Integration von Data Mining und Online Analytical Processing: Eine Analyse von Datenschemata, Systemarchitekturen und Optimierungsstrategien

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart
zur Erlangung der Würde
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von
Holger Schwarz
aus Esslingen a.N.

Hauptberichter	Prof. Dr.-Ing. habil. B. Mitschang
Mitberichter	Prof. Dr.-Ing. Dr. h.c. T. Härder
Tag der mündlichen Prüfung	04. 12. 2002

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2003

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Abteilung Anwendersoftware des Instituts für Parallele und Verteilte Systeme der Universität Stuttgart.

Besonderen Dank möchte ich Herrn Prof. Bernhard Mitschang aussprechen. Dies gilt insbesondere für seine Bereitschaft, diese Arbeit nach seinem Wechsel an die Universität Stuttgart zu betreuen. Seine Offenheit für neue Ideen, seine vielfältigen Anregungen sowie die Freiheit, die er mir bei deren Umsetzung und Ausarbeitung gab, machten die Arbeit in dieser Form erst möglich. Der Gedankenaustausch durch zahlreiche Diskussionen und seine Unterstützung während der gesamten Bearbeitungszeit trugen wesentlich zum Gelingen der Arbeit bei.

Ebenso möchte ich Herrn Prof. Theo Härder für das entgegengebrachte Interesse am Thema meiner Arbeit sowie für deren kritische Durchsicht danken. Seine interessanten und wertvollen Anregungen haben wesentlich zu einer Präzisierung verschiedener Aspekte der vorliegenden Dissertation beigetragen. Herrn Prof. Härder danke ich auch für seine Bereitschaft zur Übernahme des Mitberichts.

Bei allen Mitarbeiterinnen und Mitarbeitern der Abteilung Anwendersoftware möchte ich mich für die kollegiale und freundschaftliche Zusammenarbeit und Unterstützung bedanken. Insbesondere gilt mein Dank Ralf Rantzau für zahlreiche fachliche Ratschläge und Diskussionen im Verlauf der Arbeit sowie für die Geduld beim Korrekturlesen. Mehrere Studierende haben im Rahmen ihrer Studien- und Diplomarbeiten ebenfalls zu der vorliegenden Arbeit beigetragen. Stellvertretend sei an dieser Stelle André Bouillet, Tobias Kraft und Ralf Wagner gedankt.

Die vorliegende Arbeit wäre ohne die Hilfe meiner Familie sicherlich nicht zustande gekommen. Mein besonderer Dank gilt meiner Frau für ihre Unterstützung, Geduld und Verständnis.

Stuttgart, im Juli 2003 Holger Schwarz

Zusammenfassung

Die technischen Möglichkeiten, Daten zu erfassen und dauerhaft zu speichern, sind heute so ausgereift, dass insbesondere in Unternehmen und anderen Organisationen große Datenbestände verfügbar sind. In diesen Datenbeständen, häufig als Data Warehouse bezeichnet, sind alle relevanten Informationen zu den Organisationen selbst, den in ihnen ablaufenden Prozessen sowie deren Interaktion mit anderen Organisationen enthalten. Vielfach stellt die zielgerichtete Analyse der Datenbestände den entscheidenden Erfolgsfaktor für Organisationen dar.

Zur Analyse der Daten in einem Data Warehouse sind verschiedenste Ansätze verfügbar und erprobt. Zwei der wichtigsten Vertreter sind das Online Analytical Processing (OLAP) und das Data Mining. Beide setzen unterschiedliche Schwerpunkte und werden bisher in der Regel weitgehend isoliert eingesetzt. In dieser Arbeit wird zunächst gezeigt, dass eine umfassende Analyse der Datenbestände in einem Data Warehouse nur durch den integrierten Einsatz beider Analyseansätze erzielt werden kann. Einzelne Fragestellungen, die sich aus diesem Integrationsbedarf ergeben werden ausführlich diskutiert.

Zu den betrachteten Fragestellungen gehört die geeignete Modellierung der Daten in einem Data Warehouse. Bei der Bewertung gängiger Modellierungsansätze fließen insbesondere die Anforderungen ein, die sich durch den beschriebenen Integrationsansatz ergeben. Als Ergebnis wird ein konzeptuelles Datenmodell vorgestellt, das Informationen in einer Weise strukturiert, die für OLAP und Data Mining gleichermaßen geeignet ist. Im Bereich der logischen Modellierung werden schließlich diejenigen Schematypen identifiziert, die die Integration der Analyseansätze geeignet unterstützen.

Im nächsten Schritt sind die für Data Mining und OLAP unterschiedlichen Systemarchitekturen Gegenstand dieser Arbeit. Deren umfassende Diskussion ergibt eine Reihe von Defiziten. Dies führt schließlich zu einer erweiterten Systemarchitektur, die die Schwachstellen beseitigt und die angestrebte Integration geeignet unterstützt. Die erweiterte Systemarchitektur weist eine Komponente zur anwendungsunabhängigen Optimierung unterschiedlicher Analyseanwendungen auf. Ein dritter Schwerpunkt dieser Arbeit besteht in der Identifikation geeigneter Optimierungsansätze hierfür. Die Bewertung der Ansätze wird einerseits qualitativ durchgeführt. Andererseits wird das Optimierungspotenzial der einzelnen Ansätze auch auf der Grundlage umfangreicher Messreihen gezeigt.

Abstract

Current technology allows enterprises and all kinds of organisations to store huge amounts of data. According to the data warehouses concept, a central data store might be used to integrate data from heterogeneous sources within the organisation and combine it with external data. The data warehouses contains all relevant information about the organisation itself, about processes running in it as well as about the interaction with other organisations. Hence, the detailed analysis of data warehouses is a key success factor for modern organisations.

Several approaches for intelligent data analysis are available, tried and tested. Online Analytical Processing (OLAP) and Data Mining represent two of the most important approaches. They put the main emphasis on different aspects of the data and allow to derive different kinds of information. So far, these approaches have mainly be used in isolation. As shown in this thesis, an in-depth analysis of huge data warehouses requires an approach that integrates both, OLAP and data mining. This integration leads to several problems. Some of them related to data models, system architecture and optimization are discussed here.

One of the research problems covered by this thesis is the appropriate data model for a data warehouse supporting OLAP as well as data mining. The requirements resulting from the integration of both approaches are identified and known conceptual models are evaluated according to these criteria. The COCOM model (Common Conceptual Warehouse Model) is proposed. It covers all important modelling capabilities for both approaches of data analysis. Based on this model, several proposals for logical schemes are evaluated. This includes the flat schema, the star schema as well as the snowflake schema. The main criteria in this evaluation are: the modelling capabilities, redundancy, changeability and the performance of query processing. It turns out that mid-complex models like the star schema are well suited for OLAP as well as for data mining.

The next part of this thesis covers the analysis of different system architectures. The discussion starts with some general remarks on system architecture and explains the quality attributes for system architecture that are most important in the context of this work: scalability, load balancing, usage of database functionality, maintainability and portability. The typicall system architecture for OLAP as well as for data mining is identified and assessed according to these criteria. The analysis reveals that they do not support the integrated maintenance of meta data

and lack optimization approaches that are independent of the individual applications. An extended system architecture is proposed that offers support in these areas. The optimization in particular is achieved by an additional system component, a so called DSS optimizer, that allows to optimize statement sequences generated by different applications.

Suitable optimization approaches for this system component are identified in the last part of the thesis. For this purpose the notion of a statement sequence is formally defined and some examples are given. Several optimization approaches for statement sequences are proposed. They fall into three groups: approaches that focus on a modification of data, approaches in which the execution of statement sequences is modified and approaches that cover the modification of the statement sequence itself. The optimization approaches that are discussed in more details are: intra-statement-sequence parallelism, usage of statistical information, usage of access paths and materialized views, partitioning of base tables, rewriting statement sequences as well as combining the complete statement sequence into a single statement. All approaches are evaluated according to the following criteria: availability of the appropriate technology, optimization potential, influence on other applications, availability of information for the component that applies the strategy and suitability for an optimizer external to the data warehouse database. This evaluation of optimization approaches is completed by experimental results based on TPC-H benchmark data. It turns out that the usage of parallelism and statistics are effective strategies whereas all strategies that affect the schema of the underlying data warehouse database are not appropriate for a DSS optimizer. The most promising strategies are the rewriting of statement sequences and the combination of complete sequences into a single statement. They are applicable independently of the underlying database system and showed geat performace benefit in the experiments.

Several aspects of integrating OLAP and data mining are not coverd by this thesis. This includes details of integrated meta data support. For some of the optimization strategies further investigations are necessary to identify appropriate heuristics and cost models. It would also be interesting to study if and how they could be integrated into the database system and whether they are applicable to other query processing problems, e.g. stream data.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AST	Automated Summary Table
CRM	Customer Relationship Management
DB	Datenbasis
DBS	Datenbanksystem
DBVS	Datenbankverwaltungssystem
DSS	Decision Support System
DM	Data Mining
DWDB	Data-Warehouse-Datenbank
DWDBS	Data-Warehouse-Datenbanksystem
GB	Gigabyte
HOLAP	Hybrid OLAP
KDD	Knowledge Discovery in Databases
KI	Künstliche Intelligenz
MDBS	Multidimensionales Datenbanksystem
MIS	Management-Informationssystem
MOLAP	Multidimensional OLAP
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing

ORDBS	Objekt-relationales Datenbanksystem
PMML	Predictive Model Markup Language
ROLAP	Relational OLAP

1 Einleitung

Die Speicherung und Analyse großer Datenmengen spielt heute in vielen Bereichen eine zentrale Rolle. Für zahlreiche Organisationen und insbesondere für Unternehmen stellt die Informationsgewinnung auf der Grundlage verfügbarer Daten häufig einen, wenn nicht gar den entscheidenden Erfolgsfaktor dar. Auf Grund der technologischen Entwicklung der vergangenen Jahrzehnte stellt die dauerhafte Speicherung aller anfallenden Daten keine wesentliche Herausforderung mehr dar. Mit dem Anwachsen der verfügbaren Datenbestände rückt deren zielgerichtete und effiziente Analyse immer mehr in das Zentrum des Interesses. Als Grundlage hierfür integrieren Organisationen, die in der Regel Zugriff auf eine Vielzahl heterogener Datenbestände haben, die für Analysen relevanten Daten in einem sogenannten Data Warehouse. Bei dieser Analyse werden unterschiedliche Ansätze verfolgt. Von zentraler Bedeutung sind hierbei das Online Analytical Processing (OLAP) und Data Mining, weshalb der Fokus dieser Arbeit auch auf diesen beiden Analyseansätzen liegt. Zwischen beiden Ansätzen gibt es erhebliche Unterschiede, die dazu führen, dass für eine umfassende Analyse der Daten in einem Data Warehouse sowohl Techniken des Data Mining als auch des Online Analytical Processing eingesetzt werden müssen. So kann es beispielsweise notwendig sein, die Ergebnisse einer OLAP-Analyse mit Hilfe eines Data-Mining-Werkzeugs weiter zu analysieren, da die hierfür notwendigen Methoden durch ein OLAP-Werkzeug nicht bereitgestellt werden. Die Notwendigkeit zu einer Integration der unterschiedlichen Analyseansätze stellt den Ausgangspunkt für die vorliegende Arbeit dar.

Bei der Bereitstellung der integrierten Analysemöglichkeiten stellen sich einige wichtige Fragen. Zu diesen gehört einerseits die Frage, wie ein Data Warehouse modelliert werden muss, um den integrierten Einsatz der beiden Analyseansätze zu unterstützen. Andererseits muss auch geklärt werden, welches die geeignete Systemarchitektur ist und welche Optimierungsansätze in diesem Rahmen verfolgt werden können und erfolgversprechend sind. Diese Fragen wurden bisher ausschließlich getrennt nach den einzelnen Analyseansätzen betrachtet. Im Rahmen dieser Arbeit wird darum eine integrierte Betrachtung angestrebt, die sowohl die Anforderungen aus dem Bereich des Online Analytical Processing als auch aus dem Data Mining berücksichtigt. Für die Modellierung des Data Warehouse wurden bisher in erster Linie Ansätze publiziert, die speziell auf das Online analytical Processing zugeschnitten sind. Diese werden in der vorliegenden Arbeit hinsichtlich ihrer Eignung für einen integrierten Analyseansatz unter-

sucht. Dies erfolgt getrennt nach den einzelnen Phasen des Datenbankentwurfs. Im Bereich der Systemarchitektur wird analysiert, welche Architekturen typischen Vertretern der beiden Analyseansätze zu Grunde liegen, wie diese charakterisiert sind und welche Defizite sie hinsichtlich des Integrationsaspekts aufweisen. Darauf aufbauend wird eine Systemarchitektur vorgestellt, die die identifizierten Schwachstellen behebt und die integrierte Nutzung der beiden Analyseansätze geeignet unterstützt. Eine wichtige Neuerung dieser Systemarchitektur ist eine Komponente, die die anwendungsunabhängige Optimierung von Sequenzen von SQL-Anweisungen erlaubt. Solche Sequenzen werden typischerweise sowohl im Bereich Online Analytical Processing als auch im Bereich Data Mining generiert. Die Untersuchung verschiedener Optimierungsansätze, die hierbei verfolgt werden können, bildet einen weiteren Schwerpunkt dieser Arbeit. Diese werden einerseits qualitativ bewertet, um Aussagen bezüglich ihrer Eignung in der vorgeschlagenen Systemarchitektur treffen zu können. Andererseits wird das Potenzial der verschiedenen Optimierungsansätze auch auf der Grundlage umfangreicher Messungen bewertet.

Die Arbeit ist folgendermaßen gegliedert: In Kapitel 2 werden zunächst die grundlegenden Konzepte des Data Warehouse, des Online Analytical Processing und des Data Mining erläutert. Darüber hinaus werden zwei Anwendungsszenarien erläutert. Die betrachteten Analyseansätze werden in Kapitel 3 genauer charakterisiert und auf dieser Grundlage die Motivation für eine integrierte Betrachtung von Data Mining und OLAP dargestellt. Die folgenden drei Kapitel stellen den Schwerpunkt der vorliegenden Arbeit dar. In Kapitel 4 werden zunächst die Fragen bezüglich des Datenmodells für ein Data Warehouse, getrennt nach den einzelnen Phasen des Modellierungsprozesses, betrachtet. Systemarchitekturen existierender Systeme werden in Kapitel 5 analysiert. Schließlich wird eine Architektur für ein integriertes System vorgeschlagen. In Kapitel 6 werden potenzielle Ansätze für eine anwendungsunabhängige Optimierung erläutert und sowohl qualitativ als auch quantitativ bewertet. Die Zusammenfassung der Ergebnisse und ein Ausblick sind Bestandteil von Kapitel 7.

2 Grundlagen

Die verfügbaren Technologien zur Speicherung von Daten wurden in den vergangenen 30 Jahren stark verbessert. Dies betrifft sowohl die Kapazität der eingesetzten Speichermedien als auch die Software zur Verwaltung, Bearbeitung und Analyse der Daten. Charakteristische Merkmale von Serversystemen, die im Jahr 2002 für umfangreiche und datenintensive kommerzielle Anwendungen eingesetzt werden, sind in diesem Zusammenhang: Viele Gigabyte Hauptspeicher sowie zugeordnete Festplattenkapazität im Bereich von vielen Terabyte.

Es ist aber nicht nur die reine Zunahme des Datenvolumens zu verzeichnen, sondern auch eine Veränderung bei den Softwaresystemen, mit deren Hilfe die Daten verwaltet werden. Ausgehend von der rein auf Dateien basierenden Verwaltung wurden unterschiedliche Varianten von *Datenbankverwaltungssystemen* (DBVS) entwickelt. Zusammen mit der eigentlichen *Datenbasis* (DB) bildet das DBVS ein *Datenbanksystem* (DBS) [HR99]. Für umfangreiche, kommerzielle Anwendungen haben sich vor allem *objekt-relationale Datenbanksysteme* (ORDBS) durchgesetzt, während hierarchische oder rein objektorientierte Datenbanksysteme nur in Randbereichen eine Rolle spielen. Trotz des erfolgreichen Einsatzes von Datenbanksystemen liegt weiterhin der überwiegende Teil der global verfügbaren Daten in Dateisystemen vor und wird nicht durch ein DBVS verwaltet. Dies gilt insbesondere für Daten, die nicht numerischen oder alphanumerischen Typs sind [MK+99]. Dieser Teil von Datenbeständen wird im Rahmen der vorliegenden Arbeit nicht weiter thematisiert. Diese Einschränkung ist für das Thema dieser Arbeit sinnvoll. Hier liegt der Fokus auf der Analyse von Daten, die im Rahmen der operativen Prozesse einer Organisation anfallen. Diese werden entweder bereits von einem DBVS verwaltet oder aber, wie im folgenden Abschnitt beschrieben, in ein solches integriert.

Mit der wachsenden Möglichkeit, Daten dauerhaft zu speichern, ist auch der Bedarf zur Analyse dieser Daten gestiegen. Dies führt einerseits zu einer stetigen Erweiterung der Mächtigkeit von SQL, der Standardanfragesprache für objektrelationale Datenbanksysteme. Andererseits wurde das Spektrum und die Mächtigkeit von Software zur Datenanalyse kontinuierlich erweitert. In diesen Bereich fallen sowohl Werkzeuge, die mit Hilfe einer grafischen Oberfläche die einfache Definition und Ausführung von SQL-Anfragen erlauben als auch solche, die nach komplexen Zusammenhängen in den Daten suchen. Im Rahmen dieser Arbeit sind in erster Linie die als *Online Analytical Processing* (OLAP) und *Data*

Mining (DM) bezeichneten Ansätze zur Analyse von Daten von Bedeutung. Diese werden in den folgenden Abschnitten beschrieben und gegenseitig sowie gegenüber anderen Ansätzen abgegrenzt. Anwendungen, die auf mindestens einem dieser *Analyseansätze* beruhen, werden hier als *Analyseanwendungen* bezeichnet.

2.1 Data Warehouse

Mit *Data Warehousing* wird ein Ansatz beschrieben, eine integrierte, subjekt-orientierte, dauerhafte und zeitabhängige Datensammlung für unterschiedliche Analysen bereitzustellen [Inm92]. Die Motivation hierfür ergibt sich aus der Tatsache, dass praktisch alle Organisationen und insbesondere Unternehmen operationale Daten mit ganz unterschiedlichen Systemen verwalten und bearbeiten. Diese Daten sind nicht nur für die einzelnen Prozesse einer Organisation relevant, sondern beinhalten auch Informationen, die für Entscheidungen sowohl auf der dispositiven als auch auf der strategischen Ebene von Bedeutung sind. *Management-Informationssysteme* (MIS) stellen frühe Ansätze dar, diese Informationen direkt aus den operationalen Systemen zu extrahieren [DO85]. Zwei Probleme standen bei der Nutzung dieser Systeme im Vordergrund. Zum einen war die Durchführung der notwendigen Analysen meist mit erheblichem Aufwand verbunden, da die relevanten Daten für jeden Bericht separat aus dem operationalen System extrahiert und aufbereitet werden mussten. Zum anderen konnten Querbezüge zwischen den Daten unterschiedlicher Systeme nicht oder nur sehr aufwändig identifiziert werden. In der Regel waren die Nutzer mit unflexiblen, aufwändig zu entwickelnden und schwer zu wartenden Systemen konfrontiert, die den Erwartungen nicht gerecht werden konnten [Eme73]. Diesen Problemen wird bei der Einrichtung eines Data Warehouse begegnet, indem für die Organisation ein individueller Standardprozess definiert wird, der Daten regelmäßig aus den heterogenen Quellen extrahiert und nach den notwendigen Aufbereitungsschritten in eine separate Datenbank integriert. Diese wird in der weiteren Arbeit als *Data-Warehouse-Datenbank* (DWDB) bezeichnet. Darüber hinaus wird angenommen, dass diese Datenbank von einem relationalen bzw. objekt-relationalen Datenbankmanagementsystem verwaltet wird, im Folgenden als *Data-Warehouse-Datenbanksystem* (DWDBS) bezeichnet. Diese Einschränkung ist gerechtfertigt, da ein Großteil der existierenden Data Warehouses auf relationaler Technologie basiert. Auf der Grundlage dieser DWDB können dann alle für eine Organisation notwendigen Analysen durchgeführt werden. Da das DWDBS den Zugriff auf Daten aus unterschiedlichen Quellsystemen ermöglicht

und für alle Zugriffe dieselbe Schnittstelle anbietet, werden die beschriebenen Hauptprobleme von MIS behoben.

Neben der Integration unterschiedlicher Quellsysteme als wichtige Motivation für den Aufbau eines Data Warehouse wurden bereits weitere Merkmale eines Data Warehouse genannt. Danach ist es subjekt-orientiert, dauerhaft und zeitabhängig. “Subjekt-orientiert” heißt hier, dass nicht die Prozesse und Anwendungen einer Organisation die wichtigsten Gesichtspunkte bei der Modellierung der Daten in einem Data Warehouse sind, sondern dass hier Personen und thematische Bereiche die zentrale Rolle spielen. Das Thema Datenmodellierung wird als einer der zentralen Aspekte dieser Arbeit in Kapitel 4 ausführlich behandelt. Mit dem Attribut “dauerhaft” wird beschrieben, dass Daten in einem Data Warehouse dauerhaft in der Form gespeichert werden, in der sie nach dem Laden aus den Quellsystemen und den notwendigen Transformationen vorliegen. Es werden in der Regel keine Änderungsoperationen auf den Daten in der DWDB ausgeführt. Vielmehr werden nicht nur aktuelle Daten aus den Datenquellen gespeichert, sondern die gesamte Historie dieser Daten über einen längeren Zeitraum hinweg. Dies ermöglicht Analysen in Abhängigkeit von der Zeit, was durch das Attribut “zeitabhängig” beschrieben werden soll. Dieser Aspekt muss wiederum bei der Datenmodellierung berücksichtigt werden.

Die mit dem zuletzt genannten Attribut beschriebene Zeitabhängigkeit ist auch wichtig, um den Data-Warehouse-Ansatz von anderen Ansätzen zur Integration heterogener Datenbestände abzugrenzen. Zu diesen alternativen Ansätzen zählen unter anderem föderierte Systeme [Wie92] [Wid95] [MK+99]. Diese ermöglichen aber in der Regel nur den Zugriff auf die aktuellen Daten der Quellsysteme, so dass die zeitabhängige Analysemöglichkeit verloren geht. Darüber hinaus unterscheiden sich die beiden Ansätze in der Rechenlast, die auf den Quellsystemen durch die Analyse der Daten erzeugt wird, den Möglichkeiten, die bestehen, um die Laufzeiten von analyse-orientierten Anfragen zu verbessern und einer Reihe anderer Merkmale, die im Rahmen dieser Arbeit nicht von Bedeutung sind.

Unter einem Data Warehouse wird im Allgemeinen nicht nur das DWDBS verstanden, sondern der Begriff umfasst die in Abbildung 1 dargestellten Basiselemente [Kim96] [CD97] [Gar98] [KR+98]. Die Abbildung ist an die Darstellung in [KR+98] angelehnt. Sie zeigt neben der bereits erwähnten Data-Warehouse-Datenbank und den Quellsystemen (Source Systems) auch den Bereich zur Auf-

bereitung der Daten aus diesen Quellsystemen (Data Staging Area). Zu diesem Bereich gehören alle Komponenten, die an der Extraktion (extract) der Daten aus den Quellsystemen, deren Transformation sowie dem Laden (Populate, replicate, recover) der transformierten Daten in die DWDB (Data Warehouse Database) beteiligt sind. Auf Grund der Anfangsbuchstaben der Bezeichnungen der einzelnen Prozessschritte werden diese Komponenten häufig auch als *ETL-Komponenten* und der zugehörige Prozess als *ETL-Prozess* bezeichnet [BG01]. Zur Transformation gehören hierbei alle Schritte, die notwendig sind, um die Quelldaten in eine für das Data Warehouse geeignete Form zu überführen. Dies beinhaltet z.B. die Beseitigung von Fehlern in den Daten, das Entfernen von Duplikaten, die standardisierte Darstellung und Codierung sowie die Berechnung zusätzlicher Daten, die auf mehreren unterschiedlichen Quellen basieren. Die so aufbereiteten Daten werden in die DWDB (Data Warehouse Database) geladen und mit Hilfe unterschiedlichster Werkzeuge analysiert. Nur mit Hilfe dieser Werkzeuge erhalten die Endbenutzer Zugriff auf den konsolidierten Datenbestand (End User Data Access). Die in einem Data Warehouse für Analysen bereitgestellten Daten werden in dieser Arbeit als *Basisdaten* bezeichnet. Daten, die durch Vorverarbeitungsschritte oder als Zwischenergebnisse einzelner Analysewerkzeuge entstehen, werden hiervon abgegrenzt.

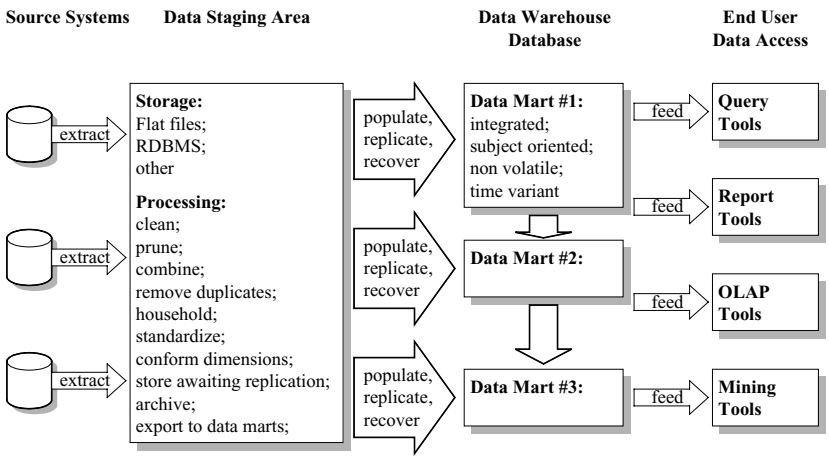

Abbildung 1: Basiselemente eines Data Warehouse

Die Data-Warehouse-Datenbank wird häufig nicht als eine homogene Einheit gesehen, sondern in mehrere logische Bereiche unterteilt, die sogenannten *Data Marts*. Ein Data Mart umfasst einen thematischen oder prozessorientierten Ausschnitt aus dem globalen Data Warehouse. Neben der logischen Trennung kann dies auch eine physisch getrennte Datenhaltung bedeuten [Gar98] [Wes01].

2.2 Online Analytical Processing

OLAP ermöglicht die mehrdimensionale Analyse von Daten, die in einer Data-Warehouse-Datenbank vorliegen sowie die meist grafische Darstellung der Analyseergebnisse [KR+98] [DS+98]. Die Dimensionen entsprechen hierbei den unterschiedlichen Blickwinkeln, aus denen die Anwender vorhandene Daten typischerweise analysieren, wie z.B. die Zeit oder geographische Bezüge. Die verschiedenen Dimensionen weisen wiederum eine interne Struktur auf. Sie umfassen mehrere, hierarchisch strukturierte Attribute. Beispielsweise besteht eine Zeitdimension häufig aus den Attributen Tag, Woche, Monat und Jahr. Die für die Analyse relevanten Daten werden als Fakten bezeichnet, die auf die einzelnen Dimensionen bezogen vorliegen. Durch die Betrachtung der Fakten entlang verschiedener Dimensionen ergibt sich im dreidimensionalen Fall die Struktur eines Datenwürfels. Unabhängig von der physischen Speicherungsstruktur wird die Metapher eines *Datenwürfels* (engl. Data Cube) für die Verdeutlichung der typischen OLAP-Operationen verwendet. Diese ermöglichen die Navigation auf den Daten sowie deren Selektion und Aggregation. Die Menge der OLAP-Operationen ist nicht fest definiert. Die folgende Beschreibung konzentriert sich auf die wichtigsten Operationen, die sowohl in wissenschaftlichen Publikationen als auch in den einzelnen OLAP-Werkzeugen als Basisoperationen identifiziert werden können [AGS97] [CD97] [Sho97] [PR00] [HK01].

Mit einer Selektion wird die Analyse auf einzelne Attributwerte innerhalb einer Dimension fokussiert. Bezieht sich die Einschränkung auf eine einzige Dimension, so wird die Operation als *slice* bezeichnet. In Abbildung 2(b) ist diese für den dreidimensionalen Fall dargestellt. Durch die Selektion in der Dimension A wird eine Ebene des Datenwürfels von Abbildung 2(a) herausgeschnitten. Wird die Analyse in zwei oder mehr Dimensionen auf einzelne Werte eingeschränkt, so spricht man von der Operation *dice*. Abbildung 2(c) zeigt diese am Beispiel einer Einschränkung auf allen drei Dimensionen. Das Ergebnis ist ein Teilwürfel der ursprünglichen Daten.

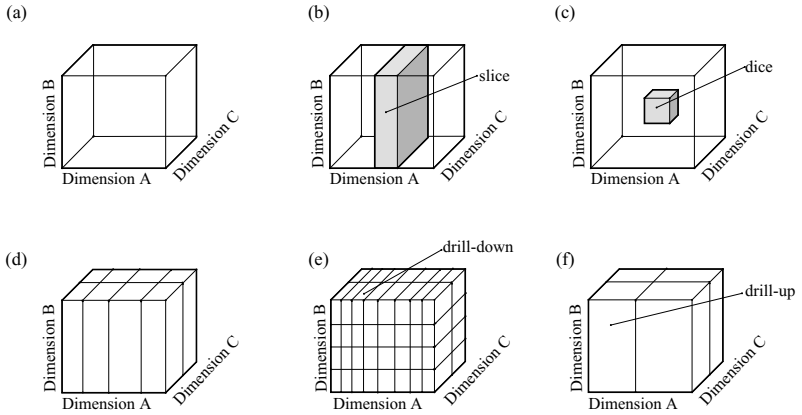

Abbildung 2: Typische OLAP-Operationen

Die Operation *drill-down* ermöglicht eine Navigation auf den Daten, indem sie zu dem betrachteten Datenwürfel zusätzliche Detailinformation liefert. Dies kann einerseits durch das Absteigen innerhalb der Hierarchie einer Dimension erfolgen. Andererseits können die Daten auch bezüglich einer oder mehrerer zusätzlicher Dimensionen aufgeschlüsselt werden. Die gegenteilige Operation, als *drill-up* oder *roll-up* bezeichnet, ermöglicht den Wechsel auf eine höhere Hierarchiestufe innerhalb einer Dimension bzw. das Eliminieren einer Dimension des Datenwürfels. Beides ist mit einer Aggregation der Daten verbunden. Drill-down ist in Abbildung 2(e) und drill-up in Abbildung 2(f) dargestellt. Ausgangspunkt ist jeweils der Datenwürfel aus Abbildung 2(d).

Eine weitere Operation, die sich auf die Darstellung der Daten bezieht, ist *pivot*. Diese ermöglicht den Wechsel der Achsen eines Datenwürfels, so dass dieser mit einer anderen Orientierung im Raum dargestellt wird.

Der Begriff OLAP wurde von E. F. Codd geprägt [CCS93]. Codd charakterisiert OLAP mit der Hilfe von 12 Regeln. Danach müssen OLAP-Werkzeuge die folgenden Eigenschaften aufweisen:

Regel 1: Sie müssen Benutzern eine mehrdimensionale Sicht auf die Daten ermöglichen, wobei die Dimensionen und die hierarchischen Bezie-

hungen innerhalb der Dimensionen entsprechend der anwendungsspezifischen Sicht auf die Daten gewählt werden können.

Regel 2: Die durch OLAP-Werkzeuge bereitgestellten Analysemöglichkeiten sollen in andere Werkzeuge zur Datenanalyse integriert werden können.

Regel 3: Alle relevanten Daten sollen in die Analyse mit einbezogen werden können, ohne dass die Benutzer im Detail wissen müssen, wo und in welcher Form die Daten abgelegt sind.

Regel 4: Die Effizienz der Analysen darf nicht wesentlich von der Komplexität des zu Grunde liegenden Datenmodells beeinflusst werden. Insbesondere darf die Zahl der verwendeten Dimensionen keinen negativen Einfluss auf die Leistung des Systems haben.

Regel 5: Der Einsatz von OLAP-Werkzeugen muss in einer Client-Server-Umgebung möglich sein.

Regel 6: Die Struktur und die anwendbaren Operationen sind für alle Dimensionen identisch.

Regel 7: Das physische Schema des Werkzeugs und die verwendeten Zugriffsmethoden müssen auf das spezifische Analysemodell angepasst werden können.

Regel 8: Mehrere Benutzer müssen parallel unterstützt werden können.

Regel 9: Die OLAP-Werkzeuge sollen die notwendigen Operationen für die mehrdimensionale Analyse auf unterschiedlichen Hierarchiestufen der einzelnen Dimensionen automatisch bestimmen.

Regel 10: Die einzelnen Analysen der Daten sollen intuitiv und mit graphischer Unterstützung möglich sein.

Regel 11: Die Werkzeuge müssen eine flexible Darstellung der Analyseergebnisse ermöglichen.

Regel 12: Es dürfen keine Einschränkungen hinsichtlich der Zahl der Dimensionen und der Zahl der Attribute innerhalb einer Dimension durch das Werkzeug vorgegeben sein.

Für OLAP werden zwei Ansätze unterschieden, die auf einer unterschiedlichen Speicherungsstruktur der Daten basieren [CD97]. Einerseits können die zu analysierenden Daten in einer relationalen Struktur abgelegt sein. Dann bietet das

OLAP-Werkzeug den Anwendern eine multidimensionale Sicht auf die Daten. Dieser Ansatz wird als *Relational OLAP* (ROLAP) bezeichnet. Der alternative Ansatz ist *Multidimensional OLAP* (MOLAP). Hier spiegelt sich der mehrdimensionale Ansatz nicht nur in der Benutzungsschnittstelle der Anwendung, sondern auch in der Speicherungsstruktur wider. Die Daten werden in einem *multidimensionalen Datenbanksystem* (MDBS) als mehrdimensionale Felder abgelegt. Beide Ansätze werden z.B. in den Analysis Services von Microsoft kombiniert und als *Hybrid OLAP* (HOLAP) bezeichnet. Hier liegen die Daten zwar relational vor, Teile davon werden allerdings zusätzlich in mehrdimensionalen Strukturen abgelegt, um einzelne Analysen effizienter durchführen zu können. Typische Vertreter im Bereich ROLAP sind die Werkzeuge von MicroStrategy [Mic95] und BusinessObjects [Abe99]. Gängige MOLAP-Tools sind beispielsweise Hyperion Essbase [Hyp01a] und Oracle Express Server [Ora96].

2.3 Data Mining

Der zweite der in dieser Arbeit genauer betrachteten *Analyseansätze* für große Datenbestände wird als *Data Mining* bezeichnet. Das Ziel ist es hier, Muster und Zusammenhänge in Daten zu erkennen, die so vorab nicht bekannt sind und die für eine bestimmte Anwendung sinnvoll eingesetzt werden können. Hierbei müssen die Anwender ihren Informationsbedarf wesentlich weniger genau spezifizieren können, als dies bei anderen Analyseansätzen wie z.B. OLAP der Fall ist. OLAP unterstützt die Überprüfung von Hypothesen, die durch den Anwender aufgestellt wurden. Im Gegensatz dazu sollen Data-Mining-Werkzeuge möglichst selbständig neue Hypothesen aufstellen [HS94] [FPS96b] [HK01].

Data Mining basiert zum Teil auf Techniken aus dem Bereich der künstlichen Intelligenz (KI) und auf Methoden der Statistik [GM+97]. Beispielsweise werden Verfahren zur Clusterbildung in der KI bereits seit vielen Jahren entwickelt [MS83]. Noch weiter zurück reichen Arbeiten im Bereich Statistik, die sich mit statistischen Maßen zur Bewertung von Modellen beschäftigen, die im Data Mining häufig eingesetzt werden [Aka74] [Ris78] [Raf95]. Die technologische Entwicklung zu Beginn der 1990er Jahre machte es möglich große Datenmengen zu verwalten und die verfügbaren Analyseverfahren auf diese Datenbestände anzuwenden und weiterzuentwickeln. In dieser Zeit wurde der Begriff Data Mining geprägt. Mit diesem Begriff wird eine ganze Reihe von unterschiedlichen Ansätzen verbunden. Hierbei muss zwischen dem zu erstellenden *Data-Mining-Modell*, der hierfür gewählten *Repräsentation*, den Kriterien zur Bewertung von

Modellen und den *Data-Mining-Algorithmen* zur Ableitung der Modelle unterschieden werden [FPS96a].

Es gibt unterschiedliche Arten von Data-Mining-Modellen. Jedes kann bestimmte Muster und Zusammenhänge zu gegebenen Daten darstellen. Hier sollen zunächst die wichtigsten Modelle und deren Repräsentationen charakterisiert werden. Berücksichtigt werden hierbei nur solche Modelle, die in der weiteren Arbeit Verwendung finden.

Unter *Clustermodellen* versteht man die Unterteilung einer Menge von Objekten in mehrere Klassen. Hierbei sollen sich die Objekte, die derselben Klasse zugeordnet werden, möglichst ähnlich sein, während sie sich in ihren Merkmalen von den Objekten anderer Klassen möglichst stark unterscheiden sollen [GGR99]. Eine mögliche Repräsentation für ein Clustermodell beinhaltet für jedes gefundene Cluster die normierten Attributwerte des Objekts im Zentrum des Clusters [Dat00]. In Abbildung 3(a) ist diese Repräsentation von Clustern durch die zentralen Objekte dargestellt.

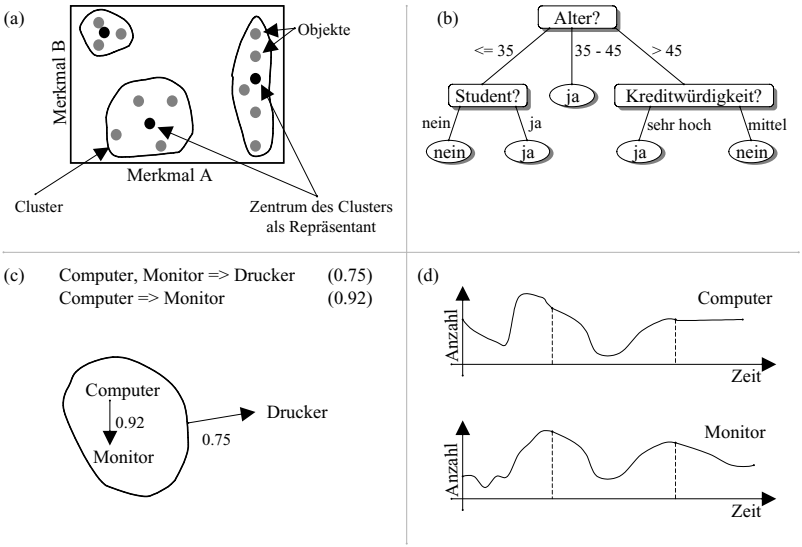

Abbildung 3: Repräsentationen für Data-Mining-Modelle

Bei den *Klassifikationsmodellen* werden dagegen die Regeln identifiziert, nach denen Objekte vorgegebenen Klassen zugeordnet werden können. Eine wichtige Repräsentationsform in diesem Bereich sind *Entscheidungsbäume* [SAM96]. Ein einfaches Beispiel für einen Entscheidungsbaum zeigt Abbildung 3(b). Hier wird dargestellt, wie Kunden auf Grund der Eigenschaften Alter, Beruf und Kreditwürdigkeit einer von zwei Klassen zugeordnet werden können. Die beiden Klassen umfassen Kunden, die einen Computer kaufen (Knoten mit der Bezeichnung 'ja') und solche die dies nicht tun (Knoten mit der Bezeichnung 'nein'). Alternativ können Klassifikationsmodelle auch als Wenn-Dann-Regeln, mathematische Formeln oder mit Hilfe neuronaler Netze dargestellt werden [HK01].

Einen Ansatz, um zu beschreiben, wie aus gegebenen Eigenschaften von Objekten andere, kontinuierliche Werte abgeleitet werden können stellen *Regressionsmodelle* dar. Diese Modelle werden durch lineare oder nicht-lineare Gleichungen repräsentiert.

Abhängigkeitsmodelle beschreiben signifikante Zusammenhänge sowohl zwischen Attributen von Objekten als auch zwischen Daten in einer Datenbank. Letztere können z.B. durch *Assoziationsregeln* repräsentiert werden [AM+96] [GGR99]. Eine textbasierte und eine grafisch unterstützte Möglichkeit solche Assoziationsregeln anzugeben ist in Abbildung 3(c) dargestellt. Die erste Regel beschreibt beispielsweise, dass 75% aller Kunden, die sowohl einen Computer als auch einen Monitor kaufen, bei demselben Einkauf auch noch einen Drucker erwerben.

*Z*eitabhängige Muster in Datenströmen werden durch *Sequenzmodelle* beschrieben. Hier ist es das Ziel, den Prozess, der die Daten generiert, zu beschreiben bzw. Abweichungen, Muster und Trends in einer Sequenz zu erkennen. Abhängig von den Ausgangsdaten können Sequenzmodelle durch Gleichungen oder durch geordnete Listen von Objekten repräsentiert werden [AFS93] [AS95]. Ein Beispiel für zwei Sequenzen, hier die Verkaufszahlen für Computer und Monitore, ist in Abbildung 3(d) gegeben. Für beide Sequenzen wurde ein gemeinsames Muster identifiziert.

Neben den unterschiedlichen Modellen und deren Repräsentationsmöglichkeiten müssen beim Data Mining die Kriterien festgelegt werden, nach denen unterschiedliche Modelle bewertet und verglichen werden können. Diese Kriterien werden teilweise von den Algorithmen zur Generierung der Modelle genutzt. Es

existiert ein breites Spektrum an Algorithmen, von denen jeder nur ein spezifisches Modell bzw. eine bestimmte Repräsentationsform abdeckt. Für jedes Modell oder jede einzelne Repräsentationsform kann es aber wiederum mehrere Algorithmen geben. Eine Übersicht verfügbarer Algorithmen gibt es z.B. in [FS+96] [HK01]. In dieser Arbeit wird außerdem der Begriff *Data-Mining-Verfahren* verwendet. Zu einem Data-Mining-Verfahren gehören alle Data-Mining-Algorithmen, die geeignet sind, eine spezifische Art von Data-Mining-Modell abzuleiten. Nach dieser Bezeichnung sind die *Clusterbildung*, die *Klassifikation*, die *Regressionsanalyse*, die *Abhängigkeitsanalyse* und die *Sequenzanalyse* die im Rahmen dieser Arbeit betrachteten Data-Mining-Verfahren. Ein spezifisches *Data-Mining-Werkzeug* bietet in der Regel mehrere Data-Mining-Verfahren an [Ibm99] [Ora00a] [Sas99] [Thi98].

Data Mining ist nicht als ein isoliert einsetzbarer Ansatz zur Analyse umfangreicher Datenbeständen zu sehen, sondern ist immer Bestandteil eines umfassenderen Prozesses. Dieser wird als *KDD-Prozess* (Knowledge Discovery in Databases) bezeichnet [BA94] [FPS96a]. Er setzt sich aus den in Abbildung 4 dargestellten Schritten zusammen und reicht von der Auswahl der relevanten Daten hin zur Gewinnung des erforderlichen Wissens. Die Selektion der für die durchzuführende Analyse notwendigen Daten stellt den ersten Schritt in diesem Prozess dar (Selection). Grundlage hierfür ist eine detaillierte Untersuchung der Analyseziele. Die beiden folgenden Schritte (Preprocessing, Transformation) beinhalten die Aufbereitung der ausgewählten Daten. Zur Vorverarbeitung gehört hier unter anderem das Entfernen von Rauschen und Ausreißern aus den Daten sowie die Definition und Berechnung abgeleiteter Attribute. Demgegenüber liegt der Schwerpunkt bei der folgenden Transformation darauf, die Daten in eine Repräsentationsform zu bringen, die für die einzusetzenden Data-Mining-Verfahren geeignet ist. Hierbei werden auch Attribute eliminiert, die für diese Verfahren nicht relevant sind. Es folgt der eigentliche Data-Mining-Schritt, der sowohl die Auswahl und Parametrisierung der geeigneten Data-Mining-Verfahren als auch

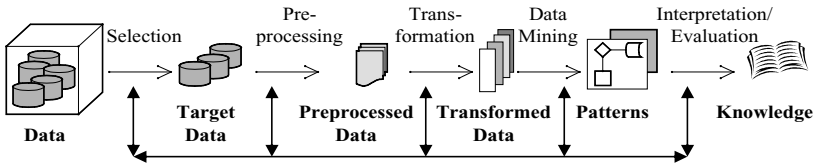

Abbildung 4: Der KDD-Prozess

die Bildung und Bewertung von Modellen umfasst. In der abschließenden Phase (Interpretation/Evaluation) werden die erstellten Modelle interpretiert und evaluiert, bevor sie schließlich auf neue Daten angewandt werden können.

Alle Phasen des KDD-Prozesses, die vor dem eigentlichen Data Mining liegen, beziehen sich auf Aufgaben, die durchgeführt werden müssen, wenn Daten aus heterogenen Quellen zu Analysezwecken integriert, aufbereitet und bereitgestellt werden sollen. Wie in Abschnitt 2.1 beschrieben, fallen vergleichbare Aufgabenstellungen auch beim Aufbau eines Data Warehouse an. In der weiteren Arbeit wird deutlich werden, dass auch bei einem vorhandenen Data Warehouse im Rahmen des KDD-Prozesses nicht alle Schritte der Datenaufbereitung entfallen können.

Der Begriff Data Mining wird in dieser Arbeit durchgängig zur Beschreibung einer Phase des KDD-Prozesses verwendet. In der Literatur und insbesondere im kommerziellen Bereich werden die Begriffe Data Mining bzw. Data-Mining-Prozess häufig aber auch als äquivalent zu KDD-Prozess angesehen [HK01].

2.4 Weitere wichtige Begriffe

Der Begriff *Business Intelligence* wird meist als Oberbegriff für Software und Prozesse verwendet, die der eher strategieorientierten Informationsgewinnung dienen. Er ist nicht klar definiert und taucht in erster Linie im wirtschaftlichen und industriellen Umfeld auf. Dort wird mit Business Intelligence in der Regel sowohl die Modellierung, der Aufbau und Betrieb eines Data Warehouse als auch die verschiedenen Ansätze zur Analyse von Daten in einem Data Warehouse bezeichnet. Im Rahmen dieser Arbeit wird der Begriff Business Intelligence nicht weiter verwendet, da er eine weniger klare Abgrenzung erlaubt, als die bereits eingeführten Begriffe.

Bereits sehr früh in der Entwicklung von Computersystemen wurde über deren Einsatz zur Unterstützung der Entscheidungsfindung nachgedacht. Die Grundlage hierfür waren einerseits Studien zu Entscheidungsprozessen und andererseits die Entwicklung interaktiver Rechnersysteme. Auf dieser Grundlage konnten Systeme entwickelt werden, die die Benutzer im Rahmen von Entscheidungsprozessen unterstützen. Dies betrifft sowohl den Zugriff auf die für eine Entscheidung relevanten Daten und deren Analyse als auch die Modellbildung und Bewertung der Entscheidungsalternativen. In den 1970er Jahren wurde für

diese Systeme der Begriff *Decision Support System* (DSS) geprägt [Mor71] [Mor72] [BHW81] [SpCa82]. Die frühen Systeme waren vorwiegend Einzelbenutzersysteme auf der Basis geringer Datenmengen. Systeme mit einer ähnlichen Zielsetzung wurden unter dem Stichwort *Management-Informationssysteme* (MIS) betrachtet. Das Spektrum dieser Systeme ist allerdings größer, da hier der Schwerpunkt nicht auf der Entscheidungsunterstützung, sondern ganz allgemein auf der Bereitstellung von Informationen liegt. In den 1980er Jahren wurden besonders Management-Informationssysteme betrachtet, die die Einbeziehung einer breiteren Datenbasis und die Unterstützung mehrerer Benutzer im Rahmen eines Entscheidungsprozesses berücksichtigten [Roc79]. Heute werden ein Data Warehouse sowie Analyseansätze wie OLAP und Data Mining als zentrale Bestandteile eines Decision Support Systems gesehen [PKB98].

Andere Klassen von Werkzeugen für den Zugriff auf und die Analyse von Daten werden allgemein als *Anfrage-Werkzeuge* (engl. Ad Hoc Query Tools) und *Report-Werkzeuge* (engl. Report Writer Tools) bezeichnet. Auch die *Tabellenkalkulation* fällt in diesen Bereich. Einfache Anfrage-Werkzeuge haben lediglich das Ziel, Benutzer bei der Formulierung von Anfragen zu unterstützen. Dies kann z.B. dadurch erfolgen, dass eine Anfrage mit grafischer Unterstützung eingegeben wird. Report-Werkzeuge und Tabellenkalkulations-Programme bieten darüber hinaus umfangreiche Möglichkeiten, die Anfrageergebnisse zu verarbeiten, indem z.B. Berechnungen darauf ausgeführt werden. Sie ermöglichen zusätzlich die grafische Darstellung der Analyseergebnisse. Die Technologie in diesem Bereich ist nicht neu und bietet gegenüber den Ansätzen OLAP und Data Mining weniger komplexe Analysemöglichkeiten. In der Regel sollte sie bereits in die Umgebung eines Data Warehouse integriert sein bzw. sollte diese Integration ohne grundlegende technische Probleme möglich sein. Deshalb werden die genannten zusätzlichen Ansätze zur Analyse und Präsentation von Daten im Rahmen dieser Arbeit nicht weiter berücksichtigt.

2.5 Anwendungsszenarien

In dieser Arbeit werden durchgängig zwei *Anwendungsszenarien* genutzt, um wichtige Aspekte zu verdeutlichen. Beide Anwendungsszenarien werden in diesem Abschnitt vorgestellt. Dies umfasst sowohl Erläuterungen zum jeweiligen Datenbestand als auch primäre Ziele, die mit dem Einsatz von Online Analytical Processing und Data Mining verfolgt werden können. Für beide Anwendungsszenarien wurde ein Handelsunternehmen gewählt. Dies ist sinnvoll, da für den

Handelsbereich die umfangreichsten Informationen zum Einsatz der hier betrachteten Analyseansätze vorliegen. Einerseits waren Handelsunternehmen Vorreiter beim praktischen Einsatz dieser Technologien, andererseits ist auch in der wissenschaftlichen Literatur der Einsatz von OLAP und Data Mining für Fragen des Handels ein Schwerpunkt der Betrachtungen.

2.5.1 Einzelhandelskette ToPCHain (TPCH)

Im ersten Anwendungsszenarium wird die hypothetische Einzelhandelskette ToPCHain betrachtet. Die im Data Warehouse dieser Handelskette für Analysen verfügbaren Daten beruhen auf dem Datenbestand des TPC-H-Benchmarks [Tra99]. Das logische Datenbankschema aus der Benchmarkspezifikation ist in Abbildung 5 dargestellt. Die Tabelle *Part* beinhaltet die durch ToPCHain vertriebenen Produkte mit ihren Eigenschaften, wie die vollständige Bezeichnung

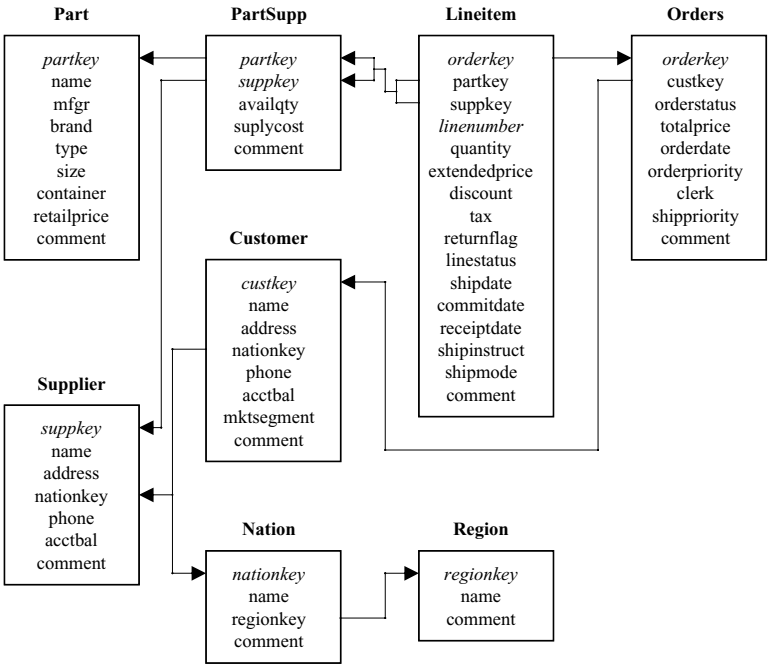

Abbildung 5: Schema zum Anwendungsszenarium ToPCHain

(*name*), die Größe (*size*) oder den Markennamen (*brand*). Diese Produkte werden von unterschiedlichen Zulieferern bezogen, die in der Tabelle *Supplier* festgehalten sind. Neben wichtigen Angaben wie Adresse (*address*) und Telefonnummer (*phone*), wird dort für jeden Zulieferer auch ein Verweis auf die Tabelle *Nation* festgehalten. Dies ermöglicht, kombiniert mit der Tabelle *Region*, die geographische Gliederung von Zulieferern und Kunden. Die Kunden selbst werden mit ihren Angaben in der Tabelle *Customer* gespeichert. Ähnlich wie bei Zulieferern wird hier auch ein aktueller Kontostand (*acctbal*) festgehalten. Kunden werden zusätzlich einem Marktsegment (*mktsegment*) zugeordnet. In der Tabelle *PartSupp* ist vermerkt, welcher Lieferant welches Produkt in welcher Menge (*availqty*) und zu welchem Preis (*suplycost*) liefern kann. Die Bestellungen der Kunden sind unter anderem mit ihrem Status (*orderstatus*), dem Bestelldatum (*orderdate*) sowie dem zuständigen Mitarbeiter (*clerk*) in der Tabelle *Orders* abgelegt. Die Lieferungen werden in der Tabelle *Lineitem* gespeichert. Hier sind neben der Menge (*quantity*) und den kostenorientierten Attributen (*extendedprice*, *discount*, *tax*) auch Informationen zu den Lieferzeitpunkten (*shipdate*, *commitdate*, *receiptdate*) und zum Status der Lieferung (*linestatus*, *shipinstruct*, *shipmode*) enthalten.

2.5.2 Pharmagroßhandel Phagroh AG

Das Anwendungsszenarium Phagroh umfasst ein hypothetisches Handelsunternehmen im Bereich Pharmagroßhandel, das aber die typischen Merkmale solcher Großhändler in der Bundesrepublik Deutschland aufweist [Kim00]. Die Phagroh AG übernimmt eine Mittlerfunktion zwischen den verschiedenen Pharmaherstellern und den Apotheken und Drogerien. Einerseits sind diese Einzelhändler auf Grund des breiten Angebots an Artikeln (mehr als 250.000) nicht in der Lage, diese in ausreichendem Maße vorrätig zu halten. Auf der anderen Seite erschwert die Tatsache, dass es in der Bundesrepublik Deutschland mehr als 20.000 Apotheken gibt, eine direkte Betreuung der Einzelhändler durch die Hersteller. Da der Zusammenschluss von Apotheken zu Einzelhandelsketten gesetzlich verhindert wird, kann die notwendige Mittlerfunktion zwischen Herstellern und Apotheken nur durch eigenständige Unternehmen wie die Phagroh AG übernommen werden.

Das Sortiment der Phagroh AG kann in drei grundlegende Bereiche eingeteilt werden. Das *Rezeptsortiment* umfasst hierbei alle Artikel, die nur gegen Vorlage eines Rezeptes erhältlich sind. Dies sind ca. 70% der Produkte. Die restlichen

Artikel werden unterteilt in das *Sichtwahlsortiment* und das *Griffwahlsortiment*. Zum Sichtwahlsortiment gehören alle Produkte, die zwar ohne Rezept, aber nur in Apotheken verkauft werden dürfen, während das Griffwahlsortiment alle Produkte umfasst, die auch in Drogerien oder Supermärkten angeboten werden können. Produkte aller Sortimente beziehen die Apotheken in der Regel nicht nur von der Phagroh AG, sondern auch von deren Konkurrenten. Dadurch versuchen die einzelnen Apotheken die Verfügbarkeit der Medikamente zu erhöhen und Lieferzeiten zu minimieren. Die Bestellungen werden vom Warenwirtschaftssystem der einzelnen Apotheken elektronisch an die Phagroh AG übermittelt. Da vor allem Medikamente aus dem Rezeptsortiment die Patienten häufig innerhalb weniger Stunden erreichen müssen, sind unter Umständen bis zu sechs Lieferungen pro Tag und Apotheke notwendig.

Alle Angaben, die zu den Bestellungen und Lieferungen bei der Phagroh AG eingehen, werden in einem zentralen Data Warehouse verwaltet. Ein für diese Arbeit relevanter Ausschnitt aus dessen Struktur ist in Abbildung 6 wiedergegeben. In der Tabelle *Customer* werden hierbei alle wichtigen Informationen zu den belieferten Apotheken und Drogerien abgelegt. Die Tabelle *Dc* enthält Angaben zu den einzelnen Niederlassungen, während alle wichtigen Attribute der Artikel in der Tabelle *Article* verwaltet werden. Informationen zur Zeit werden in der Tabelle *Time* verwaltet. Im Zentrum des Schemas steht die Tabelle *CustomerTur-*

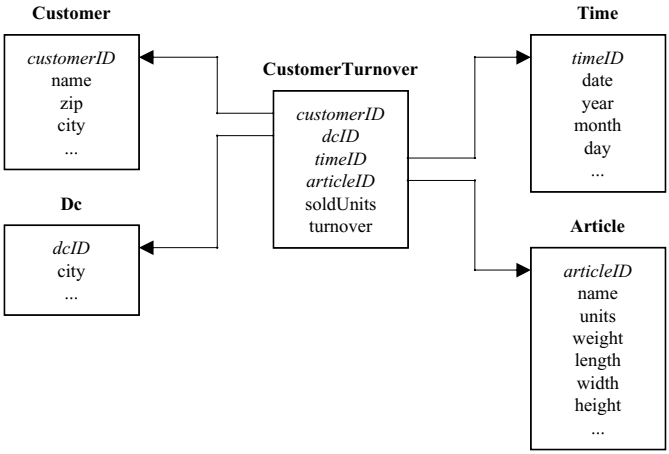

Abbildung 6: Schema zum Anwendungsszenarium Phagroh AG

nover. Sie bündelt alle wichtigen Informationen zu den eingegangenen Bestellungen und erfolgten Lieferungen. Dies sind einerseits Verweise auf die zuvor beschriebenen Tabellen, andererseits sind die Daten wie Bestellmengen (*soldUnits*), Umsätze (*turnover*) direkt in dieser Tabelle enthalten.

3 Motivation für die Integration von Data Mining und OLAP

In diesem Kapitel wird das Leistungsspektrum der beiden Analyseansätze Data Mining und OLAP näher untersucht und verglichen. Im Vordergrund stehen hierbei charakteristische Merkmale der beiden Analyseansätze sowie die Klassen von Fragestellungen, die durch die jeweilige Technik bearbeitet werden können. Diese sollen jeweils durch konkrete Fragestellungen aus den gewählten Anwendungsszenarien näher erläutert werden. Im Ergebnis wird sich zeigen, dass nur durch einen integrierten Einsatz von OLAP und Data Mining den Informationsanforderungen realer Anwendungen Rechnung getragen werden kann. Welche Unterstützung existierende Systeme in diesem Bereich bieten und in welchen Bereichen Verbesserungen notwendig sind, wird in den letzten beiden Abschnitten dieses Kapitels analysiert.

3.1 Charakterisierung des Leistungsspektrums von Analyseansätzen

Um das Leistungsspektrum der hier betrachteten Analyseansätze vergleichend bewerten zu können, wird zunächst ein Schema vorgestellt, das die Charakterisierung unterschiedlicher Analysemöglichkeiten unterstützt. Für die einzelnen Analyseansätze werden die folgenden Merkmale betrachtet:

- der Zeitbezug der Analysen,
- die Struktur der Ergebnisse und
- die Art der Operationen.

Die möglichen Merkmalsausprägungen sind in Abbildung 7 dargestellt. Das erste wichtige Merkmal ist der *Zeitbezug*. Hier ist zu unterscheiden, ob Analysen lediglich zusätzliche Informationen zu den vor der Durchführung der Analyse erfassten Daten liefern. Diese Art der Analyse wird hier als *deskriptiv* bezeichnet. Alternativ kann eine Analyse auf der Basis verfügbarer Daten auch Informationen für die Zukunft vorhersagen. Dieser Zeitbezug wird hier als *prognostizierend* bezeichnet.

Darüber hinaus muss die *Ergebnisstruktur* der bei einer Datenanalyse erzielten Ergebnisse näher betrachtet werden. Im Rahmen dieser Arbeit spielen hier zwei

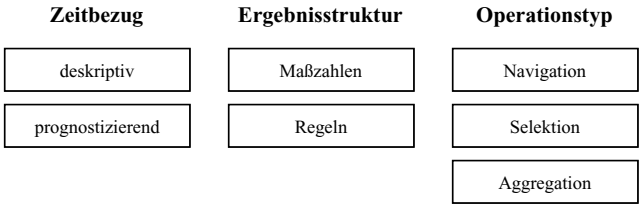

Abbildung 7: Merkmale zur Charakterisierung von Analyseansätzen

mögliche Merkmalsausprägungen eine Rolle. Das Analysewerkzeug kann einerseits Ergebnisse als reine *Maßzahlen* liefern. Dies ist beispielsweise der Fall, wenn Daten durch Aggregation verdichtet werden, um die Datenmenge für den Benutzer im Umfang soweit zu verringern, dass eine einfache Interpretation oder Weiterverarbeitung möglich ist. Andererseits können aus den Daten auch *Regeln* abgeleitet werden. Hier ist es dann das Ziel, Zusammenhänge in den Daten zu erkennen, die sich weder durch die Betrachtung der Einzeldatensätze noch durch deren Aggregation einfach erschließen.

Als letztes Merkmal wird hier der *Operationstyp* betrachtet. Jeder Analyseansatz bietet einen typischen Satz von Operationen auf den Daten an. Die einfachste Variante des Operationstyps ist die *Navigation*. Sie ermöglicht es den Benutzern, sich innerhalb des Datenbestandes von einem Ausschnitt der Daten zu den damit verbundenen Daten zu bewegen. Innerhalb einer Dimension können Daten damit beispielsweise auf unterschiedlichen Hierarchiestufen betrachtet werden. Die *Selektion* ist der zweite hier betrachtete Operationstyp. Sie umfasst die inhaltsbezogene Auswahl von Daten für detailliertere Analysen sowie die reine Suche in Daten. Darüber hinaus werden hier noch Operationen vom Typ *Aggregation* unterschieden. Dieser Typ umfasst sowohl die einfache Anwendung von Aggregatfunktionen auf Daten als auch eine komplexere Verdichtung der Daten. Dies kann z.B. das Ableiten von Regeln sein, mit deren Hilfe der Datenbestand näher beschrieben wird.

Auf der Grundlage dieser als relevant identifizierten Merkmale kann das Leistungsspektrum sowohl von OLAP als auch von Data Mining beschrieben werden. Bei dieser Beschreibung steht die grundsätzliche Zielrichtung der beiden Ansätze im Vordergrund. Die Tatsache, dass einzelne Werkzeuge in der zur Verfügung gestellten Funktionalität über den beschriebenen Bereich hinausgehen,

wird hier nicht berücksichtigt, da es für einen Vergleich der grundlegenden Ansätze nicht von Bedeutung ist.

3.2 Leistungsspektrum von OLAP

OLAP-Technologie wird für deskriptive Analysen eingesetzt, deren Ziel es in erster Linie ist, Daten entlang unterschiedlicher Dimensionen so weit zu verdichten, dass Informationen auf der jeweils gewünschten Betrachtungsebene komprimiert bereitgestellt werden können [CCS93]. Dies wird auch durch die in Abschnitt 2.2 beschriebenen wichtigen OLAP-Operationen zum Ausdruck gebracht [KR+98]. Die Operationen *slice* und *dice* bedeuten eine Selektion von Daten, während bei *drill-up* und *drill-down* der Navigationsaspekt im Vordergrund steht. Die letztgenannten Operationen können aber auch mit einer Aggregation der Daten einhergehen. Es zeigt sich, dass OLAP ein breites Spektrum an Operationen auf Daten liefert. Dagegen gibt es Einschränkungen sowohl bei der Art der Ergebnisse, die bei der Verdichtung der Daten erzielt werden können, als auch beim zeitlichen Bezug. Der Fokus ist hier die Beschreibung und Verdichtung der vor dem Analysezeitpunkt gesammelten Daten und nicht die Prognose für die Zukunft. Die Ergebnisse werden in der Form von Maßzahlen geliefert und nicht als aus den Daten abgeleitete Regeln und Muster. In Abbildung 8 wird dies im Überblick dargestellt, wobei das Kästchen für eine Merkmalsausprägung im linken Teil grau hinterlegt ist, wenn OLAP die entsprechende Eigenschaft aufweist.

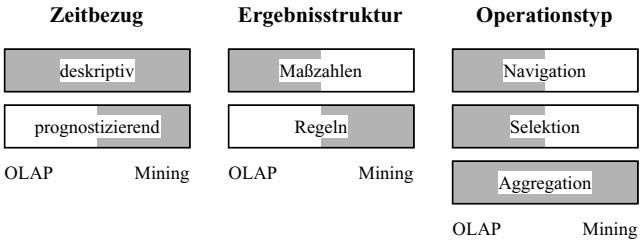

Abbildung 8: Charakterisierung von OLAP und Data Mining

3.3 Leistungsspektrum von Data Mining

Betrachtet man für Data Mining zunächst den Zeitbezug, dann kann hier sowohl ein deskriptiver als auch prognostizierender Charakter festgestellt werden [HK01]. Beispielsweise können Klassifikationsmodelle einerseits so interpretiert werden, dass sie die Kriterien liefern, auf Grund derer ein bestimmtes Objekt einer Klasse zugeordnet wurde. Andererseits kann diese Information dazu genutzt werden, die Klasse eines bislang nicht betrachteten Objektes zu bestimmen [HS94]. Data-Mining-Ergebnisse sind in ihrer Struktur meist Muster und Regeln in den unterschiedlichsten Repräsentationsformen. Einige wichtige Repräsentationsformen wurden bereits in Abschnitt 2.3 erwähnt. Ergebnisse in Form reiner Maßzahlen spielen beim Data Mining keine Rolle. Bleibt als letztes zu betrachtendes Merkmal der Operationstyp, der von Data Mining unterstützt wird. Die einzige hier relevante Operation ist die Aggregation. Data Mining unterstützt diesen Typ, da es die Verdichtung von Daten zu überschaubaren Informationseinheiten ermöglicht. Dagegen ist weder die Navigation auf den Daten noch die Selektion relevanter Daten hier von Bedeutung. Diese Operationen sind zwar Bestandteil des KDD-Prozesses insgesamt, sie liegen aber nicht im Fokus des als Data Mining bezeichneten Schrittes dieses Prozesses. Dieser Zusammenhang wurde bereits in Abschnitt 2.3 näher erläutert [BA94] [FPS96a].

3.4 Typische OLAP-Analysen

In diesem Abschnitt wird die prinzipielle Funktionalität von OLAP anhand einiger Analysen, die für die in dieser Arbeit beschriebenen Anwendungsszenarien relevant sind, verdeutlicht. Alle nachfolgend aufgeführten Analysen sind durchnummeriert und werden in der gesamten Arbeit mehrfach verwendet. Für jede der Analysen wird erläutert, welchen Anwendungsbereichen für den Handel diese primär zugeordnet werden können. Anhand dieser Zuordnung kann die Relevanz für die Praxis aufgezeigt werden. Es handelt sich hierbei nicht um eine vollständige Übersicht der Anwendungsbereiche, die für den Handel von Bedeutung sind. Für detaillierte Erläuterungen zu den Anwendungsbereichen muss an dieser Stelle auf die Literatur verwiesen werden [Wag00] [Bou01].

Analyse 1 Welches sind die Top-Produkte der Einzelhandelskette ToPCHain, deren Absatz in den Monaten M gegenüber dem jeweiligen Vormonat am stärksten gestiegen ist?

Diese Fragestellung kann dem Anwendungsbereich *Merchandise Management* zugeordnet werden, in dem es um die Informationen zu Waren und deren Lagerung geht. Da sie sich auf eine Verdichtung der Daten der Vergangenheit beschränkt handelt, es sich um eine für OLAP typische Fragestellung. Als Operation taucht hier eine Selektion zur Bestimmung der relevanten Monate auf.

Analyse 2 Wie verteilt sich bei der Einzelhandelskette ToPCHain der Absatz verschiedener Produkte P in einem Zeitraum Z auf die einzelnen Länder, in denen Kunden angesiedelt sind?

Diese Fragestellung kann sowohl den Anwendungsbereichen *Sales Analysis* als auch dem *Category Management* zugeordnet werden. In beiden Fällen geht es darum, die Verkäufe nach Produkten bzw. nach Produktkategorien näher zu analysieren. Auch diese Fragestellung hat rein deskriptiven Charakter und ist damit eindeutig als OLAP-Fragestellung qualifiziert. Als typische OLAP-Operation taucht hier lediglich eine Selektion bezüglich der relevanten Produkte auf.

Analyse 3 Welches sind die Top-Kunden der ToPCHain Handelskette, die in einem Zeitraum Z einen Mindestumsatz U erbracht haben und bei denen die Standardabweichung des Umsatzes in diesem Zeitraum am geringsten ist?

Hierbei handelt es sich um eine Fragestellung aus dem *Customer Relationship Management* (CRM), einem Bereich, in dem es in erster Linie um ein genaues Verständnis der Kunden und ihres Kaufverhaltens geht. Dies ermöglicht dem Handelsunternehmen die individuelle Anpassung der Service-Strategien. Das Ziel der Analyse 3 ist es, umsatzstarke Kunden mit möglichst geringen Änderungen des Umsatzes zu identifizieren. Es handelt sich hier um eine typische OLAP-Analyse, da sie Maßzahlen liefert, die das Verkaufsgeschehen in der Vergangenheit charakterisieren. Daneben sind die für OLAP typischen Operationen Selektion, Navigation und Aggregation enthalten. Eine Selektion liegt vor, da die Analyse auf den Zeitraum Z und einen Mindestumsatz U eingeschränkt wird. Auf Grund der Aufsummierung der Daten für einen vorgegebenen Zeitraum liegt eine Aggregation vor. Eine Navigation ergibt sich, da die Daten auf unterschiedlichen Hierarchiestufen der Zeitdimension betrachtet werden können.

Auch die folgende sehr einfache Fragestellung kann dem Customer Relationship Management zugerechnet werden.

Analyse 4 Mit welchen Kunden wurde in den Filialen in der Stadt S an den einzelnen Tagen im Monat M des Jahres J mehr als 10000 Euro Umsatz gemacht?

Eine SQL-Anfrage, die diese Information auf der Basis des Schemas in Abbildung 6 für die Filialen in München für den Dezember 2001 bereitstellt, ist in Abbildung 9 zu finden.

```
SELECT    c.name, t.year, t.month, t.day, sum(ct.turnover)
FROM      CustomerTurnover ct, Customer c, Time t, Dc d
WHERE     ct.customerID = c.customerID
AND       ct.timeID = t.timeID
AND       ct.dcID = d.dcID
AND       t.year = 2001
AND       t.month = 12
AND       d.city = 'München'
GROUP BY c.name, t.year, t.month, t.day
HAVING    sum(ct.turnover) > 10000;
```

Abbildung 9: OLAP-Anfrage für die Phagroh AG

3.5 Typische Data-Mining-Analysen

Bei den nachfolgend erläuterten Analysen handelt es sich durchweg um Fragestellungen, die dem Bereich Data Mining zugeordnet werden können. Auch hier wird neben der reinen Nennung der Fragestellungen erläutert, in welchen Anwendungsbereich sie einzuordnen sind [Kim00] [Bou01].

Analyse 5 Welche Gruppen ähnlicher Kunden von ToPCHain können auf der Basis der Angaben über Marktsegment, Kontostand, Gesamtumsatz, maximalen Einzelumsatz, Handelsregion, Nation, Gesamtstückzahl bestellter Produkte sowie der Gesamtzahl der Bestellungen des einzelnen Kunden identifiziert werden?

Hierbei handelt es sich um eine typische Fragestellung aus dem Bereich *Customer Relationship Management*. Gruppen ähnlicher Kunden sollen identifiziert werden, um diese ihren Bedürfnissen entsprechend gezielt anzusprechen. Die als Ergebnis gelieferten Cluster charakterisieren einerseits die bisherigen Kunden des Handelsunternehmens. Andererseits können auch neue Kunden anhand der gefundenen Regeln den Kundengruppen zugeordnet werden. Die Vorgehensweise hat damit nicht nur deskriptiven, sondern auch prognostizierenden Charakter.

Analyse 6 Welche Produkte und Marken kaufen Kunden des Handelsunternehmens ToPCHain häufig zusammen?

Hierbei handelt es sich um die klassische Fragestellung der *Market Basket Analysis*, deren Ziel es ist, typische Warenkörbe der Kunden zu identifizieren. Können hier stabile Muster identifiziert werden, dann lassen diese Rückschlüsse auf das zukünftige Kaufverhalten der Kunden zu. Die Information kann z.B. zur Angebotsgestaltung oder der Platzierung der Produkte genutzt werden.

Analyse 7 ToPCHain gewährt unterschiedliche Preisnachlässe. Diese sollen zu Rabattstufen zusammengefasst und die entscheidenenden Einflussfaktoren aus Angaben wie gelieferten Mengen, Transportart, Marktsegment usw. herausgefunden werden.

Das Ziel dieser Analyse ist die Erstellung eines Modells für die Gewährung von Preisnachlässen. Mit Hilfe dieses Modells kann dann ToPCHain auf der Basis verschiedener Auftrags- und Kundeneigenschaften die richtige Rabattstufe direkt bestimmen. Damit ist auch hier die Beschreibung und Modellierung der Vergangenheit nur der erste Schritt, um Prognosen für die Zukunft zu erstellen.

Analyse 8 Erstelle mit Hilfe der Daten über die Kunden der Phagroh AG und zusätzlichen demografischen Daten ein Modell, das den Einfluss dieser Daten auf den Umsatz einer Apotheke beschreibt.

Hintergrund dieser Fragestellung ist das Ziel, eine von Außendienstmitarbeitern vorgenommene, eher subjektive Einschätzung des Umsatzpotenzials von Apotheken durch eine objektivere Vorgehensweise zu ersetzen bzw. zu ergänzen.

Hier sollen Regeln und Muster identifiziert werden, mit deren Hilfe Daten in der Zukunft prognostiziert werden können. Somit handelt es sich um eine Fragestellung, die eindeutig dem Data Mining zugeordnet werden kann.

Analyse 9 Gibt es Kunden der Phagroh AG, die gemeinsame Sortimentsschwerpunkte aufweisen, d.h. die in demselben Teil des Sortiments überdurchschnittlich hohe Umsätze erzielen? Durch welche Eigenschaften lassen sich solche Kundengruppen charakterisieren?

Das primäre Ziel dieser Fragestellung aus dem Bereich *Customer Relationship Management* ist es, gemeinsame Schwerpunkte einzelner Apotheken zu identifizieren, um diese speziell ansprechen zu können. Gesucht sind hier wieder Regeln, die die Gruppen von Apotheken mit ihren gemeinsamen Sortimentsschwerpunkten anhand demografischer und geografischer Informationen beschreiben.

3.6 Identifikation des Integrationsbedarfs

Als Zusammenfassung der vorangegangenen Abschnitte soll hier erläutert werden, warum eine integrierte Betrachtung der beiden Analyseansätze OLAP und Data Mining sinnvoll und notwendig ist. In Abschnitt 3.1 wurde ein Schema vorgestellt, das eine Reihe von Merkmalen enthält, anhand derer das Leistungsspektrum verschiedener Analyseansätze beschrieben werden kann. In Abschnitt 3.2 und Abschnitt 3.3 wurde sowohl OLAP als auch Data Mining mit Hilfe dieses Schemas beschrieben. Dabei hat sich gezeigt, dass keiner der beiden Ansätze alle Merkmale einer umfassenden Analyse aufweist. OLAP liefert lediglich Maßzahlen und umfasst keine Möglichkeiten zur Prognose von Daten. Data Mining liefert demgegenüber Muster und Regeln mit sowohl deskriptivem als auch prognostizierendem Charakter. Allerdings sind hier wesentlich weniger ausgeprägte Möglichkeiten zur Navigation in den Daten und zur Selektion von Daten vorhanden. Das volle Spektrum kann also nur abgedeckt werden, wenn bei Analysen sowohl OLAP als auch Data Mining in Betracht gezogen werden.

Dies wird darüber hinaus durch die Beispielanalysen der beiden Anwendungsszenarien aus den Abschnitten 3.4 und 3.5 gestützt. Hier gibt es Anwendungsbereiche, wie beispielsweise das Customer Relationship Management, deren typischen Aufgabenstellung sowohl zu Fragestellungen aus dem Bereich OLAP

(wie bei Analyse 3) als auch zu Fragestellungen aus dem Bereich Data Mining (wie in Analyse 9) führt.

Die bisherigen Überlegungen zeigen, dass es für umfassende Analyseanwendungen notwendig ist, sowohl OLAP als auch Data Mining einzusetzen. Der Integrationsbedarf ergibt sich hier aus der Tatsache, dass für verschiedene, aber doch ähnliche Fragestellungen unterschiedliche Analyseansätze notwendig sein können. Für jede neue Fragestellung muss individuell entschieden werden, ob die angeforderten Informationen eher mit Hilfe eines Werkzeugs aus dem Bereich OLAP bereitgestellt werden können, oder ob der Einsatz von Data Mining erfolgversprechender ist. Die primäre Sichtweise ist hier also der parallele Einsatz von Data Mining und OLAP. In Abbildung 10 ist dies durch den Ablauf, der sich mit den durchgezogenen Pfeilen ergibt, dargestellt. Als Ergebnis der Analyse einer Fragestellung wird entschieden, mit welchem Analyseansatz die angestrebten Ergebnisse erzielt werden können.

Daneben gibt es aber noch eine zweite Sichtweise des Integrationsbedarfs. Diese orientiert sich an einer sequenziellen Betrachtungsweise. Hier liegt der Fokus auf Analysen, die zunächst den Einsatz von OLAP und dann, auf der Basis von Teilergebnissen, die Anwendung von Data-Mining-Technologie vorsehen. Umgekehrt kann natürlich auch OLAP eingesetzt werden, um Data-Mining-Ergebnisse näher zu analysieren. Diese sequenzielle Betrachtungsweise ist in Abbildung 10 mit den durchgezogenen und den gestrichelten Pfeilen angedeutet. Es ergibt sich also eine schrittweise Verfeinerung der Analyse durch die wechselweise Anwendung von OLAP und Data Mining. Diese Sichtweise soll im Folgenden näher erläutert werden.

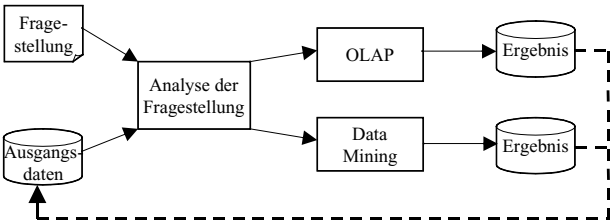

Abbildung 10: Paralleler und sequenzieller Einsatz von OLAP und Data Mining

Zunächst zum Einsatz von Data Mining auf der Basis von OLAP-Ergebnissen. Bei dieser Vorgehensweise ermöglicht OLAP zunächst eine Selektion der relevanten Daten sowie die Navigation in den Daten. Beide Operationen werden von Data-Mining-Verfahren nicht bereitgestellt. Als Ergebnis wird die weiter zu analysierende Datenmenge eingeschränkt, d.h., das einzusetzende Data-Mining-Verfahren wird sofort auf die eigentlich relevanten Bereiche fokussiert. Dies kann einerseits Performanz-Vorteile bringen, da das Data-Mining-Verfahren auf einer kleineren Datenmenge arbeitet. Andererseits wird auch die Qualität der Ergebnisse beeinflusst, da das Data-Mining-Verfahren exakt auf die mit Hilfe von OLAP als relevant erkannten Daten angewandt wird. Beispielsweise können durch OLAP auffällige Informationen auf einem bestimmten Aggregationsniveau erkannt und mit Hilfe von Data Mining weiter analysiert werden. Ohne die vorangehende Analyse wird das Data-Mining-Verfahren auf die Basisdaten angesetzt und Zusammenhänge, die sich erst auf einer höheren Aggregationsstufe zeigen, können nicht identifiziert werden.

Bei der umgekehrten Vorgehensweise werden Daten zunächst mit Hilfe eines Data-Mining-Verfahrens analysiert und OLAP wird dann verwendet, um das Ergebnis weiter zu bearbeiten. Da die verschiedenen Data-Mining-Verfahren ganz unterschiedliche Modelle liefern, muss hier die Betrachtung für die einzelnen Verfahren getrennt erfolgen. Sowohl Clustermodelle als auch Klassifikationsmodelle können auf vorhandene Daten angewandt werden. Dies liefert die Zuordnung von Objekten zu einem Cluster bzw. einer Klasse. Eine darauf aufbauende Analyse mit OLAP kann in diesem Fall detaillierte Informationen zu einer oder mehreren dieser Klassen liefern. Abhängigkeitsmodelle und die Ergebnisse einer Sequenzanalyse beschreiben Beziehungen zwischen verschiedenen Objekten. Bei der Warenkorbanalyse ist dies die Beziehung, dass mehrere Produkte häufig gemeinsam in einem Warenkorb auftauchen. Mit Hilfe des Online Analytical Processing können nun genau die Objekte, die eine solche Beziehung aufweisen, weiter analysiert werden. Bleibt als letztes der in dieser Arbeit näher betrachteten Data-Mining-Modelle die Regression. Sie liefert als Ergebnis Gleichungen, die den Zusammenhang zwischen den Attributen von Objekten und einem daraus abgeleiteten kontinuierlichen Wert darstellen. Ähnlich wie bei der Klassifikation kann OLAP hier genutzt werden, um alle Objekte, für die der abgeleitete kontinuierliche Wert in einem bestimmten Bereich liegt, detaillierter zu analysieren.

Diese Überlegungen zeigen, dass umfassende Analysen sowohl auf OLAP als auch auf Data Mining beruhen sollten. Beide Ansätze können einerseits parallel eingesetzt werden. Andererseits konnte in diesem Abschnitt auch gezeigt werden, dass ein aufeinander aufbauender Einsatz der beiden Ansätze ebenfalls möglich und sinnvoll ist. Im folgenden Abschnitt wird dies anhand der Anwendungsszenarien nochmals verdeutlicht.

3.7 Typische Fragestellungen für den integrierten Einsatz von OLAP und Data Mining

Aus der Kombination von Analyse 3 und Analyse 5 ergibt sich die folgende Fragestellung:

Analyse 10 Welches sind jeweils die Top-Kunden in den Clustern ähnlicher Kunden von ToPCHain, wobei die Cluster auf der Basis der Angaben über Marktsegment, Kontostand, Gesamtumsatz, maximalen Einzelumsatz, Handelsregion, Nation, Gesamtstückzahl bestellter Produkte sowie der Gesamtzahl der Bestellungen des einzelnen Kunden gebildet werden sollen. Als Top-Kunden der ToPCHain Handelskette sollen diejenigen betrachtet werden, bei denen die Standardabweichung des Umsatzes in einem Zeitraum Z am geringsten ist.

Mit Hilfe eines Data-Mining-Verfahrens wird zunächst ein Clustermodell gebildet. Die einzelnen Cluster enthalten hier einander ähnliche Kunden. Für jedes dieser Cluster sollen nun die Kunden mit möglichst gleich bleibendem Umsatz identifiziert werden. Hierbei handelt es sich um eine typische OLAP-Fragestellung, die hier gezielt auf die jeweiligen Kunden eines Clusters angewandt wird. Die hier beschriebene Fragestellung kann weder durch die isolierte Anwendung eines Data-Mining-Verfahrens noch durch reines OLAP vollständig beantwortet werden.

Analyse 11 Identifiziere zunächst besonders umsatzstarke Kunden der Phagroh AG und stelle dann fest, ob es unter diesen Kunden gibt, die gemeinsame Sortimentsschwerpunkte aufweisen, d.h., die in demselben Teil des Sortiments überdurchschnittlich hohe Umsätze erzielen? Durch wel-

che Eigenschaften lassen sich solche Kundengruppen charakterisieren?

Diese Fragestellung basiert auf Analyse 9. Hier wird mit Hilfe von OLAP zunächst die Gruppe der zu betrachtenden Kunden näher spezifiziert. Für die Anwendung wurde festgelegt, dass in erster Linie Sortimentsschwerpunkte bei umsatzstarken Apotheken interessant sind. Für die so eingeschränkte Menge der Apotheken wird dann ein Clustermodell erstellt, das die Apotheken nach Sortimentsschwerpunkten gliedern soll.

3.8 Unterstützung durch existierende Werkzeuge, Technologien und Algorithmen

Wenige wissenschaftliche Arbeiten beschäftigen sich mit Aspekten der Integration von Data Mining und OLAP. In [Han97] wird der kombinierte Einsatz der beiden Analyseansätze vorgeschlagen. Ähnlich wie in Abschnitt 3.6 beschrieben, wird hier Data Mining auf der Basis von OLAP-Analysen und umgekehrt angestrebt. Die Arbeit analysiert aber weder Fragen der Modellierung noch der Systemarchitektur im Zusammenhang der integrierten Betrachtung von Data Mining und OLAP. Eine Architektur für die Integration der beiden Analyseansätze wird in [Han98] und [HK01] skizziert. Diese geht von der Speicherung der Daten in einer multidimensionalen Datenbank bzw. in einer proprietären Struktur aus und verfolgt damit einen Ansatz, der nicht direkt mit ROLAP vereinbar ist.

Andere Arbeiten beschäftigen sich mit der Erkennung auffälliger und unerwarteter Werte in einem Datenwürfel [SAM98a] [SAM98b]. Dies stellt zwar eine Erweiterung der OLAP-Funktionalität dar. Die eingesetzten Methoden stammen allerdings nicht aus dem Bereich Data Mining, so dass hier nicht von einer Integration von OLAP und Data Mining gesprochen werden kann.

In [JLN00] wird der Bedarf für eine sequenzielle Anwendung unterschiedlicher Analyseoperationen identifiziert, wobei der Schwerpunkt auf der Anwendung von Data-Mining-Verfahren auf der Grundlage anderer Data-Mining-Ergebnisse liegt. Hierzu wird ein Datenmodell, das 3W-Modell, sowie eine Algebra definiert. Diese stellt Analyseoperatoren bereit, deren Ergebnisse direkt wieder von anderen Analyseoperatoren verwendet werden können. Es wird aber weder die konzeptuelle noch die logische Modellierung für die Basisdaten betrachtet.

Ebenso werden Fragen der Systemarchitektur und der Optimierung ausgeklammert.

Auf der Seite der kommerziellen Analysesoftware gibt es ebenfalls nur wenige Ansätze, um Data Mining und OLAP integriert zu betrachten. Viele Data-Mining-Werkzeuge bieten eine Reihe von Data-Mining-Verfahren an und erlauben den Export der Ergebnisse in unterschiedlichen Formaten [TBD97] [Thi98] [Ibm99] [SP99a] [SP99b] [AN00]. Typische OLAP-Operationen gehören aber in der Regel nicht zum Leistungsspektrum dieser Werkzeuge, so dass integrierte Systeme bisher nicht verfügbar sind.

Ähnlich sieht es auf der Seite der OLAP-Werkzeuge aus. Hier findet meist eine Konzentration auf die primären OLAP-Operationen statt [Mic95] [Mic99] [Abe99] [Cog99] [Cog01] [Hyp01a] [Hyp01b]. Data Mining ist dann nur durch den Export der betrachteten Datenwürfel in das dafür vorgesehene Werkzeug möglich. Selbst wenn beide Werkzeuge von demselben Anbieter stammen, ist hiermit der Wechsel in eine andere Applikation mit eigener Systemarchitektur und Benutzungsschnittstelle verbunden. Jede Applikation weist darüber hinaus ihre spezifischen Anforderungen an die zu Grunde liegenden Daten und deren interne Repräsentation auf.

3.9 Identifikation der Integrationsbereiche

Die Analyse verfügbarer wissenschaftlicher Arbeiten, Technologien und Werkzeuge des vorangegangenen Abschnitts hat gezeigt, dass bisher keine vollständige Unterstützung für die integrierte Anwendung von OLAP und Data Mining gegeben ist. Um diese durchgehend gewährleisten zu können, muss eine Reihe von Aspekten näher untersucht werden.

Hierzu gehört zunächst die Frage nach einem geeigneten Datenmodell, auf das Werkzeuge aus den Bereichen OLAP und Data Mining gleichermaßen gut aufsetzen können. Bei der Betrachtung des Datenmodells muss beispielsweise berücksichtigt werden, ob die verschiedenen Analyseansätze eine unterschiedliche Sicht auf die Daten implizieren. Darüber hinaus kann sowohl die Wahl des logischen als auch des physischen Schemas Auswirkungen auf die Performanz der Anwendung haben. In Kapitel 4 wird die Datenmodellierung für ein Data Warehouse analysiert und geeignete Datenmodelle und Schematypen für die Integration von Data Mining und OLAP werden identifiziert.

Ein weiterer Aspekt ist die Systemarchitektur, die für Data Mining und OLAP einen gleichermaßen guten Rahmen bildet. Diese Frage ist eng mit möglichen Ansätzen zur anwendungsübergreifenden Optimierung verbunden. Das Ziel aller hier betrachteten Analyseansätze ist es, eine möglichst interaktive Analyse der Daten zu ermöglichen. Die effiziente Ausführung der Anfragen ist also ein zentraler Aspekt. Hier stellt sich die Frage, ob es Optimierungsansätze gibt, die für OLAP und Data Mining gleichermaßen einsetzbar sind, so dass sie unabhängig von der konkreten Anwendung zum Einsatz kommen können. Sofern solche allgemeinen Optimierungsansätze identifiziert werden können stellt sich wieder die Frage, wie diese in der Systemarchitektur realisiert werden können. Die Frage der geeigneten Systemarchitektur wird in Kapitel 5 dieser Arbeit behandelt, während anwendungsübergreifende Optimierungsansätze Gegenstand von Kapitel 6 sind.

Für die Integration ist der Zugriff auf Analyseergebnisse notwendig. Damit ergibt sich im Bereich der Metadaten die Frage, wie Analyseergebnisse repräsentiert werden müssen, um deren Weiterverwendung in unterschiedlichsten Analyseanwendungen zu ermöglichen. Für die Speicherung und den Austausch der Ergebnisse von Data-Mining-Verfahren wurde beispielsweise die auf XML basierende *Predictive Model Markup Language* (PMML) entwickelt [Dat00]. Der Austausch von Analyseergebnissen zwischen Data-Mining-Anwendungen und OLAP-Anwendungen in beide Richtungen wird hierdurch allerdings nicht unterstützt. Weiter geht hier das *Common Warehouse Metamodel* (CWM) der Object Management Group, das sowohl die Modellierung von Metadaten für Data Mining als auch für OLAP erlaubt [Obj01a] [Obj01b]. Eine detaillierte Analyse, wie Analyseergebnisse in einem Repository repräsentiert werden müssen, um sie integriert für OLAP und Data Mining nutzen zu können, ist nicht Gegenstand dieser Arbeit.

3.10 Zusammenfassung

Die beiden Analyseansätze OLAP und Data Mining weisen unterschiedliche Leistungsspektren auf. Die Unterschiede wurden in diesem Kapitel analysiert und anhand konkreter Fragestellungen für die in Kapitel 2 vorgestellten Anwendungsszenarien erläutert. Es konnte gezeigt werden, dass eine umfassende Analyse der in einem Data Warehouse verfügbaren Daten nur möglich ist, wenn die beiden Analyseansätze gleichermaßen berücksichtigt werden. Welche zusätzlichen Analysemöglichkeiten sich hierbei ergeben, wurde wiederum im Rahmen

der Anwendungsszenarien verdeutlicht. Wie in Abschnitt 3.8 erläutert, hat die Analyse existierender Werkzeuge, Technologien und Algorithmen gezeigt, dass in diesen Bereichen hinsichtlich der Integration von OLAP und Data Mining noch erhebliche Defizite vorhanden sind. Konkrete Fragestellungen wurden hierbei in den Bereichen Datenmodellierung, Systemarchitektur und anwendungsunabhängige Optimierung identifiziert.

4 Datenmodellierung

In der Literatur gibt es umfangreiche Informationen und Handlungsanweisungen zur Erstellung des Datenmodells für ein Data Warehouse [Inm92] [Kim96] [KR+98]. Eine Reihe von Datenmodellen und Schematypen wird in diesem Kapitel beschrieben und analysiert. Es wird sich dabei zeigen, dass allen betrachteten Datenmodellen eine multidimensionale Sichtweise zu Grunde liegt und sie sich in erster Linie an Analysen aus dem Bereich OLAP orientieren. Welche Auswirkungen das Datenmodell auf Analysen aus dem Bereich Data Mining hat und welche Konsequenzen sich für die Modellierung aus den spezifischen Merkmalen der Data-Mining-Verfahren ergeben, ist bisher nicht detailliert untersucht worden. Diese Analyse wird in den folgenden Abschnitten, getrennt nach den einzelnen Phasen des Modellierungsprozesses, durchgeführt. Die entsprechenden Schlussfolgerungen für die Datenmodellierung werden in Abschnitt 4.4 dargestellt.

Der Datenbankentwurf ohne Berücksichtigung der besonderen Anforderungen im Data-Warehouse-Umfeld wird in dieser Arbeit als *klassischer Datenbankentwurf* bezeichnet. Hierbei handelt es sich um einen komplexen Entwurfsprozess, der typischerweise in drei Phasen unterteilt ist und auf einer detaillierten Anforderungsanalyse beruht. Man unterscheidet die Phasen des konzeptuellen, des logischen und des physischen Entwurfs, entsprechend der zunehmenden Abhängigkeit vom Zieldatenbanksystem und der physischen Speicherung der Daten. In der Regel beruht der Entwurfsprozess auf einer daten-orientierten Sichtweise des Systems, d.h., es werden die Daten und ihre Eigenschaften modelliert und nicht die Anwendungen, die auf die Daten zugreifen [TYF86] [BCN92].

In jeder Phase des Entwurfsprozesses werden spezifische Modelle verwendet, um das Schema, welches das Ergebnis der jeweiligen Entwurfsphase ist, zu beschreiben. Das Schema gibt jeweils die Struktur einer konkreten Datenbank an, während das Modell die Sprache darstellt, in der das Schema beschrieben wird. Beim konzeptuellen Entwurf kommt häufig das Entity-Relationship-Modell zum Einsatz [Che77] [Dat95] [Che96] [BCN92]. Es gibt aber auch eine Reihe alternativer Ansätze, wie z.B. das NIAM-Modell [LF93]. Ein Vergleich von Modellen findet sich beispielsweise in [HK87]. Das geeignete Modell für den logischen Entwurf hängt vom Zieldatenbanksystem ab. Bei der Modellierung für ein relationales Datenbanksystem basiert das logische Schema auf dem relationalen Modell. Bei der Erstellung des logischen Schemas wird häufig als Zwischen-

schritt ein modifiziertes Schema verwendet, das auf dem Entity-Relationship-Modell basiert [BCN92] [CBS99]. Für die physische Modellierung gibt es keine vergleichbaren allgemeingültigen Modelle.

Wie beim klassischen Datenbankentwurf werden auch beim Entwurf eines Data Warehouse mehrere Phasen unterschieden. Der Inhalt der wichtigsten Phasen des Entwurfsprozesses ist nach [GR98] und [GR99]:

- die Analyse der operationalen Systeme,
- die Spezifikation der Anforderungen,
- der konzeptuelle Entwurf,
- die Ermittlung relevanter Anfragen und die Überprüfung des konzeptuellen Schemas,
- der logische Entwurf sowie
- der physische Entwurf.

Die meisten anderen Autoren konzentrieren sich auf die drei Phasen des klassischen Datenbankentwurfs [WB97]. Der in [HLV00] vorgestellte Prozess umfasst z.B. nach der Anforderungsanalyse und Spezifikation nur die Phasen des konzeptuellen, logischen und physischen Entwurfs, wie dies bei der klassischen Modellierung der Fall ist. In dieser Arbeit wird diese Sichtweise zu Grunde gelegt. Da spezifische Modelle nur für konzeptuelle und logische Schemata existieren, konzentrieren sich die folgenden Betrachtungen auf die beiden zugehörigen Entwurfsphasen. Der Schwerpunkt liegt hierbei nicht auf der methodischen Vorgehensweise, sondern auf den in den einzelnen Entwurfsphasen eingesetzten Datenmodellen und Schematypen.

4.1 Der konzeptuelle Data-Warehouse-Entwurf

Das grundlegende Ziel eines konzeptuellen Datenbankentwurfs ist es, ein Schema für das Zielinformationssystem zu erstellen und damit die Grobstruktur der Daten, die das Informationssystem verwaltet, zu beschreiben. Dieses Schema soll vom Typ des zu verwendenden Datenbanksystems und der physischen Struktur, in der die Daten abgelegt werden, unabhängig sein [BCN92]. Im Data-Warehouse-Bereich bedeutet dies eine Modellierung unabhängig davon, ob die Daten in einem relationalen oder multidimensionalen Datenbanksystem verwaltet werden [WB97] [HLV00].

4.1.1 Das konzeptuelle Modell COCOM

In der Literatur werden für den konzeptuellen Entwurf eines Data Warehouse verschiedene Modelle vorgeschlagen [LW96] [AGS97] [GL97] [CT98] [GMR98a] [GMR98b] [GR98] [Len98] [GR99] [TBC99] [HLV00] [Lec01]. Die im Rahmen dieser Arbeit getroffenen Aussagen zur Modellierung sollen nicht von den spezifischen Eigenschaften eines dieser Modelle abhängig sein. Im Folgenden wird deshalb ein Datenmodell vorgestellt, das die wesentlichen Merkmale der in der Literatur vorgeschlagenen Modelle umfasst. Manche Modellierungsdetails werden nur von einem oder wenigen publizierten Modellen unterstützt. Bei diesen muss einzeln überprüft werden, ob sie von grundlegender Bedeutung sind. Nur dann werden solche spezifischen Modellierungsdetails im Folgenden berücksichtigt. Da der Fokus auf den gemeinsamen Eigenschaften der verschiedenen Modelle liegt, wird das in dieser Arbeit zu Grunde gelegte Modell als *COCOM-Modell* (COmmon COnceptual warehouse Model) bezeichnet. Die Darstellung des Modells orientiert sich an der Beschreibung des Dimensional Fact Model in [GMR98a] und dessen Erweiterung in [Lec01].

Definition 1: Maß

Ein *Maß m* besteht aus einem numerischen oder booleschen Ausdruck.

In der Regel handelt es sich bei den Maßen um die Kenngrößen, die primär analysiert werden sollen. Sie beruhen auf Daten der operationalen Systeme. Im Anwendungsszenarium Phagroh AG gibt es beispielsweise Maße für die Anzahl verkaufter Medikamente (*soldUnits*) und den damit erzielten Umsatz (*turnover*).

Definition 2: Fakt

Ein *Fakt F* besteht aus einem Namen *F* und einer Menge *M* von Maßen.

Mehrere Kenngrößen können so zu einem Fakt zusammengefasst werden. Bei der Phagroh AG ist es z.B. möglich, die beiden Maße *soldUnits* und *turnover* zu einem Fakt *CustomerTurnover* zu verbinden.

Definition 3: Dimensionsattribut

Ein *Dimensionsattribut a* wird durch eine Menge diskreter Werte oder durch Tupel diskreter Werte charakterisiert. Der Wertebereich für ein Dimensionsattribut wird als $Dom(a)$ bezeichnet.

Dimensionsattribute repräsentieren die unterschiedlichen Betrachtungsebenen für die Analyse der Fakten. Typische Dimensionsattribute bei der Phagroh AG sind Angaben zu den Kunden, wie *customerID, zip* oder *city.*

Definition 4: Charakterisierungsattribut

Ein *Charakterisierungsattribut c* wird durch eine Menge diskreter Werte oder Tupel diskreter Werte beschrieben. $Dom(c)$ ist der Wertebereich des Charakterisierungsattributs *c*.

Charakterisierungsattribute sind zwar gleich definiert wie Dimensionsattribute, unterscheiden sich von diesen aber in ihrer Verwendung. Sie liefern zusätzliche Informationen zu einem Dimensionsattribut und können bei der Analyse zwar zur Selektion relevanter Daten verwendet werden, nicht aber zu deren Gruppierung. Das Attribut *name* der Dimension *Customer* stellt bei der Phagroh AG ein Beispiel hierfür dar.

Definition 5: hierarchische Beziehung von Attributen

Sei *A* eine Menge von Dimensionsattributen, *C* eine Menge von Charakterisierungsattributen und a_0 ein Pseudodimensionsattribut, das die Rolle eines Fakts spielt. Eine *hierarchische Beziehung* von Attributen aus A und C wird durch das Tupel (a_i,a_j) mit $a_i \in A \cup \{a_0\}, a_j \in A \cup C, a_i \neq a_j$ bezeichnet.

Hiermit werden die Beziehungen der Attribute innerhalb einer Dimension festgelegt. Zu beachten ist, dass in der Definition nichts über die Kardinalität der jeweiligen Beziehung ausgesagt wird. Sowohl bei den Dimensionsattributen als auch bei den Charakterisierungsattributen sind 1:1-, 1:n- und n:m-Beziehungen möglich. In Abbildung 6 sind für die Phagroh AG keine hierarchischen Beziehungen angegeben. Innerhalb der Zeitdimension gelten aber offensichtlich die folgenden Beziehungen: (date, month), (month, year), (date, year).

Definition 6: Dimension

Eine *Dimension* $d = (A, C, H)$ besteht aus einer Menge *A* von Dimensionsattributen, einer Menge *C* von Charakterisierungsattributen sowie einer Menge $H = \{(a_i,a_j) \mid a_i \in \{a_0\} \cup A, a_j \in A \cup C, a_i \neq a_j\}$ von hierarchischen Beziehungen zwischen den Attributen der Dimension. Die Menge $\{a_0\} \cup A \cup C$ bildet mit den hierarchischen Beziehungen *H* einen gerichteten Graphen, in dem alle Knoten von a_0 aus zu erreichen sind. Alle Dimensi-

onsattribute $a_j \in A$ mit $(a_0,a_j) \in H$ werden als *Primärattribute* der Dimension *d* bezeichnet.

Eine Dimension umfasst mehrere eng zusammenhängende Aspekte, die bei der Analyse eine Sichtweise der Daten widerspiegeln. Zu einer Dimension gehören sowohl Dimensionsattribute als auch Charakterisierungsattribute und deren hierarchische Beziehungen. Eine der wichtigsten Dimensionen ist die Zeit. Bei der Phagroh AG setzt diese sich unter anderem aus den Dimensionsattributen *date*, *year*, *month* und *day* zusammen.

Definition 7: Aggregationspfad

Sei *M* eine Menge von Maßen, *d* eine Dimension und Ω ein Aggregationsoperator. Ein *Aggregationspfad* über *M* und *d* ist dann ein Tripel (m_i, d, Ω) mit $m_i \in M$.

Ein Aggregationspfad gibt an, dass das Maß m_i innerhalb der Dimension *d* mit dem Aggregationsoperator Ω aggregiert werden kann. Zu beachten ist, dass die Angabe möglicher Aggregationspfade hier immer pro Dimension erfolgt und nicht für die einzelnen Hierarchieebenen aufgeschlüsselt wird. Es ist zulässig, dass für einzelne Maße keine Aggregationspfade angegeben sind und damit keine Aggregation möglich ist. Hierfür wird in dieser Arbeit die Bezeichnung *nicht-aggregierbares Maß* verwendet. Vereinzelt werden solche Maße auch als Qualitäten bezeichnet [Mic99].

Definition 8: Fakten-Schema

Sei *M* eine Menge von Maßen, *A* eine Menge von Dimensionsattributen, *C* eine Menge von Charakterisierungsattributen, *D* eine Menge von Dimensionen mit $D = \{(A', C', H) | A' \subseteq A, C' \subseteq C\}$, wobei H die Menge der hierarchischen Beziehungen über A' und C' darstellt. P sei eine Menge von Aggregationspfaden über M und D. Ein *Fakten-Schema FS* ist dann das Fünftupel $FS = (M, A, C, D, P)$.

Ein Fakten-Schema umfasst in der Regel einen Fakt und alle Dimensionen, die für die Interpretation des Fakts von Bedeutung sind. Nach dieser Definition ist der in Abbildung 6 dargestellte Ausschnitt aus dem Schema der Phagroh AG genau ein Fakten-Schema.

Definition 9: Dimensionen-Schema

Eine Menge *DS* von Fakten-Schemata wird als *Dimensionen-Schema* bezeichnet.

Damit wird die Möglichkeit bereitgestellt, auf unterschiedliche Fakten und die mit ihnen in Beziehung stehenden Dimensionen zuzugreifen. Hierbei kann dieselbe Dimension mehrfach in den Fakten-Schemata vertreten sein.

4.1.2 Abgrenzung zu anderen konzeptuellen Modellen

Das COCOM-Modell weist alle grundlegenden Merkmale bereits publizierter Modelle auf. In einzelnen Punkten differiert es allerdings von anderen Modellen. Die Unterschiede zeigen sich häufig an Eigenschaften der Modelle, die von einzelnen Autoren als besonders wichtig identifiziert wurden und die darum bei der Erstellung der jeweiligen Modelle besondere Berücksichtigung fanden. Es folgt darum zunächst eine Zusammenstellung von Eigenschaften für das konzeptuelle Modell eines Data Warehouse, denen manche Autoren eine besondere Bedeutung zumessen. Hierbei wird jeweils auf die Arbeit verwiesen, in der die Relevanz der jeweiligen Modellierungsmöglichkeit explizit hervorgehoben wird.

Ein wichtiger Punkt, in dem sich die verschiedenen Modelle unterscheiden, ist die Frage, wie detailliert und flexibel die einzelnen Dimensionen modelliert werden können. Zunächst besteht eine Dimension in allen Modellen aus einer Reihe von Attributen. Bei der detaillierteren Modellierung der Dimension können die folgenden Aspekte unterschieden werden:

- Hierarchische Beziehungen zwischen den Attributen einer Dimension können explizit modelliert werden und sind nicht nur implizit vorhanden [PJ99].
- Innerhalb einer Dimension ist es möglich, mehrere Hierarchien zu modellieren [AGS97] [PJ99].
- Bei der Modellierung hierarchischer Beziehungen ist es möglich, zwischen strikten und nicht-strikten Beziehungen bzw. vollständigen und unvollständigen Beziehungen zu unterscheiden [TBC99] [PJ99].
- Charakterisierungsattribute können explizit modelliert werden [Len98] [BHL00].
- Es ist möglich, Einschränkungen hinsichtlich der Gültigkeit von Charakterisierungsattributen zu modellieren [Len98] [BHL00].

Für die Fakten und deren Beziehungen zu den Dimensionen lassen sich folgende Aspekte unterscheiden:

- Es besteht eine vollständige Symmetrie zwischen Fakten und Dimensionen [AGS97] [PJ99].
- Zwischen Fakten und Dimensionen können n:m-Beziehungen modelliert werden [PJ99].
- Es ist möglich, Beziehungen zwischen unterschiedlichen Fakten zu modellieren [MK00].
- Im Modell lässt sich angeben, welche Aggregationen für bestimmte Fakten und Dimensionen zulässig sind [LS97].

Neben diesen Eigenschaften gibt es noch zwei weitere Aspekte, die beachtet werden können:

- Änderungen der Daten über die Zeit hinweg werden im Modell berücksichtigt [PJ99].
- Es ist möglich, einzelne Angaben mit Wahrscheinlichkeiten zu versehen [PJ99].

Ausgehend von diesem Eigenschaften-Katalog soll im Folgenden dargestellt werden, welche Eigenschaften in den näher untersuchten Modellen aus [LW96] [AGS97] [GL97] [CT98] [GMR98a] [LAW98] [TBC99] [PJ99] [HLV00] [Lec01] gegeben sind und welche im COCOM-Modell berücksichtigt wurden.

Die explizite Modellierung von Hierarchien innerhalb einer Dimension sowie die Berücksichtigung paralleler Hierarchien ist in der Mehrzahl der Modelle gegeben [CT98] [GMR98a] [LAW98] [TBC99] [PJ99] [HLV00] [Lec01]. Das COCOM-Modell weist diese Eigenschaften darum ebenfalls auf. Selten wird dagegen eine explizite Modellierung der Striktheit [TBC99] [PJ99] oder Vollständigkeit [TBC99] [LAW98] der Beziehungen zwischen Attributen einer Dimension berücksichtigt. Diese Modellierungsmöglichkeiten spielen in der Regel weder bei der Durchführung von OLAP-Analysen noch bei der Erstellung von Data-Mining-Modellen eine Rolle und bleiben darum auch in Abschnitt 4.1.1 unberücksichtigt.

Charakterisierungsattribute liefern zusätzliche Informationen zu einzelnen Dimensionsattributen und werden in einer Vielzahl von Modellen unter unter-

schiedlichsten Bezeichnungen berücksichtigt [CT98] [GMR98a] [LAW98] [TBC99] [HLV00] [Lec01]. Diese sind in manchen Fällen nur für einzelne Werte des Dimensionsattributs relevant [Len98] [BHL00]. In einer Produkthierarchie können dies z.B. Merkmale sein, deren Betrachtung nur für bestimmte Produktgruppen sinnvoll ist. Da dies nur in einem der analysierten Modelle explizit berücksichtigt ist [LAW98], wird diese Modellierungsmöglichkeit im Rahmen dieser Arbeit nicht weiter betrachtet. Im COCOM-Modell wird lediglich die Unterscheidung zwischen Dimensions- und Charakterisierungsattributen unterstützt.

Nur in wenigen Modellen wird eine Symmetrie zwischen Fakten und Dimensionsattributen angestrebt [AGS97] [PJ99]. Ein wesentliches Argument für diese Symmetrie ist, dass in einem Data Warehouse u.U. Attribute bereitgestellt werden sollen, die sowohl im Sinne von Fakten, d.h. zur Aggregation, als auch im Sinne von Dimensionsattributen, d.h. zur Gruppierung und Selektion, verwendet werden sollen. Wie in [Lec01] wird diese Modellierungsmöglichkeit hier nicht weiter berücksichtigt, da die Trennung zwischen Fakten und Dimensionen für typische Anwendungen im OLAP-Bereich grundlegend ist.

Betrachtet man die Beziehung eines Fakts zu einer Dimension, so wird hier in der Regel von 1:n-Beziehungen ausgegangen. Jedem Fakt ist in jeder Dimension genau ein Objekt zugeordnet. Eine Modellierung von n:m-Beziehungen wird in [PJ99] vorgeschlagen. Darüber hinaus können Fakten Beziehungen zueinander aufweisen. Ein Beispiel hierfür ist eine Bestellung und die einzelnen Bestellposten, die separat als Fakten modelliert werden [MK00]. In typischen Anwendungsszenarien sind für die Modellierung weder diese Beziehungen zwischen Fakten noch die lediglich in [PJ99] berücksichtigten n:m-Beziehungen zwischen Fakten und Dimensionen notwendig. Sie werden deshalb im COCOM-Modell ebenfalls nicht weiter betrachtet.

Eine wichtigere Rolle spielt bei der Modellierung die Frage, welche Aggregations-Funktionen auf welche Fakten angewandt werden können. Dies wird in einer ganzen Reihe von Modellen explizit modelliert [CT98] [GMR98a] [TBC99] [PJ99] [HLV00] und beispielsweise in [Lec01] hervorgehoben. Darüber hinaus werden Voraussetzungen für eine korrekte Aggregation sowie entsprechende Normalformen in [LS97] [LAW98] [Lec01] betrachtet. Das vorliegende COCOM-Modell bietet ebenfalls die Möglichkeit anzugeben, in welcher Weise Fakten in bestimmten Dimensionen aggregiert werden können.

Darüber hinaus wird es in [PJ99] als relevant erachtet, dass ein Modell Änderungen der Daten über die Zeit hinweg bereits explizit berücksichtigt und dass es möglich ist, Informationen mit Wahrscheinlichkeiten zu versehen. In einem Data Warehouse besteht die wesentliche Änderung im Hinzufügen neu erfasster Fakten. Da es sich hierbei lediglich um das Hinzufügen von Datensätzen handelt, ist dies nicht relevant für das gewählte Datenmodell. Erfasste Wahrscheinlichkeiten werden weder von existierenden OLAP-Werkzeugen noch von den gängigen Algorithmen im Bereich Data Mining berücksichtigt. Darum sind diese Modellierungsmöglichkeiten im COCOM-Modell ebenfalls nicht vorgesehen.

4.1.3 Bewertung des COCOM-Modells

Im vorangegangenen Abschnitt wurde gezeigt, dass das beschriebene, allgemeine konzeptuelle Datenmodell für ein Data Warehouse alle wichtigen Eigenschaften aufweist, die auch in anderen publizierten Datenmodellen berücksichtigt wurden. Auf dieser Grundlage wird nun analysiert, inwieweit dieses Modell für OLAP-Analysen und Data-Mining-Analysen gleichermaßen geeignet ist.

4.1.3.1 Eignung für OLAP

Für den Bereich OLAP wurden wichtige Merkmale in Abschnitt 3.2 beschrieben. Eines dieser Merkmale ist, dass OLAP als Ergebnis Maßzahlen liefert. Auf welcher Basis diese berechnet werden können, wird im vorgestellten Datenmodell durch die explizite Modellierung der Fakten erfasst. Charakteristische Operationen für OLAP sind die Navigation, die Selektion und die Aggregation. Die Navigation auf den Daten wird im beschriebenen Datenmodell durch die Modellierung der hierarchischen Beziehungen innerhalb der einzelnen Dimensionen unterstützt. Diese Hierarchien sind ebenso bei der Aggregation der Daten notwendig. Befindet man sich bei der Analyse auf einer bestimmten Betrachtungsebene, dann wird durch die hierarchischen Beziehungen vorgegeben, welche Aggregationsstufen in der weiteren Analyse angestrebt werden können. Der Tatsache, dass nicht in allen Dimensionen beliebige Aggregationsfunktionen angewandt werden können, wird im COCOM-Modell durch die explizite Modellierung von Aggregationspfaden Rechnung getragen.

Betrachtet man zusätzlich die 12 Regeln zu OLAP von E. F. Codd, die in Abschnitt 2.2 beschrieben sind, dann zeigt sich auch hier, dass das gewählte Datenmodell alle wesentlichen Aspekte von OLAP unterstützt. Ein wichtiges

Merkmal hierbei ist wiederum die Identifikation einzelner Dimensionen und die Modellierung der hierarchischen Beziehungen innerhalb dieser Dimensionen (Regel 1). Anwendungsspezifische Sichten, wie sie von Codd gefordert werden, können im vorliegenden Modell unter anderem durch die Modellierung alternativer Aggregationspfade bereitgestellt werden. Weitere geforderte Merkmale, die auch auf das konzeptuelle Schema bezogen werden können, sind: die identische Strukturierung und die identischen Operationen in den verschiedenen Dimensionen (Regel 6) sowie die fehlende Einschränkung bezüglich der Anzahl der Dimensionen und deren Umfang (Regel 12). Das COCOM-Modell weist alle diese Eigenschaften auf. Alle anderen von E. F. Codd aufgestellten Regeln sind nicht auf das konzeptuelle Schema anwendbar. Insgesamt stellt das vorgestellte Modell also eine geeignete Basis für Analysen aus dem Bereich OLAP dar. Dies entspricht der Erwartung, da das Modell auf einer Reihe von Datenmodellen basiert, die speziell mit dem Fokus auf OLAP-Analysen entwickelt wurden.

4.1.3.2 Eignung für Data Mining

Es bleibt damit zu überprüfen, wie sich das Datenmodell als Grundlage für Data-Mining-Analysen eignet. Diese Betrachtung wird hier anhand der in Abschnitt 2.3 erläuterten Data-Mining-Modelle durchgeführt.

Durch die Modellierung der Fakten wird die Ableitung von Assoziationsregeln unterstützt. Dies kann exemplarisch am typischen Beispiel für Assoziationsregeln, der Analyse von Warenkörben verdeutlicht werden. Die Tatsache, dass ein bestimmtes Produkt verkauft wurde, sowie die zugehörige Stückzahl, der erzielte Umsatz usw., ist der festzuhaltende Fakt. Die zusätzlichen Informationen, die den einzelnen Warenkorb näher charakterisieren, sind in den dem Fakt zugeordneten Dimensionen enthalten. Hierzu gehört beispielsweise der Kunde und der Verkaufszeitpunkt. Mit dieser Modellierung, bei der der Inhalt von Warenkörben als Fakt und die zusätzliche Information zu den einzelnen Warenkörben als Bestandteil der Dimensionen erfasst werden, können bereits im konzeptuellen Schema alle für die Erstellung von Assoziationsregeln relevanten Informationen erfasst und strukturiert werden.

Sowohl bei Clustermodellen als auch bei Klassifikationsmodellen geht es um Objekte und deren Merkmale sowie um die Einteilung der Objekte in verschiedene Klassen basierend auf diesen Merkmalen. In der Regel kann eines der bei der Analyse betrachteten Objekte als Bestandteil einer Dimension gesehen wer-

den. Wichtige Objekte der Analyse bei der Phagroh AG sind z.B. die einzelnen Kunden oder die Artikel, die entsprechend den Dimensionen *Customer* bzw. *Article* zugeordnet werden. Durch die Modellierung der verschiedenen Dimensionen werden also auch wichtige Merkmale für die zu analysierenden Objekte strukturiert. Zusätzliche Merkmale für diese Objekte können mit Hilfe der mit einer Dimension verbundenen Fakten gewonnen werden. So kann im Modell der Phagroh AG z.B. über das Maß *turnover* für jeden einzelnen Kunden der maximale Monatsumsatz als zusätzliches Merkmal ermittelt werden.

Aggregationen spielen beim Data Mining eine zentrale Rolle. Bei der Erstellung von Clustermodellen, Klassifikationsmodellen und Assoziationsregeln werden die Daten in der Regel mehrfach und mit unterschiedlichen Gruppierungen und Selektionen aggregiert [SAM96]. Im COCOM-Modell wird dies durch die Festlegung der Hierarchien innerhalb einer Dimension unterstützt. Dadurch werden die potenziellen Aggregationspfade definiert. Darüber hinaus sind die festgelegten Hierarchien bei der Ableitung von hierarchieübergreifenden, generalisierten Assoziationsregeln relevant [SA95]. Dies kann beispielsweise bedeuten, dass Verkaufsdaten nicht nur auf Abhängigkeiten zwischen unterschiedlichen Artikeln hin analysiert werden, sondern dass auch Artikelgruppen in der Analyse Berücksichtigung finden. Das COCOM-Modell bietet eine weitere Unterstützung für die Aggregation der Daten, in dem erfasst wird, welche Aggregationsfunktionen in welchen Dimensionen zulässig sind.

Sequenzen basieren auf dem zeitlichen Verhalten von Datenströmen. Im Datenmodell kann die Zeit als eigenständige Dimension modelliert und mit den sich verändernden Fakten verbunden werden. Damit kann für die einzelnen Fakten der zeitliche Verlauf ihres Auftretens festgehalten werden. Je nach Modellierung der Zeitdimension ist eine Betrachtung auf verschiedenen Detaillierungsstufen möglich. Das konzeptuelle Modell ermöglicht es also, alle grundlegenden Informationen für die Erstellung von Sequenzmodellen zu erfassen.

Wie gezeigt werden konnte, stellt das COCOM-Modell eine gute Basis für die Modellierung von Daten dar, die mit Hilfe der in Abschnitt 2.3 beschriebenen Data-Mining-Verfahren analysiert werden sollen. Im Gegensatz zu OLAP, wo auch die typischen Operationen, wie z.B. die Aggregation, bereits im Datenmodell unterstützt werden, ist dies jedoch für Data Mining nicht möglich. Hier unterscheiden sich die notwendigen Operationen in Abhängigkeit vom eingesetzten Verfahren und Algorithmus. Darüber hinaus müssen für einzelne Verfahren

Zielgrößen vorgegeben werden. Dies kann aber nur in der individuellen Analyse und nicht allgemein im Datenmodell erfolgen.

4.1.3.3 Zusammenfassende Bewertung des COCOM-Modells

Die Betrachtungen zum konzeptuellen Data-Warehouse-Entwurf haben gezeigt, dass das COCOM-Modell Modellierungsmöglichkeiten bietet, die sowohl für OLAP als auch für Data Mining genutzt werden können. Diese Modellierungsmöglichkeiten unterstützen im Bereich OLAP einerseits die charakteristischen Operationen und berücksichtigen andererseits die in den Regeln von E. F. Codd enthaltenen Anforderungen. Sie erlauben zudem, alle für die Erstellung der unterschiedlichen Data-Mining-Modelle relevanten Informationen zu erfassen und geeignet zu strukturieren. Weitere, spezifische Modellierungsmöglichkeiten, die in einzelnen konzeptuellen Modellen für ein Data Warehouse berücksichtigt sind, wurden in Abschnitt 4.1.2 betrachtet. Viele davon haben sich allerdings im Rahmen der Aufgabenstellung dieser Arbeit als nicht grundlegend herausgestellt und wurden darum nicht in das COCOM-Modell aufgenommen. Insgesamt weist dieses Modell die meisten Gemeinsamkeiten mit dem Dimensional Fact Model auf [GMR98a]. Insbesondere wurden die folgenden Merkmale für das COCOM-Modell als besonders wichtig identifiziert:

- Ein Fakt kann sich aus mehreren Maßen zusammensetzen.
- Es wird explizit zwischen Fakten und Dimensionen unterschieden.
- Dimensionsattribute und Charakterisierungsattribute werden unterschieden.
- Die hierarchischen Beziehungen werden für die einzelnen Dimensionen explizit modelliert.
- In jeder Dimension können mehrere, parallele Hierarchien modelliert werden.
- Pro Dimension können die auf die Maße anwendbaren Aggregationsfunktionen angegeben werden.

Zusammenfassend kann hier gefolgert werden, dass das COCOM-Modell für die Modellierung von Daten geeignet ist, die sowohl mit Methoden des OLAP als auch des Data Mining analysiert werden sollen. Dies gilt ebenso für alle vergleichbaren Datenmodelle, die mindestens dieselbe Mächtigkeit hinsichtlich der Modellierung bieten. In der weiteren Arbeit wird davon ausgegangen, dass das COCOM-Modell die Grundlage für den konzeptuellen Entwurf bildet.

4.2 Der logische Data-Warehouse-Entwurf

Das Ziel des logischen Entwurfs ist es, das im konzeptuellen Entwurf erstellte Schema auf ein Schema abzubilden, das das gegebene Ziel-Datenbanksystem berücksichtigt. Beim logischen Entwurf für ein Data Warehouse muss also berücksichtigt werden, ob für ein relationales oder ein multidimensionales Datenbanksystem modelliert werden soll [GR99] [HLV00]. Im Rahmen dieser Arbeit wird generell davon ausgegangen, dass die zu analysierenden Daten in einem Data Warehouse auf der Basis eines relationalen Datenbanksystems vorliegen. Die folgenden Betrachtungen beschränken sich darum auf logische Schemata, die auf dem relationalen Modell beruhen.

Für die Abbildung des konzeptuellen Schemas auf Tabellen wurden unterschiedliche logische Schematypen entwickelt. Die wichtigsten Vertreter sind das Flat-Schema, das Star-Schema sowie das Snowflake-Schema. Hier handelt es sich um mehrere Ansätze, die zeigen, wie aus einem konzeptuellen Schema ein logisches Schema im relationalen Modell erstellt werden kann. Zu jedem dieser Ansätze gehören spezifische Strukturierungsmerkmale für das logische Schema. Diese Merkmale beschreiben charakteristische Eigenschaften eines logischen Schemas, nicht aber das Schema für eine konkrete Datenbank. Die genannten Ansätze werden darum als *Schematypen* bezeichnet. Als Bezeichnung für die einzelnen Schematypen wurden aber die in der Literatur üblichen Begriffe Flat-Schema, Star-Schema und Snowflake-Schema beibehalten.

Mit diesen Schematypen werden in erster Linie die folgenden Ziele verfolgt:

- Die mehrdimensionale Struktur der zu analysierenden Daten soll in einem relationalen Datenbanksystem abgebildet werden.
- Typische Änderungen der Ausgangsdaten sollen in den Modellen einfach nachvollzogen werden können. Dies betrifft einerseits das Hinzufügen zusätzlicher Dimensionen und andererseits Änderungen in den Attributen und Hierarchien einer Dimension.
- Typische Anfragen eines ROLAP-Tools sollen auf den zu analysierenden Daten möglichst effizient ausgeführt werden können.

Alle genannten Schemata wurden mit dem Ziel entwickelt, OLAP-Analysen zu unterstützen. Für den Bereich Data Mining wurden bisher keine vergleichbaren spezifischen Schemata publiziert.

Das COCOM-Modell bildet eine wichtige Grundlage für die Analyse der logischen Schematypen. Deren Beschreibung besteht darum zunächst aus der Darstellung der Abbildungsschritte, mit denen ein konzeptuelles Schema, das auf dem COCOM-Modell beruht, in das dem Typ entsprechende logische Schema überführt werden kann. Hierbei werden für das Fakten-Schema und seine Bestandteile jeweils dieselben Bezeichnungen verwendet wie in Abschnitt 4.1.1. Bei Dimensionsattributen, Charakterisierungsattributen und nicht-aggregierbaren Maßen ist zu beachten, dass diese sich tatsächlich aus zwei Attributen im Sinne eines Datenbankschemas zusammensetzen können. Dies ist z.B. dann der Fall, wenn es zu einem Dimensionsattribut sowohl einen Identifikator als auch eine textuelle Beschreibung gibt, die im konzeptuellen Schema nicht unbedingt getrennt erfasst wurden. Die zunächst beschriebenen drei grundlegenden Schematypen sind in Abbildung 11 schematisch dargestellt.

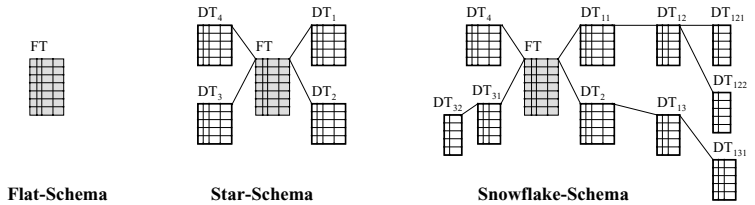

Abbildung 11: Schematypen für den logischen Data-Warehouse-Entwurf

4.2.1 Das Flat-Schema

Ein Flat-Schema besteht aus einer Faktentabelle, die einen oder mehrere Fakten sowie alle Informationen der Dimensionen umfasst, die mit den in der Tabelle repräsentierten Fakten verbunden sind [MK00]. Abbildung 12(a) zeigt ein Flat-Schema am Beispiel der Phagroh AG. Um das Flat-Schema für ein vorgegebenes Dimensionen-Schema zu erreichen, sind folgende Abbildungsschritte notwendig:

Erstelle für jedes Fakten-Schema $FS = (M, A, C, D, P)$ des Dimensionen-Schemas *DS* eine Tabelle *FT* und lege die Attribute wie folgt fest:

- Jedes Maß $m_i \in M$ des Fakten-Schemas wird zu einem Attribut,
- alle Primärattribute der Dimensionen $d \in D$ werden zu Attributen und bilden zusammen den Primärschlüssel der Faktentabelle *FT*,

- alle Dimensionsattribute $a \in A$ und alle Charakterisierungsattribute $c \in C$ werden ebenfalls zu Attributen in *FT*.

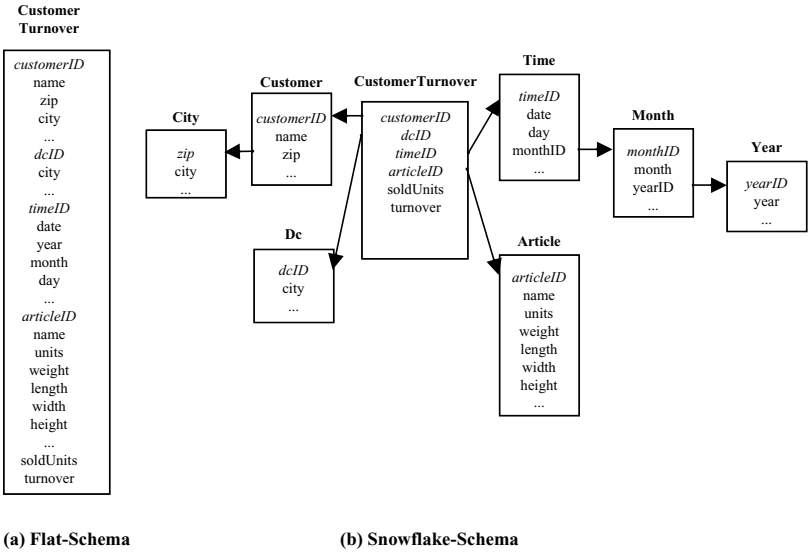

Abbildung 12: Flat- und Snowflake-Schema der Phagroh AG

4.2.2 Das Star-Schema

Ein Star-Schema besteht aus einer Faktentabelle und jeweils einer Dimensionstabelle pro Dimension [Kim96] [CD97] [KR+98] [HK01]. Die Dimensionstabellen enthalten hierbei alle Dimensionsattribute und Charakterisierungsattribute, die zu einer Dimension gehören. Die Abbildung eines gegebenen Dimensionen-Schemas in ein Star-Schema lässt sich wie folgt angeben:

Erstelle für jedes Fakten-Schema $FS = (M, A, C, D, P)$ des Dimensionen-Schemas *DS* eine Faktentabelle *FT*, deren Attribute nach folgenden Regeln festgelegt werden:

- Jedes Maß $m_i \in M$ des Fakten-Schemas wird zu einem Attribut,

- alle Primärattribute der Dimensionen $d \in D$ werden zu Attributen und bilden zusammen den Primärschlüssel der Faktentabelle *FT.*

Erstelle für jede Dimension $d = (A, C, H)$ des Dimensionen-Schemas *DS* eine Dimensionstabelle *DT* mit den folgenden Attributen:

- Jedes Dimensionsattribut $a \in A$ der Dimension d wird zu einem Attribut,
- jedes Charakterisierungsattribut $c \in C$ der Dimension *d* wird zu einem Attribut,
- bilde den Primärschlüssel der Tabelle *DT* aus allen Primärattributen der Dimension *d*.

Ein Star-Schema ermöglicht lediglich die Abbildung eines Fakten-Schemas auf Tabellen. Diese Abbildung kann für jedes Fakten-Schema separat angewandt werden, wodurch eine unverbundene Menge von Star-Schemata entsteht. Wird daraus ein konsolidiertes Schema erstellt, das sowohl gemeinsame Dimensionen als auch Beziehungen zwischen den Fakten berücksichtigt, so wird meist von einem *Constellation-Schema* oder *Galaxy-Schema* gesprochen [CD97] [MK00]. Das Star-Schema für die Phagroh AG ist bei der Einführung der Beispielszenarien in Abbildung 6 angegeben.

Einige wichtige Merkmale des auf dem beschriebenen Weg erstellten Star-Schemas sind:

- Alle Dimensionstabellen sind vollständig denormalisiert,
- die Information über die hierarchischen Beziehungen ist nicht Bestandteil des logischen Schemas und
- die Information über die möglichen Aggregationspfade innerhalb einer Dimension ist ebenfalls nicht Bestandteil des logischen Schemas.

4.2.3 Das Snowflake-Schema

Ein Snowflake-Schema besteht aus einer Faktentabelle und aus mehreren Dimensionstabellen für jede Dimension, die mit der Faktentabelle verbunden ist. Diese zusätzlichen Dimensionstabellen beruhen auf einer zumindest teilweisen Normalisierung von Dimensionen. Es gibt eine Reihe von Varianten für dieses Schema. Hier wird zunächst die einfachste, herstellerunabhängige Variante beschrieben. Bei dieser gibt es zu jedem Dimensionsattribut eine eigene Dimensionstabelle

[CD97] [MK00]. Um aus einem Dimensionen-Schema ein Snowflake-Schema zu generieren, müssen die folgenden Schritte durchlaufen werden:

Erstelle für jedes Fakten-Schema $FS = (M, A, C, D, P)$ des Dimensionen-Schemas *DS* eine Tabelle *FT*, deren Attribute nach folgenden Regeln festgelegt werden:

- Jedes Maß $m_i \in M$ des Fakten-Schemas wird zu einem Attribut,
- alle Primärattribute der Dimensionen $d \in D$ werden zu Attributen und bilden zusammen den Primärschlüssel der Faktentabelle *FT*.

Erstelle für jedes Dimensionsattribut $a \in A$ einer Dimension $d = (A, C, H)$ des Dimensionen-Schemas *DS* eine Dimensionstabelle *DT* mit folgenden Attributen:

- Das Dimensionsattribut *a* wird zum Primärschlüssel der Dimensionstabelle *DT*,
- jedes Charakterisierungsattribut c' mit $(a,c') \in H$ wird ein Attribut von *DT* und
- jedes Dimensionsattribut a' mit $(a,a') \in H$ wird ebenfalls ein Attribut von *DT*.

Ein wichtiger Unterschied beim Snowflake-Schema im Vergleich zum Star-Schema ist, dass die hierarchische Beziehung der Dimensionsattribute explizit modelliert ist. Wie in Abbildung 12(b) zu sehen ist, ergeben sich dadurch auch im Beispiel der Phagroh AG zusätzliche Tabellen. Wie beim Star-Schema wird aber die Information bezüglich möglicher Aggregationspfade nicht modelliert.

4.2.4 Varianten des Snowflake-Schemas

Es gibt eine Reihe von Varianten des Snowflake-Schemas. Hierbei handelt es sich meist um herstellerspezifische Abwandlungen. Eine dieser Alternativen stammt von Informix und wird im Rahmen dieser Arbeit als *Informix-Schema* bezeichnet. Zur Erstellung dieses Schemas sind die folgenden Schritte notwendig:

Erstelle für jedes Fakten-Schema $FS = (M, A, C, D, P)$ des Dimensionen-Schemas *DS* eine Faktentabelle *FT*, deren Attribute nach folgenden Regeln festgelegt werden [Inf97]:

- Jedes Maß $m_i \in M$ des Fakten-Schemas wird zu einem Attribut,
- alle Primärattribute der Dimensionen $d \in D$ werden zu Attributen und bilden zusammen den Primärschlüssel der Faktentabelle *FT*.

In diesem ersten Schritt unterscheidet sich das Informix-Schema nicht vom Snowflake-Schema. Unterschiedlich ist allerdings die Abbildung der einzelnen Dimensionen auf Tabellen. Für jedes Dimensionsattribut $a \in A$ einer Dimension $d = (A, C, H)$ des Dimensionen-Schemas *DS* eine Dimensionstabelle *DT* mit den folgenden Attributen gilt:

- Das Dimensionsattribut *a* wird zum Primärschlüssel der Dimensionstabelle *DT* und
- jedes Charakterisierungsattribut c' mit $(a,c') \in H$ wird ein Attribut von *DT*.

Erstelle für jede Dimension $d = (A, C, H)$ des Dimensionen-Schemas *DS* eine zentrale Dimensionstabelle *DZ* mit den folgenden Attributen:

- Jedes Dimensionsattribut $a \in A$ der Dimension *d* wird zu einem Attribut in *DZ*.
- Der Primärschlüssel der zentralen Dimensionstabelle *DZ* besteht aus den Primärattributen der Dimension *d*.

Von der klassischen Snowflake-Variante, wie sie im vorangehenden Abschnitt erläutert wurde, unterscheidet sich das Informix-Schema dadurch, dass eine explizite Modellierung hierarchischer Beziehungen innerhalb einer Dimension entfällt. Durch die Reihenfolge der Attribute innerhalb der zentralen Dimensionstabelle einer Dimension kann allerdings eine einfache Hierarchie angegeben werden.

Eine weitere Variante des Snowflake-Schemas wird hier als *Strategy-Schema* bezeichnet [Mic99]. Sie unterscheidet sich in zwei Punkten von der klassischen Snowflake-Variante:

- Die Dimensionstabelle *DT* enthält zusätzliche Attribute. Nicht nur jedes Dimensionsattribut a' mit $(a,a') \in H$ wird ein Attribut von *DT*, sondern auch alle Dimensionsattribute $\tilde{a}$, die transitiv von *a* abhängen.

- Für jedes nicht-aggregierbare Maß *m* wird eine zusätzliche Tabelle *QT* eingeführt, die sowohl den Identifikator von *m* als auch die ausführliche Beschreibung von *m* enthält.

Eine Mischung aus Star- und Snowflake-Schema stellt das in [MK00] vorgestellte *Star-Cluster-Schema* dar. Hier werden die Dimensionstabellen des Star-Schemas nur an den Stellen normalisiert, wo Dimensionsattribute unterschiedlicher Dimensionen in einer hierarchischen Beziehung zu demselben Dimensionsattribut stehen. Für dieses gemeinsame Attribut wird eine separate Dimensionstabelle verwendet. Das Attribut selbst sowie die hierarchisch übergeordneten Attribute werden als Sub-Dimension bezeichnet. Diese Art der Modellierung wird hier nicht weiter betrachtet, da es sich nur um eine leichte Variation des Star-Schemas handelt.

Im Rahmen dieser Arbeit wird unter einem Snowflake-Schema immer die in Abschnitt 4.2.3 erläuterte Variante verstanden, während die Alternativen explizit als Informix-Schema bzw. Strategy-Schema bezeichnet werden.

4.2.5 Schemata für Data-Mining-Analysen

Wie bereits erwähnt, wurden für den Bereich Data Mining bisher keine vergleichbaren Schemata publiziert. Durch eine Analyse der unterschiedlichen Data-Mining-Algorithmen sowie der Möglichkeiten, die die verschiedenen Data-Mining-Werkzeuge den Benutzern bieten, um die Ausgangsdaten für die Analyse zu spezifizieren, können allerdings Rückschlüsse auf das implizit zu Grunde gelegte Datenmodell gezogen werden [Thi98] [Ibm99] [Sas99] [Ora00a] [Bou01] [Kra01].

Die Erstellung von Clustermodellen basiert auf den Eigenschaften der zu gruppierenden Objekte. Dementsprechend benötigt die Clusterbildung diese Information als Eingabe. Typische Algorithmen operieren hierbei auf einer Datenmatrix, die für jedes Objekt eine Zeile und für jede Eigenschaft eine Spalte enthält [JMF99] [HK01]. Diese entspricht strukturell einer Tabelle in einer relationalen Datenbank. Dementsprechend erwarten Data-Mining-Werkzeuge die Angabe einer einzigen Tabelle und der relevanten Attribute, auf deren Grundlage dann ein Clustermodell erstellt wird.

Ähnlich wie bei der Clusterbildung müssen auch für Klassifikationsmodelle und Regressionsmodelle die Eigenschaften der Objekte, für die ein Modell erstellt werden soll, bekannt sein. Eine Eigenschaft, die Klassenzugehörigkeit der einzelnen Objekte bzw. der Wert des Zielattributs, ist hier explizit erforderlich. Damit weisen auch die Eingabedaten für Klassifikationsalgorithmen die Struktur einer Tabelle auf.

Beim wichtigsten Vertreter der Abhängigkeitsmodelle, den Assoziationsregeln, wird als Eingabe die Information erwartet, welche Objekte in welchen Transaktionen vorhanden sind. Hierfür kommen unterschiedliche Datenstrukturen in Frage. Die Information über die zu einer Transaktion gehörenden Objekte kann in Form einer geschachtelten Relation, einer Bitvektordarstellung oder aber auch in einer Relation, deren Tupel aus Transaktions/Objektkombinationen bestehen, bereitgestellt werden [AS94] [SA96]. Von den verfügbaren Algorithmen werden diese Strukturen als interne Repräsentation genutzt, während die Werkzeuge in der Regel von einer Relation als Eingabe ausgehen, die sämtliche vorhandenen Kombinationen aus Transaktionen und Objekten als Tupel enthält. Zusätzliche Eingabedaten sind notwendig, wenn bei der Ableitung der Assoziationsregeln Taxonomien berücksichtigt werden sollen [SA95]. Die Information bezüglich der hierarchischen Beziehungen wird von den Data-Mining-Werkzeugen in der Regel in einer Relation erwartet, die für jede Hierarchiestufe ein Attribut enthält. Diese Relation entspricht in ihrer Struktur also einer Dimensionstabelle des Star-Schemas.

Werden Sequenzmodelle auf der Basis von Transaktionsdaten erstellt, die auch die Grundlage für Abhängigkeitsanalysen bilden können, so gelten bezüglich der Ausgangsdaten die entsprechenden Aussagen zu Assoziationsregeln [AS95]. Bilden zeitbezogene, kontinuierliche Werte die Ausgangsbasis für eine Sequenzanalyse, so werden diese in der Regel in einer Relation bereitgestellt, die zu jeder Sequenz einerseits den Zeitstempel und andererseits die zu diesem Zeitpunkt festgestellten Werte enthält.

Diese Analyse zeigt, dass die wesentliche Datenstruktur, die von Data-Mining-Verfahren vorausgesetzt wird, eine Relation ist, die alle für die Erstellung des Data-Mining-Modells notwendigen Attribute umfasst. Das Datenmodell, das von Data-Mining-Verfahren implizit vorausgesetzt wird, entspricht also einem Flat-Schema, wobei dieses nicht aus einem mehrdimensionalen, konzeptuellen Schema hervorgegangen sein muss. Trotzdem besteht ein wichtiger Unterschied

zu dem in Abschnitt 4.2.1 beschriebenen Flat-Schema. Dort wird mit Blick auf OLAP-Analysen davon ausgegangen, dass für die einzelne Analyse die Fakten und damit in Verbindung stehende Dimensionen bereitgestellt werden müssen. Diese Attribute können dann in einer Tabelle zusammengefasst werden. Data-Mining-Verfahren können demgegenüber auf einer beliebigen Teilmenge der im Data Warehouse verfügbaren Attribute sowie auf davon abgeleiteten Attributen arbeiten. Ein Schema, das jede einzelne Analyse optimal unterstützen soll, muss für jede Analyse alle notwendigen Attribute in einer Tabelle bereitstellen. Da die für Analysen benötigten Attributkombinationen vorab nicht bekannt sind, muss das Schema sämtliche möglichen Attributkombinationen in separaten Tabellen oder aber eine Tabelle mit allen Attributen umfassen. Beide Ansätze führen zu redundanter Datenhaltung und steigern die Datenmenge im Data Warehouse in der Regel erheblich. Darüber hinaus wird bei der Bereitstellung aller Information in einer Tabelle für jede Analyse das Durchlaufen dieser Tabelle notwendig. Dies wird die Performanz der Analysen negativ beeinflussen, da sehr häufig nur ein kleiner Ausschnitt des Gesamtdatenbestandes für einen spezifischen Analyseschritt notwendig ist. Die skizzierten Ansätze zur Bereitstellung der Daten für Data Mining erweisen sich damit beide als nicht praktikabel.

Als Konsequenz müssen die Eingangsdaten für ein Data-Mining-Verfahren vor dem eigentlichen Analyseschritt auf der Grundlage der im Data Warehouse verfügbaren Information bereitgestellt werden. Hierfür sind separate Schritte der Datenvorverarbeitung und der Transformation notwendig, wie dies in Abschnitt 2.3 für den KDD-Prozess bereits beschrieben wurde. Diese Schritte sind in Ergänzung zu der Datenaufbereitung für das Data Warehouse notwendig, da sie individuell für die einzelne Analyse angepasst werden müssen. Data-Mining-Werkzeuge tragen diesem Umstand Rechnung, indem sie die notwendige Vorverarbeitungsfunktionalität in unterschiedlichem Umfang anbieten.

Es bleibt festzuhalten, dass es kein logisches Schema gibt, das speziell für Data Mining geeignet ist. In der Regel wird für jede Data-Mining-Analyse individuell ein Ausschnitt der im Data Warehouse verfügbaren Daten in Form einer Tabelle bereitgestellt. Für das logische Schema des Data Warehouse können also das Flat-Schema, das Star-Schema sowie das Snowflake-Schema und seine Varianten in Betracht gezogen werden.

4.2.6 Bewertungskriterien für logische Schematypen

Die zu berücksichtigenden Schematypen werden in den folgenden Abschnitten bewertet. Im Rahmen dieser Bewertung werden insbesondere Eigenschaften der Schematypen berücksichtigt, die deren Eignung für OLAP- und Data-Mining-analysen beeinflussen. Darüber hinaus werden aber auch anwendungsunabhängige Eigenschaften betrachtet.

Im ersten Abschnitt wird untersucht, inwieweit die einzelnen logischen Schemata die Semantik des konzeptuellen Schemas korrekt wiedergeben. Da sich das COCOM-Modell als geeignet erwiesen hat, sowohl OLAP als auch Data Mining zu unterstützen, erfolgt die Bewertung auf der Basis der Eigenschaften, die für dieses Modell als besonders wichtig identifiziert wurden. Für den klassischen Datenbankentwurf wird die Korrektheit der Abbildungsschritte beispielsweise in [MS89] betrachtet. Korrektheit heißt hier einerseits, dass alle im konzeptuellen Schema modellierten Zusammenhänge auch im logischen Schema erhalten bleiben. Andererseits soll im logischen Schema auch nichts modelliert werden können, was dem konzeptuellen Schema widerspricht.

Ein Data Warehouse muss permanent veränderten Rahmenbedingungen angepasst werden. So ziehen beispielsweise Änderungen in den Quellsystemen eines Data Warehouse oder in den Analyseanforderungen an das Data Warehouse ein verändertes konzeptuelles Schema nach sich. Dies betrifft insbesondere das Hinzufügen zusätzlicher Dimensionen sowie Änderungen in den Attributen und Hierarchien einer Dimension. Als eine der wesentlichen, anwendungsunabhängigen Eigenschaften der Schematypen wird deshalb im zweiten Abschnitt analysiert, welcher Aufwand mit dem Nachvollziehen dieser Änderungen im logischen Schema verbunden ist. Die verschiedenen Schematypen weisen einen unterschiedlichen Grad der Normalisierung auf. Die in den einzelnen Schematypen damit verbundene Redundanz hat sowohl Einfluss auf das Datenvolumen des Data Warehouse als auch auf die Zahl potenzieller, inkonsistenter Änderungen. In wieweit sich die einzelnen Schematypen hinsichtlich der Redundanz unterscheiden, wird ebenfalls im zweiten Abschnitt betrachtet.

Im dritten Abschnitt der Bewertung wird schließlich untersucht, welche Auswirkungen die unterschiedlichen logischen Schematypen auf die Performanz der Analyseanwendungen in den Bereichen OLAP und Data Mining haben. Dieses Kriterium betrifft die unterschiedlichen Analyseanwendungen gleichermaßen.

OLAP-Analysen werden direkt auf der Grundlage des logischen Schemas durchgeführt. Beim Data Mining bildet das logische Schema dagegen die Basis für die notwendigen Schritte zur Datenvorverarbeitung.

4.2.7 Informationsgehalt

Das in Abschnitt 4.1.1 vorgestellte konzeptuelle Modell weist wichtige Merkmale für die Unterstützung von OLAP und Data Mining auf. Ein optimales logisches Schema hierzu sollte über dieselbe Ausdrucksmächtigkeit verfügen. So gehen im Rahmen des logischen Entwurfs keine Informationen aus dem konzeptuellen Schema verloren, und es können im logischen Schema keine Widersprüche zum konzeptuellen Schema entstehen. Deshalb wird hier zunächst diskutiert, inwieweit sich die Modellierungsmöglichkeiten der unterschiedlichen logischen Schematypen von denen des konzeptuellen Modells unterscheiden. Die Grundlage hierfür bilden die in Abschnitt 4.1.3 für den Data-Warehouse-Entwurf als wichtig identifizierten Modellierungsmöglichkeiten des COCOM-Modells. Für diese wird einzeln überprüft, inwieweit es gleichwertige Modellierungsmöglichkeiten in den logischen Schematypen gibt. Die wichtigsten Merkmale des COCOM-Modells und die entsprechenden Bewertungen der Schematypen sind in Tabelle 1 zusammengefasst. Ein ‘X’ in der Tabelle bedeutet hierbei, dass der jeweilige Zusammenhang in einem Typ des logischen Schemas ausgedrückt werden kann. Ist dies nicht der Fall, dann enthält die Tabelle den Eintrag ‘-’.

Das Flat-Schema unterstützt lediglich eine der aufgeführten Modellierungsmöglichkeiten, da alle Attribute, die zu einem gegebenen Fakten-Schema gehören, in einer einzigen Tabelle enthalten sind. Damit besteht keine Möglichkeit, zwischen Fakten und Dimensionen zu unterscheiden bzw. die einzelnen Dimensionen weiter zu strukturieren. Das Flat-Schema bietet lediglich die Möglichkeit, Fakten aus mehreren Maßen zusammenzusetzen. Diese Modellierungsmöglichkeit wird auch in allen anderen Schematypen berücksichtigt.

Im Star-Schema wird zwischen Fakten und Dimensionen unterschieden. Es gibt allerdings keine Möglichkeit, die einzelnen Dimensionen weiter zu strukturieren. Es fehlt sowohl die Möglichkeit, die hierarchischen Beziehungen innerhalb einer Dimension zu modellieren, als auch die Unterscheidung zwischen Dimensionsattributen und Charakterisierungsattributen. Die Aggregationsmöglichkeiten können in diesem Schematyp ebensowenig festgelegt werden.

Die umfangreichsten Modellierungsmöglichkeiten bieten das Snowflake-Schema und das Strategy-Schema. Wie beim Star-Schema gibt es bei diesen eine Unterscheidung zwischen Fakten und Dimensionen. Hier besteht allerdings zusätzlich die Möglichkeit, die hierarchischen Beziehungen einer Dimension zu modellieren. Hierbei können auch unterschiedliche Hierarchien berücksichtigt werden. Charakterisierungsattribute sind all diejenigen Attribute in einer Dimensionstabelle, die nicht an einer Schlüssel-Fremdschlüssel-Beziehung beteiligt sind. Wie bei den anderen Schematypen besteht auch hier keine Möglichkeit, zulässige Aggregationspfade anzugeben. Beim Informix-Schema sind die Modellierungsmöglichkeiten im Vergleich dazu weiter eingeschränkt, da hier keine explizite Angabe der hierarchischen Beziehungen möglich ist und somit auch die Angabe paralleler Hierarchien entfällt. Durch die Reihenfolge der Attribute in der zentralen Dimensionstabelle kann lediglich eine einfache Hierarchie angegeben werden.

Das Snowflake-Schema und das Strategy-Schema bieten insgesamt die umfangreichsten Modellierungsmöglichkeiten. Abgesehen von den möglichen Aggregationspfaden kann alle Information, die im konzeptuellen Schema enthalten ist, auch in diesen Schematypen ausgedrückt werden.

Tabelle 1: Modellierungsmöglichkeiten der logischen Schematypen

Wichtige Modellierungsmöglichkeiten des COCOM-Modells	**Flat-Schema**	**Star-Schema**	**Snowflake-Schema**	**Informix-Schema**	**Strategy-Schema**
Fakten umfassen mehrere Maße	X[a]	X	X	X	X
explizite Unterscheidung zwischen Fakten und Dimensionen	-[b]	X	X	X	X
explizite hierarchische Beziehungen	-	-	X	(X)	X
mehrere Hierarchien	-	-	X	-	X
explizite Unterscheidung zwischen Dimensions- und Charakterisierungsattributen	-	-	X	X	X
Angabe von Aggregationsmöglichkeiten	-	-	-	-	-

a. X bezeichnet Modellierungsmöglichkeiten, die unterstützt werden.
b. - bezeichnet Modellierungsmöglichkeiten, die nicht unterstützt werden.

4.2.8 Änderungsaufwand und Redundanz

Die Quellsysteme, die die Datengrundlage für ein Data Warehouse bilden, sind laufenden Änderungen unterworfen. Dies gilt sowohl für die gespeicherten Daten als auch für deren Schemata. Änderungen im Schema der Quellsysteme können Anpassungen des konzeptuellen Schemas des Data Warehouse notwendig machen. Folgende Änderungen können im konzeptuellen Modell grundsätzlich auftreten:

- das Hinzufügen von Maßen eines Fakts,
- das Hinzufügen von Fakten und damit eines ganzen Fakten-Schemas,
- das Hinzufügen von Dimensionen,
- das Hinzufügen eines Attributs einer Dimension sowie
- die Änderung der Hierarchie(n) einer Dimension.

In der Regel bilden mehrere dieser Basis-Änderungsoperationen gemeinsam die notwendigen Anpassungen eines Data-Warehouse-Schemas. Für alle Operationen, die dem Schema etwas hinzufügen, muss natürlich auch das entsprechende Löschen berücksichtigt werden. Da sich der im Folgenden betrachtete Aufwand für das Hinzufügen und das Entfernen allerdings nicht unterscheidet, wird in diesem Abschnitt nur das Hinzufügen berücksichtigt. Die verschiedenen logischen Schematypen unterscheiden sich hinsichtlich des Änderungsaufwands, der durch die einzelnen Basis-Änderungsoperationen notwendig wird. Der jeweils aus einer Operation resultierende Änderungsaufwand ist in Tabelle 2 angegeben. Dort steht *f* für die Anzahl der Fakten, *m* für die Anzahl der Maße pro Fakt, *d* für die Anzahl der Dimensionen, *a* für die Anzahl der Dimensionsattribute pro Dimension sowie *c* für die Anzahl der Charakterisierungsattribute pro Dimension. Zur Vereinfachung wird hierbei eine gleichmäßige Struktur des Schemas angenommen, d.h., die Werte *a* und *c* gelten hier für alle Dimensionen, obwohl sie in einem realen Schema für die einzelnen Dimensionen individuell unterschiedlich sein können. In der Kopfzeile der Tabelle ist für jeden Schematyp zusätzlich die Gesamtzahl der Tabellen angegeben.

In Tabelle 2 ist in Abhängigkeit vom Schematyp beschrieben, welche Tabellen von einer bestimmten Änderungsoperation betroffen sind. Zusätzlich ist aufgeführt, wieviele Tabellen auf Grund einer Änderungsoperation neu angelegt werden müssen (Angabe nach N:) und bei wievielen Tabellen in Folge der

Tabelle 2: Änderungsaufwand für logische Schematypen

	Flat-Schema **f Tabellen**	**Star-Schema** **f + d Tabellen**	**Snowflake-Schema Strategy-Schema** **f + d * a Tabellen**	**Informix-Schema** **f+d+d*a Tabellen**
Maß	Zugehörige Faktentabelle wird ergänzt.	Zugehörige Faktentabelle wird ergänzt.	Zugehörige Faktentabelle wird ergänzt.	Zugehörige Faktentabelle wird ergänzt.
	N[a]: 0 C[b]: 1	N: 0 C: 1	N: 0 C: 1	N: 0 C: 1
Fakt	Neue Faktentabelle	Neue Faktentabelle	Neue Faktentabelle	Neue Faktentabelle
	N: 1 C: 0	N: 1 C: 0	N: 1 C: 0	N: 1 C: 0
Dimension	Alle Faktentabellen, die sich auf diese Dimension beziehen. In diesen sind alle Attribute, die zu der neuen Dimension gehören, zu ergänzen.	Neue Dimensionstabelle und alle Faktentabellen, die sich auf die neue Dimension beziehen. Dort muss jeweils der Schlüssel um die Schlüsselattribute der neuen Dimension erweitert werden.	Neue Dimensionstabellen und alle Faktentabellen, die sich auf die neue Dimension beziehen. Dort muss jeweils der Schlüssel um die Schlüsselattribute der neuen Dimension erweitert werden.	Neue Dimensionstabellen und alle Faktentabellen, die sich auf die neue Dimension beziehen. Dort muss jeweils der Schlüssel um die Schlüsselattribute der neuen Dimension erweitert werden.
	N: 0 C: max. f	N: 1 C: max. f	N: a C: max. f	N: 1 + a C: max. f
Dimensionsattribut	Alle Faktentabellen, die Attribute derselben Dimension beinhalten. Das Attribut wird dort lediglich hinzugefügt.	Die Dimensionstabelle der betroffenen Dimension. Dort wird das Attribut hinzugefügt.	Neue Dimensionstabelle sowie andere Dimensionstabellen und Faktentabellen, in denen Fremdschlüsselbeziehungen angepasst werden müssen.	Neue Dimensionstabelle sowie andere Dimensionstabellen und Faktentabellen, in denen Fremdschlüsselbeziehungen angepasst werden müssen.
	N: 0 C: max. f	N: 0 C: 1	N: 1 C: max. f + a	N: 1 C: max. f + 1 + a

Tabelle 2: Änderungsaufwand für logische Schematypen

	Flat-Schema **f Tabellen**	**Star-Schema** **f + d Tabellen**	**Snowflake-Schema Strategy-Schema** **f + d * a Tabellen**	**Informix-Schema** **f+d+d*a Tabellen**
Charakterisierungsattribut	Alle Faktentabellen, die Attribute derselben Dimension beinhalten. Das Attribut wird dort lediglich hinzugefügt.	Die Dimensionstabelle der betroffenen Dimension. Dort wird das Attribut hinzugefügt.	Eine Dimensionstabelle der betroffenen Dimension. Dieser wird das Attribut hinzugefügt.	Eine Dimensionstabelle der betroffenen Dimension. Dieser wird das Attribut hinzugefügt.
	N: 0 C: max. f	N: 0 C: 1	N: 0 C:1	N: 0 C: 1
Hierarchie	Keine	Keine	Alle Dimensionstabellen der betroffenen Dimension. Die Zuordnung der Attribute zu Dimensionstabellen und Fremdschlüsselbeziehungen müssen angepasst werden.	Die zentrale Dimensionstabelle mit den Verweisen auf alle Dimensionsattribute dieser Dimension.
	N: 0 C: 0	N: 0 C: 0	N: 0 C: max. f + a	N: 0 C: 1

a. Gibt die Anzahl notwendiger zusätzlicher Tabellen an.
b. Gibt die Anzahl der Tabellen an, deren Schema geändert werden muss.

Änderungsoperation das Schema geändert werden muss (Angabe nach C:). Die beiden letzteren Angaben haben unterschiedliches Gewicht. Das Hinzufügen einer Tabelle betrifft in erster Linie den Ladeprozess des Data Warehouse. Für jede neu hinzukommende Tabelle muss im Data Warehouse ein entsprechender Ladeprozess bereitgestellt werden, der im zur Verfügung stehenden Zeitfenster die zusätzlichen Daten für diese neue Tabelle bereitstellen muss. Änderungen am Schema bestehender Tabellen betreffen hingegen auch Daten und Anwendungen, die auf der Basis des ursprünglichen Schemas erstellt wurden. Daten müssen im neuen Schema erneut geladen werden. Anwendungen müssen entsprechend der Schemaänderung angepasst werden. Dies kann für die einzelne Anwendung mit erheblichem Aufwand verbunden sein. Die Zahl der potenziell betroffenen Anwendungen steigt mit der Zahl geänderter Tabellen. Zum Vergleich der Sche-

matypen wird darum die Anzahl der zu ändernden Tabellen als die entscheidende Größe herangezogen. Tabelle 2 enthält für das Hinzufügen eines Dimensionsattributs im Snowflake-Schema beispielsweise die Einträge *N: 1* und *C: max f + a*. Damit wird angegeben, dass für das zusätzliche Dimensionsattribut eine neue Tabelle notwendig ist und dass maximal in allen Faktentabellen und den anderen Dimensionstabellen der geänderten Dimension eine Änderung erfolgen muss.

Die Tabelle zeigt, dass in allen fünf Schematypen für das Hinzufügen von Fakten und Maßen der gleiche Aufwand notwendig ist. Unterschiede ergeben sich allerdings sobald die Attribute einer Dimension ergänzt oder deren hierarchische Beziehungen geändert werden. Im Star-Schema muss jeweils maximal eine Tabelle bei solchen Veränderungen angepasst werden. Dagegen sind im Flat-Schema von einem zusätzlichen Attribut potenziell alle Faktentabellen betroffen während sich im Snowflake-, im Strategy- und im Informix-Schema das Hinzufügen eines Dimensionsattributs sowie die Änderung der Hierarchie innerhalb einer Dimension im Schema von bis zu f + a Tabellen niederschlagen kann. Damit werden im Star-Schema insgesamt die wenigsten Änderungen durch ein verändertes konzeptuelles Schema verursacht.

Die Bewertung der Redundanz soll hier nur qualitativ erfolgen. Im Flat-Schema existiert für jedes Fakten-Schema lediglich eine Tabelle. Diese enthält damit offensichtlich sehr viel redundante Daten, da alle Informationen zu den einzelnen Dimensionen mehrfach enthalten sind. Bei der Phagroh AG müssten beispielsweise bei jedem Verkauf eines Artikels, alle zu diesem Artikel gehörenden Informationen aus der Dimension *Article* abgelegt werden. Im Star-Schema ist diese Redundanz dagegen nur noch innerhalb der einzelnen Dimensionen zu finden, während sie in den verschiedenen Varianten des Snowflake-Schemas weitestgehend vermieden wird. Die Redundanz in den einzelnen Schematypen ist in zweierlei Hinsicht von Bedeutung. Einerseits erhöht sie den Umfang der Daten im Data Warehouse, andererseits vergrößert sie die Wahrscheinlichkeit, dass Änderungen inkonsistente Daten zur Folge haben. Insbesondere beim Flat-Schema führt der hohe Grad der Redundanz in der Regel zu einem Data Warehouse, das erheblich mehr Speicherplatz für die Basisdaten in Anspruch nimmt, als dies bei den anderen Schematypen der Fall ist. Schließlich werden hier die Informationen kompletter Dimensionen mehrfach verwaltet. Die Redundanz, die sich beim Star-Schema innerhalb der einzelnen Dimensionen ergibt, fällt dagegen in Bezug auf den notwendigen Speicherplatz weniger ins Gewicht. Weitestgehend vermieden wird Redundanz allerdings nur im Snowflake-Schema und seinen Varianten.

4.2.9 Performanz

Die Wahl des logischen Schemas kann die Performanz von Data-Mining- und OLAP-Anwendungen beeinflussen. Um diesen Einfluss beurteilen zu können, muss zunächst betrachtet werden, wie diese Anwendungen auf die Daten des Data Warehouse zugreifen. In jedem Fall erzeugen die Anwendungen Folgen von SQL-Anfragen an das Data Warehouse, allerdings mit unterschiedlichen Zielen.

Im Bereich ROLAP erfolgt die Analyse direkt auf den Tabellen des Data Warehouse. Es werden Anfragen generiert, die das gewünschte Ergebnis bereitstellen. Jede dieser Anfragen greift je nach gewähltem Schema auf eine oder mehrere Tabellen des Data Warehouse zu. Für Data-Mining-Analysen ist, wie in Abschnitt 4.2.5 beschrieben, zunächst eine Vorverarbeitung der Daten notwendig, um die relevanten Daten in einer Tabelle bereitzustellen. Dies kann wiederum durch eine SQL-Anfrage oder eine ganze Sequenz von SQL-Anfragen erfolgen. Hierbei besteht die Wahl zwischen einer Materialisierung der Eingangsdaten und einer Bereitstellung über Sichten. Vom eigentlichen Data-Mining-Algorithmus werden diese Eingangsdaten dann einmal oder mehrmals gelesen.

4.2.9.1 Anfrageklassifikation

Die weitere Untersuchung des Einflusses von Schematypen auf die Performanz von Anwendungen muss sich im Kontext dieser Arbeit also auf die Anfragen stützen, die von OLAP- und Data-Mining-Werkzeugen im Rahmen der Vorverarbeitung und bei der eigentlichen Analyse generiert werden. Die zu einer Klasse gehörenden Anfragen unterscheiden sich in den folgenden Merkmalen:

- Die Zahl der relevanten Fakten. Die möglichen Merkmalsausprägungen sind: ‘kein’, ‘ein’ oder ‘mehrere’.
- Die Zahl der relevanten Dimensionen. Hier sind die Merkmalsausprägungen ebenfalls: ‘keine’, ‘eine’ oder ‘mehrere’.
- Die Zahl der relevanten Dimensionsattribute in den Dimensionen. Hier sind nur zwei Merkmalsausprägungen von Bedeutung: ‘ein’, ‘mehrere’.

Die Anfrageklassen, die daraus resultieren, sind in Tabelle 3 dargestellt. Es gibt darin nicht für alle denkbaren Kombinationen der Merkmalsausprägungen eine eigene Klasse. So sind z.B. alle Kombinationen, die sich auf keine Faktentabelle und keine Dimension beziehen ebenso wenig sinnvoll, wie diejenigen, die sich

auf keine Faktentabelle dafür aber auf mehrere Dimensionen beziehen. Im letzten Fall wäre das Resultat einer Anfrage lediglich das kartesische Produkt der beteiligten Dimensionen. Darüber hinaus wurden Kombinationen bereits zusammengefasst, deren Unterscheidung für die weiteren Untersuchungen in dieser Arbeit nicht notwendig ist. Insbesondere ist hier die Unterscheidung zwischen einer oder mehreren relevanten Dimensionen ohne Bedeutung. Die Bezeichnung der Anfrageklassen enthält jeweils ein Tripel, das an jeder Stelle die Werte 0,1 oder x aufweisen kann. Das erste Element eines Tripels steht hierbei für die Anzahl relevanter Fakten, das zweite Element für die Anzahl relevanter Dimensionen und das dritte Element schließlich für die Anzahl relevanter Dimensionsattribute. Die Anfrage in Abbildung 9 bezieht sich auf einen Fakt *turnover*, die Dimensionen *Customer*, *Time* und *Dc* sowie in der Dimension *Time* auf mehrere Attribute. Sie gehört damit zur Klasse A_{1xx}.

Tabelle 3: Klassifikation der Anfragen

Klasse	relevante Fakten	relevante Dimensionen	relevante Dimensionsattribute
$\mathbf{A_{100}}$	ein	keine	-
$\mathbf{A_{011}}$	kein	eine	ein
$\mathbf{A_{01x}}$	kein	eine	mehrere
$\mathbf{A_{1x1}}$	ein	eine/mehrere	ein
$\mathbf{A_{xx1}}$	mehrere	eine/mehrere	ein
$\mathbf{A_{1xx}}$	eine	eine/mehrere	mehrere
$\mathbf{A_{xxx}}$	mehrere	eine/mehrere	mehrere

Durch die Klassifikation wird die Auswahl einer Menge repräsentativer Anfragen und deren detaillierte Analyse überflüssig. Sie ist sowohl für die ROLAP-Anfragen als auch für die Datenaufbereitungsschritte im Bereich Data Mining relevant. Bei der Bearbeitung von OLAP-Anfragen werden in der Regel SQL-Anfragen erstellt, die auf einen Fakt, mehrere Dimensionen sowie auf ein oder mehrere Dimensionsattribute pro Dimension zugreifen. Darum sind insbesondere die Anfrageklassen A_{1x1} und A_{1xx} für OLAP von Bedeutung. Bei Data-Mining-Analysen gibt es demgegenüber eine größere Bandbreite relevanter Anfrageklassen. So können beispielsweise Cluster-Analysen häufig auf der Grundlage der Informationen in einzelnen Dimensionen durchgeführt werden. Für die Bildung von Kunden-Clustern für die Phagroh AG sind in erster Linie die Daten der

Dimension *Customer* relevant. Die Datenaufbereitung für solche Fragestellungen fällt damit in die Anfrageklassen A_{011} und A_{01x}. Aber auch Anfragen, in denen mehrere Fakten berücksichtigt werden, sind bei der Datenaufbereitung für Data Mining vertreten. Bei einer Klassifikation der Kunden kann es beispielsweise notwendig sein, neben den Informationen aus der Kundendimension auch Fakten wie Umsatz, Werbeaktionen, u.ä. zu berücksichtigen. Damit haben beim Data Mining auch die Anfrageklassen A_{xx1} und A_{xxx} eine Bedeutung.

4.2.9.2 Komplexität der Anfrageklassen

Beim integrierten Einsatz von OLAP und Data Mining müssen somit Anfragen aus praktisch allen vorgestellten Anfrageklassen berücksichtigt werden. Im nächsten Schritt soll darum geprüft werden, wie effizient Anfragen der unterschiedlichen Klassen ausgeführt werden können. Dies hängt einerseits von dem zu Grunde liegenden logischen Schema ab. Andererseits haben auch die Merkmale der Anfragen einer Klasse einen Einfluss auf die Komplexität. Dabei sind nur diejenigen Merkmale einer Anfrage zu berücksichtigen, für die Folgendes gilt:

- Sie können direkt aus den Merkmalen der Klasse abgeleitet werden,
- sie beeinflussen die Komplexität der Anfragen und
- sie sind abhängig vom zu Grunde liegenden logischen Schematyp.

Wichtige Eigenschaften von Anfragen, die Einfluss auf die Anfragekomplexität haben, sind in erster Linie die Art und Anzahl der Operatoren, die auf die Daten angewandt werden. Die folgenden Eigenschaften müssen also berücksichtigt werden:

- die Anzahl der Gruppierungen,
- die Anzahl der Sortieroperationen,
- die Anzahl und Art der Selektionen sowie
- die Anzahl und Art der Join-Operationen.

Die Anzahl der Gruppierungen und Sortierungen hat offensichtlich einen Einfluss auf die Komplexität einer Anfrage, wobei eine Gruppierung als zusätzliche Sortieroperation interpretiert werden kann. Beide Operationen sind aber weder

abhängig von der Klasse der Anfragen noch vom zu Grunde liegenden Schema. Sie sind damit für die weiteren Betrachtungen nicht relevant.

Die Anzahl und die Art der Selektionen umfasst mehrere Aspekte. Hierzu gehört die Anzahl der Attribute, über die selektiert wird, die Frage, ob es sich um Punkt- oder Bereichsanfragen handelt, sowie die Frage, ob die Selektion Fakten oder Dimensionen betrifft. Auch diese Merkmale haben einen Einfluss auf die Komplexität einer Anfrage, da sie sich z.B. auf eine mögliche Unterstützung der Anfrageverarbeitung durch Zugriffspfade auswirken. Allerdings ist auch hier keine Abhängigkeit von der Anfrageklasse oder gar dem logischen Schema vorhanden.

Bleibt die Anzahl und die Art der Join-Operationen als einziges Merkmal. Sie kann in Abhängigkeit von der Anfrageklasse und dem logischen Schema bestimmt werden. Die Anfrageklasse bestimmt die Anzahl der Tabellen, die in einer Anfrage verwendet werden, und damit auch die Anzahl der Joins. Die Art der Join-Operationen bezieht sich auf die Frage, ob es sich um Joins zwischen einer Faktentabelle und einer anderen Tabelle handelt, oder lediglich um Joins zwischen Dimensionstabellen. Die erste dieser beiden Varianten wird im Folgenden als *F-Join* bezeichnet, während sich *D-Join* auf die zweite Variante bezieht. Diese Unterscheidung hat wiederum Einfluss auf die Komplexität der Anfrage, da das Datenvolumen in Faktentabellen in der Regel um mehrere Größenordnungen über dem Datenvolumen in Dimensionstabellen liegt. Damit sind F-Joins unter Leistungsgesichtspunkten wesentlich höher zu bewerten, als Joins innerhalb einer Dimension (D-Joins). Grundsätzlich handelt es sich allerdings bei jeder einzelnen Join-Operation um eine aufwändige Operation für ein ORDBS. Insbesondere wird die Aufgabe für einen Anfrageoptimierer umso komplexer, je größer die Anzahl der Joins in einer einzelnen Anfrage ist. Ein Grund hierfür ist die zunehmende Anzahl möglicher Join-Reihenfolgen, die bei der Anfrageverarbeitung berücksichtigt werden müssen. Dieser Einfluss der Anzahl von Joins auf die Performanz einer Anwendung wird sich auch bei den in Abschnitt 6.3 dargestellten experimentellen Ergebnissen zeigen. Insgesamt stellt die Anzahl der Join-Operationen ein gutes Maß für die Komplexität der Anfragen der einzelnen Anfrageklassen dar.

In Tabelle 4 und Tabelle 5 sind für alle Anfrageklassen die Anzahl der Join-Operationen in Abhängigkeit vom logischen Modell angegeben. Hierbei sind die folgenden Größen zu Grunde gelegt:

f: Anzahl der relevanten Faktentabellen.

d: Anzahl der relevanten Dimensionen.

d_i: Anzahl der relevanten Dimensionen für die i-te Faktentabelle.

a: Anzahl der relevanten Dimensionsattribute in der einzigen zu berücksichtigenden Dimension.

$a_{i,j}$: Anzahl der relevanten Dimensionsattribute in der Dimension j, auf die die i-te Faktentabelle verweist.

$h_{i,j,k}$: Hierarchieebene des k-ten relevanten Attributs in der j-ten Dimension, auf die die i-te Faktentabelle verweist. Die jeweils unterste Hierarchieebene wird als 0 angenommen.

Tabelle 4: Komplexität der Anfrageklassen A_{100} bis A_{01x} in Abhängigkeit vom Schematyp

Klasse	A_{100}	A_{011}	A_{01x}
Flat-Schema: F-Joins	0	0	0
Flat-Schema: D-Joins	0	0	0
Star-Schema: F-Joins	0	0	0
Star-Schema: D-Joins	0	0	0
Snowflake-Schema: F-Joins	0	0	0
Snowflake-Schema: D-Joins	0	0	a
Informix-Schema: F-Joins	0	0	0
Informix-Schema: D-Joins	0	0	a
Strategy-Schema: F-Joins	0	0	0
Strategy-Schema: D-Joins	0	0	a

Aus Tabelle 4 ist ersichtlich, dass die Zahl der notwendigen Join-Operationen bei den Klassen *A_{100}*, *A_{011}* und *A_{01x}* für praktisch alle logischen Modelle identisch ist. In diesen Klassen beziehen sich die Anfragen auf genau eine Faktentabelle bzw. auf genau eine Dimension und sonst keine weiteren Tabellen. Hier sind bei der Anfrageverarbeitung keine Joins notwendig. Einzige Ausnahme ist die Anfrageklasse *A_{01x}*. Hier wird pro relevantem Attribut ein Join zwischen zwei Dimensionstabellen notwendig.

Tabelle 5: Komplexität der Anfrageklassen A_{1x1} bis A_{xxx} in Abhängigkeit vom Schematyp

Klasse		A_{1x1}	A_{xx1}	A_{1xx}	A_{xxx}
Flat-Schema	F-Joins	0	f-1	0	f-1
	D-Joins	0	0	0	0
Star-Schema	F-Joins	d	$\sum_{i=1}^{f} d_i$	d	$\sum_{i=1}^{f} d_i$
	D-Joins	0	0	0	0
Snowflake-Schema	F-Joins	d	$\sum_{i=1}^{f} d_i$	d	$\sum_{i=1}^{f} d_i$
	D-Joins	$\sum_{j=1}^{d} h_{1,j,1}$	$\sum_{i=1}^{f}\sum_{j=1}^{d_i} h_{i,j,1}$	$\sum_{j=1}^{d}\sum_{k=1}^{a_{1,j}} h_{1,j,k}$	$\sum_{i=1}^{f}\sum_{j=1}^{d_i}\sum_{k=1}^{a_{i,j}} h_{i,j,k}$
Informix-Schema	F-Joins	d	$\sum_{i=1}^{f} d_i$	d	$\sum_{i=1}^{f} d_i$
	D-Joins	d	$\sum_{i=1}^{f} d_i$	$\sum_{j=1}^{d} a_{1,j}$	$\sum_{i=1}^{f}\sum_{j=1}^{d_i} a_{i,j}$
Strategy-Schema	F-Joins	d	$\sum_{i=1}^{f} d_i$	d	$\sum_{i=1}^{f} d_i$
	D-Joins	d	$\sum_{i=1}^{f} d_i$	$\sum_{j=1}^{d} a_{1,j}$	$\sum_{i=1}^{f}\sum_{j=1}^{d_i} a_{i,j}$

In Tabelle 5 ist zu beachten, dass teilweise eine obere Schranke für die notwendige Anzahl von Join-Operationen angegeben ist. Dies gilt für die Anzahl der D-

Joins im Snowflake-Schema und im Strategy-Schema. Die Ursache hierfür ist, dass für alle relevanten Dimensionsattribute auf der untersten Hierarchieebene ein Join weniger notwendig ist. Die verfügbare Information zu diesem Dimensionsattribut ist bereits durch einen F-Join in der Anfrage berücksichtigt.

Die Angaben in Tabelle 5 zeigen, dass im Flat-Schema nur dann Verbund-Operationen bei der Anfrageverarbeitung notwendig sind, wenn Informationen aus unterschiedlichen Faktentabellen zu berücksichtigen sind. Dagegen hängt die Zahl der Join-Operationen im Star-Schema im Wesentlichen von der Anzahl zu berücksichtigender Dimensionen ab. Die Beispielanfrage der Klasse A_{1xx} aus Abbildung 9 ist auf der Grundlage eines Star-Schemas erstellt. Sie umfasst darum für jede Dimension einen Join mit der Faktentabelle (F-Joins: d), aber keine Joins innerhalb der Dimensionen (D-Joins: 0). In allen Varianten des Snowflake-Schemas kommt die Zahl der relevanten Dimensionsattribute als Einflussfaktor hinzu. Beim klassischen Snowflake-Schema schließlich muss auch noch die jeweilige Hierarchieebene des Dimensionsattributs berücksichtigt werden, wenn die Anzahl der Joins korrekt bestimmt werden soll. Dies ist beispielsweise am Eintrag für die Anzahl der D-Joins einer Anfrage der Klasse A_{xxx} auf der Basis des Snowflake-Schemas zu erkennen. Dieser Eintrag besteht aus drei Summenzeichen. Die erste Summe berücksichtigt, die f relevanten Faktentabellen, die zweite die Anzahl der für die einzelnen Fakten relevanten Dimensionen. In jeder dieser Dimensionen sind im allgemeinen Fall mehrere Dimensionsattribute auf unterschiedlichen Hierarchiestufen von Bedeutung. Die Anzahl der hierbei notwendigen Joins innerhalb einer Dimension wird durch das letzte Summenzeichen des Eintrags berücksichtigt.

Wie der Tabelle zu entnehmen ist, ist die Komplexität der Anfrageverarbeitung im Snowflake-Schema durch die Anzahl der notwendigen Join-Operationen am größten, während im Flat-Schema die geringste Komplexität zu verzeichnen ist. Das Informix-Schema und das Strategy-Schema weisen in erster Linie dann erhebliche Performanz-Vorteile gegenüber dem Snowflake-Schema auf, wenn eine komplexe und tiefe Hierarchie berücksichtigt werden muss. Für diese beiden logischen Modelle ist die Anzahl der Joins unabhängig von der Hierarchiestufe der einzelnen Dimensionsattribute. Am nächsten beim Flat-Schema liegt das Star-Schema, da hier selbst die Anzahl relevanter Dimensionsattribute keinen Einfluss auf die notwendigen Joins hat.

Für die Bewertung der Schematypen hinsichtlich der Performanz von Anwendungen bleibt Folgendes festzuhalten: Das gewählte logische Schema hat erheblichen Einfluss auf die Anzahl der Join-Operationen, die für typische Anfragen der Bereiche OLAP und Data Mining notwendig sind. Die Anzahl der Joins ist ein wichtiger Einflussfaktor hinsichtlich der Performanz der Anwendung. Andere Faktoren, wie z.B. die Anzahl von Gruppierungs- oder Sortieroperationen, wirken sich zwar ebenfalls auf die Performanz einer Anwendung aus. Sie hängen im Gegensatz zur Anzahl der Joins aber nicht vom gewählten logischen Schema ab, sondern nur von der spezifischen Anfrage. Damit bleibt die Anzahl der Joins als einziger Performanz-Faktor, der bei der Wahl des logischen Schematyps berücksichtigt werden muss.

Von den vorgestellten Typen logischer Schemata für ein Data Warehouse sind alle diejenigen unter Performanz-Gesichtspunkten besonders vorteilhaft, die eine hohe Redundanz aufweisen und wenige Joins bei der Verarbeitung typischer Anfragen benötigen. Hierzu gehört das Flat-Schema und das Star-Schema. In den anderen Schematypen (Snowflake-Schema, Informix-Schema, Strategy-Schema) wird Redundanz durch unterschiedlich starke Normalisierung vermieden. Der Preis hierfür ist ein deutlich höherer Aufwand bei der Anfrageverarbeitung.

4.2.10 Zusammenfassende Bewertung der Schematypen

Die Analysen im Bereich des logischen Entwurfs konzentrierten sich auf eine Reihe von Schematypen im relationalen Modell. Die wichtigsten Vertreter wurden bezüglich verschiedener Kriterien bewertet. Hierzu gehört der Aufwand, der aus Modifikationen des konzeptuellen Schemas resultiert und die Redundanz, die in Schemata der verschiedenen Schematypen vorzufinden ist. Zu den für die Integration von OLAP und Data Mining spezifischeren Kriterien gehört der Grad, in dem die im COCOM-Modell modellierten Zusammenhänge in den einzelnen Schematypen repräsentiert werden können sowie der Einfluss auf die Performanz der OLAP- und Data-Mining-Anwendungen. Im Snowflake-Schema und seinen Varianten bestehen die umfassendsten Möglichkeiten, die im COCOM-Modell gegebenen Strukturierungsmöglichkeiten auf das logische Schema zu übertragen. Darüber hinaus zeichnen sich diese Schematypen durch die geringste Redundanz aus. Bei den unterschiedlichen Analyseanwendungen auf einem Data Warehouse steht allerdings in der Regel die Performanz im Vordergrund. Dies bezieht sich sowohl auf den Aufwand, der erforderlich ist, um die am konzeptuellen Schema vorgenommenen Änderungen im logischen Schema nachzuziehen, als auch auf

die zu erwartende Performanz der Anfragen, die für die Datenvorverarbeitung und die eigentliche Analyse von den einzelnen Anwendungen erstellt werden. Bezüglich dieser Kriterien legt die durchgeführte Analyse der diversen Schematypen die Verwendung eines einfacher strukturierten Schemas nahe. Insbesondere das Star-Schema bietet deutliche Vorteile bezüglich der Performanz. Trotzdem erlaubt es die grundlegendsten Modellierungsmöglichkeiten, die sowohl für OLAP als auch für Data Mining von Bedeutung sind, aus dem konzeptuellen Schema zu übernehmen. Von den betrachteten Schematypen ist also das Star-Schema für ein Data Warehouse, das sowohl OLAP als auch Data Mining unterstützen soll, am besten geeignet.

4.3 Der physische Data-Warehouse-Entwurf

Der Entwurfsprozess wird durch den physischen Entwurf abgeschlossen. Die wichtigsten Entscheidungsgegenstände dieser Phase sind:

- Die Festlegung des Partitionierungsschemas für die im logischen Entwurf festgelegten Tabellen.
- Die Auswahl der geeigneten Zugriffspfade auf der Grundlage einer repräsentativen Menge von Anfragen.
- Die Auswahl geeigneter materialisierter Sichten, d.h. Teilergebnisse, die vorab berechnet und im Data Warehouse gespeichert werden.

Teilweise werden einzelne dieser Entscheidungsgegenstände auch dem logischen Entwurf zugeordnet [GR99]. In jedem Fall handelt es sich um Entscheidungen, die in ähnlicher Weise auch im Datenbankentwurf für andere Anwendungsbereiche notwendig sind. Sie sind also nicht spezifisch für den Data-Warehouse-Entwurf. Im Folgenden soll lediglich darauf eingegangen werden, welche Alternativen sich speziell bei der Modellierung eines Data Warehouse bieten. Dabei soll besonders berücksichtigt werden, ob unterschiedliche Ansätze und Vorgehensweisen für OLAP und Data Mining verfügbar und sinnvoll sind.

Die Frage des optimalen Partitionierungsschemas wurde in der Literatur bereits ausführlich untersucht. Für den Data-Warehouse-Bereich werden in [GR99] spezielle Kostenmodelle angegeben, die den Zusammenhang zwischen der Partitionierungsentscheidung und der Wahl materialisierter Sichten berücksichtigen. Als Grundlage für die Wahl des geeigneten Partitionierungsschemas wird in erster Linie ein repräsentativer Satz von Anfragen vorausgesetzt, der die Anfragen, die

vom zu erstellenden System bearbeitet werden müssen, möglichst gut widerspiegelt. Ob diese aus einer Data-Mining- oder OLAP-Anwendung stammen hat keinen Einfluss auf die Partitionierung und muss nicht weiter berücksichtigt werden.

Auch die Wahl geeigneter Zugriffspfade basiert auf einem Satz repräsentativer Anfragen, ohne dabei zu berücksichtigen, aus welcher Art Anwendung die Anfragen stammen. Im Data-Warehouse-Bereich kommen hierbei häufig spezielle Zugriffspfade, wie z.B. Bitmap-Indexe [One87], Join-Indexe oder Projektions-Indexe [OG95] zum Einsatz.

Die Auswahl geeigneter materialisierter Sichten ist ein typischer Ansatz, um die Performanz von Anwendungen auf der Basis eines Data Warehouse zu steigern. Aus der mehrdimensionalen Sichtweise ergibt sich eine große Anzahl potenzieller materialisierter Sicht aus denen unter Berücksichtigung gegebener Beschränkungen eine optimale Teilmenge ausgewählt werden muss [HRU96]. Auch in diesem Fall spielt es keine Rolle, ob es sich bei den Anwendungen, die für ihre Anfrageverarbeitung die materialisierten Sichten verwenden, um Data-Mining- oder OLAP-Anwendungen handelt.

Der Überblick über die wichtigsten Aspekte des physischen Entwurfs zeigt, dass in dieser Phase des Entwurfsprozesses keine unterschiedlichen Ansätze für Data-Mining- und OLAP-Anwendungen vorhanden sind.

4.4 Zusammenfassung

Die vorangegangene Analyse unterschiedlicher Modellierungsansätze für ein Data Warehouse hat gezeigt, dass einige dieser Ansätze für eine Integration von OLAP und Data Mining geeignet sind. In der Regel wurden die beschriebenen Modelle und Vorgehensweisen für den Data-Warehouse-Entwurf mit Blick auf Analysen aus dem Bereich OLAP entwickelt. Nicht alle eignen sich gleichermaßen gut, wenn es darum geht, auf demselben Data Warehouse zusätzlich Data-Mining-Analysen durchzuführen.

Für den konzeptuellen Entwurf werden in der wissenschaftlichen Literatur eine Reihe von Modellen beschrieben. Diese unterscheiden sich im jeweiligen Spektrum der Modellierungsmöglichkeiten. Im Rahmen dieser Arbeit wurden die für Data Mining und OLAP wichtigen Modellierungsmöglichkeiten identifiziert.

Auf dieser Grundlage wurde das COCOM-Modell für ein konzeptuelles Schema entwickelt, das alle diese Modellierungsmöglichkeiten bietet.

Die Analysen im Bereich des logischen Entwurfs konzentrierten sich ausschließlich auf eine Reihe von Schematypen im relationalen Modell. Die wichtigsten Vertreter, das Flat-Schema, das Star-Schema, das Snowflake-Schema und einige Varianten davon wurden hinsichtlich ihrer Ausdrucksmächtigkeit, des Änderungsaufwands, der aus Modifikationen des konzeptuellen Schemas resultiert sowie hinsichtlich ihres Einflusses auf die Performanz der Anwendungen verglichen. Keiner der betrachteten Schematypen hat sich in Bezug auf diese Kriterien als gleichermaßen geeignet herausgestellt. Insgesamt hat sich allerdings gezeigt, dass das Star-Schema in der Regel am besten geeignet ist, sowohl Data-Mining- als auch OLAP-Anwendungen zu unterstützen. Bei diesem Schematyp sind bei den typischen Änderungsoperationen auf dem konzeptuellen Schema weniger Änderungen im logischen Schema notwendig als bei allen anderen Schematypen. Hinsichtlich der Performanz der Anwendungen wurde die Zahl der für typische Anfragen aus den Bereich OLAP und Data Mining notwendigen Joins als Maß herangezogen. Auch hier hat ein Star-Schema Vorteile gegenüber den anderen Schematypen.

Beim physischen Entwurf gibt es für den Data-Warehouse-Bereich spezifische Entscheidungsgegenstände und Vorgehensweisen. Diese hängen allerdings nicht von der Art der Analyseanwendung ab und spielen deshalb für die Integration von OLAP und Data Mining keine weitere Rolle.

5 Systemarchitektur für die Integration von Data Mining und OLAP

In diesem Kapitel werden die Systemarchitekturen existierender Data-Mining- und OLAP-Werkzeuge untersucht und bewertet. Dies erfolgt zunächst getrennt für die beiden Bereiche. Anschließend wird eine Systemarchitektur vorgestellt, die die integrierte Anwendung von Data Mining und OLAP auf einem Data Warehouse unterstützt.

5.1 Merkmale von Systemarchitekturen

Für den Begriff der *Systemarchitektur* eines Softwaresystems gibt es unterschiedliche Definitionen [GP95] [BCK98] [BRJ99]. Gemeinsam ist diesen Definitionen, dass eine Architektur die Komponenten eines Softwaresystems, Eigenschaften dieser Komponenten sowie deren Beziehungen zueinander beschreibt. Zu den Eigenschaften einer Komponente gehört in erster Linie die Funktionalität, die ihr im Rahmen des gesamten Softwaresystems zugeordnet ist sowie jegliche Art von Einschränkungen für den Einsatz dieser Komponente.

Eine grobe Beschreibung einer Architektur umfasst neben den Komponenten aber auch die Anzahl der *Schichten*, auf die diese verteilt werden. Zu einer Schicht gehören im allgemeinen Fall mehrere logisch zusammengehörende Komponenten, die untereinander, aber auch schichtenübergreifend kommunizieren. Wichtig ist der Zusammenhang der Komponenten einer Schicht, sowohl was deren Funktionalität als auch deren Kommunikationsbeziehung betrifft. Dieser ist so groß, dass bei der Realisierung eines Softwaresystems in der Regel alle Komponenten einer Schicht einem Rechner zugeordnet werden. Bei dieser Vorgehensweise ergibt sich für eine Einschichtenarchitektur eine rein lokale Verarbeitung, während in allen Varianten mit mehreren Schichten eine verteilte Verarbeitung vorliegt. Grundsätzlich ist es aber auch möglich, mehrere Schichten auf einem Rechner oder die Komponenten einer Einschichtarchitektur verteilt zu realisieren. Die Verteilung auf mehrere Rechner spielt auch unter Leistungsgesichtspunkten eine Rolle, da diese häufig eine Erhöhung der Parallelität innerhalb einer Schicht notwendig machen. Dies kann durch die Replikation der Schicht auf mehreren Rechnern erreicht werden.

In einem ersten Schritt kann eine Architektur somit durch die Anzahl der Schichten, die sie umfasst, sowie durch ihre wichtigsten Komponenten charakterisiert

werden. Zusätzlich ist aber auch kennzeichnend, in welche Teilkomponenten ein System unterteilt wird und wie diese auf die einzelnen Schichten verteilt sind. Diese Verteilung hat einen erheblichen Einfluss auf die Leistung des Gesamtsystems.

Bei Data-Mining- und OLAP-Werkzeugen handelt es sich um typische datenbankorientierte Client/Server-Anwendungen. Diese umfassen drei Hauptkomponenten: die *Präsentation*, die *Anwendungsfunktionalität* sowie die *Datenhaltung*. Letztere beinhaltet die permanente Speicherung der Daten sowie den Zugriff auf die Daten. Hierzu gehören auch temporäre Ergebnisse, die durch die Anwendung erstellt werden. Die Anwendungsfunktionalität schließt alle durch das Werkzeug gesteuerten Schritte der Datenvorverarbeitung sowie die eigentlichen Analysealgorithmen ein. Zur Präsentation gehört schließlich die meist grafische Aufbereitung der Analyseergebnisse. Die Hauptkomponenten werden in der Regel in eine Reihe von Teilkomponenten unterteilt. Die Systemarchitektur ordnet jede dieser Teilkomponenten einer bestimmten Schicht zu. Die typischen Architekturvarianten sind in Abbildung 13 dargestellt. Sie weisen zwischen einer und drei Schichten auf. Durch die Pfeile sind weitere Variationsmöglichkeiten angedeutet. Sie machen deutlich, dass die Trennlinien zwischen den einzelnen Schichten einer Architektur nicht notwendigerweise der Trennung zwischen den Hauptkomponenten entsprechen müssen. Die Architekturvarianten entstehen also dadurch, dass Teilkomponenten einer Hauptkomponente unterschiedlichen Schichten der Architektur zugeordnet werden. In der dargestellten Variante der Zweischichtenarchitektur ist die Anwendungsfunktionalität beispielsweise weitgehend der oberen Schicht, das heißt dem Client zugeordnet. In diesem Zusammenhang wird

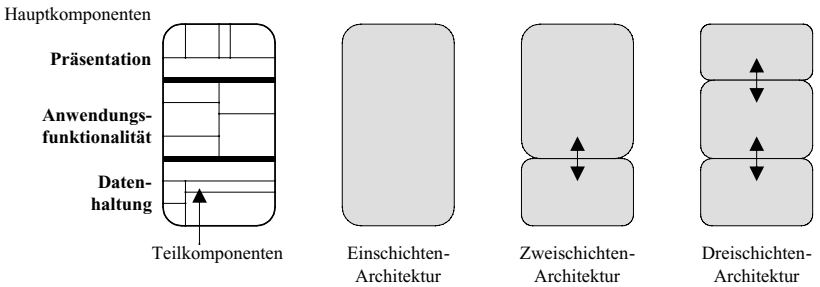

Abbildung 13: Typische Architekturvarianten

häufig von einem *Fat Client* gesprochen. Wird ein wesentlicher Teil der Logik dagegen in die untere Schicht integriert, so handelt es sich um eine Variante, die als *Thin Client* bezeichnet werden kann.

Ein weiteres, allgemeines und wichtiges Merkmal einer Systemarchitektur sind schließlich die Beziehungen der einzelnen Komponenten zueinander. Hier geht es um die Frage, welche Informationen zwischen den verschiedenen Systemkomponenten ausgetauscht werden müssen und welche Daten in welchen Schichten der Architektur verwaltet werden. Bei den auszutauschenden Informationen kann es sich um Basisdaten aus dem Data Warehouse, um temporäre Ergebisse, die bei der Anfrageverarbeitung anfallen, sowie um Funktionsaufrufe und die dafür notwendigen Parameter handeln. Die Beschreibung dieser Beziehungen zwischen Systemkomponenten wird in dieser Arbeit als das *Ausführungsmodell* bezeichnet.

Im Rahmen dieser Arbeit werden bei der Untersuchung von Systemarchitekturen die folgenden *Merkmale* berücksichtigt:

- die Komponenten und deren Eigenschaften,
- die Anzahl der Schichten,
- die Zuordnung der Komponenten zu den Schichten sowie
- das Ausführungsmodell.

5.2 Qualitätsattribute von Softwaresystemen

Mit den beschriebenen Merkmalen werden die wesentlichen Aspekte einer Systemarchitektur erfasst. Diese lassen aber noch keine Aussage bezüglich der Qualität eines Softwaresystems zu, das auf einer bestimmten Architektur beruht. Nach [BCK98] zerfallen Attribute von Softwaresystemen in zwei große Klassen: Attribute, die zur Laufzeit bewertet werden können, und solche, die zur Laufzeit des Systems keinen Einfluss haben. Zur ersten Klasse gehören Qualitätsattribute wie Performanz, Sicherheit, Verfügbarkeit, Funktionalität und Verwendbarkeit. Zur zweiten Klasse gehören Wartbarkeit, Portierbarkeit, Wiederverwendbarkeit, Integrierbarkeit sowie Testbarkeit. Abgesehen von der Funktionalität werden nach [BCK98] alle diese Qualitätsattribute wesentlich von der gewählten Systemarchitektur beeinflusst. Allerdings haben im Anwendungsumfeld dieser Arbeit nicht alle dieselbe Bedeutung. Hier soll darum zunächst erläutert werden, welche Qualitätsattribute im Rahmen dieser Arbeit nicht weiter berücksichtigt werden.

Anschließend wird dann beschrieben, welche Teilaspekte ausgewählter Attribute detailliert betrachtet werden.

Die Attribute *Sicherheit* und *Verfügbarkeit* spielen im betrachteten Anwendungsumfeld eine weniger bedeutende Rolle und werden zum größten Teil auf die entsprechenden Attribute des zu Grunde liegenden Datenbanksystems abgebildet. Analyseanwendungen greifen überwiegend lesend auf die Daten im Data Warehouse zu. Anwendungsseitig ist also nicht vorgesehen, Basisdaten zu ergänzen oder zu verändern. Welche Daten die einzelnen Benutzer lesen dürfen, kann über den Zugriffsschutz des Datenbanksystems kontrolliert werden, so dass hier im Rahmen der Analysewerkzeuge keine zusätzlichen Vorkehrungen notwendig sind. Die Verfügbarkeit von Softwaresystemen spielt immer dann eine besonders große Rolle, wenn durch das System das Kerngeschäft eines Unternehmens kontinuierlich unterstützt werden muss. Die hier betrachteten Analyseanwendungen gehören allerdings in der Regel nicht dazu. Mit ihrer Hilfe werden lediglich die im Kerngeschäft eines Unternehmens anfallenden Daten analysiert und aufbereitet. Daraus ergibt sich, dass die Verfügbarkeit der Systeme keine zentrale Rolle bei der Betrachtung der Systemarchitekturen spielen muss.

Die *Verwendbarkeit* weist einerseits Bezüge zur Performanz auf. Andererseits wird sie wesentlich durch die Gestaltung der Benutzungsschnittstelle und der Fehlerbehandlung beeinflusst. Beide Bereiche werden nicht primär durch die Systemarchitektur bestimmt, so dass die Verwendbarkeit in dieser Arbeit nicht als Attribut zur Charakterisierung von Systemarchitekturen herangezogen wird.

Bei den Qualitätsattributen, die zur Laufzeit eines Systems keinen Einfluss haben, werden die Wiederverwendbarkeit, die Integrierbarkeit sowie die Testbarkeit hier nicht berücksichtigt. Die *Wiederverwendbarkeit* wird nicht weiter betrachtet, da sie bei der Detaillierungsstufe, auf der Systemarchitekturen in diesem Kapitel betrachtet werden, nicht hinreichend genau bewertet werden kann. Hinsichtlich der *Integrierbarkeit* und *Testbarkeit* können bei den im Folgenden analysierten Architekturen keine relevanten Unterschiede identifiziert werden, so dass auch zu diesen Qualitätsattributen keine weiteren Aussagen gemacht werden können.

Bei den betrachteten Analyseanwendungen wird ein großes Augenmerk auf die *Performanz* der Systeme gelegt. Deshalb werden Teilaspekte hiervon in den folgenden drei separaten Attributen betrachtet:

- die Skalierbarkeit,
- die Lastbalancierung sowie
- die Nutzung von Datenbankfunktionalität.

Von den nicht laufzeitrelevanten Attributen werden nur die folgenden näher analysiert:

- die Wartbarkeit und
- die Portierbarkeit.

Ein weiterer Aspekt, der in den folgenden Abschnitten für die einzelnen Architekturen berücksichtigt wird, ist die *Administration des Softwaresystems*, soweit diese durch Merkmale der Architektur beeinflusst wird. Dieses Attribut findet sich nicht direkt in den Qualitätsattributen nach [BCK98] wieder, deckt dort aber Teilaspekte der Bereiche Funktionalität und Verwendbarkeit ab. Die als relevant identifizierten Qualitätsattribute werden in den folgenden Abschnitten detailliert erläutert. Bei diesen Erläuterungen wird berücksichtigt, dass diese Attribute zur Bewertung von Architekturen für Analysewerkzeuge herangezogen werden sollen.

Die *Skalierbarkeit* beschreibt einen wichtigen Aspekt des Leistungsverhaltens eines Werkzeuges. Hier geht es um die Frage, inwieweit das Verhalten, insbesondere das Antwortzeitverhalten, des Systems beeinflusst wird, wenn die Aufgabenstellung umfangreicher wird. Die Erhöhung des Aufgabenumfangs weist zwei getrennte Aspekte auf. Einerseits kann die Datenmenge, die mit Hilfe eines Werkzeugs analysiert werden soll, anwachsen. Andererseits kann sich die Anzahl der Benutzer, die einen gegebenen Datenbestand analysieren wollen, erhöhen. Ein Werkzeug weist dann eine gute (lineare) Skalierbarkeit auf, wenn eine gleichbleibende Antwortzeit dadurch erzielt werden kann, dass proportional zur wachsenden Aufgabenstellung zusätzliche Hardware eingesetzt wird. Diese zusätzliche Hardware ermöglicht die hierzu notwendige Parallelverarbeitung. Die Skalierbarkeit ist gerade im Data-Warehouse-Umfeld eine wichtige Eigenschaft der Analysewerkzeuge. Hier wachsen die Anforderungen an die Werkzeuge einerseits, da der im Data Warehouse zur Verfügung stehende Datenbestand in der Regel stetig wächst. Andererseits steigt in einem typischen Data-Warehouse-Projekt die Zahl der Nutzer über einen längeren Zeitraum stetig an, so dass auch aus diesem Grund von Beginn an auf die Skalierbarkeit einer gewählten Lösung geachtet werden muss.

Das Ziel der *Lastbalancierung* in einem System ist es, die zu bearbeitenden Aufgaben so auf die zur Verfügung stehenden Ressourcen zu verteilen, dass eine möglichst optimale Ausnutzung dieser Ressourcen gewährleistet ist. In den beschriebenen Architekturen ist dies sowohl horizontal als auch vertikal möglich. *Horizontale Lastbalancierung* kann innerhalb einer Schicht der Architektur erfolgen, wenn dieser Schicht in mehreren Rechnersystemen repliziert vorliegt. Anstehende Aufgaben können dann einem oder mehreren dieser Rechnersysteme zugeordnet werden. Durch ein Verfahren der Lastbalancierung soll sichergestellt werden, dass diese Zuordnung zu einer möglichst effizienten Ressourcennutzung führt. Darüber hinaus ist auch eine *vertikale Lastbalancierung* denkbar. Dies ist immer dann möglich, wenn dieselbe Aufgabe in unterschiedlichen Schichten ausgeführt werden kann. Dann ist es die Aufgabe der Lastbalancierung, unter Berücksichtigung der Systemlast zu entscheiden, welcher Schicht die effizientere Ausführung ermöglicht. Eine wichtige Unterscheidung muss bei den verschiedenen Verfahren zur Lastbalancierung ebenfalls berücksichtigt werden. Bei der *statischen Lastbalancierung* erfolgt die Zuordnung einer Aufgabe zu den verfügbaren Ressourcen vor der Bearbeitung der Aufgabe. Bei der *dynamischen Lastbalancierung* dagegen wird diese Zuordnung erst unmittelbar bei der Bearbeitung der Aufgabe vorgenommen [CK88]. Welcher Rechner der Schicht, die die Anwendungsfunktionalität umfasst, für eine konkrete Anfrage eines Clients zuständig ist, wird im ersten Fall also statisch festgelegt, während sich die Zuordnung bei der dynamischen Lastbalancierung bei jedem erneuten Aufruf ändern kann. Im Rahmen dieser Arbeit werden keine Verfahren zur Lastbalancierung betrachtet. Bei der Betrachtung der Systemarchitekturen geht es lediglich um die Frage, inwieweit diese eine Lastbalancierung ermöglichen oder inwieweit sie diese erschweren oder gar verhindern.

Alle betrachteten Architekturen umfassen das Data Warehouse, in dem die zu analysierenden Daten bereitgehalten werden. In Rahmen dieser Arbeit wird davon ausgegangen, dass dieses Data Warehouse auf einem relationalen Datenbanksystem beruht. Ein Teil der Funktionalität zur Analyse der Daten kann unter Umständen direkt im Datenbanksystem ausgeführt werden. Dies ist immer dann möglich, wenn sich zumindest ein Teil der Analyse direkt in SQL ausdrücken lässt. Mit der Aufnahme von Operatoren in die Anfragesprache SQL, die vor allem für die Bereiche OLAP und Data Mining wichtig sind, wird dies zunehmend möglich [GB+96] [CD+99]. Eine wichtige Eigenschaft eines Werkzeugs ist darum, inwieweit es die *Funktionalität des zu Grunde liegenden Datenbanksystems* nutzt. Auf der Architektur-Ebene soll im Rahmen dieser Arbeit darum

untersucht werden, inwieweit die einzelnen Architekturen eine möglichst weitgehende Nutzung der Funktionalität des zu Grunde liegenden Datenbanksystems unterstützen.

Im Data-Warehouse-Umfeld werden in der Regel viele Analyseanwendungen genutzt, um Informationen zu gewinnen. Bei der Administration des gesamten Data Warehouse sollte der Ressourcenverbrauch der einzelnen Anwendungen kontrolliert werden können. Ob und in welchem Maße die Möglichkeit hierzu besteht, wird wesentlich durch die Architektur des Systems beeinflusst. Deshalb wird im Folgenden bei den verschiedenen Architekturen auch untersucht, inwiefern sie die *Administrationsmöglichkeiten* beeinflussen.

Für jede Systemarchitektur soll zusätzlich betrachtet werden, inwieweit sie die *Wartbarkeit* und die *Portierbarkeit* des Data-Mining-Werkzeugs unterstützt. Die Wartbarkeit eines Systems sagt etwas darüber aus, welcher Aufwand mit dem Hinzufügen, dem Löschen oder dem Ändern von Systemfunktionalität verbunden ist. Die beiden möglichen Extremfälle sind hierbei:

- Die veränderte Funktionalität kann durch die lokale Modifikation in genau einer Systemkomponente bereitgestellt werden.
- Um die neue Funktionalität bereitzustellen sind Änderungen in allen Komponenten des Systems notwendig.

Im ersten Fall kann von einer guten Wartbarkeit des Systems gesprochen werden, während diese im zweiten Fall nicht gegeben ist. Zur Beurteilung der Wartbarkeit der in den folgenden Abschnitten vorgestellten Architekturen wird jeweils betrachtet, welches die typischen Systemänderungen für die auf den einzelnen Architekturen aufbauenden Anwendungen sind und welche Komponenten der Architektur jeweils davon betroffen sind.

Die *Portierbarkeit* stellt einen Spezialfall der Wartbarkeit dar und wird im Rahmen dieser Arbeit gesondert betrachtet. Hier steht der Aufwand im Zentrum, der notwendig ist, um einzelne Komponenten eines Systems bzw. das gesamte System auf einer anderen Hardware- und/oder Softwareplattform bereitzustellen. Auch hier wird für die einzelnen Architekturen untersucht, welchen Einfluss diese auf die Portierbarkeit eines Systems haben.

5.3 Analyse der Architektur von Data-Mining-Werkzeugen

In diesem Abschnitt wird die Architektur existierender Data-Mining-Werkzeuge und Forschungsprototypen näher untersucht. Die Betrachtungen beziehen sich nur auf Werkzeuge, die sich für diese Arbeit als relevant erwiesen haben. Als relevant werden hierbei in erster Linie Werkzeuge eingestuft, die zumindest

- eine breite Palette an Data-Mining-Methoden anbieten oder
- eine umfangreiche Unterstützung für die Datenaufbereitung aufweisen oder
- deren Systemarchitektur deutliche Unterschiede zur Architektur der Mehrzahl der Systeme aufweist.

Nach der entsprechenden Bewertung der Werkzeuge wurden die folgenden ausgewählt und genauer untersucht:

- Knowledge Studio von ANGOSS [AN00],
- Intelligent Miner for Data von IBM [Ibm99],
- Oracle Data Mining Suite (Oracle Darwin) [TBD97] [Thi98],
- Clementine SPSS 10.0 von SPSS [SP99a] [SP99b] sowie
- XpertRule Miner von Attar bzw. der Prototyp des Projekts CRITIKAL [RS99].

5.3.1 Allgemeine Data-Mining-Architektur

Die Systemarchitekturen der untersuchten Werkzeuge weisen erhebliche Gemeinsamkeiten auf. Sie basieren in der Regel auf einer Dreischichtenarchitektur. In Abbildung 14 sind die wesentlichen Komponenten, die in den verschiedenen Werkzeugen typischerweise vorhanden sind, dargestellt. Diese Architektur wird im Rahmen dieser Arbeit als die *allgemeine Data-Mining-Architektur* bezeichnet. Auf diese Architektur lassen sich alle betrachteten Data-Mining-Werkzeuge einfach abbilden.

Die erste Schicht wird als *Client* bezeichnet und umfasst alle Komponenten, die auf den Rechnern ausgeführt werden, an denen Benutzer das Data-Mining-Werkzeug nutzen. Diese Schicht enthält die Benutzungsschnittstelle, den Modellgenerator, eine eigene Datenhaltung sowie die Komponenten zur Visualisierung und zum Export der Data-Mining-Ergebnisse.

Mit *Middle-Tier* wird die zweite Schicht der Architektur bezeichnet. Sie umfasst neben einer Komponente zur Datenaufbereitung ebenfalls einen Modellgenerator sowie eine eigene Datenhaltung und die Datenzugriffsschnittstelle.

Die dritte Schicht wird schließlich vom *Data Warehouse* gebildet, das die zu analysierenden Daten enthält. In vielen Werkzeugen können hier aber auch die Data-Mining-Ergebnisse abgelegt werden. Deshalb ist in der Architektur in Abbildung 14 eine separate Datenhaltungskomponente für Data-Mining-Ergebnisse Bestandteil der Data-Warehouse-Schicht.

Die einzelnen Komponenten der Architektur werden im Folgenden genauer beschrieben.

5.3.1.1 Komponenten

Über die *Benutzungsschnittstelle* haben die Benutzer die Möglichkeit, wesentliche Teile des in Abschnitt 2.3 beschriebenen KDD-Prozesses zu steuern. Dies

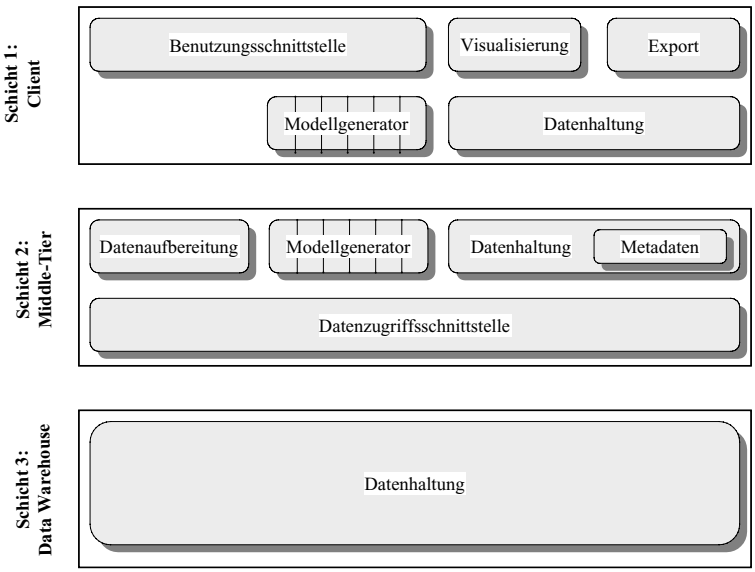

Abbildung 14: Komponenten der allgemeinen Data-Mining-Architektur

beginnt mit der Auswahl der relevanten Daten aus dem Data Warehouse und der Angabe, wohin Analyseergebnisse geschrieben werden sollen. Diese einzelnen Spezifikationen werden im Folgenden als *Datenbeschreibungsobjekte* bezeichnet. Darauf aufbauend können die notwendigen Schritte der Datenaufbereitung spezifiziert werden. Einzelne Aufbereitungsschritte werden als *Datenaufbereitungsaktion* bezeichnet, während hier unter einer *Datenaufbereitungssequenz* eine Abfolge mehrerer Datenaufbereitungsaktionen verstanden wird.

Zusätzlich wird über die Benutzungsschnittstelle angegeben, welche Data-Mining-Methoden mit welchen Parametern auf welchen Daten angewandt und wo die jeweiligen Ergebnisse abgelegt werden sollen. Die Beschreibung der Anwendung einer einzelnen Data-Mining-Methode wird hier als *Data-Mining-Aktion* bezeichnet. Analog zu den Begriffen für die Datenaufbereitung, wird *Data-Mining-Sequenz* als Bezeichnung für die Aneinanderreihung mehrerer Aktionen verwendet.

Neben der Spezifikation der Objekte, Aktionen und Sequenzen ermöglicht es die Benutzungsschnittstelle dem Benutzer auch, die Aktionen und Sequenzen zu starten sowie die Komponenten zur Visualisierung der Ergebnisse zu aktivieren.

Der Client weist ebenso wie die Middle-Tier einen *Modellgenerator* auf. Diese Komponente ist für die Ausführung der Data-Mining-Aktionen und Data-Mining-Sequenzen zuständig. Als Ergebnis liefert sie jeweils ein Data-Mining-Modell. In vielen Werkzeugen ist der Modellgenerator ausschließlich in der Middle-Tier angesiedelt [Ibm99]. Der Client hat dann in erster Linie die Aufgabe, die Data-Mining-Ergebnisse zu visualisieren. In manchen Architekturen ist es aber auch möglich, einen Teil der Data-Mining-Algorithmen direkt auf dem Client auszuführen [RS99]. Dies ist immer dann von Vorteil, wenn auf der Basis der Zwischenergebnisse eines Algorithmus mehrere Data-Mining-Modelle erstellt werden können. Liegen die Zwischenergebnisse in der Datenhaltung des Clients vor, so kann dieser ohne weitere Kommunikation mit der Middle-Tier die unterschiedlichen Data-Mining-Modelle erstellen und den Benutzern präsentieren. In der Regel ist diese Aufteilung allerdings nur gewinnbringend, wenn die berechnungsintensiven Teile der Data-Mining-Algorithmen auf der leistungsfähigeren Middle-Tier ausgeführt werden.

Die *Visualisierungskomponente* des Clients hat die Aufgabe, die einzelnen Data-Mining-Modelle grafisch darzustellen. Dies muss individuell für jedes einzelne

Data-Mining-Verfahren erfolgen. Außerdem können für ein spezifisches Data-Mining-Modell in einem Data-Mining-Werkzeug auch unterschiedliche Visualisierungsvarianten angeboten werden [FGW02].

Parallel zur Visualisierung bieten die Werkzeuge in der Regel einen weiteren Weg an, um auf erstellte Data-Mining-Modelle zuzugreifen. Die *Export*-Komponente bietet die Möglichkeit, komplette Modelle in einem speziellen Format auszulesen, um sie schließlich an anderer Stelle wiederzuverwenden. Als Format wird hierzu häufig die *Predictive Model Markup Language* verwendet [Dat00]. Aber auch der Export als SQL-Anweisungen oder in proprietären Formaten ist vielfach verfügbar. Eine Möglichkeit, exportierte Data-Mining-Modelle weiterzuverwenden, wird zunehmend durch Datenbanksysteme angeboten. So kann DB2 beispielsweise komplette Modelle importieren und auf Daten anwenden [Ibm01a].

Die *Datenaufbereitung* ist eine Komponente, die praktisch immer auf der Middle-Tier angesiedelt ist. Ihre Aufgabe ist die Ausführung der Datenaufbereitungsaktionen und Datenaufbereitungssequenzen. Sie bietet hierbei unterschiedliche Möglichkeiten zur Vorbereitung der Daten, wie z.B. das Filtern und Aggregieren, das Löschen oder Ersetzen fehlender oder ungültiger Werte sowie weitere statistische Funktionen. Zur Realisierung eines Schrittes der Datenaufbereitung stehen zwei Ansätze zur Verfügung. Einerseits können die Daten, die aufbereitet werden sollen, in die Middle-Tier geladen werden. Dann erfolgt dort die notwendige Transformation der Daten. Andererseits kann zumindest ein Teil der gewünschten Funktionalität zur Datenaufbereitung auch direkt in SQL ausgedrückt werden. Dann erstellt die Komponente auf der Middle-Tier nur die notwendigen SQL-Anweisungen, die vom Datenbanksystem der Data-Warehouse-Schicht ausgeführt werden. Die transformierten Daten werden dann in der Regel direkt im Data Warehouse bereitgestellt.

Die Middle-Tier enthält darüber hinaus noch die *Datenzugriffsschnittstelle*, die den Zugriff auf unterschiedliche darunterliegende Datenbanksysteme ermöglicht. Hierbei kann es sich um eine ODBC-Schnittstelle oder andere, proprietäre Schnittstellen handeln.

Eine *Datenhaltungskomponente* ist als einzige Komponente in allen drei Schichten vorhanden. Sie hat aber in den einzelnen Schichten unterschiedliche Aufgaben und ist auch nicht in allen Werkzeugen in allen Schichten in der gleichen

Ausprägung zu finden. Die umfangreichste Datenhaltung ist in der Data-Warehouse-Schicht angesiedelt. Dort werden von einzelnen Werkzeugen neben den Basisdaten auch Zwischenergebnisse der einzelnen Datenaufbereitungsschritte abgelegt. Darüber hinaus besteht in der Regel die Möglichkeit, die erstellten Data-Mining-Modelle im Data Warehouse zu materialisieren.

Ganz unterschiedliche Aufgaben kommen der *Datenhaltung* auf der Middle-Tier zu. In der Regel umfasst diese die Metadaten für das gesamte Data-Mining-Werkzeug. In diesen werden alle Informationen über die vom Benutzer spezifizierten Datenbeschreibungsobjekte, Datenaufbereitungsaktionen und -sequenzen sowie über die Data-Mining-Aktionen und -Sequenzen abgelegt. Darüber hinaus ist die Datenhaltung auf der Middle-Tier aber häufig auch notwendig, um Basisdaten, die aus dem Data Warehouse ausgelesen werden, als Kopie abzulegen. Dies ist immer dann der Fall, wenn die realisierten Data-Mining-Algorithmen die zu verarbeitenden Daten lokal auf der Middle-Tier voraussetzen. Eine weitere Funktion, die der Datenhaltung auf der Middle-Tier zukommt, ist das Speichern der erstellten Data-Mining-Modelle.

Es bleibt als letzte der Komponenten die *Datenhaltung* auf dem Client. Sie ist immer dann notwendig, wenn in der Middle-Tier lediglich Zwischenergebnisse eines Data-Mining-Algorithmus erstellt und diese dann an den Client weitergegeben werden. Dann muss die Datenhaltung des Client sowohl diese Zwischenergebnisse als auch die daraus erstellten Modelle aufnehmen, bevor die Ergebnisse von der Visualisierungskomponente verwendet werden können.

Auf der beschriebenen allgemeinen Dreischichtenarchitektur basiert eine Reihe von Data-Mining-Werkzeugen. Durch das Zusammenfassen der Komponenten von zwei oder drei Schichten lassen sich diese in der Regel auch in einer Ein- oder Zweischichtenarchitektur betreiben. Auf Grund der Aufteilung in drei Schichten zeichnen sich die Systeme jedoch vor allem durch eine größere Flexibilität und größere Leistungsfähigkeit aus. Dies wird in der Diskussion der Eigenschaften der beschriebenen Architektur in Abschnitt 5.3.2 deutlich.

Wie zu Beginn dieses Kapitels erläutert, kann eine Architektur nicht nur durch die Schichtenanzahl und die Zuordnung der Funktionalität zu den einzelnen Schichten beschrieben werden. Darum schließt sich hier eine Beschreibung des Ausführungsmodells für die betrachteten Data-Mining-Werkzeuge an.

5.3.1.2 Ausführungsmodell

Das Ausführungsmodell beschreibt, welche Kommunikationsbeziehungen zwischen den einzelnen Komponenten der Architektur bestehen und welche Daten diese austauschen, um die vom Benutzer geforderte Funktionalität bereitzustellen. Für die im vorangegangenen Abschnitt vorgestellte Architektur ist das Ausführungsmodell in Abbildung 15 dargestellt. Im Folgenden werden die Kommunikationsbeziehungen zwischen jeweils zwei Komponenten beschrieben. Unterschieden wird hierbei zwischen reinen Aufrufbeziehungen und Beziehungen, deren primäres Ziel der Austausch von Daten ist. Bei Aufrufbeziehungen initiiert eine Komponente die Ausführung einer Funktion/Prozedur/Methode einer anderen Komponente. Beim Datenaustausch stellt sich die Frage, welche Daten ausgetauscht werden. In Abbildung 15 sind die Beziehungen entsprechend mit einer Kurzbeschreibung der Daten gekennzeichnet. Die Reihenfolge, in der die einzelnen Kommunikationsbeziehungen beschrieben sind, orientiert sich am KDD-Prozess. Das heißt, es werden zuerst die Beziehungen im Zusammenhang mit der Vorverarbeitung der Daten und am Ende Beziehungen, die für die Visualisierung von Data-Mining-Modellen relevant sind, erläutert.

(1) Kommunikation zwischen Benutzungsschnittstelle und Datenhaltung der Middle-Tier: Die über die Benutzungsschnittstelle erfassten Datenbeschreibungsobjekte, Datenaufbereitungsaktionen und -sequenzen sowie die Data-Mining-Aktionen und -Sequenzen werden durch die Datenhaltung der Middle-Tier als Metadaten gespeichert. Abhängig davon, in welchem Format die Metadaten auf der Middle-Tier gehalten werden, erfolgt der Zugriff darauf über eine SQL-Schnittstelle oder über eine proprietäre Schnittstelle.

(2) Kommunikation zwischen Benutzungsschnittstelle und Datenaufbereitung: Die Benutzungsschnittstelle kommuniziert mit der Datenaufbereitungskomponente, um Datenaufbereitungsaktionen und Datenaufbereitungssequenzen anzustoßen, die vorab spezifiziert und in den Metadaten abgelegt wurden.

(3) Kommunikation zwischen Datenaufbereitung und Data Warehouse: Die Komponente zur Datenaufbereitung führt die spezifizierten Schritte zur Vorverarbeitung der Daten durch und greift zu diesem Zweck über die Datenzugriffsschnittstelle auf die Daten im Data Warehouse zu. Es besteht hier einerseits die Möglichkeit, dass die zu verarbeitenden Daten in die Middle-

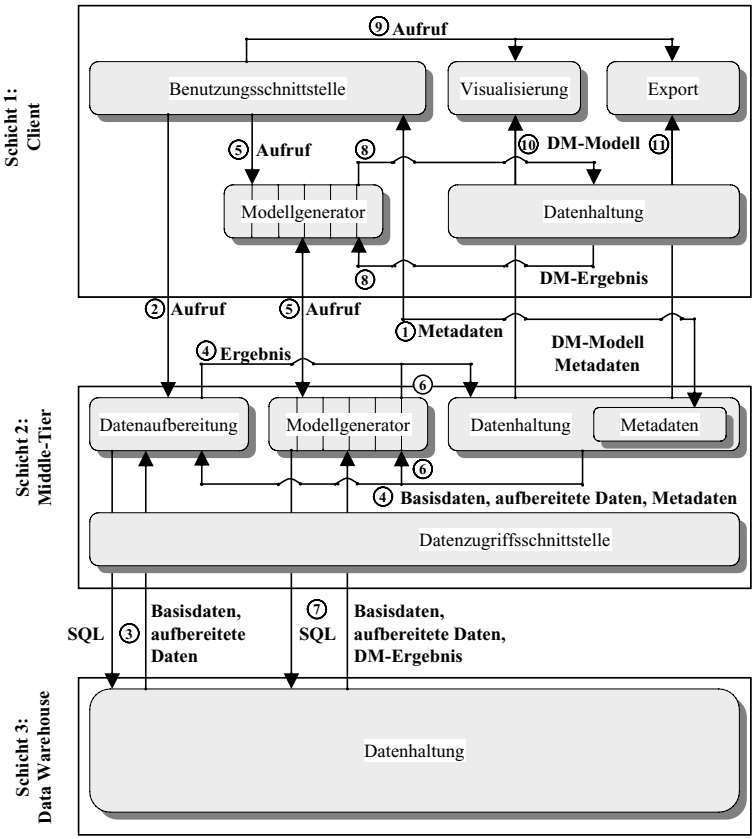

Abbildung 15: Ausführungsmodell der allgemeinen Data-Mining-Architektur

Tier geladen werden. Dann erfolgt dort die eigentliche Datenaufbereitung. Andererseits kann die Datenaufbereitungskomponente auch einen Strom von SQL-Anweisungen an das Data Warehouse schicken, der dafür sorgt, dass die aufbereiteten Daten im Data Warehouse bereitgestellt werden. Der SQL-Strom kann dann unter anderem Anweisungen zum Erstellen von Sichten und temporären Tabellen sowie die Anweisungen zum Füllen dieser temporären Tabellen enthalten.

(4) Kommunikation zwischen Datenaufbereitung und Datenhaltung der Middle-Tier: Die Datenaufbereitungskomponente muss mit der Datenhaltung der Middle-Tier kommunizieren, um die Metadaten zu den Basisdaten, den Datenaufbereitungsaktionen und den Datenaufbereitungssequenzen zu lesen. Sofern die Datenaufbereitung direkt auf der Middle-Tier erfolgt, liest die Komponente zur Datenaufbereitung die relevanten Daten sowie bereits aufbereitete Daten aus der Datenhaltung der Middle-Tier. Dort werden auch die vom Data Warehouse gelesenen Basisdaten und Zwischenergebnisse der Datenaufbereitung abgelegt.

(5) Kommunikation zwischen Benutzungsschnittstelle und Modellgenerator: Die Benutzungsschnittstelle kommuniziert mit dem Modellgenerator, sobald Benutzer eine Data-Mining-Aktion oder eine Data-Mining-Sequenz anstoßen. Deren Ausführung ist dann die Aufgabe des Modellgenerators. Sofern dieser auf Client und Middle-Tier verteilt ist, rufen sich die beiden Teilkomponenten gegenseitig auf und tauschen gegebenenfalls Zwischenergebnisse aus.

(6) Kommunikation zwischen Modellgenerator und Datenhaltung der Middle-Tier: Der Modellgenerator greift immer dann auf die Datenhaltung der Middle-Tier zu, wenn er Informationen aus den Metadaten benötigt. Bei den für den Modellgenerator relevanten Metadaten handelt es sich um Datenbeschreibungsobjekte, Data-Mining-Aktionen und -Sequenzen. Die Kommunikation zwischen dem Modellgenerator und der Datenhaltung auf der Middle-Tier ist ebenfalls wichtig, wenn die zu analysierenden Daten zu Beginn einer Data-Mining-Aktion auf die Middle-Tier geladen werden und der komplette Data-Mining-Algorithmus lokal ausgeführt wird. In diesem Fall übernimmt die Datenhaltung der Middle-Tier die relevanten Basisdaten sowie die Data-Mining-Ergebnisse. Das erstellte Data-Mining-Modell kann vom Modellgenerator ebenfalls in der Datenhaltung der Middle-Tier abgelegt werden.

(7) Kommunikation zwischen Modellgenerator und Data Warehouse: Zur Erstellung eines Data-Mining-Modells greift der Modellgenerator über die Datenzugriffsschnittstelle auf das Data Warehouse zu. Die benötigten Daten werden über SQL-Anweisungen gelesen. Dies kann einerseits einmalig zu Beginn einer Data-Mining-Aktion erfolgen. In diesem Fall werden die Basisdaten von der Datenhaltung der Middle-Tier übernommen, so dass der Modellgenerator lokal darauf zugreifen kann. Andererseits kann auch mehrfach während der Ausführung einer Data-Mining-Aktion auf die Daten im

Data Warehouse zugegriffen werden. Dann fordert der Modellgenerator bei jedem Schritt der Data-Mining-Aktion immer genau die Daten an, die für diesen Schritt notwendig sind. Die Kommunikation zwischen dem Modellgenerator und dem Data Warehouse ist ebenfalls notwendig, wenn Data-Mining-Ergebnisse direkt im Data Warehouse abgelegt werden sollen.

(8) Kommunikation zwischen Modellgenerator und Datenhaltung des Clients: Der Modellgenerator auf dem Client und die dortige Datenhaltung müssen immer dann kommunizieren, wenn Zwischenergebnisse einer Data-Mining-Aktion von der Middle-Tier an den Client übergeben werden sollen bzw. sobald bereits auf dem Client vorhandene Ergebnisse einer weiteren Analyse unterzogen werden sollen. Dies ist beispielsweise dann der Fall, wenn ein Teil eines Data-Mining-Algorithmus auf der Middle-Tier ausgeführt wird und auf der Basis der erstellten Zwischenergebnisse anschließend durch den Client das endgültige Data-Mining-Modell erstellt wird.

(9) Kommunikation zwischen Benutzungsschnittstelle und Visualisierung bzw. Export: Über die Benutzungsschnittstelle können Benutzer die Visualisierung verfügbarer Data-Mining-Modelle aktivieren bzw. das Exportieren dieser Modelle veranlassen.

(10) Kommunikation zwischen Visualisierung und Datenhaltung: Die Visualisierungskomponente auf dem Client greift auf die Datenhaltung zu, um ein Data-Mining-Modell zu lesen, das für den Benutzer dargestellt werden soll. Abhängig davon, ob ein Modell auf der Middle-Tier oder auf dem Client erstellt wurde, werden die Daten von der Datenhaltung der jeweiligen Schicht angefordert. Die Information, wo die Data-Mining-Modelle abgelegt sind, ergibt sich aus den von der Middle-Tier gelesenen Metadaten.

(11) Kommunikation zwischen Export und Datenhaltung: Für das Exportieren der Data-Mining-Modelle gilt bezüglich der notwendigen Kommunikation das für die Kommunikation zwischen Visualisierung und Datenhaltung Gesagte.

5.3.2 Bewertung der allgemeinen Data-Mining-Architektur

Nachdem in den beiden vorangegangenen Abschnitten die wichtigsten Aspekte der allgemeinen Data-Mining-Architektur beschrieben wurden, kann nun die

Bewertung erfolgen. Diese wird hier entlang der in Abschnitt 5.2 erläuterten Qualitätsattribute durchgeführt. Insbesondere wird hierbei auch auf die spezifischen Eigenschaften von Data-Mining-Werkzeugen eingegangen, die gegen die Verwendung anderer Architekturvarianten mir weniger Schichten sprechen.

5.3.2.1 Skalierbarkeit

Die beschriebene Architektur bietet gute Voraussetzungen für ein skalierbares Data-Mining-Werkzeug. Dies gilt sowohl für die Skalierbarkeit bei einer wachsenden Zahl von Benutzern als auch für die Skalierbarkeit bei wachsenden Datenbeständen.

Eine wachsende Zahl von Benutzern erhöht die Last für die Middle-Tier. Dies gilt sowohl für die Datenaufbereitung als auch für den Modellgenerator. Wird diese Last zu groß, so bietet die Architektur die Möglichkeit, die Middle-Tier auf mehreren Rechner zu replizieren. Jeder der Middle-Tier-Rechner ist dann nur für die Anfragen einzelner, zugeordneter Clients zuständig. Einerseits wird so die Verarbeitung der Datenaufbereitungs- und Analyseanfragen auf mehrere Rechner verteilt. Andererseits werden auch die aufbereiteten Daten und Data-Mining-Ergebnisse verteilt gespeichert. Durch zusätzliche Benutzer wird auch die Last für die Data-Warehouse-Schicht erhöht. Hier erhöht sich die Anzahl der zu verarbeitenden Anfragen. Allerdings wird auf dieser Ebene die Skalierbarkeit des Gesamtsystems nicht direkt von der Architektur unterstützt. Vielmehr hängt sie von der Skalierbarkeit des Datenbanksystems ab, auf dem das Data Warehouse beruht.

Für den Fall sich vergrößernder Datenmengen hängt die Skalierbarkeit des Gesamtsystems ebenfalls in erster Linie vom Data Warehouse ab. Ein umfangreicheres Data Warehouse stellt aber auch zusätzliche Anforderungen an die Komponenten der Middle-Tier. Eine Ursache hierfür ist die Tatsache, dass sich der lokal in der Middle-Tier zu haltende Datenbestand erhöht. Dies gilt insbesondere, wenn die Basisdaten zur weiteren Verarbeitung in die Middle-Tier geladen werden. Auf der Basis eines umfangreicheren Data Warehouse werden in der Regel aber auch umfangreichere Zwischenergebnisse und Data-Mining-Modelle erstellt. Darüber hinaus ist ein größerer Aufwand für die Aufbereitung der Daten sowie die Erstellung der Data-Mining-Modelle notwendig. Angesichts dieser zusätzlichen Anforderungen an die Komponenten der Middle-Tier, kann ein skalierbares System auf verschiedenen Wegen erreicht werden. Einerseits bietet sich

wiederum die bereits beschriebene Möglichkeit an, die Middle-Tier mehrfach parallel zu betreiben. Damit können die lokal in der Middle-Tier zu haltenden Daten auf mehrere Systeme verteilt werden. Dies beeinflusst allerdings nicht die Komponente zur Datenaufbereitung und den Modellgenerator. Insbesondere für die Datenaufbereitung bietet sich der Einsatz von Parallelrechnern in der Middle-Tier an. Damit können die lokal auszuführenden Operationen zur Datenaufbereitung auf mehrere Prozessoren verteilt werden. Für die Skalierbarkeit des Modellgenerators ist dagegen in erster Linie entscheidend, ob er auf skalierbaren Algorithmen basiert. Das gilt auch dann, wenn parallele Algorithmen genutzt werden. Parallele Algorithmen wurden für eine Reihe von Data-Mining-Methoden publiziert [FL98]. Diese Betrachtungen zeigen, dass die Skalierbarkeit in der vorgestellten Dreischichtenarchitektur weitestgehend von der Realisierung der Middle-Tier und dem Data Warehouse abhängt.

Bei einer Zweischichtenarchitektur entfallen dagegen die meisten der genannten Skalierbarkeitsvorteile. Bei der Thin-Client-Variante dieser Architektur ist der wesentliche Teil der Anwendungslogik auf dem Data Warehouse lokalisiert. Dort wird also für alle Clients sowohl die Datenaufbereitung als auch die Modellerstellung durchgeführt. Erhöht sich die Anzahl der Benutzer, so muss der damit verbundene Zusatzaufwand in erster Linie vom Data Warehouse erbracht werden. Dessen Replikation ist in der Praxis oft mit unverhältnismäßig großem Aufwand verbunden, so dass hier nicht von einer skalierbaren Architektur-Variante gesprochen werden kann.

Handelt es sich bei den Clients dagegen um sogenannte Fat Clients, so findet auf diesen der wesentliche Teil der Verarbeitung statt. Eine steigende Anzahl von Benutzern stellt hier aus Architektur-Sicht weniger ein Problem dar, da für zusätzliche Nutzer lediglich zusätzliche Clients bereitgestellt werden müssen. Da alle diese Clients allerdings mit dem Data Warehouse kommunizieren müssen, um auf die zu analysierenden Daten zuzugreifen, kann sich diese Kommunikation als ein erhebliches Skalierbarkeitsproblem darstellen. Dieses Problem wird allerdings erst bei einem sehr umfangreichen Data Warehouse zum tragen kommen, auf das sehr viele Clients zugreifen.

Der Einsatz von Fat Clients weist allerdings noch weitere Nachteile auf, die ihn für Data-Mining-Werkzeuge in vielen Fällen als unrealistisch erscheinen lassen. Der Ansatz stellt nämlich erhebliche Anforderungen an die Leistungsfähigkeit der Clients. Diese müssen einerseits so ausgelegt sein, dass die für Data Mining

relevanten Daten lokal auf diesen zur Verarbeitung bereitgestellt werden können. Andererseits müssen sie so ausgestattet sein, dass sie auch die Analyse der Daten vollständig ausführen können.

Wenn Data-Mining-Modelle auf der Grundlage der gesamten im Data Warehouse verfügbaren Historie erstellt werden, dann müssen hierbei sowohl im Rahmen der Datenvorverarbeitung als auch bei der eigentlichen Analyse große Datenmengen verarbeitet werden. In vielen Fällen ist es nicht möglich diese vollständig auf einem Client bereitzustellen, da diesem einerseits die erforderliche Speicherkapazität fehlt und andererseits der Datentransfer vom Data Warehouse zum Client sehr aufwändig ist. Der Aufwand für den Datentransfer erhöht sich noch zusätzlich, wenn auf Grund mangelnden Speicherplatzes die Daten mehrfach zum Client übertragen werden, so dass dort nur der für die aktuelle Verarbeitung notwendige Ausschnitt aus dem Data Warehouse bereitgestellt wird.

Data-Mining-Algorithmen sind häufig sehr komplex und berechnungsintensiv, so dass Rechner, die typischerweise als Client eingesetzt werden, schnell an ihre Grenzen stoßen und nicht mehr die von den Benutzern gewünschten Antwortzeiten liefern können. Insbesondere können diese Antwortzeiten häufig nur durch den Einsatz von Parallelrechnern und entsprechend zugeschnittene Algorithmen erzielt werden. Parallelverarbeitung ist auf typischen Client-Rechnern aber nur sehr eingeschränkt verfügbar. Der Einsatz sehr leistungsfähiger Client-Rechner ist mit zunehmender Anzahl der Clients nicht realistisch.

Diese Betrachtungen zeigen, dass beide Varianten einer Zweischichtenarchitektur bereits bei der Betrachtung der Skalierbarkeit mit erheblichen Nachteilen verbunden sind. Insbesondere legt es die Komplexität der Data-Mining-Algorithmen und der Umfang der zu verarbeitenden Daten nahe, für Data-Mining-Werkzeuge eine Dreischichtenarchitektur zu verwenden.

Die Diskussion von Fat Clients soll hier noch um einen Gesichtspunkt ergänzt werden, der sich keinem der in dieser Arbeit berücksichtigten Qualitätsattribute für Systemarchitekturen zuordnen lässt. Bei dieser Architekturvariante werden große Teile der Basisdaten des Data Warehouse zumindest temporär auf unterschiedlichen Clients gespeichert. Der Zugriff auf diese Daten und deren Weiterverwendung kann nicht kontrolliert werden. In einer Dreischichtenarchitektur sind auf einem Client dagegen nur verdichtete Daten und Data-Mining-Ergebnisse verfügbar. Unkontrollierte Zugriffe auf Basisdaten sind nicht möglich.

5.3.2.2 Lastbalancierung

Die allgemeine Data-Mining-Architektur bietet ebenfalls gute Voraussetzungen für statische und dynamische Lastbalancierung. An folgenden Stellen spielen Fragen der Lastbalancierung eine Rolle:

- Wenn die Middle-Tier aus mehreren Rechnern besteht, dann muss für jede Anfrage des Clients der zugeordnete Middle-Tier-Server festgelegt werden. Dies kann einerseits mit einer statischen Zuordnung der Clients zu Rechnern der Middle-Tier erfolgen. Eine optimale Nutzung der in der Middle-Tier vorhandenen Ressourcen lässt sich allerdings nur durch eine dynamische Lastbalancierung erreichen.
- Oft besteht die Möglichkeit, einen Teil eines Data-Mining-Algorithmus oder einen Teil der Datenaufbereitung direkt in SQL zu formulieren. Darüber hinaus besteht für weniger berechnungsintensive Teile von Data-Mining-Algorithmen die Alternative, diese direkt auf dem Client auszuführen. In diesen Fällen muss für jede Anfrage des Clients separat entschieden werden, ob die Ausführung komplett in der Middle-Tier erfolgen soll, oder ob ein Teil in das Data Warehouse oder den Client verlagert werden soll. Auch hier kann eine optimale Ressourcen-Nutzung nur dann erfolgen, wenn die Entscheidung dynamisch, unter Berücksichtigung der aktuellen Systemlast, getroffen wird.

Die für die beschriebenen Entscheidungen der Lastbalancierung notwendigen Informationen sind auf der Middle-Tier vorhanden. Für die verwendeten Algorithmen liegt im Modellgenerator die Information vor, welche Teile der Algorithmen direkt im Datenbanksystem realisiert werden können. Die Metadaten der Middle-Tier sollten Angaben darüber enthalten, inwieweit für die Middle-Tier mehrere Rechner zur Verfügung stehen.

Die Möglichkeit, dass Teile von Data-Mining-Algorithmen in allen drei Schichten der Architektur realisiert werden können, spiegelt sich auch in der Tatsache wider, dass in allen drei Schichten eine Datenhaltung vorhanden ist. Prinzipiell können so in allen Schichten der Architektur Basisdaten zur Analyse bereitgestellt und Analyseergebnisse in Form von Data-Mining-Modellen abgelegt werden. Die Lastbalancierung muss darum nicht nur berücksichtigen, in welchen Schichten die Algorithmen am effizientesten ausgeführt werden können. Daneben muss auch betrachtet werden, in welchen Schichten eines konkreten Systems die Voraussetzungen zur Speicherung der notwendigen Basisdaten, aufbereiteten Daten und erstellen Data-Mining-Modelle gegeben sind.

Die beschriebenen Möglichkeiten der Lastbalancierung entfallen bei einer Zweischichtenarchitektur weitgehend. Die Benutzer stellen ihre Anfragen jeweils von einem spezifischen Client aus. Damit ist auch der für die Anfrage zuständige Rechner bestimmt. Lediglich die Verlagerung eines Teils der Verarbeitung in das zu Grunde liegende Data Warehouse bietet hier in Ansätzen die Möglichkeit zur Lastbalancierung.

Obwohl die allgemeine Data-Mining-Architektur gute Voraussetzungen für die Berücksichtigung von Aspekten der Lastbalancierung bietet, nutzen existierende Werkzeuge diese Möglichkeiten kaum. Dies liegt u.U. daran, dass in einer typischen Anwendungsumgebung keine sehr große Anzahl von Nutzern eines Data-Mining-Werkzeugs gleichzeitig aktiv ist. Diese Situation wird sich aber mit der weiteren Verbreitung der Technologie ändern, so dass Aspekte der Lastbalancierung auch hier zunehmend relevant werden.

5.3.2.3 Nutzung von Datenbankfunktionalität

Auch die Nutzung von Datenbankfunktionalität für einzelne Data-Mining-Verfahren wird durch die beschriebene Architektur gut unterstützt. Die Trennung zwischen Client und Middle-Tier bietet den Vorteil, dass diese Nutzung von Eigenschaften des Data-Warehouse-Datenbanksystems lediglich den Modellgenerator der Middle-Tier betrifft und damit für die einzelnen Clients völlig transparent erfolgen kann. Teilschritte einzelner Data-Mining-Verfahren können natürlich nur dann in das Data Warehouse verlagert werden, wenn das zu Grunde liegende Datenbanksystem die entsprechende Formulierung in der Anfragesprache unterstützt. Der Modellgenerator benötigt für die Entscheidung, welche Teile eines Verfahrens in das Data Warehouse verlagert werden, entsprechende Informationen über das DWDBS. Diese Metadaten werden in der Middle-Tier verwaltet und vom Modellgenerator dort abgerufen.

5.3.2.4 Administrationsmöglichkeiten

Die beschriebene Dreischichtenarchitektur bietet für die Administration wesentliche Vorteile gegenüber Architekturen, die weniger Schichten umfassen. Die Trennung zwischen Client und Middle-Tier ermöglicht eine von den Clients getrennte Administration der Komponenten, die für die einzelnen Anfragen die wesentliche Last der Verarbeitung übernehmen. Da aus Administrationssicht die Leistungsfähigkeit des Gesamtsystems im Vordergrund steht, ist es wichtig, dass

für die Middle-Tier festgelegt werden kann, welche Clients welchen Anteil der Kapazität von Middle-Tier und Data Warehouse in Anspruch nehmen können. Dies kann beispielsweise bedeuten, dass für einzelne Clients der in der Middle-Tier nutzbare Speicher für Zwischenergebnisse oder die Komplexität der zulässigen Anfragen an das Data Warehouse beschränkt wird. Diese Möglichkeiten bestehen insbesondere in der Fat-Client-Variante der Zweischichtenarchitektur nicht, da hier das Data Warehouse in erster Linie als Datenspeicher fungiert und die Anfragen der einzelnen Clients beantwortet.

5.3.2.5 Wartbarkeit und Portierbarkeit

Das Attribut Wartbarkeit bezieht sich hier vor allem auf Veränderungen an einzelnen Komponenten der Architektur. Solche Veränderungen sind notwendig, wenn ein bestehender Algorithmus im System aktualisiert oder ersetzt werden muss oder wenn zusätzliche Data-Mining-Verfahren in ein Werkzeug aufgenommen werden sollen. Die allgemeine Data-Mining-Architektur unterstützt dies optimal, da die beschriebenen Änderungen ausschließlich den Modellgenerator und von diesem wiederum nur Teilkomponenten betreffen. In der Dreischichtenarchitektur müssen diese Änderungen vorwiegend auf der Middle-Tier vollzogen werden, während sie bei anderen Architekturvarianten oft sämtliche Clients betreffen. Insbesondere spricht dies gegen den Einsatz von Fat Clients, da hier Änderungen an allen Clients vorgenommen werden müssen, wenn einzelne Systemkomponenten verändert werden. Dies ist auch bei Anpassungen der Fall, die für die Benutzer nicht sichtbar sind, weil z.B. lediglich eine verbesserte Version eines Data-Mining-Algorithmus integriert wurde. In der Dreischichtenarchitektur hat dies dagegen nur Auswirkungen auf die Middle-Tier.

Weiterere Aspekte der Wartung eines Data-Mining-Werkzeugs sind die Aktualisierung der angebotenen Visualisierungsmöglichkeiten sowie die Zugriffsmöglichkeit auf zusätzliche Arten von Datenquellen. Die erste Erweiterung betrifft ausschließlich die Clients. Wobei hier auch nur diejenigen betroffen sind, die die veränderten Visualisierungsmöglichkeiten in Anspruch nehmen sollen. Die Anbindung von Datenquellen betrifft ausschließlich die Datenzugriffsschnittstelle der Middle-Tier, über welche die Änderungen im zu Grunde liegenden Data Warehouse vor den anderen Komponenten verborgen werden.

In Bezug auf die Portierbarkeit müssen hier zwei unterschiedliche Aspekte berücksichtigt werden. Ein Aspekt ist, inwiefern die Ausführung von Komponen-

ten der Middle-Tier bzw. des Clients auf unterschiedlichen Hardware- und/oder Softwareplattformen unterstützt wird. Dies wird in der vorgestellten Architektur nicht explizit unterstützt. Anders sieht es dagegen bei einem zweiten Aspekt der Portierbarkeit aus, dem Einsatz unterschiedlicher Plattformen für das Data Warehouse. Da der Zugriff auf das Data Warehouse ausschließlich über eine Datenzugriffsschnittstelle erfolgt und die relevanten Informationen über das Data Warehouse in den Metadaten der Middle-Tier abgelegt werden, können Änderungen im Zuge einer Portierung weitgehend isoliert in diesen beiden Komponenten der Middle-Tier vorgenommen werden. Die einzelnen Clients sind im Gegensatz zu Zweischichtenarchitekturen nicht betroffen.

5.4 Analyse der Architektur von OLAP-Werkzeugen

In gleicher Weise, wie dies im vorangegangenen Abschnitt erfolgt ist, soll nun die typische Architektur von OLAP-Werkzeugen beschrieben werden. In die folgende Darstellung sind in erster Linie Angaben zu den folgenden Produkten eingeflossen:

- BusinessObjects von Business Objects [Abe99],
- Hyperion Essbase von Hyperion [Hyp01a] [Hyp01b],
- MicroStrategy 7 von MicroStrategy [Mic95] [Mic99],
- PowerPlay Enterprise Server von Cognos [Cog99] [Cog01] sowie
- SAS OLAP Solution von SAS [Sas96].

Diese haben sich in der detaillierten Analyse der zu Grunde liegenden Architekturen als repräsentativ für die verfügbaren OLAP-Werkzeuge herausgestellt. Nach [PC00] hatten diese Produkte im Jahr 2000 zusammen einen Marktanteil von mehr als 50 Prozent.

5.4.1 Allgemeine OLAP-Architektur

Auch OLAP-Werkzeuge basieren in der Regel auf einer Dreischichtenarchitektur. Die wesentlichen Komponenten und deren Beziehungen sind in Abbildung 16 angegeben. Die Bezeichnungen *Client*, *Middle-Tier* und *Data Warehouse* werden wie bei der allgemeinen Data-Mining-Architektur für die einzelnen Schichten verwendet. In den folgenden Abschnitten werden die Komponenten dieser Schichten näher erläutert sowie deren Zusammenspiel beschrieben.

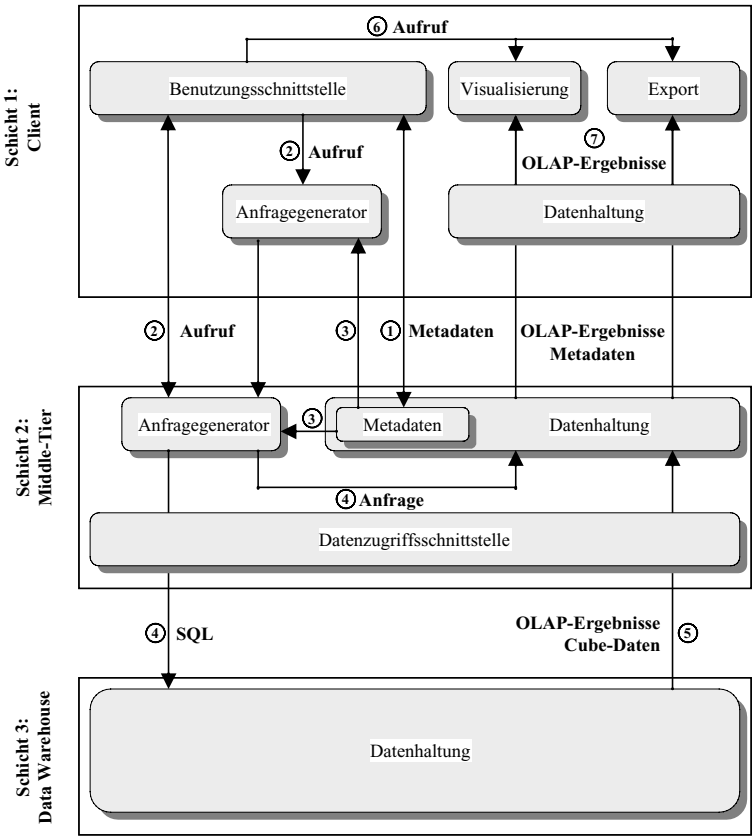

Abbildung 16: Ausführungsmodell der allgemeinen OLAP-Architektur

5.4.1.1 Komponenten

Die *Benutzungsschnittstelle* ist auch hier die zentrale, sichtbare Komponente des Gesamtsystems. Über diese haben Benutzer die Möglichkeit, die einzelnen OLAP-Analysen vorzubereiten und zu steuern. Zur Vorbereitung gehört die Festlegung der Metadaten. Hierbei werden alle für das einzelne Werkzeug relevanten Aspekte des konzeptuellen und des logischen Schemas des zu Grunde liegenden

Data Warehouse erfasst. Darüber hinaus ermöglicht die Benutzungsschnittstelle, die gewünschten OLAP-Analysen zu erstellen, diese zu starten sowie das Abrufen, die *Visualisierung* und den *Export* der erzielten Ergebnisse.

Zur Client-Schicht gehört in der Regel ein *Anfragegenerator*, der die notwendigen Anfragen für die gewünschten OLAP-Analysen erstellt. Die erzielten Ergebnisse können in der *Datenhaltung* des Clients abgelegt werden. Ein Teil des Anfragegenerators ist bei einigen Werkzeugen aber auch der Middle-Tier zugeordnet. Dies wird im Zusammenhang mit dem Ausführungsmodell erläutert.

Eine wichtige Komponente der Middle-Tier ist die *Datenhaltung*. Sie verwaltet in jedem Fall die *Metadaten* des Gesamtsystems. Bei MOLAP-Werkzeugen werden hier zusätzlich die erstellten Datenwürfel materialisiert. Die SQL-Anfragen, die notwendig sind, um diese Datenwürfel zu füllen, werden vom Anfragegenerator der Middle-Tier erstellt. Zusätzlich gibt es auf der Middle-Tier die *Datenzugriffsschnittstelle*, die den Zugriff auf unterschiedliche Datenbanksysteme ermöglicht. In der Regel werden hier eine ODBC-Schnittstelle sowie andere proprietäre Schnittstellen angeboten.

Das *Data Warehouse* bildet schließlich die Basis für die verschiedenen OLAP-Werkzeuge. Es stellt die zu analysierenden Daten bereit. Im Gegensatz zur allgemeinen Data-Mining-Architektur werden hier Zwischenergebnisse in der Regel allerdings nicht materialisiert.

5.4.1.2 Ausführungsmodell

Die Darstellung des typischen Ausführungsmodells für OLAP-Werkzeuge orientiert sich wiederum an entsprechenden Erläuterungen zur allgemeinen Data-Mining-Architektur. Wie in Abschnitt 5.3.1.2 werden im Folgenden entlang des KDD-Prozesses die Kommunikationsbeziehungen zwischen den verschiedenen Komponenten beschrieben sowie die Daten skizziert, die diese austauschen.

(1) Kommunikation zwischen Benutzungsschnittstelle und Datenhaltung der Middle-Tier: Die über die Benutzungsschnittstelle erfassten Metadaten werden in der Datenhaltung der Middle-Tier abgelegt. Sie umfassen in erster Linie alle für OLAP-Analysen relevanten Angaben zur Struktur und zum Inhalt des Data Warehouse und zu den daraus erstellten Datenwürfeln.

(2) Kommunikation zwischen Benutzungsschnittstelle und Anfragegenerator: Die Benutzungsschnittstelle ermöglicht es den Benutzern, die gewünschten OLAP-Analysen zu spezifizieren. Die hierbei erfassten Daten werden an den Anfragegenerator weitergegeben.

(3) Kommunikation zwischen Anfragegenerator und Datenhaltung der Middle-Tier: Sowohl der Anfragegenerator auf dem Client als auch der Anfragegenerator der Middle-Tier benötigen Zugriff auf die Metadaten, um die Anfragen zu erstellen, die für die Durchführung der gewünschten OLAP-Analysen notwendig sind. Bei einem MOLAP-Werkzeug generiert der Anfragegenerator des Clients Anfragen für die auf der Middle-Tier materialisierten Datenwürfel. Hierbei wird eine Anfragesprache genutzt, die von dem für die Speicherung des Datenwürfels verwendeten Multidimensionalen Datenbanksystem unterstützt wird. Bei einem ROLAP-Werkzeug dagegen werden vom Anfragegenerator des Clients SQL-Anfragen für das Data Warehouse erstellt. In der Regel wird für eine einzelne OLAP-Analyse eine ganze Sequenz von SQL-Anfragen generiert, wobei die einzelnen Anfragen der Sequenz voneinander abhängen können. Das OLAP-Ergebnis ergibt sich dann auf der Grundlage des Ergebnisses von einer oder mehreren Anfragen der Sequenz.

(4) Kommunikation zwischen Anfragegeneratoren und Data Warehouse: Bei einem ROLAP-Werkzeug gibt der Anfragegenerator des Clients über den Anfragegenerator der Middle-Tier die zur Durchführung von OLAP-Analysen erstellten SQL-Anfragen an das Data Warehouse weiter. Die Middle-Tier-Komponente berücksichtigt hierbei die Abhängigkeiten zwischen den einzelnen Anfragen einer Sequenz. Bei einem MOLAP-Werkzeug erstellt der Anfragegenerator auf der Middle-Tier die SQL-Anfragen an das Data Warehouse. Diese dienen hier aber nicht direkt der Durchführung von OLAP-Analysen, sondern liefern lediglich die Daten, die zur Materialisierung des Datenwürfels auf der Middle-Tier erforderlich sind. In Abbildung 16 sind diese als Cube-Daten bezeichnet.

(5) Kommunikation zwischen Data Warehouse und Datenhaltung der Middle-Tier: Das Data Warehouse verarbeitet die von den Anfragegeneratoren kommenden SQL-Anweisungen und gibt die jeweiligen Ergebnisse an die Datenhaltung der Middle-Tier weiter. Handelt es sich hierbei um Cube-Daten, so werden diese auf der Middle-Tier abgelegt. Wenn es sich dagegen um OLAP-

Ergebnisse handelt, werden diese in der Regel im Data Warehouse materialisiert und über die Middle-Tier direkt an den Client durchgereicht.

(6) Kommunikation zwischen Benutzungsschnittstelle und Visualisierung bzw. Export: Über die Benutzungsschnittstelle können Benutzer die Visualisierung verfügbarer OLAP-Analysen aktivieren bzw. das Exportieren dieser Ergebnisse veranlassen. In der Regel wird von Werkzeugen sowohl eine grafische Visualisierung als auch eine Darstellung in tabellarischer Form angeboten.

(7) Kommunikation zwischen Visualisierung, Export und Datenhaltung: Die Visualisierungskomponente wie auch die Export-Schnittstelle greifen auf die Datenhaltung zu, um die Ergebnisse der zur Visualisierung vorgesehenen OLAP-Analysen zu lesen. In der Regel werden diese Ergebnisse nach der Durchführung der Analyse in der Middle-Tier oder direkt im Data Warehouse auf dem Client abgelegt und sowohl für die Visualisierung als auch für die bereitgestellte Export-Funktionalität genutzt.

5.4.2 Bewertung der allgemeinen OLAP-Architektur

In diesem Abschnitt erfolgt die Bewertung der allgemeinen OLAP-Architektur unter Berücksichtigung der in Abschnitt 5.1 vorgestellten Kriterien.

5.4.2.1 Skalierbarkeit

Wie in Abschnitt 5.3.2.1 zerfällt die Skalierbarkeitsbetrachtung auch hier in zwei Teile: In die Betrachtung der Auswirkungen bei einer Vergrößerung der Benutzeranzahl und die entsprechenden Überlegungen für eine Zunahme des Datenbestands im Data Warehouse. In beiden Fällen muss die Betrachtung für MOLAP- und für ROLAP-Systeme getrennt erfolgen.

Zunächst zu den Auswirkungen, die eine größere Benutzeranzahl mit sich bringt. In einem MOLAP-System führt dies bei konstantem Anfrageaufkommen pro Benutzer zu einer größeren Anzahl von Anfragen an den in der Middle-Tier materialisierten Datenwürfel. Damit hängt die Skalierbarkeit des gesamten OLAP-Werkzeugs zunächst von den Eigenschaften des MDBS ab, das die Anfragen an den Datenwürfel verarbeitet. Ein skalierbares System kann dadurch erreicht werden, dass die gesamte Middle-Tier auf mehrere Rechnerknoten ver-

teilt wird und eine Replikation der Datenwürfel erfolgt. Hier unterstützt also die dargestellte Architektur die Skalierbarkeit des OLAP-Systems.

Weniger Einfluss hat die Architektur dagegen bei ROLAP-Systemen. Hier wird mit der Anzahl der Benutzer die Anzahl der Anfragen auf dem Data Warehouse erhöht. Die Skalierbarkeit hängt entsprechend von der Skalierbarkeit des Data-Warehouse-Datenbanksystems ab. Dies gilt ebenso bei einer Zunahme des Datenbestandes im Data Warehouse. Der Umfang des Datenbestandes hat zwar keinen Einfluss auf die Anzahl der vom Data Warehouse zu verarbeitenden Anfragen. Dafür muss von den einzelnen Anfragen aber potenziell ein größeres Datenvolumen verarbeitet werden.

In MOLAP-Systemen wiederum bietet die vorgestellte Architektur auch bei einer Vergrößerung des Datenbestands im Data Warehouse gute Voraussetzungen für ein skalierbares OLAP-System. Ein umfangreicheres Data Warehouse resultiert in umfangreicheren Datenwürfeln auf der Middle-Tier. Die Architektur bietet hier die Möglichkeit durch eine Verteilung der Middle-Tier auf mehrere Rechnerknoten und die parallele Verarbeitung der Anfragen ein skalierbares System zu erreichen. Voraussetzung ist hier allerdings, dass die entsprechenden Mechanismen zur Verteilung und Parallelverarbeitung von dem in der Middle-Tier verwendeten Multidimensionalen Datenbanksystem unterstützt werden.

5.4.2.2 Lastbalancierung

Wie in Abschnitt 5.3.2.2 für die allgemeine Data-Mining-Architektur beschrieben bietet eine Dreischichtenarchitektur auch für ein OLAP-System gute Voraussetzungen für statische und dynamische Lastbalancierung. Dies ist in erster Linie für MOLAP-Systeme relevant. Hier erfolgt die Anfrageverarbeitung für die einzelnen OLAP-Analysen auf der Middle-Tier. Fragen der Lastbalancierung spielen eine Rolle, sobald die Middle-Tier auf unterschiedliche Rechnerknoten verteilt ist. In diesem Fall kann für jede einzelne OLAP-Analyse der zuständige Rechner der Middle-Tier separat festgelegt werden.

Bei einem ROLAP-System dagegen werden die für die verschiedenen Analysen generierten SQL-Anweisungen in jedem Fall direkt im Data Warehouse ausgeführt. Damit trägt das DWDBS die Hauptlast für die Verarbeitung von OLAP-Analysen. Fragen der Lastbalancierung spielen unter Umständen zwar innerhalb dieses Datenbanksystems eine Rolle. Dies ist allerdings unabhängig von der

Architektur der darauf aufsetzenden Anwendungen. Gleiches gilt für die SQL-Anweisungen, die notwendig sind, um in einem MOLAP-System die Datenwürfel in der Middle-Tier zu füllen.

5.4.2.3 Nutzung von Datenbankfunktionalität

Die Anfragegeneratoren können SQL-Anfragen in Abhängigkeit von der Mächtigkeit des DWDBS erstellen. Die hierzu notwendigen Informationen sind in den Metadaten verfügbar. In der vorgestellten Architektur ist damit die optimale Nutzung der vom Data Warehouse bereitgestellten Datenbankfunktionalität möglich. Dies gilt allerdings unabhängig von der Anzahl der Schichten, die eine OLAP-Architektur aufweist. Auch in anderen Architekturen, die weniger Schichten umfassen, ist die Nutzung der Datenbankfunktionalität in gleicher Weise möglich. Voraussetzung ist lediglich, dass der Anfragegenerator in jeder der Architekturvarianten Zugriff auf die Metadaten hat und damit die notwendige Informationsgrundlage für eine optimale Nutzung der Datenbankfunktionalität gegeben ist.

5.4.2.4 Administrationsmöglichkeiten

Wie bereits in Abschnitt 5.3.2.4 beschrieben zeichnet sich Dreischichtenarchitektur unter anderem durch Vorteile bei der Administration aus. Dies gilt auch für die allgemeine OLAP-Architektur. Die Hauptlast der Verarbeitung liegt hier wiederum bei der Middle-Tier und dem Data Warehouse. Durch die Trennung in drei Schichten kann optimal kontrolliert werden, wie die Kapazität von Middle-Tier und Data Warehouse den einzelnen Clients zugeordnet wird.

5.4.2.5 Wartbarkeit und Portierbarkeit

Die wesentlichen Aspekte der Wartung sind bei einem OLAP-Werkzeug die Erweiterung der angebotenen Analysefunktionalität sowie die Integration zusätzlicher Visualisierungs- und Exportmöglichkeiten. Durch eine erweiterte Analysefunktionalität werden den Benutzern zusätzliche Möglichkeiten geboten, Informationen aus dem Data Warehouse zu gewinnen. Zu deren Realisierung sind in erster Linie Änderungen am Anfragegenerator notwendig. Er muss die zusätzlichen Möglichkeiten in entsprechende Anfragen an das zu Grunde liegende DBS übersetzen. Erweiterte Visualisierungs- und Exportmöglichkeiten betreffen ausschließlich die entsprechenden Komponenten der Clients. Auf

Grund der Architektur konzentrieren sich also die häufigsten Wartungsaktivitäten für ein OLAP-Werkzeug in der Regel jeweils auf eine einzelne Komponente. Soweit dies durch die Architektur überhaupt möglich ist, sind hier also gute Voraussetzungen für einen geringen Wartungsaufwand bei der Änderung von OLAP-Werkzeugen gegeben. Hinsichtlich der Portierbarkeit unterscheidet sich die allgemeine OLAP-Architektur nicht von der allgemeinen Data-Mining-Architektur.

5.5 Architektur für ein integriertes System

Die bisherige Bewertung der vorgestellten Architekturen erfolgte immer isoliert für die einzelnen Analyseansätze. Wie in Kapitel 3 beschrieben, setzen auf einem typischen Data Warehouse allerdings unterschiedliche Analyseanwendungen auf. Hieraus ergeben sich zusätzliche Kriterien für die Bewertung der Systemarchitekturen. Diese werden im folgenden Abschnitt erfasst und für die weitere Bewertung der allgemeinen Data-Mining-Architektur sowie der allgemeinen OLAP-Architektur genutzt. Auf Basis der identifizierten Defizite wird schließlich eine Architektur vorgestellt, die die integrierte Nutzung unterschiedlicher Analyseanwendungen auf demselben Data Warehouse geeignet unterstützt.

5.5.1 Defizite der beschriebenen Architekturen

Sowohl die allgemeine Data-Mining-Architektur als auch die allgemeine OLAP-Architektur weisen die folgenden Defizite auf:

- Sie bieten keine anwendungsunabhängige Optimierung für Anfragesequenzen.
- Sie erlauben keine anwendungsübergreifende Verwaltung von Metadaten.

In diesem Abschnitt werden beide Defizite ausführlich erläutert. Sie bilden die Motivation für die in Abschnitt 5.5.2 vorgestellte erweiterte Systemarchitektur für Analyseanwendungen.

In den bisher dargestellten Architekturen für Data-Mining- und OLAP-Anwendungen werden von Komponenten, die sich in der Regel auf der Middle-Tier befinden, SQL-Anfragen an das Data Warehouse erstellt. In Abbildung 15 werden diese als Modellgenerator und als Datenaufbereitung bezeichnet, während es sich in Abbildung 16 um den Anfragegenerator handelt. Hierbei ist es häufig der Fall, dass für eine durch die Benutzer angestoßene Aktion eine ganze Reihe von SQL-Anfragen generiert und vom Data-Warehouse-Datenbanksystem ausgeführt

werden müssen. Eine solche Folge von SQL-Anfragen, die insgesamt zu einer einzelnen Aktion oder Anfrage von Benutzern gehört, wird im Folgenden als *Anfragesequenz* bezeichnet. Eine exakte Definition des Begriffs Anfragesequenz erfolgt in Kapitel 6.

In den beschriebenen Systemarchitekturen wird von einer synchronen Verarbeitung der generierten Anfragesequenzen ausgegangen. Hierbei wird jeweils eine SQL-Anfrage zur Ausführung an das Data Warehouse geschickt. Erst nach der erfolgreichen Bearbeitung der SQL-Anfrage wird die nächste Anfrage der Sequenz gestartet. Dies bedeutet, dass zu jedem Zeitpunkt nur eine Anfrage aus einer Anfragesequenz im Data Warehouse aktiv ist. Das DWDBS hat damit nur Optimierungsmöglichkeiten bezogen auf diese eine Anfrage. Es ist allerdings auch eine Optimierung denkbar, die die Anfragesequenz insgesamt betrifft. Eine solche Optimierung kann beispielsweise in der Veränderung einzelner Anfragen der Anfragesequenz bestehen. Welche Möglichkeiten bestehen, dieses Optimierungspotenzial im Rahmen der beschriebenen Architekturen zu nutzen?

Ein anwendungsorientierter Ansatz ist die Verbesserung der Systemkomponenten, die die Anfragesequenzen erzeugen. Ein verbesserter Generierungsalgorithmus kann Anfragesequenzen erstellen, die vom Data-Warehouse-Datenbanksystem optimal ausgeführt werden können. Dieser Ansatz ist in zweierlei Hinsicht sehr komplex. Einerseits muss der Generierungsalgorithmus für jede einzelne Anwendung individuell verbessert werden. Andererseits müssen bei der Bestimmung der optimalen Anfragesequenz die Charakteristika des zu Grunde liegenden DBS berücksichtigt werden. Damit muss in jeder Anwendung nicht nur ein Generierungsalgorithmus realisiert werden, sondern es werden Varianten von diesem für jede unterstützte Version eines Datenbanksystems notwendig.

Als Alternative bietet sich ein datenbankorientierter Ansatz an. Bei diesem übernimmt das Data-Warehouse-Datenbanksystem die Optimierung der Anfragesequenzen. Auch dieser Ansatz ist nicht weniger komplex als der anwendungsorientierte. Die Komplexität resultiert einerseits aus der Tatsache, dass Strategien zur Optimierung von Anfragesequenzen für jedes Datenbanksystem separat entwickelt werden müssen, da die einzelnen Systeme unterschiedlichen Optimierungsparadigmen (z.B. Bottom-Up oder Top-Down) folgen. Andererseits ist selbst die Integration von Optimierungsstrategien für Anfragesequenzen in ein einzelnes Datenbanksystem mit erheblichem Aufwand verbun-

den. Hierbei muss die bereits realisierte Anfrageoptimierung berücksichtigt werden. Da diese sich in der Regel auf einzelne Anfragen und nicht auf Sequenzen bezieht, beschränkt sich die Hinzunahme von Anfragesequenz-Optimierungen nicht auf das Einfügen zusätzlicher Operatoren, Heuristiken und Kostenfunktionen in den vorhandenen Optimierer. Vielmehr muss die Optimierungsinfrastruktur grundlegend erweitert werden. Grundsätzlich ist es bei Erweiterungen in einem kommerziellen System aber notwendig, dass die Leistungsfähigkeit bestehender Anwendungen nicht negativ beeinflusst wird. Bei einer Erweiterung der Optimiererkomponente bedeutet dies, dass der für die Anfrageoptimierung notwendige Aufwand durch die Berücksichtigung von Anfragesequenzen nicht wesentlich wachsen darf und das Optimierungsergebnis für einzelne Anfragen nicht weniger effizient ausführbar sein darf, als dies vor der Erweiterung der Fall war. Damit ist auch dieser datenbankorientierte Ansatz mit wesentlichen Nachteilen behaftet. Im Rahmen der vorgestellten Architekturen gibt es keine zusätzlichen Möglichkeiten, die speziellen Anforderungen von Data-Mining- und OLAP-Anwendungen an die Anfrageoptimierung zu berücksichtigen. Dies stellt eine wesentliche Motivation für die im folgenden Abschnitt vorgestellte, erweiterte Systemarchitektur dar.

Die vorgestellten Architekturen weisen ein weiteres Defizit bezüglich der Nutzung von Metadaten auf. Typischerweise verwaltet jede der Analyseanwendungen ihre individuellen Metadaten. Dies bedeutet, dass Informationen in verschiedenen Metadaten-Repositories redundant gehalten werden. Diese Redundanz ist einerseits nicht notwendig. Andererseits hat sie auch zur Folge, dass die gemeinsame Nutzung der Metadaten erschwert wird. Dieser anwendungsübergreifende Zugriff auf Metadaten ist aber immer dann notwendig, wenn eine Analyseanwendung auf die Ergebnisse einer anderen Anwendung zugreifen und mit diesen weiterarbeiten soll.

5.5.2 Erweiterte Systemarchitektur für Analyseanwendungen

Die erweiterte Architektur, die die identifizierten Defizite berücksichtigt, ist in Abbildung 17 dargestellt. Diese unterscheidet sich von den bereits vorgestellten Architekturen im Wesentlichen in zwei Punkten. Es ist eine zusätzliche Schicht zwischen der eigentlichen Middle-Tier und dem Data Warehouse vorhanden. Darüber hinaus sind die Metadaten anders eingestuft. Diese Komponenten sind in der Abbildung hervorgehoben.

Zunächst soll aber für die einzelnen Schichten erläutert werden, welche Komponenten aus der allgemeinen Data-Mining-Architektur bzw. der allgemeinen OLAP-Architektur übernommen wurden. Der Client ist in der erweiterten Architektur weitgehend unverändert. Er umfasst die *Benutzungsschnittstelle*, eine *Datenhaltung* sowie die Komponenten zur *Visualisierung* und zum *Export* der

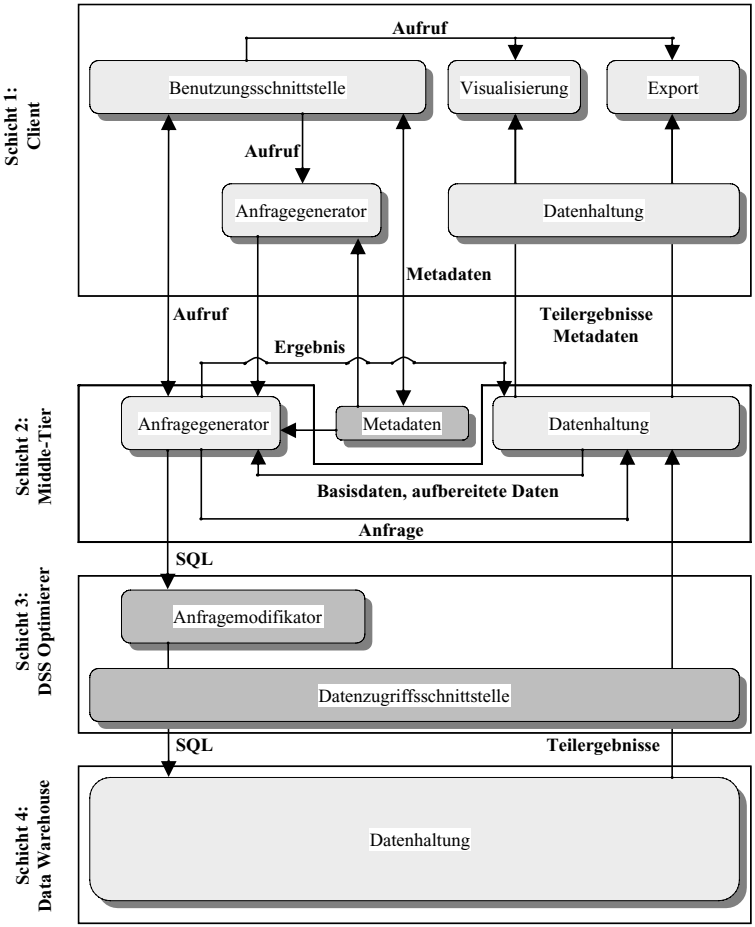

Abbildung 17: Erweiterte Systemarchitektur für Analyseanwendungen

Analyseergebnisse. Der in der erweiterten Architektur vorhandene *Anfragegenerator* entspricht dieser Komponente aus der OLAP-Architektur bzw. dem Modellgenerator aus der Data-Mining-Architektur. Abhängig davon, ob es sich bei einem konkreten System um ein Data-Mining-Werkzeug oder ein OLAP-Werkzeug handelt, werden von diesem Anfragegenerator die passenden Anfragen an Komponenten auf darunter liegenden Schichten erzeugt und die gelieferten Ergebnisse entsprechend aufbereitet.

In ähnlicher Art und Weise wurde die Middle-Tier gegenüber den ursprünglichen Architekturen verändert. Auch diese enthält eine als *Anfragegenerator* bezeichnete Komponente, die Anfragen an die darunter liegenden Schichten erstellt. Im Falle eines Data-Mining-Werkzeugs dienen diese Anfragen sowohl der Datenaufbereitung als auch der Erstellung von Data-Mining-Modellen. Bei einem OLAP-Werkzeug hingegen werden im Anfragegenerator Anfragen erstellt, die notwendig sind, um die angeforderten OLAP-Ergebnisse bereitzustellen. Der Anfragegenerator ist damit auch in der Middle-Tier eine generische Version der entsprechenden Komponenten aus den einzelnen Architekturen.

Metadaten sind auch in der Middle-Tier der erweiterten Architektur vorhanden. In dieser haben sie allerdings insofern eine andere Funktion, als sie anwendungsübergreifend genutzt werden. Als Grundlage für ein solches gemeinsam genutztes Metadaten-Repository kann beispielsweise das Common Warehouse Metamodel dienen [Obj01a] [Obj01b].

Die *Datenzugriffsschnittstelle* aus den ursprünglichen Architekturen ist in der erweiterten Architektur nur noch in der zusätzlichen Schicht, dem DSS-Optimierer vorhanden. Diese Schicht wurde speziell zur Optimierung generierter Anfragesequenzen eingefügt. Neben der erwähnten Datenzugriffsschnittstelle enthält sie den *Anfragemodifikator*. Dieser nimmt Sequenzen von SQL-Anfragen aus den darüber liegenden Schichten entgegen und optimiert sie. Strategien der Optimierung sind Gegenstand von Kapitel 6. Die modifizierten Anfragesequenzen werden über die Datenzugriffsschnittstelle auf dem Data Warehouse ausgeführt.

In der Data-Warehouse-Schicht hat sich gegenüber den ursprünglichen Architekturen keine Veränderung ergeben. Dieses nimmt die SQL-Anfragen und modifizierten Anfragesequenzen der DSS-Optimierer-Schicht entgegen, führt sie aus und liefert die Teilergebnisse an die Middle-Tier zurück. Bei Data-Mining-Analyse werden diese unter Umständen auch im Data Warehouse materialisiert.

5.5.3 Bewertung der erweiterten Architektur

Die Bewertung der Architektur erfolgt hier in zwei getrennten Abschnitten. Zunächst wird betrachtet, inwieweit die Eigenschaften der allgemeinen Data-Mining-Architektur und der allgemeinen OLAP-Architektur auch für die erweiterte Architektur gelten. Anschließend wird betrachtet, welche Vorteile die erweiterte Architektur für den Einsatz unterschiedlicher Analysewerkzeuge auf einem Data Warehouse bietet.

Die Überlegungen zur *Skalierbarkeit* gelten für die erweiterte Architektur ebenso, wie für die in Abbildung 15 und Abbildung 16 dargestellten Architekturen. Soweit die Skalierbarkeit nicht direkt vom Data Warehouse abhängt, beruht diese auf der Möglichkeit, die Middle-Tier auf mehrere Rechner zu verteilen. Diese Möglichkeit besteht natürlich ebenso in der erweiterten Architektur. Dieselbe Verteilungsmöglichkeit bietet sich hier zusätzlich für die eingeführte vierte Schicht, den DSS-Optimierer. Die erweiterte Architektur weist demnach in Bezug auf die Skalierbarkeit keinerlei Nachteile gegenüber den anderen Architekturen auf.

Auch die Möglichkeiten der *Lastbalancierung* bleiben in der erweiterten Architektur voll erhalten. Sowohl für die Zuordnung eines Rechners der Middle-Tier bzw. des DSS-Optimierers zu einer Anfrage eines Clients als auch für die Verlagerung von Teilen der Analysealgorithmen in die Datenbank des Data Warehouse können in den Metadaten alle relevanten Daten bereitgestellt werden.

In Bezug auf die Nutzung der *Datenbankfunktionalität*, die *Administrationsmöglichkeiten* sowie *Wartbarkeit* und *Portierbarkeit* weist die erweiterte Architektur im Wesentlichen dieselben Eigenschaften auf, wie die beschriebenen Dreischichtenarchitekturen für Data Mining und OLAP. Der Anfragegenerator kann auf der Grundlage der Metadaten Anfragen erstellen, die die Funktionalität des zu Grunde liegenden Datenbanksystems möglichst optimal nutzen. Durch die Aufteilung in mehrere Schichten kann auch in der erweiterten Architektur exakt kontrolliert werden, welche Kapazität von Middle-Tier und Data Warehouse den einzelnen Clients zugeordnet wird. Eine Erweiterung der Analysefunktionalität sowie der Visualisierungs- und Exportmöglichkeiten ist in der erweiterten Architektur ebenso einfach zu realisieren, wie dies in den Abschnitten 5.3.2.5 und 5.4.2.5 beschrieben ist. Da diese Erweiterungen völlig unabhängig von der in der

erweiterten Architektur hinzugekommenen DSS-Optimierer-Schicht erfolgen können, bleiben die Vorteile hinsichtlich der Wartbarkeit vollständig erhalten.

Welche Vorteile bietet nun die erweiterte Architektur für den Einsatz unterschiedlicher Analysewerkzeuge auf der Basis eines Data Warehouse? Ein Punkt ist hier die anwendungsübergreifende Verwaltung der Metadaten. Diese wird durch die Architektur unterstützt und ermöglicht die Vermeidung von Redundanzen, die entstehen, wenn jedes Werkzeug eigene Metadaten verwaltet. Allerdings bedeuten diese zentralen Metadaten auch, dass eine anwendungsübergreifende Festlegung auf ein einheitliches Format erfolgen muss. Vorhandene Anwendungen müssen schließlich so modifiziert werden, dass sie mit Metadaten, die in diesem Format abgelegt sind, arbeiten können.

Der entscheidende Vorteil der erweiterten Architektur für Analyseanwendungen ist aber, dass durch den DSS-Optimierer eine anwendungsunabhängige Optimierung von Anfragesequenzen ermöglicht wird. Gerade in den unterschiedlichen Analyseanwendungen werden häufig solche Anfragesequenzen generiert. In Abschnitt 5.5.1 wurde ein anwendungsorientierter und ein datenbankorientierter Ansatz zu deren Optimierung diskutiert. Beide sind mit erheblichem Aufwand verbunden. Beim ersten Ansatz muss eine optimierte Generierung der Anfragesequenzen für jede einzelne Anwendung entwickelt werden. Für den zweiten Ansatz ist es notwendig, die Optimierung von Anfragesequenzen in jedem der für ein Data Warehouse eingesetzten DBVS zu realisieren. In der erweiterten Architektur beschränkt sich die Implementierung von Optimierungsstrategien hingegen auf die Ebene des DSS-Optimierers. Dieser kann Anfragesequenzen bestehender und zukünftiger Anwendungen entgegennehmen und verarbeiten. Das Ergebnis dieser Verarbeitung kann beispielsweise eine modifizierte Anfragesequenz sein oder eine veränderte Ausführungsreihenfolge für die Sequenz. Die Modifikation der Anfragesequenz hat natürlich immer eine effizientere Verarbeitung durch das zu Grunde liegende Datenbanksystem zum Ziel. In der erweiterten Architektur kann dies allein durch das Hinzufügen des DSS-Optimierers als zusätzliche Komponente erreicht werden. An anderen Systemkomponenten müssen keine nennenswerten Änderungen vorgenommen werden. Die einzige Veränderung besteht darin, dass Anfragegeneratoren die erzeugten Anfragen nicht mehr direkt an das Data Warehouse schicken, sondern diese zunächst dem DSS-Optimierer übergeben. Hierzu müssen die notwendigen Schnittstellen bereitgestellt werden. Der beschriebene Vorteil der erweiterten Architektur gilt nicht nur bei der erstmaligen Realisierung der Optimierungsstrategien für Anfragesequen-

zen. Die Entwicklung weiterer Strategien, die Weiterentwicklung der zu Grunde liegenden Datenbanksysteme sowie Änderungen in den Anfragegeneratoren machen Anpassungen des DSS-Optimierers notwendig. Gegenstand dieser Wartung ist allerdings nur der DSS-Optimierer selbst und keine weitere Systemkomponente. Dies ist ein weiterer Aspekt, der die gute Wartbarkeit und Portierbarkeit der Systeme in der erweiterten Architektur stützt.

5.6 Zusammenfassung

In diesem Kapitel wurden die Architekturen existierenden OLAP- und Data-Mining-Werkzeuge analysiert. Es hat sich gezeigt, dass die Systeme in beiden Bereichen in der Regel auf Dreischichtenarchitekturen beruhen die einige Gemeinsamkeiten aufweisen. Diese haben zunächst zur Darstellung einer allgemeinen Data-Mining-Architektur sowie einer allgemeinen OLAP-Architektur geführt. Beide Architekturen wurden schließlich bezüglich der Kriterien Skalierbarkeit, Lastbalancierung, Nutzung von Datenbankfunktionalität, Wartbarkeit und Portierbarkeit bewertet. Während in diesen Bereichen eine durchweg positive Bewertung erfolgen konnte, habe sich Defizite hinsichtlich der anwendungsübergreifenden Verwaltung von Metadaten sowie der anwendungsunabhängigen Optimierung von Anfragesequenzen gezeigt. Diesen Defiziten trägt die vorgestellte erweiterte Systemarchitektur für Analyseanwendungen Rechnung. Sie zeichnet sich insbesondere durch eine zusätzliche Schicht aus, die Strategien zur Optimierung von Anfragesequenzen realisiert. Eine detaillierte Untersuchung potenzieller Strategien ist Gegenstand des folgenden Kapitels.

6 Optimierungsstrategien für einen DSS-Optimierer

In der erweiterten Architektur für die Integration von Data Mining und OLAP stellt der DSS-Optimierer die wesentliche Neuerung gegenüber den Architekturen dar, die allgemein für Data-Mining- und OLAP-Werkzeuge verwendet werden. Das primäre Ziel dieser Komponente ist es, für eine effizientere Ausführung von Anfragesequenzen zu sorgen, die von den Anfragegeneratoren der verschiedenen Analysewerkzeuge aus dem Bereich Decision Support generiert werden. In diesem Kapitel werden unterschiedliche Optimierungsansätze, die in einem DSS-Optimierer eingesetzt werden können, identifiziert und bewertet. Als Grundlage hierzu wird zunächst der Begriff der Anfragesequenz definiert und eine Reihe von Beispielen für solche Sequenzen vorgestellt. In Abschnitt 6.2 wird darauf aufbauend eine breite Palette von Optimierungsansätzen erläutert, mit der klassischen Anfrageoptimierung in relationalen Datenbanksystemen verglichen und qualitativ bewertet. Die quantitative Bewertung erfolgt in Abschnitt 6.3. Hier wird anhand von Messergebnissen die Wirksamkeit der verschiedenen Optimierungsansätze nachgewiesen. Welche Konsequenzen sich aus der qualitativen und quantitativen Bewertung der Optimierungsansätze für die Realisierung eines DSS-Optimierers ergeben wird abschließend in Abschnitt 6.4 diskutiert.

6.1 Anfragesequenzen

In Abschnitt 5.5.1 wurde bereits erläutert, dass in verschiedenen Analysewerkzeugen häufig Sequenzen von SQL-Anfragen generiert werden. Diese treten beispielsweise auf, wenn Benutzer Analysen mit einem ROLAP-Werkzeug ausführen. Aber auch die einzelnen Schritte zur Datenvorverarbeitung im Rahmen von Data-Mining-Analysen werden häufig in eine Folge von SQL-Anweisungen umgesetzt. In beiden Fällen wird der Anfragegenerator eines Werkzeugs eine einzelne, durch den Benutzer ausgelöste Aktion in der Regel nicht direkt in eine einzige SQL-Anweisung übersetzen. Mehrere Gründe sind hierfür ausschlaggebend: In manchen Fällen kann es daran liegen, dass für die durch die Benutzer angestoßene Aktion keine einzelne SQL-Anweisung formuliert werden kann. Dieser Aspekt wird in Abschnitt 6.2.7 bei der Diskussion eines der Optimierungsansätze nochmals aufgegriffen. In anderen Fällen ist die Ursache eher in einer Vereinfachung des Anfragegenerators sowie in der Tatsache, dass Anfragesequenzen leichter durch den Menschen interpretiert und überprüft werden können, zu sehen. Die Erstellung von Anfragesequenzen stellt einerseits eine Vereinfachung des Anfragegenerators dar, da eine Sequenz typischerweise die

Schritte widerspiegelt, in denen die Benutzer mit Hilfe der Benutzungsschnittstelle ihre Anforderungen definieren. Andererseits weist eine Anfragesequenz in der Regel auch weniger Abhängigkeiten hinsichtlich des zu Grunde liegenden Datenbanksystems auf. Dies liegt daran, dass die einzelnen Teilanfragen der Sequenz häufig vergleichsweise einfach strukturiert sind und keine spezifischen Eigenschaften einzelner Datenbanksysteme ausnutzen. Der Anfragegenerator muss somit nur bei wenigen Details einer Anfragesequenz eine Anpassung für ein spezifisches System vornehmen und nicht einen separaten Generierungsalgorithmus für jede Version eines Datenbanksystems realisieren.

6.1.1 Definition der Anfragesequenz

In diesem Abschnitt sind alle Definitionen zusammengefasst, die beschreiben, was im Rahmen dieser Arbeit unter einer Anfragesequenz zu verstehen ist. Die Definitionen sind so aufgebaut, dass sie alle Sequenzen von SQL-Anweisungen abdecken, die von typischen Analyseanwendungen generiert werden.

Definition 10: CREATE-Anweisung

Bei einer *CREATE-Anweisung* handelt es sich um eine gültige SQL-Anweisung, die entweder eine Tabelle (CREATE TABLE) oder eine Sicht (CREATE VIEW) erstellt.

Definition 11: INSERT-Anweisung

Bei einer *INSERT-Anweisung* handelt es sich um eine gültige SQL-Anweisung, die Daten in eine Tabelle einfügt (INSERT INTO).

Definition 12: DROP-Anweisung

Bei einer *DROP-Anweisung* handelt es sich um ein gültige SQL-Anweisung, die entweder eine Tabelle (DROP TABLE) oder eine Sicht (DROP VIEW) löscht.

Definition 13: Datenbankanweisungen

Sei O die Menge der gültigen Anweisungen, die ein Datenbanksystem ausführen kann. Sei $C \subseteq O$ die Menge der gültigen CREATE-Anweisungen, $I \subseteq O$ die Menge der gültigen INSERT-Anweisungen und $D \subseteq O$ die Menge der gültigen DROP-Anweisungen. Die Mengen O, C, I und D enthalten jeweils

die leere Anweisung, d.h. eine Anweisung, die ohne jegliche Folgen für das Datenbanksystem bleibt.

Die Definitionen 10 bis 13 legen die betrachteten Datenbankanweisungen fest. Zunächst werden hier alle korrekt formulierten Anweisungen für ein Datenbanksystem betrachtet. Diese werden schließlich, je nachdem ob es sich um CREATE-, INSERT- oder DROP-Anweisungen handelt, in die drei entsprechenden Mengen aufgeteilt. In den folgenden Definitionen werden diese Mengen genutzt, um eine Teilanfrage und die darauf aufbauende Anfragesequenz festzulegen. Die leere Anweisung wird in den Mengen *O*, *C*, *I* und *D* benötigt, um, entsprechend der folgenden Definition, Teilanfragen zu ermöglichen, von denen einzelne Bestandteile ohne Auswirkung auf die Datenbank bleiben.

Definition 14: Teilanfrage

Eine *Teilanfrage* a_i ist das 4-Tupel (c_i,i_i,d_i,o_i) mit $c_i \in C$, $i_i \in I$, $d_i \in D$, $o_i \in O$, $i \in N$ [1].

Definition 15: Abhängigkeit zwischen Teilanfragen

Eine Teilanfrage a_i ist *abhängig* von einer Teilanfrage a_j, ausgedrückt durch $a_j \rightarrow a_i$, wenn gilt: c_i, i_i oder o_i verwenden direkt oder indirekt ein in c_j erstelltes Objekt und $i \neq j$.

Definition 16: Anfragesequenz

Eine *Anfragesequenz S* ist die Folge $(a_1, a_2, .., a_n)$, wobei jedes a_i eine Teilanfrage darstellt und folgendes gilt: $a_j \rightarrow a_i \Rightarrow j < i$. n ist die Länge der Anfragesequenz.

Eine Anfragesequenz besteht aus einer Folge von Teilanfragen. Diese setzen sich wiederum aus bis zu vier separaten Anweisungen zusammen, wovon es sich bei der ersten Anweisung um eine CREATE-Anweisung, bei der zweiten um eine INSERT-Anweisung und bei der dritten um eine DROP-Anweisung handelt. Die letzte Anweisung ist eine beliebige Anweisung für die Datenbank. Sie ist in der Regel bei generierten Anfragesequenzen typischer Anwendungen nicht vorhanden, wird aber für die später erläuterten Optimierungsansätze benötigt. Dort

1. N wird hier und in den folgenden Definitionen als Symbol für die natürlichen Zahlen verwendet.

nimmt sie beispielsweise die Anweisungen zum Anlegen von Zugriffspfaden und Statistiken auf. Jede der Anweisungen einer Teilanfrage kann auch leer sein. Die Objekte, auf die in den einzelnen Anweisungen von Teilanfragen zugegriffen wird, können in zwei Klassen unterteilt werden. Die erste Klasse umfasst alle Objekte, die in der DWDB während der gesamten Ausführungszeit einer Anfragesequenz vorhanden sind. Auf diese Objekte wird nur lesend zugegriffen, sie führen deshalb nicht zu Abhängigkeiten zwischen verschiedenen Teilanfragen. Die zweite Klasse von Objekten sind solche, die durch die CREATE-Anweisung einer Teilanfrage erstellt werden. Solche Objekte implizieren eine Abhängigkeit zwischen einzelnen Teilanfragen einer Sequenz. Eine indirekte Abhängigkeit $a_j \rightarrow a_i$ liegt vor, wenn a_i zwar nicht direkt auf Objekte zugreift, die in a_j erstellt wurden, dafür aber in a_k erstellte Objekte verwendet, zu deren Erstellung wiederum auf die in a_j erstellten Objekte zugegriffen wurde. Indirekte Abhängigkeiten sind über beliebig viele Stufen hinweg möglich.

Durch die Definition einer Anfragesequenz ist noch nichts über die Ausführungsreihenfolge der in der Anfragesequenz enthaltenen Anweisungen ausgesagt. Diese Reihenfolge wird in der nachfolgend angegebenen Definition 17 durch eine Ausführungsgeschichte festgelegt. Die Funktion g_S weist jeder Anweisung in einer Anfragesequenz S eine natürliche Zahl zu. Die Ordnung über den natürlichen Zahlen wird also genutzt, um die Ausführungsreihenfolge zu bestimmen. Hier liegt die Annahme zu Grunde, dass bei der Ausführung mit der Anweisung begonnen wird, der die kleinste Zahl zugeordnet wurde. Es folgt die Anweisung, der die zweitkleinste Zahl zugeordnet wurde usw., bis schließlich alle Anweisungen bearbeitet wurden. Zu beachten ist, dass mit dieser Definition auch die parallele Ausführung einzelner Anweisungen zu beschreiben ist, da eine Position innerhalb der Ausführungsfolge auch mehrfach vergeben werden kann. In Definition 17 wird darüber hinaus festgelegt, welche Eigenschaften eine gültige Ausführungsreihenfolge aufweisen muss. Hierzu gehört, dass die einzelnen Anweisungen einer Teilanfrage in der richtigen Reihenfolge ausgeführt werden. Dies wird durch Bedingung (2) der Definition sichergestellt. Andererseits muss auch dafür gesorgt werden, dass bei Teilanfragen, die eine Abhängigkeit zueinander aufweisen, die notwendige Reihenfolge eingehalten wird. Bedingung (3) der Definition bewirkt hierzu, dass temporäre Objekte vor ihrer ersten Verwendung auch erstellt und mit Daten gefüllt werden. Bedingung (4) verhindert dagegen, dass ein in einer Teilanfrage erstelltes Objekt vor der letzten Verwendung in einer anderen Teilanfrage gelöscht wird. Hierbei muss nur auf die Anweisung o_i Bezug genommen werden, da durch Bedingung (2) impliziert wird, dass vor dieser auch

die CREATE-Anweisung c_i und die INSERT-Anweisung i_i der Teilanfrage i ausgeführt werden.

Definition 17: Gültige Ausführungsgeschichte einer Anfragesequenz

Eine Funktion g_S: $O_S \rightarrow N$ stellt eine *gültige Ausführungsgeschichte* für die Anfragesequenz S dar, wenn die folgenden vier Bedingungen erfüllt sind:

(1) O_S ist die Menge aller in S auftretenden Datenbankanweisungen.

(2) Für jedes i gilt: $g_S(c_i) < g_S(i_i) < g_S(o_i) < g_S(d_i)$.

(3) Für alle $i \leq n$ und $j \leq n$ gilt: $a_j \rightarrow a_i \Rightarrow g_S(o_j) < g_S(c_i)$.

(4) Für alle $i \leq n$ und $j \leq n$ gilt: $a_j \rightarrow a_i \Rightarrow g_S(d_j) > g_S(o_i)$.

Definition 18: Standardausführungsgeschichte einer Anfragesequenz

Für eine sequenzielle *Standardausführungsgeschichte* einer Anfragesequenz S der Länge $n \in N$ wird g_S für alle $i \leq n$ wie folgt definiert:

(1) $g_S(c_i) = 3 \cdot i$,

(2) $g_S(i_i) = 3 \cdot i + 1$,

(3) $g_S(o_i) = 3 \cdot i + 2$ und

(4) $g_S(d_i) = 3 \cdot (n + 1) + i$.

Die in der letzten Definition festgelegte Standardausführungsgeschichte beschreibt die typische Ausführungsreihenfolge für generierte Anfragesequenzen. Ohne weitere Optimierung kann davon ausgegangen werden, dass eine Anwendung die erstellten Anweisungen in der so festgelegten Reihenfolge an das DWDBS schickt. Das Grundprinzip ist: Alle Teilanfragen werden in der durch die Anfragesequenz gegebenen Reihenfolge ausgeführt. Innerhalb der einzelnen Teilanfragen wird zunächst die CREATE-, dann die INSERT-Anweisung und schließlich die zusätzliche Datenbankanweisung ausgeführt. Dies wird durch die Teile (1) bis (3) der Definition sichergestellt. Lediglich die Ausführung der DROP-Anweisungen weicht von diesem Muster ab. Diese werden, wie in Teil (4) der Definition festgelegt, für alle Teilanfragen ganz am Ende der Sequenz ausgeführt. Dadurch kann sichergestellt werden, dass alle erstellten Objekte bis zu ihrer letzten Verwendung erhalten bleiben, womit dann auch Bedingung (4) von Definition 17 erfüllt ist. Für Objekte, die das Endergebnis der Anfragesequenz enthalten sollen, darf natürlich keine DROP-Anweisung vorhanden sein. Insge-

samt besteht die Standardausführungsgeschichte für eine Anfragesequenz der Länge n aus maximal $4 \cdot n$ Anweisungen. Der Block mit den ersten $3 \cdot n$ Anweisungen enthält zu den einzelnen Teilanfragen die CREATE- und die INSERT-Anweisung sowie die zusätzliche Datenbankanweisung. Aus der Tatsache, dass hier drei Anweisungen pro Teilanfrage berücksichtigt werden müssen, resultiert die Zahl 3 in den einzelnen Teilen von Definition 18. Der Block mit den letzten n Anweisungen besteht aus sämtlichen DROP-Anweisungen. In beiden Blöcken wird die durch die Anfragesequenz vorgegebene Reihenfolge berücksichtigt.

Zu Beginn von Abschnitt 6.1 wurde bereits erläutert, dass Werkzeuge in der Regel nur vergleichsweise einfach strukturierte SQL-Anweisungen als Teilanfragen in einer Anfragesequenz generieren. Um darüber hinaus die Darstellung in den folgenden Abschnitten nicht zu komplex werden zu lassen, werden die weiteren Betrachtungen im Rahmen dieser Arbeit auf Anfragesequenzen beschränkt, deren Teilanfragen auf einer Untermenge der Syntax von SQL-92[1] aufbauen [Iso92]. Im Anhang sind die Syntaxdiagramme für die hier berücksichtigten SELECT-Anweisungen angegeben. Die Beschränkung auf einen Teil der Mächtigkeit von SQL-92 stellt im Rahmen dieser Arbeit keine Einschränkung dar, da hiermit alle exemplarisch betrachteten Anfragesequenzen dargestellt und alle relevanten Optimierungsansätze erläutert werden können.

6.1.2 Beispiele für Anfragesequenzen

Anfragesequenzen werden sowohl von OLAP- als auch von Data-Mining-Werkzeugen erstellt. Für beide Bereiche wurden in Kapitel 3 typische Fragestellungen vorgestellt. Einige der OLAP-Fragestellungen aus Abschnitt 3.4 wurden im Rahmen dieser Arbeit mit einem kommerziellen Werkzeug bearbeitet. Für dessen Verwendung war allerdings eine Überarbeitung des logischen Schemas der Daten notwendig, die dem Anwendungsszenarium ToPCHain zu Grunde liegen. Der relevante Ausschnitt des verwendeten Strategy-Schemas ist in Abbildung 18 dargestellt.

Bei der ersten Fragestellung, die hier näher betrachtet werden soll, handelt es sich um die in Abschnitt 3.4 vorgestellte Analyse 1. Die hierzu erstellte Anfragesequenz ist in Abbildung 19 dargestellt. Sie besteht aus den vier Teilanfragen a_1

1. Die im Rahmen dieser Arbeit verwendeten Bezeichnungen für die Versionen des SQL-Standards orientieren sich an [MS02]. SQL-92 bezeichnet den 1992 publizierten Standard. SQL:1999 bezeichnet den aktuellen Standard.

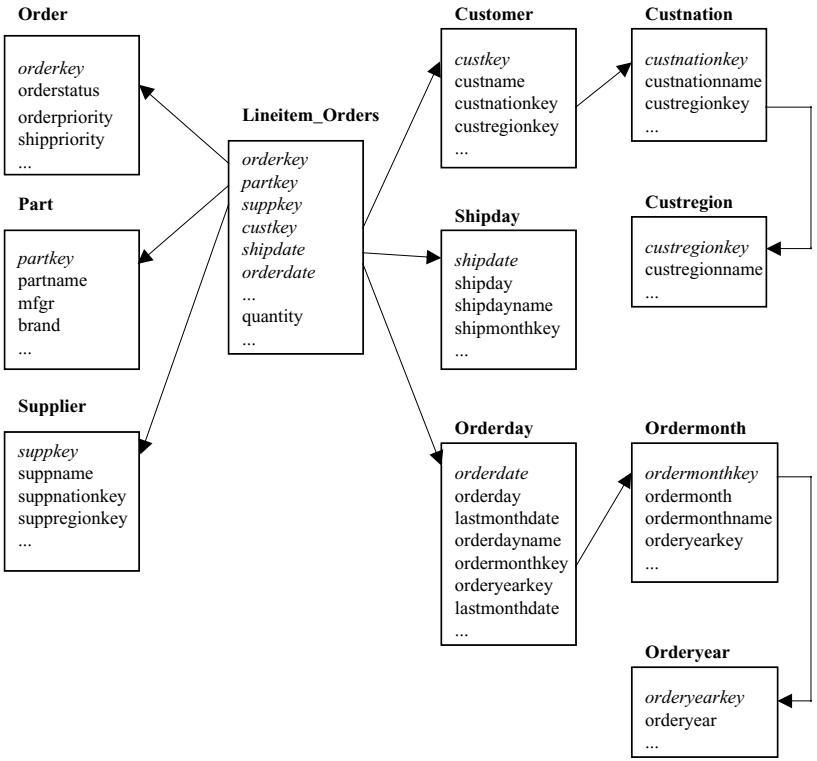

Abbildung 18: Strategy-Schema für ToPCHain

bis a_4 und ist in der durch eine Standardausführungsgeschichte gegebenen Reihenfolge dargestellt. Darum wird für jede der Teilanfragen zunächst die CREATE-Anweisung angegeben, mit deren Hilfe jeweils eine temporäre Tabelle erstellt wird. Zur Vereinfachung sind die Details dieser CREATE-Anweisungen nicht aufgeführt. Auf die CREATE-Anweisung folgt jeweils die INSERT-Anweisung, die dafür sorgt, dass Daten in die temporäre Tabelle eingefügt werden. Zusätzliche Anweisungen, in den Definitionen als o_i bezeichnet, sind in diesem Beispiel nicht enthalten. Am Ende der Ausführung stehen die DROP-Anweisungen, die alle temporären Tabellen, die nicht zum Lesen des Ergebnisses benötigt werden, wieder löschen. Die Struktur der Anfragesequenz ist in Abbildung 20 im Überblick angegeben. Hierbei wird die Notation der Definitionen 10 bis 16 ver-

```
CREATE TABLE A1 ...;                                                    /* c1 */

INSERT INTO A1 (orderyearkey, ordermonthkey, partkey, sumquantity)      /* i1 */
SELECT    od.orderyearkey, od.ordermonthkey, lo.partkey, SUM(lo.quantity)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate
AND       od.ordermonthkey IN (199401,199402)
GROUP BY od.orderyearkey, od.ordermonthkey, lo.partkey;

CREATE TABLE A2 ...;                                                    /* c2 */

INSERT INTO A2 (ordermonthkey, partkey, sumquantity)                    /* i2 */
SELECT    od.ordermonthkey, lo.partkey, SUM(lo.quantity)
FROM      lineitem_orders lo, orderday od
WHERE     od.lastmonthdate = lo.orderdate
AND       od.ordermonthkey IN (199401, 199402)
GROUP BY od.ordermonthkey, lo.partkey;

CREATE TABLE A3 ...;                                                    /* c3 */

INSERT INTO A3 (ordermonthkey, ordermonthname, orderyearkey, orderyear,  /* i3 */
                partkey, partname, sumquantity, lmsumquantity, incrquantity, incrquantity2)
SELECT    om.ordermonthkey, om.ordermonthname, oy.orderyearkey, oy.orderyear, pa.partkey,
          pa.partname, a1.sumquantity, A2.sumquantity, A1.sumquantity-A2.sumquantity,
          (A1.sumquantity - A2.sumquantity) / A2.sumquantity
FROM      A1, A2, ordermonth om, orderyear oy, part pa
WHERE     A1.ordermonthkey = A2.ordermonthkey
AND       A1.partkey = A2.partkey
AND       A1.ordermonthkey = om.ordermonthkey
AND       A1.orderyearkey = oy.orderyearkey
AND       A1.partkey = pa.partkey;

CREATE TABLE A4 ...;                                                    /* c4 */

INSERT INTO A4 (ordermonthkey, ordermonthname, orderyearkey, orderyear,  /* i4 */
                partkey, partname, sumquantity, lmsumquantity, incrquantity, incrquantity2)
SELECT    A3.ordermonthkey, A3.ordermonthname, A3.orderyearkey, A3.orderyear, A3.partkey,
          A3.partname, A3.sumquantity, A3.lmsumquantity, A3.incrquantity, A3.incrquantity2
FROM      A3
WHERE     A3.incrquantity2 >= 98;

DROP TABLE A1;                                                          /* d1 */

DROP TABLE A2;                                                          /* d2 */

DROP TABLE A3;                                                          /* d3 */
```

Abbildung 19: Anfragesequenz A (zu Analyse 1)

$S = (a_1, a_2, a_3, a_4)$
$a_1 = (c_1, i_1, - , d_1)$
$a_2 = (c_2, i_2, - , d_2)$
$a_3 = (c_3, i_3, - , d_3)$
$a_4 = (c_4, i_4, - , -)$

(a) Struktur

$g_S(c_1) = 3$
$g_S(i_1) = 4$
$g_S(c_2) = 6$
$g_S(i_2) = 7$
$q_S(c_3) = 9$
$g_S(i_3) = 10$
$g_S(c_4) = 12$
$g_S(i_4) = 13$
$g_S(d_1) = 16$
$g_S(d_2) = 17$
$g_S(d_3) = 18$

(b) Ausführungsgeschichte

Abbildung 20: Struktur und Ausführungsgeschichte der Anfragesequenz A

wendet. Für die Anfragesequenz der Länge 4 sind jeweils die Bestandteile der Teilanfragen a_1 bis a_4 aufgeführt. In der Abbildung ist auch die der Definition 18 entsprechende Standardausführungsgeschichte g_S angegeben.

Das Ziel der Analyse 1 ist es, alle Produkte zu identifizieren, bei denen der monatliche Absatz gegenüber ausgewählten Vergleichsmonaten am stärksten gestiegen ist. Für die exemplarische Anfragesequenz wurden die Monate Januar und Februar 1994 ausgewählt. In Abbildung 19 ist dies in den WHERE-Klauseln der INSERT-Anweisungen i_1 und i_2 an der Einschränkung des Attributs *ordermonthkey* auf die Werte *199401* und *199402* zu erkennen. Die erste Teilanfrage berechnet die monatlich verkauften Stückzahlen für die ausgewählten Monate, während in der zweiten Teilanfrage die jeweiligen Summen für die Vormonate gebildet werden. Der Bezug zum Vormonat wird in der INSERT-Anweisung i_2 durch die JOIN-Bedingung über dem Attribut *lastmonthdate* realisiert. In der dritten Teilanfrage wird schließlich die Differenz zwischen den jeweils zu vergleichenden monatlichen Zahlen gebildet. Aus diesem Ergebnis werden in INSERT-Anweisung i_4 diejenigen Produkte selektiert, für die besonders große Differenzen verzeichnet wurden. Dies wird durch das Prädikat *A3.incrquantity2 >= 98* erreicht. Der Wert 98 kann sich hier einerseits direkt aus den Angaben des Benutzers ergeben. Andererseits kann dieser auch das Ergebnis der Umrechnung der relativen Angabe des Benutzers in eine absolute Angabe sein. Als Endergebnis der Anfragesequenz enthält Tabelle *A4* die gesuchten Monate und Produkte

einschließlich der umgesetzten Stückzahlen und der Differenz zum jeweiligen Vormonat.

Die in Abbildung 19 dargestellte Anfragesequenz wird in der weiteren Arbeit als *Anfragesequenz A* bezeichnet. Vor der Beschreibung weiterer Beispiele für Anfragesequenzen, soll hier die im Folgenden verwendete vereinfachte Darstellungsform erläutert werden. Sie besteht einerseits darin, dass von jeder Teilanfrage lediglich die INSERT-Anweisung dargestellt ist. CREATE- und DROP-Anweisungen werden im weiteren Verlauf der Arbeit nie explizit dargestellt, sind aber immer Teil der Anfragesequenz. Darüber hinaus wurden in den INSERT-Anweisungen der Anfragesequenzen Vereinfachungen vorgenommen, wenn dies das Verständnis für die Funktion der einzelnen Teilanfragen fördert, aber keine grundlegende Änderung der jeweiligen Teilanfrage darstellt. Ein typisches Beispiel hierfür sind CASE-Anweisungen, die teilweise in den von Werkzeugen erstellten Anfragesequenzen auftreten, um Sonderfälle abzufangen. Im Text wird auf die einzelnen Teilanfragen mit der Bezeichnung der temporären Tabelle Bezug genommen, die in der jeweiligen Teilanfrage erstellt und gefüllt wird. Die Anfragesequenz in Abbildung 19 besteht also aus den Teilanfragen *A1*, *A2*, *A3* und *A4*.

Eine sehr einfache Anfragesequenz ergibt sich für die Analyse 2. Sie besteht lediglich aus der Teilanfrage *B1*. Diese reicht aus, um für ausgewählte Produkte (*pa.partkey <= 3*) den Absatz in den verschiedenen Kundenregionen für das Jahr 1994 zusammenzustellen. Die Anfragesequenz ist in Abbildung 21 dargestellt. Sie wird in der weiteren Arbeit als *Anfragesequenz B* bezeichnet.

Als drittes Beispiel soll hier schließlich eine Anfragesequenz herangezogen werden, die für die Analyse 3 zusammengestellt werden kann. Das Ziel dieser Analyse ist es, die wichtigsten Kunden der ToPCHain zu ermitteln, die in einem gewählten Zeitraum einen bestimmten Mindestumsatz vorweisen können. Zusätzlich soll bei den ausgewählten Kunden eine besonders geringe Standardabweichung des Umsatzes vorhanden sein. Die erstellte Anfragesequenz ist in Abbildung 22(a) und Abbildung 22(b) angegeben. In den ersten drei Teilanfragen wird jeweils der Umsatz pro Kunde für die drei ausgewählten Jahre 1992, 1993 und 1994 berechnet. In der Teilanfrage *C4* werden die Kunden ausgewählt, für die in allen drei Jahren ein Umsatz größer als 500000 registriert wurde, und die damit das Mindestumsatzkriterium erfüllen. In den folgenden beiden Teilanfragen *C5* und *C6* wird für die ausgewählten Kunden jeweils eine der beiden für die

```
INSERT INTO B1 (partkey, partname, custregionkey, custregionname, custnationkey,
                custnationname, orderyearkey, orderyear, sumquantity)
SELECT   pa.partkey, pa.partname, cr.custregionkey, cr.custregionname, cn.custnationkey,
         cn.custnationname, oy.orderyearkey, oy.orderyear, SUM(lo.quantity)
FROM     lineitem_orders lo, customer cu, orderday od, part pa, custregion cr, custnation cn,
         orderyear oy
WHERE    cu.custkey        = lo.custkey
AND      od.orderdate      = lo.orderdate
AND      lo.partkey        = pa.partkey
AND      cu.custregionkey  = cr.custregionkey
AND      cu.custnationkey  = cn.custnationkey
AND      od.orderyearkey   = oy.orderyearkey
AND      pa.partkey        <= 3
AND      oy.orderyearkey   = 1994
GROUP BY pa.partkey, cr.custregionkey, cn.custnationkey, oy.orderyearkey;
```

Abbildung 21: Anfragesequenz B (zu Analyse 2)

Bewertung der Standardabweichung relevanten Größen berechnet. Diese werden in *C7* mit den Umsatzangaben zu den einzelnen Kunden kombiniert und in *C8* schließlich die Kunden mit der geforderten geringen Standardabweichung ausgewählt. *C8* stellt dann auch das Endergebnis für die Analyse 3 dar. *Anfragesequenz C* wird in der weiteren Arbeit als Bezeichnung für diese Anfragesequenz verwendet.

6.2 Strategien für einen DSS-Optimierer

Die Aufgabe des DSS-Optimierers ist die Optimierung der Verarbeitung von Anfragesequenzen. Grundsätzlich handelt es sich hierbei um ein Problem der Anfrageoptimierung. Insbesondere im Data-Warehouse-Bereich kann ein breites Spektrum von *Optimierungsansätzen* betrachtet werden. Die im Rahmen dieser Arbeit relevanten Optimierungsansätze können je nach Gegenstand des Ansatzes in drei Gruppen unterteilt werden:

- *Gruppe 1: Modifikation der Daten*
  Optimierungsansätze, die die zu Grunde liegenden Daten im Data Warehouse so verändern, dass gegebene Anfragesequenzen effizienter verarbeitet werden können. Zu dieser Gruppe gehören auch alle Ansätze, die Veränderungen am physischen Schema des Data Warehouse vornehmen.
- *Gruppe 2: Modifikation der Ausführung von Anfragesequenzen*
  Optimierungsansätze, die die Art der Ausführung von Anfragesequenzen

beeinflussen. Die Teilanfragen der Sequenzen selbst bleiben hierbei weitestgehend unverändert.

- *Gruppe 3: Modifikation der Anfragesequenzen*
  Optimierungsansätze, die die Anfragesequenzen verändern, so dass die geänderten Versionen effizienter verarbeitet werden können.

Verschiedene, zu diesen Gruppen gehörende Ansätze, werden in den folgenden Abschnitten erläutert und bezüglich ihrer Eignung für einen DSS-Optimierer analysiert. Insbesondere findet hierbei eine Bewertung anhand der folgenden Kriterien statt:

```
INSERT INTO C1 (custkey, turnover1992)
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey = 1992
GROUP BY lo.custkey;

INSERT INTO C2 (custkey, turnover1993)
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey = 1993
GROUP BY lo.custkey;

INSERT INTO C3 (custkey, turnover1994)
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey = 1994
GROUP BY lo.custkey;

INSERT INTO C4 (custkey, turnover1992, turnover1993, turnover1994)
SELECT    C1.custkey, C1.turnover1992, C2.turnover1993, C3.turnover1994
FROM      C1, C2, C3
WHERE     C1.custkey = C2.custkey          AND C1.custkey = C3.custkey
AND       C1.turnover1992 >= 500000        AND C2.turnover1993 >= 500000
AND       C3.turnover1994 >= 500000;

INSERT INTO C5 (custkey, stddeviation)
SELECT    lo.custkey, STDDEV(lo.endprice)
FROM      lineitem_orders lo, orderday od, C4
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey IN (1992, 1993, 1994)
AND       lo.custkey = C4.custkey
GROUP BY lo.custkey;
```

Abbildung 22(a): Anfragesequenz C (zu Analyse 3)

```
INSERT INTO C6 (custkey, stddeviation)
SELECT    lo.custkey, STDDEV(lo.endprice) / AVG(lo.endprice)
FROM      lineitem_orders lo, orderday od, C4
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey IN (1992, 1993, 1994)
AND       lo.custkey = C4.custkey
GROUP BY lo.custkey;

INSERT INTO C7 (custkey, custname, stddev1, stddev2, turnover1992, turnover1993,
                turnover1994)
SELECT    cu.custkey, cu.custname, C5.stddeviation, C6.stddeviation, C1.turnover1992,
          C2.turnover1993, C3.turnover1994
FROM      C1, C2, C3, C5, C6, customer cu
WHERE     C5.custkey = C1.custkey          AND C5.custkey = C2.custkey
AND       C5.custkey = C3.custkey          AND C5.custkey = C6.custkey
AND       C5.custkey = cu.custkey;

INSERT INTO C8 (custkey, custname, stddev1, stddev2, turnover1992, turnover1993,
                turnover1994)
SELECT    C7.custkey, C7.custname, C7.stddev1, C7.stddev2, C7.turnover1992,
          C7.turnover1993, C7.turnover1994
FROM      C7
WHERE     C7.stddev2 <= 0.66794004454646;
```

Abbildung 22(b): Anfragesequenz C (zu Analyse 3, Fortsetzung)

- die *Technologieverfügbarkeit*,
- das *Optimierungspotenzial*,
- der *Einflussbereich* und
- die *Entscheidungsgrundlage* für die einzelnen Optimierungsansätze sowie
- die Eignung des DSS-Optimierers für die *Umsetzung* der Optimierungsansätze.

Die Erläuterung des Standes der Technik ist erforderlich, um festzustellen, ob alle notwendigen Technologien für eine Realisierung des jeweiligen Optimierungsansatzes in einem DSS-Optimierer verfügbar sind (Kriterium *Technologieverfügbarkeit*). Ergänzend muss analysiert werden, in wieweit sich ein Optimierungsansatz auch oder gerade für die Optimierung von Anfragesequenzen eignet. Das *Optimierungspotenzial* der einzelnen Ansätze im Hinblick auf Anfragesequenzen muss also bewertet werden. Bei der Betrachtung des *Einflussbereichs* der Optimierungsansätze wird einerseits etwas darüber ausgesagt, ob ein Optimierungsansatz lediglich eine Anfragesequenz im Fokus hat, oder ob auch

positive Einflüsse auf nachfolgende Anfragesequenzen zu erwarten sind. Andererseits muss auch berücksichtigt werden, in welcher Weise negative Auswirkungen auf folgende Anfragesequenzen oder gar andere Anwendungen auf einem Data Warehouse möglich und zu erwarten sind. Jeder der Optimierungsansätze setzt eine ganze Reihe von Informationen voraus, auf deren Grundlage die im Rahmen der Optimierung notwendigen Entscheidungen getroffen werden können. In wieweit diese im DSS-Optimierer vorhanden sind bzw. vorhanden sein können, muss darum ebenfalls untersucht werden (Kriterium *Entscheidungsgrundlage*). Als letztes Kriterium zur Bewertung der verschiedenen Optimierungsansätze muss schließlich betrachtet werden, ob es andere Systemkomponenten im Rahmen der erweiterten Systemarchitektur für Analyseanwendungen gibt, in denen die *Umsetzung* der Optimierungsansätze im Vergleich zur Umsetzung in einem DSS-Optimierer vorteilhaft erscheint. Dies hängt insbesondere auch davon ab, in welchem Grad ein Optimierungsansatz von den Spezifika des Datenbanksystems abhängt, das die Anfragesequenz schließlich ausführen soll.

6.2.1 Parallele Ausführung von Teilsequenzen

Parallelität spielt bei der Ausführung von Anfragen an eine Datenbank schon seit geraumer Zeit eine bedeutende Rolle. Eine sequenzielle Verarbeitung von Anfragen, die auf große Datenmengen zugreifen und/oder sehr berechnungsintensiv sind, ist in vielen Fällen nicht effizient möglich.

In der Literatur werden sowohl Aspekte der Parallelisierung von Anfrageplänen im Rahmen einer eigenständigen Optimierungsphase [HS93] als auch die Integration der Parallelisierungsentscheidungen in die kostenbasierte Anfrageoptimierung betrachtet [WFA95] [NM98]. Insbesondere im letzten Fall ist es das Ziel, eine geeignete Reduzierung des potenziell sehr umfangreichen Suchraums alternativer Ausführungspläne zu erreichen [SD90] [HS93] sowie eine optimierte Strategie für das Durchlaufen des Suchraums zu finden [LVZ93]. Im Rahmen der Parallelisierung von Anfrageplänen muss unter anderem entschieden werden, welche Teile eines Anfrageplans voneinander unabhängig sind. Diese können dann parallel ausgeführt werden, was insbesondere auch für Join-Operatoren wichtig ist [LST91]. Für einzelne Operatoren des Anfrageplans muss jeweils der optimale Parallelisierungsgrad bestimmt werden. Hierbei fließen sowohl Eigenschaften des Anfrageplans, wie auch der Laufzeitumgebung sowie die vorliegende Datenpartitionierung in das Kostenmodell ein [GGS96].

Ein Teilziel paralleler Datenbanksysteme ist es somit, Teile eines Ausführungsplans zu identifizieren, die parallel ausgeführt werden können. Dies kann einerseits ein einzelner Operator sein, der parallel auf mehreren Prozessoren ausgeführt wird, wobei auf jedem Prozessor ein getrennter Datenstrom bearbeitet wird. Andererseits kann der Ausführungsplan auch unterschiedliche Teile enthalten, die unabhängig voneinander sind und damit parallelisiert werden können. In jedem Fall handelt es sich um einen Ansatz, bei dem Intra-Operator- und Intra-Anfrageparallelität ausgenutzt wird. In der erweiterten Systemarchitektur für Analyseanwendungen werden die einzelnen SQL-Anfragen durch das zu Grunde liegende Datenbanksystem ausgeführt, ohne dass der DSS-Optimierer auf dessen Parallelisierungsstrategie Einfluss nimmt. Für den DSS-Optimierer besteht darum auch keine Möglichkeit, die Nutzung von Intra-Anfrageparallelität zu erzwingen.

Es gibt allerdings eine andere Möglichkeit, innerhalb des DSS-Optimierers eine auf Anfrageparallelität basierende Strategie einzusetzen. Sofern nicht alle Teilanfragen einer Anfragesequenz direkt voneinander abhängen, können einzelne Teilanfragen gleichzeitig an das Data Warehouse geschickt werden. Abbildung 23 zeigt die Abhängigkeiten, die innerhalb der Anfragesequenzen *A*, *B* und *C* bestehen. Die Knoten der Abhängigkeitsgraphen stellen jeweils die Teilanfragen dar. Die Namensgebung für diese orientiert sich an der Bezeichnung für die temporären Objekte, die durch die einzelnen CREATE-Anweisungen erstellt werden. Ein Pfeil von Teilanfrage *X* zur Teilanfrage *Y* bedeutet, dass die in Teilanfrage *X* erstellte temporäre Tabelle oder Sicht in Teilanfrage *Y* verwendet wird. Im Sinne der Definition 15 besteht also die Abhängigkeit $X \rightarrow Y$. Aus der Abbildung kann man ablesen, dass die Ausführung von *Anfragesequenz A* so modifiziert werden kann, dass die Teilanfragen *A1* und *A2* gleichzeitig an das zu Grunde liegenden Data Warehouse geschickt werden. In der *Anfragesequenz C* ergibt sich diese Möglichkeit zur Parallelisierung gleich an zwei unterschiedlichen Stellen. Zunächst können die drei ersten Teilanfragen gleichzeitig ausgeführt werden. Nachdem die temporäre Tabelle *C4* erstellt wurde, können auch die Teilanfragen *C5* und *C6* gleichzeitig zur Ausführung gebracht werden.

Die hier beschriebene gleichzeitige Ausführung von Teilanfragen einer Anfragesequenz stellt aus Sicht des Data Warehouse Inter-Anfrageparallelität dar. Der Zusammenhang, der zwischen den beiden Anfragen besteht, ist für das Data Warehouse völlig transparent. Lediglich der DSS-Optimierer muss die Zusam-

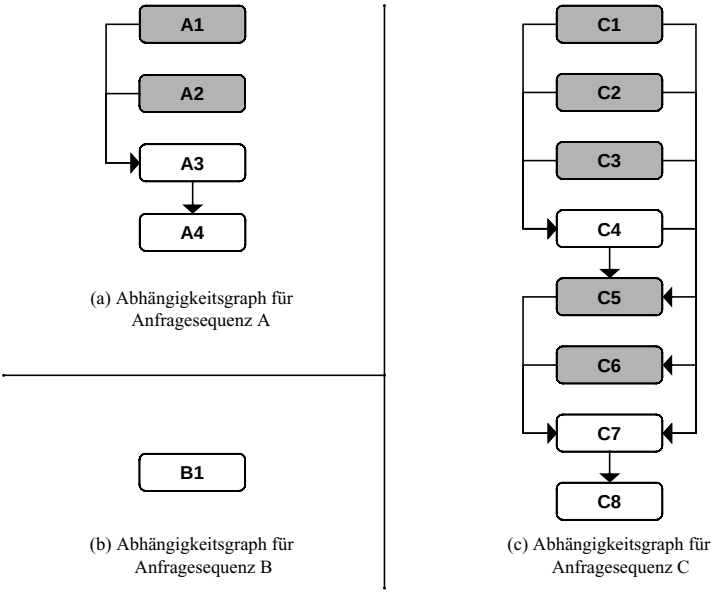

Abbildung 23: Parallelität in Anfragesequenzen

menhänge zwischen den Teilanfragen einer Anfragesequenz analysieren. Ein Zusammenhang besteht zwischen Teilanfragen insoweit, als sie zu derselben Anfragesequenz gehören. Dies allein impliziert allerdings noch keine Abhängigkeit, die eine Parallelausführung ja gerade verhindern würde. Die durch die parallele Ausführung von Teilanfragen erreichte Parallelisierung wird hier als *Intra-Anfragesequenz-Parallelität* bezeichnet. Dieser Begriff wird hier ebenfalls als Bezeichnung für die zugehörige Strategie des DSS-Optimierers verwendet.

Das Potenzial zur Parallelisierung der Anfragesequenz kann einfach identifiziert werden. Hierzu ist lediglich der Aufbau eines Abhängigkeitsgraphen G notwendig. Die einzelnen Teilanfragen der Sequenz werden sequenziell analysiert. Für jede Teilanfrage a_k wird ein Knoten des Graphen G mit der Bezeichnung a_k erstellt. In jeder INSERT-Anweisung i_k wird die FROM-Klausel analysiert. Für jede in der Teilanfrage a_i erstellte Tabelle oder Sicht, die in der FROM-Klausel von i_k verwendet wird, wird im Graph eine Kante zwischen Knoten a_i und Knoten a_k eingeführt. Zwei Teilanfragen a_i und a_j können dann parallel gestartet werden, wenn es in dem gerichteten Graphen G weder einen Pfad von a_i nach a_j noch

einen Pfad von a_j nach a_i gibt. Dies ist gleichbedeutend damit, dass keine Abhängigkeit zwischen den Teilanfragen a_j und a_i existiert, d.h., es gilt weder $a_i \rightarrow a_j$ noch $a_j \rightarrow a_i$.

Der DSS-Optimierer ist aus der Sicht des DWDBS eine gewöhnliche Anwendung. Als solche hat er die Möglichkeit, gleichzeitig mehrere Verbindungen zur Datenbank des Data Warehouse aufzubauen und über diese parallel die auszuführenden Teilanfragen abzusetzen. Diese werden dann jeweils in getrennten Transaktionen ausgeführt. Die parallele Ausführung von Teilanfragen einer Anfragesequenz kann nur dann einen Performanzvorteil aufweisen, wenn sich diese nicht gegenseitig durch den Zugriff auf gemeinsame Ressourcen blockieren. Dies können systeminterne Ressourcen des DWDBS, wie Sperren oder Protokolldateien, sein. Deren gemeinsame Nutzung durch die parallel gestarteten Teilanfragen ist unvermeidlich und es ist Aufgabe des DWDBS, die Inter-Anfrageparallelität möglichst effizient zu unterstützen. Durch die Auswahl der parallel zu startenden Teilanfragen wird sichergestellt, dass bei deren Start alle temporären Daten zur Verfügung stehen. Insbesondere wird verhindert, dass eine Teilanfrage gestartet wird, die eine temporäre Tabelle erstellt, deren Inhalt bereits von einer parallel gestarteten Teilanfrage benötigt wird. Darüber hinaus gibt es noch die im Data Warehouse vorhandenen Basistabellen, die von unterschiedlichen Teilanfragen als Ressourcen vorausgesetzt werden. Auf diese wird allerdings nur lesend zugegriffen. Damit können sich Teilanfragen auch durch die gemeinsame Nutzung dieser Ressourcen nicht gegenseitig blockieren.

Insgesamt konnte hier gezeigt werden, dass es sich bei der parallelen Ausführung von Teilanfragen einer Anfragesequenz um einen Ansatz handelt, auf dem eine für den DSS-Optimierer geeignete Strategie aufgebaut werden kann. Die Auswahl parallel auszuführender Teilanfragen kann auf der Grundlage eines einfach zu erstellenden Abhängigkeitsgraphen effizient ermittelt werden. Abhängig von den im DWDBS zur Verfügung stehenden Prozessoren und der Ein- und Ausgabekapazität kommt der Vorteil der parallelen Ausführung voll zum Tragen, da zwischen den einzelnen Teilanfragen keine weiteren Abhängigkeiten bestehen, die ein gegenseitiges Blockieren zur Folge haben. Die Strategie, die offensichtlich in die zweite Gruppe der möglichen Optimierungsansätze einzuordnen ist, soll hier nochmals kurz charakterisiert werden. Wie auch bei den folgenden Optimierungsansätzen enthält diese Kurzbeschreibung jeweils einen umgangssprachlichen Teil sowie eine Beschreibung, die sich auf die formalen Definitionen in Abschnitt 6.1 bezieht. Die letzte Zeile der folgenden Kurzbeschreibung legt fest,

wie eine Ausführungsgeschichte bei der Nutzung von Intra-Anfragesequenz-Parallelität aussehen soll. Es wird jeweils den beiden CREATE-Anweisungen, den beiden INSERT-Anweisungen sowie den beiden zusätzlichen Anweisungen zweier unabhängiger Teilanfragen dieselbe Zahl, und damit dieselbe Position in der Ausführungsreihenfolge, zugeordnet.

Optimierungsansatz: Intra-Anfragesequenz-Parallelität

Analysiere die Abhängigkeiten, die zwischen den einzelnen Teilanfragen einer Anfragesequenz bestehen und führe Teilanfragen, die nicht voneinander abhängen, gleichzeitig aus.

Für die Anfragesequenz $S = (a_1, a_2, ..., a_n)$ wird die Ausführungsgeschichte g_S so modifiziert, dass für jedes unabhängige Paar von Teilanfragen a_i und a_j (es gilt weder $a_i \rightarrow a_j$ noch $a_j \rightarrow a_i$) folgendes gilt: $g_S(c_i) = g_S(c_j)$, $g_S(i_i) = g_S(i_j)$, $g_S(o_i) = g_S(o_j)$

6.2.2 Nutzung zusätzlicher statistischer Informationen

Die Anfrageverarbeitung in modernen Datenbanksystemen verwendet Kostenmodelle, um für eine Anfrage einen Plan mit möglichst geringen Ausführungskosten zu identifizieren. Bei der Schätzung dieser Kosten spielen statistische Informationen zu den in der Anfrage verwendeten Tabellen eine zentrale Rolle. Zu den statistischen Informationen, auf die Anfrageoptimierer in der Regel zurückgreifen, gehören beispielsweise:

- die Anzahl der Zeilen in den einzelnen Tabellen,
- die Anzahl unterschiedlicher Werte pro Attribut sowie
- die Werteverteilung für einzelne Attribute.

Diese statistischen Informationen werden unter anderem dazu genutzt, die Anzahl der Tupel für die einzelnen, im Rahmen der Anfrageverarbeitung anfallenden Zwischenergebnisse abzuschätzen. Das Ziel unterschiedlicher Arbeiten in diesem Bereich ist es, die Methoden für diese Abschätzungen, insbesondere beim Auftreten mehrerer Joins und Prädikate, zu verbessern [Lyn88] [Gel93] [SS94]. Alle diese Methoden setzten möglichst aktuelle und präzise Statistiken voraus. Deshalb wurden in den letzten Jahren diese Arbeiten dahingehend ausgedehnt,

die verfügbaren statistischen Informationen im Rahmen der Anfrageverarbeitung zu aktualisieren [SL+01]. Während bei der Anfrageverarbeitung auf einzelne Tabellen zugegriffen wird, werden die für diese Tabellen verfügbaren Statistiken überprüft und gegebenenfalls aktualisiert. Häufig wird in diesem Zusammenhang von einem lernenden Optimierer gesprochen. Bei dieser Vorgehensweise wird zwar nicht die Performanz der aktuell in der Ausführung befindlichen Anfrage verbessert, jedoch hat der Anfrageoptimierer für alle nachfolgenden Anfragen potenziell exaktere Statistiken als Grundlage für seine Entscheidungen zur Verfügung.

Welche Rolle spielt nun die Bereitstellung von statistischen Informationen bei der Verarbeitung von Anfragesequenzen? Für die Optimierung einer Teilanfrage sollten diese Informationen sowohl für alle verwendeten Basistabellen, wie auch für alle verwendeten temporären Tabellen, zur Verfügung stehen. Da die Basistabellen permanent im Data Warehouse verfügbar sind, kann bei diesen davon ausgegangen werden, dass durch die Administratoren des DWDBS für eine regelmäßige Aktualisierung der Statistiken gesorgt wird. Anders sieht es dagegen bei den temporären Tabellen aus, die im Rahmen der Ausführung einer Anfragesequenz erstellt werden. Abhängig vom zu Grunde liegenden Datenbanksystem kann nicht angenommen werden, dass beim Laden der Daten in die einzelnen temporären Tabellen auch automatisch die relevanten statistischen Informationen gesammelt werden. Beispielsweise gibt es bei DB2 Version 7.2 und Oracle8i separate Befehle, die das Aktualisieren der statistischen Informationen für einzelne Tabellen erst anstoßen. Für die Tabelle *A1* wird dies in Oracle8i durch den folgenden Befehl ausgelöst: *ANALYZE TABLE A1 COMPUTE STATISTICS*. In jedem System gibt es für den entsprechenden Befehl eine Reihe von Optionen, mit deren Hilfe festgelegt wird, wie detailliert die Statistiken erfasst werden sollen. Diese Angaben haben erheblichen Einfluss auf den Aufwand der Statistikerstellung [Ora00b] [Ibm01b].

Der Ansatz eines lernenden Optimierers hilft an dieser Stelle nur wenig weiter. Bei der Verarbeitung einer Teilanfrage können zwar die Statistiken für die in der Teilanfrage verwendeten Tabellen aktualisiert werden. Für die Basistabellen des Data Warehouse bleibt dies allerdings ohne Auswirkung, wenn man davon ausgeht, dass administrationsseitig für aktuelle Statistiken für diese Tabellen gesorgt wird. Für bereits vorhandene temporäre Tabellen wird mit der Ausführung einer Teilanfrage zwar die Statistik angepasst, unter Umständen wird auf diese in den folgenden Teilanfragen aber gar nicht mehr zugegriffen. Da im Rahmen von

Anfragesequenzen erstellte temporäre Tabellen in der Regel nicht von anderen Anfragesequenzen benutzt werden, steht hier der Anpassung der Statistik kein Gewinn gegenüber. Dies kann an der *Anfragesequenz A* verdeutlicht werden. In Abbildung 23 ist zu sehen, dass die vierte Teilanfrage die Ergebnisse aus Teilanfrage *A3* verwendet. Damit kann ein lernender Optimierer bei der Ausführung von *A4* die Statistik für *A3* erstmals anpassen. Da die Anfragesequenz mit *A4* aber bereits abgeschlossen ist, haben die neu berechneten Statistiken keine Auswirkungen, wenn man einmal davon absieht, dass durch sie die Verarbeitung von *A4* mit zusätzlichem Aufwand verbunden ist.

Für den DSS-Optimierer besteht nun eine mögliche Optimierungsstrategie darin, dafür zu sorgen, dass beim Start der Ausführung einer Teilanfrage in einer Anfragesequenz jeweils alle relevanten statistischen Informationen zu den in der Sequenz bereits erstellten temporären Tabellen vorliegen. Hierzu muss die Anfragesequenz um die Befehle erweitert werden, die notwendig sind, um das Erstellen der Statistiken anzustoßen. Da mit dieser Erweiterung die wesentlichen Anweisungen der Teilanfrage einer Anfragesequenz (CREATE, INSERT, DROP) nicht verändert werden, kann dieser Ansatz zur Gruppe 2, d.h. zu den Optimierungsmöglichkeiten, die lediglich die Ausführung einer Sequenz beeinflussen, gerechnet werden.

Wenn der DSS-Optimierer diesem Optimierungsansatz folgen soll, dann ist hierfür die Information über das zu Grunde liegende Datenbanksystem notwendig. Nur so kann der für das jeweilige DWDBS korrekte Befehl zur Aktualisierung von Statistiken verwendet werden. Jedoch kann das Einfügen solcher Befehle unterbleiben, sofern das zu Grunde liegende Datenbanksystem automatisch statistische Informationen zu den angelegten temporären Tabellen bereitstellt.

Dieser Optimiereransatz ist unabhängig von der Anwendung, die eine Anfragesequenz generiert. Darüber hinaus ist auch keine detaillierte Analyse der Anfragesequenz notwendig. Der DSS-Optimierer muss aus der gegebenen Anfragesequenz lediglich die temporären Tabellen identifizieren, für die die Bereitstellung der Statistiken angestoßen werden sollte.

Die Auswirkungen dieses Optimierungsansatzes sind weitgehend auf die Ausführung der Anfragesequenz selbst beschränkt. Da zusätzliche Statistiken nur für temporäre Tabellen der Anfragesequenz erstellt werden, die in der Regel nicht von anderen Anwendungen oder bei der Ausführung anderer Anfragesequenzen

verwendet werden, kann auch nur die aktuell vom DSS-Optimierer bearbeitete Anfragesequenz von der vorgenommenen Optimierung profitieren. Negative Auswirkungen auf andere Anwendungen sind nur insofern möglich, als auch das Erstellen der Statistiken Ressourcen des DWDBS in Anspruch nimmt.

Es gibt eine Reihe von Ansätzen zur Verfeinerung dieser Strategie. Einerseits kann der DSS-Optimierer den Aufwand für die Bereitstellung der statistischen Informationen abschätzen. Auf der Grundlage dieser Schätzung können die Befehle zur Erstellung der Statistik dann nur bei ausgewählten Teilanfragen der Anfragesequenz eingefügt werden. Andererseits kann die Bereitstellung der Statistik unter Umständen parallel zu anderen Befehlen der Anfragesequenz erfolgen. In Abbildung 23 ist beispielsweise zu sehen, dass im Rahmen der Verarbeitung der *Anfragesequenz A* die statistischen Informationen zu der in der ersten Teilanfrage erstellten Tabelle *A1* erst zu Beginn der Teilanfrage *A3* und nicht bereits mit dem Start der Teilanfrage *A2* relevant sind. Die in Abschnitt 6.2.1 beschriebene Analyse der Abhängigkeiten innerhalb einer Anfragesequenz kann also auch hier verwendet werden.

Insgesamt betrachtet stellt die dargestellte Vorgehensweise eine geeignete Strategie für einen DSS-Optimierer dar. Das Bereitstellen der relevanten statistischen Informationen im DWDBS kann der DSS-Optimierer anstoßen. Notwendig ist hierzu lediglich eine einfache Analyse der Anfragesequenz sowie einige Informationen bezüglich des zu Grunde liegenden Datenbanksystems. Der Optimierungsansatz ist hier nochmals in der Kurzbeschreibung wiedergegeben:

Optimierungsansatz: Bereitstellung von Statistiken

Identifiziere die in den Teilanfragen einer Anfragesequenz erzeugten temporären Tabellen und stoße für diese - sofern dies nicht automatisch erfolgt - die Erstellung der Statistik an.

Ergänze in der Anfragesequenz $S = (a_1, a_2, ..., a_n)$ jede Teilanfrage a_i mit $i < n$, in der eine temporäre Tabelle erstellt wird, um eine Anweisung o_i, die dafür sorgt, dass für diese temporäre Tabelle die Statistiken aktualisiert werden, sofern dies nicht automatisch erfolgt.

6.2.3 Zugriffspfade

Zugriffspfade sind ein wichtiges Mittel, um eine effiziente Anfrageverarbeitung in einem Datenbanksystem zu erzielen. Sie ermöglichen den wertabhängigen Zugriff auf einzelne Tupel einer Relation, ohne dass ein vollständiges Durchlaufen der Relation, in der Regel als *Scan* bezeichnet, notwendig wird. Abhängig davon, ob die Werte eines oder mehrerer Attribute als Schlüssel für den Zugriffspfad verwendet werden können, unterscheidet man zwischen eindimensionalen und mehrdimensionalen Zugriffspfaden. Von einem eindimensionalen Zugriffspfad spricht man auch dann noch, wenn der verwendete Schlüssel aus mehreren Attributen zusammengesetzt wird. Sowohl für eindimensionale als auch für mehrdimensionale Zugriffspfade wurde jeweils eine Reihe von Verfahren entwickelt, von denen einige auch in kommerziellen Datenbanksystemen realisiert sind [HR99]. Darüber, ob in einem konkreten Ausführungsplan ein vorhandener Zugriffpfad verwendet wird oder nicht, entscheidet der Anfrageoptimierer eines Datenbanksystems auf Grund eines Kostenmodells. Die Administratoren eines Datenbanksystems müssen in der Regel festlegen, welche Zugriffspfade in einer Datenbank angelegt werden sollen. Für einige Datenbanksysteme gibt es spezielle Werkzeuge, die sinnvolle Zugriffspfade für eine gegebene Datenbank vorschlagen [CN98] [Ibm01b]. Diese berücksichtigen hierzu einerseits eine Menge repräsentativer Anfragen und andererseits Randbedingungen, wie z.B. den verfügbaren Speicherplatz oder die für die Erstellung der Zugriffspfade geschätzte Laufzeit.

Unter der Vielzahl vorgeschlagener und untersuchter Zugriffspfade gibt es einige, die speziell im Hinblick auf die Anwendung in einem DWDBS besondere Vorteile aufweisen. Teilweise wurden sie speziell für diesen Anwendungsbereich entwickelt. Zu diesen für den Data-Warehouse-Bereich besonders geeigneten Zugriffspfaden gehören *Bitmap-Indexe* [One87] [CI98] [CI99]. Sie speichern pro vorhandenem Attributwert einen in der Regel komprimierten Bitmap-Vektor. Jede Position des Bitmap-Vektors ist einem Tupel einer Tabelle zugeordnet und abhängig von dem Attributwert dieses Tupels auf ‘0’ oder ‘1’ gesetzt. Anfragen, die Prädikate über mehrere Attribute einer Tabelle und Einschränkungen auf Wertebereiche des Attributs beinhalten, können mit Hilfe der Bitmap-Vektoren und durch das Ausführen einfacher boolescher Operationen auf diesen Vektoren beantwortet werden. *Join-Indexe* stellen einen anderen, auf Bitmap-Vektoren basierenden Ansatz dar [Val87]. Hier ist es das Ziel, Joins zwischen einer Faktentabelle und den zugehörenden Dimensionstabellen zu optimieren, indem in

Bitmap-Vektoren für die einzelnen Tupel der Dimensionstabellen die zugehörigen Tupel der Faktentabelle festgehalten werden. Die Ausführung einer typischen Anfrage auf einem Star-Schema kann dann zunächst auf die Verknüpfung von Bitmap-Vektoren beschränkt werden. Ein Zugriff auf die qualifizierten Tupel der umfangreicheren Faktentabelle wird erst im letzten Schritt notwendig [OG95].

Da der Zugriff auf die Daten in einem Data Warehouse in der Regel entlang mehrerer Dimensionen erfolgt, können hier insbesondere auch mehrdimensionale Zugriffspfade eingesetzt werden. Hierzu gehört einerseits der R-Baum [Gut84] und andererseits der UB-Baum [Bay96], der eine Verallgemeinerung des B-Baums darstellt, auf dem eine ganze Reihe von Zugriffspfaden basieren. Beide Zugriffspfade clustern die Daten so, dass je nach Anfrage nur eine vergleichsweise geringe Anzahl an Blöcken vom externen Speicher gelesen werden muss.

Ein DSS-Optimierer kann Zugriffspfade in der folgenden Art und Weise nutzen, um eine effizientere Ausführung von Anfragesequenzen zu erzielen: Auf Grundlage einer Analyse der Teilanfragen einer Anfragesequenz kann festgestellt werden, welche bisher noch nicht im Data Warehouse vorhandenen Zugriffspfade zu effizienteren Ausführungsplänen führen können. Das Anlegen dieser Zugriffspfade kann dann durch den DSS-Optimierer angestoßen werden, so dass sie bei der folgenden Ausführung der Anfragesequenz im Datenbanksystem zur Verfügung stehen und vom Anfrageoptimierer berücksichtigt werden können. Dieser Optimierungsansatz für einen DSS-Optimierer lässt sich also wie folgt charakterisieren:

Optimierungsansatz: Bereitstellung von Zugriffspfaden

Analysiere die Anfragesequenz und identifiziere relevante Zugriffspfade, die im Data Warehouse noch nicht angelegt sind. Sorge für die Bereitstellung dieser Zugriffspfade vor der Ausführung der einzelnen Teilanfragen.

Ergänze die Anfragesequenz $S = (a_1, a_2, ..., a_n)$ um eine Teilanfrage a_0, in der die Anweisung o_0 für das Anlegen zusätzlicher, relevanter Zugriffspfade sorgt. Ergänze die Ausführungsgeschichte g_S so, dass gilt: $g_S(o_0) < g_S(i_k)$ für alle $k > 0$.

Auch in den als Beispiele verwendeten Anfragesequenzen gibt es eine Reihe von Möglichkeiten, zusätzliche Zugriffspfade für effizientere Ausführungspläne zu nutzen. Im Folgenden wird der Begriff *Index* für konkrete Zugriffspfade in einem relationalen Datenbanksystem verwendet. Diese Bezeichnung orientiert sich an der SQL-Anweisung zum Anlegen von Zugriffspfaden (CREATE INDEX ...). Grundsätzlich soll hierbei davon ausgegangen werden, dass bei den einzelnen Basistabellen jeweils ein Index für das Primärschlüsselattribut bzw. die Kombination der Primärschlüsselattribute existiert. Für welche Attribute zusätzlich ein Index sinnvoll ist, kann durch eine Analyse der Prädikate, insbesondere der Join-Prädikate, der Teilanfragen einer Anfragesequenz festgestellt werden. So gibt es beispielsweise in den ersten beiden Teilanfragen der *Anfragesequenz A* (siehe Abbildung 19) jeweils ein Prädikat über dem Attribut *ordermonthkey* der Relation *orderday.* Ein Index für dieses Attribut kann gegebenenfalls die Bearbeitung der Teilanfragen beschleunigen. In denselben Teilanfragen gibt es jeweils einen Join zwischen den Tabellen *lineitem_orders* und *orderday.* In beiden Fällen wird in der Join-Bedingung das Attribut *orderday* der Tabelle *lineitem_orders* verwendet. Ein Index für dieses Attribut kann für den Ausführungsplan zusätzliche Alternativen bezüglich der Join-Algorithmen eröffnen und somit zu einer effizienteren Anfrageverarbeitung beitragen. Ergänzend sind auch gewinnbringende Zugriffspfade auf den in den einzelnen Teilanfragen erstellten temporären Tabellen denkbar.

In der Regel sind Zugriffspfade ein guter Ansatz, die Performanz von Datenbankanfragen zu erhöhen. Wenn dieser Ansatz auch in einem DSS-Optimierer umgesetzt wird, sind damit allerdings auch erhebliche Nachteile verbunden.

Zu diesen Nachteilen gehört, dass für die Auswahl geeigneter Zugriffspfade ein Modell notwendig ist, das Kosten und Nutzen der potenziellen Zugriffspfade berücksichtigt. Dieses Modell muss individuell für jedes Datenbanksystem, häufig sogar für jede Version eines Datenbanksystems, entwickelt werden. Dies bedeutet einerseits, dass dieser Ansatz eine große Abhängigkeit vom DWDBS aufweist. Andererseits existieren für die einzelnen kommerziellen Datenbanksysteme Werkzeuge, die speziell für die Auswahl geeigneter Zugriffspfade unter Berücksichtigung der Umgebungsparameter, wie z.B. verfügbarer Speicherplatz, entwickelt wurden. Die Erstellung zusätzlicher Zugriffspfade im Rahmen der Optimierung durch einen DSS-Optimierer müsste exakt die Arbeit dieser Werkzeuge nachbilden, allerdings für alle unterstützten Datenbanksysteme. Das ist mit erheblichem Aufwand verbunden. Insbesondere muss in jedem einzelnen Fall

den Vorteilen der zusätzlichen Zugriffspfade auch die geschätzte Zeit zu deren Erstellung gegenübergestellt werden. Unter Umständen werden dieselben Werkzeuge aber auch parallel dazu durch die Administratoren des DWDBS genutzt und so das physische Schema des Data Warehouse verändert. Es besteht die Gefahr sich widersprechender Annahmen und Vorgehensweisen, da hier an zwei unterschiedlichen Stellen und in der Regel unabhängig voneinander Entscheidungen für denselben Bereich, nämlich das physische Schema des Data Warehouse, getroffen werden.

Ein weiteres Problem dieses Ansatzes besteht in der Veränderung des physischen Schemas eines Data Warehouse, die sich ergibt, sobald zusätzliche Zugriffspfade eingefügt werden. Diese verändern nicht nur das Verhalten der Anfragesequenz, für die der DSS-Optimierer die zusätzlichen Zugriffspfade vorgesehen hat. Auch alle anderen Anwendungen, die auf eine Tabelle zugreifen, für die es einen zusätzlichen Zugriffspfad gibt, werden in ihrem Laufzeitverhalten beeinflusst. In der Regel sollten zusätzliche Indexe zwar zu effizienteren Ausführungsplänen führen. Dies kann allerdings nicht garantiert werden. Die autonomen Entscheidungen des DSS-Optimierers über zusätzliche Zugriffspfade beeinflussen also potenziell alle Anwendungen auf einem Data Warehouse. Dieser Einfluss kann nicht durch die Administratoren des Data Warehouse begrenzt werden.

Aus den genannten Gründen ist es sinnvoll, die Berücksichtigung zusätzlicher Zugriffspfade im Rahmen der Verarbeitung von Anfragesequenzen nicht als eine Strategie für einen DSS-Optimierer vorzusehen. Zusätzliche Indexe sind in vielen Fällen zwar sinnvoll, die Frage, wann und in welcher Form sie angelegt werden sollten, beeinflusst die Anwendungen auf einem Data Warehouse allerdings so erheblich, dass die Entscheidung darüber zu den Administrationsaufgaben für ein Data Warehouse gehören sollte. Alternativ könnte diese Entscheidung auch vom DWDBS direkt bei der Ausführung der einzelnen Teilanfragen getroffen werden. Allerdings wird dieser Ansatz bisher von keinem der kommerziellen Datenbanksysteme verfolgt.

6.2.4 Materialisierte Sichten

Unter der Überschrift materialisierte Sichten wird hier einer von zwei Optimierungsansätzen betrachtet, der im weitesten Sinne etwas mit der Vorabberechnung von Daten zu tun hat, die bei der Verarbeitung einer Anfragesequenz benötigt werden. In diesem Abschnitt geht es hierbei um die Vorabberechnung von Zwi-

schenergebnissen, die in einer Anfragesequenz mehrfach bzw. in unterschiedlichen Anfragesequenzen benötigt werden. Darüber hinaus kann auch eine Partitionierung der Faktentabelle als eine Vorabberechnung angesehen werden, die eine effizientere Verarbeitung der Anfragesequenzen zulässt. Dieser Ansatz wird in Abschnitt 6.2.5 genauer betrachtet.

In den Anfragen auf einem Data Warehouse werden dieselben Fakten auf unterschiedliche Hierarchiestufen der einzelnen Dimensionen aggregiert verwendet. In einer typischen Umgebung werden einige Aggregate regelmäßig von verschiedenen Anwendungen benötigt. Moderne Datenbanksysteme bieten die Möglichkeit, solche Aggregate im Data Warehouse bereitzustellen, automatisch zu aktualisieren und für die Anwendung transparent im Rahmen der Anfrageverarbeitung zu verwenden. In Publikationen wird dieser Ansatz in der Regel unter dem Begriff *Materialisierte Sichten* (engl. Materialized Views) beschrieben. Einzelne Datenbanksysteme verwenden hierfür allerdings andere Begriffe, wie z.B. *Automated Summary Tables* (AST) in DB2 [Ibm01c]. Häufig verwendete Aggregate können bei diesem Ansatz als materialisierte Sichten definiert werden. Dies erfolgt wie bei einer gewöhnlichen Sicht. Allerdings werden die Daten, die zum Ergebnis der Anfrage gehören, die die Sicht definiert, explizit gespeichert. Deren Berechnung findet nicht erst beim Zugriff auf die Sicht statt, wie dies bei gewöhnlichen Sichten der Fall ist. Das Datenbanksystem kann automatisch für die Aktualisierung der materialisierten Sichten sorgen und diese für die Bearbeitung von Anfragen verwenden. Berechnungen, die bereits bei der Erstellung der materialisierten Sicht durchgeführt wurden, müssen nicht nochmals wiederholt werden. So können in vielen Fällen geringere Anfrageausführungszeiten erzielt werden. Sowohl für die effiziente Aktualisierung materialisierter Sichten wie auch bezüglich deren transparenter Verwendung gibt es eine Reihe von Arbeiten, die teilweise direkt in kommerziellen Datenbanksystemen Eingang gefunden haben [GM95] [LR+00] [ZC+00] [LC+01] [GL01] [MR+01].

Die im Rahmen dieser Arbeit betrachteten Anfragesequenzen bestehen zumindest zum Teil aus den für eine Data-Warehouse-Umgebung typischen Anfragen mit Aggregaten. Es ist also denkbar, materialisierte Sichten als einen Ansatz für einen DSS-Optimierer zu verwenden. Dieser Ansatz lässt sich dann wie folgt charakterisieren:

Optimierungsansatz: Bereitstellung materialisierter Sichten

Analysiere die Anfragesequenz und identifiziere Zwischenergebnisse, die innerhalb der Sequenz mehrfach benötigt werden. Stelle diese vor der Ausführung der Sequenz als materialisierte Sichten bereit.

Ergänze die Anfragesequenz $S = (a_1, a_2, ..., a_n)$ um eine Teilanfrage a_0, in der die Anweisung o_0 für das Anlegen relevanter materialisierter Sichten sorgt. Ergänze die Ausführungsgeschichte g_S so, dass gilt: $g_S(o_0) < g_S(i_k)$ für alle $k > 0$.

Die Verwendung materialisierter Sichten kann auch an den Beispielen für Anfragesequenzen in Abschnitt 6.1.2 gezeigt werden. Die *Anfragesequenz C* enthält zu Beginn drei Teilanfragen, in denen jeweils ein Join zwischen der Tabelle *lineitem_orders* und der Tabelle *orderday* gebildet wird. Wird das Ergebnis dieses Joins vorab als materialisierte Sicht bereitgestellt, dann kann das DWDBS bei der Ausführung der ersten drei Teilanfragen auf diese Sicht zugreifen und aus deren Tupeln die für die einzelnen Teilanfragen qualifizierten über das Attribut *orderyearkey* auswählen.

Es gibt in der Literatur umfangreiche Arbeiten, die sich mit der Identifikation gemeinsamer Teilausdrücke in Datenbankanfragen sowie der darauf aufbauenden Auswahl für die Materialisierung geeigneter Sichten beschäftigen [Fin82]. In anderen Arbeiten wird die Auswahl zu materialisierender Sichten speziell im Anwendungsumfeld Data Warehouse betrachtet [HRU96]. Auf diese verfügbare Technologie kann bei der Realisierung des beschriebenen Ansatzes für einen DSS-Optimierer zurückgegriffen werden. Dessen ungeachtet hat dieser Ansatz erhebliche Nachteile. Diese lassen die Bereitstellung von materialisierten Sichten nicht als geeignet für einen DSS-Optimierer erscheinen.

Ein wichtiger Nachteil ergibt sich aus der Betrachtung der Entscheidungsgrundlage, die dem DSS-Optimierer für die Erstellung materialisierter Sichten zur Verfügung steht. Im einfachsten Fall ist dies eine Anfragesequenz, für deren möglichst optimale Ausführung der DSS-Optimierer gerade sorgen soll. Auf dieser Grundlage kann der DSS-Optimierer mit Hilfe vorhandener Technologien

Sichten identifizieren, deren Materialisierung für die Anfragesequenz potenziell gewinnbringend ist. Auf der Grundlage eines Kostenmodells kann er außerdem darüber entscheiden, wie der Vorteil für die Ausführung der einzelnen Teilanfragen im Vergleich zur Erstellung der materialisierten Sicht zu bewerten ist. Eventuelle Vorteile für die Ausführung anderer Anfragesequenzen können hierbei allerdings nicht berücksichtigt werden. Dies ist nur möglich, wenn der DSS-Optimierer eine Historie verarbeiteter Anfragesequenzen verwaltet und regelmäßig daraufhin untersucht, ob zusätzliche materialisierte Sichten die Verarbeitung typischer Anfragesequenzen beschleunigen könnten bzw. ob zuvor angelegte materialisierte Sichten nicht mehr notwendig sind. Ist diese Historie vorhanden, kann der DSS-Optimierer seine Entscheidungen bezüglich materialisierter Sichten daran orientieren, welche Sichten längerfristig gewinnbringend sind.

In der ersten Variante, bei der nur die aktuelle Anfragesequenz als Grundlage für die Entscheidung des DSS-Optimierers dient, ist der DSS-Optimierer zwar die richtige Systemkomponente, um die Entscheidung über eine Materialisierung mehrfach benötigter Zwischenergebnisse zu treffen. Bei der typischen sequenziellen Ausführung einer Anfragesequenz hat das Datenbanksystem keine Möglichkeit, gemeinsame Teilausdrücke zu identifizieren, da es zu jedem Zeitpunkt genau eine Anweisung der Sequenz zur Ausführung erhält. Durch die begrenzte Entscheidungsgrundlage sind allerdings auch die positiven Auswirkungen der erstellten materialisierten Sichten zunächst auf die aktuelle Anfragesequenz begrenzt. Anders sieht dies bei der zweiten Variante unter Berücksichtigung einer Historie aus. In diesem Fall ist allerdings das DWDBS selbst der bessere Ort für die Umsetzung des Optimierungsansatzes. Auch im zu Grunde liegenden Datenbanksystem kann eine Historie über alle ausgeführten Anfragen verwaltet, analysiert und als Grundlage für Empfehlungen bezüglich materialisierter Sichten genutzt werden. Dies ist deshalb sinnvoll, da hierbei alle Anfragen, die an das Data Warehouse gestellt werden, berücksichtigt werden können und nicht nur solche, die von einigen Analyseanwendungen über einen DSS-Optimierer weitergegeben werden. Die Erstellung materialisierter Sichten durch den DSS-Optimierer weist auf Grund der eingeschränkten Entscheidungsgrundlage zunächst einen sehr begrenzten Einflussbereich auf. Verbreitert man jedoch die Entscheidungsgrundlage, so ist eine Realisierung des Optimierungsansatzes direkt im DWDBS besser geeignet.

Darüber hinaus spricht noch ein weiteres Argument gegen die Erstellung materialisierter Sichten durch den DSS-Optimierer. Wie beim Erstellen von zusätzli-

chen Zugriffspfaden sind auch mit dem Anlegen zusätzlicher materialisierter Sichten durch den DSS-Optimierer potenziell negative Auswirkungen für das Data Warehouse verbunden. Dies betrifft den zusätzlichen Speicherplatz, der für die Materialisierung benötigt wird, ebenso wie den Verarbeitungsaufwand, der für die Erstellung der Sichten notwendig ist. Da der DSS-Optimierer seine Entscheidungen autonom trifft, entziehen sich diese Auswirkungen der direkten Kontrolle durch die Administratoren eines Data Warehouse. Wie bereits in Abschnitt 6.2.3 beschrieben, ist dies ein nicht gewünschter Effekt. Insgesamt sprechen die aufgeführten Argumente gegen eine Realisierung des hier betrachteten Optimierungsansatzes innerhalb eines DSS-Optimierers.

6.2.5 Partitionierung

In diesem Abschnitt soll ein weiterer Optimierungsansatz diskutiert werden, der auf einer Veränderung des physischen Data-Warehouse-Schemas basiert. Die Partitionierung einer Relation, d.h. deren Aufteilung auf mehrere Teilrelationen, wird in erster Linie bei parallelen und verteilten Datenbanksystemen eingesetzt [Rah94]. In parallelen Systemen kann damit erreicht werden, dass die einzelnen Teilrelationen getrennt voneinander und gegebenenfalls von separaten Prozessen auf unterschiedlichen Prozessoren verarbeitet werden können. Die Lastbalancierung ist hier ein wesentliches Ziel. Im Bereich der verteilten Datenbanksysteme ist ein wichtiger Vorteil der Partitionierung, dass die verschiedenen Teilrelationen von unterschiedlichen Transaktionen bearbeitet werden können, sofern diese jeweils nur eine begrenzte Anzahl der Teilrelationen betreffen. Unmittelbar mit der Partitionierung ist die Allokation der gebildeten Partitionen verbunden. Hier wird festgelegt, wie die Partitionen auf die zur Verfügung stehenden Ressourcen verteilt werden müssen, um über eine möglichst lokale Anfrageverarbeitung, eine Minimierung der Kommunikationskosten sowie eine möglichst gute Lastverteilung zu erreichen. Fragen der Datenallokation wurden gerade auch für den Bereich Data Warehouse untersucht [SMR00] [SR01].

Bei der Partitionierung werden zwei Ansätze unterschieden, die auch in einem Data Warehouse angewandt werden können. Diese sind die horizontale sowie die vertikale Partitionierung. Bei der horizontalen Partitionierung wird jedes Tupel der zu partitionierenden Relation genau einer Teilrelation zugeordnet. Dies kann nach unterschiedlichen Kriterien erfolgen. In parallelen Datenbanksystemen erfolgt die Partitionierung häufig mit Hash-Verfahren. Sie ist damit völlig transparent für die Anwendung. Anders sieht dies bei der wertabhängigen Partitionie-

rung aus. Hier werden Attributwerte genutzt, um das Kriterium zur Aufteilung der Relation zu definieren. Erfolgt dies unter Berücksichtigung der Prädikate in typischen Anfragen, dann kann erreicht werden, dass bei der Anfrageverarbeitung vielfach nur eine begrenzte Zahl der Partitionen berücksichtigt, d.h. gelesen werden muss. Am Beispiel der ersten Teilanfrage von *Anfragesequenz A* soll die Partitionierung verdeutlichet werden. Ein Teil der WHERE-Klausel dieser Anfrage stellt ein Prädikat hinsichtlich der Zeitdimension dar, über das einzelne Monate (hier 199401 und 199402) ausgewählt werden. Dies legt eine horizontale Partitionierung der Tabelle *lineitem_orders* entlang der Zeitdimension nahe. Speziell für die genannte Teilanfrage wäre eine Aufteilung sinnvoll, bei der für jeden Monat eine eigene Partition der Tabelle *lineitem_orders* vorhanden ist. Für die Auswertung der Anfrage muss dann nur noch auf eine Partition pro Monat und nicht auf die gesamte Faktentabelle zugegriffen werden. Diese Art der Partitionierung kann im Rahmen der Anfrageverarbeitung auf zwei unterschiedliche Weisen genutzt werden. Einerseits kann die Nutzung transparent für die Anwendung erfolgen. In diesem Fall werden die Anfragen auf Grundlage der gesamten Tabelle formuliert und das Datenbanksystem identifiziert bei der Verarbeitung der Anfrage die relevanten Partitionen. Andererseits kann die Partitionierung auch bereits bei der Erstellung der Anfrage ausgenutzt werden, indem sich diese direkt auf einzelne Partitionen bezieht. Im OLAP-Bereich gibt es einige Werkzeuge, die diese anwendungsseitige Nutzung der Partitionierung unterstützten [Mic99].

Im Gegensatz zur horizontalen Partitionierung, bei der wie beschrieben einzelne Tupel den Teilrelationen zugeordnet werden, erfolgt diese Zuordnung bei der vertikalen Partitionierung für einzelne Attribute [NC+84]. Dies ist dann gewinnbringend, wenn für typische Anfragen nicht auf alle Attribute einer Relation Bezug genommen wird. Werden nur die typischerweise in Anfragen verwendeten Attribute in einer Teilrelation gespeichert, dann reicht es bei der Anfrageverarbeitung aus, nur auf diese Teilrelation zuzugreifen. Ein Leistungsvorteil kann sich ergeben, da nicht die gesamte Tabelle in die Verarbeitung der Anfrage einbezogen wird. Auch hier ist es wiederum möglich, dass die Nutzung der Partitionierung transparent für die Anwendung erfolgt. Aber auch deren Berücksichtigung bei der Anfrageerstellung ist wie bei der horizontalen Partitionierung denkbar.

Die Art und Weise, wie ein DSS-Optimierer die Möglichkeit zur Partitionierung der Daten nutzen kann, lässt sich wie folgt charakterisieren:

Optimierungsansatz: Partitionierung der Basistabellen

Analysiere die Anfragesequenz und identifiziere Basistabellen, die partitioniert werden sollten. Sorge vor der Ausführung der Sequenz für die entsprechende Partitionierung.

Ergänze die Anfragesequenz $S = (a_1, a_2, ..., a_n)$ um eine Teilanfrage a_0, in der die Anweisung o_0 für das Partitionieren von Basistabellen sorgt. Ergänze die Ausführungsgeschichte g_S so, dass gilt:
$g_S(o_0) < g_S(i_k)$ für alle $k > 0$.

Eine Partitionierung der Daten kann gerade auch in einem Data Warehouse zu erheblichen Performanzvorteilen führen. Trotzdem ist ein DSS-Optimierer nicht die geeignete Systemkomponente hierfür. Ausschlaggebend für diese Bewertung sind Argumente, die in ähnlicher Form bereits bei Zugriffspfaden und materialisierten Sichten eine Rolle gespielt haben. Hier wie dort steht als Entscheidungsgrundlage zunächst nur die aktuell durch den DSS-Optimierer zu bearbeitende Anfragesequenz zur Verfügung. Nur für diese kann also eine geeignete Partitionierung von Tabellen identifiziert werden. Deren Anwendbarkeit auf andere Anfragesequenzen kann nicht weiter berücksichtigt werden. Diesen sehr begrenzten Auswirkungen steht ein unter Umständen erheblicher Aufwand gegenüber. Beim Erstellen der Partitionierung handelt es sich in vielen Fällen um eine sehr teure Operation, da die zu partitionierende Tabelle hierbei bereits einmal komplett gelesen und verarbeitet werden muss. Dies ist insbesondere bei der Partitionierung der häufig sehr umfangreichen Faktentabellen mit erheblichen Kosten verbunden. In der Regel sind allerdings bei einer Partitionierung der Faktentabelle auch die größten Auswirkungen auf die Performanz zu erwarten. Diese müssen größer ausfallen als der Aufwand für die Bereitstellung der Partitionen.

Der begrenzten Entscheidungsgrundlage steht eine erheblicher Einflussbereich der Entscheidung über eine Partitionierung von Tabellen gegenüber. Gerade die Partitionierung einer sehr umfangreichen Tabelle nimmt in der Regel einige Zeit in Anspruch, während der die Tabelle nicht für andere Anwendungen zur Verfügung steht. Für die Performanz des gesamten Data Warehouse ist es darum nicht sinnvoll, die Partitionierungsentscheidung einer aus Sicht des Datenbanksystems

externen Komponente, wie dem DSS-Optimierer, zu überlassen. Somit stellt sich auch die Partitionierung als eine Entscheidung dar, die zum Bereich der Administration eines Data Warehouse zu zählen ist. Hierbei kann dann beispielsweise auch berücksichtigt werden, welcher zusätzliche Speicherplatz für die partitionierten Tabellen notwendig ist.

6.2.6 Restrukturierung von Anfragesequenzen

Die bisher analysierten Optimierungsansätze basieren alle entweder auf einer modifizierten Ausführung einer Anfragesequenz oder auf der Modifikation des physischen Schemas des Data Warehouse. Ein weiteres wichtiges Prinzip von Optimierungsansätzen ist darüber hinaus die Modifikation der Anfrage bzw. der Anfragesequenz. Diese Ansätze werden in diesem und dem folgenden Abschnitt analysiert. Zunächst sollen Modifikationen von Anfragesequenzen betrachtet werden, die als Ergebnis wieder eine Anfragesequenz liefern. In Abschnitt 6.2.7 wird dann ein Sonderfall separat betrachtet. Dort wird als Ergebnis der Modifikation einer Anfragesequenz gezielt eine einzelne SQL-Anfrage erstellt.

Die Anfragerestrukturierung spielt in modernen Datenbanksystemen eine bedeutende Rolle [Mit95]. Die Aufgabe dieses Schritts der Anfrageverarbeitung ist es, eine gegebene Anfrage in eine äquivalente Form zu überführen, die für die weitere Verarbeitung der Anfrage besser geeignet ist und eine effizientere Ausführung ermöglicht. Die äquivalente Form der Anfrage wird durch die Anwendung von Regeln zur Anfragerestrukturierung erreicht. Diese Regeln operieren meist auf einer internen Repräsentation der Anfrage. Sie folgen dabei allgemeinen Optimierungsheuristiken, wie z.B.:

- Selektionen sollen möglichst frühzeitig ausgeführt werden, um die zu verarbeitende Datenmenge zu reduzieren.
- Einfache Selektionen sollen zu einer Selektion zusammengefasst werden.
- Projektionen sollen möglichst frühzeitig ausgeführt werden, um die zu verarbeitende Attributmenge zu verringern.
- Gemeinsame Teilausdrücke sollen nur einmal berechnet werden.

Verschiedene Realisierungskonzepte für diese regelbasierte Anfragerestrukturierung sowie verschiedenste Restrukturierungsregeln werden in der Literatur beschrieben [HF+89] [FG91] [BG92] [PHH92] [PLH97]. Diese bilden die wesentliche Grundlage für eine Übertragung der Anfragerestrukturierung auf

Anfragesequenzen. Um die beiden Bereiche abzugrenzen, wird die Modifikation von Anfragesequenzen im Rahmen dieser Arbeit als *Sequenzrestrukturierung* bezeichnet.

Für die Restrukturierung von Anfragesequenzen können vergleichbare Optimierungsheuristiken zum Einsatz kommen wie bei der Anfragerestrukturierung. So können beispielsweise Selektionen über die Grenzen der einzelnen Teilanfragen hinweg verschoben werden oder Teilanfragen, die ähnliche Berechnungen ausführen, verschmolzen werden. Um dies zu konkretisieren, werden im Folgenden wichtige Regeln zur Sequenzrestrukturierung angegeben und die Anwendung einiger dieser Regeln am Beispiel der Anfragesequenzen aus Abschnitt 6.1.2 gezeigt. Das Ziel dieses Abschnitts ist es, zu zeigen, dass es für eine mit der Anfragerestrukturierung vergleichbaren Modifikation von Anfragesequenzen eine ganze Reihe von Restrukturierungsregeln gibt, die einfach angewendet werden können. Eine detaillierte Analyse der Regelmenge oder die Realisierung der Regelverarbeitung sind nicht Gegenstand dieser Arbeit. Deshalb konnte auf eine formalere Darstellung der Regeln sowie auf den Bezug zu einer internen Repräsentation verzichtet werden. Die Regeln werden weitgehend natürlichsprachlich angegeben und beziehen sich auf die einzelnen Klauseln der Teilanfragen von Anfragesequenzen. Entsprechend den Definitionen in Abschnitt 6.1.1 beziehen sich die hier formulierten Restrukturierungsregeln auf eine allgemeine Anfragesequenz $S = (a_1, a_2, ..., a_n)$ mit $a_1 = (c_1, i_1, d_1, o_1)$, $a_2 = (c_2, i_2, d_2, o_2)$ *und* $a_n = (c_n, i_n, d_n, o_n)$. Dabei wird davon ausgegangen, dass die CREATE-Anweisung der einzelnen Teilanfragen eine Tabelle erzeugt, die in der INSERT-Anweisung mit Daten gefüllt wird. Die einzelnen Regeln bestehen jeweils aus einem Bedingungs- und einem Ausführungsteil. Zu beachten ist, dass bei der Formulierung zu Gunsten einer vereinfachten Darstellung nicht alle Spezialfälle sowie Äquivalenzumformungen der Prädikate berücksichtigt wurden.

Regel 1 Vereinigung von Selektionen der Teilanfragen a_i und a_k der Anfragesequenz S:

IF
- Die CREATE-Anweisungen c_i und c_k sind bis auf den Bezeichner für die erzeugte Tabelle identisch und
- die INSERT-Anweisungen i_i und i_k sind bis auf die SELECT-Klausel identisch.

THEN
- Füge in i_i noch nicht vorhandenen Teile der SELECT-Klausel von i_k hinzu,
- ersetze in den Teilanfragen von S alle Verweise auf a_k durch Verweise auf a_i und
- lösche die Teilanfrage a_k.

Diese Regel erlaubt es, zwei Teilanfragen zu verschmelzen, die sich lediglich in der SELECT-Klausel unterscheiden. Dies kann z.B. in der *Anfragesequenz C* für die beiden Teilanfragen *C5* und *C6* sinnvoll sein, um damit die Anzahl der Teilanfragen zu reduzieren. Abbildung 24 enthält die sich ergebende Anfrage *C56*.

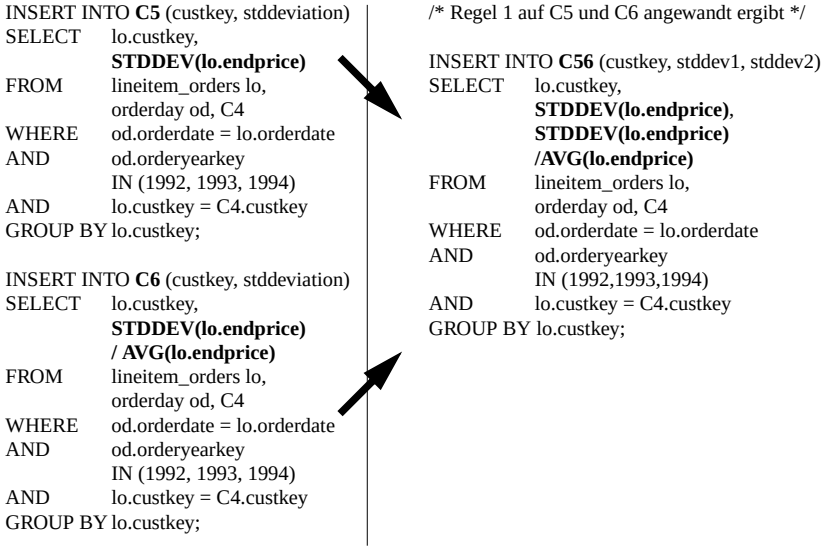

Abbildung 24: Anwendungsbeispiel für Regel 1

Die folgende Regel erlaubt eine ähnliche Verschmelzung von Teilanfragen. Hier werden allerdings im Gegensatz zu Regel 1 Teilanfragen betrachtet, die sich lediglich in der WHERE-Klausel unterscheiden. Da sich die Vorgehensweise nicht wesentlich von der für Regel 1 beschriebenen unterscheidet, wird hier auf ein Beispiel für die Anwendung dieser Regel verzichtet.

Regel 2 Vereinigung von Prädikaten der Teilanfragen a_i und a_k der Anfragesequenz S:

IF
- Die CREATE-Anweisungen c_i und c_k sind bis auf den Bezeichner für die erzeugte Tabelle identisch,
- die INSERT-Anweisungen i_i und i_k sind bis auf die WHERE-Klausel identisch und
- die INSERT-Anweisungen i_i und i_k enthalten keine Gruppierung.

THEN
- Füge in i_i noch nicht vorhandenen Teile der WHERE-Klausel von i_k hinzu, wobei die für die WHERE-Klauseln von i_i und i_k spezifischen Teile durch OR verknüpft werden,
- ersetze in den Teilanfragen von S alle Verweise auf a_i bzw. a_k durch Verweise auf a_i, ergänzt um die spezifischen Prädikate von a_i bzw. a_k, und
- lösche die Teilanfrage a_k.

Die Ersetzung von Prädikaten in einer Anfrage durch eine Gruppierung und die damit verbundene Zusammenfassung von Teilanfragen wird durch die folgende Regel beschrieben. Bei dieser kann dann auch die in Regel 2 vorhandene Einschränkung bezüglich der Gruppierung entfallen, so dass weniger Voraussetzungen für die Anwendung der Regel erfüllt sein müssen. Ein Beispiel für den Einsatz der Regel bei der Modifikation der ersten drei Teilanfragen von Anfragesequenz C ist in Abbildung 25 enthalten. Hier kann die Anzahl der Teilanfragen durch die Regelanwendung um zwei reduziert werden. Am Beispiel der Teilanfrage *C4* wird hier auch gezeigt, wie Teilanfragen verändert werden, die auf die neue Teilanfrage *C123* zugreifen.

Regel 3 Umwandlung von Prädikaten der Teilanfragen a_i und a_k der Anfragesequenz S in eine Gruppierung:

IF
- Die CREATE-Anweisungen c_i und c_k sind bis auf den Bezeichner für die erzeugte Tabelle identisch,
- die INSERT-Anweisungen i_i und i_k sind bis auf die WHERE-Klausel identisch und
- die WHERE-Klauseln von i_i und i_k unterscheiden sich genau in einem Teilprädikat, in dem sie dasselbe Attribut x auf unterschiedliche Werte einschränken.

THEN
- Ergänze in i_i das Prädikat über x um die Werte auf die x in i_k eingeschränkt wurde,
- ergänze in i_i die GROUP-BY-Klausel um eine Gruppierung über das Attribut x,
- ergänze in i_i die SELECT-Klausel um das Attribut x und
- ersetze in den Teilanfragen von S alle Verweise auf a_i oder a_k durch Verweise auf a_i, ergänzt um das Prädikat über x aus i_i oder i_k.

Joins mit einzelnen Tabellen haben in Anfragesequenzen unter Umständen das Ziel, Datensätze herauszufiltern, die sich bereits auf Grund von Selektionen in anderen Teilanfragen qualifiziert haben. Die folgende Regel beschreibt, wie diese Joins in andere Teilanfragen verlagert werden können.

Regel 4 Verschieben einer Join-Operationen von Teilanfrage a_i in die Teilanfrage a_k

IF
- Die INSERT-Anweisung i_i enthält einen Join mit Tabelle x,
- die Join-Bedingung enthält neben den Attributen aus Tabelle x nur Attribute der SELECT-Klausel von i_i und
- keines der Attribute von x wird in der SELECT-, GROUP BY- oder HAVING-Klausel von i_i verwendet.

THEN
- Verschiebe den Join mit der Tabelle x und die zugehörige Join-Bedingung in jede Teilanfrage a_k, die einen Bezug auf a_i enthält.

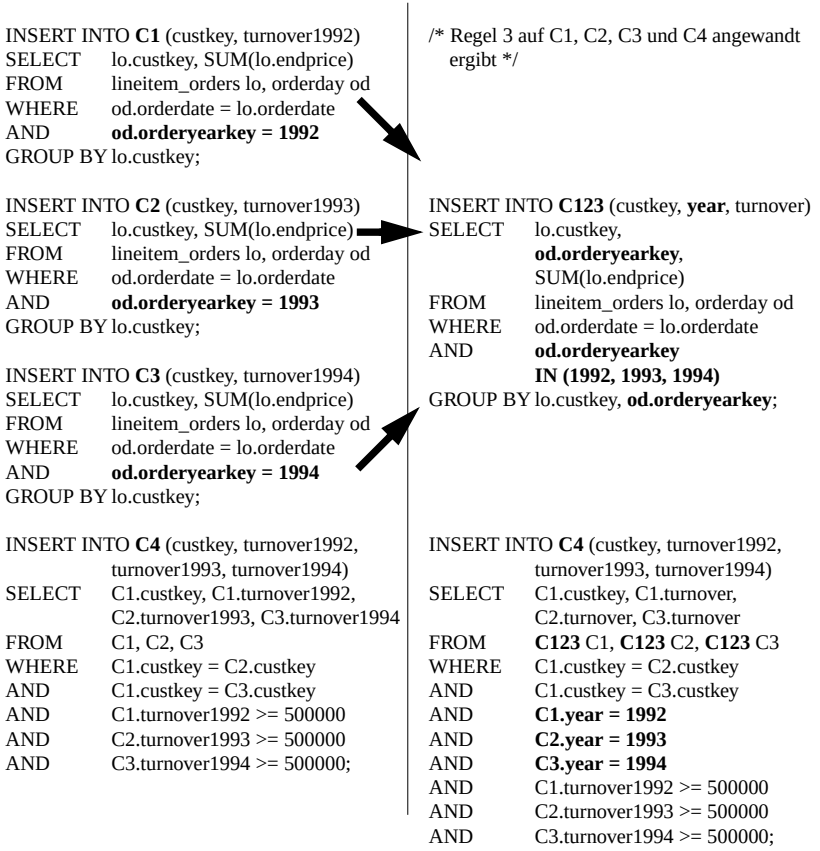

Abbildung 25: Anwendungsbeispiel für Regel 3

Ein Anwendungsbeispiel für diese Regel ist in Abbildung 26 gegeben. Ausgangspunkt ist hier die Teilanfrage *C56*, die in Abbildung 24 als Ergebnis der Anwendung von Regel 1 dargestellt ist. Diese enthält einen Join mit der Tabelle *C4*, der in die Teilanfrage *C7* verschoben werden kann. In der neuen Teilanfrage *C7* erfolgt ein Join der Tabellen *C1*, *C2* und *C3* über das Attribut *custkey*, der so auch in Teilanfrage *C4* enthalten ist. Der Join der drei Tabellen kann darum gestrichen werden und durch Bezüge auf *C4* ersetzt werden. Details der zugehörigen Regel werden hier nicht weiter erläutert.

Mit der letzten hier vorgestellten Regel zur Restrukturierung von Anfragesequenzen, können mehrfach auftretende Prädikate über die Grenzen von Teilanfragen hinweg verschoben werden, so dass sie nur in einer Teilanfrage berücksichtigt werden müssen. Die Anwendung dieser Regel 5 kann auf der Grundlage der beiden Teilanfragen *C123* und *C4* aus Abbildung 25 verdeutlicht werden. Wie in Abbildung 27 zu sehen ist, können drei Prädikate aus der Teilanfrage *C4* in die HAVING-Klausel der vorangehenden Teilanfrage *C123* verschoben werden.

```
INSERT INTO C56 (custkey, stddev1,
                 stddev2)
SELECT    lo.custkey,
          STDDEV(lo.endprice),
          STDDEV(lo.endprice)
          /AVG(lo.endprice)
FROM      lineitem_orders lo,
          orderday od, C4
WHERE     od.orderdate = lo.orderdate
AND       od.orderyearkey
          IN (1992,1993,1994)
AND       lo.custkey = C4.custkey
GROUP BY lo.custkey;

INSERT INTO C7 (custkey, custname,
          stddev1, stddev2, turnover1992,
          turnover1993, turnover1994)
SELECT    cu.custkey, cu.custname,
          C56.stddev1,
          C56.stddev2,
          C1.turnover1992,
          C2.turnover1993,
          C3.turnover1994
FROM      C1, C2, C3, C56,
          customer cu
WHERE     C56.custkey = C1.custkey
AND       C56.custkey = C2.custkey
AND       C56.custkey = C3.custkey
AND       C56.custkey = cu.custkey;
```

```
/* Regel 4 auf C56 und C7 angewandt ergibt */

INSERT INTO C56 (custkey, stddev1,
                 stddev2)
SELECT    lo.custkey,
          STDDEV(lo.endprice),
          STDDEV(lo.endprice)
          /AVG(lo.endprice)
FROM      lineitem_orders lo,
          orderday od
WHERE     od.orderdate = lo.orderdate
AND       od.orderyearkey
          IN (1992,1993,1994)
GROUP BY lo.custkey;

INSERT INTO C7 (custkey, custname,
          stddev1, stddev2, turnover1992,
          turnover1993, turnover1994)
SELECT    cu.custkey, cu.custname,
          C56.stddev1,
          C56.stddev2,
          C1.turnover1992,
          C2.turnover1993,
          C3.turnover1994
FROM      C1, C2, C3, C56,
          customer cu, C4
WHERE     C56.custkey = C1.custkey
AND       C56.custkey = C2.custkey
AND       C56.custkey = C3.custkey
AND       C56.custkey = cu.custkey
AND       C56.custkey = C4.custkey;
```

Abbildung 26: Anwendungsbeispiel für Regel 4

Regel 5 Verschieben von gemeinsamen Prädikaten in Teilanfragen, die a_i referenzieren, nach a_i:

IF
- a_i wird in einer oder mehreren Teilanfragen referenziert. Diese weisen ein gemeinsames Prädikat p über Attributen aus a_i auf.

THEN
- Füge das gemeinsame Prädikat p der WHERE-Klausel von i_i hinzu, sofern i_i keine Gruppierung enthält oder
- füge das gemeinsame Prädikat p der HAVING-Klausel von i_i hinzu, sofern i_i eine Gruppierung enthält, und
- streiche das Prädikat p in allen Teilanfragen, die a_i referenzieren.

```
                                        /* Regel 5 auf C123 und C4 angewandt ergibt */

INSERT INTO C123 (custkey, year, turnover)   INSERT INTO C123 (custkey, year, turnover)
SELECT      lo.custkey,                      SELECT      lo.custkey,
            od.orderyearkey,                 ->          od.orderyearkey,
            SUM(lo.endprice)                             SUM(lo.endprice)
FROM        lineitem_orders lo, orderday od  FROM        lineitem_orders lo, orderday od
WHERE       od.orderdate = lo.orderdate      WHERE       od.orderdate = lo.orderdate
AND         od.orderyearkey                  AND         od.orderyearkey
            IN (1992, 1993, 1994)                        IN (1992, 1993, 1994)
GROUP BY    lo.custkey, od.orderyearkey;     GROUP BY    lo.custkey, od.orderyearkey
                                             HAVING      SUM(lo.endprice)  >= 500000;

INSERT INTO C4 (custkey, turnover1992,       INSERT INTO C4 (custkey, turnover1992,
                turnover1993, turnover1994)                  turnover1993, turnover1994)
SELECT      C1.custkey, C1.turnover,         SELECT      C1.custkey, C1.turnover,
            C2.turnover, C3.turnover         ->          C2.turnover, C3.turnover
FROM        C123 C1, C123 C2, C123 C3        FROM        C123 C1, C123 C2, C123 C3
WHERE       C1.custkey = C2.custkey          WHERE       C1.custkey = C2.custkey
AND         C1.custkey = C3.custkey          AND         C1.custkey = C3.custkey
AND         C1.year = 1992                   AND         C1.year = 1992
AND         C2.year = 1993                   AND         C2.year = 1993
AND         C3.year = 1994                   AND         C3.year = 1994;
AND         C1.turnover1992 >= 500000
AND         C2.turnover1993 >= 500000
AND         C3.turnover1994 >= 500000;
```

Abbildung 27: Anwendungsbeispiel für Regel 5

Mit Hilfe der beschriebenen Regeln können unterschiedlich modifizierte Anfragesequenzen erstellt werden. Zwei der möglichen Alternativen werden hier angegeben. Dies sind die Anfragesequenzen, die auch bei den in Abschnitt 6.3 beschriebenen Leistungsmessungen verwendet wurden. Sie werden in dieser Arbeit als *Modifikation 1* und *Modifikation 2* von *Anfragesequenz C* bezeichnet. In [Kra02] sind weitere Beispiele für Restrukturierungsregeln angegeben. Bei deren Anwendung sind noch eine Vielzahl von Alternativen zur ursprünglichen Anfragesequenz möglich.

Das Ergebnis der ersten Modifikation ist in Abbildung 28 dargestellt. Hier wurden ausgehend von der ursprünglichen Anfragesequenz nur die Teilanfragen *C5*, *C6* und *C7* verändert. Im Wesentlichen wurden hierbei die Regel 1 und die Regel 4 angewandt. Durch Regel 1 konnte die Zahl der Teilanfragen in der Anfragesequenz um eins reduziert werden, während mit Hilfe der Regel 4 die Zahl der Joins in den Teilanfragen *C56* und *C7* reduziert werden konnte. Beide Regelanwendungen wurden in Abbildung 24 und in Abbildung 26 bereits getrennt dargestellt.

In Abbildung 29 ist die zweite Variante der *Anfragesequenz C* dargestellt. Diese Modifikation 2 entstand aus der ersten durch die zusätzliche Anwendung der Regel 3 sowie der Regel 5. Mit Hilfe der Regel 3 konnten die ersten drei Teilanfragen aus Abbildung 28 zusammengefasst werden. Das Verschieben der Prädikate über der Umsatzsumme in die neu entstandene Teilanfragen C123 konnte durch Regel 5 erzielt werden. Dieser Teilschritt wurde bereits in Abbildung 27 gezeigt.

Die Sequenzrestrukturierung erweist sich als ein geeigneter Ansatz für einen DSS-Optimierer. Als Grundlage für die Anwendung der Restrukturierungsregeln ist lediglich die Anfragesequenz selbst notwendig. Alle Bedingungsteile der möglichen Restrukturierungsregeln beziehen sich ausschließlich auf einzelne Klauseln der betrachteten Teilanfragen. Wie bei der Anfragerestrukturierung für Einzelanfragen kann auch der hier beschriebenen Sequenzrestrukturierung ein hohes Optimierungspotenzial zugewiesen werden. Auch bei Anfragesequenzen ist es von erheblicher Bedeutung, in welcher Reihenfolge Selektionen und Projektionen innerhalb der Anfragesequenz ausgeführt werden und ob gleiche bzw. ähnliche Anfragen einmal oder mehrfach ausgeführt werden. In Abschnitt 6.3 wird gezeigt, inwiefern das theoretisch vorhandene Optimierungspotenzial auch experimentell bestätigt werden kann.

```
INSERT INTO C1 (custkey, turnover1992)                    /* unverändert */
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate     AND od.orderyearkey = 1992
GROUP BY lo.custkey;

INSERT INTO C2 (custkey, turnover1993)                    /* unverändert */
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate     AND od.orderyearkey = 1993
GROUP BY lo.custkey;

INSERT INTO C3 (custkey, turnover1994)                    /* unverändert */
SELECT    lo.custkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate     AND od.orderyearkey = 1994
GROUP BY lo.custkey;

INSERT INTO C4 (custkey, turnover1992, turnover1993,      /* unverändert */
                turnover1994)
SELECT    C1.custkey, C1.turnover1992, C2.turnover1993, C3.turnover1994
FROM      C1, C2, C3
WHERE     C1.custkey = C2.custkey         AND C1.custkey = C3.custkey
AND       C1.turnover1992 >= 500000       AND C2.turnover1993 >= 500000
AND       C3.turnover1994 >= 500000;

INSERT INTO C56 (custkey, stddev1, stddev2)               /* nach Regel 1 und Regel 4 */
SELECT    lo.custkey, STDDEV(lo.endprice), STDDEV(lo.endprice) / AVG(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate     AND od.orderyearkey IN (1992,1993,1994)
GROUP BY lo.custkey;

INSERT INTO C7 (custkey, custname, stddev1, stddev2,      /* nach Regel 1 und Regel 4 */
                turnover1992, turnover1993, turnover1994)
SELECT    C4.custkey, cu.custname, C56.stddev1, C56.stddev2, C4.turnover1992,
          C4.turnover1993, C4.turnover1994
FROM      C4,  C56,  customer cu
WHERE     C4.custkey = C56.custkey        AND C4.custkey = cu.custkey;

INSERT INTO C8 (custkey, custname, stddev1, stddev2,      /* unverändert */
                turnover1992, turnover1993, turnover1994)
SELECT    C7.custkey, C7.custname, C7.stddev1, C7.stddev2, C7.turnover1992,
          C7.turnover1993, C7.turnover1994
FROM      C7
WHERE     C7.stddev2 <= 0.66794004454646;
```

Abbildung 28: Modifizierte Anfragesequenz C (Modifikation 1)

```
INSERT INTO C123 (custkey, year, turnover)                    /* nach Regel 3 und Regel 5 */
SELECT    lo.custkey, od.orderyearkey, SUM(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey IN (1992, 1993, 1994)
GROUP BY lo.custkey, od.orderyearkey
HAVING    SUM(lo.endprice)   >= 500000;

INSERT INTO C4 (custkey, turnover1992, turnover1993,          /* nach Regel 3 und Regel 5 */
                turnover1994)
SELECT    C1.custkey, C1.turnover, C2.turnover, C3.turnover
FROM      C123 C1, C123 C2, C123 C3,
WHERE     C1.custkey = C2.custkey          AND C1.custkey = C3.custkey
AND       C1.year = 1992                   AND C2.year = 1993
AND       C3.year = 1994;

INSERT INTO C56 (custkey, stddev1, stddev2)                   /* nach Regel 1 und Regel 4 */
SELECT    lo.custkey, STDDEV(lo.endprice), STDDEV(lo.endprice) / AVG(lo.endprice)
FROM      lineitem_orders lo, orderday od
WHERE     od.orderdate = lo.orderdate      AND od.orderyearkey IN (1992, 1993, 1994)
GROUP BY lo.custkey;

INSERT INTO C7 (custkey, custname, stddev1, stddev2,          /* nach Regel 1 und Regel 4 */
                turnover1992, turnover1993, turnover1994)
SELECT    C4.custkey, cu.custname, C56.stddev1, C56.stddev2,
          C4.turnover1992, C4.turnover1993, C4.turnover1994
FROM      C4,   C56,   customer cu
WHERE     C56.custkey = C4.custkey         AND C56.custkey = cu.custkey;

INSERT INTO C8 (custkey, custname, stddev1, stddev2,          /* unverändert */
                turnover1992, turnover1993, turnover1994)
SELECT    C7.custkey,      C7.custname,      C7.stddev1,      C7.stddev2,      C7.turnover1992,
C7.turnover1993,
          C7.turnover1994
FROM      C7
WHERE     C7.stddev2 <= 0.66794004454646;
```

Abbildung 29: Modifizierte Anfragesequenz C (Modifikation 2)

Der Einflussbereich der Restrukturierung von Anfragesequenzen ist auf die bearbeitete Anfragesequenz selbst beschränkt. Dies hat den Vorteil, dass andere Anwendungen oder die Bearbeitung weiterer Anfragesequenzen durch Restrukturierungsentscheidungen des DSS-Optimierers nicht negativ beeinflusst werden können. Der DSS-Optimierer kann also rein lokale Entscheidungen auf Grundlage der zur Verfügung stehenden Information treffen. Darüber hinaus ist der DSS-Optimierer die am besten geeignete Systemkomponente für die Sequenzrestrukturierung, da diese hier unabhängig von den einzelnen Anwendungen erfol-

gen kann und die Restrukturierungsregeln auf Grund der allgemeinen Optimierungsheuristiken, auf denen sie basieren, auch für verschiedene DWDBS angewandt werden können.

Ein Nachteil der Sequenzrestrukturierung im Vergleich zu den zuvor beschriebenen Optimierungsansätzen soll hier allerdings nicht unerwähnt bleiben. Während für die Ansätze, die auf der Verwendung von Zugriffspfaden, Statistiken oder materialisierten Sichten beruhen, weitgehende Vorarbeiten vorliegen, ist die Menge der für Anfragesequenzen sinnvollen Restrukturierungsregeln sowie die zugehörige Regelverarbeitung noch nicht näher untersucht worden. Insbesondere ist nicht klar, welche Unterschiede bezüglich der Anfragerestrukturierung für einzelne Anfragen hier berücksichtigt werden müssen und wie bei alternativen Restrukturierungsmöglichkeiten eine geeignete ausgewählt werden kann. Trotzdem kann auf Basis der Sequenzrestrukturierung eine geeignete Strategie für einen DSS-Optimierer aufgebaut werden. Diese soll zum Abschluss dieses Abschnitts nochmals kurz skizziert werden.

Optimierungsansatz: Restrukturierung von Anfragesequenzen

Analysiere die Anfragesequenz und wende eine oder mehrere Restrukturierungsregeln darauf an. Die restrukturierte Anfragesequenz wird schließlich zur Ausführung gebracht.

Modifiziere die CREATE- und die INSERT-Anweisungen von Teilanfragen der Anfragesequenz $S = (a_1, a_2, ..., a_n)$ mit Hilfe von Regeln zur Sequenzrestrukturierung.

6.2.7 Zusammenfassung einer Anfragesequenz zu einer Einzelanfrage

Im vorangegangenen Abschnitt wurden verschiedene Möglichkeiten der Restrukturierung von Anfragesequenzen betrachtet, deren Ergebnis wieder eine Anfragesequenz bestehend aus mehreren Teilanfragen ist. In diesem Abschnitt wird ein Spezialfall davon gesondert behandelt. In der Regel können die Teilanfragen einer Sequenz systematisch in eine einzelne SQL-Anweisung integriert werden. Das Ergebnis dieser Transformation ist somit eine Anfragesequenz mit genau einer Teilanfrage. Es wird im Folgenden als *Einzelanfrage* bezeichnet.

Um zu überprüfen, ob es sich hierbei um einen weiteren, für den DSS-Optimierer geeigneten Optimierungsansatz handelt, muss zunächst festgestellt werden, wie die Einzelanfrage erstellt werden kann. Es handelt sich hierbei um ein Problem, das mit der Expansion von Sichten in der Anfrageverarbeitung vergleichbar ist. Es kann also auf vorhandene Technologie zurückgegriffen werden. Eine mögliche Vorgehensweise für die Erstellung der Einzelanfrage ist das Ersetzen aller Verweise auf temporäre Tabellen in der FROM-Klausel der letzten Teilanfrage der Sequenz durch die Definition der jeweiligen temporären Tabelle. Für die Anfragesequenz $S = (a_1, a_2, .., a_n)$ wird also wie folgt vorgegangen:

> Beginne mit der vorletzten Teilanfrage a_{n-1} der Sequenz. Ersetze in der FROM-Klausel von i_n alle Verweise auf a_{n-1} durch die SELECT-Anweisung von i_{n-1} und eine entsprechende Benennung der temporären Tabelle. Fahre mit a_{n-2} fort und ersetzte in der neuen Version von i_n alle Verweise auf a_{n-2} entsprechend. Setzte dies fort bis a_1 erreicht ist. Am Ende ergibt sich eine neue Teilanfrage a_n, die allein ausreicht, um das Ergebnis der ursprünglichen Anfragesequenz bereitzustellen.

Ein Beispiel für eine Einzelanfrage, die auf diese Weise erstellt wurde, ist in Abbildung 30(a) und Abbildung 30(b) angegeben. Die Abbildungen enthalten die Einzelanfrage zu *Anfragesequenz C*. Werden komplexere SQL-Anweisungen zugelassen, als die in Abschnitt 6.1.1 beschriebenen, so kann eine Alternative zu der beschriebenen Vorgehensweise betrachtet werden. Mit Hilfe der WITH-Klausel in SQL:1999 können die Definitionen der einzelnen Teilanfragen einer Anfragesequenz vorab angegeben werden und schließlich für die eigentliche Anfrage verwendet werden [Iso99] [MS02]. Die auf diese Art und Weise definierte Einzelanfrage hat gegenüber der in den Abbildungen 30(a) und 30(b) angegebenen Version den Vorteil, dass mehrfach verwendete Teilanfragen nicht mehrfach angegeben werden müssen. Darüber hinaus werden die temporären Tabellen bei dieser Version automatisch nach Bearbeitung der Anfrage gelöscht. Da sich die beiden Vorgehensweisen hinsichtlich der Bewertung des Optimierungsansatzes nicht weiter unterscheiden, wird hier nur die auf der im Anhang angegebenen Syntax beruhende Variante betrachtet. Basieren alle Teilanfragen einer Sequenz auf dieser Syntax, so hat dies darüber hinaus den Vorteil, dass jede Anfragesequenz, in der maximal die letzte Teilanfrage eine ORDER-BY-Klausel enthält, nach dem beschriebenen Verfahren in eine Einzelanfrage umgewandelt werden kann. Dies ergibt sich aus der Tatsache, dass jede der Syntax entsprechend formulierte SQL-Anweisung ohne ORDER-BY-Klausel als Unteranfrage in der

```
INSERT INTO C8 (custkey, custname, stddev1, stddev2, turnover1992, turnover1993,
turnover1994)
SELECT C7.custkey, C7.custname, C7.stddev1, C7.stddev2, C7.turnover1992, C7.turnover1993,
       C7.turnover1994
FROM   (SELECT cu.custkey, cu.custname, C5.stddeviation, C6.stddeviation,
               C1.turnover1992, C2.turnover1993, C3.turnover1994
        FROM   (SELECT lo.custkey, SUM(lo.endprice)
                FROM   lineitem_orders lo, orderday od
                WHERE  od.orderdate = lo.orderdate AND od.orderyearkey = 1992
                GROUP BYlo.custkey) AS C1 (custkey, turnover1992),
                (SELECT lo.custkey, SUM(lo.endprice)
                FROM   lineitem_orders lo, orderday od
                WHERE  od.orderdate = lo.orderdate AND od.orderyearkey = 1993
                GROUP BYlo.custkey) AS C2 (custkey, turnover1993),
                (SELECT  lo.custkey, SUM(lo.endprice)
                FROM   lineitem_orders lo, orderday od
                WHERE  od.orderdate = lo.orderdate AND od.orderyearkey = 1994
                GROUP BY lo.custkey) AS C3 (custkey, turnover1994),
                (SELECT  lo.custkey, STDDEV(lo.endprice)
                FROM   lineitem_orders lo, orderday od,
                       (SELECT C1.custkey, C1.turnover1992, C2.turnover1993,
                        C3.turnover1994
                       FROM   (SELECT  lo.custkey, SUM(lo.endprice)
                              FROM   lineitem_orders lo, orderday od
                              WHERE  od.orderdate = lo.orderdate
                              AND    od.orderyearkey = 1992
                              GROUP BY lo.custkey)
                                     AS C1 (custkey, turnover1992),
                              (SELECT  lo.custkey, SUM(lo.endprice)
                              FROM   lineitem_orders lo, orderday od
                              WHERE  od.orderdate = lo.orderdate
                              AND    od.orderyearkey = 1993
                              GROUP BY lo.custkey)
                                     AS C2 (custkey, turnover1993),
                              (SELECT  lo.custkey, SUM(lo.endprice)
                              FROM   lineitem_orders lo, orderday od
                              WHERE  od.orderdate = lo.orderdate
                              AND    od.orderyearkey = 1994
                              GROUP BY lo.custkey)
                                     AS C3 (custkey, turnover1994)
                       WHERE  C1.custkey = C2.custkey
                       AND    C1.custkey = C3.custkey
                       AND    C1.turnover1992 >= 500000
                       AND    C2.turnover1993 >= 500000
                       AND    C3.turnover1994 >= 500000)
                       AS C4 (custkey, turnover1992, turnover1993, turnover1994)
```

Abbildung 30(a): Einzelanfrage zu Anfragesequenz C

```
                WHERE    od.orderdate = lo.orderdate
                AND      od.orderyearkey IN (1992, 1993, 1994)
                AND      lo.custkey = C4.custkey
                GROUP BY lo.custkey) AS C5 (custkey, stddeviation),
            (SELECT lo.custkey, STDDEV(lo.endprice) / AVG(lo.endprice)
            FROM     lineitem_orders lo, orderday od,
                     (SELECT   C1.custkey, C1.turnover1992, C2.turnover1993,
                               C3.turnover1994
                     FROM      (SELECT   lo.custkey, SUM(lo.endprice)
                               FROM      lineitem_orders lo, orderday od
                               WHERE     od.orderdate = lo.orderdate
                               AND       od.orderyearkey = 1992
                               GROUP BYlo.custkey)
                                    AS C1 (custkey, turnover1992),
                               (SELECT lo.custkey, SUM(lo.endprice)
                               FROM      lineitem_orders lo, orderday od
                               WHERE     od.orderdate = lo.orderdate
                               AND       od.orderyearkey = 1993
                               GROUP BY lo.custkey)
                                    AS C2 (custkey, turnover1993),
                               (SELECT lo.custkey, SUM(lo.endprice)
                               FROM      lineitem_orders lo, orderday od
                               WHERE     od.orderdate = lo.orderdate
                               AND       od.orderyearkey = 1994
                               GROUP BY lo.custkey)
                                    AS C3 (custkey, turnover1994)
                     WHERE     C1.custkey = C2.custkey
                     AND       C1.custkey = C3.custkey
                     AND       C1.turnover1992 >= 500000
                     AND       C2.turnover1993 >= 500000
                     AND       C3.turnover1994 >= 500000)
                          AS C4 (custkey, turnover1992, turnover1993, turnover1994)
            WHERE    od.orderdate = lo.orderdate
            AND      od.orderyearkey IN (1992, 1993, 1994)
            AND      lo.custkey = C4.custkey
            GROUP BY lo.custkey) AS C6 (custkey, stddeviation), customer cu
        WHERE    C5.custkey = C1.custkey
        AND      C5.custkey = C2.custkey      AND C5.custkey = C3.custkey
        AND      C5.custkey = C6.custkey      AND C5.custkey = cu.custkey)
            AS C7 (custkey, custname, stddev1, stddev2, turnover1992, turnover1993,
                   turnover1994)
WHERE  C7.stddev2 <= 0.66794004454646;
```

Abbildung 30(b): Einzelanfrage zu Anfragesequenz C (Fortsetzung)

FROM-Klausel verwendet werden kann. Unter Umständen erstellen Werkzeuge aber auch Anfragesequenzen, die nicht von vornherein vollständig vorliegen. Dies kann z.B. bedeuten, dass die Sequenz zunächst bis zu einer bestimmten Tei-

lanfrage erstellt und abgearbeitet wird. Die Ergebnisse dieser ersten Teilsequenz beeinflussen dann Details der Teilanfragen in der restlichen Sequenz. Auf diese Art und Weise können beispielsweise Rang-Kriterien berechnet und in die zweite Teilsequenz eingefügt werden. Im Sinne dieser Arbeit handelt es sich hierbei dann allerdings um zwei getrennte Anfragesequenzen. Die erste Sequenz ist die Folge von SQL-Anweisungen, die von einem Werkzeug in einem Block erstellt werden. Die nach erfolgreicher Bearbeitung dieser ersten Sequenz generierten Anweisungen stellen die zweite, separat zu verarbeitende Anfragesequenz dar.

Vergleicht man die erstellte Einzelanfrage zur *Anfragesequenz C* in den Abbildungen 30(a) und 30(b) mit der ursprünglichen Anfragesequenz, so stellt sich die Einzelanfrage als wesentlich komplexer dar. Dies betrifft allerdings zunächst nur die Formulierung der Anfrage. Trotz dieser umfangreicheren Formulierung ist es möglich, dass die Einzelanfrage wesentlich effizienter ausgeführt wird als die Anfragesequenz. Hierfür sind zwei Argumente wesentlich. Erstens ist bei der Einzelanfrage weniger Aufwand für die Erstellung der temporären Tabellen notwendig. Zweitens hat der Anfrageoptimierer des DWDBS bei der Einzelanfrage mehr Möglichkeiten, diese zu optimieren. Dies ergibt sich aus der Tatsache, dass dem Anfrageoptimierer die Einzelanfrage vollständig zur Optimierung vorliegt, während er bei einer Anfragesequenz immer nur Zugriff auf die gerade auszuführende Teilanfrage hat. Damit kann der Anfrageoptimierer für die Einzelanfrage potenziell den optimalen Ausführungsplan erstellen und die minimale Ausführungszeit für die Anfrage erreichen. Wird dagegen die zugehörige Anfragesequenz ausgeführt, dann ist die bestmögliche Ausführungszeit diejenige, die erreicht wird, wenn für jede einzelne Teilanfrage der optimale Ausführungsplan gefunden wird. Hierbei werden natürlich alle anderen in diesem Kapitel erläuterten Optimierungsansätze nicht berücksichtigt. Der im Folgenden kurz charakterisierte Optimierungsansatz weist also ein erhebliches Potenzial zur effizienteren Ausführung von Anfragesequenzen auf.

Optimierungsansatz: Erstellung einer Einzelanfrage

Verbinde alle Teilanfragen einer Anfragesequenz zu einer einzigen Anfrage und führe diese aus.

Wie schon bei der Restrukturierung von Anfragesequenzen beschrieben, hat auch dieser Optimierungsansatz den Vorteil, dass als Grundlage für seine Anwendung die gegebene Anfragesequenz ausreicht und dass durch die Anwendung keine Auswirkungen über die Ausführung der Anfragesequenz hinaus zu erwarten sind. Darüber hinaus lässt sich der Ansatz auf der Basis existierender Technologie sehr einfach umsetzen. Aus diesen Gründen handelt es sich um einen Optimierungsansatz, der optimal innerhalb eines DSS-Optimierers realisiert werden kann.

6.2.8 Bewertung der Optimierungsansätze

Die in diesem Kapitel betrachteten Optimierungsansätze für einen DSS-Optimierer wurden in den vorangegangenen Abschnitten bereits ausführlich erläutert und analysiert. Hier wird das Ergebnis dieser qualitativen Bewertung nochmals übersichtlich zusammengestellt, so dass ersichtlich wird, welche der Ansätze besser für einen DSS-Optimierer geeignet sind und welche erhebliche Nachteile aufweisen. Hierzu werden in Tabelle 6 die einzelnen Ansätze aufgeführt und die Bewertung gemäß der in Abschnitt 6.2 eingeführten Kriterien vorgenommen.

Tabelle 6: Qualitative Bewertung der Optimierungsansätze

Optimierungsansatz	**Technologie-Verfügbarkeit**	**Optimierungs-potenzial**	**Einflussbereich**	**Entscheidungs-grundlage**	**Umsetzung im DSS-Optimierer**
Intra-Anfragesequenz-Parallelität	+	+	+	0	+
Bereitstellung von Statistiken	+	+	0	0	+
Bereitstellung von Zugriffspfaden	+	+	-	-	-
Bereitstellung materialisierter Sichten	+	+	-	-	-
Partitionierung der Basistabellen	+	-	-	-	-
Restrukturierung der Anfragesequenzen	0	+	+	+	+
Erstellung einer Einzelanfrage	+	+	+	+	+

Da die Tabelle lediglich einen Überblick darüber geben soll, wie die einzelnen Optimierungsansätze einzuordnen sind, beschränkt sich die Bewertung hier auf drei Stufen: '+' wird verwendet, wenn ein Optimierungsansatz sich bezüglich eines Kriteriums als besonders geeignet erwiesen hat, '-' bedeutet, dass der Ansatz mit erheblichen Nachteilen verbunden ist und '0' wird in der Tabelle angegeben, wenn ein Optimierungsansatz bezüglich eines Kriteriums keine Besonderheiten aufweist.

Die Zusammenstellung in Tabelle 6 zeigt, dass alle Optimierungsansätze zumindest nach der qualitativen Betrachtung ein erhebliches *Optimierungspotenzial* aufweisen. In der Regel ist auch die notwendige *Technologie* zur Realisierung der Ansätze vorhanden. In erster Linie gibt es bei der Restrukturierung von Anfragesequenzen noch offene Fragen hinsichtlich der zu berücksichtigenden Restrukturierungsregeln und deren Realisierung. Beim Kriterium *Einflussbereich* wurde betrachtet, in wieweit die einzelnen Optimierungsansätze andere Anwendungen und die Ausführung anderer Anfragesequenzen auf einem Data Warehouse negativ beeinflussen können. Hier ergibt sich ein sehr heterogenes Bild bei der Bewertung. Während einige Ansätze nur die einzelne Anfragesequenz und deren Ausführung betreffen, sind bei anderen erhebliche Auswirkungen auf das Data Warehouse zu erwarten. Dies ist insbesondere dann der Fall, wenn, wie bei Zugriffspfaden oder materialisierten Sichten, Änderungen am physischen Schema des Data Warehouse vorgenommen werden. Ähnlich unterschiedlich fällt die Bewertung in Bezug auf die *Entscheidungsgrundlage* aus, die ein DSS-Optimierer zur Realisierung eines Optimierungsansatzes hat. Für einzelne Ansätze ist hier lediglich die gegebene Anfragesequenz relevant, während andere weitergehende Kenntnisse bezüglich des DWDBS und dessen Schema voraussetzen.

Gerade die beiden zuletzt genannten Kriterien weisen bei einzelnen Optimierungsansätzen darauf hin, dass der DSS-Optimierer nicht die am besten geeignete Systemkomponente zu deren Realisierung ist. Insbesondere sollten Ansätze, die eine Änderung des physischen Schemas im Data Warehouse nach sich ziehen und in der Regel umfangreiche Informationen über das zu Grunde liegende Datenbanksystem voraussetzen, besser als Bestandteil der Administration des Data Warehouse berücksichtigt werden.

6.3 Leistungsbewertung der Optimierungsansätze

Im vorangegangenen Abschnitt wurden alle relevanten Optimierungsansätze vorgestellt und qualitativ bewertet. In diesem Abschnitt folgt nun eine quantitative Bewertung, die zeigen soll, ob das theoretisch in den einzelnen Ansätzen vorhandene Optimierungspotenzial auch genutzt werden kann. Hierzu werden die in Abschnitt 6.1.2 vorgestellten exemplarischen Anfragesequenzen, unter Berücksichtigung der verschiedenen Optimierungsansätze, ausgeführt. Im folgenden Abschnitt wird zunächst die Umgebung erläutert, in der die Messungen durchgeführt wurden. Die Darstellung der Messergebnisse erfolgt in Abschnitt 6.3.2, während diese in den letzten beiden Abschnitten bewertet und in Bezug auf eine Realisierung eines DSS-Optimierers analysiert werden.

6.3.1 Umgebung für die Leistungsmessungen

Wie bereits erläutert, wurden für die Messungen unter anderem die in Abschnitt 6.1.2 vorgestellten Anfragesequenzen verwendet. Sie beruhen auf den mehr als 20 Anwendungsszenarien, die in [Wag00] beschrieben sind. Von diesen wurden für eine erste *Messreihe* sechs Anwendungsszenarien ausgewählt und die jeweils zugehörigen Anfragesequenzen erstellt. Auf der Grundlage der Erkenntnisse konnten schließlich drei der Anfragesequenzen für eine detailliertere zweite Messreihe ausgewählt werden. Bei der Auswahl wurde unter anderem berücksichtigt, dass sich die Anfragesequenzen in ihrer Komplexität deutlich unterscheiden. Alle Anfragesequenzen wurden auf dem in Abschnitt 6.1.2 bereits erläuterten Strategy-Schema der ToPCHain ausgeführt. Diesem liegt das Schema des TPC-H-Benchmarks zu Grunde. Zu diesem Benchmark gehört auch ein Werkzeug, mit dessen Hilfe Daten für dieses Schema generiert werden können [Tra99]. Mit diesem wurden die im Rahmen dieser Arbeit verwendeten Daten erstellt. Die folgende Tabelle 7 beschreibt das Mengengerüst der für die Messungen relevanten Tabellen. Angegeben ist jeweils die Anzahl der Tupel pro Tabelle. Für die erste Messreihe wurden die Daten mit dem Skalierungsfaktor 1 erzeugt, während für die zweite Messreihe der Faktor 10 verwendet wurde. Beim Skalierungsfaktor 10 belegen die Basisdaten insgesamt ungefähr 10 GB externen Speicher, wobei vorhandene Zugriffspfade nicht eingerechnet sind.

Als Datenbanksystem wurde für die Messungen IBM DB2 Universal Database, Version 7.1 verwendet. Für die Erfassung der Anfragelaufzeiten wurde ein zum Datenbanksystem gehörendes Werkzeug (db2batch) verwendet [Ibm01b]. Die

Tabelle 7: Mengengerüst des Strategy-Schemas

Tabelle im Strategy-Schema	Anzahl der Tupel mit Skalierungsfaktor 1	Anzahl der Tupel mit Skalierungsfaktor 10
Lineitem_Orders	6001215	59986052
Order	1500000	15000000
Part	200000	2000000
Supplier	10000	100000
Customer	150000	1500000
Shipday	2526	2526
Orderday	2406	2406
Ordermonth	80	80
...		

Serverumgebung, auf der die Anfragesequenzen ausgeführt wurden, bestand aus einer Sun Enterprise 4500 mit 12 Prozessoren und insgesamt 12 GB Hauptspeicher. Davon wurden aber lediglich 2 GB für den Puffer des Datenbanksystems genutzt. Somit konnte auch die umfangreichste der verwendeten Tabellen nicht vollständig im Hauptspeicher abgelegt werden und eine in dieser Hinsicht realistische Laufzeitumgebung geschaffen werden. Während der Messungen selbst wurden alle anderen Aktivitäten auf der Maschine, die die Anfragelaufzeiten potenziell beeinflussen können, unterbunden. Intra-Anfrageparallelität wurde bei allen Messungen zugelassen und vom Datenbanksystem auch genutzt.

Jede der beiden Messreihen bestand aus einer Vielzahl von Messungen. In diesem Zusammenhang wird unter einer *Messung* das Ausführen einer einzelnen Anfragesequenz unter Berücksichtigung eines oder mehrerer Optimierungsansätze verstanden. Allein in der zweiten Messreihe wurden mehr als 300 solcher Messungen durchgeführt. Die einzelnen Messungen wurden zu sogenannten *Messgruppen* zusammengefasst. Eine Messgruppe bestand jeweils aus drei unmittelbar aufeinanderfolgenden, identischen Messungen. Der einzige Unterschied innerhalb einer Messgruppe bestand darin, dass nur vor der ersten Messung der Datenbankpuffer geleert wurde. Hiermit sollte gezeigt werden, in wieweit das Vorhandensein von Daten im Puffer die Laufzeiten der beiden Wie-

derholungsmessungen innerhalb einer Messgruppe beeinflusst. Die einzelnen Messgruppen wurden mindestens dreimal, in der Regel aber häufiger, ausgeführt. Hierdurch konnte gezeigt werden, dass die erzielten Ergebnisse stabil und reproduzierbar sind. Die Ergebnisse einer Messgruppe wurden nur dann in die Bewertung mit einbezogen, wenn alle drei Messungen der Gruppe durchgeführt wurden, ohne dass in den während der Messung protokollierten Informationen irgendwelche Unregelmäßigkeiten auftraten. So wurde beispielsweise das Ergebnis der kompletten Messgruppe verworfen, wenn es nur bei einer der enthaltenen Messungen zu erkennbaren Problemen des I/O-Systems und den damit verbundenen Verzögerungen von Teilanfragen kam.

6.3.2 Ergebnisse der Leistungsmessungen

Die Messergebnisse der ersten Messreihe sind in [Wag00] angegeben. Anhand der Messungen für sechs unterschiedliche Anfragesequenzen konnte die prinzipielle Eignung der Optimierungsansätze gezeigt sowie die unterschiedliche Komplexität der einzelnen Anfragesequenzen bewertet werden. Diese Informationen bildeten die Grundlage für die Auswahl der wichtigen Anwendungsszenarien und der damit verbundenen Anfragesequenzen für die zweite Messreihe. In der ersten Messreihe wurden auch unterschiedliche Parametereinstellungen des Anfragegenerators berücksichtigt. Obwohl diese generell einen Einfluss auf die Laufzeit der generierten Anfragesequenzen haben können, konnte gezeigt werden, dass dies bei den betrachteten Anwendungsszenarien nicht der Fall ist. Entsprechend wurden diese Möglichkeiten im Rahmen der zweiten Messreihe nicht mehr berücksichtigt.

Die relevanten Messergebnisse der zweiten Messreihe sind in Tabelle 8 zusammengefasst. Angegeben ist jeweils die Laufzeit der einzelnen Anfragesequenzen unter Berücksichtigung eines Optimierungsansatzes. Die Angabe der absoluten Laufzeit erfolgt in Sekunden, die relative Angabe bezieht sich jeweils auf die Ausführung der ursprünglichen Sequenz, für die in Zeile 1 jeweils die Laufzeiten aufgeführt sind. Jede der Tabellenzellen enthält die Durchschnittslaufzeit für die Messungen der am schnellsten ausgeführten Messgruppe. Da sich die Laufzeiten bei vorab geleertem Puffer nicht signifikant von denjenigen unterschieden haben, bei denen bereits Daten im Puffer enthalten waren, werden die Ergebnisse hier nicht gesondert angegeben.

Tabelle 8: Übersicht der Messergebnisse der zweiten Messreihe

	Optimierungsansatz	**Anfrage-sequenz A**		**Anfrage-sequenz B**		**Anfrage-sequenz C**	
		Laufzeit[1]	Anteil[2]	Laufzeit[1]	Anteil[2]	Laufzeit[1]	Anteil[2]
1	Anfragesequenz	4.877	100	498	100	30.209	100
2	Anfragesequenz Modifikation 1	-	-	-	-	4.188	14
3	Anfragesequenz Modifikation 2	-	-	-	-	2.296	8
4	Einzelanfrage	2.834	58	-	-	10.734	36
5	Einzelanfrage zu Modifikation 1	-	-	-	-	3.853	13
6	Einzelanfrage zu Modifikation 2	-	-	-	-	4.900	16
7	Intra-Anfragesequenz-Parallelität	3.939	81	-	-	29.004	96
8	Bereitstellung der Statistik	3.109	64	-	-	5.263	17
9	Bereitstellung der Zugriffspfade	5.428	111	25	5	-	-
10	Partitionierung	3.712	76	26	5	27.984	93

[1] Absolute Anfragelaufzeiten in Sekunden
[2] Anteilige Anfragelaufzeit bezogen auf die Laufzeit der Anfragesequenz in Zeile 1

Nicht alle Optimierungsansätze sind auf jede der betrachteten Anfragesequenzen anwendbar. Darum sind einige Felder der Tabelle leer. Darüber hinaus konnte die Verwendung von materialisierten Sichten in der beschriebenen Umgebung nicht gewinnbringend eingesetzt werden. Die angelegten materialisierten Sichten wurden bei der Verarbeitung der Anfragesequenzen nicht eingesetzt, so dass sich gegenüber der Ausführung der ursprünglichen Sequenz keine Leistungsverbesse-

rungen ergeben haben. Im Rahmen der zweiten Messreihe konnten die folgenden Optimierungsansätze berücksichtigt werden:

- *Restrukturierung der Anfragesequenzen*: In den Zeilen 2 und 3 sind die Ergebnisse für zwei unterschiedliche Modifikationen der ursprünglichen Anfragesequenz (in Zeile 1) angegeben.
- *Erstellung einer Einzelanfrage*: Hierzu sind Ergebnisse in den Zeilen 4 bis 6 angegeben. Zeile 4 ist hierbei die aus der ursprünglichen Anfragesequenz erstellte Einzelanfrage. In den Zeilen 5 und 6 sind jeweils die Ergebnisse für Einzelanfragen, die aus den modifizierten Anfragesequenzen entstanden sind, angegeben.
- *Intra-Anfragesequenz-Parallelität*: Die Ergebnisse hierzu sind in Zeile 7 enthalten. Hierbei wurde die ursprüngliche Anfragesequenz verwendet und um Intra-Anfragesequenz-Parallelität ergänzt. Hier sei nochmals betont, dass Parallelverarbeitung innerhalb der einzelnen Anfragen (Intra-Anfrage-Parallelität) bei der gesamten Messreihe zugelassen wurde.
- *Bereitstellung der Statistik*: Die Ergebnisse für die Ausführung der ursprünglichen Anfragesequenz, unter Bereitstellung zusätzlicher Statistiken für temporäre Tabellen, ist in Zeile 8 angegeben.
- *Bereitstellung der Zugriffspfade*: Hier wurde wiederum die ursprüngliche Anfragesequenz verwendet. In Zeile 9 sind die Ergebnisse für deren Ausführung unter Berücksichtigung zusätzlicher Zugriffspfade angegeben.
- *Partitionierung der Basistabellen*: Bei der Verwendung dieses Optimierungsansatzes ergeben sich die in Zeile 10 dargestellten Laufzeiten.

Die meisten der vorgestellten Optimierungsansätze können auch kombiniert eingesetzt werden. Der Fokus dieser Arbeit ist es allerdings, das prinzipielle Optimierungspotenzial der einzelnen Ansätze aufzuzeigen, und damit deren Eignung für einen DSS-Optimierer zu untermauern. Die Kombination von Optimierungsansätzen bringt in dieser Hinsicht keine zusätzlichen Erkenntnisse und wurde bei den durchgeführten Messungen darum nicht systematisch berücksichtigt. Eine Ausnahme gibt es allerdings. Für die *Anfragesequenz C* wurde auch die Sequenzrestrukturierung in Kombination mit der Erstellung von Einzelanfragen betrachtet. Für alle durchgeführten Messungen erfolgt die Interpretation der erzielten Ergebnisse im nächsten Abschnitt.

6.3.3 Bewertung der Messergebnisse

Folgende Kernaussagen können aus den Messergebnissen in Tabelle 8 abgelesen werden:

(1) Die ursprüngliche Anfragesequenz stellt in keinem der betrachteten Fälle die am effizientesten ausführbare Form der Benutzeranfrage dar.

(2) Alle betrachteten Optimierungsansätze können zur effizienteren Verarbeitung von Anfragesequenzen eingesetzt werden.

(3) Die Anwendung eines beliebigen Optimierungsansatzes bei einer Anfragesequenz führt nicht in jedem Fall zu einer effizienteren Verarbeitung.

(4) Kein einzelner der betrachteten Optimierungsansätze hat bei allen Anfragesequenzen gleichermaßen die effizienteste Ausführung zur Folge.

Die erste Aussage ergibt sich daraus, dass für jede der betrachteten Anfragesequenzen mit Hilfe der Optimierungsansätze eine Verkürzung der Laufzeit erreicht werden konnte. So konnte beispielsweise bei den Anfragesequenzen *A* und *C* rein durch die Bereitstellung von Statistiken eine effizientere Bearbeitung erreicht werden. Bei *Anfragesequenz B* hat sich durch die Bereitstellung geeigneter Zugriffspfade die Ausführungszeit um eine Größenordnung verkürzt.

Die zweite Aussage ergibt sich daraus, dass es für jeden der Optimierungsansätze eine, in der Regel aber mehrere Messungen gibt, bei denen eine effizientere Ausführung der Anfragesequenz erzielt werden konnte. Dies bedeutet unter anderem, dass keiner der Ansätze bei der Ausführung von Anfragesequenzen grundsätzlich ausgeklammert werden sollte.

Aussage 3 kann insbesondere an zwei unterschiedlichen Optimierungsansätzen gezeigt werden. Einerseits führt die Bereitstellung von Zugriffspfaden bei der *Anfragesequenz A* zu einer Verlängerung der Laufzeit. Die Zugriffspfade wurden bei der Anfrageverarbeitung zwar verwendet, auf Grund der fehlenden statistischen Informationen für die temporären Tabellen wurden aber die Kosten für den gewählten Zugriffsplan nicht korrekt geschätzt. Andererseits tritt eine Verlängerung der Laufzeit auch bei der *Anfragesequenz C* auf, wenn aus der zweiten Modifikation der Sequenz eine Einzelanfrage erstellt wird. Die Problematik sehr komplexer Einzelanfragen wird im Zusammenhang mit der vierten Aussage diskutiert.

Die vierte Aussage schließlich wird beispielsweise bei einem Vergleich der Ergebnisse für die Anfragesequenzen *A* und *C* deutlich. Während bei der *Anfragesequenz A* die effizienteste Ausführung durch die Erstellung einer Einzelanfrage erzielt wird, ergibt sich die optimale Laufzeit für *Anfragesequenz C* durch eine Restrukturierung der Anfragesequenz. Dies hat zur Folge, dass eine Realisierung eines der Optimierungsansätze allein nicht gewinnbringend ist. Vielmehr muss für die einzelne Anfragesequenz eine Entscheidung getroffen werden, welche Optimierungsansätze verfolgt werden sollen und welche nicht. Diese vierte Aussage beinhaltet insbesondere auch, dass es nicht immer optimal ist, eine Anfragesequenz zu einer Einzelanfrage zusammenzufassen. Dies wird am Beispiel der *Anfragesequenz C* besonders deutlich. Während die Einzelanfrage die Laufzeit lediglich auf ein Drittel des ursprünglichen Wertes reduzieren kann, wird durch eine Restrukturierung der Anfragesequenz eine Verkürzung um eine Größenordnung erreicht.

Tabelle 9: Charakteristische Eigenschaften der Ausführungspläne

Optimierungsansatz	**Anfragesequenz A**		**Anfragesequenz C**	
	Join-Operationen	**Sortieroperationen**	**Join-Operationen**	**Sortieroperationen**
Anfragesequenz	max. 5	max. 10	max. 6	max. 8
Anfragesequenz, Modifikation 1	-	-	max. 3	max. 3
Anfragesequenz, Modifikation 2	-	-	max. 4	max. 3
Einzelanfrage	7	14	33	37
Einzelanfrage zu Modifikation 1	-	-	13	14
Einzelanfrage zu Modifikation 2	-	-	16	16

Eine detailliertere Analyse des vom Datenbanksystem gewählten Ausführungsplans zeigt, dass dieses Verhalten vermutlich an der Komplexität der Einzelanfrage liegt. Der für die in den Abbildungen 30(a) und 30(b) wiedergegebene Anfrage gewählte Plan enthält unter anderem 33 Join-Operatoren und 37 Sortierungen bei insgesamt mehr als 200 Operatoren. Im Rahmen dieser Messungen handelte es sich hierbei um den komplexesten Ausführungsplan. Für einige Vari-

anten der Anfragesequenzen sind die entsprechenden Werte in Tabelle 9 angegeben. Bei den Anfragesequenzen sind jeweils die Maximalwerte über die einzelnen Teilanfragen berücksichtigt.

6.4 Konsequenzen für die Realisierung eines DSS-Optimierers

Nach der sowohl qualitativen als auch quantitativen Bewertung der einzelnen Ansätze, die für die Optimierung von Anfragesequenzen eingesetzt werden können, soll in diesem Abschnitt erläutert werden, welche Konsequenzen sich aus dieser Bewertung für die Realisierung eines DSS-Optimierers ergeben. Der Ausgangspunkt für die Bewertung war die Frage, ob einer oder mehrere der vorgestellten Optimierungsansätze in einem DSS-Optimierer realisiert werden sollten und welche Schwierigkeiten mit dieser Realisierung verbunden sind.

Zunächst kann eine prinzipielle Eignung der vorgestellten Optimierungsansätze festgestellt werden. Die in Abschnitt 6.3.2 aufgeführten Ergebnisse der Leistungsmessungen zeigen, dass alle vorgestellten Optimierungsansätze das Potenzial haben, eine effizientere Ausführung von Anfragesequenzen zu bewirken. Somit wird die qualitative Bewertung aus Abschnitt 6.2.8 durch die Leistungsmessungen bestätigt.

Ein weiteres wichtiges Ergebnis der Leistungsmessungen ist, dass ein Optimierungsansatz allein nicht ausreicht, um eine optimale Ausführung der Anfragesequenzen zu erzielen. Die minimale Ausführungszeit wurde bei den verschiedenen betrachteten Anfragesequenzen jeweils mit Hilfe unterschiedlicher Optimierungsansätze erreicht. Für die Realisierung des DSS-Optimierers ergibt sich daraus, dass dieser mehrere Optimierungsansätze beherrschen und in Abhängigkeit von der gegebenen Anfragesequenz einsetzen muss. Für jede einzelne Anfragesequenz muss der am besten geeignete Ansatz durch den DSS-Optimierer ausgewählt werden. Grundsätzlich kann dies wie bei der gewöhnlichen Anfrageoptimierung auf der Basis von Heuristiken bzw. als Erweiterung auf der Grundlage eines Kostenmodells erfolgen.

Die qualitative Bewertung in Abschnitt 6.2.8 hat darüber hinaus ergeben, dass nicht alle Optimierungsansätze gleichermaßen für eine Realisierung in einem DSS-Optimierer geeignet sind. Insbesondere stellt das Bereitstellen von Zugriffspfaden und materialisierten Sichten sowie die Partitionierung von Basistabellen

einen so erheblichen Eingriff in das physische Schema des Data Warehouse dar, dass diese Optimierungsmöglichkeiten besser im Rahmen der regulären Administration des Data Warehouse und nicht autonom durch den DSS-Optimierer genutzt werden sollten. Auf diese Weise kann verhindert werden, dass eine Vielzahl von Anwendungen dadurch negativ beeinflusst werden, dass Optimierungen für einzelne Anfragesequenzen durchgeführt werden. Es bleiben vier Optimierungsansätze, die für eine Realisierung im Rahmen eines DSS-Optimierers als geeignet angesehen werden können. Diese sind:

- die Bereitstellung von Statistiken,
- die Nutzung von Intra-Anfragesequenz-Parallelität,
- die Erstellung einer Einzelanfrage sowie
- die Restrukturierung der Anfragesequenz.

Bei der ersten der genannten Strategien sorgt der DSS-Optimierer für die Bereitstellung statistischer Informationen für temporäre Ergebnisse, die bei der Verarbeitung einer Anfragesequenz entstehen. Alle kommerziellen Datenbanksysteme erfassen solche statistischen Informationen als Grundlage für die Entscheidungen des Anfrageoptimierers. Der DSS-Optimierer muss darum lediglich dafür sorgen, dass entsprechende Statistiken auch für die temporären Ergebnisse erfasst werden. Da alle relevanten Datenbanksysteme die Schnittstellen zum Anstoßen dieses Prozesses bereitstellen, ist eine einfache Realisierung dieser Strategie prinzipiell möglich. Allerdings ist das Erfassen statistischer Informationen nicht grundsätzlich für alle temporären Ergebnisse gleichermaßen sinnvoll und erfolgversprechend. Vielmehr ist das Erfassen der Statistik unter Umständen mit erheblichem Aufwand verbunden und sollte darum der zu erwartenden Laufzeitverkürzung für die gesamte Anfragesequenz gegenüber gestellt werden. Die hierzu notwendigen Kostenmodelle sind nicht Bestandteil aktueller Datenbanktechnologie und müssen neu entwickelt werden.

Ähnlich sieht es bei den anderen drei genannten Strategien aus. Sowohl für die Nutzung von Intra-Anfragesequenz-Parallelität als auch für die Restrukturierung von Anfragesequenzen und die Erstellung von Einzelanfragen als Strategien eines DSS-Optimierers muss die geeignete Technologie erst entwickelt werden. Für alle drei Bereiche gibt es zwar vergleichbare Optimierungsansätze, die bei der Verarbeitung von einzelnen Anfragen verwendet werden. Im Rahmen eines DSS-Optimierers werden diese allerdings auf einer anderen, höheren Granulari-

tätsstufe verwendet, so dass eine direkte Übertragung der verfügbaren Technologie nicht möglich ist.

Für die Intra-Anfragesequenz-Parallelität ist es notwendig, diejenigen Teile einer Anfragesequenz zu identifizieren, die parallel ausgeführt werden können. Wie in Abschnitt 6.2.1 erläutert, können diese auf der Grundlage eines Abhängigkeitsgraphen ermittelt werden. Dies entspricht der Vorgehensweise, die bei der Parallelisierung einzelner Anfragen gewählt wird. Trotzdem ist eine Weiterentwicklung bestehender Technologie notwendig. Während bei der Parallelisierung einzelner Anfragen Kostenmodelle existieren, um zu entscheiden, welche Operatoren eines Ausführungsplans parallelisiert werden sollten, fehlen solche Modelle für die beschriebene Intra-Anfragesequenz-Parallelität. Die notwendigen Modelle müssen z.B. berücksichtigen, wieviele Prozessoren für die Verarbeitung einer Anfragesequenz zur Verfügung stehen und wie diese von den potenziell zu parallelisierenden Teilanfragen genutzt werden. Darüber hinaus müssen auch die Möglichkeiten zur Parallelisierung der I/O und mögliche Verdrängungseffekte im Bereich des Datenbankpuffers berücksichtigt werden. Ein wesentlicher Unterschied zur Parallelisierung von einzelnen Operatoren eines Ausführungsplans besteht darin, dass die parallel auszuführenden Teile bei der Intra-Anfragesequenz-Parallelität eine große Heterogenität aufweisen können.

Die Erstellung einer Einzelanfrage ist einfach zu realisieren, da hier Techniken der Sichten-Expansion eingesetzt werden können. Allerdings wurde in diesem Kapitel bereits gezeigt, dass die Einzelanfrage nicht immer die am effizientesten ausführbare Form einer Anfragesequenz darstellt. Daraus resultiert ein wichtiger Unterschied dieser Strategie eines DSS-Optimierers zu der in Datenbanksystemen üblichen Sichten-Expansion. Während bei dieser die Expansion der Sichten eine Voraussetzung für die Ausführung einer Anfrage darstellt, ist sie bei Anfragesequenzen optional. Die temporären Ergebnisse können als Sichten erstellt und in der zu erstellenden Einzelanfrage expandiert werden, sie müssen es aber nicht. Bestandteil dieser Strategie ist also die Entscheidung, in welchen Fällen das Erstellen einer Einzelanfrage sinnvoll ist und in welchen nicht. Welche Einflussgrößen hier zu berücksichtigen sind und wie diese zusammenwirken ist bisher nicht gezielt untersucht worden. Insbesondere muss hier auch ermittelt werden, welche dieser Größen vom zu Grunde liegenden Datenbanksystem abhängen und welche unabhängig davon sind.

Auch für die Restrukturierung von Anfragesequenzen existiert eine entsprechende Strategie für einzelne Anfragen. Die Anfragerestrukturierung ist ein in vielen Arbeiten untersuchter und in kommerziellen Systemen umgesetzter Optimierungsansatz. Allerdings führt das unterschiedliche Granulat, auf dem die Restrukturierung ganzer Anfragesequenzen ansetzt, dazu, dass die verfügbare Technologie nicht einfach in diesen Bereich übernommen werden kann. So ist beispielsweise nicht von vornherein klar, welche Regeln zur Anfragerestrukturierung direkt auf Anfragesequenzen übertragen werden können, bei welchen eine Modifikation notwendig ist und welche Restrukturierungsregeln speziell für Anfragesequenzen hinzukommen müssen. Aber auch bei der Anwendung der Restrukturierung müssen die entsprechenden Techniken entwickelt werden. So muss untersucht werden, in wieweit Heuristiken der Anfragerestrukturierung auf Anfragesequenzen übertragbar sind. Für Restrukturierungsregeln, die für Anfragesequenzen spezifisch sind, müssen entsprechenden Heuristiken erarbeitet werden. Darüber hinaus ist zu untersuchen, welche Kostenmodelle bei einer Restrukturierung von Anfragesequenzen gewinnbringend eingesetzt werden können.

7 Zusammenfassung und Ausblick

Ausgangspunkt für diese Arbeit war die Tatsache, dass zur Analyse von Daten in einem Data Warehouse unterschiedliche Techniken zur Verfügung stehen. Die wichtigsten Vertreter sind hier das Online Analytical Processing sowie das Data Mining. In Kapitel 3 dieser Arbeit wurde zunächst gezeigt, dass für eine umfassende Analyse der Daten in einem Data Warehouse beide Analyseansätze verfolgt werden müssen. Nur so können strategierelevante Informationen gewonnen werden. In der Praxis wird im Rahmen von Data-Warehouse-Projekten in der Regel mit dem Einsatz von OLAP begonnen und Data Mining als weiterer Analyseansatz ergänzend hinzugenommen, sobald die Grundfunktionalität des Data Warehouse beherrscht wird und sich Fragestellungen ergeben, die eine tiefergehende Analyse von Mustern und Abhängigkeiten in den Daten notwendig machen. Sowohl OLAP als auch Data Mining sind darüber hinaus seit mehreren Jahren Gegenstand zahlreicher Forschungsarbeiten mit ganz unterschiedlichen Zielrichtungen. Sowohl in diesen Arbeiten als auch in der produktseitigen Umsetzung der Ergebnisse wurden Fragestellungen, die sich aus der Integration von OLAP und Data Mining ergeben, bisher nicht ausreichend fokussiert. Deshalb wurden in der vorliegenden Arbeit eine Reihe wichtiger Fragestellungen, die sich in diesem Zusammenhang ergeben, bearbeitet. Im Folgenden werden die Ergebnisse zusammengefasst und als Ausblick eine Reihe offener Fragen und Probleme erläutert, die in Zukunft einer weiteren Untersuchung bedürfen.

7.1 Zusammenfassung der Ergebnisse

Ein erster Bereich, in dem Integrationsaspekte betrachtet werden müssen, betrifft die Datenmodellierung für ein Data Warehouse. In Kapitel 4 wurde der Einfluss der Datenmodellierung auf dem gleichzeitigen Einsatz von OLAP und Data Mining analysiert. Die Betrachtungen wurden hierbei entsprechend den typischen Phasen des Entwurfsprozesses unterteilt, wobei der Schwerpunkt auf dem konzeptuellen und dem logischen Entwurf lag. In beiden Bereichen wurden existierende Modellierungsansätze analysiert, wobei sich gezeigt hat, dass in bisherigen Arbeiten der Fokus grundsätzlich auf einer Unterstützung von OLAP-Analysen lag. Spezifische Anforderungen aus dem Bereich Data Mining wurden an keiner Stelle berücksichtigt. Im Rahmen dieser Arbeit wurde darum das COCOM-Modell für den konzeptuellen Entwurf entwickelt. Mit diesem konnten die folgenden Ziele erreicht werden: Erstens sollte ein konzeptuelles Modell bereitgestellt werden, das die für OLAP relevanten Modellierungsmöglichkeiten

bietet. Zweitens sollte es das Modell ermöglichen, die Informationen in einer Art und Weise zu strukturieren, die eine gute Grundlage für die unterschiedlichen Data-Mining-Methoden bildet. Drittens sollte für das konzeptuelle Modell eine formale Notation verfügbar sein, so dass aus dieser verschiedene logische Schematypen abgeleitet werden können.

Für den logischen Data-Warehouse-Entwurf gibt es bereits eine breite Palette von Vorschlägen, die sich durchgehend an den Anforderungen aus dem Bereich OLAP orientieren. Alle vorgeschlagenen Schematypen können direkt aus dem COCOM-Modell abgeleitet werden und wurden in Kapitel 4 darum auch aufbauend auf dessen formaler Notation erläutert. Zur Bewertung der Schematypen wurde zunächst ein Kriterienkatalog erstellt, der sowohl Aspekte des OLAP als auch des Data Mining berücksichtigt. Auf dieser Basis konnte das Star-Schema als der logische Schematyp identifiziert werden, der die besten Voraussetzungen für ein Data Warehouse, das integrierte Analysemöglichkeiten unterstützen soll, bietet. Es ermöglicht einerseits die Abbildung der wichtigsten Modellierungsmöglichkeiten des COCOM-Modells und bietet andererseits eine gute Unterstützung hinsichtlich verschiedener Aspekte der Performanz. Insgesamt konnte in Kapitel 4 gezeigt werden, welche Modelle und Schematypen im Rahmen des Data-Warehouse-Entwurfs verwendet werden sollten, wenn die Integration von Data Mining und OLAP im Fokus des Data-Warehouse-Einsatzes liegt.

Den zweiten Schwerpunkt dieser Arbeit bildete die Entwicklung einer Systemarchitektur für die Integration von Data Mining und OLAP. In Kapitel 5 wurden die Systemarchitekturen der wichtigsten Vertreter aus beiden Bereichen zunächst getrennt analysiert und bewertet. In beiden Bereichen weisen die Architekturen der kommerziellen Werkzeuge jeweils eine Reihe von Gemeinsamkeiten auf. In einem ersten Schritt wurde darum eine allgemeine OLAP-Architektur sowie eine allgemeine Data-Mining-Architektur spezifiziert. Bei der detaillierten Analyse hat sich gezeigt, dass diese bezüglich allgemeiner Kriterien zur Beurteilung einer Systemarchitektur, wie z.B. Skalierbarkeit, Lastbalancierung, Wartbarkeit und Portierbarkeit durchgehend positiv zu bewerten sind. Defizite konnten jedoch in zweierlei Hinsicht identifiziert werden. So bieten beide Architekturen weder eine Unterstützung für die anwendungsunabhängige Optimierung von Anfragesequenzen, noch ist durch sie automatisch eine koordinierte Verwaltung der Metadaten über die Grenzen der einzelnen Anwendungen hinweg gewährleistet. Die in Kapitel 5 vorgestellte Architektur für ein integriertes System behebt diese Defizite. Wie in dieser Arbeit gezeigt werden konnte, unter-

stützt die integrierte Architektur den Einsatz von OLAP und Data Mining auf einem Data Warehouse. Dies wird unter anderem durch eine gemeinsame Metadaten-Verwaltung und eine zusätzliche vierte Schicht erreicht. In dieser ist ein sogenannter DSS-Optimierer angesiedelt, der für eine anwendungsunabhängige Optimierung von Anfragesequenzen sorgt. Diese Komponente hat sich als sinnvoll erwiesen, da sowohl durch OLAP- als auch durch Data-Mining-Werkzeuge Anfragen der Benutzer typischerweise in eine Sequenz von Anfragen an das DWDBS übersetzt werden. Durch den modularen Aufbau der Architektur werden Änderungen, die notwendig werden, wenn existierende Werkzeuge in die vorgestellte Architektur integriert werden sollen, auf wenige Komponenten des Gesamtsystems beschränkt. Die integrierte Architektur stellt somit eine wichtige Grundlage für eine umfassende und integrierte Analyse der Daten in einem Data Warehouse dar.

Der letzte Teil der Arbeit hatte die konkrete Ausgestaltung des DSS-Optimierers als der zentralen, neuen Komponente der integrierten Architektur zum Gegenstand. Das Ziel war es zu zeigen, welche Möglichkeiten zu einer anwendungsunabhängigen Optimierung von Anfragesequenzen bestehen. Hierzu wurden zunächst sämtliche relevanten Optimierungsansätze, die im weiteren Data-Warehouse-Umfeld existieren, untersucht. Einige dieser Ansätze konnten als nicht für eine Umsetzung im Rahmen eines DSS-Optimierers geeignet identifiziert werden. Wichtige Argumente waren in diesem Zusammenhang unter anderem das auf Grund der spezifischen Eigenschaften von Anfragesequenzen zu geringe Optimierungspotenzial, die Tatsache, dass dem DSS-Optimierer einige wichtige Informationen nicht als Entscheidungsgrundlage zur Verfügung stehen sowie potenzielle Konflikte mit Optimierungsentscheidungen, die im Rahmen der Administration eines Data Warehouse an anderen Stellen getroffen werden. Dagegen konnte gezeigt werden, dass die Intra-Anfragesequenz-Parallelität, die Bereitstellung von Statistiken, die Restrukturierung von Anfragesequenzen sowie die Erstellung von Einzelanfragen geeignete Strategien für einen DSS-Optimierer darstellen. Dieses Ergebnis konnte sowohl durch die qualitative Bewertung der verschiedenen Strategien wie auch durch umfangreiche Experimente erhärtet werden. Bereits beim isolierten Einsatz der Optimierungsstrategien konnte die Laufzeit der in den Experimenten betrachteten Anfragesequenzen teilweise um eine Größenordnung reduziert werden.

Insbesondere durch die Analyse der Strategievariante, bei der Einzelanfragen erstellt werden, konnte darüber hinaus das Hauptergebnis der Arbeit bezüglich

geeigneter logischer Schematypen weiter untermauert werden. In Kapitel 4 zeichnete sich das Star-Schema unter anderem dadurch aus, dass bei typischen Anfragen auf dem Data Warehouse eine höhere Performanz zu erwarten ist, als dies bei den meisten anderen Schematypen der Fall ist. Dies kann mit der vergleichsweise geringen Komplexität der erstellten Anfragen begründet werden. Gerade die Messungen in Kapitel 6 zu den erstellten Einzelanfragen haben gezeigt, wie wenig effizient komplexe Anfragen von aktuellen Datenbanksystemen ausgeführt werden. Die Systeme sind in der Regel nicht in der Lage, die Anfrageverarbeitung so zu strukturieren, dass Zwischenergebnisse ähnlich materialisiert werden, wie dies bei den vergleichbaren Anfragesequenzen der Fall ist. Alternative Ausführungspläne, die sich auch in den Messungen als vorteilhaft erwiesen haben, wurden von den Anfrageoptimierern offensichtlich nicht berücksichtigt oder falsch bewertet. Hier sind erhebliche Fortschritte in der Technologie zur Anfrageoptimierung für komplexe Anfragen notwendig. Allerdings sind diese kurzfristig nicht zu erwarten und werden unter Umständen nur in einzelnen Datenbanksystemen realisiert. Die Konsequenzen hieraus wurden in dieser Arbeit aufgezeigt. Einerseits kann durch die Wahl geeigneter logischer Schematypen die Komplexität der Anfragen reduziert werden. Andererseits kann eine zusätzliche Systemkomponente, der DSS-Optimierer, durch den Einsatz verschiedener Optimierungsstrategien zu diesem Ziel beitragen. So stellt diese Arbeit nicht nur eine für den integrierten Einsatz von Data Mining und OLAP geeignete Datenmodellierung und Systemarchitektur vor, sondern liefert darüber hinaus einen Beitrag zu einer möglichst effizienten Umsetzung der beschriebenen Analyseansätze.

7.2 Offene Aspekte

Untersuchungen zur Integration unterschiedlicher Technologien weisen in der Regel eine hohe Komplexität auf. So konnten in dieser Arbeit nicht alle Fragestellungen, die sich im Zusammenhang mit der integrierten Nutzung unterschiedlicher Analyseansätze auf der Basis eines Data Warehouse ergeben, bearbeitet werden. Offene Aspekte gibt es in erster Linie in Bezug auf die Breite der betrachteten Analyseansätze, den Austausch von Informationen zwischen einzelnen Analyseanwendungen sowie die Realisierung der anwendungsunabhängigen Optimierung von Anfragesequenzen.

Im Rahmen dieser Arbeit wurde ausschließlich die Integration von Data Mining und OLAP fokussiert. Darüber hinaus existieren allerdings noch andere Analyse-

ansätze, die z.B. in Statistik-Werkzeugen und Report-Werkzeugen realisiert werden. Die in dieser Arbeit vorgenommenen Integrationsbetrachtungen sollten auf dieses breitere Spektrum der Analyseansätze ausgedehnt werden.

Die in der vorliegenden Arbeit vorgestellte integrierte Architektur ermöglicht eine koordinierte Verwaltung der Metadaten. Sie unterstützt ebenfalls den Austausch von Zwischenergebnissen zwischen unterschiedlichen Analysewerkzeugen. Die Frage, welche Formate hierfür geeignet sind, wurden allerdings ausgeklammert. Hierzu existieren zwar einige Vorarbeiten, eine detaillierte Untersuchung dieses Aspekts steht allerdings noch aus.

Im Rahmen dieser Arbeit wurde ein DSS-Optimierer als eine wesentliche Komponente einer integrierten Systemarchitektur für Data-Mining- und OLAP-Anwendungen vorgestellt. Die Bedeutung dieses Optimierers wurde aufgezeigt und im Spektrum verfügbarer Optimierungsansätze wurden die geeigneten identifiziert. Bei der Analyse der Ergebnisse der vorliegenden Arbeit bezüglich der konkreten Realisierung eines DSS-Optimierers haben sich jedoch einige Fragestellungen ergeben, die einer weiteren Bearbeitung bedürfen. Diese beziehen sich einerseits auf die Frage, welche Heuristiken bzw. Kostenmodelle für die Umsetzung in einem DSS-Optimierer geeignet sind. Andererseits gibt es auch im Bereich der Anfragesequenz-Restrukturierung noch eine Reihe offener Problemstellungen. Hier ist insbesondere eine detaillierte Analyse geeigneter Restrukturierungsregeln und deren Anwendungsspektrum notwendig. In diesem Zusammenhang sind Ergebnisse zu erwarten, die weit über den Bereich eines DSS-Optimierers hinausgehen. So ist z.B. zu untersuchen, in wieweit eine Integration der Techniken zur Restrukturierung von Anfragesequenzen in das Datenbanksystem selbst möglich und sinnvoll ist. Darüber hinaus erscheint es ebenfalls als ein erfolgversprechender Ansatz, Erkenntnisse, die im Zusammenhang mit der Anfragesequenz-Restrukturierung gewonnen wurden, auf die Optimierung sehr komplexer Anfragen, den Bereich kontinuierlicher Anfragen sowie die Optimierung von Anfrageströmen zu übertragen.

8 Literaturverzeichnis

[Abe99] AberdeenGroup: Bringing Analytical Reporting to Enterprise Business Intelligence. Aberdeen Group, Boston, 1999.

[Aka74] H. Akaike: A new look at the statistical model identification. IEEE Trans. Automat. Contr., AC-19, 1974.

[AFS93] R. Agrawal, C. Faloutsos, A. Swami: Efficient Similarity Search in Sequence Databases. In: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, Chicago, October 1993.

[AGS97] R. Agrawal, A. Gupta, S. Sarawagi: Modeling Multidimensional Databases. In: Proceedings of the 13th International Conference on Data Engineering (ICDE), Birmingham, U.K., April 1997.

[AM+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, I. Verkamo: Fast Discovery of Association Rules. In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge, Menglo Park, California, 1996.

[AN00] ANGOSS: Knowledge STUDIO and Related Products: A new generation of data mining technologies from ANGOSS Software Corporation. ANGOSS White Paper, 2000.

[AS94] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules. In: Proceedings of the 20th International Conference on Very Large Databases (VLDB), Santiago, Chile, September 1994.

[AS95] R. Agrawal, R. Srikant: Mining Sequential Patterns. In: Proceedings of the 11th International Conference on Data Engineering (ICDE), Taipei, Taiwan, March 6-10, 1995.

[BA94] R. Brachman, T. Anand: The Process of Knowledge Discovery in Databases: A First Sketch. In: Proceedings of KDD-94; AAAI-94 Workshop on Knowledge Discovery in Databases, Seattle Washington, July 31 - August 1, 1994.

[Bay96] R. Bayer: The universal B-Tree for multidimensional Indexing. Technical Report TUM-I9637, Institut für Informatik, TU München, 1996.

[BCK98] L. Bass, P. Clements, R. Kazman: Software Architecture in Practice. Reading, Massachusetts u.a., Addison-Wesley, 1998.

[BCN92] C. Batini, S. Ceri, S. B. Navathe: Conceptual Database Design: An Entity-Relationship Approach. Benjamin Cummings, Redwood City, California, u.a., 1992.

[BG92] L. Becker, R. H. Güting: Rule-Based Optimization and Query Processing in an Extensible Geometric Database System. In: ACM Transactions on Database Systems (TODS), Vol 17, No. 2, 1992.

[BG01] A. Bauer, H. Günzel (Hrsg.): Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung. dpunkt-Verlag, Heidelberg, 2001.

[BHL00] A. Bauer, W. Hümmer, W. Lehner: An Alternative Relational OLAP Modeling Approach. In: Proceedings of the 2nd International Conference, Data Warehousing and Knowledge Discovery, DaWaK 2000, London, UK, September 4-6, 2000.

[BHW81] R. H. Bonczak, C. W. Holsapple, A. Whinston: Foundations of Decision Support Systems. Academic Press, 1981.

[Bou01] A. Bouillet: Realisierung und Optimierung einer Data-Mining-Anwendung. Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1805, 2001.

[BRJ99] G. Booch, J. Rumbough, I. Jacobson: The Unified Modeling Language User Guide. Addison Wesley, 1999.

[CBS99] T. Conolly, C. Begg, A. Strachan: Database Systems: A Practical Approach to Design, Implementation and Management. 2. Auflage, Addison-Wesley, Harlow u.a., 1999.

[CCS93] E. F. Codd, S. B. Codd, C. T. Salley: Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. Technical Report, E.F. Codd & Associates, 1993.

[CI98] C. Y. Chan, Y. E. Ioannidis: Bitmap Index Design and Evaluation. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, June 1-4, 1998.

[CI99] C. Y. Chan, Y. E. Ioannidis: An Efficient Bitmap Encoding Scheme for Selection Queries. In: Proceedings of the ACM SIGMOD Inter-

national Conference on Management of Data, Philadelphia, Pennsylvania, USA, June 1-3, 1999.

[CD97] S. Chaudhuri, U. Dayal: An Overview of Data Warehousing and OLAP Technology. In: SIGMOD Record, Vol. 26, No. 1, 1997.

[CN98] S. Chaudhuri, V. R. Narasayya: AutoAdmin 'What-if' Index Analysis Utility. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, June 1-4, 1998.

[CD+99] J. Clear, D. Dunn, B. Harvey, M. Heytens, P. Lohmann, A. Mehta, M. Melton, L. Rohrberg, A. Savasere, R. Wehrmeister, M. Xu: Non-Stop SQL/MX Primitives for Knowledge Discovery. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999.

[Che96] P. P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. In: ACM Transactions on Database Systems (TODS), Vol. 1, No. 1, 1976.

[Che77] P. P. Chen: The Entity-Relationship Model: A Basis for the Enterprise View of Data. In: Proceedings of IFIPS NCC 46, No. 46, 1977.

[CK88] T. L. Casavant, J. G. Kuhl: A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. In: IEEE Transactions on Software Engineering, Vol. 14, No. 2, February 1988.

[Cog99] Cognos: Cognos PowerPlay Enterprise Server. White Paper, Cognos Inc., 1999.

[Cog01] Cognos: Online Analytical Processing: The Cognos Solution. White Paper, Cognos Inc. 2001.

[CT98] L. Cabbibo, R. Torlone: A Logical Approach to Multidimensional Databases. In: Proceedings of the 6th International Conference on Extending Database Technology (EDBT), Valencia, Spain, March 23-27, 1998.

[Dat95] C. J. Date: An Introduction to Database Systems. 6th Edition (The Systems Programming Series) Addison-Wesley, Reading, Massachusetts u.a., 1995.

[Dat00] Data Mining Group: Predictive Model Markup Language (PMML) 1.1, 2000. http://www.dmg.org

[DS+98] B. Dinter, C. Sapia, G. Höfling, M. Blaschka: The OLAP Market: State of the Art and Research Issues. In: Proceedings of the First International Workshop on Data Warehousing and OLAP (DOLAP), Bethesda, Maryland, USA, November 7, 1998.

[DO85] G. B. Davis, M. H. Olsen: Management information systems : conceptual foundations, structure, and development. 2nd ed. (McGraw-Hill series in management information systems), McGraw-Hill, New York u.a., 1985.

[Eme73] J. C. Emery: An Overview of Management Information Systems. DATA BASE, Vol. 5, No.2-3-4, 1973.

[FG91] B. Finance, G. Gardarin: A Rule-Based Query Rewriter in an Extensible DBMS. In: Proceedings of the Seventh International Conference on Data Engineering (ICDE), Kobe, Japan, April 8-12, 1991.

[FGW02] U. Fayyad, G. G. Grinstein, A. Wierse (Eds.): Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann, San Francisco u.a., 2002.

[Fin82] S. J. Finkelstein: Common Subexpression Analysis in Database Applications. In: Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data, Orlando, Florida, June 2-4, 1982.

[FL98] A. A. Freitas, S. H. Livington: Mining very large databases with parallel processing. Kluwer Academic, Boston u.a., 1998.

[FPS96a] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: The KDD Process for Extracting Useful Knowledge from Volumes of Data. In: Communications of the ACM, Vol. 39, No. 11, November 1996.

[FPS96b] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: From Data Mining to Knowledge Discovery: An Overview. In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Ed.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge, Menglo Park, California, 1996.

[FS+96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Ed.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge, Menglo Park, California, 1996.

[Gar98] S. R. Gardner: Building the Data Warehouse. In: Communications of the ACM, Vol. 41, No. 9, 1998.

[GB+96] J. Gray, A. Bosworth, A. Layman, H. Pirahesh: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: Proceedings of the 12th International Conference on Data Engineering (ICDE), New Orleans, Louisiana, February 26 - March 1, 1996.

[Gel93] A. van Gelder: Multiple Join Size Estimation by Virtual Domains. In: Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems PODS), Washington, DC, May 25-28, 1993.

[GGR99] V. Ganti, J. Gehrke, R. Ramakrishnan: Mining Very Large Databases. In: IEEE Computer, Vol. 32, No. 8, 1999.

[GGS96] S. Ganguly, A. Goel, A. Silberschatz: Efficient and Acurate Cost Models for Parallel Query Optimization. In: Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Montreal, Canada, June 3-5, 1996.

[GL97] M. Gyssens, L. V. S. Lakshmanan: A Foundation for Multi-dimensional Databases. In: Proceedings of 23rd International Conference on Very Large Data Bases (VLDB), Athens, Greece, August 25-29, 1997.

[GL01] J. Goldstein, P. A. Larson: Optimizing Queries Using Materialized Views: A Practical, Scalable Solution. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, USA, May 21-24, 2001.

[GM95] A. Gupta, I. S. Mumick: Maintenance of Materialized Views: Problems, Techniques and Applications. In: Bulletin of the Technical Committee on Data Engineering, Vol. 18, No. 2, June 1995.

[GM+97] C. Glymour, D. Madigan, D. Pregibon, P. Smyth: Statistical Themes and Lessons for Data Mining. In: U. Fayyad, H. Mannila, G. Pia-

tetsky-Shapiro (Ed.): Data Mining and Knowledge Discovery, Vol. 1, No. 1, Kluwer Academic Publisher, 1997.

[GMR98a] M. Golfarelli, D. Maio, S. Rizzi: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. In: International Journal of Cooperative Information Systems, Vol. 7, No. 2&3, 1998.

[GMR98b] M. Golfarelli, D. Maio, S. Rozzi: Conceptual Design of Data Warehouses from E/R Schemes. In: Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31), Vol. 7, Kona, Hawaii, 1998.

[GP95] D. Garlan, D. E. Perry: Introduction to the Special Issue on Software Architecture. In: IEEE Transactions on Software Engineering (TSE), Vol. 21, No. 4, April 1995.

[GR98] M. Golfarelli, S. Rizzi: Methodological Framework for Data Warehouse Design. In: Proceedings of the ACM First International Workshop on Data Warehousing and OLAP, Bethesda, Maryland, USA, November 7, 1998.

[GR99] M. Golfarelli, S. Rizzi: Designing the Data Warehouse: Key Steps and Crucial Issues. In: Journal of Computer Science and Information Management, Vol. 2, No. 1, 1999.

[Gut84] A. Guttman: R-Trees: A Dynamic Index Structure for Spatial Searching. In: SIGMOD 94: Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984.

[Han97] J. Han: OLAP Mining: An Integration of OLAP with Data Mining. In: Proceedings of the 7th IFIP 2.6 Working Conference on Database Semantics (DS-7), Leysin, Switzerland, 1997.

[Han98] J. Han: Towards On-Line Analytical Mining in Large Databases. In: SIGMOD Record, Vol. 27, No. 1, 1998.

[HF+89] L. Haas, J. C. Freytag, G. Lohman, H. Pirahesh: Extensible Query Processing in Starburst. In: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989.

[HK87] R. Hull, R. King: Semantic Database Modeling: Survey, Applications, and Research Issues. In: ACM Computing Surveys, Vol. 19, No. 3, 1987.

[HK01] J. Han, M. Kamber: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco u.a., 2001.

[HLV00] B. Hüsemann, J. Lechtenbörger, G. Vossen: Conceptual Data Warehouse Design. In: Proceedings of the Second International Workshop on Design and Management of Data Warehouses, DMDW 2000, Stockholm, Sweden, June 5-6, 2000.

[HR99] T. Härder, E. Rahm: Datenbanksysteme: Konzepte und Techniken der Implementierung. Springer Verlag, Berlin u.a., 1999.

[HRU96] V. Harinarayan, A. Rajaraman, J. D. Ullman: Implementing Data Cubes Efficiently. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996.

[HS93] W. Hong, M. Stonebraker: Optimization of Parallel Query Execution Plans in XPRS. In: Distributed and Parallel Databases, Vol. 1, No. 1, 1993.

[HS94] M. Holsheimer, A. Siebes: Data Mining: The Search for Knowledge in Databasees. Centrum voor Wiskunde en Informatica, Computer Science, Department of Algorithics and Architecture, Amsterdam, The Netherlands, Report CS-R9406, 1994.

[Hyp01a] Hyperion: The Role of the OLAP Server in a Data Warehouse Solution. White Paper, Hyperion Inc., 2001.

[Hyp01b] Hyperion: An Overview of Hyperion's Data Warehousing Methodology. White Paper, Hyperion Inc., 2001.

[Ibm99] IBM Corp.: Using the Intelligent Miner for Data, Version 6 Release 1. IBM Corp., 1999.

[Ibm01a] IBM Corp.: IBM Intelligent Miner Scoring: Administration and Programming for DB2, Version 7.1, 2001.

[Ibm01b] IBM Corp.: IBM DB2 Universal Database: Command Reference, Version 7, 2001.

[Ibm01c] IBM Corp.: IBM DB2 Universal Database: SQL Reference, Version 7, 2001.

[Inf97] Informix: Entwurf eines Data Warehouse auf relationalen Datenbanken. White Paper, Informix Software GmbH, 1997.

[Inm92] W. H. Inmon: Building the Data Warehouse. Wiley, New York u.a., 1992.

[Iso92] International Organization for Standardization (ISO): Information Technology - Database Language SQL. Standard No. ISO/IEC 9075:1992.

[Iso99] International Organization for Standardization (ISO): Information Technology - Database Languages - SQL - Part 2: Foundation (SQL/ Foundation). Standard No. ISO/IEC 9075-2:1999.

[JLN00] T. Johnson, L. V. S. Lakshmanan, R.T Ng.: The 3W Model and Algebra for Unified Data Mining. In: Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt, September 10-14, 2000.

[JMF99] A. K. Jain, M. N. Murty, P. J. Flynn: Data Clustering: A Review. ACM Computing Surveys, Vol. 31 , No. 3, 1999.

[Kim96] R. Kimball: The Data Warehouse Toolkit. Wiley, New York u.a., 1996.

[Kim00] J. Kimmerle: Data Mining im Pharma-Großhandel. Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1821, 2000.

[KR+98] R. Kimball, L. Reeves, M. Ross, W. Thornthwaite: The Data Warehouse Lifecycle Toolkit. Wiley, New York u.a., 1998.

[Kra01] T. Kraft: Datenmodellierung für Data Mining und OLAP. Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1806, 2001.

[Kra02] T. Kraft: Rewrite-Strategien für generierte Anfragesequenzen im Online Analytical Processing. Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1967, 2002.

[LAW98] W. Lehner, J. Albrecht, H. Wedekind: Normal Forms for Multidimensional Databases. In: Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM), Capri, Italy, July 1-3, 1998.

[LC+01] W. Lehner, R. Cochrane, H. Pirahesh, M. Zarioudakis: fAST Refresh using Mass Query Optimization. In: Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, April 2-6, 2001.

[LF93] Laender, A. H. F., Flynn, D. J.: A Semantic Comparison of the Modelling Capabilities of the ER and NIAM Models. In: Proceedings of the 12th International Conference on the Entity-Relationship Approach, Arlington, Texas, USA, December 15-17, 1993.

[Lec01] J. Lechtenbörger: Data Warehouse Schema Design. Dissertationen zu Datenbanken und Informationssystemen, Band 79, Berlin, Akademische Verlagsgesellschaft, 2001.

[Len98] W. Lehner: Modeling Large Scale OLAP Scenarios. In: Proceedings of the 6th International Conference on Extending Database Technology (EDBT), Valencia, Spain, March 23-27, 1998.

[LR+00] W. Lehner, R. Sidle, H. Pirahesh, R Cochrane: Maintenance of Cube Automatic Summary Tables. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 16-18, 2000.

[LS97] H.-J. Lenz, A. Soshani: Summarizability in OLAP and Statistical Data Bases. In: Proceedings of the 9th International Conference on Scientific and Statistical Database Management, August 11-13, Olympia, Washington, USA, 1997

[LST91] H. Lu, M. C. Shan, K. L. Tan: Optimization of Multi-Way Join Queries for Parallel Execution. In: Proceedings of the 17th International Conference on Very Large Data Bases (VLDB), Barcelona, Catalonia, Spain, September 3-6, 1991.

[LVZ93] R. S. G. Lanzelotte, P. Valduriez, M. Zaid: On the Effectiveness of Optimization Search Strategies for Parallel Execution Spaces. In: Proceedings of the19th International Conference on Very Large Data Bases (VLDB), Dublin, Ireland, August 24-27, 1993.

[LW96] C. Li, X. S. Wang: A Data Model for Supporting On-Line Analytical Processing. In: Proceedings of the 5th International Conference on Information and Knowledge Management, Rockville, Maryland, USA, November 12-16, 1996.

[Lyn88] C. A. Lynch: Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distribution of Column Values. In: Proceedings of the 14th International Conference on Very Large Data Bases (VLDB), Los Angeles, California, USA, August 29 - September 1, 1988.

[Mic95] MicroStrategy: The Case for Relational OLAP. White Paper, MicroStrategy, 1995.

[Mic99] MicroStrategy: MicroStrategy Architect Version 6.0: User Guide. MicroStrategy, 1999.

[Mit95] B. Mitschang: Anfrageverarbeitung in Datenbanksystemen: Entwurfs- und Implementierungskonzepte. Vieweg, Braunschweig, Wiesbaden, 1995.

[MK+99] N. M. Mattos, J. Kleewein, M. T. Roth, K. Zeidenstein: From Object-Relational to Federated Databases. In: A. P. Buchmann (Hrsg.): Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Freiburg, 1.-3. März, 1999, Informatik Aktuell, Springer, Berlin u.a., 1999.

[MK00] D. L. Moody, M. A. R. Kortink: From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. In: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses, DMDW 2000, Stockholm, Sweden, June 5-6, 2000.

[Mor71] M. S. Morton: Management decision systems: computer-based support for decision making. Graduate School of Business Administration, Division of Research, Harvard University, 1971.

[Mor72] M. S. Morton: Management-Entscheidungen im Bildschirmdialog. Girardet, Essen, 1972.

[MR+01] H. Mistry, P. Roy, S. Sudarshan, K. Ramamritham: Materialized View Selection and Maintenance Using Multi-Query Optimization. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Santa Barbara, CA, USA, May 21-24, 2001.

[MS83] R. S. Michalski, R. E. Stepp: Learning from observation: Conceptual clustering. In: R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Ed.): Machine Learning: An Artificial Intelligence Approach, Vol. 1, San Mateo, CA, Morgan Kaufmann, 1983.

[MS89] V. M. Markowitz, A. Soshani: On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989.

[MS02] J. Melton, A. R. Simon: SQL:1999: Understanding Relational Language Components. Morgan Kaufmann, San Francisco u.a., 2002.

[NC+84] S. B. Navathe, S. Ceri, G. Wiederhold, J. Dou: Vertical Partitioning Algorithms for Database Design. In: ACM Transactions on Database Systems (TODS) , Vol. 9, No. 4, 1984.

[NM98] C. Nippl, B. Mitschang: TOPAZ: A Cost-Based, Rule-Driven, Multi-Phase Parallelizer. In: Proceedings of 24th International Conference on Very Large Data Bases (VLDB), New York City, New York, USA, August 24-27, 1998.

[Obj01a] Object Management Group: Common Warehouse Metamodel (CWM) Specification. Version 1.0, Object Management Group, 2 February 2001.

[Obj01b] Object Management Group: Common Warehouse Metamodel (CWM) Specification, Volume 2 Extensions. Version 1.0, Object Management Group, 2 February 2001.

[OG95] P. E. O'Neil, G. Graefe: Multi-Table Joins Through Bitmapped Join Indices. In: SIGMOD Record, Vol. 24, No. 3, September 1995.

[One87] P. E. O'Neil: Model 204: Architecture and Performance. In: 2nd International Workshop on High Performance Transaction Systems (Lecture Notes in Computer Science No. 359), Springer, September 1987.

[Ora96] Oracle Corp.: Oracle Express Server Database Administration Guide. Release 6.0, Part No. A47232-1, 1996.

[Ora00a] Oracle Corp.: Oracle Darwin New Features, Release 3.6. Oracle Corp., Part No. A83710-01, February 2000.

[Ora00b] Oracle Corp.: Oracle8i SQL Reference, Release 3 (8.1.7), Oracle, September 2000.

[PC00] N. Pendse, R. Creeth: The OLAP Report. www.olapreport.com, 2000.

[PHH92] H. Pirahesh, J. M. Hellerstein, H. Waqar: Extensible/Rule Based Query Rewrite Optimization in Starburst. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5, 1992.

[PJ99] T. B. Pedersen, C. S. Jensen: Multidimensional Data Modeling for Complex Data. In: Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Austrialia, March 23-26, 1999.

[PKB98] Vidette Poe, Patricia Klauer, Stephen Brobst: Building a data warehouse for decision support. Prentice Hall, Upper Saddle River, 1998.

[PLH97] H. Pirahesh, T. Y. Leung, H. Waqar: A Rule Engine for Query Transformation in Starburst and IBM DB2 C/S DBMS. In: Proceedings of the 13th International Conference on Data Engineering (ICDE), Birmingham, U.K., April 7-11, 1997.

[PR00] E. Pourabba, M. Rafanelli: Hierarchies and Relative Operators in the OLAP Environment. In: SIGMOD Record, Vol. 29, No. 1, 2000.

[Raf95] A. E. Raftery: Bayesian model selection in social research (with discussion). In: Sociological Methodology, P. V. Marsden (Ed.). Blackwells, Oxford, U.K., 1995.

[Rah94] E. Rahm: Mehrrechner Datenbanksysteme - Grundlagen der parallelen und verteilten Datenbankverarbeitung. Addison-Wesley, Bonn u.a., 1994.

[Ris78] J. Rissanen: Modeling by shortest data description. Automatica, 14, 1978.

[Roc79] J. F. Rockart: Chief Executives Define Their Own Data Needs. Harvard Business Review, Vol. 67, No. 2, 1979.

[RS99] R. Rantzau, H. Schwarz: A Multi-Tier Architecture for High-Performance Data Mining. In: A. P. Buchmann (Hrsg.): Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft. Freiburg, 1.-3. März 1999, Informatik Aktuell, Springer, Berlin u.a., 1999.

[Sas99] SAS Institute: SAS Enterprise Miner 3.0 Manual. SAS Institute Inc., 1999.

[Sas96] SAS Institute: OLAP Tools and Techniques within the SAS System. SAS Institute Inc., 1996.

[SA95] R. Srikant, R. Agrawal: Mining Generalized Association Rules. IBM Research Report RJ9963, June 1995.

[SA96] R. Srikant, R. Agrawal: Mining Quantitative Association Rules in Large Relational Tables. In: Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.

[SAM96] J. Shafer, R. Agrawal, M. Mehta: SPRINT: A Scalable Parallel Classifier for Data Mining. In: Proceedings of the 22nd International Conference on Very Large Databases (VLDB), Mumbai (Bombay), India, September 3-6, 1996.

[SAM98a] S. Sarawagi, R. Agrawal, N, Megiddo: Discovery-driven Exploration of OLAP Data Cubes. Research Report RJ 10102 (91918), IBM Almaden Research Center, San Jose, CA 95120, January 1998.

[SAM98b] S. Sarawagi, R. Agrawal, N, Megiddo: Discovery-driven Exploration of OLAP Data Cubes. In: Proceedings of the 6th International Conference on Extending Database Technology (EDBT), Valencia, Spain, March 23-27, 1998.

[SD90] D. A. Schneider, D. J. DeWitt: Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines. In: Proceedings of the 16th International Conference on Very Large Data Bases (VLDB), Brisbane, Queensland, Australia, August 13-16, 1990.

[Sho97] A. Shoshani: OLAP and Statistical Databases: Similarities and Differences. In: Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Tucson, Arizona May 12-14, 1997.

[SL+01] M. Stillger, G. Lohman, V. Markl, M. Kandil: LEO - DB2's LEarning Optimizer. In: Proceedings of the 27th International Conference on Very Large Databases (VLDB), Roma, Italy, September 11-14, 2001.

[SMR00] T. Stöhr, H. Märtens, E. Rahm: Multi-Dimensional Database Allocation for Parallel Data Warehouse. In: Proceedings of the 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, September 10-14, 2000.

[SpCa82] R. H. Sprague, E. D. Carlson: Building Effective Decision Support Systems. Prentice-Hall, Englewood Cliffs, 1982.

[SP99a] SPSS Inc.: SPSS 10.0 and the Distributed Analysis Architecture. SPSS White Paper, 1999.

[SP99b] SPSS Inc.: An introduction to Clementine Server Distributed Architecture. SPSS White Paper, 1999.

[SR01] T. Stöhr, E. Rahm: WARLOCK: A Data Allocation Tool for Parallel Warehouses. In: Proceedings of 27th International Conference on Very Large Data Bases (VLDB), Roma, Italy, September 11-14, 2001.

[SS94] A. N. Swami, K. B. Schiefer: On the Estimation of Join Result Sizes. In: Proceedings of the 4th International Conference on Extending Database Technology (EDBT), Cambridge, United Kingdom, March 28-31, 1994.

[TBC99] N. Tryfona, F. Busborg, J. G. Borch Christiansen: starER: A Conceptual Model for Data Warehouse Design. In: Proceedings of the ACM Second International Workshop on Data Warehousing and OLAP, Kansas City, Missouri, USA, November 6, 1999.

[TBD97] P. Tamayo, J. Berlin, N. Dayanand, G. Drescher, D. R. Mani, C. Wang: Darwin: A Scalable Integrated System for Data Mining. Thinking Machines Inc., White Paper, 1997.

[Thi98] Thinking Machines: Using Darwin, Release 3.01. Thinking Machine Corp., Massachusetts, May 1998.

[Tra99] Transaction Processing Performance Council: TPC Benchmark H. Technical Report Standard Specification Revision 1.2.1, Transaction Processing Performance Council, 1999.

[TYF86] T. J. Teorey, D. Yang, J. P. Fry: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. In: ACM Computing Surveys, Vol. 18, No. 2, 1986.

[Val87] P. Valduriez: Join Indeces. In: ACM Transactions on Database Systems (TODS) , Vol. 12, No. 2, 1987.

[Wag00] R. Wagner: Realisierung und Optimierung einer OLAP-Anwendung im Handelsbereich. Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1770, 2000.

[WB97] M. Wu, A. P. Buchmann: Research Issues in Data Warehousing. In: K. R. Dittrich, A. Geppert (Hrsg.): Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Ulm, 5.-7. März 1997, Informatik Aktuell, Springer, Berlin u.a., 1997.

[Wes01] Paul Westerman: Data Warehousing: Using the Wal-Mart Model. Morgan Kaufmann, San Francisco u.a., 2001.

[WFA95] A. N. Wilschut, J. Floistra, P. M. G. Apers: Parallel Evaluation of Multi-Join Queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995.

[Wid95] J. Widom: Research Problems in Data Warehousing. In: Proceedings of 4th International Conference on Information and Knowledge Management, Baltimore, Maryland, USA, November 28 - December 2, 1995.

[Wie92] G. Wiederhold: Mediators in the architecture of future information systems. In: IEEE Computer, Vol. 25, No. 3, March 1992.

[ZC+00] M. Zaharioudakis, R. Cochrane, G. Lapis, H. Pirahesh, M. Urata: Answering Complex SQL Queries Using Automatic Summary Tables. In: SIGMOD Record, Vol. 29, No. 2, June 2000.

[ZD+98] Y. Zhao, P. M. Deshpande, J. F. Naughton, A. Shukla: Simultanous Optimization and Evaluation of Multiple Dimensional Queries. In: SIGMOD Record, Vol. 27, No. 2, June 1998.

Anhang A

Das folgende Syntaxdiagramm gibt die im Rahmen dieser Arbeit berücksichtigte Syntax für SQL-Anweisungen wieder. Diese stellt eine Untermenge von SQL-92 dar. Im Syntaxdiagramm sind die Pfeile zur ORDER-BY-Klausel grau dargestllt. Hiermit soll angedeutet werden, dass diese Klausel nicht für Subanfragen genutzt werden kann.