

**Universität
Konstanz**

Fakultät für Mathematik und Informatik

Fachgruppe Informationswissenschaft

Diplomarbeit
zum Thema

**Datenmodellierung für das Data Warehouse -
Vergleich und Bewertung
konzeptioneller und logischer Methoden**

vorgelegt von

Ulrike Schlenker
Hauptstr. 71
78250 Tengen

1. Gutachter: Prof. Dr. Harald Reiterer
2. Gutachter: Prof. Dr. Marc Scholl

Konstanz, im Juni 1998

Abstract

Gegenstand der Arbeit ist die Beschreibung, der Vergleich und die Bewertung von Methoden zur konzeptionellen und logischen Datenmodellierung für Data Warehouse. Dabei werden zunächst die Grundlagen und Eigenschaften eines Data Warehouse beschrieben und verwendbare Datenbanktypen vorgestellt. Es werden die für Data Warehouse gültigen Aspekte der Datenmodellierung aufgezeigt und ein Kriterienkatalog erstellt, mit dem die unterschiedlichen Methoden verglichen und bewertet werden können. Als Hauptteil der Arbeit werden verschiedene konzeptionelle und logische Modellierungsmethoden vorgestellt, mit einem einheitlichen Beispiel modelliert und anhand des Kriterienkatalogs verglichen.

Schlüsselwörter: Data Warehouse, OLAP, Datenmodellierung

Subject of this work is the description and comparison of conceptual and logical data modelling methods for data warehouse. First the fundamentals and features of a data warehouse are described. This is followed by a presentation of usable database types. After that, the aspects of data modelling, which are relevant for data warehouse, are explained. A catalogue with evaluation criteria is introduced to compare and evaluate the various conceptual and logical modelling methods. Main part of this work is the description of different conceptual and logical data modelling methods for data warehouse, which will be modelled with a unique example and evaluated with the criteria catalogue.

Keywords: data warehouse, OLAP, data modelling

1 GRUNDLAGEN ZUM THEMA DATA WAREHOUSE	6
1.1 Anforderungen an ein Data Warehouse	8
1.2 Kennzeichen eines Data Warehouse	8
1.3 Objekte, Beziehungen, Aggregationen und Operationen im Data Warehouse	9
1.3.1 Objekte	9
1.3.2 Beziehungen	9
1.3.3 Aggregationen	10
1.3.4 Operationen	10
2 DATENBANKEN FÜR DAS DATA WAREHOUSE	12
2.1 Differenzierung Data Warehouse - OLAP	12
2.2 Relationaler Ansatz	13
2.2.1 Relationale Datenbanken	13
2.2.2 Relationale OLAP-Server	14
2.3 Multidimensionaler Ansatz	14
2.3.1 Multidimensionale Datenbanken bzw. MOLAP-Server	14
2.4 Objektorientierter Ansatz	14
2.4.1 Objektorientierte Datenbanken	14
2.4.2 Objektorientierte OLAP-Server	15
2.5 Kommerzielle Produkte	15
3 DATENMODELLIERUNG FÜR DATA WAREHOUSE	17
3.1 Architektur eines Datenmodells	17
3.2 Vorgehensweise zur Erstellung eines Datenmodells	18
3.2.1 Klärung des Analysebedarfs	18
3.2.2 Aussagensammlung	18
3.2.3 Bereinigung	19
3.2.4 Fachlexikon	19
3.2.5 Konzeptionelles Datenmodell	19
3.2.6 Vom konzeptionellen zum relationalen logischen Modell	20
3.2.7 Vom konzeptionellen zum multidimensionalen logischen Modell	20
3.2.8 Vom konzeptionellen zum objektorientierten logischen Modell	21
3.2.9 Logisches Datenarchitektur	21
3.2.10 Physischer Aspekte	22
3.2.10.1 Steigerung der Abfrageperformance	22
3.2.10.2 Aufteilung großer Datenbestände	22
3.3 Schema-Integration	23
3.4 Metadatenmodellierung	23
4 KRITERIENKATALOG	26
4.1 Bewertungskriterien	26

4.2 Bewertungsmetrik	29
5 KONZEPTIONELLEN METHODEN UND IHRER ELEMENTE	30
5.1 Methode ADAPT	30
5.1.1 Kernelemente	30
5.1.2 Dimensionstypen	31
5.1.3 Zusatzelemente	31
5.1.4 Teilaggregate	31
5.2 Methode DF	32
5.3 Methode ER	33
5.4 Methode UML	34
6 MODELLIERUNGBEISPIEL	37
6.1 Vorstellung des Beispiels	37
6.2 Klärung des Analysebedarfs	38
6.3 Konzeptionelle Modellierung	39
6.3.1 ADAPT	40
6.3.1.1 Von ADAPT zum relationalen logischen Schema	42
6.3.1.2 Von ADAPT zum multidimensionalen logischen Schema	42
6.3.1.3 Von ADAPT zum objektorientierten logischen Schema	43
6.3.1.4 Subjektive Bewertung von ADAPT	43
6.3.2 DF	46
6.3.2.1 Von DF zum relationalen logischen Schema	47
6.3.2.2 Von DF zum multidimensionalen logischen Schema	47
6.3.2.3 Von DF zum objektorientierten logischen Schema	48
6.3.2.4 Subjektive Bewertung von DF	48
6.3.3 ER	50
6.3.3.1 Von ER zum relationalen logischen Schema	51
6.3.3.2 Von ER zum multidimensionalen logischen Schema	52
6.3.3.3 Von ER zum objektorientierten logischen Schema	52
6.3.3.4 Subjektive Bewertung von ER	53
6.3.4 UML	55
6.3.4.1 Von UML zum relationalen logischen Schema	56
6.3.4.2 Von UML zum multidimensionalen logischen Schema	57
6.3.4.3 Von UML zum objektorientierten logischen Schema	57
6.3.4.4 Subjektive Bewertung der UML	58
6.3.5 Vergleich der konzeptionellen Methoden	60
6.4 Logische Architekturen	61
6.4.1 Star-Schema, Hypercube	61
6.4.1.1 Subjektive Bewertung des Star-Schemas	62
6.4.2 Galaxy-Schema, Multicube	63
6.4.2.1 Subjektive Bewertung des Galaxy-Schemas	63
6.4.3 Snowflake-Schema	64
6.4.3.1 Subjektive Bewertung des Snowflake-Schemas	65
6.4.4 Vergleich der logischen Architekturen	67
6.5 Vom OLTP-Modell zum logischen relationalen DW-Modell	68

6.6 Vom OLTP-Modell zum multidimensionalen DW-Modell	76
6.7 Vom OLTP-Modell zum objektorientierten DW-Modell	78
6.8 Bewertung der Technologien anhand des Kriterienkatalogs	79
6.9 Subjektive Bewertung der Technologien	79
6.10 Vergleich der einzelnen Technologien	81
7 SCHLUßBEMERKUNG	83
8 LITERATUR	84

Abbildungsverzeichnis

Abbildung 1.1: Von OLTP zu OLAP	aus [ChauDay]	7
Abbildung 2.1 Technologiekombinationen für Datenspeicherung und -abfrage		12
Abbildung 2.2 Produktübersicht		15
Abbildung 3.1: Metamodell eines Data Warehouse	aus [Sach98]	25
Abbildung 4.1: Bewertungskriterien nach DIN/ISO 9126		26
Abbildung 5.1: ADAPT-Symbole]		30
Abbildung 5.2: DF-Symbole		32
Abbildung 5.3: ER-Symbole		33
Abbildung 5.4: UML-Symbole		34
Abbildung 5.5 Notation einer multidimensionalen Datenstruktur		35
Abbildung 5.6 Notation von speziellen Klassen		36
Abbildung 6.1: Modellierungsbeispiel Bestellungen als relationales Datenbankschema		38
Abbildung 6.2: Konzeptionelles ADAPT-Modell Umsatz		40
Abbildung 6.3: Konzeptionelles ADAPT-Modell Frachtkosten		41
Abbildung 6.4: DF-Modelle Umsatz und Frachtkosten		46
Abbildung 6.5: ER-Modell Umsatz		50
Abbildung 6.6: ER-Modell Frachtkosten		51
Abbildung 6.7: UML-Modell Umsatz		55
Abbildung 6.8: UML-Modell Frachtkosten		56
Abbildung 6.9: Vergleich der konzeptionellen Methoden		60
Abbildung 6.10: Star-Schema		61
Abbildung 6.11: Galaxy-Schema		63
Abbildung 6.12: Snowflake-Schema		65
Abbildung 6.13: Vergleich der logischen Architekturen		67
Abbildung 6.14: Data Warehouse-Beispiel in MS-Access		74
Abbildung 6.15: Data Warehouse-Beispiel in Oracle7		75
Abbildung 6.16: Data Warehouse-Beispiel in MIK-OLAP		76
Abbildung 6.17: Importierte Dimensionselemente in MIK-OLAP (Dimension Artikel)		77
Abbildung 6.18: Data Warehouse-Beispiel in ObjectStore		78
Abbildung 6.19: Technologievergleich		81

1 Grundlagen zum Thema Data Warehouse

In den letzten Jahren ist der Begriff Data Warehouse zunehmend bekannt geworden, als die Grundlage für die Aufbereitung und Extraktion brachliegender Unternehmensdaten zu strategisch relevanten Informationen. Das Data Warehouse bildet dabei die Datenbasis für sogenannte analytische Informationssysteme, welche Führungskräfte eines Unternehmens bei der strategischen Unternehmensanalyse und -planung unterstützen sollen. Darunter fallen zum Beispiel Führungsinformationssysteme (Executive Information Systems, EIS) oder Entscheidungsunterstützungssysteme (Decision Support Systems, DSS), also Systeme, die dem Benutzer helfen sollen, eine Situation zu analysieren, um Entscheidungen treffen zu können. Dieser Vorgang der computergestützten Analyse wird auch als OLAP (Online Analytical Processing) bezeichnet. Analytische Informationssysteme stehen im Gegensatz zu sogenannten operativen Informationssystemen, mit denen in einem Unternehmen die täglichen Geschäftsvorfälle, wie Aufträge, Buchungen usw., in Form von Transaktionen erledigt werden. Diese Vorgänge werden wiederum als OLTP (Online Transaction Processing) bezeichnet. Operativen Informationssystemen liegt meist eine normalisierte, relationale Datenbank zugrunde, wodurch eine effiziente, für das Tagesgeschäft optimierte Datenhaltung ermöglicht wird. Ein solches Transaktionssystem ist im allgemeinen für die Beantwortung von Fragen wie

„Wann ist die Lieferung für Kunde A herausgegangen?“ oder

„Wieviel Rollen Klebeband sind noch im Lager?“

geeignet.

Es setzt sich jedoch zunehmend durch, die operativen Daten, die normalerweise nach Abschluß des jeweiligen Vorgangs archiviert werden, als Ressource in Form von strategisch wichtigen Informationen zu nutzen. Nach entsprechender Aufbereitung zu einem Data Warehouse lassen sich damit umfangreiche Analysen tätigen, deren Ergebnisse insbesondere für das Finanzwesen, für Marketing u.ä. von großer Bedeutung sind. Für diese Bereiche sind beispielsweise folgende Fragen relevant:

„Wie verkaufen sich bestimmte Produkte pro Kundengruppe zur Weihnachtszeit?“ oder

„Gibt es einen Zusammenhang zwischen“ oder

„Wie sehen unsere Umsatzzahlen im nächsten Jahr aus, wenn sich dieser Trend fortsetzt?“

Man sieht bereits an den Beispielen, daß sich dieser strategische Informationsbedarf aus einer größeren Anzahl von Komponenten zusammensetzt, als der des operativen Geschehens, welcher eher funktionalen Charakter hat. Strategische Informationen haben einen sogenannten

multidimensionalen Charakter, das heißt, es müssen Informationen aus mehreren Dimensionen eines Unternehmens zusammengefügt und verdichtet werden, um eine Schlußfolgerung ziehen zu können. Solche Analysen sind mit Transaktionssystemen (OLTP) nur sehr schwer und zeitintensiv zu realisieren, da diesen Systemen ein Datenmodell zugrunde liegt, das diese Dimensionalität nicht berücksichtigt. Aus diesem Grund versucht man, die historischen Daten der operativen Informationssysteme in ein Datenmodell mit dimensionalem Charakter zu überführen, dem Data Warehouse.

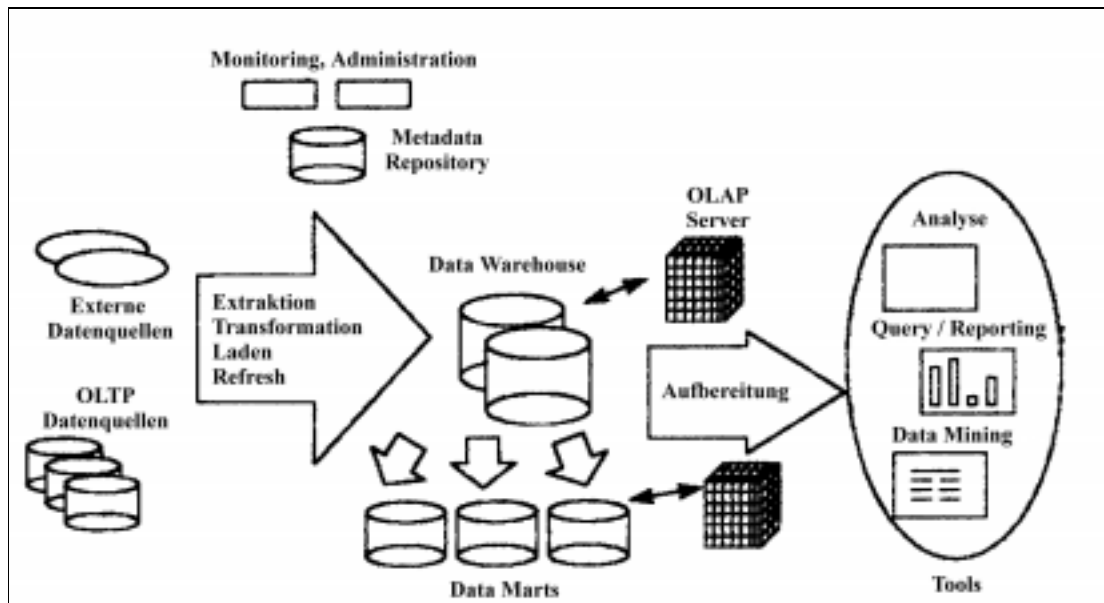


Abbildung 1.1: Von OLTP zu OLAP

aus [ChauDay]

Um ein Data Warehouse aufzubauen, werden strategisch relevante Informationen aus den OLTP-Systemen und externen Quellen (z.B. Marktforschungsdaten) gesammelt, bereinigt und anschließend in ein Data Warehouse-Datenmodell transformiert. Diese Aufbereitung der Daten und das anschließende Laden der Daten in das Data Warehouse geschieht in bestimmten Zeitabständen (täglich, wöchentlich, monatlich), so daß dessen Datenbasis immer in einem zwar nicht topaktuellen, aber konsistenten Zustand ist. Die Data Warehouse-Datenbank läßt sich in sogenannte Data Marts aufteilen, die quasi Mini-Data Warehouses für einzelne Geschäftsbereiche darstellen. Das Data Warehouse bzw. die Data Marts speisen die OLAP-Systeme, welche die Daten für Analyse, Reporting und Data Mining optimal aufbereiten und hierfür erweiterte Analysefunktionalität bereitstellen. Man ist bemüht, diesen ganzen Aufbereitungs- und Verwaltungsprozeß mittels eines Metainformationssystems zu steuern, um eine Automatisierung zu erreichen.

Wie man sieht, nimmt das Data Warehouse für die analytische Informationsverarbeitung in etwa dieselbe Stellung ein, wie ein normalisiertes, relationales Datenbanksystem für die operative Datenverarbeitung, natürlich mit anderen Anforderungen. Aus diesem Grund sollte auch der Datenmodellierung für ein Data Warehouse ebensoviel Aufmerksamkeit zukommen, wie der Entwicklung eines unternehmensweiten Datenmodells für ein operatives Informationssystem.

1.1 Anforderungen an ein Data Warehouse

Aus den aufgezeigten Unterschieden zu herkömmlichen Transaktionssystemen ergeben sich somit als wichtigste Anforderungen an ein Data Warehouse [Codd et.al.93], [Buy95]:

Multidimensionale Datenhaltung: Die Sicht einer Führungskraft auf ein Unternehmen ist multidimensional. Um eine einfache und intuitive Analyse der Unternehmensdaten zu ermöglichen, sollten die Daten ebenfalls in Dimensionen angeordnet sein.

Konstant schnelle Antwortzeiten: Alle Analysen sollten mit einer annähernd gleichen, möglichst hohen Geschwindigkeit vom System durchgeführt werden, um dem Benutzer optimale Bedingungen zu bieten.

Verwaltung großer Datenmengen: Da in einem Data Warehouse historische Daten, auch aus länger zurückliegenden Zeitabschnitten, in einer speicherintensiven Struktur gespeichert werden, muß das DBMS diese Datenmenge verwalten können.

Flexible Abfrage großer Datenmengen: Da bei strategischen Analysen oftmals größere Zeiträume über mehrere Dimensionen hinweg flexibel abgefragt werden, muß das Datenmodell diesbezüglich optimiert sein.

1.2 Kennzeichen eines Data Warehouse

W.H. Inmon, einer der Begründer der Data Warehouse-Technologie, bezeichnet das Data Warehouse als die zentrale Architektur für Informationssysteme der 90er Jahre. Es ist die ideale Plattform für die Analyse integrierter, historischer Daten. Ein Data Warehouse ist dadurch gekennzeichnet, daß die Daten themenorientiert, integriert, zeitbezogen und nichtflüchtig gespeichert werden [Inm95].

Themenorientiert heißt, die Daten werden zu sogenannten Dimensionen zusammengefaßt und geordnet. Dimensionen eines Unternehmens sind beispielsweise Kunden, Regionen oder Produkte, also Komponenten des strategischen Informationsbedarfs.

Integriert bedeutet, daß alle eine Dimension betreffenden Daten aus den OLTP-Systemen bereinigt und so zusammengefaßt werden, daß sich keine Überschneidungen und Inkonsistenzen mehr ergeben.

Zeitbezogen meint zum einem, daß die historische Daten der letzten 5-10 Jahre gespeichert werden (langer Zeithorizont). Zum anderen ist damit gemeint, daß bei der Analyse der Data Warehouse-Daten immer ein Zeitraum angegeben werden muß, über den ausgewertet wird (Zeit ist Komponente des Schlüssels). Außerdem bedeutet zeitbezogen noch, daß die Daten im Data Warehouse eine Reihe zeitlicher Schnappschüsse darstellen.

Nichtflüchtig bedeutet somit, daß einmal korrekt geladene Daten nie geändert, sondern nur ergänzt werden. Nur so lassen sich Veränderungen zeitraumbezogen erfassen.

1.3 Objekte, Beziehungen, Aggregationen und Operationen im Data Warehouse

1.3.1 Objekte

Eine multidimensionale Datenstruktur im Data Warehouse besteht immer aus drei grundlegenden Objekten, den **Fakten**, den **Dimensionen** und den **Regeln** [GaGlu97].

Fakten sind Variablen, die eine Unternehmenskennzahl repräsentieren (z.B. Umsatz, Menge, Kosten). Sie sind das Ergebnis unternehmerischer Tätigkeit. Fakten können in verschiedene **Versionen** aufgeteilt werden (z.B. Ist-Umsatz, Soll-Umsatz).

Dimensionen spiegeln die unternehmerische Sichtweise wieder, nach der die Fakten aufgeschlüsselt werden (z.B. Kunden, Produkte, Regionen, Zeit). Diese Dimensionen lassen sich zu **Hierarchien** verdichten (z.B. Kunde - Kundengruppe). Innerhalb einer Dimension unterscheidet man zwischen **Dimensionselementen**, nach denen verdichtet wird (z.B. Produkt), und **Dimensionsattributen**, die lediglich beschreibenden Charakter haben (z.B. Abmessungen).

Regeln sind Vorschriften, die festlegen, wie eine Kennzahl berechnet wird (z.B. Gesamtpreis = Menge * Einzelpreis).

1.3.2 Beziehungen

Es gibt zwei grundsätzliche Beziehungstypen, die kennzeichnen, wie Fakten und Dimensionen zueinander stehen. Das eine ist das sogenannte **Star-Schema**. Synonym kann der Begriff multidimensionaler Datenwürfel oder **Hypercube** verwendet werden. Hier sind alle Kennzahlen zusammengefaßt und alle Dimensionen darum herum angeordnet.

Üblicherweise ergibt eine Kennzahl (Fakt) aber nicht immer über alle Dimensionen einen Sinn. Aus diesem Grund wird das Star-Schema zum sogenannten **Galaxy-Schema** erweitert, in dem die einzelnen Kennzahlen nur den sinnvollen Dimensionen zugeordnet werden. Synonym hierzu ist der Begriff **Multicube**, der mehrere multidimensionale Datenwürfel repräsentiert.

Diese beiden Beziehungstypen werden später als logische Architekturmodelle noch ausführlicher beschrieben.

1.3.3 Aggregationen

Sinn und Zweck von OLAP-Anwendungen ist es, die Masse der Daten zunächst auf höchster Hierarchiestufe zu einer aussagekräftigen Kennzahl zu verdichten, die dann zum Zweck der Planung und Entscheidungsfindung wieder aufgeschlüsselt werden kann, um positive oder negative Ursachen zu erkennen. Dies geschieht meist über mehrere Hierarchieebenen und auch -pfade. Dieses Aufsplitten bzw. Zusammenfassen einer Kennzahl wird auch als Drill down bzw. Roll up bezeichnet.

Um eine Kennzahl sinnvoll verdichten zu können, sollte sie sich allen beteiligten Dimensionen gegenüber **additiv** verhalten. Dadurch kann sie auf allen Dimensionshierarchiestufen mittels eines Summen-Operators zusammengefaßt werden. Ist dies nicht über alle Dimensionen der Fall, wird die Kennzahl als **semi-additiv** bezeichnet. Die Lösung ist hier ein Galaxy-Schema, welches die Kennzahl den entsprechenden Dimensionen zuordnet. Läßt sie sich über keine Dimension aufsummieren, ist die Kennzahl **nicht-additiv**. In diesem Fall läßt sich eine Aggregation teilweise über Operatoren wie Durchschnitt, Minimum oder Maximum erreichen. Manchmal liegt der Grund für die Nicht-Additivität einer Kennzahl auch in einer zu geringen Granularität einer Dimension. In diesem Fall würde eine Aggregation zu inkonsistenten Daten führen [Kimb96] [GoMaRi98].

1.3.4 Operationen

Die für das analytische Vorgehen wichtigen Operationen sind **Navigieren**, **Auswählen** und **Anordnen**. Diese Operationen werden zur Navigation, sowie Datenauswahl und -anordnung innerhalb von Datenwürfeln oder über Datenwürfel hinweg verwendet.

Die wichtigsten Navigationsoperationen sind **Drill down** bzw. **Roll up**, **Drill across** und **Drill around**:

Drill down entspricht dabei dem Sprung auf eine tiefere Hierarchiestufe (z.B. von Kundengruppe nach Einzelkunde), wobei die Aggregation aufgelöst wird.

Roll up bewirkt genau das Gegenteil. Es wird eine Hierarchiestufe höher gesprungen, die Daten somit verdichtet (z.B. von Monat nach Jahr).

Drill across ist eine Navigationstechnik, mit der eine Kennzahl über mehrere Würfel, die eine Wertkette bilden, verfolgt werden kann. Dabei ist zu beachten, daß die Dimensionen der einzelnen Würfel die gleiche Granularität aufweisen müssen.

Bei **Drill around** sind die einzelnen Würfel nicht wie oben nacheinander, sondern kreisförmig umeinander angeordnet. Das heißt, es können so alle an einer Dimension beteiligten Kennzahlen ausgewertet werden (Kennzahlen aus verschiedenen Fakttabellen).

Drill across und Drill around kennzeichnen dabei auch die Beziehungen, die zwischen einzelnen Datenwürfeln möglich sind. Beim Drill across werden gleiche Kennzahlen miteinander verknüpft, beim Drill around werden gleiche Dimensionen miteinander verbunden [Kimb3/96].

Operationen zum Auswählen von Daten sind **Slice** und **Dice**:

Slice entspricht einem Schnitt durch einen Datenwürfel, der dazu dient, eine bestimmte Datenauswahl innerhalb einer Dimension zu treffen (z.B. alle Kunden aus Hamburg).

Dice verkleinert den Würfel insgesamt über alle Dimensionen und entspricht damit einem Würfelausschnitt.

Zum Anordnen von Daten verwendet man die Operationen **Rotation** und **Ranging** [MuBe96]:

Bei der **Rotation** wird die Sichtweise auf die ausgewählten Dimensionen eingestellt (z.B. Kunden pro Produkt oder Produkte pro Kunde). Dies würde bei einer tabellarischen Darstellung die Vertauschung von Zeilen und Spalten repräsentieren.

Ranging beschreibt die Anordnung der Elemente innerhalb einer ausgewählten Dimension, also der Sortierung.

Rotation und Ranging wird auch als **Pivoting** bezeichnet.

2 Datenbanken für das Data Warehouse

Derzeit werden für Data Warehouse und OLAP, je nach Hersteller, verschiedene Datenbank-Technologien eingesetzt. Da jede Technologie Vorzüge, aber auch Nachteile hat und das Thema Data Warehouse, jedenfalls im Vergleich zu Transaktionsdatenbanken, im Prinzip noch in den Kinderschuhen steckt, hat sich noch keine dieser Technologien so herausragend bewährt, um als Standard zu gelten.

2.1 Differenzierung Data Warehouse - OLAP

Bezüglich Data Warehouse und OLAP herrscht in der Literatur eine teilweise synonyme, teilweise differenzierte Verwendung der Begriffe. Um die für Data Warehouse und OLAP verwendeten Datenbanken einordnen zu können, lassen sich die Begriffe folgendermaßen differenzieren. Das Data Warehouse, wo Daten multidimensional als Star- oder Galaxy-Schema **abgespeichert** werden, kann ein relationales DBMS (RDBMS), ein multidimensionales DBMS (MDBMS) oder ein objektorientiertes DBMS (OODBMS) sein. Um die Daten **abzufragen**, werden Teile dieser Data Warehouse-Daten in einen sogenannten OLAP-Server geladen, der ebenfalls eine Datenbank enthält, zudem aber noch erweiterte Analysefunktionen bereitstellt (siehe 1.3.4 Operationen). Dort können diese Datenausschnitte ebenfalls relational (ROLAP), multidimensional (MOLAP) oder theoretisch auch objektorientiert (OOLAP) abgespeichert werden. Bezüglich der verwendeten Technologiekombinationen für Datenspeicherung und -abfrage haben sich bestimmte Varianten durchgesetzt, die Vorteile in der Erfüllung der unter 1.1 gestellten Anforderungen bieten [Pen2].

	Multidimensionaler Datenspeicher		
Multidimensionale Abfrageausführung	RDBMS	MDBMS	OODBMS
ROLAP	häufig	nicht sinnvoll	nicht sinnvoll
MOLAP	häufig	häufig	möglich
OOLAP	möglich	nicht sinnvoll	Zukunft ?

Abbildung 2.1 Technologiekombinationen für Datenspeicherung und -abfrage

Die Angabe **häufig** meint, daß es für diese Kombination eine Reihe von Produkten gibt, die sich mehr oder weniger bewährt haben. Kombinationen, die **nicht sinnvoll** wären, da sie einen Rückschritt bedeuten, sind entsprechend gekennzeichnet. Kombinationen, die zwar theoretisch **möglich** sind, aber für die es keine entsprechenden Produkte gibt (jedenfalls keine mir bekannten) ebenfalls.

2.2 Relationaler Ansatz

2.2.1 Relationale Datenbanken

Als relationale Datenspeicher für Data Warehouse-Daten lassen sich alle bekannten RDBMS verwenden, die auch für OLTP-Systeme verwendet werden. Als Vorteile sind hier anzuführen, daß es sich um ausgereifte, gut erforschte Produkte handelt, die auf einem verständlichen relationalen Modell basieren (Tabellen). Alle bekannten Produkte können mit großen Datenmengen umgehen (verteilte und partionierte Datenhaltung). Relationale Datenbanken besitzen eine standardisierte Abfragesprache (SQL), eine ausgereifte Metadatenverwaltung und sind für Mehrbenutzerbetrieb ausgelegt.

Nachteile ergeben sich jedoch besonders dadurch, daß SQL keine speziellen Konstrukte zur Durchführung multidimensionaler Operationen besitzt. Die Nachbildung dieser Konstrukte mit herkömmlichen SQL-Befehlen ergibt umfangreiche Abfragegebilde, die nur paarweise abgearbeitet werden können (pairwise join) und somit mehrfach durchlaufen werden müssen. Außerdem ist es nur bedingt möglich, das Abfrageergebnis hinsichtlich seiner Struktur nachträglich zu verändern (z.B. Pivoting). Das Ergebnis ist eine langsame Abfragegeschwindigkeit und eine inflexible Abfrageabarbeitung. Durch das abfragespezifische Leistungsverhalten lassen sich deshalb keine konstanten Antwortzeiten realisieren. Ein weiterer Punkt ist, daß die, im Data Warehouse üblichen, hierarchischen Strukturen nur mangelhaft darstellbar sind und so Aggregationen nicht explizit unterstützt werden. (u.a. [MuBe96], LeTesch)

Die Bestrebungen der Hersteller gehen dahin, diesen Produkten eine verbesserte Unterstützung für den Umgang mit Data Warehouse-Daten zu geben.

2.2.2 Relationale OLAP-Server

Relationale OLAP-Server bestehen prinzipiell auch aus einer relationalen Datenbank, die jedoch bezüglich multidimensionaler Speichertechnik, Indexierung und Analysefunktionen optimiert ist, um eine bessere Performance und Auswertung zu erreichen. Mit anderen Indexierungsmethoden lassen sich so mehrere Verknüpfungen in einer Abfrage gleichzeitig durchführen, wobei zugleich auf eine optimale Reihenfolge der Abarbeitung geachtet wird. Ein erweiterter SQL-Befehlssatz ermöglicht verbesserte Analyseoperationen. Spezielle für OLAP ausgelegte Speichertechniken (Caches usw.) helfen die Forderung nach konstanten Antwortzeiten zu erfüllen und steigern die Abfrageperformance.

2.3 Multidimensionaler Ansatz

2.3.1 Multidimensionale Datenbanken bzw. MOLAP-Server

Multidimensionale Datenbanken sind noch nicht allzu lange auf dem Markt, aber dafür speziell auf die Bedürfnisse von multidimensionaler Datenhaltung und -analyse zugeschnitten. Die Daten werden in mehrdimensionalen Arrays gespeichert, die miteinander verknüpft werden können. Diese Struktur entspricht im Prinzip dem logischen Abbild des mehrdimensionalen Datenwürfels, wodurch sich ein intuitives Verständnis ergibt. Multidimensionale Datenbanken haben eine auf multidimensionale Analysen zugeschnittene Abfragesprache, wodurch sich die Abfrageperformance gegenüber relationalen Systemen im allgemeinen deutlich verbessert.

Multidimensionale Datenbanken sind jedoch noch in einem proprietären Stadium, das keine Vereinheitlichung zuläßt. Jedes System ist anders aufgebaut und strukturiert. Es existiert keine standardisierte Abfragesprache, wodurch auch kein einheitlicher Funktionsumfang gegeben ist. Erste Erfahrung mit multidimensionalen Datenbanken haben gezeigt, daß sich keine besonders großen Datenbestände verwalten lassen, weshalb ihr Einsatz momentan auf MOLAP-Server beschränkt ist. (u.a. [MuBe96], [LeTesch])

2.4 Objektorientierter Ansatz

2.4.1 Objektorientierte Datenbanken

Objektorientierte Datenbanksysteme sind ebenfalls noch neueren Datums und im praktischen Einsatz noch lange nicht ausgereift. Trotzdem ist ihre Bedeutung auch für multidimensionale Systeme nicht zu vernachlässigen. Sie bieten dieselben Möglichkeiten bezüglich der Modellierung wie relationale Datenbanken. Darüber hinaus sind die Vorteil von objektorientierten Datenbanken

zum einen, daß es keine Technologiebruch zwischen Anwendungen und Datenbanken gibt und somit ein nahtloser Übergang von flüchtigen zu nichtflüchtigen Daten möglich ist. Da objektorientierte Datenbanken nicht auf die klassischen Datentypen beschränkt sind, sondern auch komplexe und unstrukturierte Datentypen verwalten können, gibt es keine Beschränkungen bezüglich der zu verwendenden Datenarten (ideal z.B. für Multimediadaten). Auch hierarchische Strukturen lassen sich im objektorientierten Modell gut abbilden. Die Abfrageperformance ist der relationaler Datenbanken überlegen.

Objektorientierte Datenbanksysteme bieten jedoch keine explizite Unterstützung multidimensionaler Daten. Es gibt keine spezielle Abfragesprache. Alles muß sozusagen "von Hand" programmiert werden (z.B. C++), was jedoch in Bezug auf die Möglichkeiten durchaus positiv zu bewerten ist. Über ein Architekturkonzept (Framework) ließe sich hier vielleicht eine Standard schaffen [LeTesch].

2.4.2 Objektorientierte OLAP-Server

Derzeit gibt es meines Wissens noch keine objektorientierten Datenbanken mit speziellen OLAP-Fähigkeiten. Diese könnten jedoch in Verbindung mit objektorientierten Analyse-, Reporting- oder Data Mining-Tools eine interessante Alternative sein.

2.5 Kommerzielle Produkte

Die folgende Liste ist sicherlich nicht vollständig und soll nur der Einordnung der jeweiligen Produkte dienen [Pen2].

	relational	multidimensional	objektorientiert
Data Warehouse	IBM DB2, Informix Universal Server, MS-SQL-Server, Oracle 7, Sybase SQL-Server,		Gemstone, ObjectStore,
OLAP-Server	IBM DB2 OLAP, Informix MetaCube, Microsoft OLAP-Server, MicroStrategy DSS Agent, Oracle Express (ROLAP), Seagate Holos (ROLAP),	Applix TM1, Arbor Essbase, MIK-OLAP, Oracle Express,	??

Abbildung 2.2 Produktübersicht

Bestimmte OLAP-Produkte sind doppelt aufgeführt, da bei ihnen als Datenbasis ein relationales oder multidimensionales DBMS dienen kann (Hybrid OLAP). Da alle multidimensionalen Produkte auch OLAP-Funktionalität bereitstellen, sind sie dort eingeordnet.

3 Datenmodellierung für Data Warehouse

Die Auswertung strategischer Informationen im Data Warehouse setzt voraus, daß die auszuwertenden Daten in integrierter und konsistenter Form vorliegen. Um diese Datenqualität zu erreichen, müssen die Anforderungen an das Data Warehouse sehr genau bestimmt werden. Dazu werden Methoden benötigt, mit denen auch der Laie gut zurechtkommt, da die Anforderungen an das Data Warehouse durch ihn, den Benutzer, in Zusammenarbeit mit den Data Warehouse-Entwicklern festgelegt werden. Zudem sollte das Data Warehouse eine multidimensionale Globalsicht auf das Unternehmen bieten. Die Anwendung dieser Methoden zur multidimensionalen Datenorganisation ist die Aufgabe der Datenmodellierung, die bereits auf Geschäftsebene beginnen muß, um die unternehmerische Sichtweise zu erfassen.

3.1 Architektur eines Datenmodells

Ein ideales Datenmodell für eine Data Warehouse baut auf einem unternehmensweiten konzeptionellen Data Warehouse-Schema auf, welches den Anwendungsbereich aus Unternehmenssicht (multidimensional) darstellt. Das konzeptionelle Schema sollte:

- anwendungsübergreifend, d.h., für zukünftige Ziele und Aufgaben im Unternehmen erweiterbar sein.
- systemunabhängig, d.h., unabhängig von den vorhandenen oder zukünftigen Arbeitsmitteln sein.
- sprach- und sachgerecht, d.h., den Aufgabenstellungen und Handlungszielen angemessen sein (fachlich konsistent) [Ort94].

Aus diesem konzeptionellen Schema läßt sich die von ANSI/SPARC 1975 vorgestellte 3-
Schema-Architektur für Datenbanksysteme entwickeln. Diese besteht aus:

- einem logischen Schema, welches die Tabellen-, Objekt- oder Array-Struktur festlegt.
- einem externen Schema, welches die Sicht der Benutzer und Anwendungen auf das logische Schema beschreibt (Views).
- und einem internen Schema, welches die physische Organisation der Tabellen, Objekte oder Arrays auf externen Speichern beschreibt [Balz96].

3.2 Vorgehensweise zur Erstellung eines Datenmodells

Ein Datenmodell repräsentiert einen Ausschnitt aus der Realwelt. Der Realitätsausschnitt eines Data Warehouse ist die multidimensionale Sichtweise von Führungskräften (Anwendern) auf Unternehmensbereiche. Diese multidimensionale Sicht sollte sich im Datenmodell wiederfinden. Um diesen Realitätsausschnitt möglichst ideal auf ein Datenmodell abzubilden, muß versucht werden, alle notwendigen Elemente des Realwelt zu erfassen und zu dokumentieren.

Gelegentlich existieren bei der Entwicklung eines Data Warehouse bereits logische und konzeptionelle Unternehmensdatenmodelle (für OLTP-Systeme), die die Datengrundlage des Data Warehouse bilden. Oft müssen jedoch aus heterogene Datenquellen Informationen über die zur Verfügung stehenden Daten gewonnen werden, aus denen dann die Analysemöglichkeiten herausgearbeitet werden können, die jedoch nicht immer auch dem Analysebedarf entsprechen.

3.2.1 Klärung des Analysebedarfs

Um den Analysebedarf der einzelnen Anwender zu klären, müssen zuerst mittels Interviewtechnik oder Fragebogen Aussagen gesammelt werden, in denen die Anwender beschreiben, welche Informationen sie für ihre Analysen benötigen. Dabei sind folgende Fragen relevant [Rad96]:

- Welche Geschäftsprozesse sollen analysiert werden?
- Mit welchen Kennzahlen (Fakten) wird gearbeitet?
- Welchen Detaillierungsgrad (Granularität) müssen die Daten besitzen (täglich, monatlich)?
- Über welche Dimensionen ist eine Auswertung sinnvoll?
- Wie lassen sich diese Dimensionen verdichten?
- Welche Dimensionsattribute werden benötigt?
- Sind die Attribute stabil oder ändern sie sich mit der Zeit?

Das Ergebnis dieser Fragen ist eine Aussagensammlung, die die Sicht einzelner Anwender auf ihren Analysebereich widerspiegelt.

3.2.2 Aussagensammlung

Aus dieser Aussagensammlung werden nun die für die Datenmodellierung relevanten Fachbegriffe ermittelt. Oft werden diese Fachbegriffe jedoch nicht einheitlich benutzt, sondern jede Abteilung

oder sogar jeder Anwender verwendet für bestimmte Kennzahlen, Dimensionselemente oder -attribute unterschiedliche Begriffe.

3.2.3 Bereinigung

Aus diesem Grund werden die sprachlichen Defekte in den Fachbegriffe zunächst bereinigt. Das heißt, Synonyme müssen erfaßt und kontrolliert, Homonyme müssen beseitigt, Äquipollenzen müssen aufgedeckt, Vagheiten müssen geklärt und falsche Bezeichner ersetzt werden [Ort94].

3.2.4 Fachlexikon

Nachdem die Begriffe bereinigt sind, wird ein einheitliches Lexikon der Fachbegriffe erstellt, das als Grundlage für das Datenmodell verwendet wird und das auch den Anwender als gemeinsame Kommunikationsbasis dienen soll.

3.2.5 Konzeptionelles Datenmodell

Auf Grundlage dieses Fachlexikons kann nun mit der semantischen Modellierung des Realitätsausschnitts begonnen werden und mit einer der nachfolgend vorgestellten Methoden ein konzeptionelles Datenmodell erstellt werden. Das konzeptionelle Datenmodell für ein Data Warehouse sollte zumindest folgende Anforderungen erfüllen [GaGlu97]:

- Die zu analysierenden Kennzahlen (Fakten) müssen ersichtlich sein.
- Die Dimensionen müssen erkennbar sein.
- Es muß unterschieden werden können, ob die Dimension hierarchisch oder nicht-hierarchisch ist.
- Die Hierarchiestufen einer Dimension müssen abgebildet werden können.
- Es müssen multiple Konsolidierungspfade einer Hierarchie darstellbar sein.
- Eine Zeitdimension muß eingeführt werden.
- Die Beziehungen zwischen Dimensionen und Fakten müssen ersichtlich sein.
- Es muß zwischen Dimensionselementen und Dimensionsattributen unterschieden werden können.
- Die Herleitung von Kennzahlen muß ersichtlich sein (Regeln).
- Die Granularität der Daten muß erkennbar sein.

Weitere Forderungen wären:

- Verschiedene Versionen einer Kennzahl sollen erkennbar sein (Soll-Ist).
- Kennzahlen mit gleichen Dimensionen sollen als Kennzahlendimension zusammengefaßt werden können.

Bevor mit der logischen Modellierung begonnen wird, muß die Entscheidung für die zu verwendende Architektur und Technologie getroffen werden, die mit der vorhandenen Software und der zukünftigen Planung abgestimmt werden sollte. Anschließend wird das konzeptionelle Datenmodell in das entsprechende logische Data-Warehouse-Schema umgesetzt.

3.2.6 Vom konzeptionellen zum relationalen logischen Modell

Um vom konzeptionellen Modell zu einer relationalen Tabellenstruktur zu kommen, sollte folgendermaßen vorgegangen werden:

- Kennzahlen mit gleichen Dimensionen werden in einer Relation (Fakt-Tabelle) zusammengefaßt, deren Attribute die Kennzahlen sind.
- Jede Dimension wird als Relation definiert, welche die Dimensionselemente und -attribute aller Hierarchiestufen als Attribute enthält (denormalisiert).
- Jede Dimensionsrelation erhält einen künstlichen Primärschlüssel.
- Der Schlüssel der Fakt-Tabelle setzt sich aus den Primärschlüsseln der zugehörigen Dimensionsrelationen zusammen.
- Jede Dimensionsrelation unterhält zur entsprechenden Kennzahlenrelation eine 1:n-Beziehung.

3.2.7 Vom konzeptionellen zum multidimensionalen logischen Modell

Der Übergang vom konzeptionellen zum multidimensionalen Modell gestaltet sich folgendermaßen:

- Im MDBMS wird zunächst ein Würfel definiert, der die Kennzahlen und Dimensionen aufnimmt.
- Die Kennzahlen werden entweder als einzelne Variablen oder als Kennzahlendimension definiert (je nach Produkt).

- Die Dimensionen werden festgelegt.
- Je nach Produkt muß die Zeitdimension explizit gekennzeichnet werden.
- Die Kennzahlen und Dimensionselemente werden nicht als Attribute, sondern direkt als Ausprägungen in den Dimensionen (oder Variablen) abgelegt.
- Beziehungen können 1:1 oder 1:n zwischen Dimensionen definiert werden.
- Eine Hierarchie ist gekennzeichnet durch die Relation einer Dimension auf sich selbst.
- Jedes Dimensionselement muß eine eindeutige Bezeichnung haben (Schlüssel). Doppelte Elemente (bei multiplen Hierarchien) müssen referenziert werden.
- Mit Formeln (Regeln) werden Kennzahlen hergeleitet.

3.2.8 Vom konzeptionellen zum objektorientierten logischen Modell

Die Umsetzung des konzeptionellen Datenmodells in ein objektorientiertes logisches Modell gestaltet sich wie folgt:

- Kennzahlen mit gleichen Dimensionen werden in einer Klasse zusammengefaßt, deren Attribute die Kennzahlen sind.
- Jede Dimension wird als Klasse definiert, welche die Dimensionselemente und -attribute aller Hierarchiestufen als Attribute enthält.
- Jede Dimensionsklasse unterhält zur entsprechenden Kennzahlenklasse eine 1:n-Beziehung.
- Durch die Definition der Beziehung generieren sich die künstlichen Schlüssel automatisch.
- Methoden zur Berechnung von Kennzahlen (Regel) müssen direkt im logischen Objektmodell verankert werden.

3.2.9 Logisches Datenarchitektur

Ungeachtet dessen, welche Technologie (relational, objektorientiert oder multidimensional) verwendet wird, muß, auch abhängig vom verwendeten Softwareprodukt, eine Entscheidung zwischen den beiden Modell-Architekturen Star-Schema, Galaxy-Schema und eventuell auch Snowflake-Schema getroffen werden. Diese Architekturen unterscheiden sich hinsichtlich ihres Speicher- und Zeit- und Konsistenzverhaltens, was bei der Entscheidung für ein Modell berücksichtigt werden muß.

3.2.10 Physischer Aspekte

Um ein Data Warehouse-Modell zu optimieren, müssen noch einige Gedanken an physische Aspekte verloren werden, die insbesondere bei sehr großem Datenumfang ins Gewicht fallen. Dazu gehören die Steigerung der Abfrageperformance und die Aufteilung des Datenbestands.

3.2.10.1 Steigerung der Abfrageperformance

Hier gibt es mehrere Möglichkeiten, das Problem zu geringer Performance zu lösen oder zumindest die Situation zu verbessern.

Vorverdichtung: Um die Abfrage-Performance zu erhöhen, werden Aggregationen vorverdichtet und entweder in der Relation, dem Würfel oder der Klasse selbst abgespeichert oder als separates, verdichtetes Modell integriert (materialized views). Die Entscheidung, ob alle oder nur die wichtigsten Aggregationen vorverdichtet werden, muß durch Benchmarking herausgefunden werden.

Indexierung: Durch spezielle Indexierungstechniken kann ebenfalls versucht werden die Abfrageperformance zu verbessern. Damit ist es beispielsweise möglich, mehrere Verknüpfungen einer Abfrage gleichzeitig auszuführen.

Abfrageoptimierung: Mit speziellen für Data Warehouse ausgelegten Abfrageoptimierern wird versucht, die Reihenfolge der Abfrageabarbeitung zu optimieren.

Parallelisierung: Mittels Multiprozessorsystemen können Abfragen aufgeteilt und parallel bearbeitet werden (Hardwarelösung).

3.2.10.2 Aufteilung großer Datenbestände

Um große Datenbestände auf begrenzten physikalischen Medien ablegen zu können, müssen Verteilungs- und Partionierungsstrategien ausgearbeitet werden.

Verteilung: Große Datenbestände können über Client/Server-Systeme auf verschiedenen Rechnern abgelegt werden.

Partionierung: Sehr große Dimensions- oder Fakttabellen können horizontal (Datengruppen) oder vertikal (Attributgruppen) partioniert werden. u.a. [ChauDay]

3.3 Schema-Integration

Beim Aufbau einer Data Warehouse-Datenbank geht man im allgemeinen so vor, daß zunächst Data Marts für einzelne Geschäftsbereiche entwickelt werden, die nach und nach zu einem Data Warehouse-Modell für das gesamte Unternehmen zusammengefügt werden. Damit dieses Gesamtmodell realisiert werden kann, müssen jedoch bereits beim Aufbau der Data Marts Festlegungen getroffen werden, die ein späteres Zusammenfügen dieser Data Marts zu einem sogenannten Supermart (dem Gesamtmodell) möglich machen. Nur so kann ein erfolgreiches Architekturkonzept für eine Data Warehouse entstehen [Kimb1/98].

Besonders wichtig ist dabei die Definition von einheitlichen Stamm-Dimensionen (z.B. Kunde, Produkt, Region), auf die das gesamte Unternehmen zugreift. Diese setzen sich in der Regel aus einer Kombination von Attributen verschiedenster Quellen zusammen. Dabei sollte die Granularität der Daten so klein wie möglich gehalten werden, damit die Flexibilität gewahrt bleibt. Wichtig ist auch, daß nicht der Schlüssel der Unternehmensstammdaten verwendet wird, sondern, daß ein eigener anonymer Data Warehouse-Schlüssel definiert wird.

Für die Definition der Fakten gilt, daß diese ebenfalls unternehmensweit definiert werden sollten. Es muß ein Glossar angelegt werden, das eine standardisierte Terminologie für die Unternehmenskennzahlen vorgibt. Jede Kennzahl wird nur unter dieser einen Definition angesprochen. Dadurch ist auch die einheitliche Herleitung der Kennzahl gewährleistet.

Geschäftsbereichsspezifische Attribute sollten dagegen in eigenen Dimensionen zusammengefaßt werden, auf die über das zugehörige Data Mart zugegriffen werden kann und die auch dort gewartet werden. Ansonsten würde sich ein unnötig hoher Verwaltungsaufwand für das gesamte Data Warehouse ergeben.

Die korrekte Schema-Integration ist eine Aufgabe für das Metamodell, dessen Wichtigkeit im folgenden Abschnitt ersichtlich wird.

3.4 Metadatenmodellierung

Die Aufbereitung von Metadaten spielt im Data Warehouse-Bereich eine gewichtige Rolle, da sie zum einen der Wahrung der Übersicht in einem solch komplexen System dient, zum anderen lassen sich im Data Warehouse viele Vorgänge mittels Metadaten steuern. Dazu gehört die Datenextraktion und -transformation aus den OLTP-Systemen, der automatisierte Ladevorgang zu bestimmten Zeitpunkten, die Verteilung der Datenbestände auf Data Marts, sowie der Zugriff auf die Analyse- und Reportingtools.

Metadaten für das Data Warehouse lassen sich in drei Kategorien unterteilen [Sach98]:

1. Metadaten für den Data Warehouse-Benutzer

Hier findet der Benutzer Informationen darüber, welche Daten im Data Warehouse enthalten sind, wo sie zu finden sind, wie und ob er darauf zugreifen kann, wie lange der Zugriff dauert und welche Datenqualität er erwarten kann.

2. Metadaten für den Data Warehouse-Administrator

Der Data Warehouse-Administrator findet hier Informationen, die ihn bei der Wartung und Pflege des Data Warehouse unterstützen. Dies sind insbesondere Informationen über die Herkunft der Daten, über die Ladeprogramme und die Tabellen und Abfragen des Data Warehouse. Auch Informationen über die zu verwaltenden Zugriffsrechte sind hier zu finden.

3. Metadaten für den Data Warehouse-Entwickler

Der Data Warehouse-Entwickler hat Zugriff auf Informationen, die ihm bei der Erweiterung und Änderung des Data Warehouse hilfreich sind. Dies sind beispielsweise Informationen über die Quelldaten und -systeme, die Transformations- und Ladeprogramme, sowie über die Data Warehouse-Tabellen und -Abfragen.

Das folgende Bild zeigt ein Metamodell für ein Data Warehouse, in dem gekennzeichnet ist, welche Informationen für wen wichtig sind (Administrator, Entwickler, Benutzer).

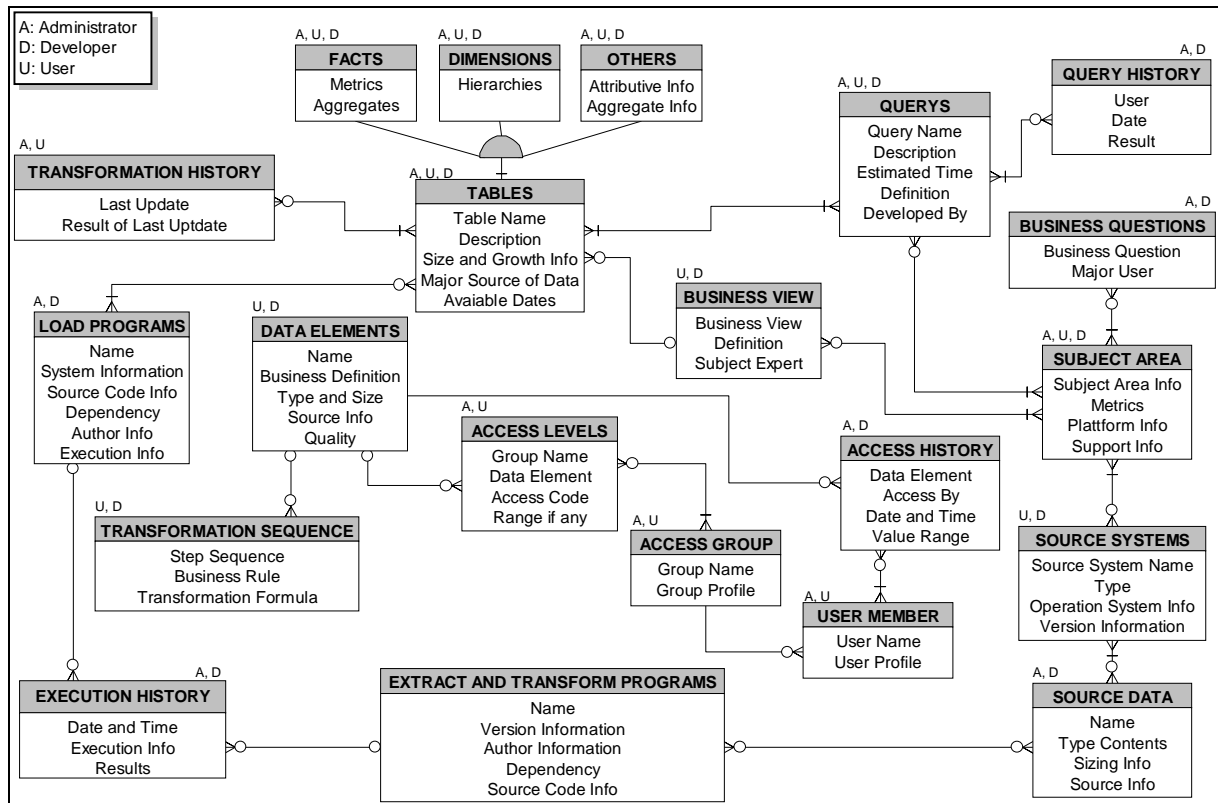


Abbildung 3.1: Metamodell eines Data Warehouse

aus [Sach98]

Hier werden alle Elemente einer Data Warehouse-Implementierung (siehe Abb. 1.1), von den Quellsystemen (OLTP) bis zu den Analysen (OLAP), aufgezeigt und in Beziehung zueinander gesetzt. Es werden die Quellsysteme und -Daten genannt, aus denen das Data Warehouse gespeist wird. Die Extraktions- und Transformationsprogramme werden beschrieben und Informationen über die Ladeprogramme werden bereitgestellt. Es wird angegeben, welche Tabellen von welchen Programmen geladen werden und was in diesen Tabellen enthalten ist (Dimensionen, Fakten u.a.). Es ist ersichtlich welche Benutzer es gibt und wie die Zugriffsrechte auf die einzelnen Datenelemente geregelt sind. Gezeigt wird außerdem, welche Abfragen es auf den Datenbestand gibt und welchen Geschäftsbereichen die Daten angehören. Zudem werden Abläufe dokumentiert; wann welche Daten geladen worden sind, welche Abfrage abgefragt werden und wann welcher Benutzer auf welche Datenelemente zugreift.

Man sieht, daß mittels dieses zwar komplexen, aber sinnvollen Modells eine gewisse Übersicht geschaffen werden kann, die zur Qualitätssicherung der Data Warehouse-Daten unerlässlich ist.

4 Kriterienkatalog

Um konzeptionelle Modellierungsmethoden, logische Architekturmodelle und physische Datenbankeigenschaften für ein Data Warehouse untersuchen zu können, wird eine Kriterienkatalog benötigt, der Merkmale festlegt, mit denen sich die verschiedenen Methoden, Modelle und Systeme vergleichen lassen. Eine normierte Bewertungsrichtlinie ist die DIN ISO 9126 für Software-Qualitätsmerkmale. Dort werden benutzerorientierte Qualitätsmerkmale definiert, die durch Unterpunkte verfeinert werden, in denen softwareorientierte Faktoren zum Tragen kommen, die mittels Qualitätsindikatoren bzw. Metriken gemessen und bewertet werden können [Balz96].

4.1 Bewertungskriterien

Ein Teil dieser Qualitätsmerkmale eignet sich durchaus auch, um die Anforderungen miteinander zu vergleichen, die bei der Entwicklung eines Data Warehouse eine Rolle spielen. Die folgende Grafik zeigt, welche der DIN ISO 9126-Kriterien für den Vergleich konzeptioneller Modellierungsmethoden, logischer Architekturmodelle und physischer Datenbankeigenschaften geeignet sind. Da keine speziellen Softwareprodukte bewertet werden sollen, sind nicht alle Qualitätsmerkmale des DIN ISO 9126-Modells relevant.

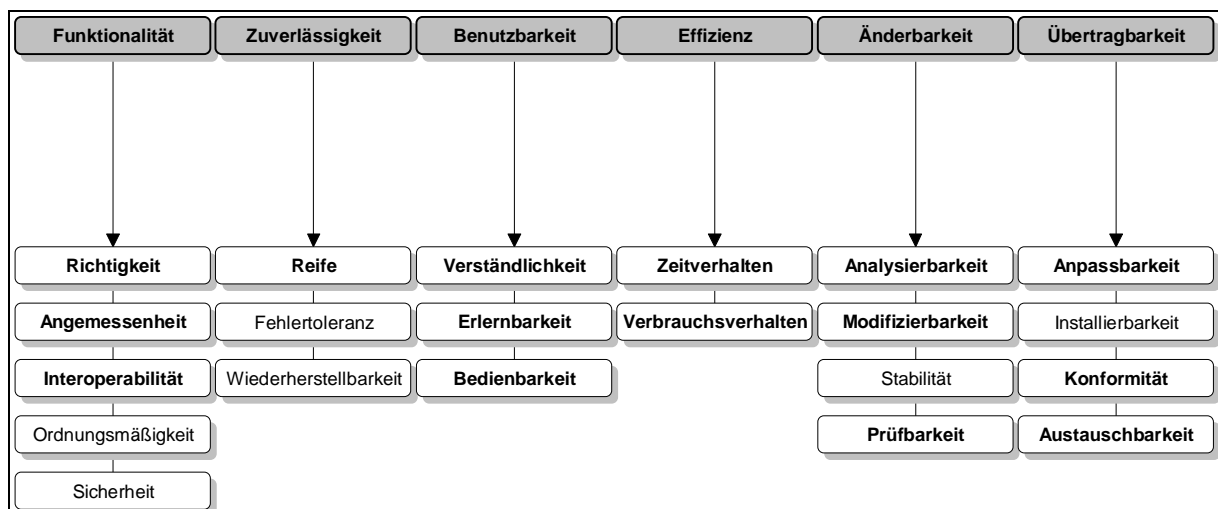


Abbildung 4.1: Bewertungskriterien nach DIN/ISO 9126

Funktionalität:

Um die Funktionalität einer konzeptionellen Modellierungsmethode (A), eines logischen Architekturmodells (B) oder eines bestimmten Datenbanktyps (C) zu überprüfen, werden folgende Teilmerkmale untersucht:

Richtigkeit: Werden die richtigen bzw. die vereinbarten Ergebnisse geliefert? Lässt sich der zu modellierende Sachverhalt mit der Methode, Architektur bzw. Technologie korrekt abbilden? (A, B, C)

Angemessenheit: Lassen sich die speziellen Konstrukte eines Data Warehouse-Modells mit der Methode, Architektur bzw. Technologie darstellen? Dies betrifft insbesondere die multidimensionale Sichtweise des Sachverhaltes mit allen unter 3.2.5 angesprochenen Punkten zur Erstellung spezieller Konstrukte. (A, B, C)

Interoperabilität: Besteht die Fähigkeit mit anderen Systemen zusammenzuwirken? Wird die Methode von CASE-Tools unterstützt (A)? Gibt es Schnittstellen zu anderen Systemen (C)?

Zuverlässigkeit:

Zuverlässigkeit ist die Fähigkeit, über längere Zeit als Standard zu gelten. Ein Kriterium, mit dem die Zuverlässigkeit bewertet werden kann, ist die

Reife: Ist die Methode bzw. Technologie bereits ausreichend wissenschaftlich erforscht und in der Praxis bewährt (A, C)? Sind durch die Architektur Fehlzustände möglich (B)?

Benutzbarkeit:

Die Benutzbarkeit beschreibt den Aufwand, der zur Benutzung erforderlich ist. Unter dieses Merkmal fallen folgende Teilkriterien:

Verständlichkeit: Wie hoch ist der Aufwand für den Benutzer, das Konzept der Methode, der Architektur bzw. der Technologie zu verstehen (A, B, C)?

Erlernbarkeit: Wie hoch ist der Aufwand für den Benutzer, die Methode, die Architektur bzw. die Technologie zu erlernen (A, B, C)?

Bedienbarkeit: Wie kompliziert ist es für den Benutzer, mit der Methode, der Architektur bzw. der Technologie zu arbeiten (A, B,C)?

Effizienz:

Die Effizienz beschreibt das Verhältnis zwischen dem Leistungsniveau der Methode, der Architektur und der Technologie und dem Umfang der dafür eingesetzten Betriebsmittel. Die Effizienz ist bewertbar durch:

Zeitverhalten: Wie hoch ist der Zeitaufwand, um mit der Methode, Architektur bzw. Technologie ein Modell zu erstellen (A, B)? Wie ist das Zeitverhalten der Architektur bzw. der Technologie (B,C)?

Verbrauchsverhalten: Wie hoch ist der Einsatz an Betriebsmittel, um mit der Methode, der Architektur bzw. der Technologie ein Modell zu erstellen (A, B, C)? Betriebsmittel sind Platzbedarf (A), Modellgröße (B) und Speicherplatz (B, C).

Änderbarkeit:

Die Änderbarkeit gibt an, wie hoch der Aufwand zur Durchführung von Änderungen ist. Dies können Korrekturen, Verbesserungen oder Anpassungen sein. Die Änderbarkeit läßt sich bewerten durch die:

Analysierbarkeit: Gibt es für die Methode, die Architektur bzw. die Technologie softwaretechnische Unterstützung, um Mängel zu diagnostizieren und Fehler zu erkennen (A, B, C)?

Modifizierbarkeit: Wie hoch ist der Aufwand, um mit der Methode, der Architektur oder der Technologie Verbesserungen, Fehlerbeseitigungen oder Anpassungen an Umgebungsänderungen durchzuführen (A, B, C)?

Prüfbarkeit: Gibt es softwaretechnische Unterstützung zur Überprüfung des geänderten Modells (A, B)?

Übertragbarkeit:

Die Übertragbarkeit beschreibt die Eignung der Methode, der Architektur bzw. der Technologie, von einer Umgebung (konzeptionell, logisch, physisch) in eine andere übertragen zu werden.

Anpaßbarkeit: Wie hoch ist der Aufwand, ein Modell von einer konzeptionellen Methode in eine bestimmte Architektur umzusetzen (A, B)? Wie hoch ist der Aufwand, ein Modell von einer konzeptionellen Methode in eine bestimmte Technologie umzusetzen (A)?

Konformität: Ist die konzeptionelle Methode neutral gegenüber der folgenden Weiterverarbeitung? Wird eine bestimmte Architektur impliziert (A, B)? Wird eine bestimmte Technologie impliziert (A)?

Austauschbarkeit: Muß das Modell geändert werden, wenn eine andere Methode, Architektur bzw. Technologie eingesetzt wird (A, B, C)?

Es muß angemerkt werden, daß die Bewertungen trotz objektiver Bewertungskriterien natürlich meinem subjektiven Eindruck unterliegen und somit nicht als allgemein gültig angesehen werden können.

4.2 Bewertungsmetrik

Da die Bewertungen der konzeptionellen Methoden, logischen Architekturen und phys. Technologien miteinander verglichen werden sollen, werden als Vergleichsmaß die Erfüllungsgrade gut (+1), mittel (0) und schlecht (-1) gewählt.

5 Konzeptionellen Methoden und ihrer Elemente

Im Rahmen dieser Arbeit werden nun einige konzeptionelle Datenmodellierungsmethoden für Data Warehouse vorgestellt, die nachfolgend anhand eines einheitlichen Beispiels miteinander verglichen und bewertet werden.

5.1 Methode ADAPT

ADAPT (Application Design for Analytical Processing Technologies) ist eine von Dan Bulos 1996 veröffentlichte Methode zur konzeptionellen Modellierung von OLAP-Datenbanken [Bulo96]. ADAPT orientiert sich an den OLAP-Forderungen und enthält Symbole für alle relevanten OLAP-Komponenten. Diese Symbole sind unterteilt in Kernelemente, Dimensionstypen, Zusatzelemente und Teilaggregate [GaGlu98].
















Kernelemente		Zusatzelemente	
	Würfel		Dimensionshierarchiestufe
	Berechnungsvorschrift		Dimensionselement
	Hierarchie		Dimensionsattribut
	Datenquelle		Relation zwischen Dimensionen
Dimensionstypen		Teilaggregate	
	Dimensionshierarchie		Würfelausschnitt
	Eigenschaftsdimension		Dimensionsausschnitt
	Versionendimension		Exklusiv-Oder
	Kennzahlendimension		

Abbildung 5.1: ADAPT-Symbole

aus [GaGlu97]

5.1.1 Kernelemente

Kernelemente von ADAPT sind der Datenwürfel, die Berechnungsvorschrift, die Hierarchie und die Datenquelle.

Datenwürfel: Der Datenwürfel repräsentiert den sogenannten Hypercube. Jeder Datenwürfel besitzt eine Bezeichnung und besteht aus mehreren Dimensionen, die auf Kennzahlen zugreifen. Eine OLAP-Anwendung kann aus einem oder mehreren Datenwürfeln bestehen.

Berechnungsvorschrift: Eine Berechnungsvorschrift gibt an, wie eine Kennzahl berechnet wird. Die Herleitung einer Berechnung kann mehrere Dimensionen umspannen.

Hierarchie: Eine Hierarchie ist in mehrere Ebenen (Hierarchiestufen) unterteilt. Pro Dimension können mehrere Hierarchien vorhanden sein.

Datenquelle: Die Datenquelle gibt den Ursprung eines Dimensionselements oder -attributs an, das heißt, aus welcher externen Datenquelle das Element oder Attribut stammt.

5.1.2 Dimensionstypen

ADAPT unterteilt die Dimensionen eines Datenwürfels in vier Typen, nämlich Dimensionshierarchie, Eigenschaftsdimension, Versionendimension und Kennzahlendimension.

Dimensionshierarchie: Diese Dimension besteht aus mehreren hierarchisch angeordneten Ebenen (Hierarchiestufen).

Eigenschaftsdimension: Dimensionsattribute werden in einer separaten Dimension verwaltet.

Versionendimension: Eine Versionendimension repräsentiert unterschiedliche Varianten der Fakten, beispielsweise Istdaten und Solldaten.

Kennzahlendimension: Mit einer Kennzahlendimension lassen sich die Fakten gruppieren (z.B. Währung in DM oder Euro).

5.1.3 Zusatzelemente

Zusatzelemente beschreiben die einzelnen Dimensionen eines Datenwürfels näher. Hier gibt es Symbole für die Dimensionshierarchiestufe, das Dimensionselement, das Dimensionsattribut und für die Beziehung (Relation) zwischen Dimensionen.

Dimensionshierarchiestufe: Eine Dimensionshierarchiestufe bezeichnet eine Ebene einer Hierarchie.

Dimensionselement: Ein Dimensionselement beschreibt ein abfragbares Element einer Dimension.

Dimensionsattribut: Ein Dimensionsattribut ist ein beschreibendes Element einer Dimension.

Relation: Eine Relation zeigt eine Beziehung zwischen zwei Dimensionen an (ER-Notation).

5.1.4 Teilaggregate

Teilaggregate beschreiben eine Untermenge des Datenvolumens. Dazu gehören der Würfelausschnitt, der Dimensionsausschnitt und das Exklusiv-Oder.

Würfelausschnitt: Volumenausschnitt aus einem Datenwürfel (Dice).

Dimensionsausschnitt: Schnitt durch einen Würfel (Slice).

Exklusiv-Oder: Unterteilung einer Dimension.

5.2 Methode DF

DF (Dimensional Fact) ist eine erst kürzlich von Golfarelli/Maio/Rizzi [GoMaRi98] veröffentlichte konzeptionelle Modellierungsmethode, die ebenfalls den Anspruch erhebt, speziell auf Data Warehouse-Anwendungen zugeschnitten zu sein. Sie ist graphenorientiert, wodurch sich ein strukturierter, baumartiger Aufbau ergibt. DF besteht aus den Grundelementen Fakt, Dimensionselement, Dimensionsattribut, Dimension, Hierarchie und Aggregatsfunktion, aus denen sogenannte Fakt-Schemata gebildet werden.

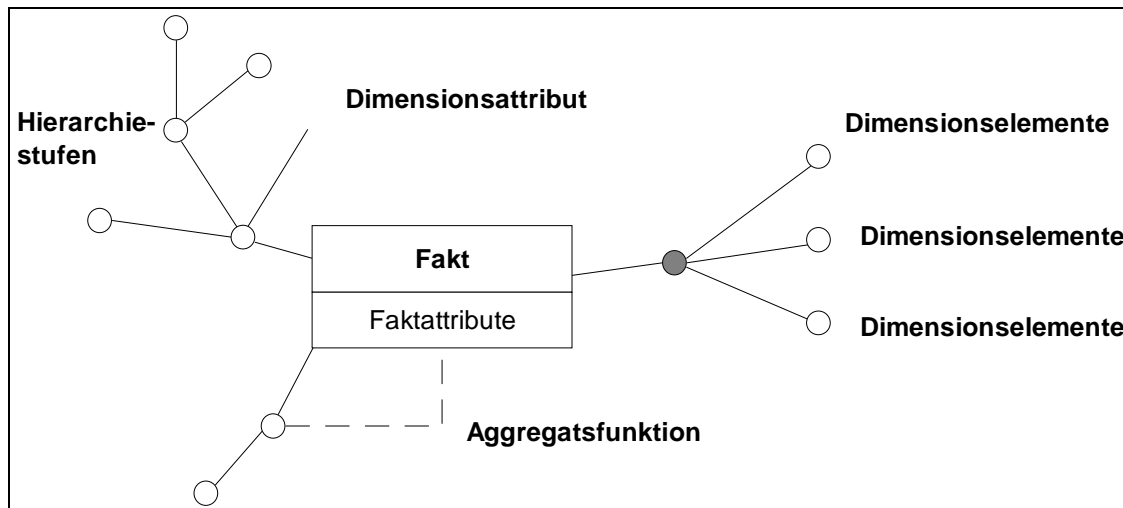


Abbildung 5.2: DF-Symbole

Mehrere Kennzahlen werden dabei unter einer gemeinsamen Bezeichnung (Fakt) zusammengefasst. Diese Kennzahlen (Fakt-Attribute) bilden dabei das Zentrum eines baumartigen Gebildes (sozusagen die Wurzel des Baumes), wovon sich die Dimensionen in Form von Ästen hierarchisch ausbreiten. Eine Hierarchiestufe wird dabei durch einen Knoten gekennzeichnet der gleichzeitig das Dimensionselement angibt. Die den Kennzahlen am nächsten befindliche Hierarchiestufe stellt dabei die Stufe der geringsten Granularität dar. Äste, die nicht mit einem Knoten enden, stellen nicht-dimensionale Attribute dar. Sie können nicht zur Aggregation genutzt werden, sondern geben lediglich beschreibende Informationen zu einem Element einer Hierarchiestufe an. Nicht-hierarchische Dimensionen lassen sich mittels eines dunkel gefärbten Hilfsknotens darstellen. Im DF-Modell sind alle Kennzahlen per Voreinstellung additiv über alle Dimensionen. Bei Abweichungen wird mittels einer bezeichneten Strichlinie von der Kennzahl zur Hierarchiestufe gekennzeichnet, ob überhaupt eine, bzw. welche Aggregatsfunktion statt des Summenoperators verwendet wird. Regeln, die beschreiben, wie Kennzahlen berechnet werden, müssen mit DF extern in einem Glossar beschrieben werden [GoMaRi98].

5.3 Methode ER

Die Entity-Relationship-Methode von Chen [Che76] dient eigentlich der konzeptionellen Modellierung relationaler operativer Datenbanksystemen und ist aufgrund dessen weit verbreitet.

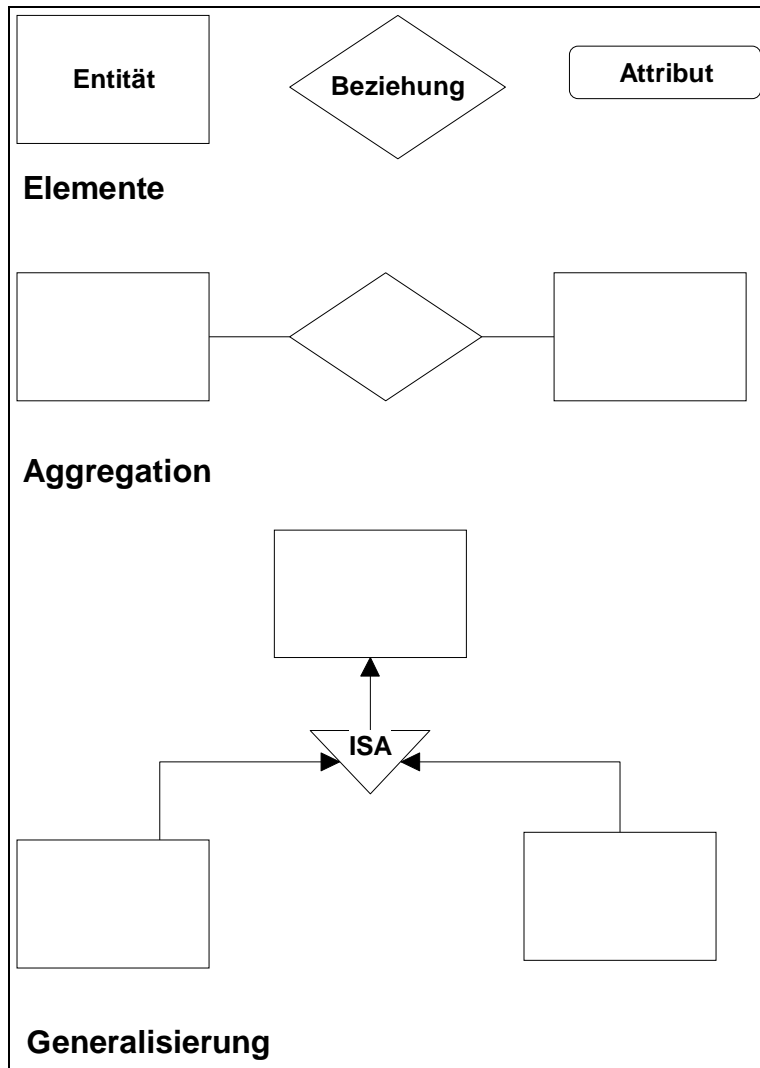


Abbildung 5.3: ER-Symbole

ER bietet keine expliziten Symbole für OLAP, sondern beruht auf dem Prinzip von Entitäten und deren Beziehungen zueinander. Sie besteht deshalb auch nur aus den drei Symbolen für Entität, Beziehung und Attribut. An Beziehungskonstrukten gibt es die Aggregation, mit der sich allgemein Beziehungen darstellen lassen, und die Generalisierung, die eine Erweiterung der ER-Methode von Chen darstellt, und mit der sich Allgemein/Speziell-Beziehungen abbilden lassen.

5.4 Methode UML

Die Unified Modeling Language [Oes97] ist eine relativ neue Modellierungsmethode, die auf den Methoden von Booch, Jacobsen und Rumbaugh aufbaut. Sie ist eigentlich speziell für die Modellierung von objektorientierten Anwendungen entwickelt worden. UML ist eine sehr umfangreiche Methode, mit der eine ganze Reihe verschiedener statischer und dynamischer Diagrammtypen modelliert werden können. Hier soll nur das statische Klassendiagramm beschrieben werden, mit dem Datenstrukturen abgebildet werden können.

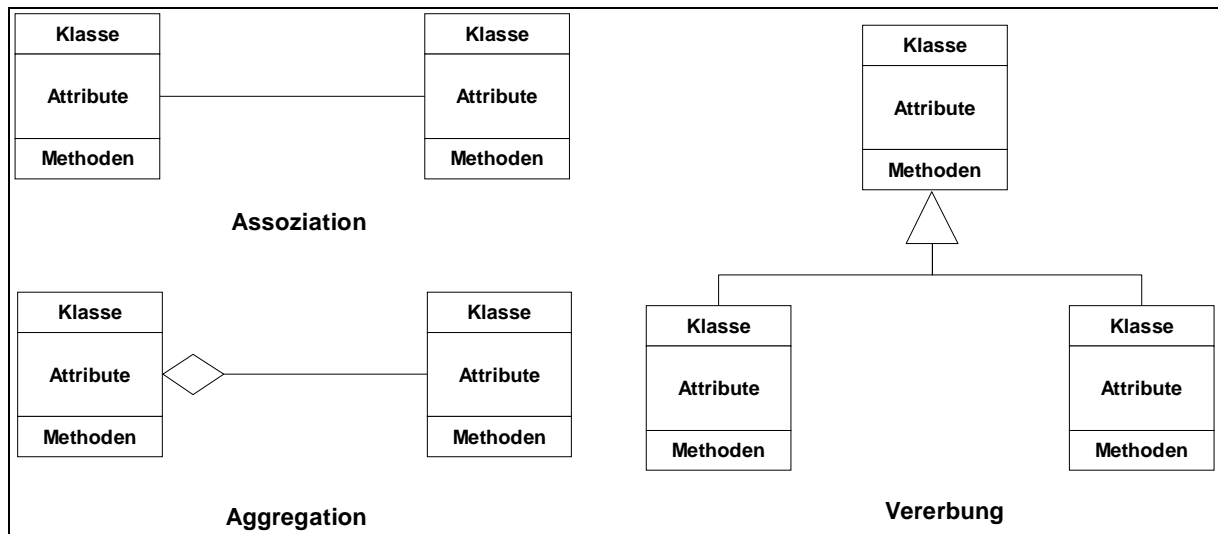


Abbildung 5.4: UML-Symbole

Das Grundelement der UML ist die Klasse. Sie besteht aus dem Klassennamen und den der Klasse zugehörigen Attributen und Methoden (Operationen). An Beziehungskonstrukten gibt es die Assoziation, mit der Menge/Element-Beziehungen zwischen Objekten dargestellt werden, die Aggregation, mit der Teile-Ganzes-Hierarchien abgebildet werden und die Vererbung mit der sich Generalisierung/Spezialisierung darstellen lässt.

Unter [Tot98] findet sich eine Notation zur Modellierung einer multidimensionalen Datenstruktur und zur Bildung von speziellen Klassen mit der Unified Modeling Language.

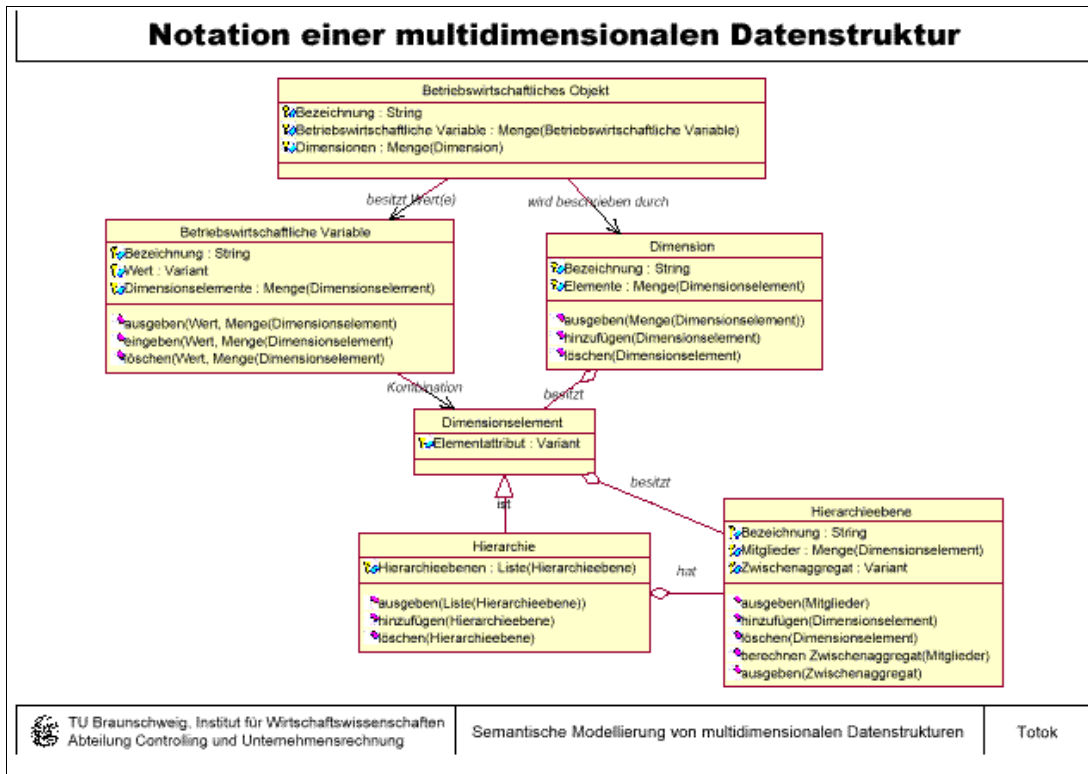


Abbildung 5.5 Notation einer multidimensionalen Datenstruktur aus [Toto98]

Hier wird der das Metamodell einer typischen multidimensionalen Struktur auf Basis von UML gezeigt. Die Klasse Betriebswirtschaftliches Objekt kennzeichnet den sogenannten Datenwürfel, der die Kennzahlen in Form einer Menge der Klasse Betriebswirtschaftliche Variable und die einzelnen Dimensionen in Form einer Menge der Klasse Dimension besitzt. Die Beziehung zwischen Datenwürfel und Kennzahl bzw. Datenwürfel und Dimension ist als gerichtete Assoziation dargestellt, das heißt, der Datenwürfel weiß, welche Kennzahlen und Dimensionen er besitzt, diese jedoch wissen nicht, welchem Datenwürfel sie angehören. Eine Klasse Dimension besteht aus einer Menge von Dimensionselementen. Eine Kennzahl in Form der Klasse Betriebswirtschaftliche Variable leitet sich aus einer Kombination von Dimensionselementen her. Das Klasse Dimensionselement ist Teil einer Klasse Hierarchie, die aus einer Sequenz (Liste) von Hierarchiestufen besteht. Eine Hierarchieebene besteht wiederum aus einer Menge von Dimensionselementen.

Um verschiedene Typen von Dimensionen und Hierarchieebenen zu erhalten, werden mittels Vererbung Subtypen der Klasse Dimension und der Klasse Hierarchieebene gebildet (Abb.5.6).

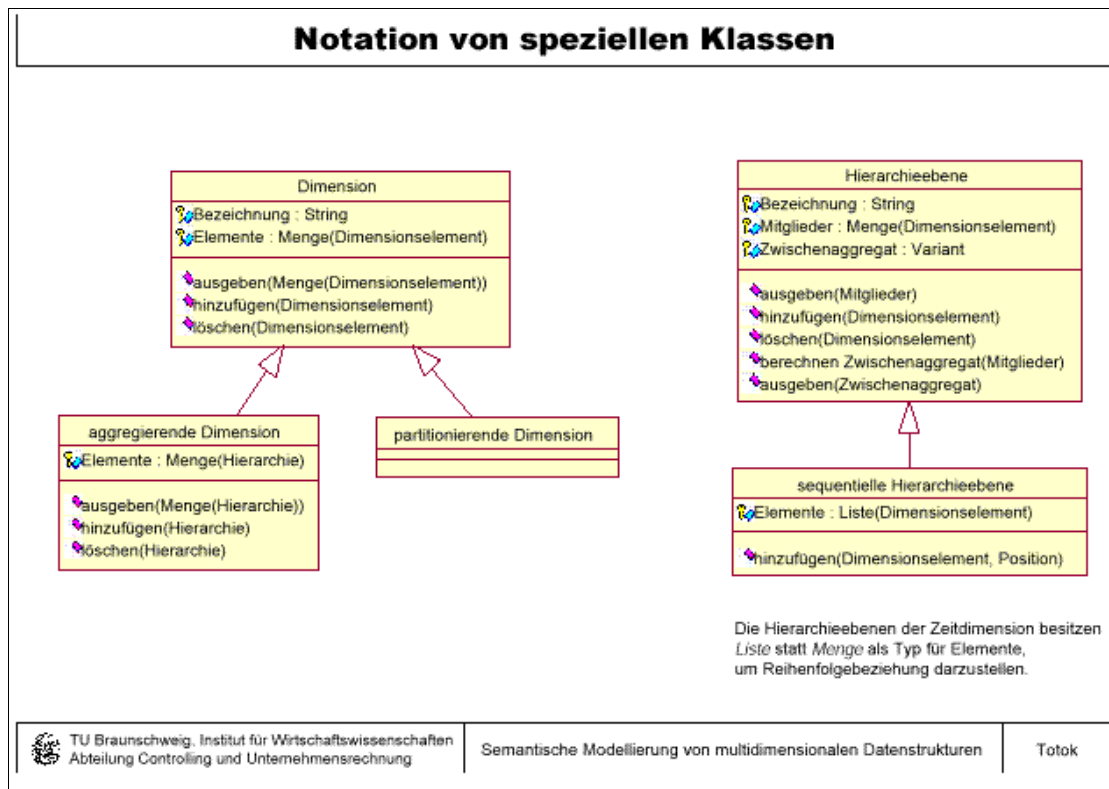


Abbildung 5.6 Notation von speziellen Klassen aus [Toto98]

Eine Dimension kann eine aggregierende (hierarchische) Dimension oder eine partitionierende (nicht-hierarchische) Dimension sein. Nicht-hierarchische Dimensionen sind beispielsweise Versions- oder Kennzahlendimensionen.

In einer Hierarchieebene kann die Menge von Dimensionselementen auch sequentiell angeordnet sein, wie beispielsweise in einer Zeit-Hierarchie.

6 Modellierungsbeispiel

Anhand eines durchgängig modellierten Beispiels soll der gesamte Modellierungsprozeß abgebildet werden, damit im Anschluß daran ein Vergleich und eine Bewertung der konzeptionellen Methoden, logischen Architekturen und der verwendeten Technologien stattfinden kann.

Insgesamt war es schwierig ein adäquates Beispiel für ein Data Warehouse-Modell zu finden. Zum einen sollte das Beispiel aus einem für Data Warehouse-Anwendungen relevanten Geschäftsbereich stammen, zum andere sollten ausreichend Datensätze enthalten sein, um die Probleme mit Datenvolumen und Geschwindigkeit realistisch zu erfassen. Es gibt einige Beispiele, die bereits als Data Warehouse-Modell vorliegen [Kimb96]. Da ich jedoch den Transformationsprozeß von OLTP nach OLAP in die Modellierung einbringen wollte, habe ich letztendlich eine Datenbank gewählt, die als relationales Datenmodell unter MS-Access im Internet zu finden ist (<http://www.rz.fhtw-berlin.de/s4694/agent/datenbank.html>). Es wird sich zeigen, ob sie für die Modellierung geeignet ist.

6.1 Vorstellung des Beispiels

Das Modell eines Unternehmensgeschäftsbereichs, der in ein Data Warehouse-Schema (oder in diesem Fall ein Data Mart) umgesetzt werden soll, liegt als relationales Datenbankschema aus einem OLTP-System vor. Bei dem Beispiel handelt es sich um den Geschäftsbereich Vertrieb, über den die Bestellungen des (fiktiven) Unternehmens abgewickelt werden.

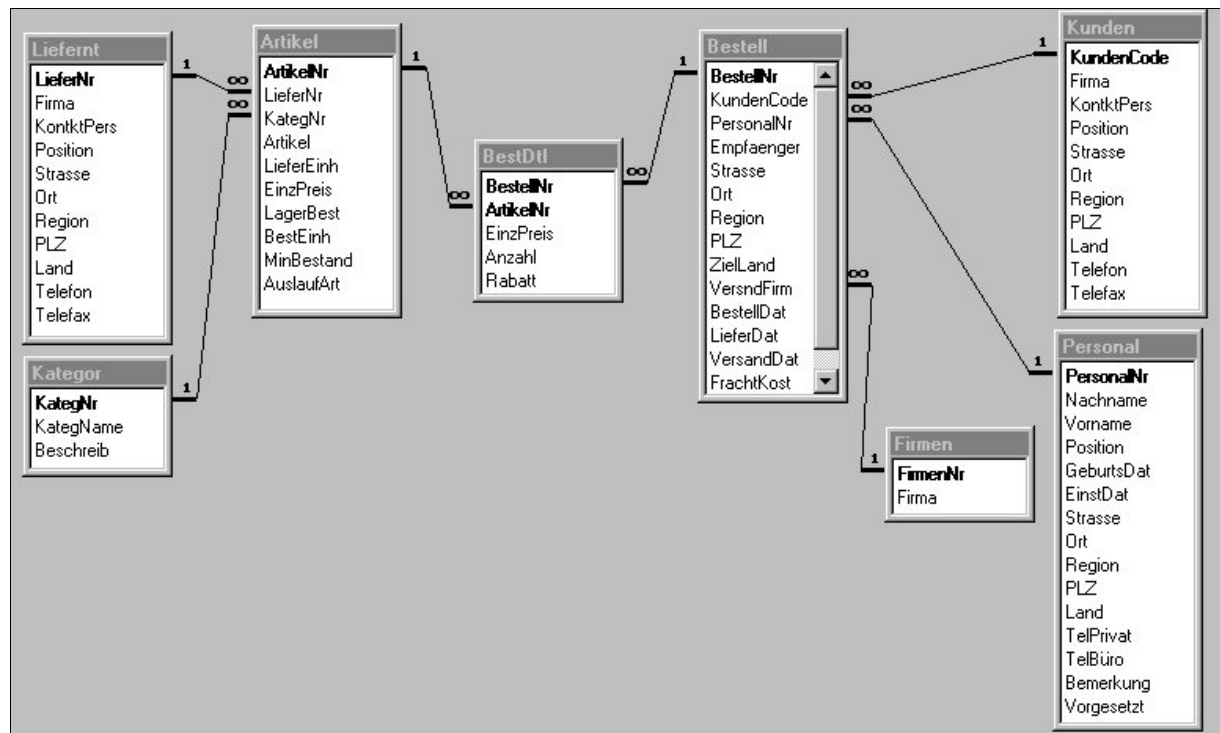


Abbildung 6.1: Modellierungsbeispiel Bestellungen als relationales Datenbankschema

Dieses Modell besteht aus sechs Tabellen mit Stammdaten, die in die Bestellungen und Bestelldetails einfließen. Die Stammdateien Kunden, Personal und Versandfirmen stehen jeweils in einer 1-n-Beziehung zur Bestelldatei, welche die Kopfdaten einer Bestellung enthält. Da eine Bestellung jedoch mehrere Artikel umfassen kann, ist der Bestell-Datei eine Bestelldetail-Tabelle untergeordnet (1-n), die wiederum eine n-1-Beziehung zur Stammdatei Artikel enthält. Die Artikel-Datei steht zudem in n-1-Beziehung zu einer Lieferanten- und einer Kategorie-Datei, da eine Artikel von mehreren Lieferanten geliefert und zu mehreren Kategorien (Produktgruppen) gehören kann. Insgesamt enthält die Bestell-Datei 1078 Datensätze und die Bestelldetail-Datei 2820 Datensätze. Es sind 77 Artikel, 29 Lieferanten, 8 Kategorien, 91 Kunden, 15 Mitarbeiter und 3 Versandfirmen als Stammdaten vorhanden.

6.2 Klärung des Analysebedarfs

In dem zu modellierenden Data Mart sollen nun die Bestellungen und die Bestelldetails ausgewertet werden. Da keine realen Benutzer vorhanden sind, werden folgende Festlegungen getroffen.

Welche Geschäftsprozesse sollen analysiert werden?

Bestellungen

Mit welchen Kennzahlen (Fakten) wird gearbeitet?

Umsätze, Frachtkosten

Welchen Detaillierungsgrad (Granularität) müssen die Daten besitzen?

Monatlich

Über welche Dimensionen ist eine Auswertung sinnvoll?

Umsatz: Artikel, Kunden, Personal, Versandfirma, Zeit

Frachtkosten: Kunden, Personal, Versandfirma, Zeit

Für die Frachtkosten wäre sicher auch die Auswertung pro Artikel interessant. Da jedoch im OLTP-System die Frachtkosten nur pro Bestellung und nicht pro Einzelartikel erfaßt sind, käme es hier aufgrund der zu geringen Granularität der Frachtkosten zu falschen Daten.

Wie lassen sich diese Dimensionen verdichten?

Artikel: Artikel - Lieferant - Ort - Land - Gesamt, Artikel - Kategorie - Gesamt

Kunde: Kunde - Ort - Land - Gesamt

Personal: Mitarbeiter - Gesamt

Versandfirma: Versandfirma - Gesamt

Zeit: Monat - Quartal - Jahr

Welche Dimensionsattribute werden benötigt?

Keine

Sind die Attribute stabil oder ändern sie sich mit der Zeit?

Stabil

Da hier die Daten aus einer Quelle stammen, befinden sich die Begriffe bereits in bereinigter Form, so daß direkt mit ihnen gearbeitet werden kann.

6.3 Konzeptionelle Modellierung

Der für das Data Warehouse relevante Geschäftsprozeß wird nun mit den vier vorgestellten konzeptionellen Methoden ADAPT, DF, ER und UML modelliert. Um ein korrektes Modell zu erstellen, in dem alle Abhängigkeiten ersichtlich sind, empfiehlt es sich, die Modellierung immer als Galaxy-Schema darzustellen. Da hier zwei Kennzahlen ausgewertet werden sollen, wird für jede Kennzahl ein Diagramm mit den zugehörigen Dimensionen erstellt. Diese dadurch teilweise doppelt vorkommenden Dimensionen sind später im logischen Modelle nur

einmal vorhanden. Um die Übersichtlichkeit zu wahren ist es jedoch sinnvoll, Kennzahlen, die unterschiedliche Dimensionen besitzen separat zu modellieren.

6.3.1 ADAPT

Um die Fähigkeiten von ADAPT besser aufzeigen zu können, wird das recht einfache Beispiel um eine Kennzahlen-Dimension (Variablen) und eine Szenario-Dimension erweitert.

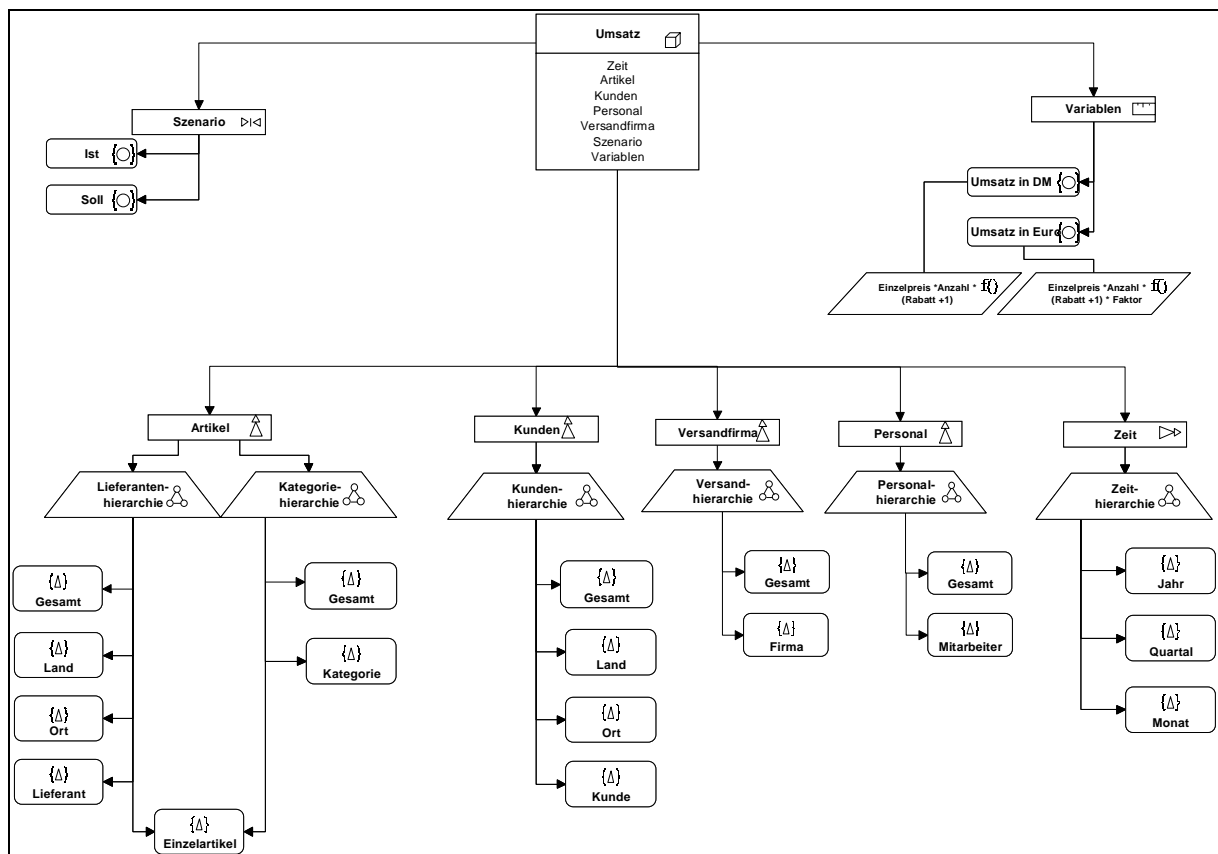


Abbildung 6.2: Konzeptionelles ADAPT-Modell Umsatz

Der mittig angeordnete Umsatzwürfel zeigt auf, über welche Dimensionen die Kennzahl Umsatz ausgewertet werden kann. Die Dimension Artikel besitzt zwei Hierarchien, die Lieferantenhierarchie, mit den Hierarchiestufen Gesamt, Land, Ort, Lieferant und Einzelartikel, und die Kategoriehierarchie mit den Hierarchiestufen Gesamt, Kategorie und Einzelartikel. Die Dimension Kunden ist ebenfalls hierarchisch angeordnet. Die Kundenhierarchie besitzt die Hierarchiestufen Gesamt, Land, Ort und Kunde. Die Dimension Versandfirma lässt sich über die Hierarchiestufen Gesamt und Firma als Versandhierarchie verdichten. Ebenso die Dimension Personal, die in der Personalhierarchie die Hierarchiestufen Gesamt und Mitarbeiter besitzt. Die Zeithierarchie der Dimension Zeit wird in die Hierarchiestufen Jahr, Quartal und Monat eingeteilt. Da sich die Zeit

zwar hierarchisch verdichten läßt, aber ihre Elemente sequentiell aufeinanderfolgen, wird zur Kennzeichnung das Dimensionssymbol um 90 Grad gedreht. Die Szenario-Dimension zeigt auf, daß die Kennzahl Umsatz als Ist- und Sollwert vorhanden sein kann. In der Kennzahlen-Dimension Variablen sieht man, in welchen Ausprägungen (hier Währungen) eine Kennzahl vorhanden sein kann. Die Berechnungsvorschrift ($f()$) gibt an, aus welchen Kennzahlen des OLTP-Systems sich der Umsatz berechnet.

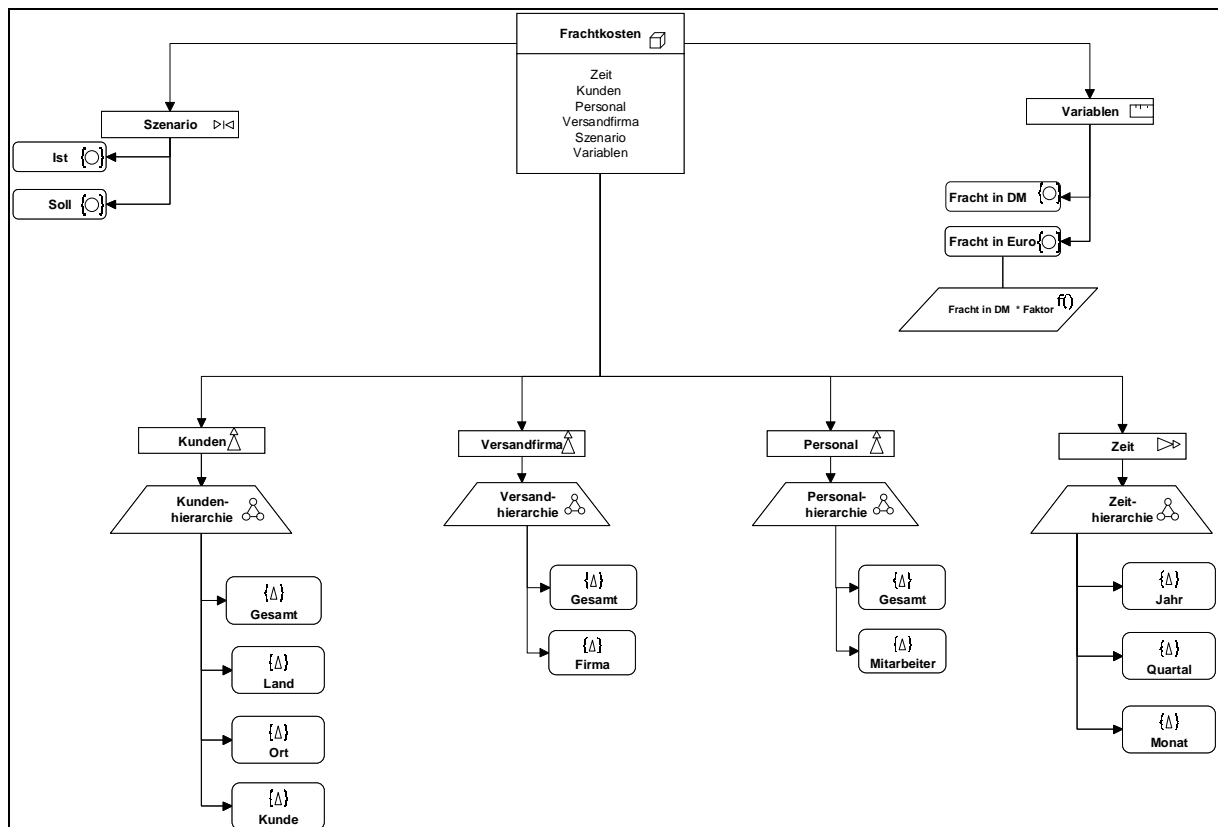


Abbildung 6.3: Konzeptionelles ADAPT-Modell Frachtkosten

Das Schema für die Kennzahl Frachtkosten enthält ebenfalls die Dimensionen Kunden, Versandfirma, Personal, Zeit, Szenario und Variablen. Die Dimension Artikel fehlt jedoch aus den oben genannten Gründen. Da die Frachtkosten direkt aus dem OLTP-System übernommen werden können, ist keine zusätzlich Berechnungsfunktion nötig.

6.3.1.1 Von ADAPT zum relationalen logischen Schema

Um aus einem ADAPT-Modell ein relationales logisches Schema zu erstellen, sind folgende Schritte notwendig:

- Aus der Kennzahlendimension wird eine Fakttable erstellt, welche die Dimensionselemente der Kennzahlendimension als Attribute enthält. Die Berechnung der Kennzahlen erfolgt im DMBS und ist nicht im logischen Schema ersichtlich.
- Die Dimensionselemente der Szenariodimension erweitern die Attributmenge der Fakttable um jeweils eine Ist- und eine Sollversion der Kennzahl.
- Die Dimensionshierarchien werden zu Dimensionstabellen zusammengefaßt, deren Attribute die einzelnen Hierarchiestufen sind.
- Für jede Dimensionstabelle wird eine Primärschlüssel definiert.
- Die Fakttable erhält einen zusammengesetzten Schlüssel aus den Primärschlüsseln der Dimensionstabellen.
- Die Fakt- und Dimensionstabellen werden gemäß Star- oder Galaxy-Schema angeordnet.
- Die Dimensionstabellen können normalisiert als Snowflake-Schema dargestellt werden.

6.3.1.2 Von ADAPT zum multidimensionalen logischen Schema

Um ein ADAPT-Modell auf eine multidimensionale logische Struktur abzubilden, ist folgendes notwendig:

- Der Datenwürfel, die Kennzahlen- und die Szenariodimension können unverändert in das multidimensionale Modell übernommen werden. Die Berechnungsvorschriften können pro Kennzahl definiert werden (Rules).
- Pro Dimensionshierarchie wird eine Dimension definiert. Die Zeitdimension wird explizit gekennzeichnet.
- Die Hierarchiestufen werden direkt als Ausprägungen (Werte) hierarchisch in jeder Dimension abgelegt. Diese müssen eindeutige Werte besitzen (Schlüssel).

6.3.1.3 Von ADAPT zum objektorientierten logischen Schema

Um von ADAPT zu einem objektorientierten logischen Schema zu kommen, muß folgendes getan werden:

- Aus der Kennzahlendimension wird eine Faktklasse erstellt, welche die Dimensionselemente der Kennzahlendimension als Attribute enthält. Die Berechnung der Kennzahlen erfolgt als Methode in dieser Klasse.
- Die Dimensionselemente der Szenariodimension erweitern die Attributmenge der Faktklasse um jeweils eine Ist- und eine Sollversion der Kennzahl.
- Jede Dimensionshierarchie wird als Klasse definiert. Die einzelnen Hierarchiestufen werden als Dimensionselemente zu Attributen dieser Klasse.
- Die Fakt- und Dimensionsklassen werden gemäß Star- oder Galaxy-Schema angeordnet.
- Die Dimensionsklassen können auch als Snowflake-Schema dargestellt werden.

6.3.1.4 Subjektive Bewertung von ADAPT

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Mit der Methode ADAPT lassen sich, bei richtiger Anwendung, korrekte multidimensionale Datenmodelle erstellen. (+1)

Angemessenheit: Die Methode ADAPT enthält sehr viele verschiedene Elemente, durch die eine sehr gute Strukturierung erzielbar ist. Hier werden noch nicht einmal alle verwendet. Es wird angegeben, welche Kennzahl ausgewertet wird und welche Dimensionen daran beteiligt sind. Die Beziehungskardinalität zwischen Dimensionen und Fakten wird dabei jedoch nicht deutlich. Für die einzelnen Dimensionen lassen sich Hierarchien festlegen, deren Hierarchiestufen gekennzeichnet sind. Mehrfache Hierarchien sind hierbei ebenso gut darstellbar, wie multiple Konsolidierungspfade. Es lassen sich auch nicht-hierarchische Dimensionen erkennbar aufzeigen, wie beispielsweise Ist- und Solldaten. Außerdem sind Dimensions- und Nicht-Dimensionsattribute durch unterschiedliche Symbole ersichtlich. Die Berechnung von Fakten ist im Schema integriert und die Granularität der Daten ist in den Hierarchiestufen ersichtlich. Die ADAPT-Methode spiegelt somit eine multidimensionale, betriebswirtschaftliche Sicht auf die Daten wieder. (+1)

Interoperabilität: Meines Wissens wird ADAPT noch nicht in CASE-Tools als Modellierungsmethode verwendet, so daß daraus automatisch logische Datenmodelle erstellt

werden können. Für das Programm Visio gibt es jedoch eine Symbolpalette, so daß zumindest die Diagrammerstellung vereinfacht wird. (-1)

Reife: Über die Methode ADAPT ist noch nicht sehr viel veröffentlicht worden, seien es wissenschaftliche Abhandlungen oder Praxisbeispiele. Dennoch denke ich, das ADAPT im Bezug auf die Darstellung multidimensionaler Strukturen eine ziemlich ausgereifte Methode ist. Soviel ich weiß gibt es jedoch keine Forschungen bezüglich der automatischen Umsetzung in logische Designs oder ähnliches, was der Weiterverarbeitung der Daten aus dieser Methode heraus dienlich wäre. (0)

Verständlichkeit: Bezüglich der Verständlichkeit des Konzepts der Methode kommt es, denke ich, darauf an, ob der Anwender bereits mit der multidimensionalen Sichtweise vertraut ist. Insbesondere wenn der Anwender ein MOLAP-Produkt kennt, dürfte es ihm sehr leicht fallen ADAPT zu verstehen. Aber auch Anwender, für die dieses Konzept neu ist, finden in ADAPT eine Methode, mit der sich der Sachverhalt strukturiert und somit verständlich darstellen läßt. (+1)

Erlernbarkeit: Ich denke, der Aufwand, um ADAPT zu erlernen, ist recht hoch. ADAPT ist zwar sehr verständlich, wenn das Konzept bekannt ist, aber die große Anzahl von Symbolen erfordert doch eine gewisse Merkfähigkeit. (0)

Bedienbarkeit: Wenn die Methode erlernt und verstanden ist, ist es leicht mit ADAPT zu arbeiten, da die Methode die multidimensionalen Sicht auf natürliche Weise widerspiegelt. (0)

Zeitverhalten: Aufgrund der vielen verschiedenen Symbole dauert es doch etwas länger, um mit ADAPT ein Modell zu erstellen. (-1)

Verbrauchsverhalten: Die zwar intuitiven, aber doch komplexen Strukturen benötigen sehr viel Platz auf dem Papier. (-1)

Analysierbarkeit: Für ADAPT gibt es keine softwaretechnische Unterstützung, um Mängel zu diagnostizieren und Fehler zu erkennen. (-1)

Modifizierbarkeit: Da sich im multidimensionalen Modell die Abhängigkeiten in Grenzen halten, ist es nicht schwer, an einem ADAPT-Modell Änderungen durchzuführen. (0)

Prüfbarkeit: Es gibt keine softwaretechnische Unterstützung zur Überprüfung eines geänderten Modells. (-1)

Anpaßbarkeit: Mit ADAPT lassen sich spätere logische Modelle sowohl als Star- als auch Galaxy-Architektur gut umsetzen. Die physische Abbildung eines ADAPT-Modells in ein MOLAP-Produkt ist besonders einfach, da hier dieselbe Struktur und Terminologie verwendet

wird. Die Umsetzung in relationale bzw. objektorientierte Technologie benötigt dagegen noch Zwischenschritte, um die jeweilige Struktur bzw. Terminologie abzubilden. (0)

Konformität: Da bereits bei der konzeptionellen Modellierung die Architektur (Star- oder Galaxy-Schema) festgelegt wird (zwar kann diese später leicht geändert werden), als auch Struktur und Terminologie einer bestimmten Technologie (MOLAP) impliziert wird, ist ADAPT nicht neutral gegenüber der folgenden Weiterverarbeitung. (-1)

Austauschbarkeit: Um das ADAPT-Modell in ein DF-, ER- oder UML-Modell umzusetzen, sind einige Anpassungen erforderlich. (0)

Fazit:

ADAPT ist eine umfassende Modellierungsmethode, die alle notwendigen Elemente zur Beschreibung einer OLAP-Anwendung enthält. Sie orientiert sich eher an der Sichtweise einer Führungskraft auf die Anwendung, denn auf die Entwicklersicht. So gesehen ist ADAPT benutzerfreundlich, wegen des Einarbeitungsaufwands aufgrund der vielen Elemente jedoch nur bedingt für Laien empfehlenswert. Die Nähe zu multidimensionalen Produkten ist unübersehbar. Für die Umsetzung in andere Technologien ist ein Zusatzaufwand einzukalkulieren. Ich denke gerade zur Dokumentation multidimensionaler betriebswirtschaftlicher Zusammenhänge ist ADAPT die ideale Methode.

6.3.2 DF

Dasselbe Modell mit der Dimensional-Fact-Methode dargestellt, sieht wie folgt aus. Die beiden Fakt-Schemata für Umsatz und Frachtkosten sind diesmal auf einer Seite untergebracht.

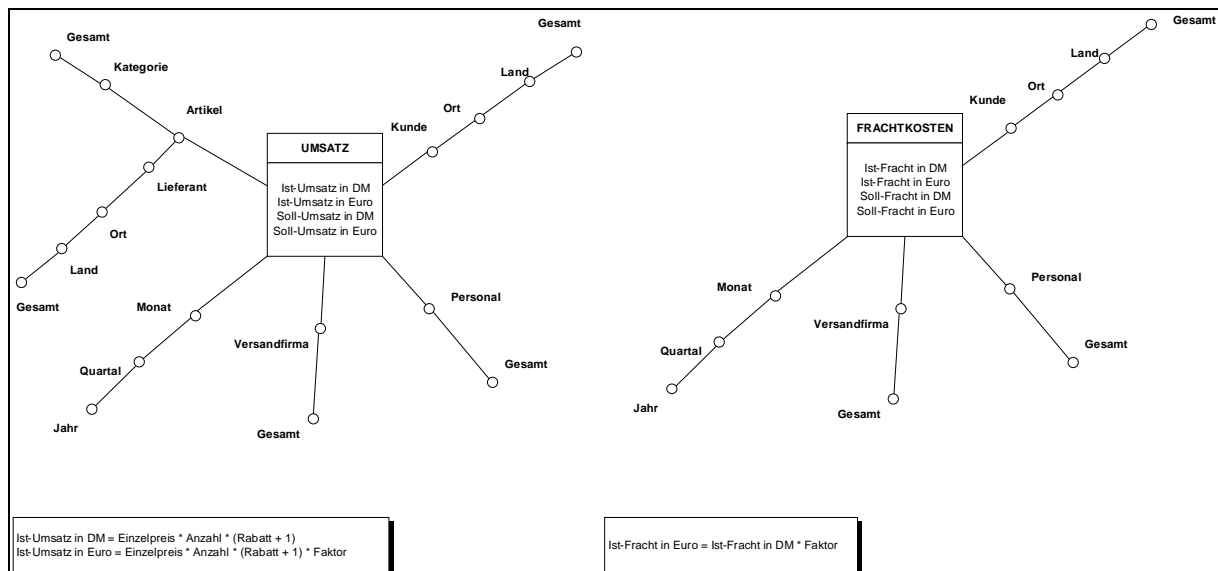


Abbildung 6.4: DF-Modelle Umsatz und Frachtkosten

Die auszuwertenden Kennzahlen sind in den Feldern mit den Bezeichnungen UMSATZ und FRACHTKOSTEN untergebracht. Davon gehen bei UMATZ die Dimensionen Artikel, Zeit, Versand, Personal und Kunde, und bei FRACHTKOSTEN die Dimensionen Zeit, Versand, Personal und Kunde als Zweige ab. Die Dimension Artikel ist in zwei Hierarchien aufgeteilt, zum einen die Kategorie-Hierarchie mit den Hierarchiestufen Artikel, Kategorie, Gesamt, zum anderen die Lieferanten-Hierarchie mit den Hierarchiestufen Artikel, Lieferant, Ort, Land, Gesamt, wobei sich Gesamt natürlich entspricht und auch gemeinsam hätte verwendet werden können. Die Zeit ist ebenfalls hierarchisch angeordnet und zwar in die Hierarchiestufen Monat, Quartal und Jahr. Die Dimension Versand ist auch hierarchisch mit den Hierarchiestufen Versandfirma und Gesamt. Ebenso die Dimension Personal mit den Hierarchiestufen Personal und Gesamt. Kunden werden in einer Kundenhierarchie mit den Ebene Kunde, Ort, Land und Gesamt geführt. Berechnungsvorschriften für Kennzahlen werden hier separat am unteren Bildrand ausgewiesen.

6.3.2.1 Von DF zum relationalen logischen Schema

Um aus einem DF-Modell ein relationales logisches Schema zu erstellen, sind folgende Schritte notwendig:

- Das Kennzahlenfeld kann als Fakttable mit den einzelnen Kennzahlen als Attribute übernommen werden. Die Berechnung der Kennzahlen erfolgt im DMBS und ist nicht im logischen Schema ersichtlich.
- Aus jedem Dimensionszweig wird eine Dimensionstabelle gebildet. Die Hierarchiestufen bilden dabei die einzelnen Attribute.
- Für jede Dimensionstabelle wird eine Primärschlüssel definiert.
- Die Fakttable erhält einen zusammengesetzten Schlüssel aus den Primärschlüsseln der Dimensionstabellen.
- Die Fakt- und Dimensionstabellen werden Star- oder Galaxy-Struktur angeordnet.
- Die Dimensionstabellen können als Snowflake-Schema dargestellt werden.

6.3.2.2 Von DF zum multidimensionalen logischen Schema

Um ein DF-Modell auf eine multidimensionale logische Struktur abzubilden, ist folgendes notwendig:

- Das Kennzahlenfeld wird nach Szenarien (Soll-/Istdaten) getrennt. Aus den einzelnen Kennzahlen wird eine Kennzahlendimension gebildet, aus den Soll-/Istdaten wird eine Szenariendimension gebildet.
- Die Berechnungsvorschriften können der jeweiligen Kennzahl zugeordnet werden (Rule).
- Aus den einzelnen Dimensionszweigen werden Dimensionen gebildet, deren Hierarchiestufen direkt als Attribute im multidimensionalen Schema abgelegt werden. Diese müssen eindeutige Werte besitzen.
- Der Dimensionszweig Zeit wird explizit als Zeitdimension definiert.

6.3.2.3 Von DF zum objektorientierten logischen Schema

Um von DF zu einem objektorientierten logischen Schema zu kommen, muß folgendes getan werden:

- Aus dem Kennzahlenfeld wird eine Faktklasse erstellt, welche die Kennzahlen des Kennzahlenfeldes als Attribute enthält. Die Berechnung der Kennzahlen erfolgt als Methode in dieser Klasse.
- Jeder Dimensionszweig wird als Klasse definiert. Die einzelnen Hierarchiestufen werden als Dimensionselemente zu Attribute dieser Klasse.
- Die Fakt- und Dimensionsklassen können als Star- oder Galaxy-Schema angeordnet werden.
- Die Dimensionsklassen können als Snowflake-Schema ausgeführt werden.

6.3.2.4 Subjektive Bewertung von DF

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Mit der Methode DF lassen sich, bei richtiger Anwendung, korrekte multidimensionale Modelle erstellen. (0)

Angemessenheit: DF zeigt, daß es auch mit einer pragmatischen Symbolik möglich ist, multidimensionale Datenmodelle zu erstellen. Im Modell sind die auszuwertenden Kennzahlen sowie die Dimensionen ersichtlich. Es ist erkennbar, ob Dimensionen hierarchisch oder nicht-hierarchisch sind. Die Hierarchiestufen sind angegeben und multiple Konsolidierungspfade darstellbar. Zeit wird als Dimension abgebildet (der sequentielle Charakter ist nicht erkennbar). Beziehungskardinalitäten müssen als bekannt vorausgesetzt werden. Dimensionsattribute sind explizit gekennzeichnet. Dimensionselemente verwenden jedoch das gleiche Symbol wie Hierarchiestufen. Die Herleitung von Kennzahlen ist mittels eines Glossars erkennbar. Die Granularität der Daten ist in der, der Kennzahl am nächsten liegenden Hierarchiestufe ersichtlich. Mit DF wird der multidimensionale Charakter der Daten erkennbar dargestellt. (0)

Interoperabilität: Da das Konzept von DF erst kürzlich veröffentlicht wurde, wird die Methode noch nicht von CASE-Tools unterstützt. (-1)

Reife: Die Methode DF ist noch so neu, daß sie vermutlich kaum in der Praxis erprobt ist. In dem Konzeptpapier gibt es jedoch bereits Regeln zur Umformung von ER nach DF, zur Überlagerung von Fakt-Schemata, sowie zur Entwicklung sogenannter Abfragemuster (query-patterns) um

OLAP-typische Abfragekonstrukte darzustellen. Zudem arbeiten die Autoren anscheinend an einer automatischen Umsetzung in ein logisches Schema. Diese Punkte geben also noch Anlaß zu weiteren Forschungen. (-1)

Verständlichkeit: Ich denke, das Konzept von DF ist sehr leicht zu verstehen, wenn der Benutzer mit der multidimensionalen Sichtweise vertraut ist. Für eine benutzerorientierte Modellierung ist die Symbolik jedoch zu sparsam. (+1)

Erlernbarkeit: DF besitzt nur wenige Elemente und zudem eine einfache Struktur, wodurch der Aufwand zum Erlernen der Methode klein gehalten wird. (+1)

Bedienbarkeit: Ich denke, mit der Methode DF ist es auf unkomplizierte Weise möglich, multidimensionale Datenmodelle zu erstellen. (+1)

Zeitverhalten: Mit der Methode DF lassen sich in kurzer Zeit konzeptionelle Datenmodelle erstellen. (+1)

Verbrauchsverhalten: Aufgrund der einfachen Symbolik und Struktur lassen sich kompakte Datenmodelle erstellen, die nur wenig Platz benötigen. (+1)

Analysierbarkeit: Für DF gibt es keine softwaretechnische Unterstützung, um Mängel zu diagnostizieren und Fehler zu erkennen. (-1)

Modifizierbarkeit: Bei DF lassen sich Modifikationen ebenfalls ohne großen Aufwand durchführen. (+1)

Prüfbarkeit: Es gibt keine softwaretechnische Unterstützung zur Überprüfung eines geänderten Modells. (-1)

Anpaßbarkeit: Mit DF lassen sich spätere logische Modelle als Star- oder auch Galaxy-Schemata realisieren. Die Abbildung auf eine bestimmte Technologie (relational, multidimensional, objektorientiert) benötigt sicher noch einigen Zusatzaufwand, zumal diesbezüglich noch keine Veröffentlichungen vorhanden sind. (0)

Konformität: Es wird zwar bei der Modellierung der Genauigkeit halber eine Galaxy-Struktur verwendet, die Änderung in ein logisches Star-Schema ist jedoch problemlos möglich. DF ist zudem neutral gegenüber der verwendeten Technologie, d.h., es wird kein relationaler, multidimensionaler oder objektorientierter Ansatz impliziert. (+1)

Austauschbarkeit: Um ein DF-Modell in ein ADAPT-, ER- oder UML-Modell umzusetzen sind Anpassungen notwendig. (-1)

Fazit:

DF ist eine konzeptionelle Modellierungsmethode, mit der sich multidimensionale Datenstrukturen gut abbilden lassen. Sie ist für den Anwender aufgrund der weniger ausdrucksstarken Symbolik nicht ganz so intuitiv wie ADAPT, aber sie ist neutral und der Einarbeitungsaufwand hält sich in

Grenzen. Zudem gibt es Ansätze der Weiterentwicklung, die vielversprechend sind. DF erscheint mir für den Bereich Prototyping aufgrund seiner Pragmatik ideal.

6.3.3 ER

Wie das konzeptionell zu modellierende Data Mart als multidimensionales Datenmodell mit der Entity-Relationship-Methode entworfen aussieht, ist nachfolgend zu sehen.

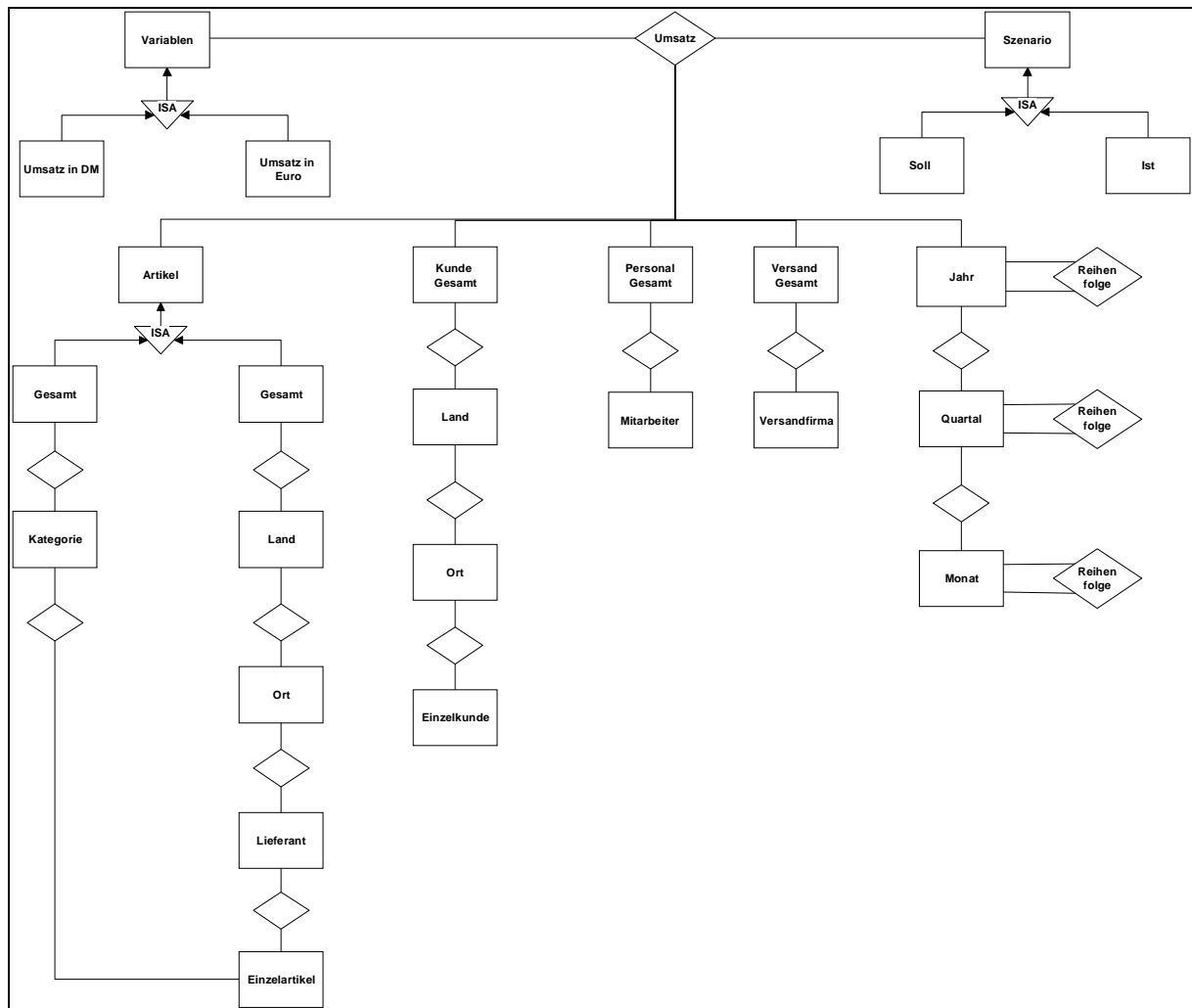


Abbildung 6.5: ER-Modell Umsatz

Die Dimensionsentitäten sind hier um die zentrale Beziehungsentität Umsatz angeordnet. Es bestehen 1:n-Beziehungen von Umsatz zu den einzelnen Dimensionen. Die Dimensionsentität mit der höchsten Granularität ist dabei der Entität Umsatz am nächsten. Hierarchiestufen werden durch darauffolgende Entitäten dargestellt, die jeweils in einer 1:n-Beziehung zueinander stehen. Die Dimensionen sind dieselben wie bei den vorigen Beispielen.

Für die Auswertung der Frachtkosten sieht das Schema ähnlich aus, nur fehlt die Dimension Artikel.

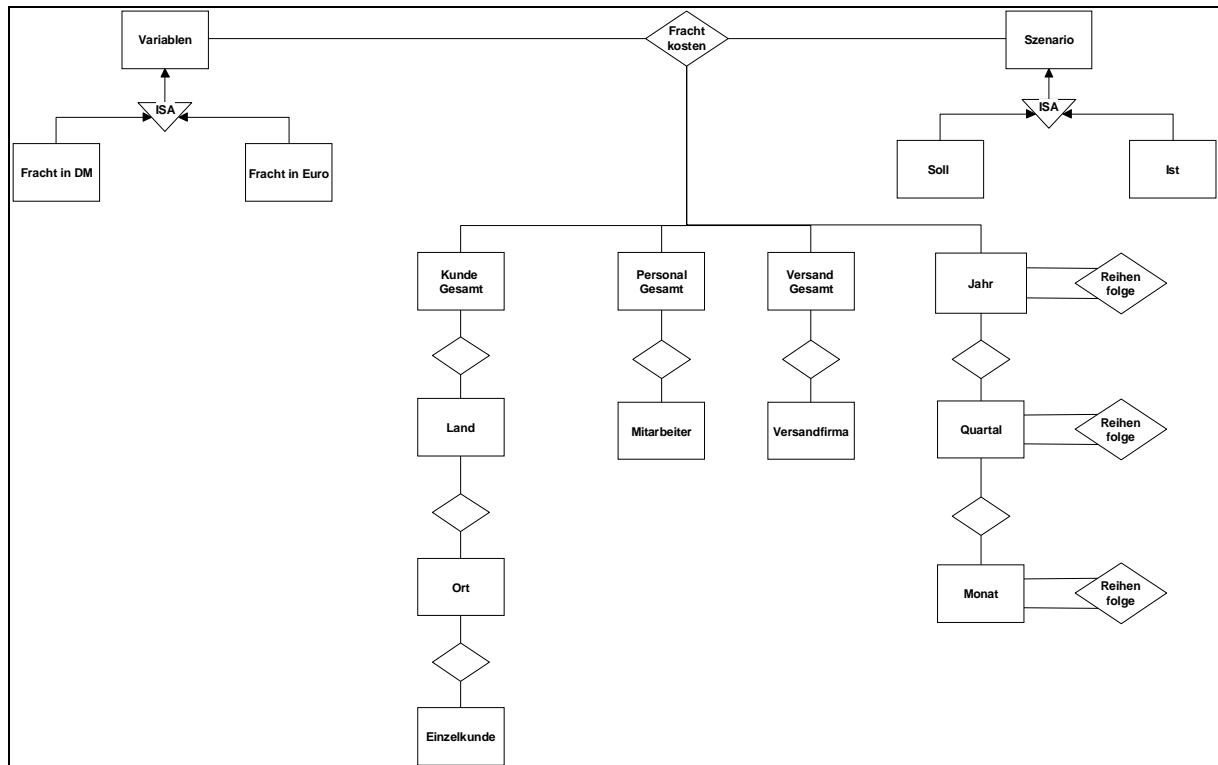


Abbildung 6.6: ER-Modell Frachtkosten

6.3.3.1 Von ER zum relationalen logischen Schema

Um aus einem ER-Modell ein relationales logisches Schema zu erstellen, sind folgende Schritte notwendig:

- Die Entitäten Szenario und Variablen werden zu einer Faktabelle zusammengefaßt, die Generalisierung aufgelöst und als Attribute gespeichert. Die Berechnung der Kennzahlen erfolgt im DMBS und ist nicht im logischen Schema ersichtlich.
- Aus den einzelnen Dimensionsentitäten werden Dimensionstabellen gebildet. Die Hierarchiestufen bilden dabei die einzelnen Attribute.
- Für jede Dimensionstabelle wird eine Primärschlüssel definiert.
- Die Faktabelle erhält einen zusammengesetzten Schlüssel aus den Primärschlüsseln der Dimensionstabellen.
- Die Fakt- und Dimensionstabellen können in Star- oder Galaxy-Struktur angeordnet werden.
- Die Dimensionstabellen können auch als Snowflake-Schema normalisiert gespeichert werden.

6.3.3.2 Von ER zum multidimensionalen logischen Schema

Um ein ER-Modell auf eine multidimensionale logische Struktur abzubilden, ist folgendes notwendig:

- Die Entität Variablen wird als Kennzahlendimension definiert, die die einzelnen Kennzahlen als Dimensionselemente enthält.
- Die Berechnungsvorschriften können der jeweiligen Kennzahl nicht zugeordnet werden, da sie im ER-Modell nicht ersichtlich sind.
- Die Entität Szenario wird als Versionendimension definiert, welche die einzelnen Versionsentitäten (Soll/Ist) als Dimensionselemente enthält.
- Die einzelnen Dimensionsentitäten-Zweige werden als Dimensionen definiert und deren Hierarchieentitäten direkt als Wertausprägungen im multidimensionalen Schema abgelegt. Diese müssen eindeutige Werte besitzen.
- Der Dimensionszweig Zeit wird explizit als Zeitdimension definiert.

6.3.3.3 Von ER zum objektorientierten logischen Schema

Um von ER zu einem objektorientierten logischen Schema zu kommen, muß folgendes getan werden:

- Aus der Variablenentität und der Szenarioentität wird eine Fakt-Klasse erstellt, welche die Kennzahlen in jeder Szenarioausprägung als Attribute enthält. Die Berechnung der Kennzahlen kann nicht als Methode in dieser Klasse erfolgen, da die Berechnung nicht ersichtlich ist.
- Jeder Dimensionszweig wird als Dimensionsklasse definiert. Die einzelnen Hierarchiestufen-Entitäten werden als Dimensionselemente zu Attribute dieser Klasse.
- Die Fakt- und Dimensionsklassen können als Star- oder Galaxy-Schema angeordnet werden.
- Die Dimensionsklassen können als Snowflake-Schema ausgeführt werden.

6.3.3.4 Subjektive Bewertung von ER

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Mit der Methode ER lassen sich keine, im Sinne von 3.2.5, korrekten multidimensionalen Modelle erstellen. (-1)

Angemessenheit: Mit ER lassen sich zwar Data Warehouse-ähnliche Beziehungsmodelle erstellen, diese besitzen jedoch Defizite. Die Unterscheidung zwischen Fakten und Dimensionen fällt schwer. Sie ist nur durch die Anordnung der Entitäten, jedoch nicht grafisch gekennzeichnet. Es ist schwierig Hierarchien zu erkennen, da die Beziehung dieselbe ist wie zwischen Fakten und Dimensionen. Nicht-hierarchische Beziehungen lassen sich als Generalisierung abbilden. Es lassen sich Hierarchien mit multipler Konsolidierung darstellen, wiederum mit der Einschränkung, daß sie als Hierarchien erkannt werden. Eine Zeitdimension ist über eine Reihenfolge-Beziehung sequentiell darstellbar. Zwischen Dimensionselementen und -attributen kann nicht unterschieden werden. Zudem kann die generelle Bezeichnung Attribut (=Element?) im ER-Modell zu Verwirrung führen. Die Herleitung von Kennzahlen ist nicht darstellbar. Die Granularität der Daten ist erkennbar. (-1)

Interoperabilität: Die Unterstützung von ER durch CASE-Tools ist gegeben. Durch die Verwendung von ER zur Modellierung von OLTP-Daten gibt es eine breite softwaretechnische Unterstützung. (+1)

Reife: ER ist eine sehr ausgereifte und erforschte Methode in Bezug auf die Modellierung von OLTP-Datenbanken. Bei der Modellierung von multidimensionalen Daten sind die Defizite erkannt worden. (+1)

Verständlichkeit: Für Anwender (Führungskräfte) ist der zugrundeliegende relationale Ansatz nicht verständlich [Kimb96]. (-1)

Erlernbarkeit: Aufgrund der wenigen Elemente und einfachen Strukturen ist es einfach, die Methode ER zu erlernen. (+1)

Bedienbarkeit: Prinzipiell ist es einfach mit ER zu arbeiten. Da beim multidimensionalen Modell die Beziehungskonstrukte aber bereits per Definition vorgegeben sind, ist es eigentlich überflüssig, jedesmal explizit Beziehungssymbole definieren zu müssen. (0)

Zeitverhalten: Der Aufwand für die Erstellung eines Modells ist auch aufgrund der wenigen Symbole erträglich. (0)

Verbrauchsverhalten: Die Modelle könnten kompakter sein, wenn das Beziehungssymbol wegfallen könnte (dann wäre es aber kein ER-Modell mehr). (0)

Analysierbarkeit: Für ER gibt es mit Sicherheit eine Reihe von Werkzeugen, die Mängel diagnostizieren und Fehler erkennen. (+1)

Modifizierbarkeit: Da die multidimensionale Sichtweise von einfacher Struktur ist, sind Änderungen auch mit ER leicht zu bewerkstelligen. (0)

Prüfbarkeit: Auch hier gibt es Werkzeuge, mit denen sich ein ER-Modell überprüfen läßt. (+1)

Anpaßbarkeit: Aus einem ER-Modell lassen sich sowohl logische Star- als auch Galaxy-Schemata erstellen. Die Umsetzung eines ER-Modells in ein relationales Datenbanksystem ist sehr einfach und auch definiert. Für die Verwendung eines objektorientierten bzw. multidimensionalen DBMS ist Mehraufwand erforderlich. (0)

Konformität: Auch hier läßt sich eine Galaxy-Struktur leicht in ein logisches Star-Schema wandeln und ist somit neutral. Bezüglich der Neutralität gegenüber der später eingesetzten Technologie, geht die Tendenz klar zum relationalen DBMS. (-1)

Austauschbarkeit: Soll das ER-Modell durch eine ADAPT-, DF- oder UML-Modell ersetzt werden, sind Änderungen erforderlich, damit Hierarchien besser erkennbar sind. (0)

Fazit:

ER eignet sich im Prinzip nicht zur konzeptionellen Modellierung multidimensionaler Datenstrukturen. Die Defizite wurden bereits genannt. Gerade für Laien ist die Darstellung nicht geeignet, eine dimensionale, hierarchische Struktur zu erkennen. Trotzdem wird ER häufig zur Modellierung angewandt, da eine Unterstützung durch CASE-Tools etc. gegeben ist. Bei einem relationalen Data Warehouse läßt sich ER sicher im Bereich der Entwicklung (jedoch nicht der Dokumentation) einsetzen.

6.3.4 UML

Wird das Beispiel mit der Unified Modeling Language modelliert, sieht die Darstellung wie folgt aus.

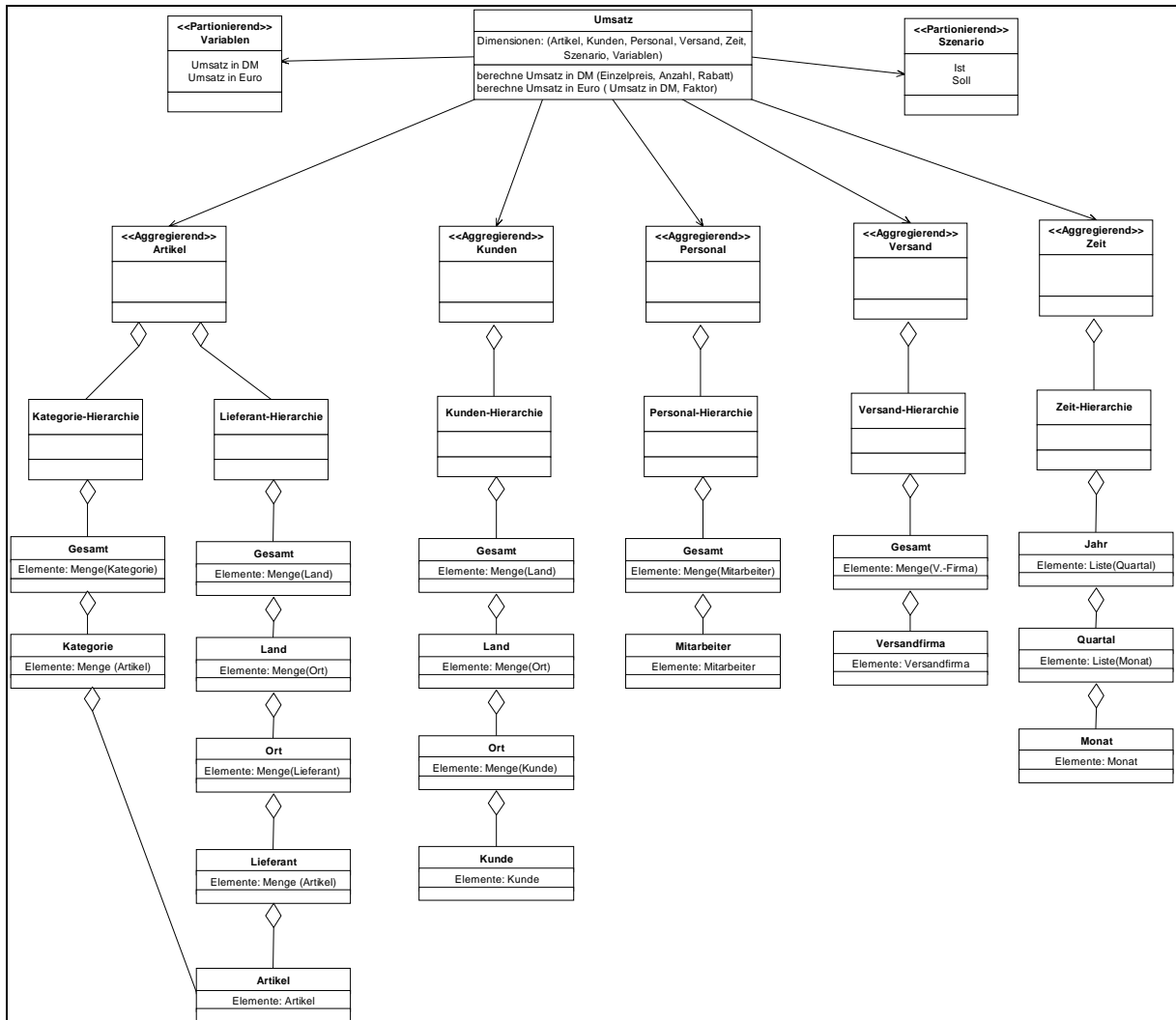


Abbildung 6.7: UML-Modell Umsatz

Von der Fakt-Klasse Umsatz geht eine gerichtete Assoziation zu den partitionierenden Dimensionen Szenario und Variablen und zu der aggregierenden Dimensionen Artikel, Kunde, Personal, Versand und Zeit. Mittels Aggregationsbeziehungen lassen sich die Hierarchien und Hierarchiestufen abbilden. Die verdichtete Klasse enthält dabei die Elemente der Unterklasse als Elemente-Menge, bei der Zeit-Hierarchie als Elemente-Liste. Die Berechnung der Kennzahlen wird über Methoden der Fakt-Klasse realisiert.

Das Modell zur Auswertung der Frachtkosten ist bis auf die Artikel-Hierarchie und die Berechnungen identisch.

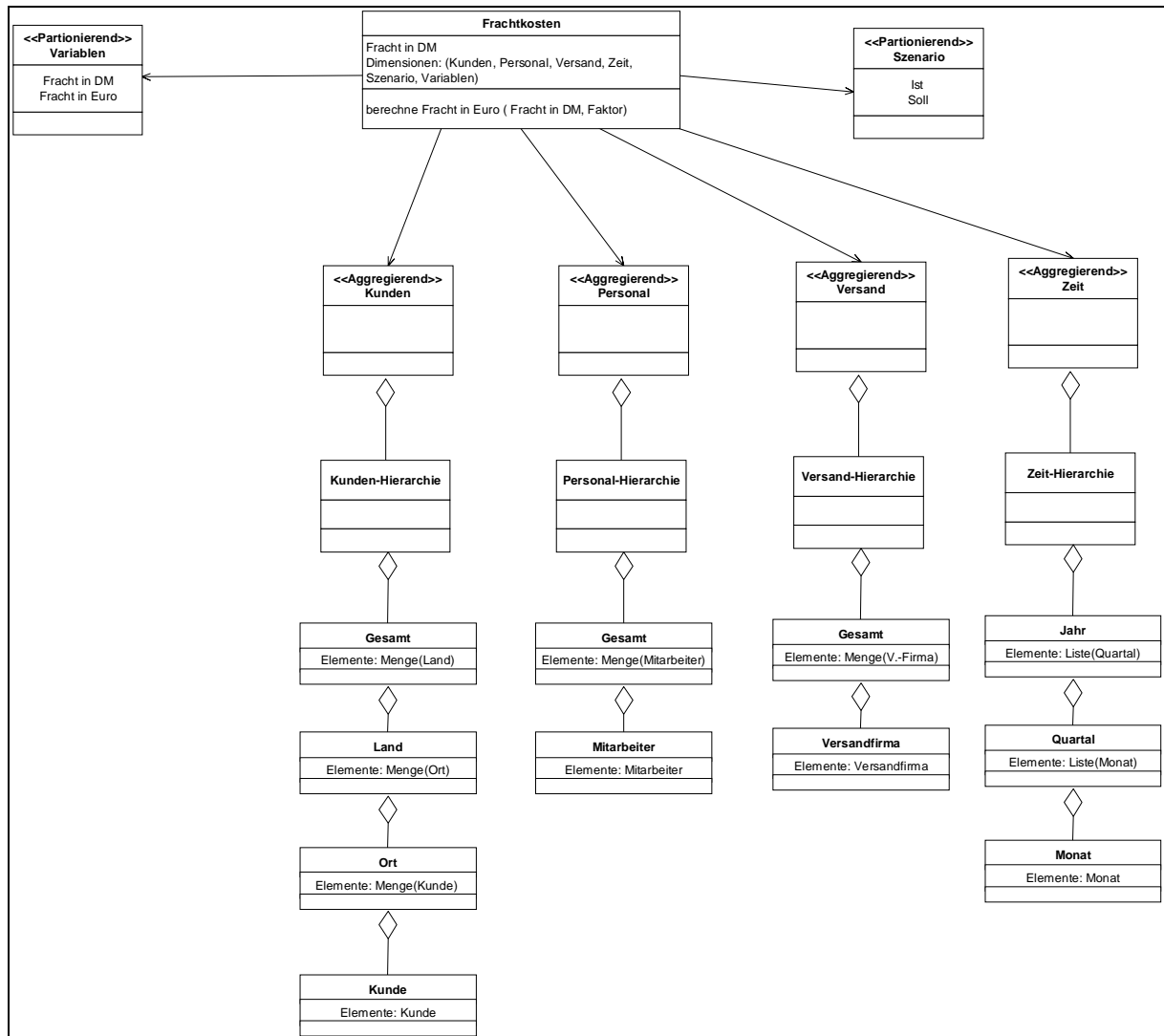


Abbildung 6.8: UML-Modell Frachtkosten

6.3.4.1 Von UML zum relationalen logischen Schema

Um aus einem UML-Modell ein relationales logisches Schema zu erstellen, sind folgende Schritte notwendig:

- Die Klassen Szenario und Variablen werden zu einer Faktabelle zusammengefaßt, die einzelnen Attribute zusammengefügt und als Kennzahlen darin abgelegt. Die Berechnung der Kennzahlen erfolgt im DMBS und ist nicht im logischen Schema ersichtlich.
- Aus den einzelnen Aggregationsklassen werden Dimensionstabellen gebildet. Attribute werden durch die einzelnen Hierarchieklassen-Elemente gebildet.
- Die Fakt- und Dimensionstabellen können in Star- oder Galaxy-Struktur angeordnet werden.

- Die Dimensionstabellen können auch als Snowflake-Schema normalisiert gespeichert werden.

6.3.4.2 Von UML zum multidimensionalen logischen Schema

Um ein UML-Modell auf eine multidimensionale logische Struktur abzubilden, ist folgendes notwendig:

- Die Klasse Variablen wird als Kennzahlendimension definiert, welche die einzelnen Kennzahlen-Attribute als Dimensionselemente enthält.
- Die Berechnungsmethoden werden der jeweiligen Kennzahl als Berechnungsvorschrift (Rule) zugeordnet.
- Die Klasse Szenario wird als Versionendimension definiert, welche die einzelnen Versionsattribute (Soll/Ist) als Dimensionselemente enthält.
- Die einzelnen Aggregationsklassen werden als Dimensionen definiert. Die Attribute der Hierarchieklassen werden direkt als Ausprägungen in den Dimensionen hierarchisch abgelegt. Diese müssen eindeutige Werte besitzen.
- Die Aggregationsklasse Zeit wird explizit als Zeitdimension definiert.

6.3.4.3 Von UML zum objektorientierten logischen Schema

Um von UML zu einem objektorientierten logischen Schema zu kommen, muß folgendes getan werden:

- Aus der Variablenklasse und der Szenarioklasse wird eine Fakt-Klasse erstellt, welche die Kennzahlen in jeder Szenarioausprägung als Attribute enthält. Die Berechnungen der Kennzahlen können über Methoden in dieser Klasse erfolgen.
- Jede Aggregationsklasse wird als Dimensionsklasse definiert. Die einzelnen Hierarchiestufen-Klassen werden als Dimensionselemente zu Attribute dieser Klasse.
- Die Fakt- und Dimensionsklassen können als Star- oder Galaxy-Schema angeordnet werden.
- Die Dimensionsklassen können als Snowflake-Schema ausgeführt werden

6.3.4.4 Subjektive Bewertung der UML

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Mit der UML lassen sich bei korrekter Anwendung multidimensionale Datenmodelle erstellen. (+1)

Angemessenheit: Mit der UML lassen sich speziellen Konstrukte eines Data Warehouse-Modells angemessen darstellen. Fakten und Dimensionen lassen sich anhand unterschiedlicher Beziehungen unterscheiden, somit sind die Dimensionen als solche erkennbar. Hierarchien lassen als Aggregationen mit einzelnen Hierarchiestufen abbilden. Multiple Konsolidierungspfade sind ebenfalls ersichtlich. Der sequentielle Charakter der Zeitdimension kommt zur Geltung. Zwischen Dimensionselementen und -attributen läßt sich nicht unterscheiden, mit einer Erweiterung der Notation von [Toto98] wäre jedoch auch dies möglich. Die Herleitung von Kennzahlen läßt sich über Methoden implementieren. Die Granularität der Daten ist erkennbar. Es lassen sich Szenarien- und Variablendimensionen darstellen. (+1)

Interoperabilität: Die UML wird von CASE-Werkzeugen unterstützt, mit denen sich automatisch relationale und objektorientierte Datenmodelle erstellen lassen (z.B. Oracle-DB und ObjectStore-DB via Rational Rose). (+1)

Reife: Die UML ist zwar noch relativ neu, besitzt jedoch einen ausgereiften Background, da sie auf bewährte OO-Methoden aufbaut. Im Bereich der multidimensionalen Modellierung gibt es sicherlich noch Forschungsbedarf. (0)

Verständlichkeit: Die UML ist für Laien schwer zu verstehen, da hierfür eine objektorientierte Denkweise unumgänglich ist. (-1)

Erlernbarkeit: Der Aufwand die UML zu erlernen ist hoch. Sie besteht aus vielen Symbolen und Diagrammtypen, die auf objektorientierten Konzepten aufbauen. (-1)

Bedienbarkeit: Aufgrund der Komplexität von UML, ist es nicht ganz einfach damit zu arbeiten. (0)

Zeitverhalten: Ein umfassendes UML-Modell zu erstellen dauert zwar etwas länger, aber man besitzt danach eine solide Basis für die weitere Entwicklung. (0)

Verbrauchsverhalten: UML-Modelle benötigen viel Raum, der durch die Bildung von Untermodellen jedoch strukturiert aufgeteilt werden kann. (-1)

Analysierbarkeit: Für die UML gibt es softwaretechnische Unterstützung bei der Modelldiagnose. (+1)

Modifizierbarkeit: Änderungen lassen sich mit der UML gut bewerkstelligen, da aufgrund des multidimensionalen Modells und des objektorientierten Ansatzes die Abhängigkeiten gering sind. (+1)

Prüfbarkeit: Für die UML gibt es softwaretechnische Überprüfungsmöglichkeiten. (+1)

Anpaßbarkeit: Mit einem UML-Modell läßt sich logisch sowohl ein Star- als auch ein Galaxy-Schema realisieren. Die Abbildung auf eine physisch objektorientierte Datenbank ist dabei am einfachsten. (0)

Konformität: Bezüglich der Architektur (Star-Galaxy) gilt hier dasselbe wie vorher. Als zu verwendende Technologie wird hier allein schon durch die Terminologie ein objektorientierter Ansatz impliziert. Auch UML kann deshalb nicht als neutral bezeichnet werden. (-1)

Austauschbarkeit: Um ein UML-Modell durch ein ADAPT-, DF- oder ER-Modell zu ersetzen sind nur bei DF größere ansonsten geringfügige Änderungen durchzuführen. (0)

Fazit:

UML ist eine umfangreiche Modellierungsnotation, mit der sich auch komplexe multidimensionale Modelle erstellen lassen. Durch den objektorientierten Ansatz ist sie sehr flexibel, wodurch den Modellierungsmöglichkeiten durch die Verwendung komplexer Datentypen eigentlich keine Grenzen gesetzt sind. Für den Laien (Führungskraft) ist sie jedoch aus dem gleichen Grund nicht verständlich genug und deshalb nicht empfehlenswert. Für die professionelle Data Warehouse-Entwicklung bietet UML aber mit Sicherheit die beste Grundlage.

6.3.5 Vergleich der konzeptionellen Methoden

Kriterium	ADAPT			DF			ER			UML		
	Erfüllungsgrad			Erfüllungsgrad			Erfüllungsgrad			Erfüllungsgrad		
	-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1
Richtigkeit			●		●		●					●
Angemessenheit			●		●		●					●
Interoperabilität	●			●					●			●
Reife		●		●					●		●	
Verständlichkeit			●			●	●			●		
Erlernbarkeit		●				●			●	●		
Bedienbarkeit		●				●		●			●	
Zeitverhalten	●					●		●			●	
Verbrauchsverhalten	●					●		●		●		
Analysierbarkeit	●			●					●			●
Modifizierbarkeit		●				●		●				●
Prüfbarkeit	●			●					●			●
Anpassbarkeit		●			●			●			●	
Konformität	●					●	●			●		
Austauschbarkeit		●		●				●			●	

Abbildung 6.9: Vergleich der konzeptionellen Methoden

Generell lassen sich die vier Methoden in zwei Kategorien unterteilen. ER und UML bauen auf den klassische Ansatz von Objekten (Entitäten) und deren Beziehung zueinander. Da bei der multidimensionalen Modellierung die Beziehungen aber eigentlich keine große Rolle mehr spielen, da sie durch die Star- bzw. Galaxy-Struktur bereits vorab definiert sind, legen die speziell für Data Warehouse entwickelten Methoden ADAPT und DF mehr Wert auf die benutzerfreundliche, verständliche Darstellung der Multidimensionalität mit ihren Elementen. Leider sind sie jedoch noch nicht ausreichend erforscht und die softwaretechnische Unterstützung läßt im Gegensatz zu ER und UML zu wünschen übrig.

Als Empfehlung würde ich ADAPT vorschlagen, um das Data Warehouse aus Benutzersicht zu modellieren und zu dokumentieren und UML, um das Data Warehouse aus Entwicklersicht darzustellen und softwaretechnisch damit zu arbeiten.

6.4 Logische Architekturen

Das konzeptionelle Datenmodell muß nun in eine logische Datenstruktur umgesetzt werden. Das Beispiel wird nun auf alle drei logischen Architekturen abgebildet, diese werden bewertet und miteinander verglichen.

6.4.1 Star-Schema, Hypercube

Das Star-Schema, das in Ansichten meist als relationales Modell aufgebaut ist, entspricht in seiner Struktur einem multidimensionalen Hypercube und läßt sich ebenfalls als objektorientiertes Schema aufbauen.

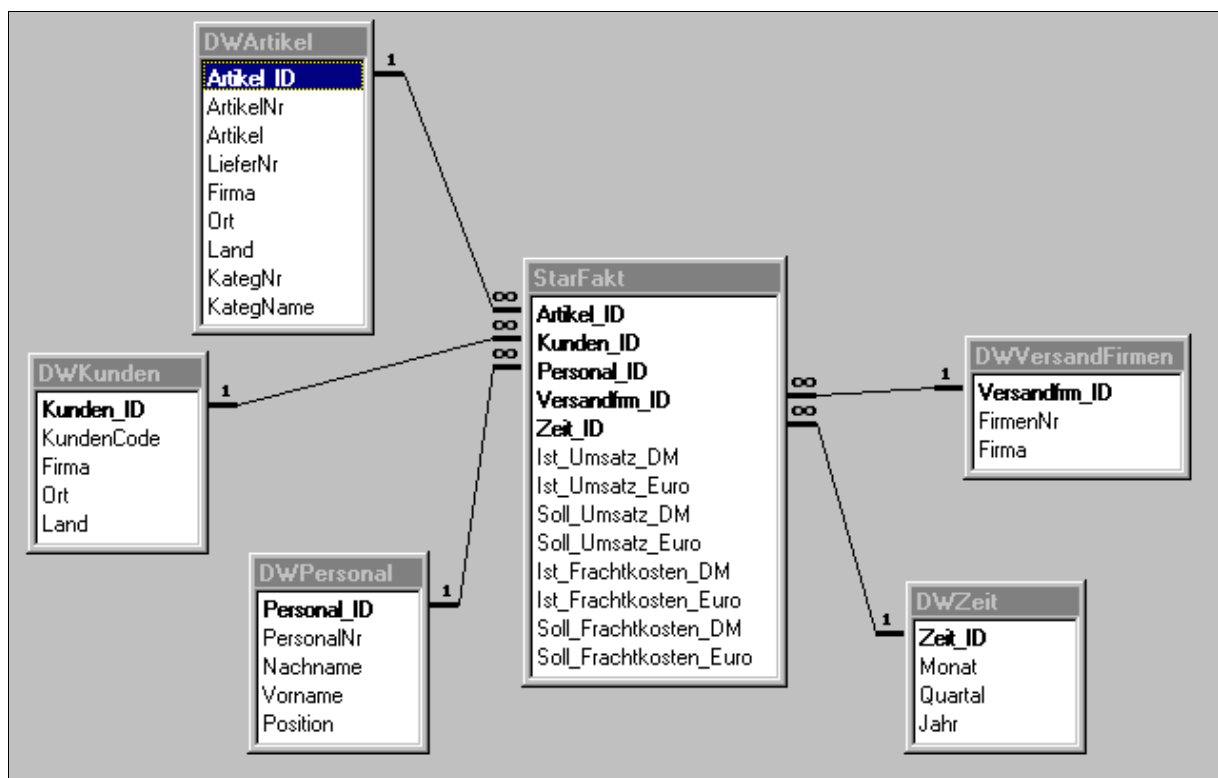


Abbildung 6.10: Star-Schema

Die Dimensionen sind beim Star-Schema sternförmig um eine Fakttable angeordnet, welche alle auszuwertenden Kennzahlen enthält. Die Dimensionen sind in denormalisierter Form

gespeichert, d.h., alle Elemente und Attribute einer Dimension werden in einer Dimensionstabelle gehalten.

6.4.1.1 Subjektive Bewertung des Star-Schemas

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Sofern nicht alle Kennzahlen den gleichen Dimensionen zugehörig sind (was so gut wie nie der Fall ist), lassen sich mit dem Star-Schema keine korrekten Datenmodelle erstellen. Es kann beim Star-Schema zu Kennzahl-Dimension-Kombinationen kommen, die keine korrekten Werte (hier z.B. Artikel-Frachtkosten) ergeben. (-1)

Reife: Durch die Star-Schema-Architektur sind Fehlzustände möglich. Diese Unrichtigkeiten können nur in Kauf genommen werden, wenn inkorrekte Kombinationen bereits bei der Abfrageerstellung definiert und dadurch ausgeschlossen werden können. (0)

Verständlichkeit: Aufgrund seiner einfachen Struktur ist das Konzept des Star-Schemas sehr gut zu verstehen. (+1)

Zeitverhalten: Der Aufwand zur Erstellung eines Star-Schemas ist gering (+1). Das Zeitverhalten bei der Abfragebearbeitung ist aufgrund der denormalisierten Struktur zwar als gut, aber wegen der immens großen Faktttabelle nicht als sehr gut zu bewerten. (0)

Verbrauchsverhalten: Ein Star-Schema benötigt wenig Platz auf dem Papier und ist somit übersichtlich (0). Die Architektur selbst benötigt jedoch sehr viel Speicherplatz. Insbesondere die Faktttabelle wird beim Star-Schema sehr groß. Hier beispielsweise 77 Artikel x 78 Kunden x 9 Mitarbeiter x 3 Versandfirmen x 12 Monate x 13 Felder x ca. 4 Byte = 101 MB. Auch die Dimensionstabellen besitzen aufgrund der denormalisierten Struktur keine optimale Speicherhaltung. (-1)

Modifizierbarkeit: Das Star-Schema läßt sich aufgrund seiner einfachen Struktur leicht an Änderungen anpassen und ist problemlos zu warten. (0)

Anpaßbarkeit: Das Star-Schema läßt sich ohne Probleme als relationales, multidimensionales und als objektorientiertes Modell realisieren. (+1)

Fazit:

Das Star-Schema ist ein leicht zu verstehendes logisches Architekturkonzept, das viel Speicherplatz benötigt, angemessen schnell ist und einfach zu implementieren und zu warten ist. Inkorrekte Werte sind möglich, aber extern verhinderbar.

6.4.2 Galaxy-Schema, Multicube

Das meist in relationaler Form dargestellte Galaxy-Schema entspricht dem multidimensionalen Multicube und läßt sich auch objektorientiert realisieren.

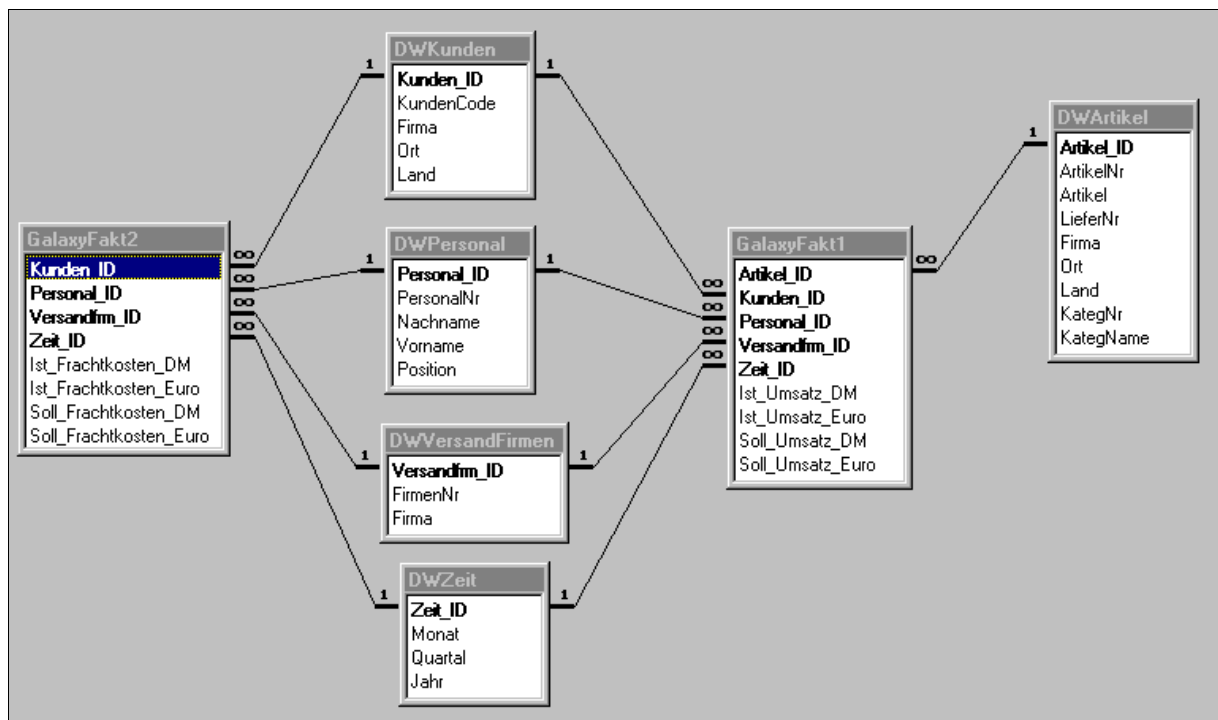


Abbildung 6.11: Galaxy-Schema

Beim Galaxy-Schema werden Kennzahlen mit unterschiedlichen Dimensionen in verschiedenen Faktentabelle zusammengefaßt. Man erhält dadurch mehrere durch gleiche Dimensionen verbundene Star-Schemata. Die Dimensionen werden ebenfalls denormalisiert in Dimensionstabellen gespeichert.

6.4.2.1 Subjektive Bewertung des Galaxy-Schemas

Metrik: gut (+1), mittel (0), schlecht (-1)

Richtigkeit: Mit dem Galaxy-Schema lassen sich korrekte Datenmodelle erstellen. (+1)

Reife: Dadurch, daß eine Kennzahl nur mit den dazugehörigen Dimensionen kombiniert wird, sind Fehlzustände nicht möglich. (+1)

Verständlichkeit: Das Galaxy-Schema ist aufgrund seiner komplexeren Struktur schwieriger zu verstehen. (0)

Zeitverhalten: Der Zeitaufwand, um ein Modell in Galaxy-Architektur zu erstellen, ist größer, als ein Star-Schema aufzubauen (0). Das Zeitverhalten bei der Abfragebearbeitung ist aufgrund der denormalisierten Form und den kleineren Fakttabellen sehr gut. (+1)

Verbrauchsverhalten: Ein Galaxy-Schema kann große Ausmaße annehmen und ist wenig übersichtlich (0). Ein Modell in Galaxy-Architektur benötigt weniger Speicherplatz als ein Star-Schema. Die Fakttabellen sind kleiner, da weniger Dimensionen daran beteiligt sind. Hier beispielsweise (77 Artikel x 78 Kunden x 9 Mitarbeiter x 3 Versandfirmen x 12 Monate x 9 Felder x 4Byte) + (78 Kunden x 9 Mitarbeiter x 3 Versandfirmen x 12 Monate x 8 Felder x 4 Byte) = 70,8 MB. Die Dimensionstabellen besitzen aber auch hier aufgrund der denormalisierten Struktur keine optimale Speicherhaltung. (+1)

Modifizierbarkeit: Das Galaxy-Schema ist aufgrund seiner komplizierteren Struktur schwieriger zu ändern und aufwendiger zu warten. (-1)

Anpaßbarkeit: Das Galaxy-Schema läßt sich ohne Probleme als relationales, multidimensionales (falls vom Produkt unterstützt) und als objektorientiertes Modell realisieren. (+1)

Fazit:

Das Galaxy-Schema ist ein komplexes logisches Architekturkonzept, das im Vergleich wenig Speicherplatz benötigt und ein gutes Abfrageverhalten besitzt. Es liefert korrekte Werte, ist aber aufgrund der komplexen Struktur nicht ganz einfach zu warten und zu implementieren.

6.4.3 Snowflake-Schema

Für relationale und objektorientierte Implementierungen gibt es noch die Möglichkeit, eine Architektur als sogenanntes Snowflake-Schema aufzubauen.

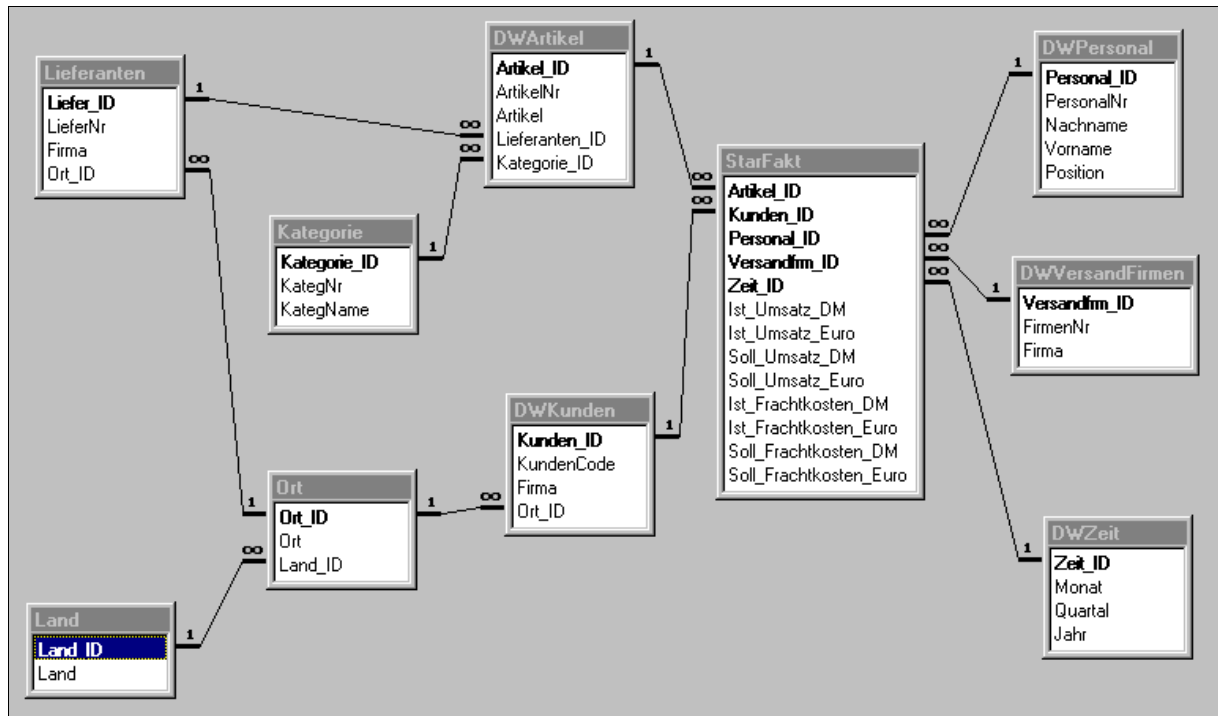


Abbildung 6.12: Snowflake-Schema

Beim Snowflake-Schema kann die Anordnung zwischen Dimensionen und Fakten prinzipiell als Star- oder Galaxy-Schema ausgeführt werden (hier Star-Schema). Die Dimensionen sind jedoch zusätzlich normalisiert gespeichert.

6.4.3.1 Subjektive Bewertung des Snowflake-Schemas

Metrik: gut (+1), mittel (0), schlecht (-1)

S/S = Snowflake-Schema in Star-Architektur, S/G = Snowflake-Schema in Galaxy-Architektur

Richtigkeit: Beim Snowflake-Schema hängt die Bewertung der Richtigkeit davon ab, ob die Dimension-Fakt-Beziehung als Star- oder Galaxy-Schema ausgeführt wurde. S/S (-1), S/G (+1)

Reife: Die Frage, ob Fehlzustände möglich sind, hängt wiederum von der Dimension-Fakt-Beziehung ab (Star- oder Galaxy-Anordnung). S/S (0), S/G (+1)

Verständlichkeit: Das Snowflake-Schema ist aufgrund seiner normalisierten Struktur schwieriger zu verstehen. Durch die Normalisierung ist auch kein flüssiges Navigieren im Datenbestand mehr möglich [Kimb96]. S/S(0), S/G(-1)

Zeitverhalten: Der Zeitaufwand, um ein Modell in Snowflake-Architektur zu erstellen, ist groß, besonders, wenn die Dimension-Fakt-Beziehung als Galaxy-Schema ausgeführt wurde S/S (0), S/G (-1). Das Zeitverhalten bei der Abfragebearbeitung ist aufgrund der normalisierten Form mit schlecht zu bewerten. Verknüpfungen müssen hier über mehrere Tabellen durchgeführt werden. S/S (-1), S/G (-1)

Verbrauchsverhalten: Ein Snowflake-Schema ist aufgrund der normalisierten Struktur besonders in Galaxy-Architektur sehr unübersichtlich S/S (0), S/G (-1). Ein Modell in Snowflake-Architektur benötigt weniger Speicherplatz, als das entsprechende denormalisierte Star- oder Galaxy-Schema.. Dennoch ist die Speicherplatzersparnis nicht allzu hoch, da die Dimensionstabellen im Vergleich zur Faktentabelle sowieso klein sind. S/S (0), S/G (+1)

Modifizierbarkeit: Das Snowflake-Schema ist aufgrund seiner normalisierten Struktur einfacher zu warten, als die entsprechende denormalisierte Variante. S/S (+1), S/G (0)

Anpaßbarkeit: Das Snowflake-Schema läßt sich ohne Probleme als relationales oder objektorientiertes Modell realisieren. Die Umsetzung in multidimensionale Technologie ist meines Wissens nicht möglich. S/S (0), S/G (0)

Fazit:

Das Snowflake-Schema ist ein komplexes logisches Architekturkonzept, das im Vergleich weniger Speicherplatz benötigt, als die entsprechende denormalisierte Variante. Durch die Normalisierung verschlechtert sich jedoch das Abfrageverhalten drastisch. Durch die normalisierte Struktur ist das Snowflake-Schema einfach zu warten.

6.4.4 Vergleich der logischen Architekturen

Kriterium	Star			Galaxy			Snowflake/Star			Snowfl/Galaxy		
	Erfüllungsgrad			Erfüllungsgrad			Erfüllungsgrad			Erfüllungsgrad		
	-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1
Richtigkeit	●					●	●					●
Reife		●				●		●				●
Verständlichkeit			●		●			●		●		
Zeitverhalten												
Modellerstellung			●		●			●		●		●
Abfrageverhalten		●				●	●			●		
Verbrauchsverhalten												
Modellerstellung			●		●			●		●		●
Speicherbedarf	●					●		●		●		●
Modifizierbarkeit		●		●					●		●	
Anpassbarkeit			●			●		●			●	

Abbildung 6.13: Vergleich der logischen Architekturen

Man sieht hier, daß das Galaxy-Schema eindeutig am besten abschneidet. Es ist zwar komplexer aufgebaut und deshalb auch nicht so einfach zu warten, aber die in der Praxis wichtige Kriterien-Kombination von Richtigkeit, Geschwindigkeit und geringer Speicherbedarf ist hier gegeben. Bei einfachen Modellen oder geringerem Datenumfang ist sicher auch ein Star-Schema zu empfehlen, das wegen seiner Verständlichkeit Vorteile zu verbuchen hat. Hier müssen jedoch die möglichen Fehlzustände extern abgefangen werden. Das Snowflake-Schema sollte eigentlich nur angewandt werden, wenn eine einfache Wartbarkeit wichtig ist. Die dabei zu erreichende (eher geringe) Speicherplatzersparnis wird zudem durch ein schlechtes Abfrageverhalten bestraft.

6.5 Vom OLTP-Modell zum logischen relationalen DW-Modell

Unter Berücksichtigung der konzeptionellen Modellierungen und logischen Architekturen soll nun das gewählte Beispiel aus dem OLTP-System praktisch in ein Data Warehouse transformiert werden. Da das gewählte Beispiel bereits als MS-Access-Datenbank modelliert ist, erfolgt die Umwandlung in das Galaxy-Schema ebenfalls auf diesem System, wobei später eine Übernahme nach Oracle 7 geschehen soll.

Es wurde festgelegt, daß nur die Daten in das Data Mart aufgenommen werden, die auch über die Bestell- und die Bestelldetail-Tabelle abgefragt werden können (also z.B. werden nur die Kunden übernommen, die auch tatsächlich etwas bestellt haben).

Daraus ergibt sich folgende Vorgehensweise zur Erstellung des Galaxy-Schemas:

1.) Zunächst werden die beiden Dateien **Bestell** und **BestDtl** zusammengefaßt, das sie die Grundlage (Datenherkunft) unseres Data Warehouse bieten. Rein operative Daten werden hier bereits entfernt (Kundenanschrift) (Abfrage1 in Access).

```
SELECT DISTINCTROW Bestell.KundenCode, Bestell.PersonalNr, Bestell.VersndFirm,  
Bestell.BestellDat, Bestell.FrachtKost, BestDtl.BestellNr, BestDtl.ArtikelNr, BestDtl.EinzPreis,  
BestDtl.Anzahl, BestDtl.Rabatt INTO gesamtBestellung  
FROM Bestell INNER JOIN BestDtl ON Bestell.BestellNr = BestDtl.BestellNr;
```

Diese Zusammenfassung wird als Tabelle **gesamtBestellung** mit 2820 Datensätzen gespeichert.

2.) Um die Datenflut der später zu erstellenden Fakttabellen etwas zu reduzieren, werden aus den Gesamtbestellungen (über 3 Jahre) die Bestellungen für ein Jahr (1992) extrahiert, die für das Beispiel genügen müssen (Abfrage gesamtBestellungen92 in Access).

```
SELECT gesamtBestellung.KundenCode, gesamtBestellung.PersonalNr,  
gesamtBestellung.VersndFirm, gesamtBestellung.BestellDat, gesamtBestellung.FrachtKost,  
gesamtBestellung.BestellNr, gesamtBestellung.ArtikelNr, gesamtBestellung.EinzPreis,  
gesamtBestellung.Anzahl, gesamtBestellung.Rabatt INTO gesamtBestellungen92  
FROM gesamtBestellung  
WHERE (((Year([BestellDat])=1992));
```

Diese Auswahl wird als Tabelle **gesamtBestellungen92** mit 786 Datensätzen gespeichert.

3.) Aus der Tabelle **gesamtBestellungen92** und den Stammdateien **Artikel**, **Kunden**, **Personal** und **Firmen** (Versandfirmen) werden nun die Daten für die einzelnen Dimensionen extrahiert. Auch hier werden nur die zur Auswertung benötigten Felder selektiert.

Dimension Artikel:

(Abfrage2 in Access)

```
SELECT DISTINCTROW Artikel.ArtikelNr, Artikel.LieferNr, Artikel.KategNr, Artikel.Artikel,
Artikel.LagerBest INTO DWArtikel
FROM Artikel, gesamtBestellungen92
WHERE (((Artikel.ArtikelNr)=[gesamtBestellungen92].[ArtikelNr]));
```

Diese Zusammenfassung wird als Tabelle **DWArtikel** mit 77 Datensätzen gespeichert.

Dimension Kunden:

(Abfrage3 in Access)

```
SELECT DISTINCTROW Kunden.KundenCode, Kunden.Firma, Kunden.Ort, Kunden.Land INTO
DWKunden
FROM Kunden, gesamtBestellungen92
WHERE Kunden.KundenCode = gesamtBestellungen92.KundenCode;
```

Diese Zusammenfassung wird als Tabelle **DWKunden** mit 78 Datensätzen gespeichert.

Dimension Personal:

(Abfrage4 in Access)

```
SELECT DISTINCTROW Personal.PersonalNr, Personal.Nachname, Personal.Vorname,
Personal.Position INTO DWPersonal
FROM Personal, gesamtBestellungen92
WHERE Personal.PersonalNr = gesamtBestellungen92.PersonalNr;
```

Diese Zusammenfassung wird als Tabelle **DWPersonal** mit 9 Datensätzen gespeichert.

Dimension Versand:

(Abfrage5 in Access)

```
SELECT DISTINCTROW Firmen.FirmenNr, Firmen.Firma INTO DWVersand
FROM Firmen, gesamtBestellungen92
```

```
WHERE Firmen.FirmenNr = gesamtBestellungen92.VersndFirm;
```

Diese Zusammenfassung wird als Tabelle **DWVersand** mit 3 Datensätzen gespeichert.

4.) Als nächstes wird die Datei **DWArtikel** denormalisiert, indem sie mit den Dateien **Liefernt** und **Kategor** zusammengefaßt wird, welche Lieferanten- und Kategoriedaten enthalten, nach denen später verdichtet werden kann.

(Abfrage6 in Access)

```
SELECT DISTINCTROW DWArtikel.ArtikelNr, DWArtikel.Artikel, DWArtikel.LieferNr, Liefernt.Firma,
Liefernt.Ort, Liefernt.Land, DWArtikel.KategNr, Kategor.KategName, DWArtikel.LagerBest INTO
DWArtikelHierarchie
FROM DWArtikel, Liefernt, Kategor
WHERE DWArtikel.LieferNr = Liefernt.LieferNr AND DWArtikel.KategNr = Kategor.KategNr;
```

Diese Zusammenfassung wird als Tabelle **DWArtikelHierarchie** mit 77 Datensätzen gespeichert.

5.) In die Dimensionstabellen **DWArtikelHierarchie**, **DWKunden**, **DWPersonal** und **DWVersand** werden nun separate Data Warehouse-Schlüssel-Felder (Artikel_ID, Kunden_ID, Personal_ID, Versand_ID) eingefügt, die als Primärschlüssel definiert und über die Felddefinition AutoWert erzeugt werden. Diese müssen jedoch nach der Erzeugung wieder auf Felddatentyp Zahl zurückgesetzt werden, da es sonst zu Problemen bei Verknüpfungen kommt.

6.) Jetzt wird eine Tabelle **DWZeit** definiert, in welche die monatlichen Zeit-Perioden (1/1/1992, 2/1/1992, 3/1/1992...) eingegeben werden.

```
CREATE TABLE DWZeit (Zeit_ID Number, Monat Number, Quartal Number, Jahr Number);
```

Anschließend wird mit der Bildung der Fakttable für die Kennzahl Umsatz begonnen. Dazu werden zunächst die Dimensionen bestimmt, mit denen die Kennzahl Umsatz verknüpft werden soll:

7a.) Der Umsatz wäre theoretisch über alle fünf Dimensionen (Artikel, Kunden, Personal, Versand und Zeit) additiv. Dies würde jedoch für das Beispiel eine zu große Fakttable ergeben, da das karthesische Produkt dieser fünf Dimensionen $77 \times 78 \times 9 \times 3 \times 12 =$

1.945.944 Datensätze ergibt. Ich beschränke mich deshalb hier auf die Dimensionen Artikel, Kunden und Zeit, was letztendlich 72.072 Datensätze ergeben müßte.

Die Primärschlüssel der drei Dimensionstabellen, die benötigten Kennzahl Umsatz und die Tabelle **gesamtBestellungen92** werden zunächst zu einer Tabelle **Faktwerte** zusammengesetzt, welche die getätigten Umsätze (786 Datensätze) enthält (Abfrage7 in Access).

```
SELECT DISTINCTROW DWZeit.Zeit_ID, DWArtikelHierarchie.Artikel_ID, DWKunden.Kunde_ID,
[EinzPreis]*[Anzahl]*([Rabatt]+1) AS Umsatz INTO Faktwerte
FROM DWKunden, DWPersonal, DWVersand, DWZeit, gesamtBestellungen92, DWArtikelHierarchie
WHERE (((DWArtikelHierarchie.ArtikelNr)=[gesamtBestellungen92].[ArtikelNr]) AND
((DWKunden.KundenCode)=[gesamtBestellungen92].[KundenCode]) AND
((DWZeit.Monat)=Month([gesamtBestellungen92].[Bestelldat])));
```

8a.) Um nun das karthésische Produkt für die Fakttabelle zu erhalten, werden die Primärschlüssel der Dimensionstabellen bedingungslos zu einer weiteren Tabelle **Leerfakt** verknüpft und ein leeres Umsatzfeld eingefügt (Abfrage8 in Access).

```
SELECT DISTINCTROW DWZeit.Zeit_ID, DWArtikelHierarchie.Artikel_ID, DWKunden.Kunde_ID, 0
AS Umsatz INTO Leerfakt
FROM DWArtikelHierarchie, DWKunden, DWZeit;
```

Die Tabelle **Leerfakt** enthält nun 72.072 Datensätze.

9a.) Die Tabelle **Faktwerte** enthält noch einige doppelte Datensätze, so daß die Umsätze zunächst in einer weiteren Tabelle **FaktwerteKum** (778 Datensätze) kumuliert werden (Abfrage10 in Access).

```
SELECT DISTINCTROW Faktwerte.Zeit_ID, Faktwerte.Artikel_ID, Faktwerte.Kunde_ID,
Sum(Faktwerte.Umsatz) AS Umsatz INTO FaktwerteKum
FROM Faktwerte
GROUP BY Zeit_ID, Artikel_ID, Kunde_ID;
```

10a.) Zuletzt muß nun die Tabelle **FaktwerteKum** in die Tabelle **Leerfakt** eingefügt werden (Abfrage9 in Access).

```
SELECT DISTINCTROW Leerfakt.Zeit_ID, Leerfakt.Artikel_ID, Leerfakt.Kunde_ID,
FaktwerteKum.Umsatz INTO UmsatzFakt
FROM FaktwerteKum RIGHT JOIN Leerfakt ON (FaktwerteKum.Kunde_ID = Leerfakt.Kunde_ID)
AND (FaktwerteKum.Zeit_ID = Leerfakt.Zeit_ID) AND (FaktwerteKum.Artikel_ID =
Leerfakt.Artikel_ID);
```

11a.) Das Ergebnis ist die endgültige Fakttable **UmsatzFakt** (72.072 Datensätze), die mit abschließender Definition des zusammengesetzten Schlüssels Zeit ID, Artikel ID, Kunde ID vollständig ist. Die Fakttable **UmsatzFakt** ließe sich bei Bedarf um weitere Kennzahlen erweitern, wie beispielsweise den Umsatz in Euro (Umsatz * Umrechnungskurs) oder mit Umsatz-Sollwerten aus der Planung.

Nun kann mit der Erstellung der zweiten Fakttable begonnen werden. Diese Tabelle soll die Werte für die Frachtkosten beinhalten. Wir gehen zurück zu Schritt 7.

7b.) Die Frachtkosten sind über die Dimensionen Kunden, Personal, Versand und Zeit additiv. Dies ergibt eine Fakttable von $78 \times 9 \times 3 \times 12 = 25.572$ Datensätze.

Die Primärschlüssel der drei Dimensionstabellen, die benötigten Kennzahl Frachtkosten und die Tabelle **gesamtBestellungen92** werden zunächst zu einer Tabelle **Faktwerte2** zusammengefügt, welche die umgesetzten Frachtkosten (786 Datensätze) enthält.

(Abfrage7b in Access)

```
SELECT DISTINCTROW DWZeit.Zeit_ID, DWKunden.Kunde_ID, DWPersonal.Personal_ID,
DWVersand.Versand_ID, gesamtBestellungen92.FrachtKost INTO Faktwerte2
FROM DWKunden, DWPersonal, DWVersand, DWZeit, gesamtBestellungen92
WHERE (((DWKunden.KundenCode)=[gesamtBestellungen92].[KundenCode]) AND
((DWPersonal.PersonalNr)=[gesamtBestellungen92].[PersonalNr]) AND
((DWVersand.FirmenNr)=[gesamtBestellungen92].[Versndfirm])
AND((DWZeit.Monat)=Month([gesamtBestellungen92].[Bestelldat])));
```

8b.) Um nun das karthesische Produkt für die Fakttable zu erhalten, werden die Primärschlüssel der Dimensionstabellen bedingungslos zu einer weiteren Tabelle **Leerfakt2** verknüpft und eine leeres Frachtkostenfeld eingefügt.

(Abfrage8b in Access)

```
SELECT DISTINCTROW DWZeit.Zeit_ID, DWKunden.Kunde_ID, DWPersonal.Personal_ID,
DWVersand.Versand_ID, 0 AS Frachtkost INTO Leerfakt2
```


FROM DWKunden, DWZeit, DWPersonal, DWVersand;

Die Tabelle **Leerfakt2** enthält nun 25.572 Datensätze.

9a.) Die Tabelle **Faktwerte2** enthält noch doppelte Datensätze, so daß die Umsätze zunächst in einer weiteren Tabelle **FaktwerteKum2** (297 Datensätze) kumuliert werden.

(Abfrage10b in Access)

```
SELECT DISTINCTROW Faktwerte2.Zeit_ID, Faktwerte2.Kunde_ID, Faktwerte2.Personal_ID,
Faktwerte2.Versand_ID, sum(Faktwerte2.Frachtkost) AS Frachtkost INTO FaktwerteKum2
FROM Faktwerte2
GROUP BY Faktwerte2.Zeit_ID, Faktwerte2.Kunde_ID, Faktwerte2.Personal_ID,
Faktwerte2.Versand_ID;
```

10b.) Zuletzt muß nun die Tabelle **FaktwerteKum2** in die Tabelle **Leerfakt2** eingefügt werden.

(Abfrage9b)

```
SELECT Leerfakt2.Zeit_ID, Leerfakt2.Kunde_ID, Leerfakt2.Personal_ID, Leerfakt2.Versand_ID,
FaktwerteKum2.Frachtkost INTO FrachtFakt
FROM FaktwerteKum2 RIGHT JOIN Leerfakt2 ON (FaktwerteKum2.Zeit_ID = Leerfakt2.Zeit_ID)
AND (FaktwerteKum2.Kunde_ID = Leerfakt2.Kunde_ID) AND (FaktwerteKum2.Personal_ID =
Leerfakt2.Personal_ID) AND (FaktwerteKum2.Versand_ID = Leerfakt2.Versand_ID);
```

11b.) Das Ergebnis ist die endgültige Fakttable **FrachtFakt** (25.572 Datensätze), die mit abschließender Definition des zusammengesetzten Schlüssels Zeit ID, Kunde ID, Personal ID, Versand ID vollständig ist.

Wünschenswert wäre es eventuell noch gewesen, Daten über den durchschnittlichen monatlichen Lagerbestand zu erhalten. Da im Beispiel jedoch nur die Abgänge und keine Zugänge dokumentiert sind, ergeben sich keine aussagekräftigen Ergebnisse.

Zusammengefaßt ergibt sich also das folgende in MS-Access umgesetzte Galaxy-Schema.

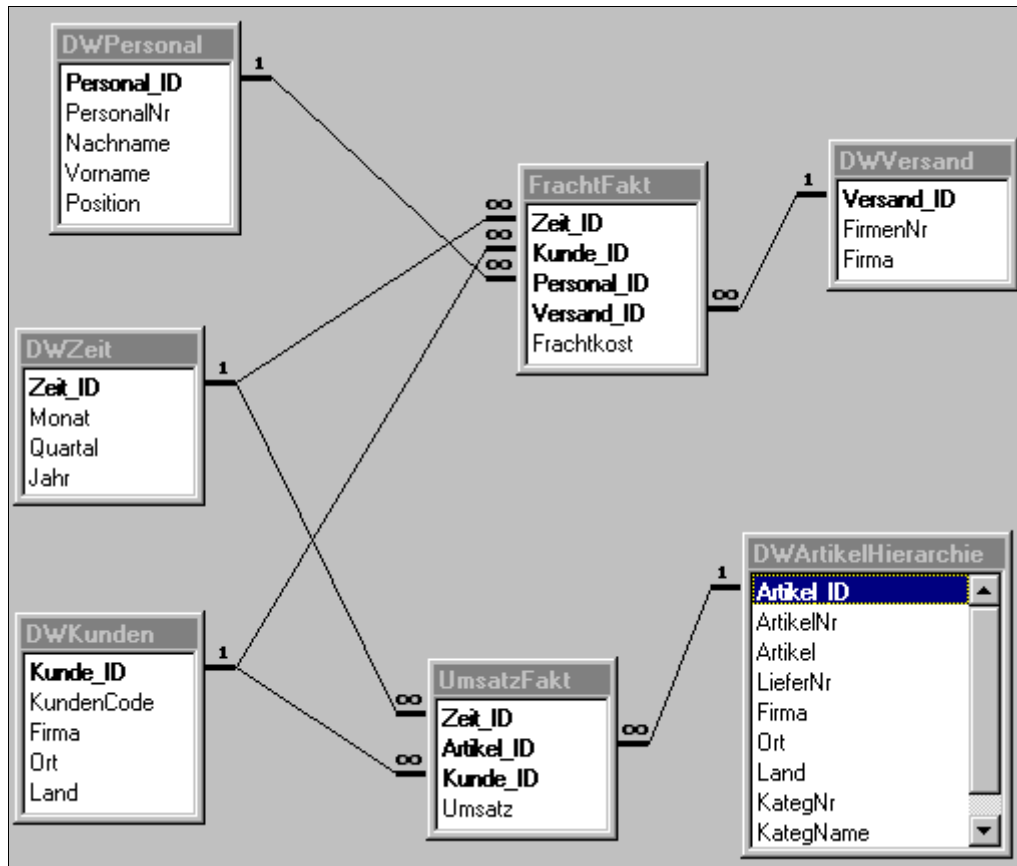


Abbildung 6.14: Data Warehouse-Beispiel in MS-Access

Um nun dieses Galaxy-Schema von MS-Access nach Oracle 7 zu exportieren, muß zunächst jede Tabelle für den Export als Textdatei (Feste Breite) aufbereitet werden. Dann müssen die einzelnen Tabellen in Oracle definiert werden. Anschließend müssen die Textdateien mit dem Oracle-Werkzeug SQL-Loader nach Oracle importiert werden. SQL-Loader benötigt neben der jeweiligen Datendatei noch eine Steuerungsdatei für jede Importtabelle. Für die Artikeltable wäre dies beispielsweise die Datei ORARTIKELHIERARCHIE.CTL in der nach dem Ladebefehl die Textdatei (ORArtikelHierarchie.dat), die Oracle-Datei (DWARTIKELHIERARCHIE) und die Position der einzelnen Felder angegeben werden muß:

```
LOAD DATA
INFILE 'd:\orawin95\rdbs72\loader\ORArtikelHierarchie.dat'
INTO TABLE DWARTIKELHIERARCHIE
(ARTIKEL_ID POSITION(01:11) INTEGER EXTERNAL,
ARTIKELNR POSITION(12:22) INTEGER EXTERNAL,
ARTIKEL POSITION(23:62) CHAR,
LIEFERNR POSITION(63:73) INTEGER EXTERNAL,
FIRMA POSITION(74:113) CHAR,
```

ORT POSITION(114:128) CHAR,
LAND POSITION(129:143) CHAR,
KATEGNR POSITION(144:154) INTEGER EXTERNAL,
KATEGNAME POSITION(155:170) CHAR)

Danach kann mit dem Kommandozeilen-Befehl

```
sqlldr userid=ULI/ULI control=ORARTIKELHIERARCHIE.CTL log=ORARTIKELHIERARCHIE.LOG
```

der SQL-Loader in einem MS-DOS-Fenster gestartet werden, mit dem die Daten nach Oracle übertragen werden. Nachdem diese Prozedur für alle zu exportierenden Dateien durchgeführt wurde, ist der Export abgeschlossen.

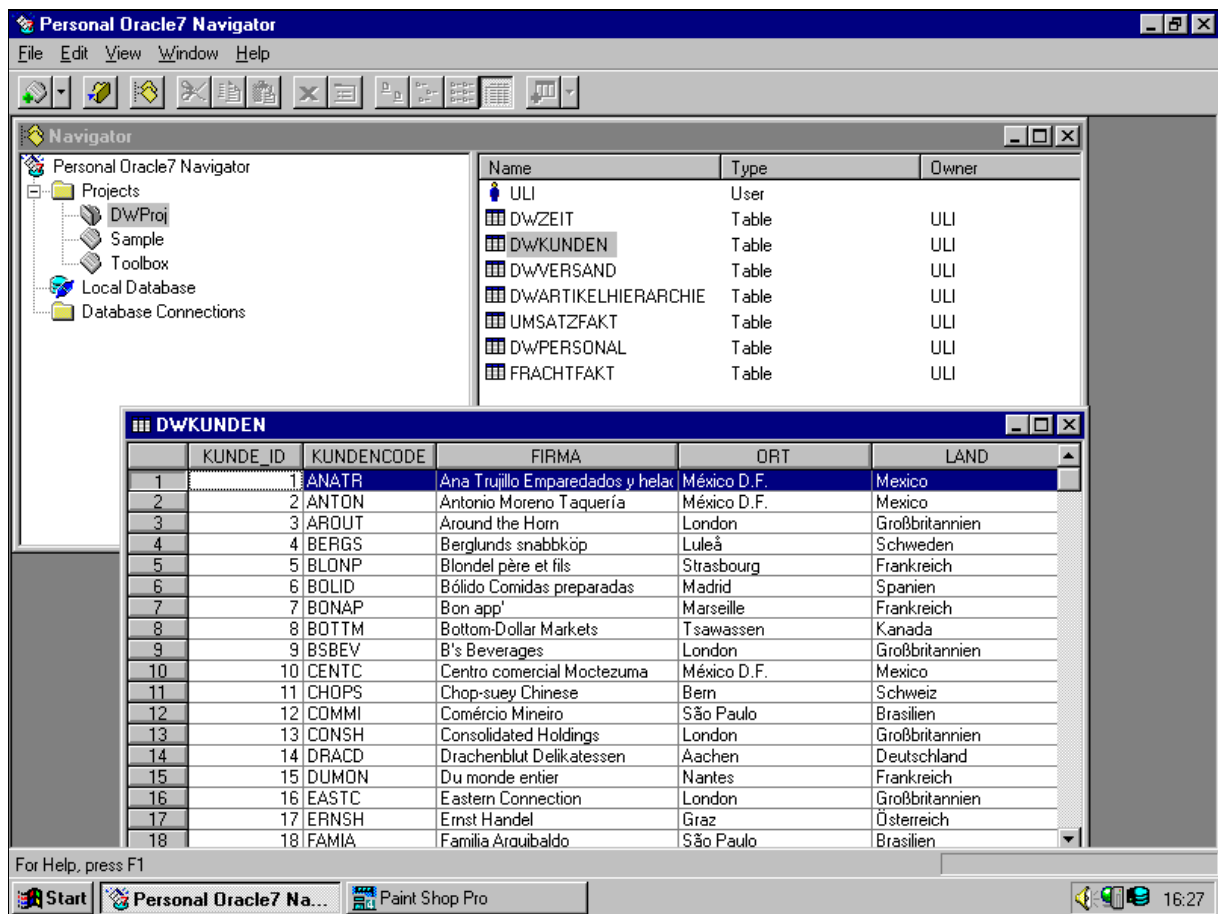


Abbildung 6.15: Data Warehouse-Beispiel in Oracle7

6.6 Vom OLTP-Modell zum multidimensionalen DW-Modell

Zuerst muß in MIK-OLAP ein Modell **Bestellungen** angelegt werden, unter dem die Würfel, Dimensionen und Elemente abgelegt werden.

Anschließend werden die Würfel **Umsatz** und **Frachtkosten** definiert, die den Ordnungsrahmen für die einzelnen Dimensionen darstellen.

Bei der Definition der Dimensionen werden **Artikel**, **Kunden**, **Personal** und **Versand** als freie Dimensionen definiert. Die Kennzahlen **Umsatzwert** und **Frachtkosten** werden als Variablen-Dimension festgelegt. Die Dimension **Monat** wird als freie Periode die Dimension **Jahre** als Jahre definiert.

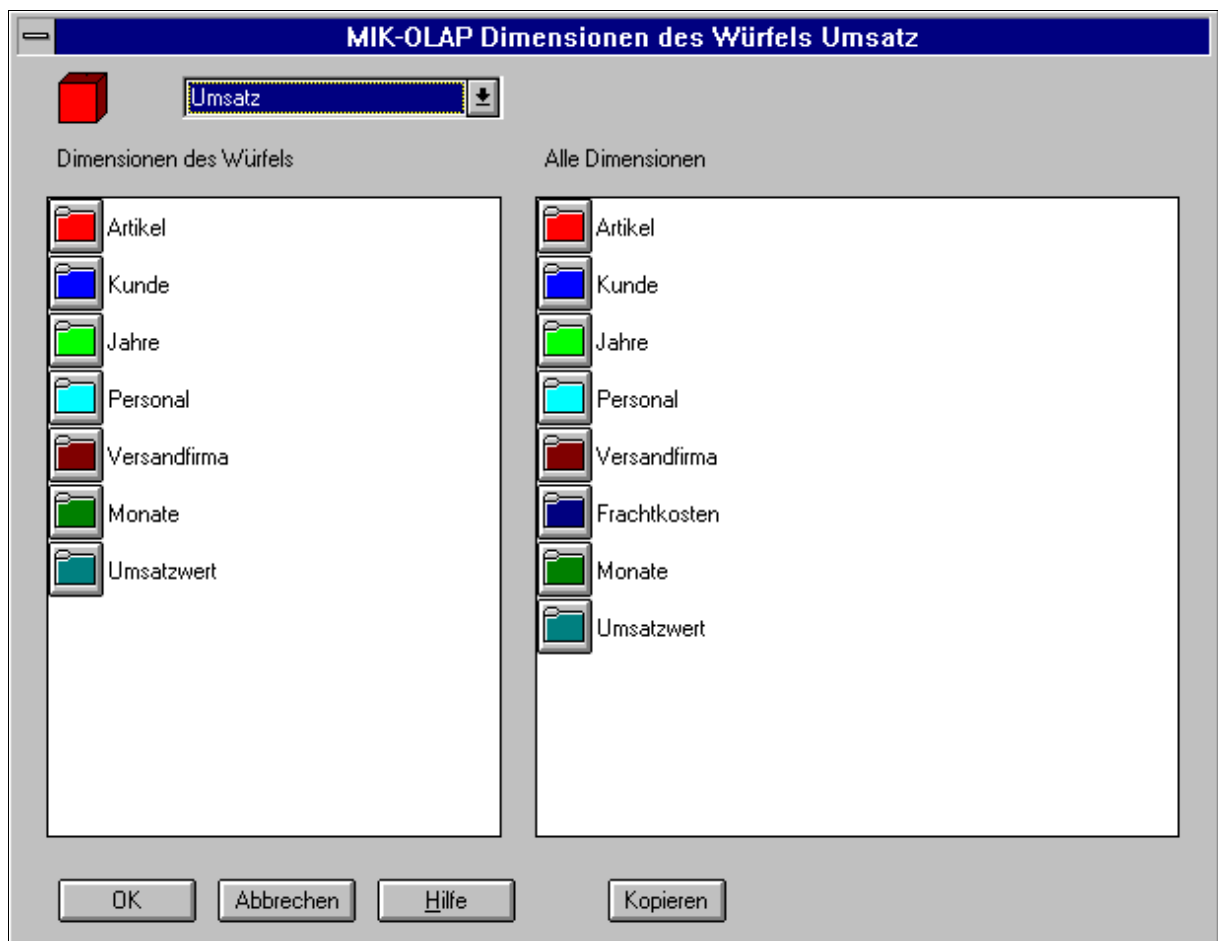


Abbildung 6.16: Data Warehouse-Beispiel in MIK-OLAP

Um die Daten in die einzelnen Dimensionen zu transferieren, müssen zunächst aus MS-Access die Elemente bzw. Daten in der entsprechende Struktur als Textdatei exportiert werden. In

MIK-OLAP kann anschließend für jede Dimension bzw. Kennzahl eine Importdefinition erstellt werden, mit der anschließend die Elemente bzw. Daten übernommen werden können.

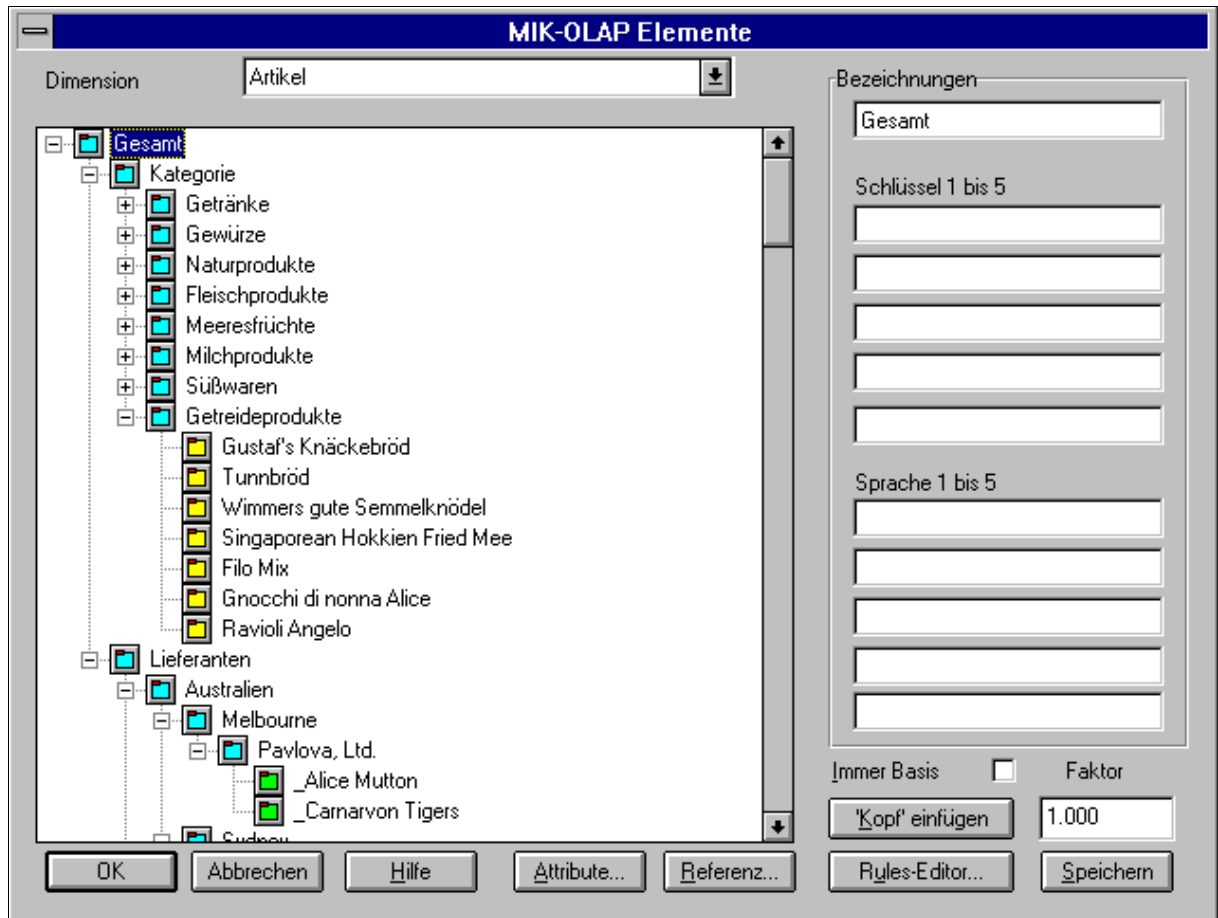


Abbildung 6.17: Importierte Dimensionselemente in MIK-OLAP (Dimension Artikel)

6.7 Vom OLTP-Modell zum objektorientierten DW-Modell

Dieses Modell wurde nicht praktisch realisiert, da der Einarbeitungsaufwand für den zur Verfügung stehenden Zeitraum zu umfangreich gewesen wäre. Folgendes Szenario wäre jedoch denkbar: Als objektorientiertes Datenbankmanagementsystem könnte ObjectStore verwendet werden (Demo unter <http://www.odi.com>). Object Store besitzt einen sogenannten Database Designer, mit dem in OMT Datenmodelle erstellt werden können. Als Beispiel sei das folgende Star-Schema angeführt.

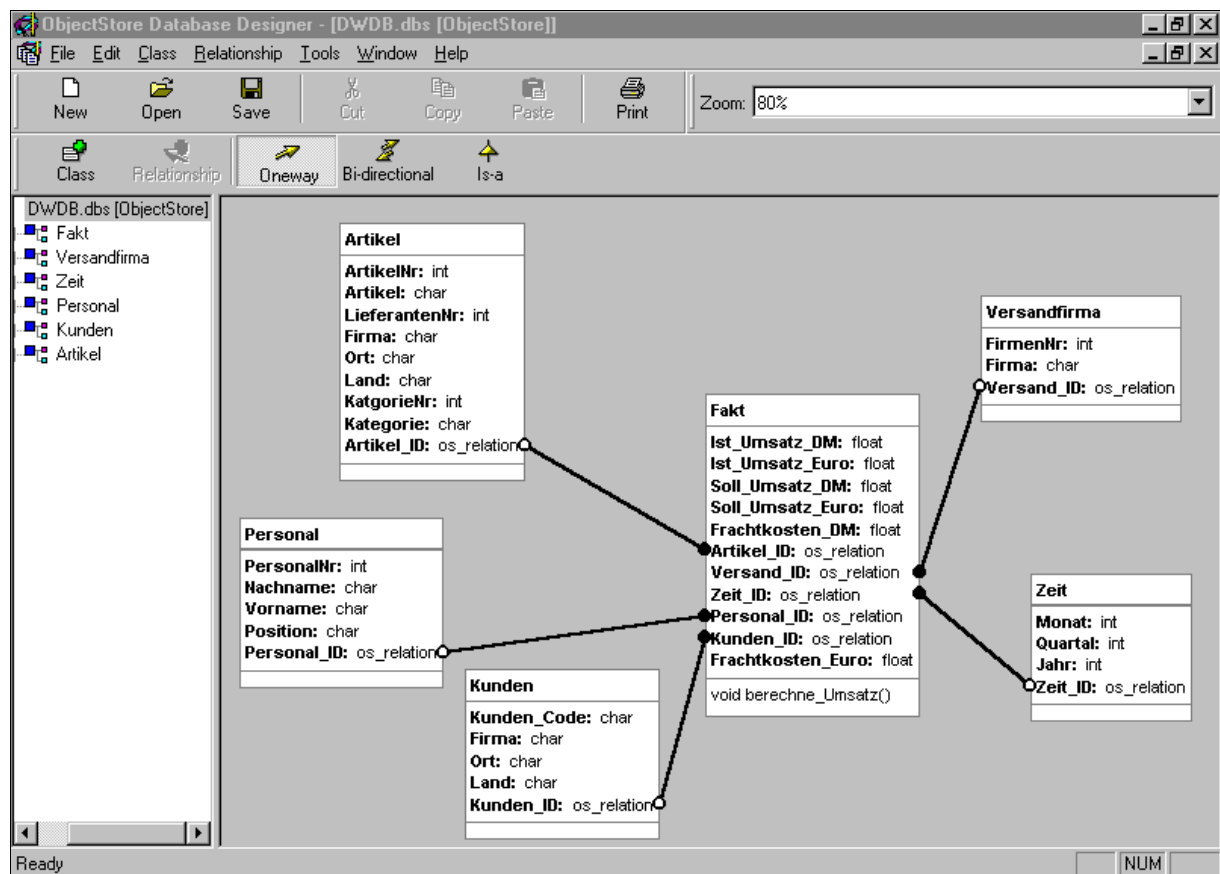


Abbildung 6.18: Data Warehouse-Beispiel in ObjectStore

Mit dem ObjectStore Component Wizard, einem Microsoft C++ 5.0 Plugin, kann aus dem mit dem Database Designer erstellte Modell direkt eine ObjectStore-Anwendung erzeugt werden. Um die Bestelldaten aus der Access-Datenbank nach ObjectStore zu transferieren, könnte das enthaltene Tool DBConnect verwendet werden, mit dem relationale Daten in eine ObjectStore-Datenbank übernommen und als Klassen und Objekte abgebildet werden können. Die so erstellte objektorientierte Datenbank kann mittels einer C++-, Java- oder Active X-API

verwaltet und mittels SQL bzw. OQL abgefragt werden. Es wird jedoch keine spezielle OLAP-Funktionalität bereitstellt. Hier wäre noch Programmieraufwand notwendig.

6.8 Bewertung der Technologien anhand des Kriterienkatalogs

Um die Abbildung des Modells auf eine relationale, multidimensionale und eine objektorientierte Datenbank zu bewerten, werden diese wieder anhand der Kriterien des Katalogs (C-Kriterien) verglichen. Da die Beurteilung hier wirklich nur sehr subjektiv erfolgen kann, da die Modellierung nur an einzelnen Produkten getestet werden konnte, sollen die allgemeinen Datenbankmerkmale aus Kapitel 2 hier mit einfließen, um den Eindruck zu verallgemeinern.

6.9 Subjektive Bewertung der Technologien

Metrik: gut (+1), mittel (0), schlecht (-1)

R = relationale Technologie, M = multidimensionale Technologie, O = objektorientierte Technologie

Richtigkeit: Mit allen drei Technologien läßt sich der Sachverhalt korrekt abbilden. R (+1), M (+1), O (+1)

Angemessenheit: Mit allen drei Technologien lassen sich multidimensionale Data Warehouse-Modelle erstellen. R (+1), M (+1), O(+1)

Interoperabilität: Alle drei Technologien besitzen Schnittstellen zu anderen Systemen (Textimport, ODBC), bei relationalen Datenbanken gibt es hier jedoch die besten Anbindungsmöglichkeiten. R(+1), M(0), O (0)

Reife: Was den Stand der Forschung angeht, ist die relationale Technik sicher am besten erforscht. Objektorientierte Datenbanken sind prinzipiell zwar reifer als multidimensionale Technologie, aber nicht für diesen Anwendungszweck erforscht. R (+1), M (0), O (-1)

Verständlichkeit: Für den Anwender ist die multidimensionale Technologie am verständlichsten. Danach folgen relationale und letztlich objektorientierte Systeme. R (0), M (+1), O (-1)

Erlernbarkeit: Auch hier gibt es wieder klare Vorteile für die multidimensionale Technologie. Um den Umgang mit relationalen Datenbanken zu erlernen ist ein größerer

Einarbeitungsaufwand nötig. Bei objektorientierten Datenbanken ist dieser noch höher. R (0), M (+1), O (-1)

Bedienbarkeit: Die Entwicklung eines Datenbank-Schemas ist mit relationaler Technik gut und mit multidimensionaler Technik sehr gut zu bewerkstelligen. Mit objektorientierter Technologie ist es komplizierter zu arbeiten. R (0), M (+1), O (-1)

Zeitverhalten: Die Performance ist bei multidimensionalen und objektorientierten Datenbanken sehr gut. Relationale Datenbanken sind aufgrund der Verknüpfungen zur Laufzeit langsamer. R (0), M (+1), O (+1)

Verbrauchsverhalten: Relationale und objektorientierte Datenbanken eignen sich sehr gut zur Speicherung großer Datenbestände. Die multidimensionale Technologie hat hier Defizite. R (+1), M (0), O (+1)

Analysierbarkeit: Bei allen drei Technologien gibt es Möglichkeiten, die Daten bzw. die Datenstruktur zu überprüfen. R (0), M (0), O (0)

Modifizierbarkeit: Bei relationalen und objektorientierten Datenbanken ist es einfacher als bei multidimensionalen Datenbanken die Datenstruktur nachträglich zu modifizieren. R (+1), M (0), O (+1)

Austauschbarkeit: Die Austauschbarkeit ist bei allen drei Technologien gegeben. Über Import-/Export-Schnittstellen lassen sich die Daten von einer Technologie in eine andere übertragen. Dabei ist jeweils zuvor die entsprechende Struktur anzulegen. R (0), M (0), O (0)

6.10 Vergleich der einzelnen Technologien

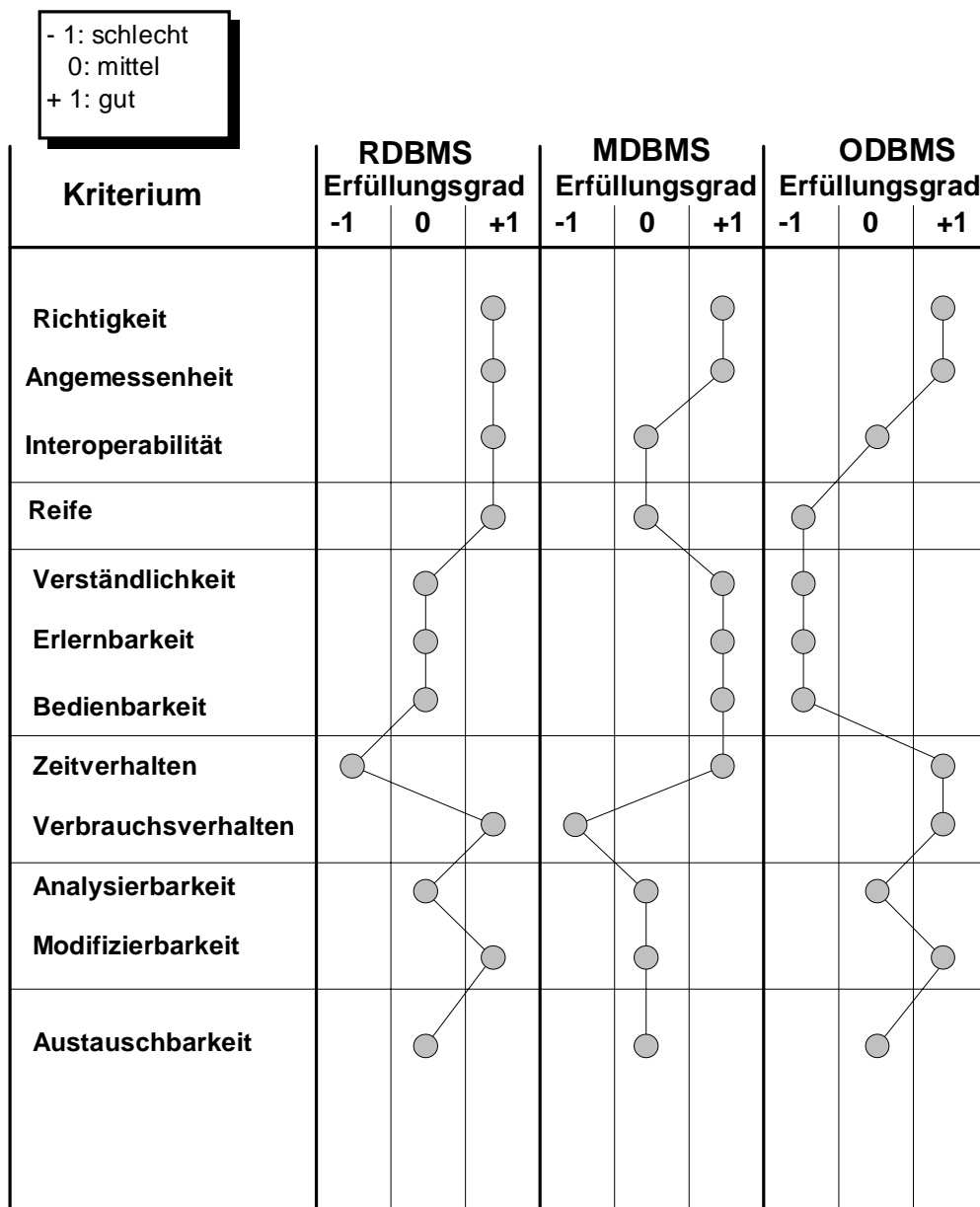


Abbildung 6.19: Technologievergleich

Fazit:

Diese Auswertung zeigt eine Zusammenfassung der in der Literatur zu findenden Informationen, die auch nicht immer ganz schlüssig sind. Zudem fließen eigene Erfahrungen ein, die auf den modellierten Beispielen beruhen. Da jedoch eine relationale Vorbelastung besteht, ist auch hier die Objektivität sicher nicht immer vorhanden. Man sieht jedoch, daß sich mit allen drei Technologien ein multidimensionales Data Warehouse-Modell aufbauen läßt. Die Vorteile relationaler Datenbanken liegen im Bereich der Verwaltung hoher Datenvolumen, in der Wartung der Datenstruktur und in der allgemeinen Anbindung an andere Systeme.

Multidimensionale Datenbanken bieten guten Performance, analytische Flexibilität und eine intuitive Entwicklung. Es lassen sich jedoch keine sehr großen Datenbestände verwalten. Die objektorientierten Datenbanken besitzen gleichzeitig eine gute Performance und können mit großen Datenmengen umgehen. Leider sind sie komplexer zu verstehen und auch noch nicht so bekannt, was in der Praxis dazu führt, daß man, um diese Kombination (Performance/Datenmenge) zu erreichen, eine RDBMS/MOLAP-Server-Struktur für das Data Warehouse verwendet, anstatt die objektorientierte Technologie in Bezug auf OLAP-Fähigkeiten voranzutreiben, die für die Zukunft glaube ich das größere Potential bietet.

7 Schlußbemerkung

Zuletzt möchte ich noch einmal kurz auf das gewählte Beispiel eingehen. es hat sich gezeigt, daß es nicht das ideale Beispiel gewesen ist. Es zeigt zwar einen Data Warehouse relevanten Geschäftsprozeß, aber insgesamt bietet es doch zu wenig Auswertungsmöglichkeiten. Durch Erweiterungen ließ sich hier etwas Abhilfe schaffen, jedoch nicht ausreichend, um alle Möglichkeiten, die beispielsweise ADAPT bietet, auszuschöpfen.

Bezüglich der Verwendung der einzelnen konzeptionellen Methoden zeigt sich, daß jede Methode ihre Berechtigung hat, sei es im Bereich der Dokumentation oder der Entwicklung. Diese Vielfältigkeit ist sicherlich auch gut so. Trotzdem werden Schnittstellen benötigt, um zum einen die Methoden untereinander transformieren zu können und zum anderen, um den Technologiesprung zu überbrücken, den fast jede Methode aufgrund ihrer mangelnden Neutralität impliziert. Hier steht die Entwicklung gerade bei den neueren für Data Warehouse ausgelegten Methoden ADAPT und DF noch am Anfang.

Zu den Architekturen möchte ich weiter nicht viel sagen. Diese haben sich bewährt, wobei die Anforderungen des jeweiligen Anwendungsfalls die Struktur bestimmen. Man hat gesehen, daß die neueren konzeptionellen Methoden diese definierten Strukturen bereits fest integrieren, wodurch sich die Darstellung vereinfacht.

Bei den Technologien scheint sich gerade im Bereich multidimensionaler Datenbanken einiges zu tun. Es gibt inzwischen verschiedene APIs, die von einigen Produkten unterstützt werden, so daß hier ein besserer und vor allen Dingen standardisierter Zugang zu multidimensionalen Daten und Metadaten möglich ist [Spo98], [Rad98]. Objektorientierte Datenbanken bieten für die Zukunft sicher ein großes Potential, insbesondere in Bereichen, wo unstrukturierte Daten ausgewertet werden müssen.

Insgesamt glaube ich, daß trotz der subjektiven Bewertungen die Vor- und Nachteile der einzelnen Methoden, Architekturen und Technologien in dieser Arbeit herausgearbeitet wurden und ersichtlich sind. Es hat sich gezeigt, daß es ein komplexes Thema ist, bei dem sicher noch eine Reihe weiterer Vergleiche möglich sind. Prinzipiell wäre es vielleicht sinnvoll, gerade die konzeptionellen Methoden noch etwas genauer unter die Lupe zu nehmen, um hier wirklich die Potentiale genauer definieren zu können.

8 Literatur

- [AgGuSa] Agrawal Rakesh, Gupta Ashish, Sarawagi Sunita: Modelling Multidimensional Databases, IBM Research Report,
<http://www.almaden.ibm.com/cs/people/ragrawal/>
- [AlLehTesch] Albrecht Jens, Lehner Wolfgang, Teschke Michael: Building a real Data Warehouse for Market Research,
<http://www6.informatik.uni-erlangen.de/dept/staff/lehner.html>
- [And] HyperCube Data Modeling, Andyne Computing Limited
- [And96] Andrews Dave: OLAP, MOLAP, Overlap, Byte, 8/1996,
<http://www.byte.com/art/intervu/oracle.htm>
- [Arb] The Role of the OLAP Server in a Data Warehousing Solution, Arbor Software, White Paper,
http://www.arborsoft.com/essbase/wht_ppr/olapdw.html
- [Bab98] Babst, Jan: Schnelle Informationen für Entscheider, Office Managment 1/98, 12-13
- [Bag et.al. 97] Bager Jo, Becker Jörg, Munz Rudolf Dr.: Zentrallager, c`t, 3/97
- [Balz96] Balzert, Helmut: Lehrbuch der Softwaretechnik, Band 1 + 2, Spektrum, 1996
- [BalzH97] Balzert, Heide: Wie erstellt man ein objektorientiertes Analysemodell, Informatik Spektrum, 1/97, 38-47
- [Bla98] Blaha, M., Premerlani, W.: Object-Oriented Modeling and Design for Database Applications, Prentice Hall, 1998

- [BoHoSche97] Bold M., Hoffmann M., Scheer A.-W.: Datenmodellierung für das Data Warehouse, IWI-Forschungsbericht 139, März 1997,
<http://www.iwi.uni-sb.de/iw-hefte>
- [BuGre] Burton Patrick, Green Stephanie: Meta Data: The Key to Data Warehouse Design, <http://isr.umd.edu/Courses/ENSE623/Data Warehouse>
- [Bulo96] Bulos Dan: A New Dimension, OLAP Database Design, Database Programming and Design, 1996
- [Buyt95] Buytendijk Frank: OLAP: Palying for keeps, 1995
- [ChauDay] Chaudhuri Surajit, Dayal Umeshwar: An Overview of Data Warehousing and OLAP Technologies
- [Che76] Chen, P: The entity-relationship-model, ACM Transactions on Database Systems, 1/76, 9-36
- [Codd et.al.93] Codd E.F., Codd S.B., Salley C.T.: Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate
- [DePo97] DePompa Reimers, Barbara: Demystifying OLAP, Data Warehousing, 5/97,
<http://www.sentrytech.com/data whs/dw05dem.htm>
- [DISC] Bringing Performance to Your Data Warehouse, DISC, White Paper,
<http://www.disc.com/dwhpaper.html>
- [Doag] VITAL - ein generischer Ansatz für eine Datawarehouse-Architektur,
<http://www.doag.org/sig/olap/doc971104.cw3>
- [DW1] Data Warehouse Engines, <http://www.data-warehouse.com/issues/engines.htm>
- [DW2] OLAP/ROLAP/MOLAP, <http://www.data-warehouse.com/issues/olap.htm>
- [Ede95] Edelstein Herb: Technology Analysis: Faster Data Warehouses, 1995,
<http://techweb.cmp.com/iw/556/56olbit.htm>
- [Esp] Esprit Project 22469 - DWQ Foundations of Data Warehouse Quality
<http://www.dbnet.ece.ntua.gr/~dwq>
- [GaGlu97] Gabriel Roland, Gluchowski Peter: Semantische Modellierungstechniken für multidimensionale Datenstrukturen, HMD 195/1997
- [Glu97] Gluchowski, Peter: Data Warehouse, Das aktuelle Schlagwort, Informatik-Spektrum 20: 48-49 (1997)
- [GluGaCha97] Gluchowski P., Gabriel R., Chamoni P.: Management Support Systeme, Springer, 1997

- [GluSche97] Gluchowski Peter, Schelp Joachim: Data Warehouse - Konzepte und Produkte im Internet, Wirtschaftsinformatik 39, 1997, 405-410
- [GoMaRi98] Golfarelli Matteo, Maio Dario, Rizzi Stefano: Conceptual Design of Data Warehouses from E/R Schemes, Proceedings of the Hawaii International Conference On System Science, Januar 1998, <http://www.csr.unibo.it/~golfare/>
- [Gupta Vivek] An Introduction to Data Warehousing, System Services Corporation, Chicago, August 1997
- [Gyss97] Gyssens Marc, Lakshmanan Laks: A Foundation for Multi-Dimensional Databases, Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997
- [Heu97] Heuer Andreas: Objektorientierte Datenbanken, Addison-Wesley, 1997
- [Hön98] Hönig, Thomas, Dr: Multidimensionale Analyse erleichtert Entscheidungen, Office Management, 1/98, 24-25
- [Hox97] Hoxmeier ,John A.: A Framework for Assessing Database Quality, ER 97, <http://osm7.cs.byu.edu/ER97/workshop4/jh.html>
- [Inm95] Inmon W.H.: Tech Topic What is a Data Warehouse?, Prism Vol.1 No.1, Prism Solutions, 1995, http://www.cait.wustl.edu/cait/papers/prism/vol1_no1
- [Jac98] Jacob, Joachim, Dr: Auf dem Weg zum gläsernen Kunden?, Office Mangement, 1/98, 28-31
- [Kimb96] Kimball Ralph: The Data Warehouse Toolkit, John Wiley&Sons, 1996
- [Kimb3/96] Kimball Ralph: Drilling Down, Up and Accross, DBMS Online 3/96, <http://www.dbmsmag.com/9603d05.html>
- [Kimb6/96] Kimball Ralph: Mastering Data Extraction, DBMS Online 6/96, <http://www.dbmsmag.com/9606d05.html>
- [Kimb11/96] Kimball Ralph: Causal (Not Causal) Dimensions, DBMS Online 11/96. <http://www.dbmsmag.com/9611d05.html>
- [Kimb7/97] Kimball Ralph: It´s Time for Time, DBMS Online 7/97. <http://www.dbmsmag.com/9707d05.html>
- [Kimb8/97a] Kimball Ralph: A Dimensional Modeling Manifesto, DBMS Online 8/97, <http://www.dbmsmag.com/9708d15.html>
- [Kimb8/97b] Kimball Ralph: Data Warehouse Role Models, DBMS Online 8/97. <http://www.dbmsmag.com/9708d05.html>
- [Kimb1/98] Kimball Ralph: Bringing up Supermarts, DBMS Online 1/98, <http://www.dbmsmag.com/9801d14.html>

- [Kimb3/98] Kimball Ralph: Meta Meta Data Data, DBMS Online 3/98.
<http://www.dbmsmag.com/9803d05.html>
- [Kimb5/98] Kimball Ralph: Surrogate Keys, DBMS Online 5/98.
<http://www.dbmsmag.com/9805d05.html>
- [Kirk et.al.] Kirkgoetze R., Kurz A., Reiterer H., Tjoa A.M.: The Relevance of Meta Modeling and Data Warehouses for Executive Information Systems
- [Lam] Lambert Bob: Data Modeling for Data Warehouse Development,
<http://www.data-warehouse.com/resources/articles/lamber6.htm>
- [LeEll] Lehmann Peter, Ellerau Peter: Definierte Fachbegriffe als Erfolgsfaktoren eines Data Warehouses
- [Leo98] Leonberg, Max: Multidimensionale Analyse statt Medienbruch, Office Mangement, 1/98, 14-15
- [LeAl] Lehner W., Albrecht J.: Anfrageverarbeitung in multidimensionalen Datenbanksystemen,
<http://www6.informatik.uni-erlangen.de/dept/staff/lehner.html>
- [LeTesch96] Lehner W., Teschke M.: Modellierungsalternativen im "Scientific Computing", GI-Workshop "Multidimensionale Datenbanken", 4.März 1996, Ulm
- [LeTeWe] Lehner W., Teschke M., Wedekind H.: Über Aufbau und Ausertung multidimensionaler Daten (Kurzbeitrag),
<http://www6.informatik.uni-erlangen.de/dept/staff/lehner.html>
- [Mat98] Mattison Rob: Understanding Database Management Systems, McGraw-Hill, 1998
- [Mat96] Mattison Rob: Data Warehousing, McGraw-Hill, 1996
- [McGu96] McGuff Frank: Data Modeling for Data Warehouses, 1996,
<http://members.aol.com/fmcguff/dwmodell/dwmodel.html>
- [MeiWü] Meier Andreas, Wüst Thomas: Objektorientierte Datenbanken, dpunkt, 1997
- [MeKha97] Meredith Marie, Khader Aslam: Divide and Aggregate: Designing Large Warehouses, Database Programming & Design online,
<http://www.dbpd.com/khader.htm>
- [Menn96] Menninger Dave: Designing a Database for OLAP, Oracle Magazine, 3/4 1996,
<http://www.oramag.com/archives/26meth.html>
- [MicrStr1] Relational OLAP: An Enterprise-Wide Data Delivery Architecture, MicroStrategy Inc., White Paper, <http://www.strategy.com>
- [MicStr2] The Case for Relational OLAP, MicroStrategy Inc., White Paper
- [MuBe96] Mucksch, Behme Hrsg: Das Data Warehouse-Konzept, Gabler, 1996

- [Oes97] Oestereich B.: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language, Oldenbourg, 1997
- [OICounc1] OLAP Council White Paper
<http://www.olapcouncil.org>
- [OICounc2] OLAP and OLAP-Server Definitions
<http://www.olapcouncil.org/research/glossaryly.htm>
- [Ora] Delivering OLAP to the Enterprise, Oracle Express Server, White Paper
- [Orr96] Orr Ken: Data Warehousing Technology, Ken Orr Institute 1996, White Paper,
<http://www.kenorrinst.com/dwpaper.html>
- [Ort] Ortner, Erich, Prof.Dr.: Metainformationssysteme, Skript zur Vorlesung
- [Ort94] Ortner, Erich, Prof.Dr.: Datenmodellierung - Systemunabhängiger Entwurf von Datenstrukturen, Kursmaterialien SS 1994, April 1994
- [Pen1] Pendse, Nigel: What ist OLAP?, The OLAP Report, Februar 1998,
<http://www.olapreport.com/fasmi.html>
- [Pen2] Pendse, Nigel: OLAP Architectures, The OLAP Report, Februar 1998,
<http://www.olapreport.com/Architecures.html>
- [Pen3] Pendse, Nigel: Multidimensional data structures, The OLAP Report, Februar 1998, <http://www.olapreport.com/Architecures.html>
- [Pet94] Peterson Stephen: Stars: A Pattern Language for Query Optimized Schema, 1994, <http://c2.com/ppr/stars.html>
- [PoRe97] Poe Vidette, Reeves Laura: Aufbau eines Data Warehouse, Prentice Hall, 1997
- [Rad95] Raden Neil: Star Schema101, 1995
<http://www.strategy.com/dwf/raden/str101.htm>
- [Rad96] Raden Neil: Modeling the Data Warehouse, 1996,
http://www.strategy.com/dwf/raden/iw0196_1.htm
- [Rad98] Raden Neil: Dialog on Plato, Database Programming & Design online 3/98,
<http://www.dbpd.com/9803radn.htm>
- [Renn98] Rennhackkamp Martin: Red Brick Warehouse 5.1, DBMS Online 6/98.
<http://www.dbmsmag.com/9806d17.html>
- [Rens98] Rensmann, Jörg: Wissen ist Macht, Strategisches Informationsmangement, Office Management, 1/98, 8-10
- [Reut98] Reuter, Andreas, Prof.Dr.: Neue Anforderungen an Datenbanksysteme, OfficeMangement, 1/98, 20-23

- [Ruf] Ruf Thomas PD Dr-Ing.: Einsatz von Data Warehousing und OLAP im Bereich der Marktforschung, GfK Marketing Services Europe,
<http://fg-db.informatik.tu-chemnitz.de/DBR/Ausgabe/FGDB97.html>
- [Sach98] Sachdeva Satya: Meta Data Architecture for Data Warehousing, DM Review Magazine, April 98, http://dmreview.com/issues/1998/apr/articles/apr98_66.htm
- [SiGra98] Silverston Len, Graziano Kent: Creating a Data Warehouse Design from a Corporate Data Model, DM Review Quarterly, Februar 1998,
<http://www.data-warehouse.com/article4>
- [Sonn98] Sonntag, Dirk: Data Warehouse Macht Daten zu Fakten, Office Mangement, 1/98, 17-19
- [Speed] Hybrid OLAP: The Best of Both Worlds, Speedware Corporation, White Paper
<http://www.speedware.com>
- [Spo98] Spofford George: Attack of the Killer APIs, Database Programming & Design online 3/98, <http://www.dbpd.com/9803spof.htm>
- [Toto98] Totok Andreas: Semantische Modellierung von multidimensionalen Datenstrukturen, Vortrag am 4. Workshop des GI-Arbeitskreises, April 1998,
<http://www.tu-bs.de/institute/wirtschaftswi/controlling/staff/atotok/atotok.html>
- [Vetsch95] Vetschera Rudolf: Informationssysteme der Unternehmensführung, Springer, 1995
- [Voss94] Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme, Addison-Wesley, 1994
- [WeLeTeAl] Wedekind H., Lehner W., Teschke M., Albrecht J.: Preaggregation in multidimensional Data Warehouse Environments,
<http://www6.informatik.uni-erlangen.de/dept/staff/lehner.html>
- [Wis95] Wiseth Kelli: Seven Steps to Building a Data Warehouse, Oracle Magazine, 1/2 1995 <http://www.oramag.com/archives/15cov5.html>
- [WuBu] Wu Ming-Chuan, Buchmann Alejandro: Research Issues in Data Warehousing, DVS1, Fachbereich Informatik, TH Darmstadt