „snowflake schema“ enthält dagegen durch die Normalisierung der Dimensionstabellen mehrere miteinander verbundene Tabellen für jede Dimension. Beim snowflake schema wird durch Redundanzvermeidung der Speicherbedarf bei umfangreichen Dimensionen erheblich verringert. Im Gegenzug wird durch die Verwendung zusätzlicher Joins als Kombination der normalisierten Dimensionstabellen ein Nachteil in Kauf genommen.

2.2.4 MDX

MDX (Multidimensional Expressions) ist eine OLAP-Anfragesprache von Microsoft, die gute Chancen besitzt zum Standard der multidimensionalen Abfragesprachen zu werden. Durch die Integration in das Konzept OLE DB for OLAP ist MDX überall dort verfügbar, wo ein Datenbank-Provider diese Schnittstelle unterstützt. In Anlehnung an SQL, was sowohl die Syntax als auch die Umsetzung betrifft, ermöglicht MDX durch die spezielle Ausrichtung auf OLAP Bedürfnisse eine einfache Formulierung von komplexen Anfragen.

Im Unterschied zu SQL ist das Ergebnis einer Abfrage mit Select kein zweidimensionaler „rowset“, sondern ein multidimensionaler „dataset“, der als Teilmenge eines Datacube gebildet wird [OLE]. Durch dieses Konzept stehen für das Ergebnis einer Abfrage wieder alle multidimensionalen Analysemöglichkeiten auf Client-Seite zur Verfügung.

3 Verwendete Technologien

Das Kapitel beschreibt die im Rahmen dieser Arbeit eingesetzten Technologien und Produkte. Der erste Abschnitt stellt kurz die Entwicklungsumgebung dar, die aus der Kombination von XML, Java und Visual Basic besteht. Das Kapitel 3.2 erläutert die verwendeten Datenbanksysteme, den Microsoft SQL-Server und dessen Komponente den Microsoft OLAP-Server.

Das Kapitel schließt mit einem Überblick über die Schnittstellen zwischen diesen Komponenten, die auf verschiedenen Microsoft-Technologien basieren.

3.1 Entwicklungsumgebung

Die gesamte Entwicklung erfolgte auf einem Windows-NT System. Neben den unten beschriebenen Komponenten sind das Microsoft Software Development Kit (SDK) for Java und in Verbindung hiermit die Microsoft Virtual Machine for Java installiert worden. Die Microsoft VM [MVM] stellt einen eigenen Java-Interpreter für die Ausführung von Java-Applikationen dar, der darüber hinaus das Component Object Model⁵ unterstützt und somit die COM API für Java realisiert.

3.1.1 XML

Die Metasprache XML ist vom „World Wide Web Consortium“ (W3C) für die anwendungsneutrale Beschreibung und den Austausch strukturierter Informationen entwickelt worden [W3C98a]. Eine Einführung in das Konzept und die wichtigsten Kriterien beim Einsatz von XML sind umfassend in [GOL98] zusammengestellt.

Mit XML ist es möglich, das benötigte Format für Anwendungsdaten frei zu definieren. Die Document Type Definition (DTD) definiert eine Menge von Elementen und Regeln für XML-Dokumente. Anhand dieser Anleitung können anschließend Instanzen (XML-Dokumente) erzeugt werden, die allen Regeln der DTD entsprechen müssen. Die Designaspekte bei der Erstellung einer DTD sind ausführlich in [MEG98] behandelt. Ein großer Vorteil von XML ist das Fehlen jeglicher Darstellungsinformationen innerhalb eines Dokuments und damit die exakte Trennung zwischen Inhalt und Präsentation.

⁵ Siehe auch Kapitel 3.3.1.

Die Integration von XML-Dokumenten in eine Anwendung erfordert einen Parser, der die Gültigkeit des Dokuments mit der vorhandenen DTD überprüft und den Zugriff auf das Dokument durch das Erzeugen des entsprechenden Objekts (SAX oder DOM⁶) ermöglicht.

Bei SAX (Simple API for XML) handelt es sich um eine Schnittstelle, die ein XML-Dokument als Sequenz von Ereignissen (Events) abbildet. Der Parser erzeugt beim Traversieren des Dokuments je nach gefundenem Elementtyp einen „Event“. Die Nutzung von SAX erfordert die Erstellung einer Java-Klasse, die Methoden bereitstellt um auf die Events zu reagieren und, die ein eigenes anwendungsspezifisches Objektmodell erzeugt.

Für die Implementierung der Diplomarbeit ist DOM verwendet worden. Die Struktur der Katalog-Daten besitzt einen hierarchischen Charakter und entspricht der Baumstruktur von DOM. SAX gilt als einfach und schnell und besitzt dann Vorteile, wenn individuelle Strukturen benötigt werden oder nur Teile der XML-Dokumente verarbeitet werden sollen. Für den Katalog ist es jedoch immer erforderlich die gesamte Struktur als Objektmodell zu erzeugen.

Für XML existieren momentan mehrere frei verfügbare Parser in verschiedenen Sprachen. Die meisten sind in Java erstellt worden und bilden die Grundlage für den häufigen Einsatz der Kombination von Java und XML. Die Hauptvorteile solcher Anwendungen sind in der Objektorientierung und Plattformunabhängigkeit zu sehen [LEV98].

Für Anwendungen zur Extrahierung von Daten aus Web Seiten, wie bei [LHB99] beschrieben, ist XML notwendig, weil Web Seiten für den Anwender konzipiert sind und nicht für die Nutzung durch Programme. Mit XML ist es möglich den Inhalt von der Darstellung zu trennen und strukturiert zu sammeln. In Verbindung mit der zusätzlich vorhandenen Interaktivität stellt XML die Basis der aktuellen Generation der Electronic Commerce Anwendungen dar [MSX98].

3.1.2 XML for Java Parser

Der XML-Parser für Java von IBM (xml4j) ist vollständig in Java implementiert und steht für alle gängigen Plattformen zur Verfügung.⁷ Der Parser ermöglicht das Lesen, Erzeugen, Verändern und die Validierung von XML-Dokumenten. Dabei werden unter anderem die Standards SAX und DOM unterstützt.

⁶ Siehe auch Kapitel 3.1.3.

⁷ Der Parser ist unter <http://www.alphaworks.ibm.com> zu beziehen.

Die Integration des Parsers in die Applikation erfolgt durch das Erzeugen eines Parser-Objekts in Java und die anschließenden Aufrufe der API-Methoden, um das Dokument als Objekt im Speicher der Java Virtual Machine [NAZ99] zu erzeugen.

Durch die modulare Architektur des Parsers ist eine komponentenweise Nutzung ebenso möglich wie eine individuelle Anpassung der Funktionen [PFE99].

3.1.3 DOM

DOM ist die Bezeichnung für Document Object Model und stellt ein logisches Schnittstellen-Modell zu XML-Dokumenten dar. Mit Hilfe von DOM, das sowohl sprach- als auch plattformunabhängig ist, kann auf ein XML-Dokument lesend und schreibend zugegriffen werden.

Um die Unabhängigkeit von der Programmiersprache zu erreichen, stellt DOM selbst nur eine Sammlung von Schnittstellen dar, die in der jeweiligen Sprache implementiert werden müssen und dann als DOM API bezeichnet werden. Die größte Bedeutung hat dabei die DOM API für Java erlangt, da ein hoher Anteil der XML-basierten Anwendungen mit Java erstellt werden und die Entwicklung der XML-Parser für Java sehr weit fortgeschritten ist.

Die Schnittstelle DOM API erlaubt den hierarchischen Zugriff auf ein XML-Dokument in der Struktur eines Baumes. Der Parser wandelt die XML-Elemente automatisch in Knoten um und erzeugt ein Baumobjekt für das gesamte Dokument. Der Zugriff erfolgt über den Wurzelknoten des Baumes, der das gesamte Dokument repräsentiert und kann durch Traversieren der Knoten fortgesetzt werden. Die Stärke von DOM besteht darin, dass dieser Baum unabhängig vom verwendeten Parser immer dieselbe Struktur besitzt [NAZ99].

Durch die hierarchische Struktur von XML ist es immer möglich, die Umwandlung zu einem Baum vorzunehmen. Sinnvoll ist es aber nur in Fällen, in denen die zugrundeliegende Struktur des Objektes, das in XML-Form vorliegt, ebenfalls eine hierarchische Struktur aufweist.

Die DOM-Schnittstelle stellt zahlreiche Methoden und Eigenschaften zur Verfügung, die einen effektiven Zugriff auf das gesamte Dokument oder auch nur auf einzelne Knoten ermöglicht [W3C98b]. Dabei besitzt jeder Knoten einen zugewiesenen Typ, den er im Rahmen des XML-Dokuments darstellt. Folgende Typen werden unterschieden:

- DOCUMENT
- ELEMENT
- ATTRIBUTE
- PROCESSING_INSTRUCTION

3. Verwendete Technologien

- COMMENT
- TEXT
- CDATA_SECTION
- DOCUMENT_FRAGMENT
- ENTITY
- ENTITY_REFERENCE
- DOCUMENT_TYPE

Für die Suche nach bestimmten Knoten oder die systematische Traversierung stehen folgende Methoden zur Verfügung:

- GetDocument → Zugriff auf den Wurzelknoten
- GetChildNodes → Zugriff auf die nächste Ebene
- GetFirstChild → Zugriff auf den ersten Sohnknoten
- GetLastChild → Zugriff auf den letzten Sohnknoten
- GetParentNode → Zugriff auf den Vaterknoten
- GetPreviousSibling → Zugriff auf den vorderen Knoten derselben Ebene
- GetNextSibling → Zugriff auf den nächsten Knoten derselben Ebene
- GetAttributes → Zugriff auf die Attribute des aktuellen Knotens.

Darüberhinaus sind weitere Methoden vorhanden, die für die Bestimmung der Eigenschaften eines Knotens verwendet werden:

- GetNodeValue → Lesen des Wertes des Knotens
- GetNodeType → Lesen des Typs des Knotens
- GetTagName → Lesen des Namen des Knotens
- GetAttributes → Zugriff auf die Attribute des Knotens
- GetElementsByTagName → Zugriff auf Elementknoten, die einen zu übergebenden Namen besitzen

3.1.4 MS Developer Studio

Das Developer Studio von Microsoft beinhaltet unter anderem Entwicklungsumgebungen für die beiden objektorientierten Sprachen Java (Visual J++) und Visual Basic. Die Vorteile von

Visual Studio liegen hauptsächlich im Bereich der Erstellung von „graphical user interfaces“ (GUI) und im Datenaustausch auf Basis des Microsoft Universal Data Access (UDA).

Visual Basic gehört zu den Sprachen, mit denen sich einfach und schnell Windows Applikationen erstellen lassen. Eine große Anzahl von Entwicklungswerkzeugen und grafischen Komponenten erleichtern die Programmierung.

Bei Visual J++ liegen die Vorteile in der Kombination der grafischen Oberfläche und Werkzeugen mit der Plattformunabhängigkeit und Flexibilität der Sprache Java. Java zeichnet sich nach [LEV98] vor allem durch das Prinzip des Bytecode Interpreters und das zur Laufzeit erfolgende „Linken“ als Sprache für Internet Anwendungen aus.

3.2 Datenbankserver

Als Datenbank ist im Rahmen der Arbeit der Microsoft SQL-Server der Version 7.0 eingesetzt worden. Der OLAP-Server als multidimensionale Datenbank ist Bestandteil dieses Servers.

3.2.1 MS SQL-Server

Der Microsoft SQL-Server ist ein auf SQL basierendes relationales Datenbanksystem. Die Architektur ist auf den Einsatz als Client/Server-System ausgelegt. Hierbei werden neben weiteren die Datenzugriffskonzepte OLE DB und ADO von Microsoft unterstützt. Der Server ermöglicht den zeitgleichen Zugriff verschiedener Benutzer auf unterschiedliche „user databases“. Als „system databases“ werden bei der Installation die vier Datenbanken master, model, tempdb und msdb eingerichtet, die zur Verwaltung des Servers benötigt werden. Der Zugriff darauf sowie die gesamte Administration kann über die Microsoft Management Console erfolgen.

3.2.2 MS OLAP-Server

Der OLAP-Server als Komponente des Microsoft SQL-Server ist auch unter dem Entwicklungsnamen „Plato“ bekannt [BRO99]. Microsoft hat zahlreiche Assistenten und Hilfen für die Bedienung als Bestandteil des OLAP-Servers integriert und versucht, den stark diversifizierten Markt für OLAP mit eigenen Standards zu lenken.

Besonders erfolgreich ist diese Strategie beim offenen Standard OLE DB und der Erweiterung OLE DB for OLAP, die mittlerweile von fast allen Anbietern entsprechender Produkte als Schnittstelle zum Datenzugriff unterstützt werden. Als ein Bestandteil dieses Standards ist es MDX gelungen, sich als Abfragesprache durchzusetzen. Die Weiterentwicklungen ADO und

ADO MD versprechen dabei ähnlich erfolgreich zu werden, da hiermit eine Vereinfachung und eine Erweiterung der einsetzbaren Sprachen erfolgt ist.

Der OLAP-Server bietet eine einfache, aber komfortable Plattform zur Arbeit mit multidimensionalen Daten. Die umfangreiche Funktionalität enthält zahlreiche Optionen der Datenspeicherung (MOLAP, ROLAP, HOLAP, Speicherung in Partitionen), das Konzept des virtuellen Cube, sowie das Erzeugen berechneter Elemente („Calculated Member“).

Mit dem Konzept der Calculated Member existiert die Möglichkeit aus vorhandenen Werten, sowohl auf der Ebene der Measures als auch bei Dimensionen, neue Elemente automatisch zu berechnen. Beispielsweise kann das Measure „Durchschnitt“ aus den Werten „Summe“ und „Anzahl“ berechnet werden (MDX=„[MEASURES].[Summe] / [MEASURES].[Anzahl]“). Diese Möglichkeit ist nicht auf die Measures beschränkt, sondern kann für beliebige Dimensionen genutzt werden. Als Beispiel lassen sich bei einer geografischen Dimension die Elemente eines Levels, wie „Jamaika“, „Cuba“ und „Antigua“ zu einem Member „Karibik“ zusammenfassen. Für die Definition dieser Elemente stehen zahlreiche Funktionen zur Verfügung, die durch das Generieren von MDX-Ausdrücken genutzt werden können [BRO99].

Die folgende Abbildung ist an eine Darstellung aus [MSO] angelehnt und gibt einen Überblick über die Architektur des Systems:

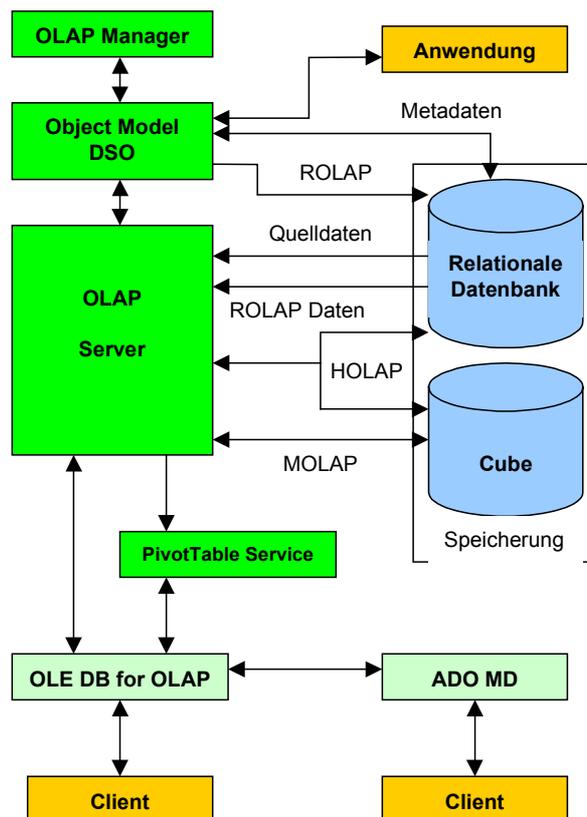


Abbildung 4: MS OLAP-Server

Während der Zugriff auf die Daten durch den Client auf Basis von OLE DB for OLAP oder ADO MD realisiert ist, gibt es für die Arbeit mit dem Server zwei Alternativen:

Microsoft Management Console (MMC)

Die Benutzerschnittstelle des OLAP-Managers bietet eine menugesteuerte grafische Oberfläche, die an das Layout des Windows-Explorers angelehnt ist. Mit Hilfe zahlreicher Editoren und Assistenten lassen sich sämtliche Metadaten dialoggesteuert bestimmen. Darüber hinaus lässt sich das Datenmodell grafisch darstellen und es wird eine kontextsensitive Hilfe angeboten.

Decision Support Objects (DSO)

Der OLAP Manager selbst wird durch DSO implementiert. Es ist möglich auch direkt auf dieses Objektmodell zuzugreifen. Mit Hilfe des Component Object Model lassen sich alle Funktionen des Servers durch das Importieren des DSO-Objektes als ActiveX-Objekt ausführen. Dies bedeutet, dass durch den Zugriff über die API auf das Objektmodell des Servers, Daten-

3. Verwendete Technologien

banken erzeugt, bearbeitet und gelöscht werden können. Durch die so definierten Metadaten kann der OLAP-Server auch über ein verteiltes System programmiert werden.

Zentrales Objekt dieser Bibliothek ist die Collection „MDStores“, die je nach Anordnung die Sammlung der Datenbanken, der Cubes, der Partitionen oder der Aggregationen enthält. Der Zugriff auf einzelne Elemente erfolgt mit Hilfe einer Indexangabe. Abbildung 5 [BSO99] gibt eine Übersicht über die Elemente, die jeweils unterhalb des MDStores-Objekts angeordnet sind. Die Programmierung erfolgt über Zuweisungen zu den Elementen oder das Ausführen von Methoden zum Hinzufügen oder Löschen von Objekten.

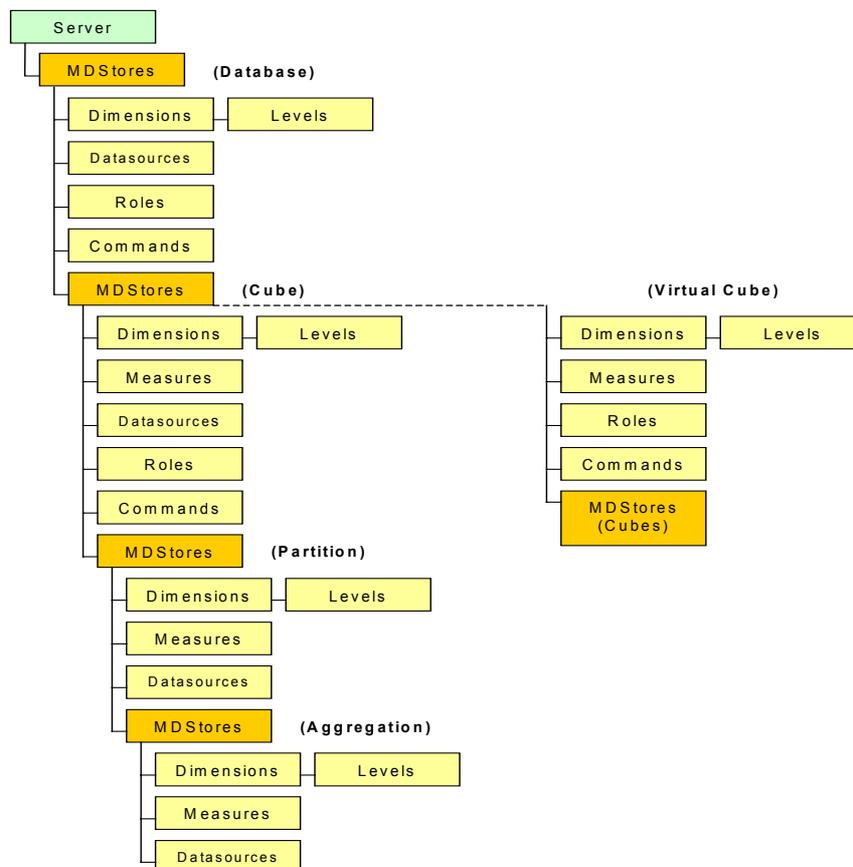


Abbildung 5: DSO-Objektmodell

Der Aufbau einer OLAP-Anwendung erfordert stets die Verbindung zu einer relationalen Datenquelle über OLE DB. Diese Verbindungsdaten werden als „DataSource“ definiert und stellen den Zugang zu den Metadaten dar, die in Form der fact-tables und Dimensionstabellen vorliegen müssen.

Bei der Definition der Metadaten ist zu beachten, dass der OLAP-Server keine Dimensionen unterstützt, die mehrere Hierarchien besitzen. Um semantisch eine solche Konstellation zu erreichen, muß der Umweg über die Bildung mehrerer Dimensionen gewählt werden [TSC99].

Neben den gewöhnlichen Dimensionen existiert ein spezieller Typ für Dimensionen, deren Werte sich auf Datum oder Zeit beziehen. Diese Zeitdimensionen können im OLAP-Server, sowohl über DSO als auch über MMC ohne die Erzeugung relationaler Tabellen definiert werden. Hierzu werden die Metadaten anstelle der Dimensionstabelle aus der fact-table bestimmt. Eine Spalte mit Datums- oder Zeitwerten wird in Verbindung mit vordefinierten Zeit-Levels zur Erzeugung der Members verwendet. Folgende Levels stehen zur Verfügung [TSC99]:

- Years
- HalfYears
- Quarters
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

Für die Measures stehen als Aggregationsfunktionen folgende Möglichkeiten zur Verfügung:

- Summation (default)
- Maximum
- Minimum
- Count

3.3 Schnittstellen

Dieses Kapitel gibt eine kurze Übersicht über die verwendeten Schnittstellen und Konzepte, die zur Verbindung der einzelnen Module benötigt wurden.

3.3.1 MS COM

Bei COM (Component Object Model) handelt es sich um ein weit verbreitetes Objektmodell zur Erstellung von Anwendungen mit Hilfe unabhängiger Software-Komponenten. COM fungiert dabei als Schnittstelle zwischen einer Server-Komponente und einer Client-Komponente und soll die Wiederverwendbarkeit und Kapselung von Software-Komponenten fördern [VER99a].

Diese Schnittstelle stellt einen Zeiger auf einen gemeinsamen Speicherblock zur Verfügung, der auf eine Tabelle von Funktions-Speicheradressen („virtual function table“) verweist. So kann die Client-Komponente auf die COM Systemmethoden, sowie auf die von der Server-Komponente implementierten Methoden zugreifen [GHL98].

Die COM API in Verbindung mit Java wird durch die Microsoft Virtual Machine realisiert. Hierdurch kann jedes COM-Objekt zur Laufzeit in Java integriert werden und jede Java Klasse als COM-Objekt erzeugt werden.⁸ Das Bereitstellen einer Java Klasse als COM Komponente erfolgt durch den sogenannten COM-Callable Wrapper (CCW), während der umgekehrte Weg durch den Java-Callable Wrapper (JCW) erfolgt. Hier wird eine DLL-Bibliothek (Dynamic Link Library) eines COM-Objektes als normale Java-Klasse behandelt [VER99b].

Die ActiveX Technologie von Microsoft, die für den Einsatz im World Wide Web konzipiert wurde, ist ebenfalls eine Anwendung des Component Object Model.

3.3.2 MS UDA

Die von Microsoft entwickelte Strategie UDA (Universal Data Access) stellt eine Plattform dar, beliebige Daten verschiedener Quellen zugänglich zu machen. Vor allem solche Anwendungen, die auf Basis von Internet oder Intranet auf Daten zugreifen, standen nach [BP98] bei der Entwicklung von UDA im Vordergrund.

Ziel ist ein direkter Zugang zu den Daten ohne jegliche Transformation oder Replikation, um den Eindruck einer Universellen Datenbank zu erhalten.

Die folgenden vier Abschnitte beschreiben die im Rahmen des Projektes relevanten Modelle die mit UDA als Schnittstelle zur Verfügung stehen.

OLE DB

OLE DB ist die auf der Basis von COM definierte Kernkomponente von UDA. Mit dem offenen Standard OLE DB ist es möglich, die Datenzugriffsobjekte und -methoden zu nutzen oder Schnittstellen für Anwendungen zu entwerfen.

OLE DB stellt somit die Schnittstelle zwischen der Datenquelle als Provider und der Anwendung als Consumer dar. Für den Bereich der relationalen Datenbanken übernimmt OLE DB die Aufgaben des ODBC Standards. Der Vorteil von OLE DB besteht in der verbesserten Zugriffsmöglichkeit und in der Ausweitung auf Datenquellen, die nicht relational sind.

⁸ Tatsächlich war es jedoch trotz COM-API von Java aus nicht möglich gewesen, auf das DSO-Objekt zuzugreifen.

OLE DB for OLAP

OLE DB for OLAP stellt eine Erweiterung des OLE DB Konzeptes für multidimensionale Datenquellen dar. Die OLE DB Spezifikation wurde um Objekte und Methoden ergänzt, die benötigt werden, um einen MDP (Multidimensional Data Provider) zu entwickeln oder zugänglich zu machen.

Das Schnittstellen-Standard unterstützt sämtliche in Kapitel 2.2.2 beschriebenen Komponenten eines OLAP-Systems. Als Bestandteil von OLE DB for OLAP ist die Abfragesprache MDX (Multidimensional Expression) eingeführt worden. MDX übernimmt die Rolle von SQL für den multidimensionalen Bereich. Über diese Schnittstelle ist es möglich, sowohl lesend als auch schreibend auf die Datenquelle zuzugreifen.

ADO

ActiveX Data Objects (ADO) sind eine Weiterentwicklung von OLE DB, die den Datenzugriff weiter vereinfachen sollen. Unter Nutzung des OLE DB Konzeptes wurde ein Modell entwickelt, das eine Ebene höher, auf Anwendungsebene, anzusiedeln ist. ADO ist demnach eine API-Schnittstelle, die im Vergleich zu OLE DB einen beschränkten Funktionsumfang besitzt, aber für den einfachen Einsatz im Rahmen von Entwicklungen in C++, Java, Visual Basic oder auch Skriptsprachen geeignet ist. Beim lesenden Zugriff auf Daten steht der selbe Funktionsumfang wie bei OLE DB zur Verfügung.

ADO MD

Äquivalent zu OLE DB for OLAP ist auch das ADO Konzept um eine multidimensionale Ebene erweitert worden. Eine wichtige Ergänzung stellt das Objekt „cellset“ an Stelle des Objekts „rowset“ für den relationalen Fall dar. Ein cellset repräsentiert eine View auf einen Datacube und kann das Ergebnis einer Abfrage auf einem multidimensionalen Datenraum aufnehmen. Die Dimensionen werden hierbei als Achsen bezeichnet und die Sammlung der Datacube-Definitionen ist über das Objekt „Catalog“ zugänglich.

Die Verbindung zu einem ADO MD Provider wird über das „Connection“ Objekt hergestellt, das selbst jedoch nicht zu ADO MD, sondern zur ADO Bibliothek gehört. Zu beachten ist, dass mit dieser Schnittstelle nur lesend auf die Datenquelle zugegriffen werden kann, und somit auch keine Veränderung der Datenbankstruktur unterstützt wird.

4 Entwurf

Dieses Kapitel beschreibt den Entwurf des multidimensionalen Katalogsystems, ohne auf Einzelheiten der Implementierung einzugehen.

4.1 Anforderungen

Das Ziel der Implementierung liegt in der Integration der XML-Technologie unter Verwendung des Java XML-Pasers und des DSO-Modells zur Generierung der OLAP-Datenbank. Hieraus lassen sich die folgenden Aufgaben ableiten:

- Definition der Metadaten und Anwendungsdaten eines Katalogsystems in XML mittels DTD
- Validieren und Auslesen dieser Daten mit Hilfe des Java XML-Parsers
- Automatische Generierung einer durch die Katalog-Metadaten definierten multidimensionalen Datenbank über DSO
- Einfügen der Anwendungsdaten des Last Minute Beispiels

4.2 Datenmodell

Im Mittelpunkt des Systems steht der Aufbau der OLAP-Datenbank auf Basis der in XML definierten Metadaten. Die zur Definition eines Katalogs benötigten Komponenten ergeben sich aus der Struktur einer OLAP-Datenbank allgemein und der Möglichkeiten des Microsoft-Servers im Speziellen. Die folgende Abbildung stellt die verwendeten Metadaten des Katalogs als Objekte dar.

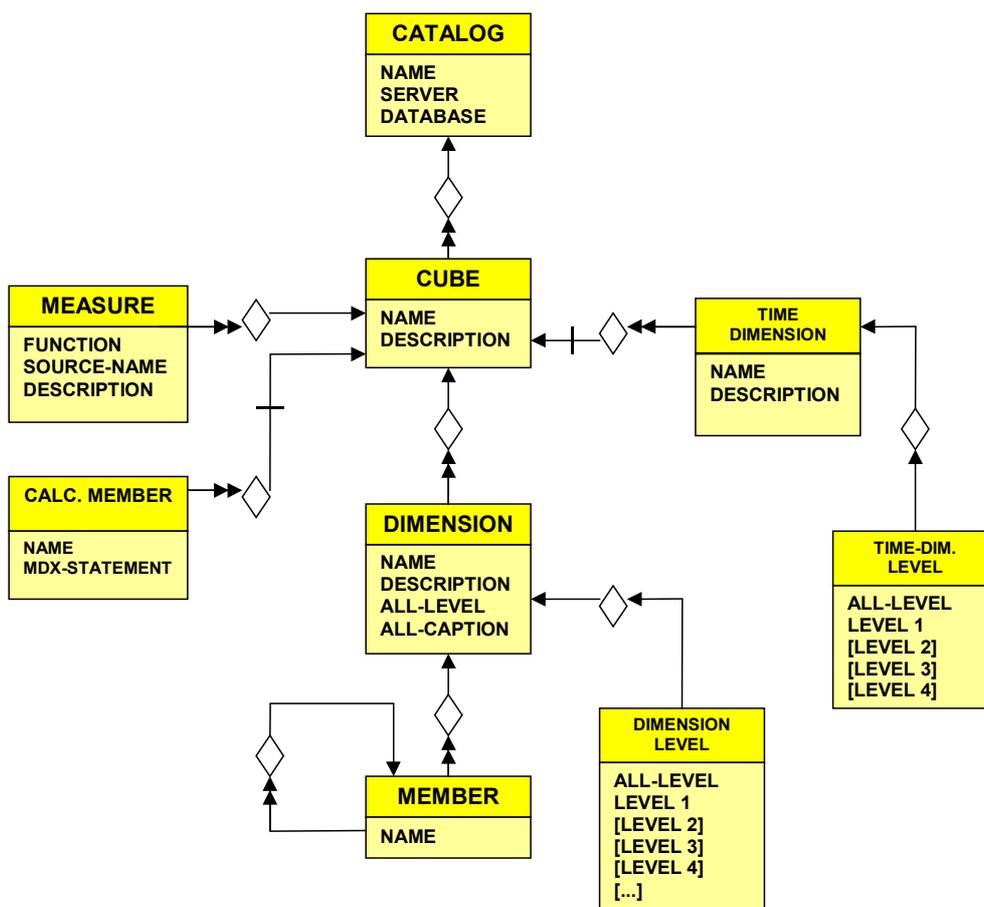


Abbildung 6: Objektmodell der Metadaten

Ein Katalog kann demnach aus einem oder mehreren Datacubes bestehen, die ihrerseits beliebig viele Measures, berechnete Elemente, Dimensionen und Zeitdimensionen enthalten können. Die Dimensionen werden über die Angabe ihrer Members definiert, die rekursiv als Baumstruktur modelliert sind. Diese hierarchische Struktur ist durch die Umsetzung als DTD eine Vorgabe für alle in XML zu erstellenden Ausprägungen. Die Gültigkeit des jeweiligen XML-Dokuments mit dieser DTD wird durch den Parser validiert.

4.3 Ablaufbeschreibung

Der Aufbau der OLAP Datenbank erfordert zunächst die Generierung einer relationalen Datenbank als Informationsquelle.

Die Transformation der relationalen Daten in das multidimensionale Modell wird durch das OLAP-Datenbanksystem realisiert. Der Aufbau des Datacube erfolgt durch die Erstellung der Dimensionen nach Vorgabe der Dimensionstabellen. Diese Tabellen enthalten, wie in

Abbildung 7 zeigt, eine Spalte für jeden zu definierenden Level. Die Zeilen der Dimensionstabellen werden mit der Struktur der Members gefüllt. Jedes Blatt im Baum der Members wird als eine Zeile mit dem gesamten Pfad zur Wurzel in die Tabelle eingetragen.

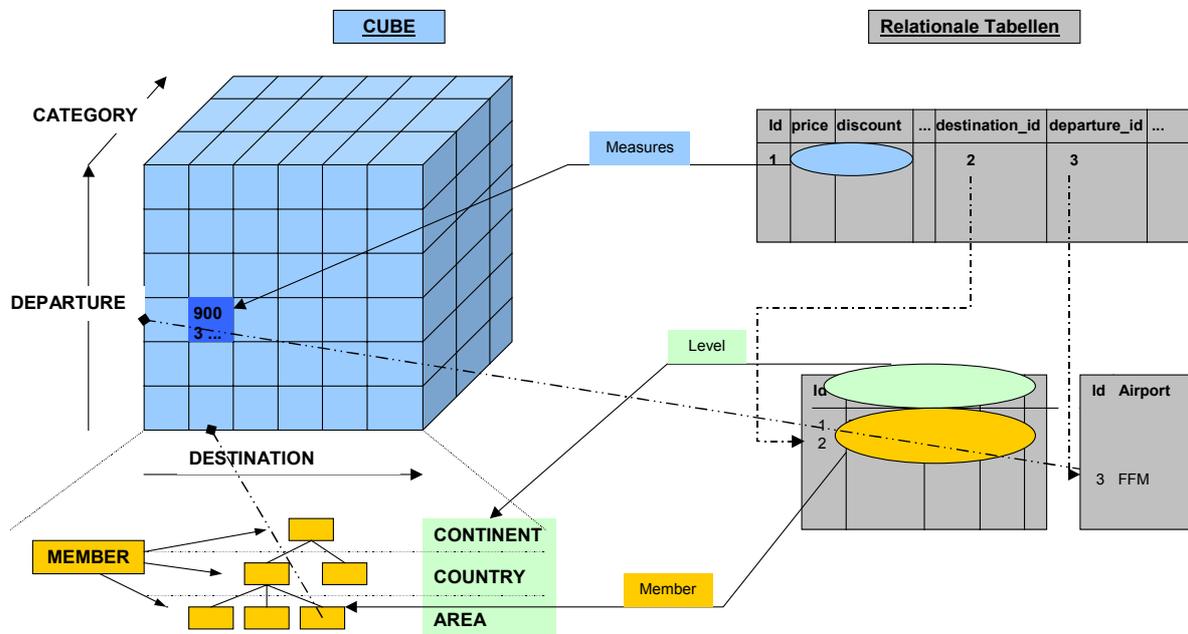


Abbildung 7: Transformation vom relationalen in das multidimensionale Modell

Des Weiteren werden die Zeilen der fact-table so in den Cube übertragen, dass durch Auswertung der Dimensionsangaben die definierte Zelle mit den jeweiligen Measures gefüllt werden. Durch den Verweis auf ein Member für jede Dimension wird die Zelle des zugrundeliegenden Eintrags der fact-table eindeutig bestimmt.

Aus diesen Zusammenhängen ergeben sich folgende Schritte zum Aufbau des OLAP Systems:

1. Aufbau einer relationalen Datenbank im SQL Server

- Verbindungsaufbau über ADO
- Anlegen einer relationalen Datenbank
- Erzeugen einer fact-table für jeden Cube
 - Hinzufügen einer Spalte für jedes Measure
 - Hinzufügen einer Spalte als Fremdschlüssel für jede Dimension

- Erzeugen der Dimensionstabellen
 - Hinzufügen einer Spalte für jeden Level
 - Füllen der Tabellen mit den Member Definitionen
- Füllen der fact-table mit den Werten

2. Aufbau der OLAP Datenbank

- Verbindungsaufbau über DSO
- Anlegen einer OLAP Datenbank
- Erzeugen der Cubes
 - Angabe der DataSource als Verbindung zur fact-table
 - Erzeugen der Dimensionen
(Hinzufügen der Levels und Angabe der DataSource als Verbindung zur Dimensionstabelle)
 - Erzeugen der Measures
(Angabe der DataSource als Spalte der fact-table und Festlegung der Aggregationsfunktion)
 - Erzeugen der berechneten Elemente
(Definition mit Hilfe des Command Objekts von DSO, das jeweils beim Aufbereiten eines Cube ausgeführt wird und über Angabe eines MDX Statements ein Calculated Member erzeugt)
 - Erzeugen der Zeitdimensionen
(Hinzufügen der vordefinierten Zeit-Levels und Angabe der DataSource als Verbindung zur entsprechenden Spalte der fact-table)
 - Schema definieren
(Bilden der Joins zwischen fact-table und Dimensionstabellen)

3. Aufbereiten der Cubes

- Über die als DataSource definierte Verbindung wird anhand der relationalen Tabellen die Transformation zum OLAP-Cube vorgenommen. Hierbei werden die Dimensionen aufgebaut und alle möglichen Aggregationen berechnet.

4.4 System-Architektur

Das System enthält aufgrund eines Problems bei der Einbindung der DSO-Bibliothek in die MS Visual J++ Umgebung als ActiveX Komponente eine in Java implementierte Komponente und eine Visual Basic Komponente. Die gesamte Architektur und das Zusammenspiel der Komponenten wird durch Abbildung 8 dargestellt.

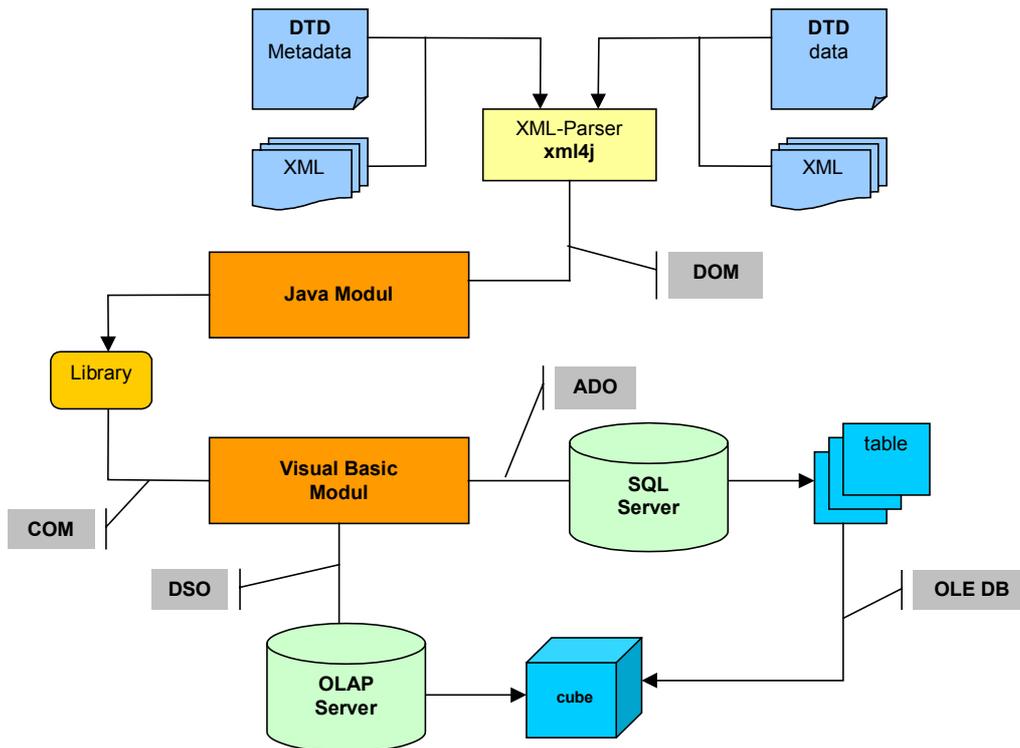


Abbildung 8: Architektur des Systems

Der Input des Systems sind die Metadaten und Anwendungsdaten in XML-Form. Die Struktur dieser XML-Dokumente wird durch die entwickelte Document Type Definition für Metadaten (metadata.dtd) und für Daten (data.dtd) festgelegt. Die Strukturierung der DTD lehnt sich an die Architektur des OLAP Servers an. Die entsprechenden Ausprägungen als XML-Dateien enthalten die anwendungsspezifischen Informationen. Das XML-Dokument, das die Metadaten enthält, beinhaltet die Struktur der zu erzeugenden anwendungsspezifischen multi-dimensionalen Datenbank.

Die Validierung in Verbindung mit dem Parsen der XML-Dokumente erfolgt mit Hilfe des XML-Parsers von IBM (xml4j). Dieser in Java implementierte Parser wird als Klasse in das entwickelte Java-Modul integriert. Der Zugriff auf das XML-Dokument über den Parser er-

folgt durch das Document Object Model (DOM). Die Validierung prüft, ob die Katalog-Metadaten in einer zulässigen Form bereitgestellt werden. Das XML-Dokument stellt somit eine alternative Administrator-Schnittstelle dar, die als Basis für die Dateneingabe durch den Broker dient.

Im Rahmen der implementierten Java Methoden wird das Dokument-Objekt in Baum-Form über die Verwendung der DOM-Funktionalität traversiert und ausgelesen. Die Java-Methoden werden auf der Grundlage des Component Object Model als Type Library exportiert.

Die Visual Basic-Komponente importiert das Java-Modul durch Einbindung der Library und Erzeugen eines COM-Objekts. Über den Aufruf der Methoden zum Traversieren des Dokuments werden die Informationen über den Aufbau des Data Warehouse, sowie die Anwendungsdaten in der Visual Basic-Komponente verfügbar. Diese Komponente stellt eine Verbindung zum SQL-Server her, durch den eine relationale Datenbank aufgebaut und gefüllt wird, die zur Generierung der OLAP-Datenbank benötigt wird. Als Schnittstelle wird hierbei ADO verwendet. Die Übertragung der Anwendungsdaten erfolgt ebenfalls über ADO und füllt die fact-table mit den vorhandenen Werten.

Parallel hierzu wird während des Auslesens der XML-Daten eine Verbindung zum OLAP-Server aufgebaut. Durch Integration der DSO-Bibliothek als ActiveX Komponente steht das Objektmodell der OLAP-Datenbank im Visual Basic-Modul zur Erzeugung der Struktur zur Verfügung.

Das abschließende Aufbereiten und Erzeugen der Cubes erfolgt durch die ebenfalls spezifizierte OLE DB-Verbindung der OLAP-Datenbank zur relationalen Datenquelle des SQL-Servers.

5 Beispiel

Die Entwicklung des multidimensionalen Katalogsystems wird durch die Realisierung eines spezialisierten Katalogsystems für Last Minute-Reisen begleitet. Der nächste Abschnitt beschreibt das Anwendungsszenario und erklärt warum sich Online-Reiseangebote besonders gut durch multidimensionale Kataloge umsetzen lassen.

5.1 Motivation

Die Motivation für einen multidimensionalen Katalog für Last Minute-Reisen stützt sich auf folgende Punkte:

- Das Produkt Last Minute-Reisen ist für die Präsentation und den Verkauf über das Internet sehr gut geeignet und entsprechend ist dort bereits ein umfangreiches Angebot vorhanden. Zahlreiche große Reiseveranstalter⁹ bieten dem Nutzer große Datenbestände zur Recherche. Die Eignung des Produktes für die Angebotspräsentation im Internet hat folgende Gründe:
 1. Das Angebot von Last Minute Reisen ist auf eine sehr große Aktualität angewiesen. Der Datenbestand unterliegt permanent aktuellen Änderungen.
 2. Das Produkt Last Minute-Reise ist kaum erklärungsbedürftig. Es ist nicht sehr komplex und kann anhand weniger Fakten beschrieben werden.
 3. Das Angebot ist sehr umfangreich und kann von Interessenten nur durch entsprechende Selektionsmöglichkeiten sinnvoll genutzt werden.
- Die Struktur der Angebote für Last Minute-Reisen eignet sich sehr gut für die Übertragung in ein OLAP-System. Die Zusammensetzung eines Angebots aus einer bestimmten Anzahl an Merkmalen ermöglicht eine einfache Kategorisierung und führt zu einer Datenstruktur mit multidimensionalem Charakter.
- Das vielfältige Angebot im Internet ist nicht ohne weiteres überschaubar. Eine integrierte OLAP-Datenbank ermöglicht eine Suche im gesamten Datenbestand verschiedener Anbieter.

⁹ Beispiele für Anbieter sind: `<http://www.ltur.de>`, `<http://www.lastminute.de>`, `<http://www.tui.de>`, `<http://www.last-minute-travel.de>`, `<http://www.clever-lastminute.com>`

- Die zahlreichen Anbieter von Last Minute-Reisen stellen dem Nutzer unterschiedliche Suchsysteme zur Verfügung. Eine Integration der verschiedenen Angebote in eine gemeinsame Datenbank bietet einen einheitlichen Zugriff.
- Die existierenden Funktionen und Zugriffsmöglichkeiten für die Suche und Navigation im Datenbestand sind sehr beschränkt. Häufig besteht lediglich die Möglichkeit über eine statisch festgelegte Reihenfolge von Navigationsschritten den Ergebnisraum einzuschränken. Eine angebotene Alternative ist das Ausfüllen eines Suchformulars. Hierbei wird jedoch kein Überblick über die Ergebnismenge hinaus erzielt, so dass die Suche zwangsläufig nach dem Prinzip „trial and error“ verläuft.

5.2 Katalog-Struktur

Für das Anwendungsszenario Last Minute ist ein Datacube mit insgesamt zehn Dimensionen und acht Measures erstellt worden:

Dimensionen

1. Destination	Hierarchische Gliederung des Reiseziels
2. Departure Airport	Abflughafen
3. Category	Klassifizierung der Unterkunft
4. Boarding	Art der Verpflegung
5. Company	Reiseveranstalter
6. Offer Date	Datum der Angebotserstellung
7. Offer Deleted Date	Ende der Angebotsgültigkeit
8. Actuality	letzte Aktualisierung des Angebots
9. Departure Date	Abflugdatum
10. Arrival Date	Ankunftsdatum

Measure

1. Min Price	kostengünstigstes Angebot
2. Max Price	teuerstes Angebot
3. Sum Price	Summe der Angebotspreise
4. Visitors	Anzahl der Besucher dieser Angebotskategorie
5. Discount	Rabatt
6. Additional Charge	zusätzliche Gebühren
7. Number	Anzahl der Angebote der aktuellen Kategorie
8. Average	Durchschnittspreis der Angebote

Bei der Bildung des Datenmodells sollten lediglich Informationen berücksichtigt werden, die für alle angebotenen Produkte verfügbar sind. Die Unterteilung der einzelnen Dimensionen in

5. Beispiel

ihre Member-Hierarchien kann anhand des in Anhang B dargestellten XML-Dokuments nachvollzogen werden.

Damit sieht der Cube in der Darstellung des Editors des OLAP-Servers folgendermaßen aus:

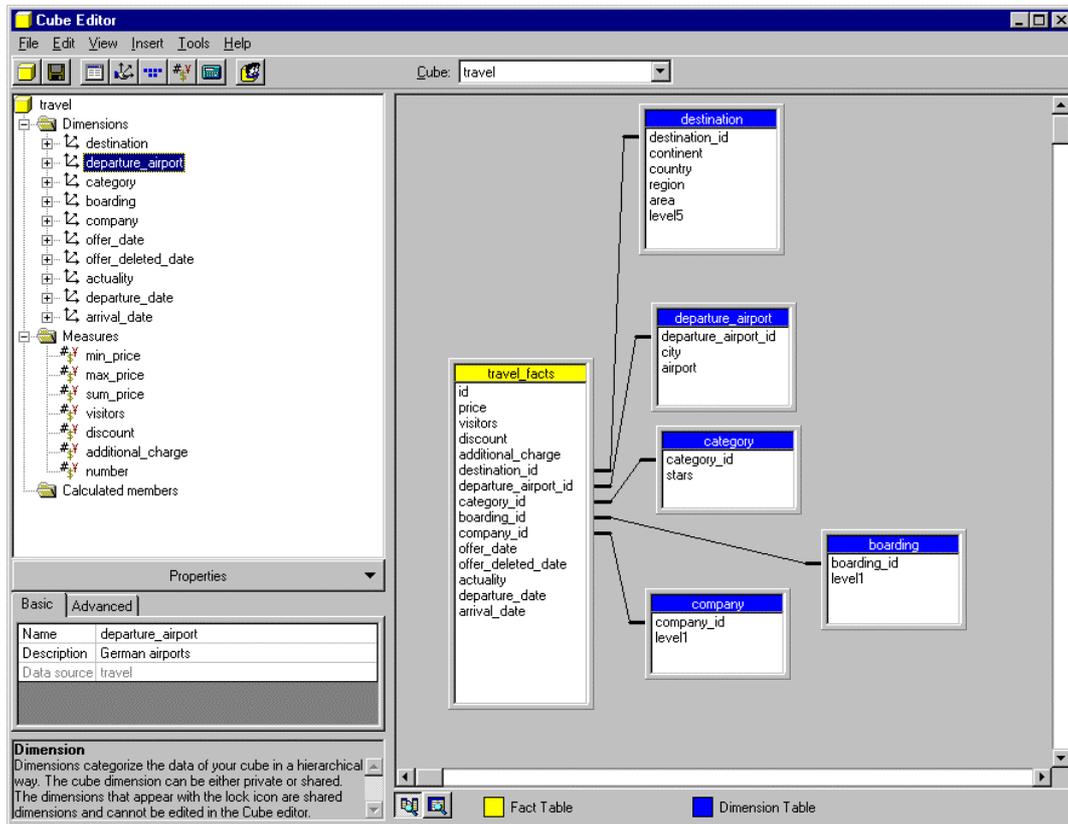


Abbildung 9: Struktur des Last Minute-Cube

Das Datenmodell der Last Minute-Anwendung wurde nach dem Prinzip des star schema entworfen. Die fact-table des Cube enthält die Measure-Werte, die Angaben für die fünf Zeitdimensionen, sowie die Fremdschlüssel zu den übrigen fünf Dimensionen.

Weitere Informationen zu den einzelnen Angeboten, wie z.B. die Angabe des Hotels, sollte im fertiggestellten Katalog in eine zusätzliche Tabelle zusammengefasst werden, die in das System integriert wird.¹⁰

¹⁰ Das Beispiel beschreibt lediglich das Datenmodell und läßt offen wie die Benutzerschnittstelle implementiert wird. Im OLAP-Bereich existieren allerdings bereits verschiedene Realisierungen für GUI's.

6 Ausblick

Die Umsetzung der Aufgabenstellung hat gezeigt, dass es trotz verschiedener Probleme möglich war, anhand der Katalog-Metadaten in Form eines XML-Dokuments automatisch ein Data Warehouse zu generieren.

Die Verwendung der XML-Technologie hat den Vorteil, dass durch den Parser eine Validierung der Daten erfolgt und damit Angaben zurückgewiesen werden können, die nicht mit der Katalog-Struktur konform sind. Bei einer späteren Entwicklung eines Standards für multidimensionale Kataloge könnte unter Beibehaltung der XML-Schnittstelle ein solcher Standard berücksichtigt werden. Mit der XML Technologie wird es auch möglich sein, die benötigten Daten aus den Web-Seiten zu gewinnen und sie als Quelldaten in das Katalogsystem zu integrieren. Vergleichbare Broker-Anwendungen unter Nutzung des XML-Formats existieren bereits.¹¹ Solche Anwendungen sind auch in der Lage die Struktur der Informationen auszulesen [LHB99], die anschließend als Metadaten für das Katalogsystem dienen.

Als weitere Entwicklung des Gesamtprojekts wird eine Explorer-Komponente erstellt werden, die es einem Client ermöglicht, auf das Data Warehouse zuzugreifen. Hierfür bietet sich der Einsatz der MDX-Sprache zum Auslesen der Daten und XSL (eXtensible Stylesheet Language) als Formatvorlage zur Visualisierung der Informationen an.

Für den Bereich dieser Diplomarbeit ist an zukünftigen Aufgaben ein Komprimierungsverfahren zu nennen, das erforderlich wird, wenn große Datenbestände als XML-Dokumente verwaltet werden sollen. Des weiteren kann das Modell der Katalog-Metadaten um eine Komponente Virtual Cube erweitert werden. Hierdurch wäre es möglich einen Katalog zu definieren, der Produktinformationen aus mehreren Datacubes integriert. Bei der Abbildung dieses Objekts in XML, muß jedoch beachtet werden, dass es keine direkte Möglichkeit gibt, einen Verweis auf andere Elemente des Dokuments zu setzen.

Das Anwendungsszenario Last Minute stellt lediglich eine von vielen denkbaren Möglichkeiten eines solchen Systems dar. Neben der Verwendung für beliebige andere Produkte ist auch eine Anwendung im Bereich der Finanzanalysen denkbar, die im Internet momentan einen stark wachsenden Markt darstellen. Die wachsende Bedeutung des Aktien- und Kapitalmarktes in Deutschland hat zu einem Boom bei der Emission von Wertpapieren geführt. Zu jedem dieser börsennotierten Wertpapiere werden in unregelmäßigen Abständen zahlreiche Analysen, Unternehmensbewertungen, Ratings oder Kommentare veröffentlicht [BGL98].

¹¹ Vgl. z.B. das Projekt XML Broker der GMD IPSI unter <http://www.darmstadt.gmd.de/oasys/projects/xmlbroker/xmlbrokerd.htm>

Erstellt werden diese Einschätzungen von Banken, Investmenthäusern oder Börsenbriefen, wobei im Moment keine vollständige Sammlung dieser Meldungen verfügbar ist. Ein multidimensionales Katalogsystem auf Basis dieser Informationen könnte in Verbindung mit aktuellen Kursdaten und Kennzahlen eine interessante Anwendung darstellen.

Bei der Weiterentwicklung des Systems durch Integration zusätzlicher Komponenten dürfte sich die OLAP-Struktur der Datenbank als Vorteil erweisen. Durch die fortschreitende Standardisierung der OLAP-Technologie sind zahlreiche Werkzeuge, wie z.B. Data Mining in Verbindung mit OLAP verfügbar, die auf das System angewendet werden können.

Abschließend läßt sich prognostizieren, dass die Entwicklung auf dem Gebiet der elektronischen Katalogsysteme in Folge der höheren Anforderungen an die Analysemöglichkeiten immer häufiger die Data Warehouse-Technologie nutzen wird. Parallel zu dieser Entwicklung wird der Stellenwert der XML-Technologie vor allem im Bereich des Datenaustauschs weiter wachsen, weshalb mit dem hier vorgestellten Konzept des multidimensionalen Katalogsystems der Versuch unternommen wurde, die verschiedenen Technologien zusammen zu führen, um den zukünftigen Bedürfnissen der Internetanwendungen Rechnung zu tragen.

Literaturverzeichnis

- [BGL98] Betsch, Oskar; Groh, Alexander; Lohmann, Lutz: Corporate Finance. Vahlen Verlag München, 1998.
- [BP98] Blakeley, Jose A.; Pizzo, Michael J.: Microsoft Universal Data Access Platform. In Proceedings of ACM SIGMOD, International conference on management of data, Seattle, WA, S. 502-503, 1998.
- [BRO99] Brosius, Gerhard: Microsoft OLAP Services. Addison-Wesley, Bonn (u.a.), 1999.
- [BSH99] Blaschka, Markus; Sapia, Carsten; Höfling, Gabriele: On Schema Evolution in Multidimensional Databases, 1999. <http://www.forwiss.tu-muenchen.de/~system42/publications>.
- [BS97] Berson, Alex; Smith, Stephen J.: Data Warehousing, Data Mining & OLAP. McGraw-Hill, New York, N.Y.(u.a.), 1997.
- [CDM97] Catalogs for the Digital Marketplace, 1997. http://www.commerce.net/research/free-report/97_03_r.htm.
- [DSH98] Dinter, Barbara; Sapia, Carsten; Höfling, Gabriele; Blaschka, Markus: The OLAP Market: State of the Art and Research Issues, 1998. <http://www.forwiss.tu-muenchen.de/~system42/publications>.
- [GHL98] Gray, David N.; Hotchkiss, John; LaForge, Seth; Shalit, Andrew; Weinberg, Toby: Modern Languages and Microsoft's Component Object Model. In Communications of the ACM, Vol. 41, No. 5, S. 55-65, 1998.
- [GOL98] Goldfarb, Charles F.: The XML handbook. Prentice Hall, Upper Saddle River, N.J., 1998.
- [KE97] Kemper, Alfons; Eickler, Andre: Datenbanksysteme. R. Oldenbourg, München, Wien, 1997.
- [LEV98] Leventhal, M.: Designing XML Internet applications. Prentice Hall, Upper Saddle River, N.J., 1998. (The Charles F. Goldfarb series on open information management).

- [LHB99] Liu, Ling; Han, Wei; Buttler, David; Pu, Calton; Tang Wei: An XML-based Wrapper Generator for Web Information Extraction. In Sigmod Record, Philadelphia, PA., Vol. 28, No. 2, S. 540-543, 1999.
- [MDC] MDIS, Meta Data Interchange Specification, 1999.
<http://www.mdcinfo.com/MDIS/MDIS11.html>.
- [MEG98] Megginson, David: Structuring XML documents. Prentice Hall, Upper Saddle River, N.J., 1998. (The Charles F. Goldfarb series on open information management).
- [MER99] Merz, Michael: Electronic Commerce. d.punkt Verlag, 1999.
- [MSO] Introducing OLAP Services, Microsoft Corporation, 1999.
http://msdn.microsoft.com/library/psdk/olap/adintroducing_1.htm.
- [MSX98] XML: Enabling Next-Generation Web Applications, Microsoft Corporation, 1998. <http://msdn.microsoft.com/xml/articles/xmlwp2.asp>.
- [MVM99] Technology Overview: The Microsoft Virtual Machine for Java, Microsoft Corporation, 1999. <http://www.microsoft.com/java/resource/vm.htm>.
- [NAZ99] Nazmul, Idris: Introduction to DOM, 1999.
<http://developerlife.com/domintro/default.htm>.
- [OLE] OLE DB for OLAP Spezifikation, Microsoft Corporation, 1999.
<http://www.microsoft.com/data/oledb/olap/default.htm>.
- [PFE99] Pfeiffer, Ralf I.: XML tutorials for programmers. IBM XML Technology Group, 1999. <http://www.software.ibm.com/developer/education/tutorial-prog/parsing.htm>.
- [THO97] Thomsen, Eric: OLAP Solutions, Building Multidimensional Information Systems, Wiley Computer Publishing, New York, N.Y. (u.a.), 1997.
- [TSC99] Thomsen, Eric; Spofford, George; Chase, Dirk: Microsoft OLAP solutions. Wiley, New York, N.Y.(u.a.), 1999.
- [VER99a] Verbowski, Chad: Using COM Objects from Java, Microsoft Corporation, 1999. <http://www.microsoft.com/java/resource/java-com.htm>.

- [VER99b] Verbowski, Chad: Integrating Java and COM, Microsoft Corporation, 1999. <http://www.microsoft.com/java/resource/java-com2.htm>.
- [W3C98a] XML-Spezifikation, 1998. <http://www.w3.org/TR/1998/REC-xml-19980210>.
- [W3C98b] DOM-Level 1-Spezifikation, 1998. <http://www.w3.org/TR/REC-DOM-Level1/cover.html>.

Anhang A: XML DTD

Um die Beispieldaten in Form von XML-Dokumenten zu definieren, sind zunächst DTD Vorgaben für die Metadaten und die Anwendungsdaten erstellt worden. Die DTD zur Definition des Katalogschemas hat folgendes Aussehen:

```
<!-- Version 1.0 catalog.dtd -->

<!ENTITY % time.par
    '(Years|Quarters|Month|Weeks|Days)' >

<!ELEMENT catalog (cube+)>
<!ATTLIST catalog
    name          CDATA          #REQUIRED
    server        CDATA          #IMPLIED
    db            CDATA          #REQUIRED
>
<!ELEMENT cube (measures+,calculated_m*,dimension+,time_dimension*)>
<!ATTLIST cube
    name          CDATA          #REQUIRED
    description   CDATA          #IMPLIED
>
<!ELEMENT dimension (level,member*)>
<!ATTLIST dimension
    name          CDATA          #REQUIRED
    description   CDATA          #IMPLIED
    all_level     (yes|no)       #IMPLIED
    all_caption   CDATA          #IMPLIED
>
<!ELEMENT time_dimension (td_level)>
<!ATTLIST time_dimension
    name          CDATA          #REQUIRED
    description   CDATA          #IMPLIED
>
<!ELEMENT level EMPTY>
<!ATTLIST level
    level1       CDATA          "level1"
    level2       CDATA          "level2"
    level3       CDATA          "level3"
    level4       CDATA          "level4"
    level5       CDATA          "level5"
    level6       CDATA          "level6"
    level7       CDATA          "level7"
    level8       CDATA          "level8"
    level9       CDATA          "level9"
>
<!ELEMENT td_level EMPTY>
<!ATTLIST td_level
    all_level     (yes|no)       "yes"
    level1       %time.par;     "Years"
    level2       %time.par;     "Month"
    level3       %time.par;     "Days"
    level4       %time.par;     #IMPLIED
>
<!ELEMENT measures EMPTY>
```

```

<!ATTLIST measures
    function      CDATA      #REQUIRED
    source_name   CDATA      #REQUIRED
    description   CDATA      #IMPLIED
>
<!ELEMENT calculated_m EMPTY>
<!ATTLIST calculated_m
    name          CDATA      #REQUIRED
    mdx           CDATA      #REQUIRED
>
<!ELEMENT member (member*)>
<!ATTLIST member
    name          CDATA      #REQUIRED
>

```

Die DTD ist so modelliert, dass die eigentlichen Informationen der jeweiligen Objekte als Attribute deklariert werden. Diese Vorgehensweise entspricht der Darstellung der Metadaten als Objekte und zugehöriger Eigenschaften. Zu beachten ist das Attribut „server“ des „Catalog“-Objektes, das für den Verbindungsaufbau zum OLAP-Server benötigt wird. Durch Angabe eines gültigen Netzwerknamens kann damit ein entfernter Server angesprochen werden. Wird kein Attribut gesetzt, verwendet das Programm den Standardausdruck „localhost“.

Das Element Dimension wird durch Angabe der Bezeichnungen für die Levels und einer Menge von Members definiert. Die Members werden hierarchisch als Baum spezifiziert, wodurch die Tiefe des Baumes für jeden Knoten seine Zugehörigkeit zum entsprechenden Level widerspiegelt. Die Attribute „all_level“ und „all_caption“ der Dimensionen stehen im Zusammenhang mit der Möglichkeit, eine übergeordnete Ebene zur Darstellung der gesamten Dimension einzurichten.

Die Zeitdimensionen stellen eine Sonderform dar, die statt über Members lediglich über Time Levels spezifiziert wird. Hierbei besteht die Möglichkeit verschiedene Ebenen als Jahre, Quartale, Monate, Wochen oder Tage zu bestimmen.

Das Element Measure benötigt als Attribute einen Verweis auf die Datenquelle in Form der entsprechenden Spalte der fact-table, sowie die Angabe einer Aggregationsfunktion.

Ein „Calculated Member“ kann durch die Angabe eines Ausdrucks in Form von MDX erfolgen, der direkt in die OLAP-Spezifikation übernommen wird.

Die zweite DTD beschreibt das Format der Anwendungsdaten:

```

<!-- Version 1.0 data.dtd -->
<!ELEMENT data (table,recordset+)>
<!ATTLIST data
    server        CDATA      #IMPLIED
    db            CDATA      #REQUIRED
>

```

```
<!ELEMENT table (column*)>
<!ATTLIST table
    name          CDATA          #REQUIRED
>

<!ELEMENT column (#PCDATA)>

<!ELEMENT recordset (value+)>

<!ELEMENT value (#PCDATA)>
```

Äquivalent zum obigen Fall, kann auch hier über das Attribut „server“ ein entfernter Datenbank-Server angesteuert werden. Das Element table enthält eine Liste der Spaltennamen, und die Werte sind in Form von Datensätzen modelliert.

Die in dieser Form gespeicherten Daten werden als Anwendungsdaten in die fact-table übertragen. Als Voraussetzung wird an dieser Stelle von der Übereinstimmung des Datenformates mit der im Rahmen der Metadaten spezifizierten fact-table ausgegangen.

Anhang B: XML-Dokument

Als Beispiel für ein XML-Dokument der Metadaten ist nachfolgend die zu Testzwecken definierte XML-Datei für das Last Minute-Szenario aufgeführt:

```
<?xml version="1.0"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">

<!-- Version: 1.0 catalog.xml -->

<catalog name="travel" server="LOCALHOST" db="travel">
    <cube name="travel" description="version 1.0">
        <measures
            source_name="price"
            function="MIN"
            description="lowest price">
        </measures>
        <measures
            source_name="price"
            function="MAX"
            description="highest price">
        </measures>
        <measures
            source_name="price"
            function="SUM"
            description="total">
        </measures>
        <measures
            source_name="visitors"
            function="SUM"
            description="visitors of this constellation">
        </measures>
        <measures
            source_name="discount"
            function="MAX"
            description="highest discount">
        </measures>
        <measures
            source_name="additional_charge"
            function="MIN"
            description="lowest charge">
        </measures>
        <measures
            source_name="id"
            function="COUNT"
            description="total number">
        </measures>
        <calculated_m
            name="average"
            mdx=" [MEASURES].[sum_price] / [MEASURES].[count_id]">
        </calculated_m>
        <dimension
            name="destination"
            description="travel destinations"
            all_level="yes"
            all_caption="all destination">
            <level
                level1="continent"
                level2="country"
                level3="region"
                level4="area">
            </level>
            <member name="Europa">
                <member name="Spanien">
                    <member name="Costa del Sol">
                    </member>
                    <member name="Balearen">
                        <member name="Mallorca">
                        </member>
                        <member name="Ibiza">
                        </member>
                        <member name="Formentera">
                        </member>
                    </member>
                <member name="Kanaren">
                    <member name="Lanzarote">
                    </member>
                </member>
            </member>
        </dimension>
    </cube>
</catalog>
```

```

        <member name="Teneriffa">
        </member>
        <member name="Gran Canaria">
            <member name="Playa del Ingles">
            </member>
            <member name="Mogan">
            </member>
        </member>
        <member name="Fuerteventura">
            <member name="Jandia">
            </member>
        </member>
    </member>
</member>
<member name="Portugal">
    <member name="Algarve">
    </member>
</member>
</member>
<member name="Amerika">
    <member name="USA">
        <member name="Florida">
            <member name="Miami">
            </member>
            <member name="Naples">
            </member>
            <member name="St. Petersburg">
            </member>
            <member name="Orlando">
            </member>
            <member name="Jacksonville">
            </member>
        </member>
        <member name="Kalifornien">
            <member name="San Diego">
            </member>
            <member name="San Francisco">
            </member>
            <member name="Los Angeles">
            </member>
        </member>
        <member name="Nevada">
            <member name="Las Vegas">
            </member>
        </member>
    </member>
    <member name="Kuba">
    </member>
    <member name="Dom. Republik">
        <member name="Punta Cana">
            <member name="Bavaro">
            </member>
        </member>
        <member name="Puerto Plata">
        </member>
    </member>
    <member name="Mexiko">
        <member name="Acapulco">
        </member>
        <member name="Cancun">
        </member>
        <member name="Playa del Carmen">
        </member>
        <member name="Cozumel">
        </member>
    </member>
    <member name="Jamaica">
    </member>
</member>
<member name="Australien">
</member>
</dimension>
```

```
<dimension      name="departure_airport"
                description="German airports"
                all_level="yes">
  <level  level1="city"
          level2="airport">
  </level>
  <member name="Frankfurt">
    <member name="Rhein-Main">
    </member>
  </member>
  <member name="Duesseldorf">
  </member>
  <member name="Berlin">
    <member name="Tegel">
    </member>
    <member name="Schoenefeld">
    </member>
  </member>
  <member name="Stuttgart">
  </member>
  <member name="Hamburg">
  </member>
</dimension>
<dimension      name="category"
                description="international standard"
                all_level="yes">
  <level  level1="stars">
  </level>
  <member name="*">
  </member>
  <member name="**">
  </member>
  <member name="***">
  </member>
  <member name="****">
  </member>
  <member name="*****">
  </member>
  <member name="*****">
  </member>
</dimension>
<dimension name="boarding" all_level="yes">
  <level></level>
  <member name="lodging">
  </member>
  <member name="bed and breakfast">
  </member>
  <member name="halfboard">
  </member>
  <member name="full board">
  </member>
  <member name="all inclusive">
  </member>
</dimension>
<dimension name="company" all_level="yes">
  <level></level>
  <member name="TUI">
  </member>
  <member name="Neckermann">
  </member>
  <member name="Jahn Reisen">
  </member>
  <member name="LTur">
  </member>
  <member name="Mc Flight">
  </member>
</dimension>
<time_dimension name="offer_date">
  <td_level      all_level="yes"
                level1="Years"
                level2="Weeks"
                level3="Days">
  </td_level>
</time_dimension>
```

Anhang B: XML-Dokument

```
<time_dimension name="offer_deleted_date">
  <td_level      level1="Years"
                level2="Month"
                level3="Days">
    </td_level>
</time_dimension>
<time_dimension name="actuality">
  <td_level>
    </td_level>
</time_dimension>
<time_dimension name="departure_date">
  <td_level      all_level="no"
                level1="Years"
                level2="Month"
                level3="Days">
    </td_level>
</time_dimension>
<time_dimension name="arrival_date">
  <td_level      level1="Years"
                level2="Quarters"
                level3="Month"
                level4="Days">
    </td_level>
</time_dimension>
</cube>
</catalog>
```