Fachhochschule Liechtenstein

Einführung in MDX

Seminararbeit im Modul

Seminar II Wirtschaftsinformatik

WS 2004/ 05

Prof. Dr. Klaus Kruczynski

Autor:

Sebastian Willand

Matrikel-Nr.: 030240

Abstract

Nachdem im ersten Kapitel eine Abgrenzung der Thematik MDX vorgenommen wird, wird im zweiten Kapitel dieser Arbeit der Aufgabenbereich einer Anfragsprache wie MDX in den logischen Gesamtkontext des Data Warehouse Konzeptes eingegliedert.
In den Kapitel 3 und 4 wird dann explizit auf die multidimensionale Anfragesprache MDX eingegangen. Dabei wird zunächst der Unterschied zwischen der relationalen Anfragesprache SQL und MDX hervorgehoben. Im Anschluss daran erfolgt dann eine Einführung in die Syntax und die Anfragestruktur von MDX. Außerdem wird anhand von ausgesuchten Beispiel-Anfragen das Potential von MDX illustriert.
Zum Abschluss erfolgt noch ein kurzes Resümée, in welchem die Bedeutung von MDX für OLAP noch einmal hervorgehoben wird.

After an delimitation of the MDX-topic in the first chapter, the specific task of MDX as an query language in the context of data-warehouses is described in chapter 2.
Then, in the chapters 3 and 4, the multidimensional query language MDX is explicit introduced. First, the differences between the relational query language SQL and MDX are figured. After that, there is an introduction in the syntax and the query structure of MDX. Furthermore, the potential of MDX is emphasized by giving some examples of MDX-Queries.
Finally, there is a short summary at MDX at the end of this paper, where the impact of MDX for OLAP is emphasized one more time.

Inhaltsverzeichnis

Abbildungsverzeichnis

Formelverzeichnis

Vorwort

Diese Arbeit entstand im Rahmen des Moduls „Seminar Wirtschaftsinformatik II“ an der Fachhochschule Liechtenstein im Wintersemester 2004/05.
Während in der Vorlesung die grundlegenden Kenntnisse zur Thematik Data-Warehouse vermittelt wurden, ist die Seminararbeit dazu gedacht, dass die teilnehmenden Studenten einen speziellen Themenbereich aus dem Umfeld Data-Warehouse selbstständig näher untersuchen.

1 Zielstellung und Abgrenzung

Diese Arbeit hat zum Ziel, einen Einblick in die Funktionsweise der multidimensionalen Anfragesprache MDX zu geben. Sie ist gerichtet an all diejenigen, die sich im Rahmen ihrer Ausbildung, ihres Studiums oder von Berufswegen aus mit der Thematik Data-Warehouse auseinandersetzen und dabei etwas über die datenbank- und anfragetechnischen Sachverhalte in Erfahrung bringen möchten, die im Rahmen des Data-Warehousing zu berücksichtigen sind.

Dagegen hat die Arbeit nicht zum Ziel, ein umfassendes Kompendium zu MDX oder ähnliches darzustellen. Dieses würde zum einen den gesteckten Rahmen dieser Seminararbeit bei weitem übersteigen und zum anderen auch nicht der Absicht des Autors entsprechen.

Vielmehr soll im Rahmen dieser Arbeit geklärt werden, weshalb eine Anfragesprache wie MDX in Data-Warehouse-Systemen, insbesondere, wenn diese auf relationalen Datenbanken aufbauen, mehr als nützlich ist und weshalb sich die seit Jahren etablierte und standardisierte Anfragesprache SQL in diesem Kontext – ohne entsprechende multidimensionale Erweiterung – als unzureichend erweist.

Vor diesem Hintergrund will diese Arbeit einen Einblick in die Funktionsweise und die Einsatzmöglichkeiten von MDX geben und dabei auch die Syntax von MDX sowie die Unterschiede zu SQL illustrieren.

2 Anfragesprachen im Data-Warehouse-Kontext

2.1 Das Konzept des Data-Warehouse

Das Konzept des Data-Warehouse spiegelt den aktuellen Stand der Entwicklungen auf dem Gebiet der Integrationsbemühungen im Rahmen unternehmensweiter Managementinformationssysteme wider.[1]
Dabei ist zu beachten, dass bisher noch keine einheitliche oder gar normierte Begriffsdefinition zu „Data-Warehouse“ existiert. Die „Ur-Definition“ des gedanklichen Vaters des Data-Warehouse, William Inmon, lautet:

> “A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management´s decision-making process.“[2]

Generell wird mit dem Begriff „Data-Warehouse“ ein von den operativen DV-Systemen strikt getrenntes Datenbanksystem verstanden, welches die unternehmensweite Datenintegration sicherstellen und durch Kopieren und Aufbereiten von (operativen) Daten aus unterschiedlichsten Quellen entsteht.[3] Ziel des Data-Warehouse-Konzeptes ist die Vereinheitlichung der Unternehmensdaten. Ein Data-Warehouse ist demnach eine Datenbankanwendung, die speziell für analytische Auswertungszwecke gedacht ist und stets dispositiven Charakter hat.

2.2 Aufbau von Data-Warehouse-Systemen

Data-Warehouse-Systeme unterscheiden sich stets signifikant von anderen, im operativen Geschäft eingesetzten Datenbanksystemen. Versucht man nun, die wesentlichen Unterscheidungsmerkmale herauszuarbeiten, so stellt man fest, dass man dies aus zwei verschiedenen Betrachtungswinkeln tun kann: zum einen aus der

---

[1] Vgl. Muksch, Harry/ Behme, Wolfgang (2000), S. 6

[2] Inmon, W. H./ Hackethorn, R. D. (1994) Using the Data Warehouse, John Wiley & Sons, New York

[3] Vgl. Muksch, Harry/ Behme, Wolfgang (2000), S. 6

Perspektive der Anwender und zum anderen aus der Sicht der Datenhaltung.

2.2.1. Aus Sicht der Anwender

Aus Sicht der Anwender zeichnen sich operative Datenbanksysteme, d.h. Datenbanksysteme, die im Rahmen der operativen Geschäftstätigkeit eingesetzt werden, besonders durch eine transaktionale Ausrichtung aus. Kennzeichnend für diese Art von Datenbanksystemen sind - aus technischer Sicht - viele kurze Lese- und Schreibtransaktionen von meist einzelnen Datensätzen pro Zeitraum t, die wiederum durch eine relativ hohe Zahl von Benutzern generiert werden.

Dahingegen ist es für analytisch ausgerichtete Data-Warehouse-Systeme typisch, dass relativ wenige Zugriffe auf die Datenbasis pro Zeitraum t erfolgen, wobei es sich im Wesentlichen um Lesezugriffe handelt, bei denen dann meist eine Vielzahl von Datensätzen auf einmal abgefragt werden.

Weiterhin ist zu beachten, dass Zugriffe auf die Datenbasis bei den operativen Systemen stets deterministisch erfolgen, d.h. Anfragen an die Datenbank erfolgen nach einem starren Muster. Dies zeigt sich etwa daran, dass der Anwender in einem operativen System stets nur Anfragen mit fest vorgegebener Struktur durchführen kann. Ein wesentliches Kennzeichen für operative Datenbanksysteme ist schließlich der Anfragetypus: so ist dieser zumeist von einfacher Gestalt, d.h. Lese- oder Schreibzugriffe beziehen jeweils nur eine geringe Anzahl von Tabellen[4] mit ein.

Im Falle eines Data-Warehouse-Systems stellen sich die Anfragen an die Datenbank dagegen in der Regel weitaus komplexer dar. So ist es an der Tagesordnung, dass bei Anfragen eine Vielzahl von Tabellen über sog. „Joins“ miteinander verknüpft werden und der Anwender darüber hinaus

[4] Anmerkung des Autors: der Begriff „Tabelle“ wird hier synonym mit dem Begriff „Relation“ gebraucht

nicht nur vorgegebene Anfragen durchführen, sondern auch eigene, individuelle Anfragen generieren kann.[5]

2.2.2. Aus Sicht der Datenhaltung

In operativen Systemen finden sich typischerweise nur aktuelle Daten, d.h. die Daten der operativen Systeme sind volatil. In der Praxis bedeutet dies, dass die Daten, sobald sie sich ändern, mit den neueren Werten überschrieben werden. In einem Data-Warehouse werden dagegen auch historische Daten gesammelt, d.h. hier liegt eine Nicht-Volatilität der Daten vor. Statt die Daten mit den aktuellen Werten zu überschreiben, findet vielmehr eine Versionierung bzw. Historisierung der Daten statt. Diese Art der Datenhaltung bläht das Datenvolumen zwar immens auf, ist jedoch notwendig, um zeitraumbezogene Analysen und Auswertungen vornehmen zu können.

Ein ganz wesentliches Unterscheidungsmerkmal zwischen den betrachteten Systemarten ist schließlich die Datenstruktur. Im Bereich der operativen, transaktionalen Datenbanksysteme hat sich auf breiter Front eine zweidimensionale Datenstruktur auf Basis des relationalen Datenmodells durchgesetzt. Die Daten werden hier in zweidimensionalen Tabellen (Relationen) gespeichert, wobei die Spalten die verschiedenen Attribute der Daten beschreiben, während die Zeilen die einzelnen Datensätze (Tupel) beinhalten. Die große Stärke dieser Form der Datenhaltung liegt darin, große Volumina an Daten aufnehmen zu können. Weiterhin gestalten sich Datenmanipulationen (Einfügen, Ändern, Löschen) als unproblematisch, d.h. das Relationenmodell steht für eine äußerst robuste Form der Datenhaltung. Schließlich gewährleistet das Relationenmodell wirkungsvoll eine redundanzfreie Datenhaltung, was gerade in transaktionsorientierten Systemen von großer Wichtigkeit ist. Dagegen sind relationale Datenstrukturen nicht mit dem Ziel entwickelt worden, große Datenmengen für Analysezwecke zu verwalten.[6]

---

[5] Vgl. Lehner, Wolfgang (2003), S. 16-18

[6] Vgl. Muksch, Harry/ Behme, Wolfgang (2000), S. 150

In analyseorientierten Datenbanksystemen spielen dagegen Kriterien wie Redundanzfreiheit eine untergeordnete Rolle, ja es werden teilweise sogar bewusst Redundanzen erzeugt, um etwa - zum Zeitpunkt der Anfrage - aufwendige Aggregationen zu vermeiden. Hier steht vielmehr die Fähigkeit zur Analyse im Vordergrund. Für die Struktur der Daten bedeutet das, dass es nicht ausreicht, die Daten in herkömmlichen zweidimensionalen Relationen abzulegen. Aus diesem Grund haben sich in den letzten Jahren multidimensionale Datenstrukturen für den Aufbau von Data-Warehouse-Systemen etabliert.[7]

2.2.3. OLAP und Multidimensionalität

Der Begriff „OLAP“ steht für „Online Analytical Processing“ und umschreibt die Analyse der Daten aus dem Data-Warehouse. Die OLAP-Komponente bildet damit gewissermaßen das Front-End eines Data-Warehouse-Systems, mit der das Management arbeitet und versucht, aus den Daten, die im Data-Warehouse abgelegt sind, Informationen zur Führung des Unternehmens zu gewinnen.

Die Thematik OLAP verlangt nun geradezu multidimensionale Datenstrukturen. Dies wird deutlich, wenn man sich vergegenwärtigt, welche Informationen sich das Management von einem Date-Warehouse-System verspricht. Es geht in erster Linie um quantitative Informationen in Form von Kennzahlen, da Kennzahlen zweifellos ein elementares Instrument zur Führung von Unternehmen und zur Unterstützung von Entscheidungsprozessen darstellen. Dabei lassen sich grundsätzlich Basiskennzahlen (z.B. Umsatz) und abgeleitete Kennzahlen (z.B. Deckungsbeitrag) unterscheiden. Gerade bei den abgeleiteten Kennzahlen ist es nun von großer Bedeutung, dass diese stets transparent, d.h. nachvollziehbar bleiben.[8] Außerdem versucht das Management typischerweise, einen Sachverhalt aus verschiedenen Perspektiven zu betrachten und anhand mehrerer Kriterien zu analysieren, was häufig komplexe Anfragen an das System zur Folge hat.

---

[7] Vgl. Muksch, Harry/ Behme, Wolfgang (2000), S. 151

[8] Vgl. Muksch, Harry/ Behme, Wolfgang (2000), S. 150

Anders ausgedrückt: betriebswirtschaftliche Daten für die Entscheidungsunterstützung sind von Natur aus multidimensional.[9] Eine typische betriebswirtschaftliche Fragestellung im Rahmen von OLAP könnte etwa lauten: „zeige mir den Absatz von Produkt XY pro Verkaufsregion und pro Quartal“. In diesem Fall beinhaltet die Anfrage drei Dimensionen, nämlich die Dimension „Produkt“, die Dimension „Verkaufsregion“ und die Dimension „Zeitraum“.

Wie kann man sich nun eine multidimensionale Datenstruktur vorstellen? Grundsätzlich lässt sich eine multidimensionale Anordnung von Daten in Form eines Würfels (Cube) illustrieren (siehe Abbildung 1).

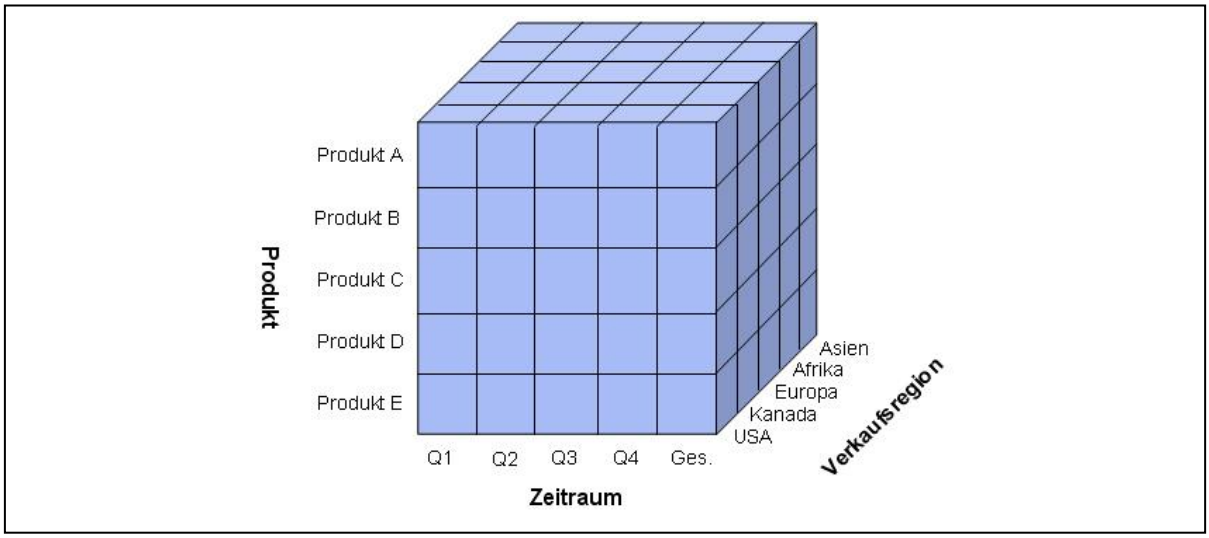


Abbildung 1: Multidimensionale Datenstruktur, dargestellt in Form eines Cube, eigene Darstellung

Dabei ist jedoch zu beachten, dass die Würfelform allein deshalb gewählt wird, weil sich in einer Graphik nicht mehr als drei Dimensionen vernünftig darstellen lassen – eine multidimensionale Datenbankstruktur umfasst in der Praxis jedoch eine weitaus größere Anzahl an Dimensionen.

Im Data Warehouse wird nun der qualifizierende Anteil eines multidimensionalen Datenbankschemas durch das Dimensionenkonzept dargestellt. Eine Dimension erhält man, indem die gesammelten Daten klassifiziert und im Anschluss kategorisiert, d.h. unter einem gemeinsamen Obergriff, eben einer Dimension, zusammengefasst

[9] Vgl. Totok, Andreas (1997), S. 23

werden.[10] Die gewählten Dimensionen sind dabei abhängig vom betriebswirtschaftlichen Kontext und der Art der Analysen. Allerdings existiert eine Reihe von „Standard-Dimensionen“, die sich praktisch in jedem Data-Warehouse-System finden lassen. Dazu gehören etwa die Dimension Zeit oder auch gängige Maßeinheiten.[11] Die Dimensionen werden wiederum vertikal in verschiedene Ebenen unterteilt. Das Ergebnis ist ein Konsolidierungspfad, der sich von der untersten bis zur obersten Dimension erstreckt. Man spricht hier auch von einer Dimensionshierarchie. Als Beispiel sei die Dimension Zeit genannt, bei der die einzelnen Dimensionsebenen in einem direkten hierarchischen Zusammenhang stehen (siehe Abbildung 2).

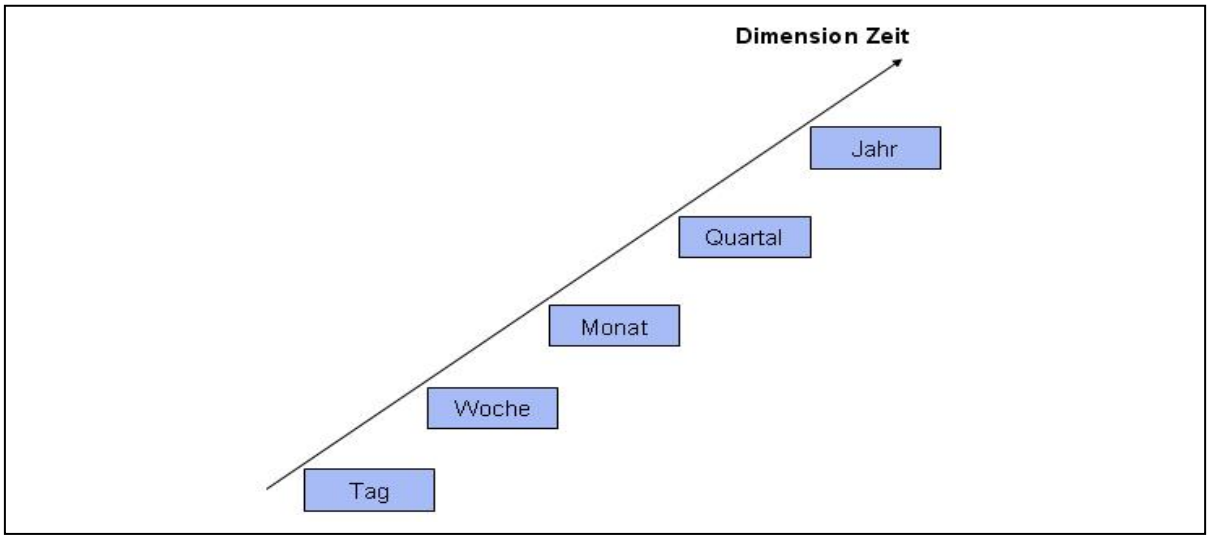


Abbildung 2: Die Zeit-Dimension mit ihren verschiedenen Ebenen, eigene Darstellung

Schließlich wird der quantifizierende Anteil eines multidimensionalen Datenbankschemas durch Basiskennzahlen („Fakten“) bestimmt, aus denen wiederum abgeleitete Kennzahlen generiert werden können. Zusammen mit dem Dimensionenkonzept entsteht auf diese Weise ein multidimensionaler Datenwürfel.[12] Das große Potential dieser Form der Datenstrukturierung liegt in der Möglichkeit der vielseitigen und flexiblen

[10] Vgl. Lehner, Wolfgang (2003), S. 65

[11] Vgl. Totok, Andreas (1997), S. 24

[12] Vgl. Lehner, Wolfgang (2003), S. 67

Navigation durch die Datenbestände, was wiederum Voraussetzung für aussagekräftige OLAP-Analysen ist.

2.3 Zugriff auf die Daten in einem Data-Warehouse

Um nun adäquat OLAP betreiben zu können, muss man in der Lage sein, in geeigneter Art und Weise auf die im Data Warehouse abgelegten multidimensionalen Daten zugreifen zu können. Hier existieren nun zwei generelle Möglichkeiten. Zum einen kann eine Navigation durch die Daten bzw. eine Anfrage der Daten über sog. Navigationsoperatoren erfolgen. Zum anderen können die Daten aus dem Data-Warehouse auch mit Hilfe von Anfragesprachen abgerufen werden.

2.3.1. Navigationsoperatoren

Datenbanktechnisch existieren für OLAP eine Reihe von Mechanismen bzw. Operatoren, über die eine Navigation durch multidimensionale Datenstrukturen ermöglicht werden kann Diese sog. Navigationsoperatoren nutzen dabei die vordefinierten Anwendungsstrukturen in den Dimensionshierarchien aus.[13]

Zu den wichtigsten Vertretern dieser Navigationsoperatoren gehören u.a.:

- Drill Down

  Hier wird quasi die „Auflösung“ des Datenwürfels verändert, d.h. es wird hier entlang eines Konsolidierungspfades nach unten gegangen, um, ausgehend von Ergebnissen höherer Granularität, zu detaillierten Ergebnissen feiner Granularität zu gelangen.[14] Bsp.: von „Umsatz in Deutschland“ zu „Umsatz in München“

- Drill Up

  Drill Up ist die gegenteilige Operation zu Drill Down; es erfolgt eine Konsolidierung von einer Detailstufe hin zu einer Stufe höherer Granularität[15]

---

[13] Vgl. Lehner, Wolfgang (2003), S. 82

[14] Vgl. Lehner, Wolfgang (2003), S. 74

[15] Vgl. Lehner, Wolfgang (2003), S. 75

- Dicing

  Mittels Dicing lassen sich nur die interessierenden Attribute einer Dimension betrachten, während alle anderen ausgeblendet werden. So erfolgt ein Dicing, wenn sich z.B. ein Manager nur die Sollwerte, nicht aber die Istwerte zu den Absatzzahlen eines Produktes innerhalb einer Region anzeigen lässt. Beim Dicing wird demnach ein kleinerer Würfel als ursprünglich erzeugt[16]

- Slicing

  Beim Slicing werden einzelne Scheiben aus dem multidimensionalen „Datenwürfel“ geschnitten.[17] Auf diese Weise lassen sich punktuelle Betrachtungen auf einer bestimmten Ebene durchführen, z.B. die Verkaufszahlen eines Produktes an einem bestimmten Tag (z.B. dem letzten Samstag vor Weihnachten)

Nun reichen diese Standard-Mechanismen jedoch nicht für sämtliche gewünschten OLAP-Anfragen aus. Die Generierung komplexer multidimensionaler Ergebniswürfel kann über diese Operatoren bisweilen nicht zufrieden stellend realisiert werden. Es bedarf somit in jedem Fall geeigneter Anfragesprachen, um das Potential von OLAP tatsächlich in vollem Umfang ausschöpfen zu können.

2.3.2. Anfragesprachen für OLAP

Eine Anfragesprache, die für OLAP eingesetzt wird, muss in der Lage sein, multidimensionale Anfragen möglichst einfach und effizient umsetzen zu können. Wichtig ist dabei insbesondere, dass auch Anfragen von hoher Komplexität ausdrückbar sind. Fragestellungen, welche die Komplexität einer OLAP-Anfrage beträchtlich steigern können, sind z.B. der Wunsch nach mathematischen oder statistischen Grundoperationen oder auch die Visualisierung der Ergebnisse.

Um dieser Komplexität gerecht werden zu können, sind in den letzten Jahren verstärkt Anstrengungen dahingehend unternommen worden, geeignete Anfragesprachen für OLAP zu entwickeln.

---

[16] Totok, Andreas (1997), S. 29

[17] Vgl. Bauer, Andreas et al. (2001), S. 106

Eine gute Anfragesprache für OLAP muss es erlauben, dass alle für das Management einer Unternehmung relevanten und nützlichen Anfragen in ihr formuliert werden können. Gleichzeitig muss eine solche Sprache gewährleisten, dass all die in ihr formulierten Anfragen in der Praxis auch mit vertretbarem Aufwand ausgewertet werden können.[18] Schließlich sollte eine Anfragesprache nicht zu komplex und damit nur schwer erlernbar sein.

Bei der Planung und Konzeption eines Data Warehouse ist nun stets eine Entscheidung dahingehend zu treffen, wie und in welcher Form das entwickelte multidimensionale Datenmodell auf das interne Datenbankschema des Data Warehouse abgebildet wird und wie dementsprechend auch der Zugriff auf die multidimensionalen Datenstrukturen erfolgen soll. Dabei bieten sich zwei grundsätzliche Alternativen, nämlich das sog. „ROLAP“ (relationales OLAP) und das sog. „MOLAP“ (multidimensionales OLAP) an.

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 aus dem OLAP-System werden hierbei als SQL-Anfragen an das zugrunde liegende relationale Datenbankmanagementsystem übertragen.

Werden die Daten hingegen analog zum Datenmodell in mehrdimensionalen Datenstrukturen gespeichert, so handelt es sich um MOLAP-Systeme (Multidimensional OLAP).

Betrachtet man das relationale OLAP, so wurden dort in den letzten Jahren verschiedene Anstrengungen im Bereich der Entwicklung geeigneter Anfragesprachen unternommen. Typischerweise wird hier direkt auf der relationalen Standard-Anfragesprache SQL aufgesetzt, wobei diese dann um multidimensionale Aspekte erweitert wird. So ist beispielsweise im SQL:1999-Standard der bisherige SQL-Standard um den sog. Data-Cube-Operator ergänzt worden. Diese Erweiterung macht es möglich, dass via

[18] Vgl. Vossen, Gottfried (2000), S. 397 f.

SQL Kreuztabellen, Super-Aggregate und Roll-Up-Berichte generiert werden können.[19]

Im Bereich der multidimensionalen Anfragesprachen ist neben diversen objektorientierten Ansätzen insbesondere der Ansatz des Unternehmens Microsoft zu nennen. So hat Microsoft im Zuge seines Produktes „Microsoft SQL Server“ eine komplett eigene Anfragesprache entwickelt, die sich in ihrer Syntax zwar stark an SQL orientiert, dennoch aber eine eigenständige multidimensionale Sprache darstellt. Diese Sprache trägt das Kürzel „MDX“ und steht für „Multidimensional Expressions“. Die Sprache ist eingebettet in die Datenbankschnittstelle „OLE DB for OLAP“, welche wiederum bereits von über 40 Herstellern von OLAP-Tools unterstützt wird und damit bereits eine gewisse Interoperabilität zwischen verschiedenen Anwendungssystemen gewährleistet. MDX hat – wohl nicht zuletzt deshalb, weil der Software-Riese Microsoft dahinter steht - gegenwärtig wohl die besten Aussichten, sich zu einem echten Standard zu entwickeln, ähnlich wie es auch bei SQL im Bereich der traditionellen relationalen Datenbanken der Fall war.[20] Aus diesem Grund wird diese Sprache auch im weiteren Verlauf dieser Arbeit näher vorgestellt.

[19] Vgl. Rondot, René (2003), S. 6

[20] Vgl. Lehner, Wolfgang (2003), S. 80 f.

3 Die multidimensionale Anfragesprache MDX

3.1 Möglichkeiten und Grenzen von SQL

Es wurde bereits oben erwähnt, dass gegenwärtig noch ein großer Teil der existierenden Data-Warehouse-Systeme auf relationalen Datenbanken basiert. Nun mag man sich fragen, warum dann nicht einfach die voll etablierte und zum Quasi-Standard avancierte Datenbanksprache SQL benutzt wird, um OLAP-Anfragen durchzuführen.

Dazu ist zunächst einmal sagen, dass SQL durchaus für OLAP in Frage kommt und dies nicht nur – wie oben ebenfalls angesprochen – in Form von spezifischen Erweiterungen. Allerdings sind dem Einsatz von SQL für OLAP Grenzen gesetzt.

Die Datenbanksprache SQL wurde primär für klassische relationale Datenbanken entworfen – also für Datenbanken, die im operationalen Betrieb (OLTP) eingesetzt werden. In diesem Kontext werden die anfallenden Daten einerseits typischerweise in zweidimensionalen Tabellenstrukturen abgelegt und andererseits finden nur einfache Anfragen an die Datenbasis statt. Dagegen wurde SQL nicht für komplexe OLAP-Anfragen entwickelt, was logischerweise zur Folge hat, dass SQL nur begrenzt für die Formulierung multidimensionaler Anfragen geeignet ist.

Einfache Slice&Dice–Operationen können auf jeden Fall durchaus mit Hilfe von SQL formuliert werden. Auch einfache Aggregationen lassen sich mit SQL noch vernünftig realisieren. Denn für derartige Operationen reichen die bordeigenen Mittel der SQL-Syntax noch aus (JOIN, SELECT, GROUP by).[21]

Um die Möglichkeiten und Grenzen von SQL im Bereich OLAP etwas zu veranschaulichen, wird im Folgenden ein Beispiel für ein mögliches Zusammenspiel von SQL und OLAP gegeben. Zunächst jedoch noch einige einführende Informationen: um ein multidimensionales Datenmodell auf ein relationales Datenbanksystem abbilden zu können, d.h. in

[21] Vgl. Leser, Ulf (2004), Folie 22

tabellarische Form zu bringen, müssen die Daten stets in zwei Gruppen klassifiziert werden, nämlich in Fakt- und Dimensionsdaten. Entsprechend werden zwei grundlegende Arten von Tabellen benötigt – nämlich die sog. Fakten- und Dimensionstabellen. Die Faktentabellen enthalten hierbei die zu analysierenden betriebswirtschaftlichen Kennziffern. Ferner verweist der zusammengesetzte Primärschlüssel einer jeden Faktentabelle auf die unterste Ebene der die Fakten beschreibenden Dimensionen. Die Dimensionstabellen enthalten ihrerseits die deskriptiven Daten der Dimensionen.[22] Es existieren verschiedene Ausprägungen dieser Form der Datenstrukturierung, bekannte Vertreter sind u.a. das „Star-Schema“ und das „Snowflake-Schema“. Auf die Unterschiede zwischen den einzelnen Schemata soll an dieser Stelle nicht näher eingegangen werden – vielmehr sei auf die einschlägige Literatur zu dieser Thematik verwiesen.[23]

Gegeben sei nun folgendes Star-Schema:

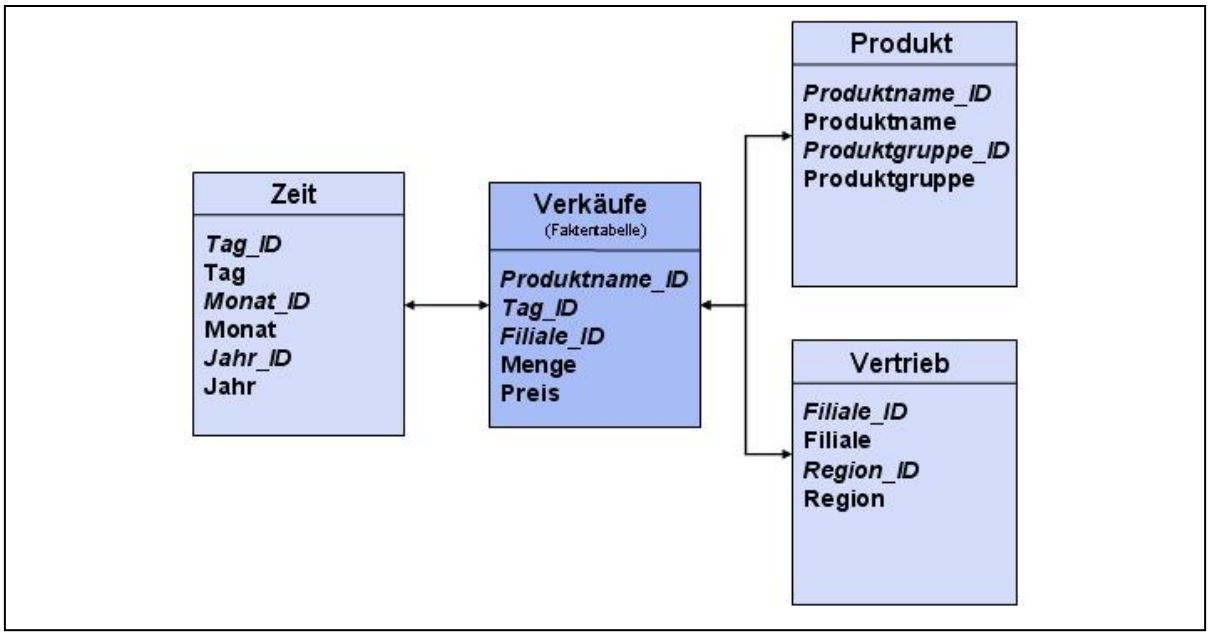


Abbildung 3: relationales Star-Schema, in Anlehnung an Leser, Ulf (2004), Folie 23

Will man nun beispielsweise den Verkaufsumsatz eines bestimmten Produktes „XY“ in einer bestimmten Filiale, aufgeschlüsselt nach Tagen,

[22] Vgl. Kruczynski, Klaus (2004), S. 42, DWH_DaMi_1.pdf

[23] Siehe etwa Bauer, Andreas/ Günzel, Holger (2001), S. 197 ff.

aus den Daten ermittelt haben, so kann dies mittels folgendem SQL-Statement geschehen:[24]

```
SELECT Z.Tag_ID, sum(Menge*Preis) FROM Verkäufe V, Produkt P, Zeit Z,
Vertrieb Vb
WHERE P.Produktname = "XY" AND P.Produktname_ID = V.Produktname_ID AND
Z.Tag_ID = V.Tag_ID AND Vb.Filiale_ID = V.Filiale_ID
GROUP BY Z.Tag_ID
```

Gleichung 1: SQL-Statement, in Anlehnung an Leser, Ulf (2004), Folie 27

Gleichung 1 ist an sich ein Beispiel dafür, dass OLAP-Anfragen via SQL durchaus möglich und akzeptabel sind. Erweitert man allerdings das Beispiel aus Gleichung 1, indem man z.B. den Verkaufsumsatz des Produktes „XY" in einer bestimmten Filiale, aufgeschlüsselt nicht nur nach Tagen, sondern auch nach Monaten und Jahren anfragt, so stößt man bereits an die Grenzen des sinnvollen Einsatzes von SQL für OLAP. Denn nach dem reinen SQL-Standard ist eine solche Anfrage nicht in einem Query, also innerhalb eines Anfrage-Statements, in SQL formulierbar.

```
SELECT Z.Jahr_ID, Z.Monat_ID, Z.Tag_ID, sum(Menge*Preis) FROM Verkäufe V,
Produkt P, Zeit Z, Vertrieb Vb
WHERE P.Produktname = "XY" AND P.Produktname_ID = V.Produktname_ID AND
Z.Tag_ID = V.Tag_ID AND Vb.Filiale_ID = V.Filiale_ID
GROUP BY Z.Jahr_ID, Z.Monat_ID, Z.Tag_ID
```

Gleichung 2: SQL-Statement, in Anlehnung an Leser, Ulf (2004), Folie 26

Eine Anfrage, formuliert wie in Gleichung 2 würde lediglich die Summen nur für die Tage, unterteilt nach Monaten und Jahren, als Ergebnis liefern, jedoch keine Summen pro Monat bzw. pro Jahr. Vielmehr wären für die gewünschte Anfrage ein Query pro Klassifikationsstufe nötig – in diesem Fall somit drei Queries. Überdies lässt dann die Ergebnisdarstellung zu wünschen übrig, da SQL eine nur unbefriedigende Sortierung der Ergebnisse liefert.

[24] Vgl. Leser, Ulf (2004), Folie 27

3.2 MDX ist nicht SQL

Da die Möglichkeiten des reinen SQL für die komplexen Anfragen in OLAP zu begrenzt sind und zudem eine Anfragesprache wünschenswert wäre, die auf unterschiedliche Arten von Datenbanksystemen (relational, multidimensional) einsetzbar ist, erscheint es nur folgerichtig, dass die Firma Microsoft mit MDX eine Anfragesprache entwickelt hat, die sowohl technikübergreifend einsetzbar ist (über die Datenbankschnittstelle OLE DB for OLAP praktisch mit allen OLAP-Systemen, die diese Schnittstelle unterstützen, interoperabel) als auch den sprachlichen Anforderungen komplexer OLAP-Anfragen gerecht werden kann.

MDX wurde speziell für OLAP-Anfragen an multidimensionale Datenwürfel geschaffen. Entsprechend stehen mit MDX eine ganze Reihe von Funktionalitäten zur Verfügung, um in einer Weise auf multidimensionale Daten zugreifen zu können, wie es in SQL so nicht möglich ist.

Um eine erste Vorstellung von der Syntax von MDX zu erhalten, nachfolgend ein beispielhaftes MDX-Statement:

```
SELECT {[Deutschland], [Bayern], [Frankfurt], [Erlangen]} ON COLUMNS,
{[Quartal1].Children, [Quartal2], [Quartal3], [Quartal4].Children} ON ROWS
FROM [SalesCube]
WHERE ([Measures].[Verkäufe], [Zeit].[2002], [Produkte].[Alle Produkte]);
```

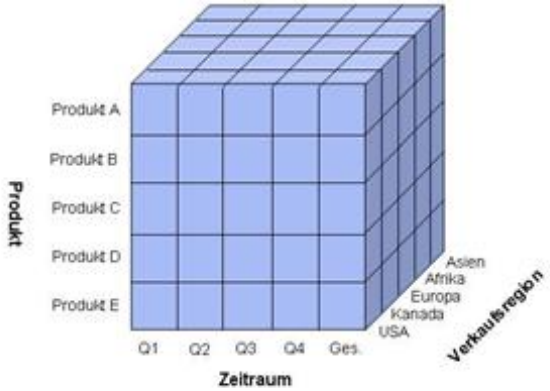
Gleichung 3: MDX-Statement [1], nach Lehner, Wolfgang (2003), S. 76

Es fällt sogleich die syntaktische Ähnlichkeit zu SQL auf, weshalb MDX auch häufig als multidimensionale Erweiterung von SQL bezeichnet wird. Dies ist jedoch nicht der Fall! Zwar orientiert sich MDX stark an der SQL-Syntax, dennoch handelt es sich bei MDX um eine völlig eigenständige Anfragesprache, die speziell für OLAP entwickelt wurde.

> "MDX is similar in many ways to the Structured Query Language (SQL) syntax, but is not an extension of the SQL language; in fact, some of the functionality that is supplied by MDX can be supplied, although not as efficiently or intuitively, by SQL."[25]

[25] Microsoft Internetpräsenz [1]

Dennoch bestehen, neben der Syntax, noch einige weitere Gemeinsamkeiten zwischen MDX und SQL. MDX erlaubt es, OLAP-Datenwürfel in einer Weise anzufragen, in der es SQL erlaubt, zweidimensionale Tabellenstrukturen anzufragen.[26] Sowohl bei SQL als auch bei MDX werden die Anfrageergebnisse dabei in Form von zweidimensionalen Tabellen dargestellt (siehe Abbildung 4), wobei bei MDX in der Regel verschachtelte Tabellen generiert werden, um die Multidimensionalität der Daten hinreichend abzubilden.



	Q1					Q2					Q3					Q4					GESAMT				
	USA	Kanda	Europa	Afrika	Asien	USA	Kanda	Europa	Afrika	Asien	USA	Kanda	Europa	Afrika	Asien	USA	Kanda	Europa	Afrika	Asien	USA	Kanda	Europa	Afrika	Asien
Produkt A																									
Produkt B																									
Produkt C																									
Produkt D																									
Produkt E																									

Abbildung 4: MDX – vom Datenwürfel zur Tabelle, eigene Darstellung

Schließlich wird die Schnittstelle OLE DB for OLAP, auf der MDX aufsetzt, bereits von über 40 namhaften Herstellern von Datenbanksystemen unterstützt. Es sieht demnach gegenwärtig ganz danach aus, dass MDX in einigen Jahren die Quasi-Standardanfragesprache für OLAP-Systeme sein wird – ähnlich wie es SQL schon seit längerer Zeit für relationale Datenbanksysteme ist.

[26] Vgl. Whitehorn, Mark et al. (2004), S. XIII

4 Anfragen in MDX

4.1 MDX – Definitionen und Terminologien

An dieser Stelle seien kurz die grundlegenden Definitionen und Terminologien von MDX näher erörtert, um ein Verständnis für die Begriffswelt und die Funktionsweise von MDX zu erhalten.

Wie oben bereits dargestellt wurde, werden multidimensionale Datenstrukturen durch Datenwürfel (MDX: cubes) repräsentiert. Ein solcher Würfel hat stets eine Vielzahl von Dimensionen (MDX: dimension). Jede dimension besteht nun aus mehreren Hierarchiestufen, die in MDX als „levels“ bezeichnet werden. Jeder level verfügt seinerseits über eine Anzahl von Knoten, die sog. „members“.

Beispiel dimension Zeit:
Levels der dimension Zeit: Jahr, Quartal, Monat
Members des Dimensionslevels Quartal: Q1, Q2, Q3, Q4

Die Werte bzw. die Fakten, die ein cube enthält, werden in den einzelnen Zellen (MDX: cells) des Würfels hinterlegt und in MDX als „measures“ bezeichnet. Bei den measures handelt es sich also um die Werte, auf die mit Hilf der dimensions zugegriffen werden soll.[27] Measures sind meist numerisch und repräsentieren z.B. Preise, Kosten, und ähnliches. Measures sind weiterhin stets von einem bestimmten Datentyp (z.B. integer) und haben ein bestimmtes Format (z.B. Währung). Wichtig zu wissen ist ferner, dass die Gesamtheit alle measures in MDX wiederum als eine dimension betrachtet wird.[28]

Erfolgt eine Anfrage via MDX auf die Datenbasis, so wird als Ergebnis entweder der Wert einer einzelnen cell oder aber zumeist ein Block von cells geliefert. Damit die cells eines cube auch tatsächlich extrahiert werden können, muss man sie zunächst zuverlässig identifizieren können.

---

[27] Vgl. Brosius, Gerhard (2001), S. 27

[28] Vgl. Microsoft Internetpräsenz [2]

Dies geschieht mittel sog. „tuples“. Ein tuple ist also quasi eine Art Zeiger, der einzelne Zellen referenziert. Eine ausführliche Definition zu „tuple“ liefern Whitehorn et al.:[29]

„ A tuple is defined as an intersection of exactly a single member from each dimension (hierarchy) in the cube. […] A tuple always identifies […] a single cell in the multi-dimensional matrix. That could be an aggregate or a leaf level cell, but nevertheless one cell and only one cell is ever implied by a tuple.”

Mehrere tuple wiederum bilden ein sog. „set“, d.h. ein set ist eine Ansammlung mehrer tuples derselben Dimensionalität.[30]

4.2 Aufbau einer MDX-Anfrage im Vergleich zu SQL

Wie aus Gleichung 3 bereits hervorgeht, werden in MDX wie auch in SQL Anfragen stets mit dem SELECT- Kommando eingeleitet. Auch sonst ähneln sich die beiden Sprachen, was den Aufbau von Anfrage-Kommandos angeht. Doch auch wenn in MDX teilweise identisch lautende Befehle wie in SQL zum Einsatz kommen, so unterscheiden sich die Kommandos in beiden Sprachen doch mehr oder weniger deutlich voneinander.

In SQL wird der SELECT-Befehl innerhalb einer Anfrage dazu benutzt, um das Layout der Spalten festzulegen, d.h. es werden die für die gewünschte Ergebnismenge benötigten Spalten einer oder mehrer Tabellen ausgewählt. Aus welchen Tabellen nun diese Spalten entnommen werden sollen, wird mit der FROM-Anweisung festgelegt. Mittels der (optionalen) WHERE-Klausel kann schließlich das Zeilen-Layout bestimmt werden, d.h. die Ergebnismenge wird nach Zeilen (Datensätzen) gefiltert, die einem bestimmten Kriterium entsprechen.[31]

In MDX wird dagegen die SELECT-Anweisung dazu benutzt, um die Achsendimensionen zu bestimmen. Die Achsendimensionen beschreiben den Ergebnisraum, wobei jeder Dimension bereits eine Rolle zugewiesen

---

[29] Whitehorn, Mark, et al. (2004), S. 25

[30] Vgl. Whitehorn, Mark, et al. (2004), S. 14 ff.

[31] Vgl. Hernandes, M. et al. (2001), S. 104

wird (ON COLUMNS, ON ROWS). Weiterhin werden innerhalb der SELECT-Klausel normalerweise eine Menge von Klassifikationsknoten (members) ausgewählt, wobei diese aus unterschiedlichen Klassifikationsebenen (levels) stammen können. Die FROM-Klausel gibt in MDX die Menge der Datenwürfel an, aus denen der Ergebniswürfel generiert werden soll. Wird mehr als ein Datenwürfel herangezogen, ist zu beachten, dass die Datenwürfel jeweils mindestens eine gemeinsame Dimension aufweisen müssen, damit eine Verbundkompatibiltät gewährleistet ist. Die WHERE-Anweisung wird in MDX schließlich dazu benutzt, um den Ergebnisraum des zu Grunde liegenden Datenwürfels einzugrenzen. Dabei können in der WHERE-Klausel nicht nur measures benannt werden, sondern auch dimensions bzw. deren members. Durch die Definition einer WHERE-Klausel findet in MDX damit ein echtes Slicing statt.[32]

Zusammenfassend lässt sich festhalten, dass eine MDX-Anfrage typischerweise wie folgt aufgebaut ist:

```
SELECT axis_specification ON COLUMNS, axis_specification ON ROWS
FROM cube_name
WHERE slicer_specification
```

Gleichung 4: Bestandteile einer MDX-Anfrage, aus: Böhm, Klemens (2004), Folie 15

Zur weiteren Illustration ist in

t

```
SELECT {[Verkaufsregion].[Kontinent].[USA], [Verkaufsregion].[Kontinent].
[Kanada]} ON COLUMNS, {[Zeitraum].[Quartal].[Q1.], [Zeitraum].[Quartal].[Q2],
[Zeitraum].[Quartal].[Q3], [Zeitraum].[Quartal].[Q4]} ON ROWS
FROM [VerkaufsCube]
WHERE ([Measures].[Umsatz], [Zeitraum].[Jahr].[2004]);
```

Gleichung 5: MDX-Statement [2], eigene Darstellung

Die SELECT-Anweisung wählt zum einen die members „USA“ und „Kanada“ des levels „Kontinent“ der Achsendimension „Verkaufsregion“,

[32] Lehner, Wolfgang (2003), S. 76 f.

zum anderen die members „Q1“, „Q2“, „Q3“ und „Q4“ des levels „Quartal“ der dimension „Zeitraum“. Die FROM-Klausel referenziert den Datenwürfel „VerkaufsCube“ an, aus dem die Daten bezogen werden. Die WHERE-Anweisung schränkt schließlich den Ergebnisraum auf den Verkaufsumsatz (Kennzahl „Umsatz“) ein, der im Jahre 2004 erwirtschaftet wurde.

4.3 Die Syntax von MDX

Neben der Verwendung von SELECT-, FROM- und WHERE-Klauseln ist bei der Generierung von Anfragen in MDX noch die spezifische Verwendung verschiedener Klammerausdrücke, sowie von Punkt und Komma zu beachten. Grundsätzlich kommen in MDX zum Einsatz:

- Geschweifte Klammern { }
- Runde Klammern ()
- Eckige Klammern []
- Punkt (.) und Komma (,)

Geschweifte Klammern { } dienen dazu, sets, also eine Ansammlung von tuples der gleichen Dimensionalität, als solche zu kennzeichnen. Geschweifte Klammern dienen somit dazu, eine Menge von Elementen für die Erzeugung von Ergebnisdimensionen zu markieren.

In runden Klammern () werden die tuple eingeschlossen, die für die WHERE-Klausel benötigt werden.[33]

In eckigen Klammern [] werden typischerweise sowohl die Namen von dimensions, levels und deren members, als auch die Namen von cubes sowie die jeweiligen measures eingeschlossen. Grundsätzlich ist der Einsatz dieser Art von Klammern optional. Obligatorisch ist er allerdings dann, wenn darin z.B. numerische Werte, Leerzeichen oder sonstige Sonderzeichen mit eingeschlossen werden. Soll beispielsweise das Jahr „2004“ als member des levels „Jahr“ der dimension „Zeit“ in eine MDX-Anfrage eingeflochten werden, so muss es zwingend in eckige Klammern [2004] eingeschlossen werden. Andernfalls interpretiert der MDX Parser

[33] Vgl. Lehner, Wolfgang (2003), S. 77

die Jahresangabe nicht als member, sondern als gewöhnliche Zahl (integer).[34]

Punkte (.) dienen als Separatoren. Separatoren benötigt man immer dann, wenn man innerhalb einer MDX-Anfrage – ausgehend vom Namen einer dimension – eine bestimmte Hierarchieebene dieser dimension und dort wiederum ein bestimmtes member-Element identifizieren möchte.

Beispiel:
[Verkaufsregion].[Kontinent].[USA]

Kommas (,) schließlich werden ebenfalls als Separatoren benutzt und zwar dann, wenn es gilt, bei einer Aufreihung mehrer tuples (z.B. innerhalb eines sets) einzelne tuple voneinander zu trennen.[35]

4.4 Einführung in die Möglichkeiten von MDX anhand ausgesuchter Beispiel-Anfragen

In diesem Kapitel wird eine Einführung in den Funktionsumfang von MDX gegeben, um dem Leser eine Vorstellung von dem Potential und den Möglichkeiten von MDX zu geben. Es wird insbesondere auf die Navigationsfunktionen von MDX, die Möglichkeit der Mengenschachtelung in MDX sowie einige weitere nützliche MDX-Funktionalitäten eingegangen. Es sei jedoch zugleich darauf hingewiesen, dass hier bei weitem nicht auf alle Möglichkeiten, die sich mit MDX bieten, eingegangen werden kann, da eine umfassende Darstellung der Funktionsvielfalt von MDX den Rahmen dieser Arbeit sprengen würde.

4.4.1. Navigationsfunktionen in MDX

Die Navigationsfunktionen, die MDX bereitstellt, dienen dazu, innerhalb der Dimensionshierarchien navigieren zu können. Dabei erlauben sie es insbesondere, effizient auf member-Elemente einer dimension zugreifen zu können, ohne dass dazu umfangreiche und lange Anfrage-Statements generiert werden müssen. Mit der in Gleichung 6 aufgeführten MDX-

[34] Vgl. Whitehorn, Mark et al. (2004), S. 53 ff.

[35] Vgl. Whitehorn, Mark et al. (2004), S. 54 f.

Anfrage werden beispielhaft die Verkaufsumsätze der Produktgruppe „Automobil“, sortiert nach den einzelnen Modellen, angefragt.

```
SELECT {[Measures].[Umsatz]} ON COLUMNS
{[Produkt].[Produktgruppe].[Automobil].[Pkw1],
  [Produkt].[Produktgruppe].[Automobil].[Pkw2],
  [Produkt].[Produktgruppe].[Automobil].[Pkw3],
  [Produkt].[Produktgruppe].[Automobil].[Transporter1],
  [Produkt].[Produktgruppe].[Automobil].[Transporter2]} ON ROWS
FROM [VerkaufsCube]
WHERE ...
```

Gleichung 6: MDX-Statement [3], eigene Darstellung

Sämtliche Modelle werden hier explizit angefragt – dies ist zwar syntaktisch korrekt und liefert auch das gewünschte Ergebnis, jedoch wird die Anfrage dadurch unnötig lang. MDX bietet die Möglichkeit, derartige Aufzählungen zu verkürzen, ohne dass das Ergebnis in irgendeiner Weise verändert wird.

```
SELECT {[Measures].[Umsatz]} ON COLUMNS
{[Produkt].[Produktgruppe].[Automobil].Children} ON ROWS
FROM [VerkaufsCube]
WHERE ...
```

Gleichung 7: MDX-Statement [4], eigene Darstellung

Gleichung 7 illustriert, wie die in Gleichung 6 dargestellte Anfrage über die MDX-Navigationsfunktion „Children“ deutlich verkürzt werden kann. Mit der Children-Funktion lassen sich stets die members eines levels unterhalb des angegebenen levels identifizieren. Sollen dagegen einfach die members eines levels der gleichen Stufe identifiziert werden, so bietet sich die Navigationsfunktion „Members“ an. Da sich die beiden Funktionen „Children“ und „Members“ ähneln, sei in Gleichung 8 der Unterschied zwischen diesen beiden Navigationsfunktionen noch einmal illustriert.

Navigationsfunktion „Children“:

[Zeit].[2004].Children = Q1,Q2,Q3,Q4

Navigationsfunktion „Members“:

[Zeit].[2004].[Quartal].Members = Q1,Q2,Q3,Q4

Gleichung 8: MDX-Statement [5], eigene Darstellung

Die Children-Funktion kommt schließlich noch in zwei Varianten vor, nämlich den Navigationsfunktionen „FirstChild“ und „LastChild“. Mit „FirstChild“ werden im Gegensatz zu „Children“ nicht alle members des unteren levels, sondern nur das erste member ausgewählt. Mit „LastChild“ entsprechend nur das letzte member (siehe auch Gleichung 9).

Navigationsfunktion „FirstChild“:

[Zeit].[2004].FirstChild = Q1

Navigationsfunktion „LastChild“:

[Zeit].[2004].LastChild = Q4

Gleichung 9: MDX-Statement [6], eigene Darstellung

Zur Children-Navigationsfunktion existiert weiterhin eine Invers-Funktion namens „Parents“. Dabei ist zu beachten, dass „Parent“ stets nur ein member als Ergebnis liefert, während „Children“ meist mehrere members zum Ergebnis hat. So hat die in Gleichung 10 dargestellte Anfrage das

Navigationsfunktion „Parent“:

[Zeit].[2004].[Q1].Parent = 2004

Gleichung 10: MDX-Statement [7], eigene Darstellung

Jahr 2004 als Ergebnis der Parent-Funktion zum Ziel.

Eine Auflistung aller wesentlichen Navigationsfunktionen in MDX sei schließlich in Abbildung 5 gegeben:

Wichtige Navigationsfunktionen in MDX:	
Member	hat set zum Ergebnis
Children	hat set zum Ergebnis
FirstChild	hat member zum Ergebnis
LastChild	hat member zum Ergebnis
Parent	hat member zum Ergebnis
Descendants	hat set zum Ergebnis
Ancestor	hat member zum Ergebnis
Siblings	hat set zum Ergebnis
FistSibling	hat member zum Ergebnis
LastSibling	hat member zum Ergebnis
Cousin	hat member zum Ergebnis

Abbildung 5: Wichtige MDX-Navigationsfunktionen, in Anlehnung an Whitehorn, Mark, et al. (2004), S. 97

4.4.2. Mengenschachtelungen in MDX

Schachtelungen von Mengen sind ein wichtiges Thema in MDX, da es erst über verschachtelte Konstrukte möglich wird, einen multidimensionalen Ergebnisraum auf eine zweidimensionale Tabellenstruktur abzubilden. Die zweidimensionale, tabellenartige Darstellung von Anfrageergebnissen ist wiederum notwendig, da nur so eine adäquate, druckfähige Ergebnispräsentation für den Anwender gegeben ist. Die MDX-Funktionalität nun, die Schachtelungen von Mengen ermöglicht, lautet „CROSSJOIN()“.[36]

```
SELECT {[Zeit].[2004].Children} ON COLUMNS
CROSSJOIN ({[Verkaufsregion].[Kontinent].[Europa]},
{[Produkt].[Produktgruppe].[Automobil].Children}) ON ROWS
FROM [VerkaufsCube]
WHERE ([Measures].[Umsatz])
```

		Q1	Q2	Q3	Q4
Europa	Pkw1				
	Pkw2				
	Pkw3				
	Transporter1				
	Transporter2				

Gleichung 11: MDX-Statement [8], eigene Darstellung

Wie in Gleichung 11 zu sehen, ermöglicht der CROSSJOIN-Operator die Kombination einzelner Dimensionen. In diesem Beispiel werden die Verkaufsumsätze nach Kontinent (hier: Europa) sowie nach den einzelnen Modellen aufgeschlüsselt, wobei die Ergebnisse quartalsmäßig für das Jahr 2004 ausgegeben werden.

Es sei noch angemerkt, dass durchaus mehrere CROSSJOIN-Operationen innerhalb einer Anfrage möglich sind, wodurch sich auch komplexere Schachtelungen erzeugen lassen.[37]

4.4.3. Weitere Funktionalitäten in MDX

Mitunter kann es vorkommen, dass Anfragen an den Datenwürfel auch

---

[36] Vgl. Lehner, Wolfgang (2003), S. 78

[37] Vgl. Whitehorn, Marc, et al. (2004), S. 225

leere Zellen mit in den Ergebnisraum aufnehmen und darunter wiederum die Güte und Übersichtlichkeit der Ergebnisdarstellung leidet. Um leere Zellen im Ergebnisraum zu verhindern, stellt MDX eine einfache Anweisung namens „NON EMPTY“ zur Verfügung.

So verhindert das „NON EMPTY“ vor den beiden Teilen (ON COLUMNS, On ROWS) der SELECT-Klausel in Gleichung 12, dass leere Ergebniszellen entstehen und damit eine unschöne Darstellung der Ergebnisse.[38]

```
SELECT NON EMPTY {[Verkaufsregion].[Kontinent].[USA],
[Verkaufsregion].[Kontinent]. [Kanada]} ON COLUMNS, NON EMPTY
{[Zeitraum].[Quartal].[Q1.], [Zeitraum].[Quartal].[Q2], [Zeitraum].[Quartal].[Q3],
[Zeitraum].[Quartal].[Q4]} ON ROWS
FROM [VerkaufsCube]
```

Gleichung 12: MDX-Statement [9], eigene Darstellung

Schließlich beinhaltet der Sprachschatz von MDX unter anderem auch eine Reihe von **Filter- und Sortierfunktionen**, die ebenfalls die Darstellungsqualität der Anfrageergebnisse deutlich verbessern können. So liefert etwa die Sortier-Funktion „TOPCOUNT()“ die Möglichkeit, die Ergebnisse der Größe nach zu sortieren, z.B. um die Produkte mit dem höchsten Verkaufsumsatz schneller und besser identifizieren zu können

```
SELECT {TOPCOUNT([Produkt].[Produktgruppe].[Automobil].Children, 3,
[Measures].[Umsatz])}
FROM [VerkaufsCube]
```

Gleichung 13: MDX-Statement [10], eigene Darstellung

Das in Gleichung 13 dargestellte MDX-Statement fragt die Top 3 der Modelle aus der Produktgruppe „Automobil“ an, wobei der Verkaufsumsatz hier das Kriterium für die TOPCOUNT-Anweisung ist. Im Gegenzug dazu lassen sich mit der Funktion „BOTTOMCOUNT()“ auf die gleiche Weise die umsatzschwächsten Produkte herausfiltern.[39] Weitere wichtige Funktionen in diesem Kontext sind die Funktionen „TOPPERCENT()“, „TOPSUM()“ sowie „FILTER()“.[40]

---

[38] Vgl. Whitehorn, Mark, et al. (2004), S. 226 ff.

[39] Vgl. Whitehorn, Mark, et al. (2004), S. 230 ff.

[40] Vgl. Lehner, Wolfgang (2003), S. 80

5 Zusammenfassung

Die multidimensionale Anfragesprache MDX stellt zweifelsohne eine mächtiges Werkzeug zur Generierung auch komplexer OLAP-Anfragen an eine Data-Warehouse-Datenbasis dar. Der Einsatz von MDX verspricht dabei flexiblere und komplexere Anfragen als der reine Einsatz von systemtechnisch eingebundenen Navigationsoperatoren.

Da MDX besonders gut mit relationalen Datenbanksystemen zusammenarbeitet, empfiehlt sich für ROLAP damit der Einsatz von MDX.

Für MDX spricht außerdem, dass es in Aufbau und Syntax der relationalen Anfragesprache SQL ähnelt. Kenntnisse in SQL sind wiederum bei IT-Experten weit verbreitet, so dass der Umgang mit MDX für die IT-Mitarbeiter eines Unternehmens relativ schnell zu erlernen ist, was natürlich die Kosten für Schulungen reduziert und zugleich den Einsatz von MDX für OLAP generell erleichtert.

Da die Funktionalitäten von MDX überzeugen können und zudem mit Microsoft ein mächtiges Unternehmen hinter MDX steht, unterstützen bereits zahlreiche Hersteller die für MDX relevante Schnittstelle OLE DB for OLAP. Vor diesem Hintergrund kann es als wahrscheinlich angesehen werden, dass MDX sich zu einem Quasi-Standard unter den multidimensionalen Anfragesprachen entwickelt.

Literaturverzeichnis

Bauer, Andreas/ Günzel, Holger (2001)	Data Warehouse Systeme, 1. Auflage, dpunkt.verlag GmbH, Heidelberg
Böhm, Klemens (2004)	Unterlagen zur Vorlesung Data Warehousing und Data Mining (WS2004/05), Universität Karlsruhe, URL: http://www.ipd.uni-karlsruhe.de/~ipd/institut/dwm/dwm-kap9-4auf1.pdf
Brosius, Gerhard (2001)	Data Warehouse und OLAP mit Microsoft, 1. Auflage, Galileo Press GmbH, Bonn
Hernandez, Michael J./ John L. Viescas	Go To SQL, Addison-Wesley Verlag, Imprint der Pearson Education Deutschlanf GmbH, München
Kemper, Alfons/ Eickler André (2004)	Datenbanksysteme, 5. Auflage, Oldenbourg Wissenschaftsverlag GmbH, München
Kruczynski, Klaus (2004)	Seminarunterlagen „Data Warehousing und Data Mining“ (WS 2004/05), Fachhochschule Liechtenstein, Seminar II Masterstudiengang Wirtschaftsinformatik
Lehner, Wolfgang (2003)	Datenbanktechnologie für Data-Warehouse-Systeme, 1. Auflage, dpunkt.verlag GmbH, Heidelberg
Leser, Ulf (2004)	Data Warehousing, Vorlesung im Bereich Wissensmanagement der Bioinformatik an der Humboldt-Universität Berlin, URL: http://www.informatik.hu-berlin.de/wbi/teaching/sose04/dwh/06_zugriff_mddm.pdf
Microsoft Internetpräsenz [1]	URL: http://msdn.microsoft.com/library/en-us/olapdmad/agmdxbasics_3md4.asp
Microsoft Internetpräsenz [2]	URL: http://msdn.microsoft.com/library/en-us/olapdmad/agmdxbasics_1bzs.asp

Muksch, Harry/ Behme, Wolfgang (2000)	Das Data Warehouse-Konzept, Architektur – Datenmodelle – Anwendungen, 4. Auflage, Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH, Wiesbaden
Rondot, René (2003)	Anfragesprachen für On-Line Analytical Processing (OLAP), Seminararbeit an der TU Kaiserslautern, URL: http://wwwdvs.informatik.uni-kl.de/courses/seminar/SS2003/folien3.pdf
Totok, Andreas (1997):	Data Warehouse und OLAP als Basis für betriebliche Informationssysteme, Berichte des Instituts für Wirtschaftswissenschaften der Technischen Universität Braunschweig
Vossen, Gottfried (2000)	Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 4. Auflage, Oldenbourg Wissenschaftsverlag GmbH, München
Whitehorn, Mark/ Zare, Robert/ Pasumansky, Mosha (2004)	Fast Track to MDX, 4. Auflage, Springer Verlag, London, Heidelberg, Berlin, Heidelberg