Qualitätsgetriebene Datenproduktionssteuerung in Echtzeit-Data-Warehouse-Systemen

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inf. Maik Thiele
geboren am 24. November 1979 in Freiberg

Gutachter:

Prof. Dr.-Ing. Wolfgang Lehner
Technische Universität Dresden
Fakultät Informatik, Institut für Systemarchitektur
Lehrstuhl für Datenbanken
01062 Dresden

Prof. Dr.-Ing. habil. Thomas Ruf
Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät
Lehrstuhl für Informatik 6 (Datenmanagement)
91054 Erlangen

Tag der Verteidigung: 31.05.2010

Dresden im Juli 2010

Zusammenfassung

Wurden früher Data-Warehouse-Systeme meist nur zur Datenanalyse für die Entscheidungsunterstützung des Managements eingesetzt, haben sie sich nunmehr zur zentralen Plattform für die integrierte Informationsversorgung eines Unternehmens entwickelt. Dies schließt vor allem auch die Einbindung des Data-Warehouses in operative Prozesse mit ein, für die zum einen sehr aktuelle Daten benötigt werden und zum anderen eine schnelle Anfrageverarbeitung gefordert wird. Daneben existieren jedoch weiterhin klassische Data-Warehouse-Anwendungen, welche hochqualitative und verfeinerte Daten benötigen. Die Anwender eines Data-Warehouse-Systems haben somit verschiedene und zum Teil konfligierende Anforderungen bezüglich der Datenaktualität, der Anfragelatenz und der Datenstabilität. In der vorliegenden Dissertation wurden Methoden und Techniken entwickelt, die diesen Konflikt adressieren und lösen. Die umfassende Zielstellung bestand darin, eine Echtzeit-Data-Warehouse-Architektur zu entwickeln, welche die Informationsversorgung in seiner ganzen Breite – von historischen bis hin zu aktuellen Daten – abdecken kann.

Zunächst wurde ein Verfahren zur Ablaufplanung kontinuierlicher Aktualisierungsströme erarbeitet. Dieses berücksichtigt die widerstreitenden Anforderungen der Nutzer des Data-Warehouse-Systems und erzeugt bewiesenermaßen optimale Ablaufpläne. Im nächsten Schritt wurde die Ablaufplanung im Kontext mehrstufiger Datenproduktionsprozesse untersucht. Gegenstand der Analyse war insbesondere, unter welchen Bedingungen eine Ablaufplanung in Datenproduktionsprozessen gewinnbringend anwendbar ist.

Zur Unterstützung der Analyse komplexer Data-Warehouse-Prozesse wurde eine Visualisierung der Entwicklung der Datenzustände, über die Produktionsprozesse hinweg, vorgeschlagen. Mit dieser steht ein Werkzeug zur Verfügung, mit dem explorativ Datenproduktionsprozesse auf ihr Optimierungspotenzial hin untersucht werden können.

Das den operativen Datenänderungen unterworfene Echtzeit-Data-Warehouse-System führt in der Berichtsproduktion zu Inkonsistenzen. Daher wurde eine entkoppelte und für die Anwendung der Berichtsproduktion optimierte Datenschicht erarbeitet. Es wurde weiterhin ein Aggregationskonzept zur Beschleunigung der Anfrageverarbeitung entwickelt. Die Vollständigkeit der Berichtsanfragen wird durch spezielle Anfragetechniken garantiert.

Es wurden zwei Data-Warehouse-Fallstudien großer Unternehmen vorgestellt sowie deren spezifische Herausforderungen analysiert. Die in dieser Dissertation entwickelten Konzepte wurden auf ihren Nutzen und ihre Anwendbarkeit in den Praxisszenarien hin überprüft.

Danksagung

Zum erfolgreichen Abschluss dieser Arbeit haben eine Reihe von Personen beigetragen. Denen, die mich in den letzten fünf Jahren, auf unterschiedliche Weise unterstützt haben, möchte ich hiermit danken.
Der größte Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. Wolfgang Lehner für seine Unterstützung, sein Vertrauen und für seine zahlreichen Ideen und Anmerkungen die wesentlich zum Erfolg dieser Arbeit beigetragen haben. Er hat durch seinen Enthusiasmus meine Begeisterung für dieses Fach geweckt und diese durch Weitergabe seines reichen Wissens ständig neu entfacht. Ich danke ihm auch für die vielen Kontakte zu Kollegen aus Forschung und Industrie durch die ich zahlreiche Anstöße gewinnen konnte, die zum Gelingen dieser Arbeit beigetragen haben. Darüber hinaus bedanke ich mich bei ihm für die Möglichkeit in verschiedenen Industrieprojekten andere Sichtweisen kennenzulernen und neue Erfahrungen sammeln zu können. Er hat die Rolle des Mentors in all seine Facetten ausgefüllt – Danke.
Herrn Prof. Dr.-Ing. Thomas Ruf danke ich für die Bereitschaft zur Begutachtung dieser Arbeit, die Aufopferung seiner wertvollen Zeit und seine zahlreichen Fragen und Hinweise.
Weiterhin danke ich Dr.-Ing. Jens Albrecht, der mir bei meinen Aufenthalten in der GfK in Nürnberg viel über Data Warehousing beibegebracht und maßgeblich am Anstehen meiner ersten Veröffentlichungen beigetragen hat.
Für ihre Hilfe bei meinen Fallstudienrecherchen danke ich Volker Schlemmer und Michael Martin von der GfK Marketing Services in Nürnberg sowie Bärbel Bohr, Bernhard Eisenbarth, Matthias Seitz und Jürgen Peick von der UBS in Zürich.
Darüber hinaus danke ich meinen ehemaligen Studenten Andreas Bader, Ulrike Fischer und Steffen Schmidt für die Unterstützung bei den prototypischen Umsetzungen der in dieser Arbeit entwickelten Konzepte.
Nicht vergessen möchte ich meine ehemaligen und gegenwärtigen Kollegen, die durch ihre hilfsbereite und kameradschaftliche Art zu einem angenehmen Arbeitsklima beigetragen haben. Besonderer Dank gilt Dr.-Ing. Dirk Habich, Dr. rer. nat. Rainer Gemulla, Steffen Preissler und Hannes Voigt die in zahlreichen Gesprächsrunden ihr Zeit geopfert und wertvolle Anregungen und Hilfestellungen gegeben haben.
Großer Dank gebührt auch Simone Linke die viel Zeit und Geduld für die Durchsicht und Korrektur dieser Arbeit aufgewendet hat.
Meinen Freunden und meiner Familie danke ich für ihre Unterstützung, die aufmunternden Worte und die gemeinsamen Aktivitäten, durch die ich stets neue Kraft und Motivation sammeln konnte. Ganz speziellen Dank bedarf es an meine Verlobte Judith, die zur gleichen Zeit ihre eigene Dissertation anfertigte. Mit einer bewundernswerten Selbstverständlichkeit hat sie Freud und Leid mit mir geteilt.

Danksagung

Besonderer Dank gilt meine Eltern Andrea und Lothar Thiele, die mich ausnahmslos unterstützt und immerzu mit mir gefiebert haben. Ihnen widme ich die Arbeit von ganzem Herzen.

Maik Thiele
10. April, 2010.

Inhaltsverzeichnis

1 Einleitung

„The only reason for time is so that everything doesn't happen at once.“
Albert Einstein (Deutscher Physiker, 1879-1955)

Die Begriffe Echtzeit und Data-Warehouse im Kontext der Datenanalyse erscheinen zunächst widersprüchlich. Ersteres beschreibt den Bedarf nach aktuellen und Letzteres die Notwendigkeit für historische Daten. Jedoch existieren zahlreiche Anwendungen, in denen eben diese Verbindung aktueller und historischer Daten einen entscheidenden Mehrwert generieren kann. Ein Anwendungsfall aus dem Handel ist etwa die Vermeidung sogenannter Regallücken (engl. *Out-of-Stock*), d.h. Situationen, in denen Waren nachgefragt werden aber nicht mehr verfügbar sind. Um diese Fälle zu minimieren, sind zum einen historische Verkaufstransaktionen zur Bildung von Vorhersagemodellen erforderlich, zum anderen aber auch aktuelle Lagerbestände und Transaktionsdaten zum Abgleich mit bzw. zur Verfeinerung der Vorhersagen.
In der Autoversicherungsbranche existieren Feldstudien [97], die Beiträge bzw. die Tarif- und Risikoberechnung nutzungsabhängig statt, wie sonst üblich, pauschal erheben. Dafür sind zum einen Unfall- und Schadensstatistiken aufgeschlüsselt nach Tageszeit, Jahreszeit und Geographie erforderlich, um die Risiken bzw. die daraus abgeleiteten Tarife zu errechnen. Zum anderen bedarf es der aktuellen Fahrzeugkoordinaten sowie der Information, ob das Auto bewegt wird oder steht. Durch die Kombination beider Informationen kann der Versicherungsbeitrag nutzungsabhängig erhoben werden.
Flughafenbetreiber, wie zum Beispiel die Fraport AG, deren Geschäftsmodell auf der optimalen Abwicklung des Flug- und Terminalbetriebs (engl. *aviation management*) sowie der Bodendienste (engl. *ground handling*) basiert, sind ebenfalls in einem hohen Maß auf operative Informationen angewiesen. Einem zu startenden oder zu landenden Flugzeug können bis zu 1.000 Attribute zugewiesen sein [Fraport], wie zum Beispiel das Wetter am Flugstandort, die Auslastung oder die Nationalität der Passagiere. Diese Daten werden mit Vergangenheitsinformationen, Statistiken und Prognosen abgeglichen und somit die genannten Prozesse gesteuert. Ein Beispiel ist die Einplanung des Bodenpersonals zur Visaabfertigung in Abhängigkeit der Nationalitäten der Flugzeugpassagiere.
Hatten in der Vergangenheit Data-Warehouse-Systeme den Auftrag der Informationsversorgung von Management und Wissensarbeitern (engl. *knowledge worker*), haben sie sich inzwischen zur zentralen Plattform für die integrierte Informationsversorgung eines Unternehmens entwickelt. Dies schließt neben strategischen Analysen vor allem auch die Unterstützung operativer Prozesse mit ein. Eine klare Trennung

zwischen operativen und analytischen Systemen, wie sie bisher propagiert wurde, wird es daher in Zukunft nicht mehr geben.
Zur Verwirklichung dieses Ziels sind jedoch noch eine Vielzahl offener Probleme zu lösen. Bislang wurden die Daten eines Data-Warehouses in batchbasierten Prozessen eingepflegt, unter Verwendung periodischer Snapshots der operativen Quellen. Zur Unterstützung der Datenintegration in Echtzeit ist ein Paradigmenwechsel weg von den klassischen, pull-basierten Batch-Prozessen hin zu kontinuierlichen, push-basierten Ladevorgängen notwendig. Mit der Aktualisierungsperiodizität müssen sich auch die auf die Verarbeitung von Massendaten spezialisierten ETL-Prozesse (Extraktion, Transformation und Laden) in Richtung einer strombasierten Verarbeitung (engl. *streaming ELT*) ändern. Ralph Kimball hat diese Entwicklung umschrieben mit dem Satz: „Real Time is anything that is too fast for your current ETL“. Die Optimierung von ETL-Prozessen hinsichtlich einer tupelbasierten Verarbeitung wurde bereits in mehreren Papieren [59, 83, 42, 80] untersucht und bleibt aktuelles Forschungsthema.
Der Gegenstand dieser Dissertation ist jedoch ein Anderer: Ein Data-Warehouse als zentrale Plattform der Informationsversorgung eines Unternehmens muss auch in Zukunft neben hochaktuellen Informationen weiterhin qualitative, geprüfte und über viele Verarbeitungsschritte verfeinerte Daten zur Verfügung stellen. Data-Warehouse-Systeme sehen sich somit einer heterogenen Anwendergruppe gegenüber die verschiedene Anforderungen an die Aktualität, die Anfragelatenz, die Konsistenz und die Stabilität der Daten stellt. Am Beispiel der oben dargestellten Anwendungsszenarien, sind in einem Data-Warehouse neben den Echtzeitdaten zur Erkennung von Regallücken auch klassische Anwendungen wie das Controlling oder Marketing mit Daten zu versorgen. Für diese gelten offensichtlich andere Anforderungen als für Ad-hoc-Datenanalysen, deren Ergebnisse mitunter schnell wieder verworfen werden. In der vorliegenden Dissertation werden Methoden und Techniken entwickelt die diesen Konflikt adressieren. Die globale Zielstellung war dabei, eine Echtzeit-Data-Warehouse-Architektur zu realisieren welches die Informationsversorgung in seiner ganzen Breite, von historischen bis hin zu aktuellen Daten, abdecken kann.

Beiträge der Arbeit

Die Architektur eines Echtzeit-Data-Warehouse-Systems aus Sicht der Datenproduktion ist in Abbildung 1.1 skizziert. Zunächst werden die Daten aus den operativen Systemen übernommen und in das Data-Warehouse übertragen. Dort durchlaufen sie einen mehrstufigen Datenverfeinerungsprozess und stehen parallel – in den verschiedenen Verfeinerungszuständen – der Anfrageverarbeitung zur Verfügung. Um der Forderung nach Stabilität, wie sie für einige Anwendungen notwendig ist, nachzukommen, ist eine zusätzliche Datenschicht notwendig. Diese ist von der Echtzeitproduktion der Daten entkoppelt, wodurch unkontrollierte Datenänderungen vermieden werden können. Die Beiträge dieser Dissertation sind horizontal, entlang des dargestellten Datenproduktionsprozesses, wiederzufinden:

- Es wird ein Verfahren zur Datenproduktionssteuerung bzw. zur Ablaufplanung kontinuierlicher Aktualisierungsströme erarbeitet. Dies ist zunächst einstufig, d.h. es wird nur die Datenübernahme von den operativen Systemen in das Data-Warehouse betrachet. Das Verfahren berücksichtigt die widerstreitenden Anforderungen der Nutzer des Data-Warehouse-Systems und erzeugt bewiesenermaßen optimale Ablaufpläne.

- Die entwickelte Ablaufplanung wird im Kontext mehrstufiger Datenproduktionsprozesse untersucht. Insbesondere wird analysiert, unter welchen Bedingungen eine Ablaufplanung in Datenproduktionsprozessen gewinnbringend anwendbar ist.

- Die Ablaufplanung macht die Bestimmung der Abhängigkeiten zwischen Anfragen und Aktualisierungen notwendig. Dazu wird ein partitionsbasiertes Verfahren entwickelt, mit dem die entsprechenden Korrelationen effizient bestimmt werden können. Die Genauigkeit der Abhängigkeitsbestimmung wird durch eine Reihe statistischer Methoden ergänzt und verbessert.

- Zur Unterstützung der Analyse komplexer Data-Warehouse-Prozesse wird eine Visualisierung der Entwicklung der Datenzustände, über die Produktionsprozesse hinweg, vorgeschlagen. Mit dieser steht ein Werkzeug zur Verfügung, mit dem explorativ Datenproduktionsprozesse auf ihr Optimierungspotenzial hin untersucht werden können.

- Das den operativen Datenänderungen unterworfene Echtzeit-Data-Warehouse-System führt in der Berichtsproduktion zu Inkonsistenzen. Daher wird eine entkoppelte und für die Anwendung der Berichtsproduktion optimierte Datenschicht erarbeitet. Es wird weiterhin ein neuartiges Aggregationskonzept zur Beschleunigung der Anfrageverarbeitung entwickelt. Die Vollständigkeit der Berichtsanfragen wird durch spezielle Anfragetechniken garantiert.

- Die Anfrage erfordert die explizite Speicherung von Nullwerten. Zur effizienten Speicherung dieser Werte werden Komprimierungsverfahren erarbeitet. Zur Rekonstruktion des ursprünglichen Datenraums werden partielle Dekomprimierungstechniken entwickelt und in die Anfrageverarbeitung integriert.

- Zur Bewertung der entwickelten Verfahren werden zwei komplexe Data-Warehouse-Szenarien aus der Praxis in Form von Fallstudien vorgestellt.

- Die Anforderungen der situativen Datenanalyse überschneiden sich stark mit denen der Echtzeit-Data-Warehouse-Systeme. Dieses Forschungsgebiet wird daher umfangreich untersucht. In dieser Arbeit wird ein neuer Ansatz zur Unterstützung der situativen Datenanalyse vorgestellt und mit bereits bekannten Techniken verglichen.

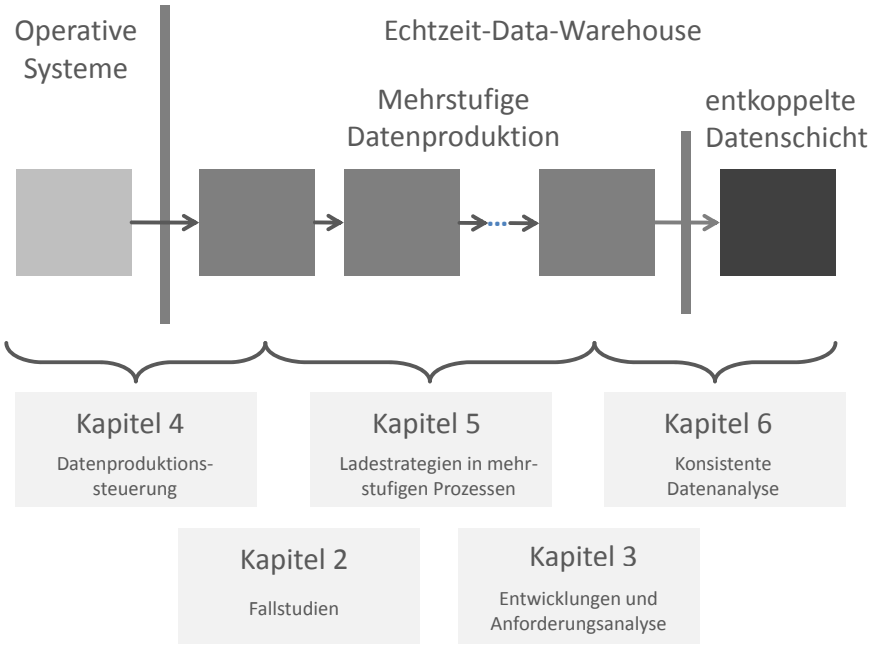


Abbildung 1.1: Gliederung der Arbeit am Aufbau eines Echtzeit-Data-Warehouse-Systems

Aufbau der Arbeit

Kapitel 2 stellt zwei Data-Warehouse-Architekturen großer börsennotierter Unternehmen vor, arbeitet deren Besonderheiten heraus und analysiert die zukünftigen Herausforderungen. Kapitel 3 gliedert sich in vier größere Abschnitte, die den gesamten Forschungsrahmen für diese Arbeit aufspannen: Im ersten Teil wird der klassische Data-Warehouse-Begriff definiert und ein Architektur-Referenzmodell für Data-Warehouse-Systeme beschrieben. Der zweite Abschnitt befasst sich mit der situativen Datenanalyse, einem verwandten Forschungsgebiet der Echtzeit-Data-Warehouse-Systeme. Der dritte Teil diskutiert die Rolle von Data-Warehouse-Systemen heute und erörtert neue Facetten und Anwendungen. Die Erkenntnisse des Kapitels werden im vierten Teil in einer Anforderungsanalyse zusammengefasst, gemeinsam mit der Formulierung der Problemstellungen dieser Dissertation. Die drei anschließenden Kapitel bilden den fachlichen Kern dieser Arbeit: Kapitel 4 stellt einen Algorithmus zur multikriteriellen Ablaufplanung vor und diskutiert dessen Anwendung in einem dynamischen Umfeld. Die Untersuchung von Ablaufplanungsalgorithmen in mehrstufigen Data-Warehouse-Szenarien ist Inhalt von Kapitel 5. Des Weiteren wird eine Visualisierungstechnik vorgestellt, mit deren Hilfe komplexe Datenproduktionsprozesse analysiert werden können. In Kapitel 6 werden Methoden und Techniken erarbeitet, welche die Datenstabilität auch in Echtzeit-Data-

Warehouse-Systemen garantieren. Kapitel 7 fasst die Ergebnisse dieser Arbeit zusammen und schließt sie mit einem Ausblick ab.

2 Fallstudien

Die in dieser Arbeit entwickelten Algorithmen und Verfahren werden unter Verwendung von sowohl synthetischen als auch realen Daten und Workloads ausführlich evaluiert. Nichtsdestotrotz sind die Experimentierumgebungen stets gewissen Annahmen unterworfen und können daher nur eine Annäherung an die Praxis darstellen. Die in dieser Dissertation erarbeiteten Lösungen sollen jedoch auch einem Abgleich mit realen Data-Warehouse-Projekten standhalten. Aus diesem Grund werden in den folgenden Abschnitten zwei Data-Warehouse-Projekte aus dem Bankensektor und der Marktforschung vorgestellt, deren Besonderheiten herausgearbeitet und Herausforderungen analysiert. Die Komplexität der einzelnen Szenarien ist zum Teil enorm, weshalb sich die Darstellungen nur auf wenige Details beschränken.
Es bleibt anzumerken, dass die in diesem Kapitel präsentierten Fallstudien keine konkreten Echtzeit-Data-Warehouse-Architekturen beschreiben. Stattdessen werden Data-Warehouse-Landschaften vorgestellt in denen Echtzeitanforderungen existieren und diesbezüglich in den nächsten Jahren ein Umbruch zu erwarten ist. Data-Warehouse-Architekturen, die Echtzeiteigenschaften implementieren, sind zum heutigen Zeitpunkt kaum vorhanden. Nach Aussage der Unternehmen die im Rahmen dieser Arbeit befragt wurden, ist nach dem starken Wachstum der Data-Warehouse- und Analyseanwendungen in den letzten Jahren zunächst eine Konsolidierung der Systeme notwendig, um im nächsten Schritt neue strategische Herausforderungen anzugehen. Das Thema Echtzeit wurde dabei stets hoch priorisiert.
Die Ergebnisse der Kapitel 4, 5 und 6 die den fachlichen Kern dieser Dissertation bilden sind im Folgenden jeweils im Kontext der Fallstudien zu diskutieren und zu bewerten.

2.1 Fallstudie A: UBS AG

Data-Warehouse-Systeme werden im Bankensektor seit vielen Jahren erfolgreich eingesetzt und sind für den erfolgreichen Betrieb zahlreicher Anwendungen sowie zur Unternehmenssteuerung unverzichtbar. Beispielhaft seien nur das Marketing bzw. das Customer Relationship Management, die Geldwäschebekämpfung, das Risikomanagement, die Missbrauchsprognose beim Einsatz von elektronischen Bezahlsystemen oder die Personalverwaltung genannt. Die Data-Warehouse-Landschaft der UBS AG, einer der weltweit führenden Banken, ist Inhalt der folgenden Abschnitte.

2.1.1 Unternehmen und Anwendungsdomäne

Die Schweizer UBS AG ist eine Großbank mit Unternehmenssitzen in Zürich und Basel und beschäftigt zirka 69.000 Mitarbeiter in 50 Ländern. Sie erbringt Finanzdienstleistungen für Privat- und Firmenkunden sowie institutionelle Anleger. Die UBS AG ist mit 2.233 Milliarden Franken verwalteter Kundengelder der größte Vermögensverwalter der Welt (Stand 2009 [2]). Der Konzern gliedert sich in vier Geschäftsbereiche: Global Wealth Management & Swiss Banking, Wealth Management Americas, Investment Bank und Global Asset Management.

Geschäftsbereiche Die Strategie des Konzerns konzentriert sich auf die Vermögensverwaltung, welche neben der Investmentbank den größten Anteil am Umsatz und Gewinn des Unternehmens ausmacht. Der Bereich Global Wealth Management verwaltet das Vermögen internationaler (mit Ausnahme Amerikas) privater und institutioneller Kunden, z.B. privatwirtschaftlicher und staatlicher Pensionskassen, Stiftungen, Gemeinden, wohltätiger Organisationen, Versicherungsgesellschaften, Regierungen, Zentralbanken sowie supranationaler Organisationen. Die Anlageentscheidungen werden durch eine vorherige Vermögensberatung durch den Kunden selbst oder durch einen Vermögensverwalter selbstständig getroffen. Die Schwerpunkte und Risikobegrenzungen werden bei der Vermögensverwaltung durch Anlagerichtlinien festgehalten. Der Bereich Wealth Management Americas bietet diese Dienstleistungen den amerikanischen Kunden. Die Swiss Bank stellt ihren Schweizer Kunden Finanzdienstleistungen wie Kontenverwaltung, E-Banking, Kartenzahlung, Finanzberatungen, Vorsorge, Versicherungen und Hypotheken zur Verfügung.
Der Bereich Investment Bank bietet Wertschriften sowie Studien und Beratung in den Bereichen Aktien, Festverzinsliches, Zinsen, Devisen und Edelmetalle an. Die Kunden der Investmentbank sind Firmen, Institutionen, Finanzintermediäre und Vermögensverwalter. Des Weiteren unterstützt die Investment Bank Unternehmen bei Kapitalaufnahmen, z.B. Börsengänge.
Der Geschäftsbereich Global Asset Management ist eine internationale Vermögensverwaltung und bietet Anlagelösungen für Privatkunden, Finanzintermediäre und institutionelle Anleger. Der Unternehmensbereich stellt Investment-Möglichkeiten in verschiedenen Anlageklassen, wie Aktien, Festgelder, Währungen, Hedge Funds, Immobilien, Infrastruktur- und Private-Equity-Anlagen, zur Verfügung.
Im Folgenden wird lediglich der Geschäftsbereich Global Wealth Management & Swiss Banking betrachtet.

2.1.2 Systemarchitektur

Das größte und umfassendste Data-Warehouse der UBS AG wird im Geschäftsbereich Global Wealth Management & Swiss Banking (im Weiteren abgekürzt mit WM&SB) betrieben. Dieses wird bezüglich der Anforderung für die Datenanalyse in drei Bereiche, sogenannte analytische Domänen, unterteilt: 1) *Sales Management*, das unter anderem die Produkt-, Markt- und Partneranalyse sowie die Überwa-

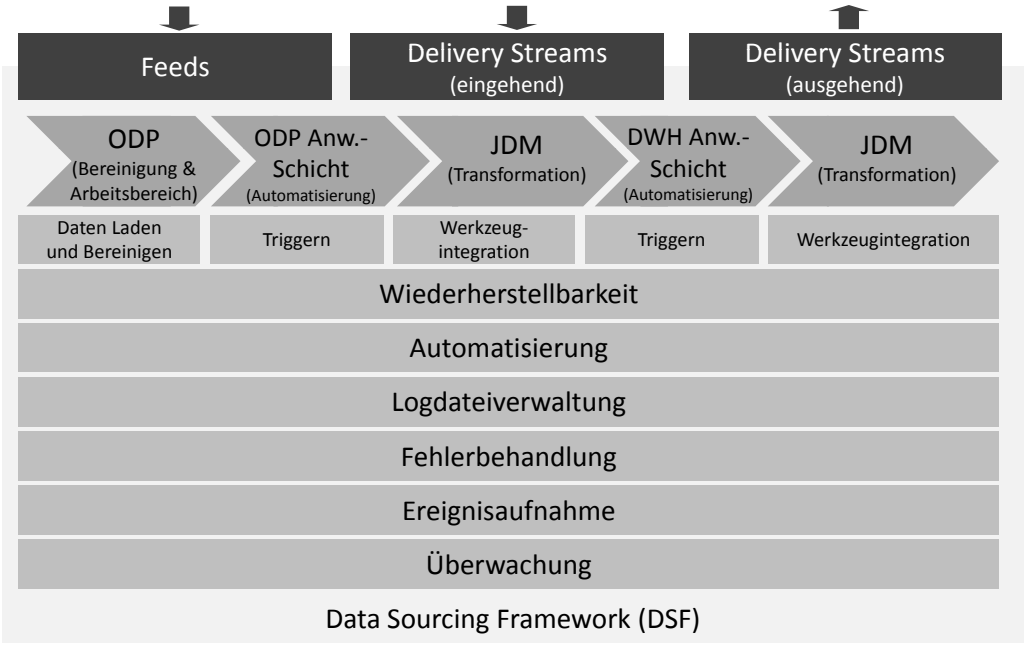


Abbildung 2.1: Übersicht des Data Sourcing Framework (DSF)

chung von Verkaufskampagnen beinhaltet, 2) *Production Management*, das die Optimierung von Finanzprodukten und -prozessen sowie die Analyse des Portfolios von Korrespondenzbanken umfasst und 3) *Finance Risk and Control*, in dem sich die Verantwortlichen mit der Rentabilität von Produkten und Kunden, der Risikoüberwachung von Kreditlinien sowie der ordnungsgemäßen Vergabe von Krediten beschäftigen. Weiterhin existiert eine analytische Domäne *Shared*, in der alle gemeinsam genutzten Daten, wie zum Beispiel Kalender-, Organisations- und Strukturdaten sowie Daten zu Finanzinstrumenten, der Bank gemeinsam verwaltet werden. Die gemeinsame Data-Warehouse-Architektur der analytischen Domänen sowie die Lade- und Transformationsprozesse werden im Folgenden detailiert betrachtet.

Data Sourcing Framework (DSF)

Die UBS verfolgt den sogenannten Single-Sourcing-Ansatz, d.h. jede Information wird nur einmal von den operativen Systemen in die Analyseplattform übernommen bzw. extahiert, tranformiert und bereinigt. Durch die vereinheitlichte Interpretation der Daten und die Vermeidung von Redundanzen wird zum einen der Implementierungsaufwand seitens der Datenquelle und des Data-Warehouse gesenkt und zum anderen die Datenkonsistenz und -qualität erhöht. Die Zahl der Schnittstellen zwischen operativen Systemen und Data-Warehouse kann somit reduziert und die Kopplung zwischen diesen Systemen vermindert werden.
Zwischen je zwei persistenten Datenschichten (z.B. das Data-Warehouse oder die

Data-Marts), dem Quell- und dem Zielsystem, existiert eine sogenannte Sourcing-Schicht, welche die Extraktion, die Transformation und das Laden der Daten implementiert. Zur Entkopplung der durch die Sourcing-Schicht verbundenen Schichten werden die Daten per Datentransfer oder über eine Datenbankverbindung ausgetauscht. Zwischen Datenquellen und Data-Warehouse werden die Daten per Datentransfer übermittelt. Die Kontrolle über den Zeitpunkt der Datenübermittlung obliegt den Datenquellen und wird typischerweise im Rahmen des Tagesabschlusses getätigt. Pull-Mechanismen bei dem Transfer in das Data-Warehouse werden unterstützt. Die Daten zwischen Data-Warehouse und Data-Marts werden per Push-Mechanismus transferiert. Obwohl die Sourcing-Schicht primär für den Datentransfer zwischen Quell- und Zielsystem konzipiert ist, können Datenkorrekturen bzw. abgeleitete Daten des Zielsystems zurück in das Quellsystem übertragen werden.
Die verschiedenen Sourcing-Schichten sowie deren einzelne Prozesse folgen einer standardisierten, Inhouse-entwickelten Implementierung, die durch das Data Sourcing Framework (DSF) bereitgestellt wird. Die Architektur und die Komponenten des Framework sind in Abbildung 2.1 illustriert. Es werden standardisierte Komponenten zum Laden der Daten über verschiedene Schnittstellen, zur Datenbereinigung (z.B. Datentypprüfung, Wertebereichsprüfung, Primärschlüsselvalidierung), zur Überwachung der Datenqualität, zur Fehlerbehandlung usw. bereitgestellt.
Des Weiteren ist die Datenproduktion durch die Definition von Ereignissen voll automatisiert. So können Zeitpunkte oder Zeitfenster durch die Verwendung verschiedener Kalenderdefinitionen und Perioden definiert werden, zu denen bestimmte Daten geliefert oder verarbeitet werden müssen. Die Festlegung von Datenabhängigkeiten erlaubt es, logisch zusammenhängende Datenpakete gemeinsam zu verarbeiten. Die zu übertragenden Daten werden zunächst im Operational Data Pool, kurz ODP, gesammelt und in der ODP-Anwendungsschicht zeitlich synchronisiert (siehe Abbildung 2.1).
Die ODP-Anwendungsschicht bildet den Kern des Data Sourcing Frameworks. Durch die Verwendung von Metadaten kann ermittelt werden, aus welcher der über 13.000 Staging-Tabellen die Anwendungen mit Daten zu versorgen sind und in welcher Frequenz die Daten propagiert werden müssen. Insgesamt existieren über 750 Datenquellen und über 3.000 verschiedene Daten- bzw. Dokumenttypen. Täglich werden zirka 2.000 bis 2.500 Dateien verarbeitet, die in summa je nach Tagesabschluss 290 bis 1.000 GB groß sind bzw. aus 700 bis 1.200 Millionen Datensätzen bestehen.
Unter Anwendung des Job Dependency Managers, kurz JDM, werden die Daten gegebenenfalls transformiert und in die Zieldatenbank geladen. Tranformationsprozesse werden durch PowerCenter™-Sessions (Informatica) bzw. durch SQL-Anweisungen realisiert. Diese sind fest in das DSF eingebettet und können über das Framework gestartet und zurückgesetzt werden. Weiterhin werden die Log- und Fehlerinformationen der PowerCenter-Session an das DSF propagiert (eingehender Delivery Stream in Abbildung 2.1) und dort zentral behandelt. Zur Durchführung der Datentransformationen existieren zirka 2.500 verschiedene PowerCenter-Sessions und 46.000 SQL-Anweisungen. Diese sind auf über 12.000 Jobs abgebildet, welche die Einheiten darstellen, in der die Daten in das Data-Warehouse bzw. in die Data-Marts geladen

werden. Einem Job wird unter Verwendung der Metadaten eine Priorität und der Grad der Parallelisierung zugewiesen. Höher priorisierte Jobs, die etwa Stammdaten enthalten, werden stets bevorzugt verarbeitet. Der Parallelisierungsgrad bestimmt die Anzahl der Jobs, die parallel verarbeitet werden können. Hoch priorisierten Jobs wird meist eine geringe Parallelisierung zugewiesen, um das Verhungern von niedrig priorisierten Jobs zu vermeiden. Jeden Tag müssen zirka 3.200 bis 5.000 Jobs verarbeitet werden.
Die DWH-Anwendungsschicht stellt ähnliche Funktionalität wie die ODP-Anwendungsschicht zur Verfügung. Der Ladezustand der Data-Warehouse-Tabellen wird überwacht und mit den Datenanforderungen der Anwendungen verglichen. Ist die Vollständigkeit für eine einzelne oder eine Menge von Datentabellen gegeben, so werden Trigger ausgelöst, durch die die Daten weiter propagiert werden.
Fehlerfälle im Data Sourcing Framework werden zum großen Teil automatisch behandelt und die Datenübertragungen neu eingeplant. Unbekannte Fehler werden zentral gespeichert und müssen von den Verantwortlichen bearbeitet werden. Eine Überwachungskomponente erlaubt die Erkennung von Überlastsituationen bzw. von Engpässen in der Datenverarbeitung. Durch die Aufzeichnung von Statistiken können damit frühzeitig Engpässe erkannt und durch den Ausbau der Hardwarekomponenten bzw. durch die Optimierung der Software vermieden werden.
Durch die im Data Sourcing Framework zentral entwickelten und bereitgestellten Komponenten für das Laden, die Bereinigung und die Transformation der Daten ist der Entwicklungs- und Wartungsaufwand erheblich reduziert. Ergänzt durch zentral steuerbare Überwachungs-, Ablaufplanungs-, Automatisierungs- und Fehlerbehandlungsmechanismen, kann die Datenqualität kontrolliert und in Folge erhöht werden.

Aufbau des Data-Warehouse

Das Data-Warehouse des Geschäftsbereichs WM&SB verwaltet Daten im Umfang von 60 TB (komprimiert) und hat zirka 17.500 Nutzer, von denen 16.000 mit statischen Berichten versorgt werden und weitere 1.500 mit Analysewerkzeugen auf die Daten zugreifen. Im Mittel muss das System jede Stunde zirka 500 bis 3.000 Anfragen verarbeiten. Die gesamte Data-Warehouse-Umgebung basiert auf IBM-pSeries-Hardware mit einem AIX-Betriebssystem. Als Datenbank kommt DB2 V9.5 LUW von IBM zum Einsatz.
Der vollständige Aufbau des Data-Warehouse ist in Abbildung 2.2 dargestellt. Bei der Datenaufnahme und -extraktion werden die Dateinamen entsprechend der Namenskonvention angepasst, Dateiparameter für die Weiterverarbeitung vorbereitet und die Dateien archiviert. Die Landing Area, als Teil des im vorhergehenden Abschnitt beschriebenen Data Sourcing Framework, dient zur temporären Speicherung der in das Data-Warehouse zu ladenden Daten. Diese werden unter Verwendung des DSF zunächst in das Data-Warehouse und dann weiter in die Data-Marts geladen, wo sie für die verschiedenen Analysewerkzeuge und das Reporting zur Verfügung stehen. Ausgewählte Komponenten der Data-Warehouse-Architektur werden im Folgenden genauer betrachtet.

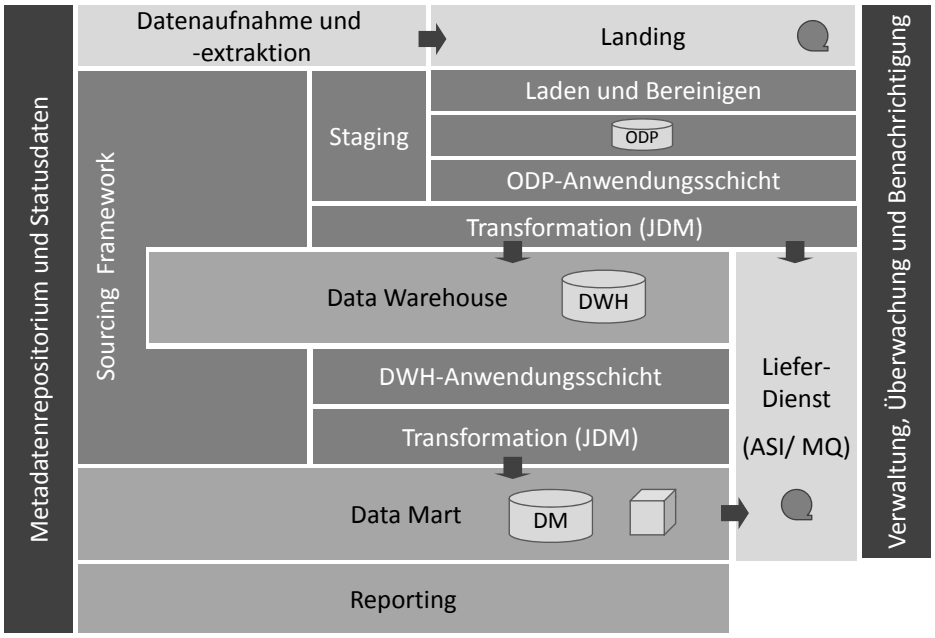


Abbildung 2.2: Data-Warehouse-Architektur

Data-Warehouse Die Data-Warehouse-Schicht konsolidiert die Daten aus über 700 verschiedenen Quellen, die logisch in eine Menge analytischer Domänen bzw. Data-Warehouses aufgeteilt sind. Jede dieser Domänen verwendet jedoch die gleiche physische Plattform.
Das logische Datenmodell für operative und analytische Systeme ist in vielen Fällen unterschiedlich, wodurch der Aufwand für die Datenintegration erhöht wird sowie die Datenqualität bei Fehlern in der Integration sinkt. In der UBS WM&SB wurden daher innerhalb der einzelnen analytischen Domänen die Terminologie, Semantik, Beziehungen, Schlüsselbezeichnungen usw. konsolidiert und vereinheitlicht. Dies ermöglicht die vereinfachte Integration sowie die Bestimmung der Herkunft der Daten (engl. *lineage*). In den analytischen Modellen werden zur optimalen Unterstützung der Auswertung noch weitere Beziehungen hinzugefügt. Ein unternehmensweites Datenmodell ist aufgrund der Größe und Komplexität des Systems bisher nicht realisiert. Für bereichsübergreifende Analysen ist daher eine wechselseitige Anpassung der Datenmodelle notwendig. Um die Wiederverwendbarkeit zu maximieren, werden lediglich Basisdaten in dritter Normalform vorgehalten. Für Zeitreihenauswertungen werden historische bzw. zeitraumbezogene Daten benötigt, die durch das Data-Warehouse zur Verfügung gestellt werden müssen.

Data-Marts und Cubes Die Data-Mart-Schicht stellt die für Geschäftsanalysen notwendigen Daten in effizienter und fachbereichsspezifischer Form zur Verfügung. Die Data-Marts werden ausschließlich durch die Data-Warehouse-Schicht versorgt.

Ein Data-Mart kann aus mehreren Data-Warehouses abgeleitet werden, wodurch redundante Datenhaltung auf der Data-Warehouse-Schicht vermieden wird. Der direkte Datenaustausch zwischen zwei Data-Marts ist gemäß des Architekturframeworks der analytische Systeme nicht gestattet. Ein Data-Mart kann stets neu aus den entsprechenden Data-Warehouses abgeleitet werden.
Für Analyseanforderungen, die zeitlich begrenzt sind, können sogenannte Sandbox-Data-Marts verwendet werden. Diese stehen lediglich einer bestimmten Nutzergruppe zur Verfügung und werden nach Abschluss der Analyse entweder gelöscht oder, falls die entsprechenden Ergebnisse bzw. Geschäftsmethodiken von breiterem Interesse sind, in die reguläre Data-Warehouse- oder Data-Mart-Schicht übernommen.
Zur Verbesserung der Anfrageperformanz wird das Datenmodell der Data-Marts hinsichtlich den Analyse- und Berichtsanforderungen optimiert (Denormalisierung) und die Komplexität reduziert. Dazu werden Ausschnitte gebildet, die Daten voraggregiert, durch die Materialisierung von Verbundergebnissen denormalisiert und abgeleitete Kennzahlen vorberechnet. In summa existieren zirka 5.000 Data-Mart-Tabellen und 2.000 Sandbox-Tabellen, die zur Datenanalyse benötigt werden. Zusätzlich werden zur Unterstützung spezieller OLAP-Werkzeuge, wie Business Objects oder Cognos, Cube-Strukturen erzeugt und für Analysen vorgehalten.

Reporting Die Reporting-Schicht bietet verschiedene Möglichkeiten, die Daten des Data-Warehouse und der Data-Marts für Analysen zu verwenden. Die Daten sind in den Data-Marts vorzuhalten, falls diese regelmäßig benötigt werden und das Datenvolumen sehr hoch ist. Andernfalls ist das Data-Warehouse vorzuziehen, um die Anzahl der Data-Marts gering zu halten. Eingebettet in ein Informationsportal werden statische Berichte in Form von Listen und PDF-Dokumenten zur Verfügung gestellt (monatlich über 5.000). Das Informationsportal überprüft die Datenauthorisierung einzelner Nutzer, um unberechtigten Zugriff zu vermeiden und schlägt anhand der Nutzerrolle bestimmte Berichte vor. Des Weiteren werden für zirka 1.500 Nutzer Adhoc-Analysen durch OLAP-Werkzeuge wie Business Objects, Cognos oder SAS unterstützt.

2.1.3 Besonderheiten und Herausforderungen

Die Data-Warehouse-Landschaft der WM&SB sieht sich vielen Herausforderungen gegenüber. Besonders hervorzuheben sind das wachsende Datenvolumen, die Entwicklung hin zu dienstorientierten Architekturen, um domänenübergreifende Analysen zu unterstützen sowie die zunehmenden Anforderungen nach echtzeitunterstützenden Analysewerkzeugen.

Wachsende Datenvolumina

Das Data-Warehouse umfasst zum Zeitpunkt Dezember 2009 zirka 60 TB an Daten sowie Indexstrukturen. Diese wurden bereits durch Anwendung der DB2-Kompressionsmechanismen reduziert, wodurch gleichzeitig auch die Anfrageperformanz ver-

bessert werden könnte. Das Datenvolumen wuchs zwischen den Jahren 2007 bis 2009 um den Faktor vier. Ein weiteres Wachstum ist zu erwarten. Da die Konsolidierung der Data-Warehouse- und Data-Mart-Schicht weitestgehend abgeschlossen ist und der Single-Sourcing-Ansatz bereits in zu großen Teilen umgesetzt wurde, ist eine Reduzierung des Datenvolumens durch eine Verschlankung der Data-Warehouse-Architektur kaum mehr möglich. Somit ist eine weitere Vergrößerung des Systems nur durch einen Ausbau der Hardware zu erreichen. Der Einsatz von AIX in Verbindung mit DB LUW und seiner Möglichkeit zur Partitionierung ermöglicht eine nahezu lineare Skalierung der Architektur.

Fachbereichsübergreifende und situative Datenanalysen

Die UBS AG ist aufgrund ihrer Größe in viele Fachbereiche und somit getrennte Verantwortungsbereiche unterteilt, die jeweils eine sogenannte analytische Domäne bilden. Innerhalb dieser Domänen ist das logische Datenmodell konsolidiert und vereinheitlicht. Für fachbereichsübergreifende Analysen ist jedoch eine Anpassung der Datenmodelle notwendig. Im Idealfall formulieren die Fachbereiche ihre Datenanforderungen an die IT-Abteilung in Form von Projekten, die in einem gewissen Zeitrahmen umgesetzt werden. Spezielle oder auch temporär begrenzte Reportingbedürfnisse haben in der Vergangenheit teil zu Eigenentwicklungen der Fachbereiche ohne Beteiligung der IT geführt. Diese Eigenentwicklungen fördern das Entstehen sogenannter Daten-Silos, durch welche unnötige Redundanz eingeführt wird und mit denen ein Kontrollverlust der IT einher geht. Des Weiteren ist die Korrektheit der so integrierten Daten nicht garantiert.
Diesen Entwicklungen gilt es entgegen zu wirken. Ein erster Schritt wurde durch die Schaffung einer gemeinsamen technischen Data-Warehouse-Plattform bereits getan, wodurch die technische Integration bereits entfällt. Für die fachliche Integration müssen seitens der IT flexible Schnittstellen über Fachbereichsebenen hinweg geschaffen werden. Zusätzlich ist eine föderierte Datenintegrationsschicht erforderlich, welche dem Nutzer eine globale und einheitliche Sicht auf die Daten bietet.
Neben den vorhandenen über 700 Datenquellen, die bereits in das Data-Warehouse geladen werden, existieren jedoch noch viele weitere für die Datenanalyse interessanten Ressourcen. Dies sind zum Beispiel erweiterte Anlageproduktinformationen oder Hintergrundinformationen zu Fondsmanagern, die aus frei verfügbaren Internetquellen integriert werden können. Die zumeist unstrukturiert vorliegenden Daten müssen jedoch mittels semantischer Analysen zunächst in eine strukturierte Form überführt werden. In der föderierten Datenintegrationsschicht muss die IT die externen Daten den Fachbereichen proaktiv zur Verfügung stellen bzw. Werkzeuge entwickeln, mit denen die Fachbereiche selbst fremde Daten erschließen können. Die konkrete technische Umsetzung eines solchen föderierten Systems wird in Abschnitt 3.2.3 tiefergehend diskutiert.

Echtzeitanforderungen

Viele der derzeitigen Data-Warehouse-Anwendungen benötigen ihre Daten lediglich auf tages-, wochen- oder monatsaktueller Basis. Dabei spielt vor allem die Stabilität und die Qualität der Daten eine Rolle, die oft nur nach Tagesabschluss garantiert werden kann. Die Forderung nach aktuellen Daten innerhalb des Tagesgeschäfts nimmt jedoch zu. So muss das Investitionsvolumen bestimmter Anlagen sehr schnell in Berichten sichtbar sein, obwohl die entsprechende Transaktion noch nicht abschließend bilanziert wurde. Eine schlechtere Datenqualität wird somit bewusst in Kauf genommen, um im Gegenzug sehr aktuelle Daten zu erhalten. Das Laden der Daten nach Tagesabschluss bleibt davon unberührt. Weitere Anwendungen, die Echtzeitdaten benötigen sind zum Beispiel die Betrugserkennung im Fall von Kreditkartenzahlungen oder das Aufdecken von Geldwäschegeschäften. In beiden Fällen müssen historische Daten, die ein gewisses Verhalten repräsentieren, gegen aktuelle Daten verglichen werden, um entsprechende Abweichungen zu erkennen.
Das in Abschnitt 2.1.2 vorgestellte Data Sourcing Framework ist in seiner jetzigen Form auf die Verarbeitung von Massendaten ausgelegt. Zur Prozessierung kontinuierlicher, kleiner Datenlieferungen, bei annähernd gleichbleibendem Durchsatz, sind daher Veränderungen am DSF erforderlich. Die auf diese Weise produzierten Daten können jedoch nur unter Vorbehalt in das Data-Warehouse geladen werden, da zur endgültigen Validierung stets der gesamte Datensatz eines Tages benötigt wird.

2.2 Fallstudie B: GfK Retail and Technology

Die zweite Fallstudie adressiert ein Anwendungsszenario aus dem Bereich der panelorientierten Marktforschung, für das aufgrund der anfallenden Datenvolumen und der Mengen und Komplexität der anfallenden Berichte der Einsatz eines Data-Warehouse-Systems ein zentrale Rolle spielt.

2.2.1 Unternehmen und Anwendungsdomäne

Die GfK-Gruppe ist ein weltweit führendes Marktforschungsinstitut mit mehr als 120 Tochtergesellschaften und rund 10.000 Mitarbeitern weltweit. Die Tochtergruppe GfK Retail&Technology ist verantwortlich für das Geschäftsfeld „Non-Food Tracking“ (langlebige Verkaufsgüter) und stellt periodisch internationale Kennzahlen zu technischen Gebrauchsgütern bereit. Dieser Unternehmensbereich verfügt mit STAR-TRACK (System To Analyze and Report on TRACKing Data) über ein Produktionssystem zur Erfassung, Konsolidierung und Analyse von Markforschungsdaten aus ca. 350.000 Geschäften in über 70 Ländern. Das System wird in Deutschland entwickelt und betrieben. Grundlage für das sogenannte Non-Food Retail Panel der GfK sind Datenlieferungen von Handelsunternehmen zu zentralen Marktkennzahlen wie verkauften Einheiten, Verkaufspreisen und Bestandszahlen, aus denen die GfK dann ein repräsentatives Marktbild ableitet. Bevor im folgenden Abschnitt die

Architektur von STARTRACK beschrieben wird, erfolgt zunächst eine detailierte Vorstellung des Anwendungszenarios.

Methodik der panelorientierten Marktforschung Die Analyse der Marktdaten geschieht durch Methoden der panelorientierten Marktforschung. Ein Panel ist dabei definiert als eine gleichbleibende Menge von Untersuchungseinheiten (in diesem Fall Verkäufe), die regelmäßig zu verschiedenen Zeitpunkten mit den gleichen Untersuchungsinstrumenten erhoben werden. Das Ziel besteht darin, Marktveränderungen wie sich verändernde Martkanteile oder Distributionen zu erkennen. Die zentralen Begriffe der Panelmethoden sind Grundgesamtheit, Stichprobe und Hochrechnung, auf die kurz eingegangen werden soll: Die Grundgesamtheit beschreibt eine sachliche, räumliche und zeitliche Abgrenzung innerhalb einer Menge von Untersuchungseinheiten. Sollen etwa Elektrohändler betrachtet werden, so geschieht eine sachliche Abgrenzung der Geschäfte anhand ihres Umsatzes oder des Verkaufsanteils von Elektrogeräten, die ein bestimmtes Mindestmaß erreichen müssen, um sich für die Grundgesamtheit zu qualifizieren. Die zeitliche Abgrenzung meint vor allem die Fixierung der Grundgesamtheit auf einen bestimmten Zeitraum. Zur Bestimmung der Grundgesamtheiten werden in der Regel alle zwei bis drei Jahre sogenannte Basisstudien durchgeführt [62].
Zur wirtschaftlichen Realisierung der Datenerhebungen werden Stichproben angelegt, die auf Basis der Grundgesamtheit nur einen kleinen Teilbereich aller Geschäfte bzw. aller Verkäufe beinhalten. Die Stichprobengröße wird dabei so gewählt, dass ein für die Anwendung akzeptierbarer statistischer Fehler nicht überschritten wird. Neben der Fehlergrenze ist die Größe der Stichprobe vor allem von der Heterogenität bzw. der Varianz der Grundgesamtheit abhängig. Um die Varianz, und damit die Stichprobengröße, zu reduzieren, wird die Grundgesamtheit in nach Jahresumsatz gestaffelte Schichten unterteilt. In einem Hochrechnungsschritt werden durch Multiplikation mit einem Hochrechnungsfaktor die tatsächlichen Verkäufe für eine Grundgesamtheit ermittelt.
Fehler bei der Dateneingabe oder -verarbeitung bzw. durch Sonderangebote in einzelnen Geschäften können zu Ausreißerwerten und damit zur Verzerrung der Stichproben führen. Um den dadurch entstehenden Fehler zu kompensieren, müssen Extremwerte bzw. Ausreißer durch Vergleiche mit anderen Warengruppen oder Geschäften identifiziert werden. Ein Handelsforscher kann anschließend durch Anpassung des Hochrechnungsfaktors den Einfluss des Extremwertes auf die Stichprobe reduzieren.

Verteilung des Datenbeschaffungsprozesses Da die wichtigsten Nutzer von Marktforschungsleistungen heute global engagiert sind, ist die Notwendigkeit für internationale Analysen offensichtlich. Die Beschaffung der Marktforschungsdaten kann jedoch, aufgrund fehlender länder- und marktspezifischer Kenntnisse, nicht zentral durchgeführt werden. Daher existieren weltweit viele strukturell unabhängige Niederlassungen, die für die jeweilige Beschaffung der Daten vor Ort und deren Bereitstellung für ein länderübergreifendes Reporting verantwortlich sind. Diese Struk-

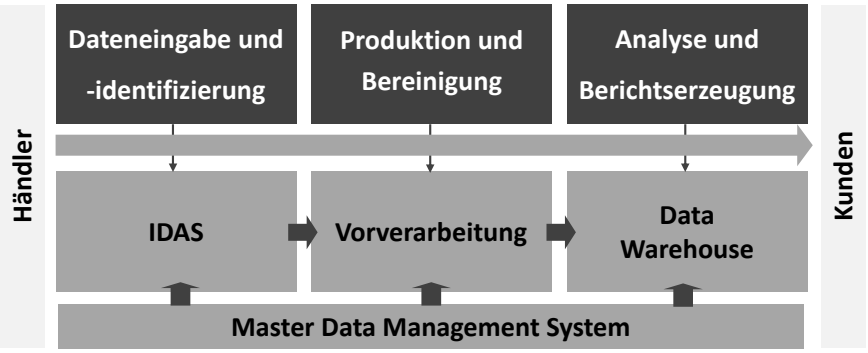


Abbildung 2.3: Aufbau von STARTRACK

tur hat jedoch zur Folge, dass die Marktforschungsdaten standortbedingt zeitlich versetzt produziert werden. Das heißt, der Prozess der Datenbereitstellung, beginnend bei der Identifikation, über die Qualitätskontrolle bis hin zur Speicherung im Data-Warehouse, findet zeitlich unkoordiniert statt. Während also die Marktberichte für die Länder sofort produziert werden können, sobald die Daten auf nationaler Ebene verfügbar sind, kann beispielsweise der Europabericht erst nach Datenverfügbarkeit im letzten Land erstellt werden. Zwischenzeitliche Fehlerkorrekturen, die unter Umständen erst nach Auslieferung der Länderberichte durchgeführt wurden, haben dann zur Folge, dass die im Europabericht für ein Land ausgewiesenen Zahlen von den Länderberichten abweichen können. Um die Konsistenz der Berichte zu gewährleisten, müssen derartige Korrekturen entweder durch Neuberechnung der Länderberichte publik gemacht oder, wenn die Auswirkungen auf die Berichtsqualität vernachlässigbar sind, zurückgestellt werden.

2.2.2 Systemarchitektur

Das System STARTRACK bildet die Wertschöpfungskette der GfK Marketing Services mit Hilfe einer Reihe von Komponenten ab, die in Abbildung 2.3 skizziert sind. Dies sind das Master Data Management (MDM) zur Verwaltung der Stammdaten für den gesamten Prozess, das International Data Acquisition System (IDAS) zur Datenerfassimg, ein System zur Vorverarbeitung und Bereinigung der Daten und das Data-Warehouse als Analyseplattform zur Erzeugung von Berichten. Der gesamte Wertschöpfungsprozess im Überblick beinhaltet die Erfassung der global produzierten Verkaufsdaten, die Prüfung und Bereinigung, die Abbildung auf international einheitliche Artikelkodierungen, die Überführung in Analyseformate, die Erzeugung von Kundenberichten und die Sicherstellung der Datenqualität über den kompletten Prozess hinweg. In summa sind 60 Verarbeitungsschritte notwendig, um die Rohdaten ausgehend von den Quellsystemen in Kundenberichte zu überführen [63].
Die Komponenten des STARTRACK-Systems sowie des Datenproduktionsprozesses im Detail werden im Folgenden dargestellt.

MDM Das Master Data Management (MDM) ist die zentrale Komponente des STARTRACK-Systems und dient zur Pflege der Artikel-, Geschäfts- und Kundenstammdaten. Als Beispiele für Artikelstammdaten lassen sich der Modellname oder die Warengruppen nennen, welche Artikel gleichen Typs zusammenfassen. Die Geschäftsstammdaten umfassen Informationen zu den Geschäftsstellen im Handel, deren geographische Lage, den Jahresumsatz des Geschäfts und weitere Kenngrößen. Kundenstammdaten enthalten zum Beispiel die vertraglichen Vereinbarungen der Kunden mit der GfK und die Produkte, die von ihnen bezogen werden.
Eine besondere Herausforderung bildet die Menge der Merkmalsattribute, die jeder Artikel haben kann und die von besonderem Interesse für die Auswertung sind. Je nach Warengruppe können bis zu 100 verschiedene Produktmerkmale erhoben werden. Am Beispiel der Warengruppe „Mobiltelefone“ sind dies etwa Bildschirmauflösung, Touch-Funktion, die Anzahl der unterstützten Netze usw. Die Eigenschaften der Artikel und Warengruppen sind durch technische Weiterentwicklung ständigen Veränderungsprozessen unterworfen, wodurch auch das Datenschema betroffen ist. Mit Hilfe des MDM können daher auf Basis eines Metaschemas die konkreten Datenschemata ermittelt werden.

IDAS Die Aufgabe des International Data Acquisition System, kurz IDAS, ist die Erfassung und Integration der international produzierten Handelsdaten. Hierbei handelt es sich im Gegensatz zum MDM um Bewegungsdaten, etwa Verkaufsmengen, Bestände, Preise usw. Die besondere Herausforderung bei diesem System besteht darin, die marktspezifischen Anforderungen in den einzelnen Ländern zu berücksichtigen und gleichzeitig die Integration der Daten, im Hinblick auf eine Speicherung im Data-Warehouse, zu gewährleisten. Dazu müssen Artikel, die im Handel unter verschiedenen Bezeichnungen geführt werden, durch eine Begriffskonsolidierung unifiziert werden. Des Weiteren ist eine Vereinheitlichung der Daten hinsichtlich der Formate, Granularitäten und Segmentebenen (filialgenau oder auf Ebene der Zentrale) erforderlich. Darüber hinaus findet durch einen Vergleich mit historischen Datenlieferungen eine Vollständigkeits- und Plausibilitätsprüfung statt. In bis zu 30% bis 50% der Fälle wird die Lieferung der Datenpakete verzögert und damit der Datenproduktionsprozess gestört. Eine häufige Ursache dafür ist, dass die Qualität der Daten eines Lieferanten unzureichend ist und sie durch andere Daten einer anderen Periode oder eines anderen Händlers ersetzt werden müssen. Dies ist möglich, da die integrierten Daten später nicht mehr einzeln betrachtet, sondern mittels Hochrechnung zu Marktübersichten aggregiert werden. Abschließend werden die Bewegungsdaten über eine Schnittstelle zur Weiterverarbeitung zur Verfügung gestellt.
Die Verarbeitung eines Datenpakets in einem Produktionsschritt wird als Produktionsauftrag bezeichnet. Durchschnittlich werden zirka 20.000 Produktionsaufträge pro Tag verarbeitet, in Spitzenzeiten sogar bis zu 200.000 Aufträge. Die Bearbeitungsdauer einzelner Aufträge reicht von wenigen Sekunden bis hin zu einigen Stunden, abhängig von der Datenpaketgröße, dem Produktionsschritt und der Notwen-

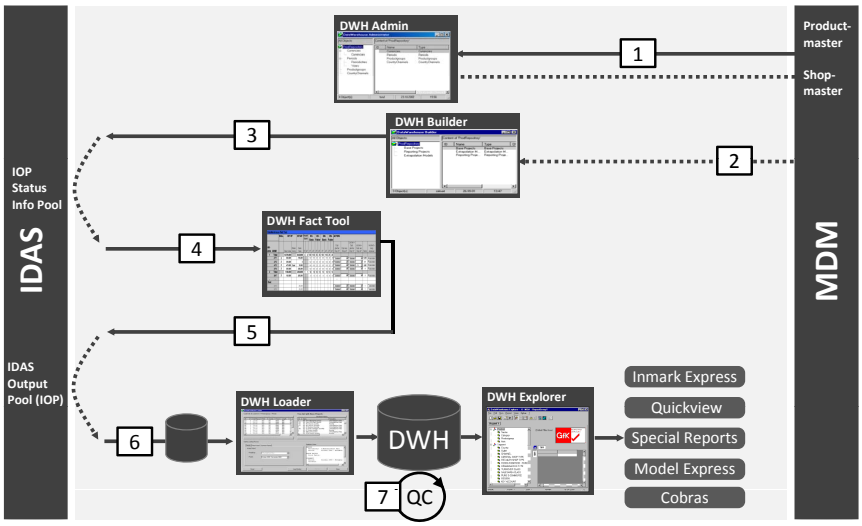


Abbildung 2.4: Ablauf eines STARTRACK-Prozesses

digkeit zur Nutzerinteraktion.
Der gesamte Verarbeitungsprozess kann über eine desktop- bzw. webbasierte Anwendung betrachtet und gesteuert werden, wobei die Datenhaltung in einer Oracle-Datenbank stattfindet.

Vorverarbeitung Hinter dem Begriff Vorverarbeitung stehen eine Reihe von Verarbeitungsschritten (siehe Abbildung 2.4), die inbesondere mit der Erzeugung von Berichtsprojekten, deren Befüllung mit Daten und der Qualitätssicherung befasst sind. Dazu werden sowohl die bereits vorgestellten Systeme MDM und IDAS als auch noch weitere Softwarewerkzeuge verwendet.
In einem ersten Schritt werden über den DWH-Administrator alle notwendigen Stammdaten, d.h. Warengruppen und Absatzkanäle, die zur Erzeugung der Berichtsprojekte notwendig sind, aus dem MDM extrahiert. Im nächsten Schritt erfolgt das Anlegen der sogenannten BaseProjects, die Daten einer oder mehrerer Produktgruppen gemeinsamer Periodizität beinhalten. Von den BaseProjects abgeleitet, werden des Weiteren ProductionProjects und ReportingProjects definiert, die spezifizieren, in welchen Einheiten die Daten geladen werden sollen bzw. welche BaseProjects zu einem Bericht zusammengefasst werden. In einem dritten Schritt erfolgt über die ProductionProjects der Bewegungsdatenabzug. Zu diesem Zweck werden zunächst in einem Produktionsprojekt mit Hilfe einer DataOrder neue Bewegungsdaten aus IDAS angefordert. IDAS nimmt diese Anforderung entgegen und stellt im sogenannten Output Pool zunächst nur aggregierte Kennzahlen bereit. Der Anwender kann in Schritt 4 mit Hilfe des FactTools die verdichteten Kennzahlen betrachten, fehlende Informationen erkennen und entsprechend mit Daten aus Vergleichsperioden kompensieren. Diese Modifikationen werden in sogenannten Load Definitions gespei-

chert. In Schritt 5 werden diese Load Definitions ausgeführt und über IDAS die Detailbewegungsdaten in entsprechende QC-Projekte (Quality Control) geladen. Dabei verändert IDAS die Bewegungsdaten entsprechend der Vorschrift aus der Load Definition. Im letzten Schritt öffnet der Anwender im DWH-Explorer die QC-Projekte, wo er die Detaildaten einsehen, Berichte erzeugen und feinere Korrekturen durchführen kann (Schritt 7). Sind diese Modifikationen beendet, erfolgt die Freigabe der Daten für die jeweiligen Basisprojekte.

Data-Warehouse Die in den vorhergehenden Schritten produzierten Daten werden abschließend zentral im Data-Warehouse der GfK Marketing Services abgelegt. Die Datenhaltung orientiert sich an einem Star-Schema mit den drei Hauptdimensionen Geschäft, Artikel und Zeit und einer Faktentabelle zur Speicherung von Kennzahlen wie etwa verkaufte Stückzahlen und Preise. Aus diesen Fakten werden verschiedene höherwertige, additive und nicht-additive Kennzahlen wie Umsätze bzw. Marktanteile und Distributionen errechnet. Die Analyse erfolgt seitens der Geschäfte hierarchisch nach Ländern und Vertriebskanälen sowie auf Produktseite nach Warengruppen und Kategorien. Zu diesem Zwecke werden mit dem DWH-Explorer sogenannte Standardberichte definiert, die regelmäßig berechnet werden. Pro Monat werden so zirka 100.000 nationale und internationale Berichte erzeugt, wobei zur Erstellung eines Berichts bis zu 100.000 SQL-Anfragen notwendig sind [25]. Die hohe Komplexität ist zum einen durch die Vielzahl der auszuweisenden Kombinationen von Artikeleigenschaften gegeben und zum anderen durch die Menge der Vertriebskanäle und Länder sowie die Unterteilung der Kennzahlen in Preisklassen bestimmt. Zur Verteilung der Anfragelast werden, abhängig von der Komplexität der Berichte und des Datenumfangs der Produktgruppe, periodisch kunden- bzw. domänspezifische Data-Marts angelegt.
Neben der statischen Berichtsproduktion ist es auch möglich, dass Kunden direkt über ein Online-Portal ad hoc eigene Berichte erzeugen. Diese besitzen niedrigere Komplexität als die Standardberichte, da sie nur einzelne Berichtsseiten umfassen. Allerdings sind die Anforderungen bezüglich der Antwortzeiten, welche im Bereich von wenigen Sekunden liegen, höher im Vergleich zur statischen Berichtsproduktion.

2.2.3 Besonderheiten und Herausforderungen

Die Anwendungsdomäne bzw. die daraus abzuleitende Data-Warehouse-Infrastruktur der GfK Marketing Services hat mehrere besondere Eigenschaften, die es hervorzuheben gilt und die die Entwickler vor besondere Herausforderungen stellen.

Operative Datenänderungen

Anders als in „klassischen“ Data-Warehouses (siehe Abschnitt 3.1.1), sind die Daten im Data-Warehouse der GfK nicht eingefroren, sondern können durchaus noch Modifikationen erfahren, sei es durch Korrekturen an den Dimensionsdaten oder auch

durch die Aufnahme nachträglicher Datenlieferungen auf Faktenebene. Letzteres resultiert im Wesentlichen aus der räumlich verteilten und dadurch zeitlich versetzten Datenproduktion (siehe Abschnitt 2.2.1). Die Menge der Verarbeitungsschritte bzw. die Länge des Datenproduktionsprozesses verstärkt diesen Effekt noch einmal und führt dazu, dass der Zeitpunkt der Datenlieferungen schwer vorhersagbar und daher nicht mit der Berichtsproduktion synchronisierbar ist.
Aufgrund dieser nachgelagerten Datenlieferungen bzw. Änderungen bereits eingebrachter Daten kann das Data-Warehouse der GfK Marketing Services auch als operatives Data-Warehouse bezeichnet werden.

Echtzeitanforderungen für Online-Analysen

Das bisherige Geschäftsmodell der GfK Retail&Technology befasste sich mit der Bereitstellung von Kennzahlen zu technischen Gebrauchsgütern, haupsächlich in Form von Wochen- und Monatsberichten. In Zukunft könnte sich das Geschäftsfeld um die Bereitstellung von Online-Analysen für elekronische Warenhäuser erweitern. Insbesondere für die Steuerung des Vertriebs virtueller Waren wie Musik, Spiele, Anwendung für das Mobiltelefon oder elektronische Bücher, steigt die Anforderungen nach hochaktuellen Daten. Durch den Wegfall von Warenhaltung, Vertriebslogistik, Kunden-Support usw. ist der Vertrieb von virtuellen Gütern viel dynamischer, so das die Preise ständig angepasst werden können und müssen. Die Verkaufsaktivitäten der elektronische Handelsplattformen sind dazu in Echtzeit auszuwerten und die Kennzahlen unmittelbar bereitzustellen. Für das neue Geschäftsfeld müssen daher die mehrstufigen Datenproduktionsprozesse vereinfacht und manuelle Qualitätsprüfungen und -änderungen entfallen.

Schema- und Anfragekomplexität

Das Datenvolumen des Data-Warehouse ist mit nur einem Terabyte eher gering. Die eigentliche Herausforderung liegt stattdessen in der Menge und Komplexität der Berichte und der daraus resultierenden Anfragenmenge. Die hohe Komplexität beruht vor allem auf der Vielzahl der dimensionalen Eigenschaftsattribute, die insbesondere zur Auswertung der Produkte herangezogen werden. Pro Warengruppe können dies allein in der Produktdimension 100 Dimensionen sein.
Um zum einen das hohe Anfragevolumen zu bewältigen und zum anderen die Antwortzeiten für Ad-hoc-Anfragen gering zu halten, ist eine sorgfältige Optimierung des Data-Warehouses, etwa durch Anlegen von Indizes und materialisierten Sichten, zwingend notwendig. Gerade Letztere sind jedoch nur sehr stark eingeschränkt nutzbar: Auf der einen Seite ist eine geeignete Auswahl der Aggregationsebenen aufgrund der hohen Dimensionalität nicht sinnvoll zu treffen – allein durch die 100 Dimensionen einer Warengruppe ergeben sich potenziell 2^{100} Aggregationsniveaus. Auf der anderen Seite sind viele wichtige Kennzahlen der Marktforschung, wie zum Beispiel Distributionen, nicht additiv berechenbar und daher nicht für Vormaterialisierungen

geeignet. Darüber hinaus ist die Wartung der materialisierten Sichten angesichts der ständigen Datenänderungen nicht effizient durchführbar.

3 Evolution der Data-Warehouse-Systeme und Anforderungsanalyse

Data-Warehouses haben sich seit ihrer ersten Erwähnung 1988, damals noch unter dem Begriff „Information Warehouse“, enorm weiterentwickelt und bilden heutzutage die Grundlage für wichtige Entscheidungsfindungen in vielen Unternehmen. Im Zuge der Globalisierung und der damit einhergehenden zunehmenden Vernetzung und dem wachsenden Wettbewerb haben sich jedoch auch die Anforderungen an entscheidungsunterstützende Systeme verändert. Die Auswirkungen dieser Änderungen auf das Konzept des Data-Warehouses sowie auf dessen unterstützende Technologien bilden den wesentlichen Inhalt dieses Kapitels.
Im ersten Teil wird zunächst der klassische Data-Warehouse-Begriff definiert und ein Architektur-Referenzmodell für Data-Warehouse-Systeme beschrieben. Der zweite Abschnitt behandelt die situative Datenanalyse, die eine häufige Anforderung im Kontext von Echtzeit-Data-Warehouse-Systemen darstellt. Bezugnehmend auf die Fallstudien aus Kapitel 2, diskutiert Abschnitt 3.5 die Rolle von Data-Warehouse-Systemen und erörtert neue Aspekte und Anwendungen. Abschnitt 3.5 fasst das Kapitel in Form einer Anforderungsanalyse zusammen und definiert die dieser Dissertation zugrunde liegenden Problemstellungen.

3.1 Der Data-Warehouse-Begriff und Referenzarchitektur

Bevor in den folgenden Abschnitten auf konkrete Veränderungen der Data-Warehouse-Systeme eingangen wird und die Anforderungen an diese analysiert werden, erfolgt zunächst eine Konkretisierung der grundlegenden Konzepte. So wird zu Beginn der Begriff Data-Warehouse als Namensgeber des Data-Warehouse-Systems definiert. Anschließend werden der typische Aufbau eines Data-Warehouse-Systems sowie dessen Bestandteile anhand einer Referenzarchitektur erläutert. Die Referenzarchitektur ist als idealtypisches Modell zu interpretieren, das als Leitfaden für die Entwicklung eines Data-Warehouse-Systems dient. Praktische Umsetzungen dieser Modellarchitektur können, aufgrund bestehender Altsysteme oder organisatorischer Restriktionen, durchaus abweichen. Langfristig ist aber stets eine Annäherung der konkreten Data-Warehouse-Systeme an die Referenzarchitektur anzustreben.

3.1.1 Definition des klassischen Data-Warehouse-Begriffs

Der Begriff Data-Warehouse, als zentraler Bestandteil eines Data-Warehouse-Systems, bezeichnet weder eine Technik noch ein Produkt, sondern vielmehr ein Konzept

zur Speicherung und Verarbeitung von Daten. Da dieser Begriff in der Literatur nicht eindeutig festgelegt ist, werden anhand der verbreitetsten Definitionen dessen wesentliche Charakteristika herausgearbeitet: Nach Kimball et al. ist ein Data-Warehouse „[..] a copy of transaction data specifically structured for querying and reporting." Diese Definition grenzt somit Data-Warehouses von operativen Systemen ab und beschreibt die Art der Verwendung. Während transaktionale Systeme wesentlich im Tagesgeschäft eines Unternehmens zu finden sind (beispielsweise bei der Bearbeitung von Flugbuchungen), liegt der Fokus bei Data-Warehouses auf der Datenanalyse. Die Verschiedenheit in der Anwendung hat auch unterschiedliche Nutzerzahlen zur Folge. Während die operativen Daten von einem großen Nutzerkreis verwendet werden, ist der Einsatz des Data-Warehouses nur auf einen kleineren Kreis, meist auf das Management eines Unternehmens, beschränkt. Ein Data-Warehouse hat somit die Aufgabe der organisatorischen Trennung der operativen Geschäftsdaten von den Daten, die zur Geschäftsanalyse bzw. zur Entscheidungsunterstützung verwendet werden. Die Trennung geschieht durch Kopieren der Daten bzw. durch eine Datenübernahme aus den Quellsystemen in das Data-Warehouse.
Eine zweite und sehr verbreitete Definition bieten Inmon et al. mit ihrer Beschreibung der Datenhaltung bzw. der in einem Data-Warehouse gespeicherten Daten: „A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions." Die Daten eines Data-Warehouses sind somit entsprechend des Analysezwecks themen- bzw. fachorientiert, d.h. sie werden nach bestimmten Attributen wie z.B. Region oder Produktgruppe ausgewählt. Das zweite Merkmal, „Integration", beschreibt die Eigenschaft eines Data-Warehouses als integrierte Sicht auf Daten verschiedener Quellsysteme in vereinheitlichter und strukturierter Form. Um dies zu erreichen, ist eine Anpassung und Transformation der Daten notwendig. Das Merkmal „nichtvolatil" formuliert den Anspruch einer dauerhaften und physischen Datenspeicherung. Der Zugriff auf das Data-Warehouse erfolgt nur lesend, d.h. Daten werden lediglich hinzugefügt und bereits in das Data-Warehouse eingebrachte Daten werden nicht mehr entfernt oder geändert. Dies grenzt Data-Warehouses von föderierten Datenbanksystemen ab, die zwar ebenfalls eine integrierte Sicht auf die Daten bereitstellen, jedoch ohne diese zu materialisieren. Zur Unterstützung der Datenanalyse im Hinblick auf zeitliche Veränderungen („time-variant") ist des Weiteren eine Zeitdimension erforderlich. Jedes Datum erhält somit einen Zeitstempel, wodurch eine Historisierung der Daten als auch Analysen über die Zeit hinweg ermöglicht werden.

3.1.2 Referenzarchitektur

Als Ausgangspunkt für weitere Diskussionen bzw. die Vertiefung spezifischer Aspekte wird zunächst das bekannteste Architekturreferenzmodell für Data-Warehouse-Systeme vorgestellt. Dieses Modell wurde in der Literatur unter anderem in [49, 8, 60] beschrieben. Es adressiert die allgemeinen Forderungen, wie Persistenz, Effizienz, Wiederverwendbarkeit, Flexibilität, Skalierbarkeit usw., die an eine Architektur zur Datenanalyse gestellt werden und beschreibt die Rolle einzelner Komponenten sowie

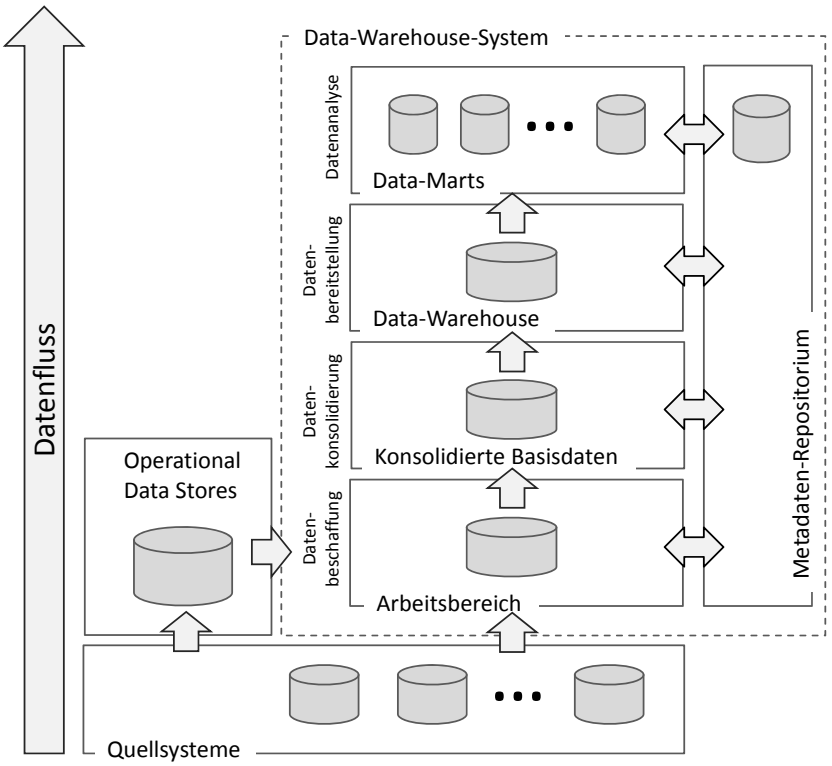


Abbildung 3.1: Data-Warehouse-Referenzarchitektur

den Datenfluss zwischen diesen. Parallel zur Beschreibung der Referenzarchitektur wird die Rolle der einzelnen Bestandteile in einem Echtzeit-Data-Warehouse-System diskutiert.
Wie im vorhergehenden Abschnitt bereits erläutert, bildet ein Data-Warehouse-System die Infrastruktur zur Datenanalyse. Da sich die zu analysierenden Daten aus einer Vielzahl von Quellsystemen speisen, müssen diese in einem aufwendigen Prozess zunächst bereinigt, transformiert, konsolidiert und in ein für die Analyse optimiertes Format gebracht werden. An dieser schrittweisen Verfeinerung der Daten sind eine Vielzahl von Systemen beteiligt, die im gemeinsamen Zusammenwirken das Data-Warehouse-System bilden. Die Einzelsysteme sowie deren Kopplung zu einem mehrstufigen Datenproduktionsprozess werden im Folgenden beschrieben. Eine schematische Darstellung dieser Referenzarchitektur ist dazu in Abbildung 3.1 gegeben.

Quellsysteme

Aus architektonischer Sicht nicht zum Data-Warehouse gehörend, jedoch Ursprung aller im Data-Warehouse erfassten Daten sind die Quellsysteme. In der Praxis sind dies einfach strukturierte Dateien, XML-Daten, Excel-Tabellen, komplexe relationale

Datenbanken, SAP-Systeme usw. Die in diesen Systemen enthaltenen Daten werden in einem Extraktionsschritt in das Data-Warehouse-System überführt. Aus konzeptioneller Sicht werden Quellsysteme bezüglich zweier Dimensionen klassifiziert: zum einen nach der Verfügbarkeit der Datenquellen, zum anderen nach der Herkunft der Daten. Die Verfügbarkeit, in [49] auch als Kooperationsbereitschaft der Quellsysteme bezeichnet, beschreibt, inwieweit der Extraktionsprozess durch das Quellsystem unterstützt wird. Eine Extremform bilden Replikationsquellen, in denen Änderungen am Quellsystem synchron in den Arbeitsbereich des Data-Warehouse-Systems übernommen werden. Abstufungen im Hinblick auf eine schwächer werdende Kooperationsbereitschaft bilden Aktive Quellen, Snapshot-Quellen, Exportbasierte Quellen, Logbasierte Quellen und Nicht Unterstützende Quellen. Quellen letzteren Typs können jedoch auch – durch die Implementierung spezieller Adaptoren – dem Extraktionsprozess zugänglich gemacht werden.
Einen weiteren Aspekt der Kooperationsbereitschaft bildet der Umfang der Daten, der jeweils pro Extraktionsvorgang kopiert werden kann. Im schlechtesten Fall kann immer nur der vollständige Datenbestand eines Quellsystems extrahiert werden. Dieses Vorgehen ist jedoch nur bei sehr vielen Änderungen an den Quelldaten zwischen den einzelnen Extraktionsschritten sinnvoll. Bei wenigen Änderungen zwischen den Extraktionsschritten sind Quellsysteme zu favorisieren, welche die Extraktion der Nettoänderungen unterstützen. Bei diesen Systemen werden jeweils nur die Änderungen, also hinzugefügte oder gelöschte Daten bzw. Aktualisierungen, an das Data-Warehouse-System propagiert.
In Bezug auf die Herkunft erfolgt eine Unterscheidung der Quellsysteme in Datenlieferanten für externe bzw. interne Daten. Externe Datenquellen sind organisatorisch vom Data-Warehouse-System separiert, wodurch die Möglichkeiten zur Datenextraktion meist begrenzt sind. Eine Ausnahme bilden Quellsysteme, die als Dienst in einer service-orientierten Architektur (SOA) bereitgestellt werden. Als Beispiele für externe Quellsysteme sind die Daten fachfremder Abteilungen oder weitergehende Daten von Drittanbietern wie Marktforschungsinstituten, Kommunen usw. zu nennen. Bei Quellsystemen, die interne Daten bereitstellen, handelt es sich meist um operative Systeme, die Produktionsdaten zur Verfügung stellen, oder um von Mitarbeitern gepflegte Daten, z.B. Excel-Tabellen, oder um Daten, die aus dem Data-Warehouse-System selbst in das Quellsystem zurückfließen.
Im Hinblick auf eine echtzeitfähige Architektur müssen Quellsysteme mehrere Voraussetzungen erfüllen: Zum einen müssen sie, um der Anforderung nach schneller Sichtbarkeit von Datenänderungen im Data-Warehouse-System gerecht zu werden, Datennettoänderungen zur Verfügung stellen. Da jede einzelne Änderung an das Data-Warehouse propagiert wird, muss dies durch das Quellsystem unterstützt werden. Aus Sicht der Unterstützung des Extraktionsprozesses sind Replikationsquellen oder Aktive Quellen zu bevorzugen, da diese selbstständig Änderungen an das Data-Warehouse-System propagieren. Dies entspricht dem Paradigma der push-basierten Propagierung von Datenänderungen, einer Grundanahme in echtzeitfähigen Data-Warehouse-Systemen (siehe Abschnitt 3.5). Für weniger unterstützende Quellsysteme kann diese Eigenschaft durch die Implementierung einer Zwischenschicht emu-

liert werden. Im Fall von Snapshot-Quellen müssten dazu zwei aufeinanderfolgende Snapshots miteinander verglichen und daraus die Datenänderungen abgeleitet werden. Der dazu nötige Aufwand kann jedoch den Extraktionsprozess stark verzögern, was im Widerspruch zu den Echtzeitanforderungen steht (siehe Abschnitt 3.5.1). Eine Anpassung des Quellsystems hinsichtlich der Anforderungen einer Echtzeitarchitektur ist für interne Quellen meist einfacher realisierbar als für externe Quellen, an denen keine Änderungen erlaubt sind.
Im weiteren Verlauf dieser Arbeit werden die Begriffe Quellsystem, Datenproduzent und Datenlieferant synonym verwendet.

Arbeitsbereich

Der Arbeitsbereich, im Englischen als „staging area“ bezeichnet, ist ein temporärer Datenspeicher, eingebettet zwischen den Quellsystemen und der Basisdatenbank. Die Transformationen zur Konsolidierung, Bereinigung und Integration der Daten werden auf dieser Datenbank ausgeführt. Erst nach Abschluss der Transformationen erfolgt das Laden in die Basisdatenbank. Dadurch wird der mitunter sehr aufwendige Transformationsprozess von den Quellsystemen und der Basisdatenbank entkoppelt und diese dadurch entlastet. Nach erfolgreicher Einbringung in die Basisdatenbank werden die Daten aus dem Arbeitsbereich gelöscht. Die Zeit des Hinzufügens der Daten in den Arbeitsbereich, und damit in das Data-Warehouse-System, wird als „load time“ bezeichnet [49].

Konsolidierte Basisdatenbank

Zwischen dem Arbeitsbereich und den Data-Warehouses ist die konsolidierte Basisdatenbank zur organisationsübergreifenden und anwendungsunabhängigen Datenspeicherung angesiedelt. Die Basisdatenbank, oft auch als Kooperativer Datenspeicher bezeichnet, dient somit zur Datenintegration als auch zur Weiterverteilung der Daten an anwendungsspezifische Data-Warehouses bzw. Anwendungen.
Die Basisdatenbank ist eine optionale Datenschicht, die besonders bei hohen Anforderungen bezüglich der Betriebsbereitschaft bei der Datenwiederherstellung oder bei der Reorganisation des Data-Warehouses verwendet wird. Die Daten der Basisdatenbank sind sehr feingranular bzw. unaggregiert, vollständig und umfassend. Dies bedeutet, dass die Daten unabhängig von ihrer aktuellen Verwendung persistiert werden, d.h. Spalten einer Tabelle werden auch gespeichert, wenn sie aktuell nicht benötigt werden. Dadurch kann im Fall unvorhergesehener Ereignisse im Data-Warehouse-Betrieb schnell reagiert werden.
Die konsolidierte Basisdatenbank stellt technisch vereinheitlichte, integrierte und somit anwendungsneutrale Daten zur Verfügung. In Abhängigkeit von den Anforderungen der Anwendungen werden eine Reihe von Optimierungen durchgeführt: So werden Datensätze, die häufig in Kombination abgefragt werden, persistiert. Des Weiteren erfolgt in dieser Schicht eine Anreicherung um Daten, die häufig auf Geschäftsseite benötigt werden, z.B. die Umrechnungswerte von Umsätzen in andere

Währungen. Diese Maßnahmen verringern die technische Komplexität auf der Anwendungsseite und sorgen damit für eine hohe Akzeptanz seitens der Anwender.
Statt des Indirektionsschritts über das Data-Warehouse können die Daten der Basisdatenbank auch direkt zu Analysezwecken genutzt werden. Dies ist vor allem für Anfragen mit hohen Datenaktualitätsanforderungen von Nutzen. Ab dem Zeitpunkt des Hinzufügens neuer Daten in die Basisdatenbank sind diese für den Nutzer des Data-Warehouse-Systems verwendbar. Dieser Zeitpunkt wird daher, in Abgrenzung zur „load time“ des Arbeitsbereiches, als „refresh time“ bezeichnet.
In der Praxis wird häufig auf die Basisdatenbank verzichtet und die Daten werden stattdessen direkt in das Data-Warehouse geladen. Dies liegt zum einen in den Kosten begründet, die durch diese zusätzliche Datenschicht verursacht werden. Meist sind die Gründe jedoch organisatorischer Natur: Eine zentrale unternehmensweite Datenbasis scheitert oft an der fehlenden Kommunikation und den widerstreitenden Anforderungen beteiligter Fachbereiche sowie an der Frage, durch welchen Fachbereich die Kosten zu finanzieren sind.

Data-Warehouse

Aufbauend auf der konsolidierten Basisdatenbank werden die Daten in einem weiteren Schritt in das Data-Warehouse übernommen. Dieses ist speziell für Analyseszenarien konzipiert. Bei relationalen Datenbanken beginnt dies beim Entwurf durch Verwendung eines Star- bzw. Snowflake-Schemas. Auf Instanzebene werden im Zuge sogenannter redundanzbehafteter Optimierungen Daten vorverdichtet, komplexe Kennzahlen vorberechnet und materialisiert. Des Weiteren implementieren die meisten Datenbanksysteme spezielle Zugriffsstrukturen, z.B. Bitmap-Indexe, oder Verbundoperationen, z.B. den Star-Join-Verbund, zur Unterstützung von Data-Warehouse-Anfragen. Eine Ergänzung der Zugriffsstrukturen leistet die horizontale Partitionierung der Daten. Diese wird in Abhängigkeit des Anfrageprofils meist für die Faktentabelle umgesetzt.
Neben dem Analyseprozess wird auch das Laden der Daten in das Data-Warehouse durch verschiedene Techniken unterstützt. Die Daten können für relationale Datenbanken über eine SQL-Schnittstelle geladen werden oder, wenn große Datensätze möglichst effizient eingebracht werden müssen, über den Einsatz sogenannter Massenlader (engl. bulk loader). Die verbesserte Effizienz im Vergleich zum herkömmlichen Laden wird über die Deaktivierung bestimmter Datenbankfunktionen, wie transaktionale Sicherheit und Konsistenzprüfung, erreicht.

Data-Marts

Data-Marts werden, wie auch schon das Data-Warehouse, ebenfalls für die Datenanalyse verwendet, bilden jedoch strukturell als auch inhaltlich nur eine Teilmenge des unternehmensweiten Datenbestandes ab. Auch ist die Granularität der Daten durch Aggregationen verringert, was jedoch in einer verbesserten Anfrageperformanz im Vergleich zum Data-Warehouse resultiert. Eine weitere Bereinigung oder Transfor-

mation der Daten findet in der Regel nicht statt, so dass Anfrageergebnisse aus dem Data-Warehouse und den Data-Marts, bei gleichem Aktualisierungsgrad, stets konsistent zueinander sind. In der hier vorgestellen Referenzarchitektur werden die Data-Marts intern durch das Data-Warehouse aktualisiert und aufgrund dieser Interdependenz als abhängige Data-Marts bezeichnet. Den Gegensatz dazu bilden die sogenannten unabhängigen Data-Marts, die direkt über die Quellsysteme versorgt werden. Der Vorteil dieses Typs liegt in der einfacheren, schnelleren und meist kostengünstigeren Erstellung, oft auch durch die Fachabteilungen selbst. Die Unabhängigkeit zu anderen Systemen begünstigt auch ein schnelles Laden bzw. Aktualisieren dieser Data-Marts. Mit wachsender Anzahl solcher unabhängigen Data-Marts wird es jedoch zunehmend schwieriger, diese gegeneinander konsistent zu halten. Aufgrund dessen und wegen der unterschiedlichen organisatorischen Verantwortungsbereiche ist eine unternehmensweite Analysemöglichkeit mit unabhängigen Data-Marts nicht möglich.
Im Hinblick auf die Skalierbarkeit der Anfrageverarbeitung mit steigenden Nutzerzahlen bildet ein einzelnes Data-Warehouse den Engpass eines Data-Warehouse-Systems. Data-Marts verteilen den Data-Warehouse-Datenbestand und dienen somit der Lastverteilung und -optimierung, müssen jedoch auch aktualisiert und gepflegt werden.
Die Abgrenzung zwischen Data-Warehouse und Data-Mart ist in der Praxis nicht immer klar gegeben, da Data-Warehouses nur selten eine unternehmensweite Integration leisten und somit nur einen Ausschnitt abbilden. Die Anforderung der Anwendungsneutralität ist somit nicht erfüllt. Andererseits sind viele Systeme, trotz der eingeschränkten Datensicht, zu komplex, um noch von einem Data-Mart sprechen zu können. Die Referenzarchitektur als Leitbild für den Entwurf eines Data-Warehouse-Systems bleibt gleichwohl gegenwärtig, muss jedoch für den konkreten Anwendungsfall variiert werden.

Operational Data Store

Das Data-Warehouse kann zur Verarbeitung von sehr aktuellen Informationen zusätzlich um sogenannte *Operational Data Stores*, kurz ODS, erweitert werden (siehe Abbildung 3.1). Inmon beschreibt in [38] einen Operational Data Store als „eine themenorientierte, integrierte, nicht dauerhafte detaillierte Sammlung von Daten, um Organisationen mit aktuellen, betrieblichen, integrierten und gesamtheitlichen Informationen zu unterstützen.“ Die Daten des Operational Data Store sind, im Gegensatz zum Data-Warehouse, von feinerer Granularität und zeigen stets nur ein aktuelles Abbild, d.h. Analysen auf historischen Daten sind nicht möglich. Die wesentliche Eigenschaft eines ODS besteht jedoch in den kurzen Aktualisierungszyklen. Ein ODS unterstützt somit die zeitnahe Auswertung einfach strukturierter Anfragen auf meist kleineren Datenbeständen. Ein Beispiel hierfür ist die Ermittlung der letzten Kundenbestellung in einem Call-Center.
Neben dem parallelen und unabhängigen Betrieb eines Operational Data Store kann dessen Datenbestand ähnlich denen der Quellsysteme dem Data-Warehouse-System

zugeführt werden. Aus Projektsicht werden Operational Data Stores meist von den Fachabteilungen selbst implementiert und verwaltet. Wie auch bei den unabhängigen Data Marts können aus solchen Projekten über die Zeit hinweg Data-Warehouse-Systeme erwachsen.
Im Zuge der steigenden Echtzeitanforderungen an Data-Warehouse-Systeme bilden Operational Data Stores einen möglichen Lösungsansatz, sind aber im Hinblick auf den Umfang der Daten, die Anzahl integrierter Quellsystem und die Qualität der Datenaufbereitung stark beschränkt und daher nur für sehr einfache Szenarien anwendbar.

Metadatenrepositorium

Das Metadatenrepositorium speichert Informationen zu allen vom Data-Warehouse verwalteten Daten sowie prozessbezogene Informationen über die Datenverarbeitung. Das sichert zum einen die Nachvollziehbarkeit des Datenproduktionsprozesses aus Sicht der Endanwender und zum anderen die effiziente Verwaltung des Systems durch die Administratoren. Um eine durchgängige Dokumentation des Gesamtsystems zu gewährleisten, muss, wie in Abbildung 3.1 dargestellt, jede Komponente des Data-Warehouse-Systems Änderungen mit dem Metadatenrepositorium synchronisieren.

3.2 Situative Datenanalyse

Der Untersuchungsgegenstand dieser Arbeit besteht in der Bereitstellung von Echtzeitdaten im Kontext von Data-Warehouse-Systemen. Das Thema motiviert sich aus der zunehmenden Forderung der Data-Warehouse-Anwender nach sowohl historischen als auch hochaktuellen Daten für ihre Analysen. In diesem Zusammenhang wird häufig der Begriff der *situativen Datenanalyse* (engl. situational BI) verwendet. Das Wesen der situativen Datenanalyse im Vergleich zu Echtzeit-Data-Warehouse-Systemen besteht darin, dass die zu verwendenden Datenquellen dem Data-Warehouse-Betreiber a priori nicht bekannt und ad hoc dem Nutzer zur Verfügung zu stellen sind. Die Anforderung nach der zeitnahen Integration neuer Datenquellen stellt für die heutigen Data-Warehouse-Architekturen jedoch ein Problem dar. Diese sind aufgrund der Forderungen nach hoher Datenqualität und Konsistenz relativ starr und dürfen nur innerhalb wohldefinierter Entwicklungszyklen verändert werden.
Zur Unterstützung situativer Datenanalysen existieren diverse Methoden und Lösungsansätze sowohl auf organisatorischer als auch auf technischer Ebene. Im ersten Fall ist dies die Verbesserung der Kommunikation zwischen IT (Dienstleister) und Fachbereichen (Anwender) sowie des Projektmanagements (Abschnitt 3.2.1). Auf technischer Ebene werden sogenannte Spreadmart-Lösungen eingesetzt, die im Abschnitt 3.2.2 diskutiert werden. Ein zweiter Lösungsansatz basierend auf dienstorientierten Architekturen und Mashups wird in Abschnitt 3.2.4 vorgestellt. In Abschnitt

3.2.4 werden abschließend die Vor- und Nachteile der verschiedenen Ansätze zusammengefasst sowie deren Einfluss auf die Projektkosten der IT und der Fachbereiche verglichen.

3.2.1 Interaktion zwischen IT und Fachbereich

Für den erfolgreichen Betrieb eines Data-Warehouse-Systems ist die Integration von Fachbereichswissen unbedingt erforderlich. Nur die Fachbereiche sind in der Lage, ihre inhaltlichen und operativen Anforderungen zu formulieren, die dann von der IT umgesetzt werden müssen. Dieses Zusammenspiel zwischen IT und Fachbereich führt in der Praxis jedoch häufig zu Problemen und wird durch den Trend der situativen Datenanalysen noch weiter verstärkt. Im Folgenden werden die Interaktionspunkte zwischen der IT und den Fachbereichen betrachtet sowie Lösungsansätze zur Optimierung von Arbeitsabläufen im Data-Warehouse-Projektmanagement vorgestellt.

Prozess der Informationsbereitstellung

In Unternehmen sind die einzelnen Fachbereiche organisatorisch getrennt und werden von einer zentralen IT-Abteilung im Sinne eines Dienstleisters versorgt. Im Idealfall formulieren die Fachbereiche ihre Anforderungen an die IT-Abteilung in Form von Projekten, die in einem gewissen Zeitrahmen umgesetzt werden. Sind die Datenbeschaffungs- bzw. Entwicklungszyklen der IT-Abteilung zu langsam, führt dies oft zu Eigenentwicklungen der Fachbereiche ohne Beteiligung der IT. Diese Eigenentwicklungen fördern das Entstehen sogenannter Daten-Silos, durch welche unnötige Redundanz bezüglich der Datenspeicherung und den Lade- und Extraktionsprozessen eingeführt wird.
Der gesamte Prozess der Informationsbereitstellung ist in Abbildung 3.2 dargestellt. Der Informationsbedarf der Fachabteilung muss im ersten Schritt in Form einer Datenanfrage spezifiziert werden. Die IT nimmt die Anfragen entgegen und prüft diese auf ihre strukturelle Plausibilität. Herrschen Unklarheiten, müssen diese in Abstimmung mit dem Fachbereich geklärt werden. Ist die Plausibilitätsprüfung abgeschlossen, wird die Verfügbarkeit der angeforderten Informationen überprüft. Das Nichtvorhandensein der Informationen kann mehrere Ursachen haben: Zum einen können abgeleitete Informationen fehlen, die entsprechend nachberechnet werden müssen. Zum anderen kann die Ursache in fehlenden Basisinformationen liegen, so dass zusätzliche Extraktoren und Ladeprozesse definiert werden müssen. In beiden Fällen ist die Integration der Datenbeschaffung in die bestehenden Datenproduktionsprozesse erforderlich. Stehen die Daten bereit, so können diese von den Fachbereichen auf Vollständigkeit und Übereinstimmung mit den Anforderungen hin überprüft werden. Besteht weiterer Informationsbedarf, so ist eine neue Datenanfrage zu formulieren.
Aus der Prozessbeschreibung wird ersichtlich, dass die Informationsbereitstellung mitunter erheblichen Komplikationen und daraus resultierenden Verzögerungen unterworfen ist. Im nächsten Abschnitt werden diese genauer beschrieben und Empfehlungen abgeleitet, welcher Bereich welche Kompetenzen innehaben sollte.

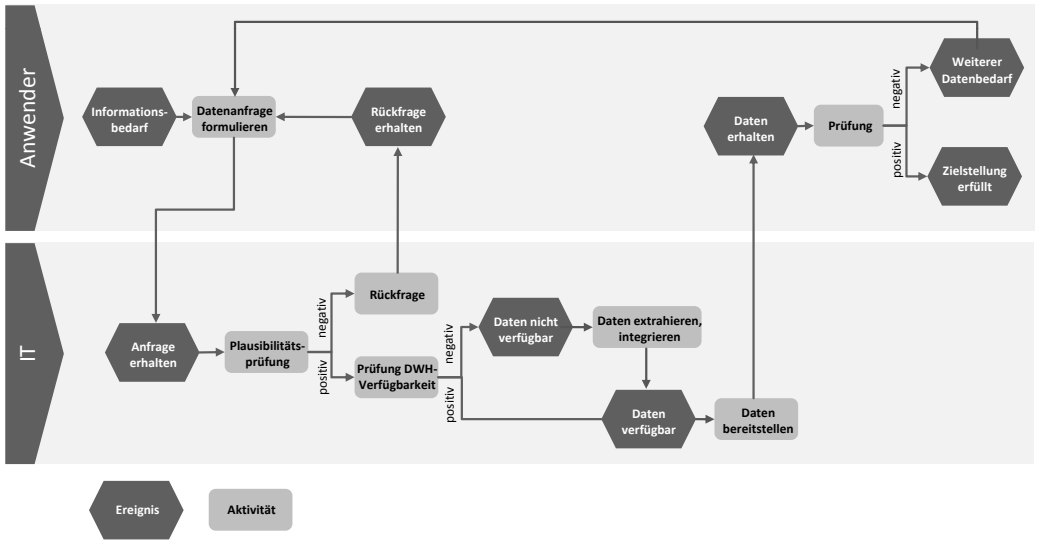


Abbildung 3.2: Prozess der Informationsbereitstellung nach [26]

Aufwandstreiber und Kompetenzverteilung

Die Interessen und Aufwände bei der Entwicklung neuer Data-Warehouse-Anwendungen können aus zwei Blickwinkeln betrachtet werden. Aus Sicht der Fachbereiche ist der Aufwandstreiber vor allem die sehr ausführliche Spezifikation der umzusetzenden Fachlogik, die für eine korrekte Umsetzung durch die IT notwendig ist. Sind die Spezifikationen unvollständig oder inkonsistent, so resultiert dies in zusätzlichen Iterationsschritten, wodurch wiederum ein Mehraufwand erzeugt wird. Weitere Kosten entstehen bei der Kontrolle der von der IT gelieferten Daten, um die fachliche Richtigkeit zu garantieren. Sind die Fachbereichsanforderungen häufigen Änderungen unterworfen, wie das bei situativen Analyseszenarien der Fall ist, multiplizieren sich die Spezifikations- und Kontrollkosten.
Die IT als Betreiber eines Data-Warehouses ist als Dienstleister gegenüber den Fachbereichen dazu verpflichtet, Daten und Dienste entsprechend definierter Dienstgütevereinbarungen (engl. service-level agreements) bereitzustellen. Änderungsanforderungen sind daher systematisch, anhand standardisierter Vorgehensmodelle umzusetzen. Insbesondere dürfen bereits vorhandene Daten nicht geändert und bestehende Datenproduktionsprozesse nicht über die Dienstgütevereinbarung hinaus verzögert werden. Des Weiteren ist die IT bestrebt, das Data-Warehouse kosteneffizient zu betreiben und dementsprechend automatisierte Prozesse einzusetzen. Der Entwicklungsaufwand zur Automatisierung von Datenproduktionsprozessen steht ebenfalls im Konflikt mit der situativen Datenanalyse.
Zur Umsetzung kurzfristiger Datenanforderungen ist die IT bzw. die Kommunikation zwischen IT und Fachbereichen zu starr organisiert. Stattdessen muss von Fall zu Fall abgewägt werden, in welcher Einheit welche Kompetenz anzusiedeln ist. Diese

Entscheidung kann unter Zuhilfenahme der folgenden vier Kriterien getroffen werden:

- Änderungshäufigkeit der Fachbereichsanforderungen
- Automatisierbarkeit der Prozesse bzw. Grad der Interaktion mit den Fachbereichen
- Allgemeingültigkeit der Daten und Prozesse (für eine Menge von Fachbereichen)
- Juristische Gewährleistung hinsichtlich des Betriebs und der Datenqualität.

In Abhängigkeit von der Gewichtung dieser Kriterien sind die Kompetenzen zur Datenextraktion, -integration und -verwaltung, zur Informationsgewinnung und zum Informationszugriff entsprechend auf die IT und die Fachbereiche zu verteilen.

Kompetenzzentren Neben der optimalen Verteilung der Verantwortlichkeiten auf IT und Fachbereiche, muss des Weiteren die Kommunikation zwischen den beiden Organisationseinheiten verbessert werden. Zu diesem Zweck wurde in [77] erstmals die Einrichtung sogenannter Kompetenzzentren (engl. BI competence center, kurz BICC) als Verbindungsglied zwischen der IT und den Fachbereichen vorgeschlagen. Diese sind durch Mitarbeiter sowohl der IT als auch aller Fachbereiche besetzt und formulieren die BI-Strategie einer Organisation. Dies umfasst insbesondere die proaktive Bereitstellung von Daten bzw. die Schaffung notwendiger Infrastrukturen. Das Kompetenzzentrum ist Ansprechpartner für die Fachbereiche bei der Informationsbeschaffung und wacht über die Einhaltung von Prozessstandards.
Der Prozess der Informationsbeschaffung wird durch die Einrichtung eines Kompetenzzentrums wesentlich beschleunigt, da die Kommunikation verbessert wird und der Informationsbedarf strategisch und vorausschauend bestimmt werden kann. Für die situative Datenanalyse ist das Kompetenzzentrum daher ein wichtiger Erfolgsfaktor.

3.2.2 Spreadmart-Lösungen

Ist der Prozess der Informationsbereitstellung aus Fachbereichssicht zu langwierig, um situative Datenanalysen zu unterstützen bzw. zu kostenintensiv, so resultiert dies in Eigenlösungen der Fachbereiche, die in der Literatur als Spreadmarts bezeichnet werden. Diese wurden in einer Studie des TDWI [22] aus dem Jahr 2008 wie folgt definiert:
„A spreadmart is a reporting or analysis system running on a desktop database (e.g., spreadsheet, Access database, or dashboard) that is created and maintained by an individual or group that performs all the tasks normally done by a data mart or data warehouse, such as extracting, transforming, and formatting data as well as defining metrics, submitting queries, and formatting and publishing reports to others. Also known as data shadow systems, human data warehouses, or IT shadow systems.“

Diese Spreadmart-Lösungen bergen jedoch hohe Risiken. Insbesondere kann die Qualität und Konsistenz der mit Spreadmarts erstellten Analysen nicht gewährleistet werden. Verschiedene Fachbereiche verwenden unterschiedliche Kalenderdefinitionen, Namenskonventionen und Berechnungen für ihre Datenanalysen, was zu inkonsistenten Sichten auf die Daten führt. Durch die Nichtbeachtung von IT-Standards sind die durch Spreadmarts bereitgestellten Daten weniger verlässlich. Diese Fehler potenzieren sich bei der Erzeugung von Spreadmarts durch Ableitung von bestehenden Spreadmarts. Des Weiteren liegt die Datenintegration und -aufbereitung nicht im Aufgabenbereich der Geschäftsanalysten, wodurch den Fachbereichen Arbeitszeit verloren geht und Zusatzkosten entstehen. In [22] wurde erhoben, dass Geschäftsanalysten 40% ihrer Zeit lediglich auf die Erstellung von Spreadmarts verwenden.
Trotz der genannten Risiken werden Spreadmarts in über 90% aller Organisationen eingesetzt [22]. Einer der Gründe, die Verzögerung bei der Informationsbeschaffung, wurde bereits in Abschnitt 3.2.1 genannt. Weitere Ursachen, die in der Studie des TDWI [22] als besonders relevant identifiziert werden konnten, sind der hohe Grad an Autonomie, die niedrigeren Kosten, das Bedürfnis Interessen zu schützen und das Fehlen geeigneter Analysewerkzeuge seitens der IT. Spreadmart-Lösungen haben somit ihre Daseinsberechtigung, sollten jedoch nicht zur Speicherung und Verwaltung zentraler Unternehmensdaten eingesetzt werden, sondern sich stattdessen auf die Datenanalyse und Entscheidungsunterstützung beschränken. Im Rahmen von Kompetenzzentren sind die bestehenden Spreadmarts zu sichten und anhand dieser die Bedürfnisse der Fachbereiche zu analysieren. Die so ermittelten Transformationen und Geschäftsregeln müssen anschließend in die Datenproduktion integriert werden.

Spreadmart-Werkzeuge

Einer der wesentlichen Gründe für die geringen Kosten der Spreadmart-Lösungen sind die Werkzeuge, die zu deren Erstellung verwendet werden. Die drei am häufigsten genutzten Werkzeuge sind nach [22] Microsoft Excel (mit 41%), Microsoft Access (mit 11%) und Microsoft Powerpoint (mit 9%), die auf den meisten Bürocomputern zu finden sind und daher keine Zusatzkosten verursachen. Die Mächtigkeit dieser drei Werkzeuge ist jedoch beschränkt. Für größere Datenvolumen existieren daher hauptspeicherbasierte Analysewerkzeuge, wie zum Beispiel IBM Cognos TM1 [99], QlikTech Qlikview [QlikView], Tableau Desktop [Tableau], Panoratio PANOSight [PANOsight], Comma Soft Infonea Cube [Infonea], HumanIT InfoZoom [InfoZoom] und PivotLink [PivotLink], mit denen mehrere Gigabyte große Datensätze auf herkömmlichen Desktoprechnern analysiert werden können. Besonderes Augenmerk ist auf das 2010 erscheinende Microsoft PowerPivot für Excel 2010 [PowerPivot] (Projektname „Gemini“) zu richten. PowerPivot integriert hauptspeicherbasierte Analysemethoden in Excel und ermöglicht es den Nutzern, in ihrer gewohnten Tabellenkalkulationsumgebung Daten zu integrieren, zu bereinigen, zu analysieren und Berichte zu erstellen. Unter Berücksichtigung der allgemeinen Verfügbarkeit von Excel in Unternehmen und des bereits verbreiteten Know-how zu Excel, besteht die Gefahr, dass die Anzahl der Spreadmart-Lösungen in Zukunft weiter steigen wird.

3.2.3 Analytische Mashups und dienstorientierte Architekturen

Der isolierte Betrieb von Spreadmart-Lösungen ist im Sinne einer unternehmensweiten BI-Strategie kontraproduktiv. Wichtige Analyseergebnisse und Erkenntnisse bleiben den Betreibern der Spreadmarts vorenthalten, wodurch zum einen wertvolles Wissen ungenutzt bleibt und zum anderen vielfach redundante Analysearbeit verrichtet wird. Kompetenzzentren können zum Teil Abhilfe schaffen, jedoch schränkt die Fülle der verschiedenen Spreadmart-Werkzeuge sowie die mangelnde Schnittstellenkompatibilität die Wiederverwendbarkeit von Informationen ein.
Der Aspekt der Verteilung von Daten und Mitarbeiterkompetenzen führt zu dem Ansatz, das Data-Warehouse-Konzept und dienstorientierte Architekturen (engl. service-oriented architecture, kurz SOA) miteinander zu verknüpfen. Die grundlegende Idee ist dabei, die Data-Warehouse-Daten als Dienste bereitzustellen, die durch die Fachbereiche frei kombinierbar sind, und die resultierenden Analyseergebnisse ebenfalls wieder als Dienste zur Verfügung zu stellen. Als Mittel zur Komposition von Diensten hat sich in den letzten Jahren das Mashup-Konzept durchgesetzt. Die Verwendung von Mashups und dienstorientierten Architekturen zur Entwicklung von BI-Lösungen ist Inhalt der folgenden Abschnitte. Insbesondere wird anhand einer Diskussion aktueller Enwicklungen und Trends der Mehrwert dieses Ansatzes gegenüber Spreadmart-basierten Lösungen herausgestellt.

Treiber der Entwicklung hin zu Mashup-basierten Lösungen

Mashups haben zum Ziel, bestehende Daten und Dienste aus verschiedensten Quellen miteinander zu verbinden und dadurch neue Inhalte zu generieren. Der Erfolg des Mashup-Ansatzes ist somit maßgeblich an die Verfügbarkeit von Daten und Diensten geknüpft. Dabei sind für die Mashup-Datenanalyse nicht nur unternehmensinterne Informationen relevant sondern vor allem auch externe Inhalte. An dieser Stelle wirken zwei Entwicklungen der letzten Jahre als Treiber hin zu Mashup-basierten Lösungen: Zum einen ist dies die zunehmende Öffnung von Datenbeständen durch Behörden und Unternehmen. Die US-Internetseite data.gov etwa stellt bereits heute Daten von über 100 Behörden frei zur Verfügung. Dies sind zum Beispiel demographische Informationen, Wetter- und Klimastatistiken, geologische Daten usw. Der Trend hin zur Öffnung von Datenbeständen wird durch Regierungsinitiativen (Open Government) wie zum Beispiel E-Government 2.0 [101] in Deutschland sowie auf europäischer Ebene die Initiative i2010 [i2010] bewusst vorangetrieben. Auf Seiten der Unternehmen gibt es ebenfalls das Bestreben, Inhalte öffentlich zugänglich zu machen, um in Kombination mit anderen Diensten einen Mehrwert zu generieren. Hier sei nur die von Google entwickelte OpenSocial-API [OpenSocial] genannt, welche Programmierschnittstellen für Funktionen und Inhalte sozialer Netzwerke zur Verfügung stellt.
Die andere Entwicklung sind die im Zuge von Web 2.0 entstandenen Webdienste und Mashups selbst, die beste Voraussetzungen zur computergestützten Entscheidungsunterstützung bieten. Die Umsetzung komplexer Berechnungen, Visualisierun-

gen oder semantischer Analysen von Dokumenten muss nicht mehr selbst implementiert, sondern kann durch einfache Dienstaufrufe in die Datenanalyse integriert werden. Das auf der Software Mathematica basierende WolframAlpha [WolframAlpha] etwa bietet Dienste zur Berechnung mathematischer Ausdrücke und zur Darstellung von Informationen. Der von Thomsen Reuters entwickelte Dienst OpenCalais [OpenCalais] nimmt unstrukturierte Dokumente entgegen und extrahiert Entitäten (Personen, Organisationen, Produkte, usw.), Fakten und Ereignisse und fügt diese in Form von Metadaten den Dokumenten an. Zusätzlich werden durch Zurückgreifen auf eine Wissensbasis Assoziationen zu anderen Dokumenten hergestellt. Dadurch können Dokumente in eine strukturierte Form überführt und für weitere Datenanalysen verwendet werden. Zur Visualisierung von Analyseergebnissen stehen zahlreiche Dienste bereit, wie zum Beispiel IBM Manyeyes [ManyEyes], Swivel [Swivel], Track-n-Graph [TracknGraph], iCharts [iCharts] oder Mycrocosm [Mycrocosm].

Studien In einem Treffen von Analysten der Marktforschungsgruppe Gartner im Januar 2009 [58] wurden folgende Feststellungen bzw. Vorhersagen getroffen:

- Bis 2010 werden 20% aller Organisationen industriespezifische Analyseanwendungen via „Software as a Service“ (SAAS) standardmäßig in ihr BI-Portfolio integriert haben.

- 2009 wird sich die kollaborative Entscheidungsfindung als neue Produktkategorie etablieren und soziale Software mit BI-Plattformfunktionalität kombinieren.

- Bis 2012 werden ein Drittel aller auf Geschäftsprozessen angewendeten Analyseanwendungen in Form von Mashups publiziert.

Dies unterstreicht nochmals die Bedeutung von Mashup-Technologien im Data-Warehouse- und BI-Umfeld und verweist auf eine weitere Facette der BI-Anwendungen: Während bisher Datenanalyseapplikationen von einzelnen Analysten entwickelt und betrieben wurden (siehe Abschnitt 3.2.2), wird dies in Zukunft kollaborativ geschehen. Dienstorientierte Architekturen und Mashups bieten dafür gute Voraussetzungen.

Analytische Mashups

Mashups repräsentieren eine Gattung interaktiver Webanwendungen, die bestehende Inhalte und Dienste kombinieren, um neue und innovative Dienste anzubieten. Ein typisches Beispiel ist die Verknüpfung von Nachrichtenmeldungen mit Google Maps, um dem Rezipienten zusätzliche Informationen zum Ort des Geschehens liefern zu können. Dieser Ansatz kann weiterentwickelt und in der Form sogenannter *Analytical Mashups* zur Datenintegration und -analyse verwendet werden.

Abgrenzung zu ETL Die zur klassischen Datenintegration verwendeten ETL-Prozesse (Extraktion, Transformation, Laden) werden jedoch durch analytische Mashups nicht abgelöst. Während ETL die skalierbare Integration von Massendaten für langlebige Anwendungen adressiert, fokussiert der Ansatz der analytischen Mashups die situative Datenanalyse. Aufgrund der eingeschränkten Datenmenge sowie in Anbetracht der zeitlichen Flüchtigkeit situativer Analysen, spielen nichtfunktionale Aspekte wie die Leistung und Skalierbarkeit für analytische Mashups nur eine untergeordnete Rolle. Auch sind die Anforderungen an die Qualität der Mashup-Daten im Allgemeinen geringer.
Das dreistufige ETL-Vorgehensmodell, bestehend aus der Extraktion, der Transformation und dem Laden der Daten, ist grundsätzlich auf den Mashup-Ansatz übertragbar. Im Kontext von Mashups werden diese Stufen mit Fetch, Shape und Pick bezeichnet. Der Fetch-Schritt beschreibt das Abgreifen der Daten unter Verwendung interner oder externer Dienste, wofür im Vergleich zum ETL keine komplexe Extraktionslogik nötig ist. Der Begriff Shape beschreibt eine Abschwächung des Transformationsprozesses, da aufwendige Bereinigungsschritte nicht notwendig sind und die Transformation in ein Data-Warehouse-Schema entfällt. Es sind keine komplexen Ladeprozeduren notwendig, sondern der Nutzer kann direkt die Daten verwenden (Pick), statt sie in Zieldatenbanken abzulegen. Zusammenfassend lässt sich sagen, dass ETL der IT-getriebene Ansatz zur Datenintegration ist, wohingegen analytische Mashups fachbereichsgetrieben sind.

Mashup-Beschreibungssprache EMML Die Open Mashup Alliance [OMA] (OMA) ist ein Konsortium bestehend aus Technologieunternehmen wie Adobe Systems, Hewlett-Packard und Intel sowie Technologieanwendern wie der Bank of America und Capgemini, mit der Zielsetzung, die Interoperabilität von Mashups zu verbessern und im Zuge dessen eine Mashupbeschreibungssprache zu entwickeln. Das Ergebnis ist die Enterprise Mashup Markup Language, kurz EMML, eine XML-basierte Beschreibungssprache zur Verarbeitung und Kombination von Datenquellen verschiedenster Formate wie XML, JSON oder JDBC. EMML unterstützt eine Reihe von Datenoperationen wie das Filtern, Sortieren, Verbundoperationen heterogener Dienste und Datenformate, Gruppieren und Aggregrieren sowie die Annotation um zusätzliche Semantik. Die Ausführungslogik kann durch Verwendung von Bedingungen, Schleifen und parallelen Aufrufen strukturiert werden. Zur Erweiterung der Logik können Skripte verschiedener Sprachen wie JavaScript, JRuby, Groovy oder XQuery in EMML eingebettet werden. Des Weiteren unterstützt EMML das Parsen von HTML-Seiten, um so neue Datenquellen zu erschließen. Eine EMML-Referenzimplementierung steht unter [OMA] zur Verfügung.

Mashup-Entwicklungsplattform Die EMML bietet gute Voraussetzungen für eine Mashup-Entwicklungsplattform. Diese muss zur Unterstützung situativer Datenanalysen einer Reihe von Anforderungen genügen, von denen nur die wichtigsten genannt werden:

- Suche von Datenquellen und Diensten: vorhandene Daten, sogenannte Mashables, und Dienste müssen zentral registriert werden, um so die Wiederverwendbarkeit zu erhöhen und Redundanz zu vermeiden

- Kollaborative Entwicklung: Nutzern muss es möglich sein, Mashups individuellen Bedürfnissen anzupassen und weiterzuentwickeln; das gemeinsame Arbeiten mehrerer Nutzer muss unterstützt werden (Mehrbenutzerbetrieb, Versionierung)

- Optimierte Verarbeitung: die Mashupausführung, unter dem Aspekt der Datenverteilung und der verteilten Ausführung, ist zu optimieren; dies schließt insbesondere eine effiziente Abbildung auf die IT-Infrastruktur ein.

Die Rolle von SOA in der Mashup-Entwicklung Der Begriff SOA beschreibt ein Konzept zur Ausrichtung der IT-Landschaft eines Unternehmens an seinen Geschäftsprozessen. Dies wird durch die Bereitstellung lose gekoppelter, atomarer Dienste erreicht, die flexibel miteinander kombinierbar sind. Aus technischer Sicht kann SOA mittels jeder beliebigen dienstbasierten Architektur umgesetzt werden. In der Praxis haben sich jedoch die drei Standards SOAP, WSDL und UDDI durchgesetzt: hierbei dient SOAP als Kommunikationsprotokoll zwischen Diensten, WSDL ist eine Sprache zur Schnittstellenbeschreibung der Dienste und UDDI ist der Verzeichnisdienst zur Registrierung und Lokalisierung von Diensten.
Übertragen auf die Vision einer Mashup-Entwicklungsplattform sind die Daten eines Data-Warehouses als Datendienst bereitzustellen und können dann von Anwendern dieser Plattform für die Datenanalyse verwendet werden. Diese Art der Nutzung unterscheidet sich signifikant von der klassischen Nutzungsweise der Data-Warehouse-Daten. Die Mehrheit der Nutzer eines Data-Warehouses bekommt lediglich vorberechnete Berichte in verschiedenen Dateiformaten bereitgestellt. Nur einer Minderheit ist der direkte Zugriff auf das Data-Warehouse bzw. die Data-Marts gestattet. Die Datennutzung in Form von Diensten durch eine große und weitestgehend unbekannte Anwendergruppe führt zu den folgenden Problemen:

- Optimierung: Durch das willkürliche und nicht vorhersagbare Anfrageverhalten ist eine zielgerichtete Optimierung des Data-Warehouses kaum möglich.

- Fehlerhafte Datennutzung: Im Gegensatz zum klassischen Datenbanknutzer, der mit dem Datenschema und den Daten selbst vertraut ist, können externe Nutzer das Datenmodell falsch interpretieren und die Daten falsch oder inadäquat nutzen.

- Wartbarkeit: Änderungen am Datenbankschema können unerwartete Auswirkungen auf darüberliegende Anwendungen haben.

- Sicherheit: Bei der Verwendung von Datendiensten ist eine Autorisierung erforderlich.

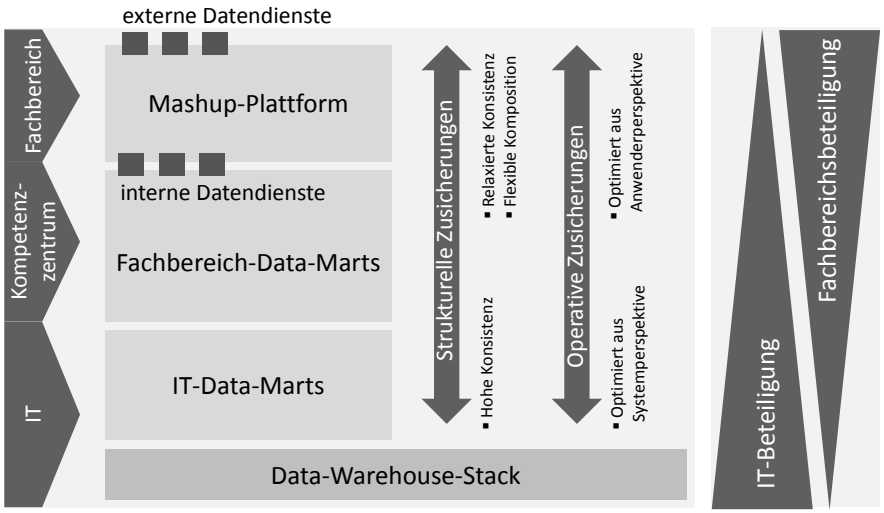


Abbildung 3.3: Integration der Mashup-Plattform in eine DWH-Architektur

Eine pragmatische Lösung, die diese Probleme adressiert, jedoch mit dem Gedanken der situativen Datenanalyse in Konflikt steht, besteht darin, nur vordefinierte Anfragen als Dienste zu kapseln und bereitzustellen. Der Aspekt der Wartbarkeit stellt für kurzlebige Analysen kein Problem dar. Nur wenn gegen die Philosophie der situativen Datenanalyse verstoßen wird und Mashups für langfristige Analyseprojekte verwendet werden, ist deren Wartung zu berücksichtigen.

Organisatorische Einordnung einer Mashup-Plattform Eine Data-Warehouse-Architektur, die eine Mashup-Plattform integriert sowie eine organisatorische Trennung in IT, Fachbereiche und Kompetenzzentrum vollzieht, ist in Abbildung 3.3 skizziert. Die Verantwortung der IT liegt in der technischen Bereitstellung der gesamten Infrastruktur des Data-Warehouses und der Mashup-Plattform. Hinsichtlich der Datenbereitstellung zeichnet die IT für das Data-Warehouse sowie die weitestgehend anwendungsneutralen IT-Data-Marts zuständig. Die anwendungsspezifischen Fachbereich-Data-Marts werden durch ein Kompetenzzentrum, bestehend aus IT- und Fachbereichsmitarbeitern (siehe Abschnitt 3.2.1), definiert. Die Fachbereiche werden für strategische Analysen weiterhin mit vordefinierten Berichten oder Cubes versorgt (nicht dargestellt in Abbildung 3.3). Besteht in den Fachbereichen der Bedarf nach situativen Datenanalysen, wird dies mittels der Mashup-Entwicklungsplattform umgesetzt. Die dazu verwendeten internen Datendienste werden von der IT und dem Kompetenzzentrum bereitgestellt. Die IT als Betreiber der Mashup-Plattform kann unter Absprache mit dem Kompetenzzentrum besonders häufig verwandte Mashups oder Mashup-Teile in den regulären Datenproduktionsprozess überführen.
Die Nutzer der Mashup-Plattform müssen sich darüber bewusst sein, dass die Zusicherung zur Datenkonsistenz (strukturelle Zusicherungen) und zum operativen Be-

trieb (operative Zusicherungen in Abbildung 3.3) mit zunehmendem Einfluss der Fachbereiche und abnehmendem Einfluss der IT sinkt.

3.2.4 Werkzeuge und Methoden im Kostenvergleich

Die in diesem Abschnitt beschriebenen Konzepte und Lösungen zur Umsetzung situativer Datenanalysen werden abschließend in einem Kostenvergleich evaluiert. Zu diesem Zweck sind in Abbildung 3.4 die Kosten der Umsetzung einer Datenanalyseanwendung in Abhängigkeit von der Dynamik der Anforderungen skizziert. In einer statischen Umgebung sind die Entwicklungskosten für eine IT stets geringer als für die Fachbereiche ohne die entsprechenden Kompetenzen. Dieser Vorteil schwindet jedoch mit zunehmender Dynamik der Anforderungen, bis schießlich ein Punkt erreicht wird, an dem der Fachbereich selbst die Anwendung günstiger entwickeln kann. Es ist zu beachten, dass dieser Punkt aus subjektiver Sicht des Fachbereichs schon früher erreicht werden kann.
Neben dem kosteneffizienten Betrieb ist es das Ziel einer IT-Abteilung, ihre Dienste wesentlich günstiger anzubieten als es den Fachbereichen möglich wäre, d.h. den Schnittpunkt in Abbildung 3.4 zu senken. Dies ist seitens der IT zum einen durch die Verschärfung von Prozessstandards zu erreichen, die bei Eigenentwicklungen berücksichtigt werden müssen. Ein Beispiel dafür ist die Verpflichtung zur Abschaltung von Spreadmarts. Höhere IT-Standards sind jedoch aus gesamtunternehmerischer Sicht nicht das beste Mittel. Stattdessen muss die IT bei der Umsetzung von Analysenanwendungen auch unter schnell veränderlichen Anforderungen kostengünstig operieren. Die Organisationsform der Kompetenzzentren zur Verbesserung der Kommunikation zwischen Fachbereichen und IT (siehe Abschnitt 3.2.1) bildet hierzu einen Beitrag. Auch die Verkürzung und Optimierung von Entwicklungsprozessen durch Anwendung der Methoden der agilen Softwareentwicklung, wie zum Beispiel das Scrum-Vorgehensmodell, können die IT-Kosten senken. Die Kosten auf Seiten der Fachbereiche werden jedoch durch zunehmende Verbesserung der Werkzeugunterstützung (siehe Spreadmart-Werkzeuge in Abschnitt 3.2.2) ebenfalls gesenkt und stehen somit in Konkurrenz zu den IT-Entwicklungen. Die Umsetzung von Datenanalyseprojekten durch die Anwendung von Mashup-Technologien ist hinsichtlich der Kosten neutral zu bewerten, da sie sowohl auf IT- als auch Fachbereichsseite zur Optimierung von Abläufen beiträgt.

3.3 Evolution der Data-Warehouse-Systeme

Die in Abschnitt 3.1 erarbeitete Definition des klassischen Data-Warehouse-Begriffs hat trotz den Entwicklungen in diesem Bereich nach wie vor ihre Gültigkeit. Jedoch erfordert ein zunehmend breiteres Spektrum an Data-Warehouse-Anwendungen und Nutzern eine Erweiterung des Begriffs um zusätzliche Facetten. Das Ziel dieses Kapitels besteht darin, diese Veränderungen zu motivieren und zu beschreiben, um in Abschnitt 3.5 die Anforderungen an ein Data-Warehouse-System um den Aspekt der Echtzeit zu erweitern.

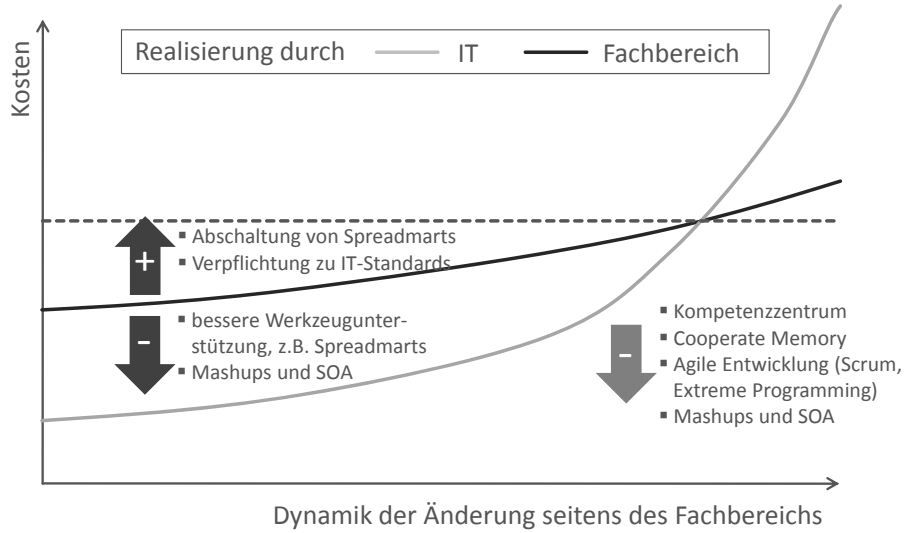


Abbildung 3.4: Kosten der Realisierung durch IT und Fachbereich in Abhängigkeit zur Änderungsdynamik

3.3.1 Nutzung von Data-Warehouse-Systemen

Das Konzept der Data-Warehouse-Systeme ist für eine Vielzahl von Einsatzbereichen nutz- und anwendbar. Initial jedoch werden Data-Warehouse-Projekte für sehr konkrete Anwendungsbereiche geplant und umgesetzt. Erst mit zunehmender Nutzungsdauer und wachsender Nutzerakzeptanz wächst die Zahl der Anwendungen, wodurch die Anforderungen an das System und die Data-Warehouse-Architekten zunehmen. Die im Folgenden skizzierte Evolution der Data-Warehouse-Anwendungen ist sowohl im Kleinen, innerhalb spezifischer Data-Warehouse-Projekte, als auch im Großen, als generelle historische Entwicklung der letzten Jahre, beobachtbar.

Art der Anwendungen

Zunächst wird der vierstufige Entwicklungsverlauf der Data-Warehouse-Anwendungen beschrieben (siehe Abbildung 3.5). In der ersten Stufe einer Data-Warehouse-Instal- lation werden diese vorwiegend für die Berichtsproduktion verwendet. Sie werden einmalig definiert und dann in bestimmten Berichtszyklen neu berechnet. Charakteristisch für diese Art der Anwendung sind die bereits a priori bekannten Anfragen sowie die festgelegten Zeitpunkte, zu denen die Berichte erzeugt werden müssen. Aufgrund des starren Verarbeitungsmusters wird dieses Vorgehen auch als Batch-Verarbeitung bezeichnet. Für diese Art der Anwendung lassen Datenbanken sich sehr spezifisch optimieren, z.B. durch Materialisierung von Summendaten.
In der nächsten Anwendungsstufe kommen zusätzliche Fragestellungen hinzu. Neben der reinen Repräsentation von Fakten wird darüber hinaus die Analyse der Ursachen untersucht. Ausgehend von grobgranularen Fakten kann der Nutzer, z.B. durch

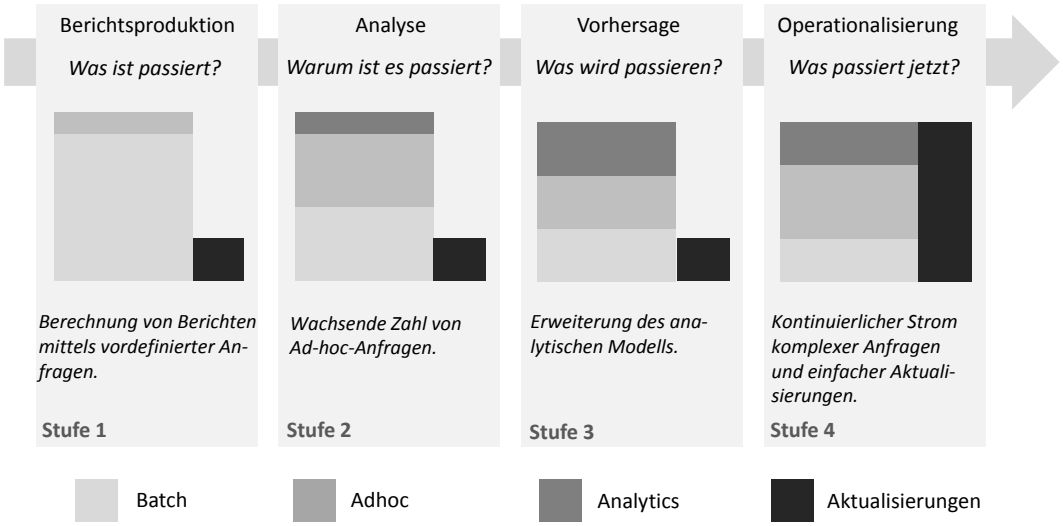


Abbildung 3.5: Die vier Stufen der Nutzung eines Data-Warehouse-Systems (nach [12])

Drill-down-Navigation, die Informationen immer weiter verfeinern. Die Anfragemuster sind somit nicht mehr statisch, wie in Stufe 1, sondern durch die notwendige Interaktivität sehr dynamisch und nicht vorhersagbar. Diese interaktiven Nutzeranfragen werden als Ad-hoc-Anfragen bezeichnet. Der Zeitpunkt, zu dem diese Anfragen an das System gestellt werden, ist nicht vorhersagbar, was bei hoher Nebenläufigkeit zu Performanzengpässen führen kann.
Nachdem Unternehmen das „was“ und „warum“ von Abläufen in der Vergangenheit beantworten können, ist in der nächsten Anwendungsstufe die Vorhersage von zukünftigen Entwicklungen von Interesse. Um Vorhersagemodelle realisieren zu können, ist aufbauend auf dem Data-Warehouse-System der Einsatz von Data-Mining-Software notwendig. Durch die Komplexität der Data-Mining-Software ist deren Anwenderkreis sehr beschränkt. Dessen ungeachtet ist der Ressourcenverbrauch dieser Anwendung, aufgrund der Datenintensivität der Vorhersagealgorithmen, sehr hoch. Des Weiteren wird die Parametrierung der Vorhersagemodelle zunehmend vereinfacht, bis hin zu parameterfreien Algorithmen, so dass sich der Anwenderkreis noch vergrößern wird.
Die bisherigen Anwendungen haben vor allem die langfristige bzw. strategische Entscheidungsfindung unterstützt. Das Data-Warehouse-System wurde dazu turnusmäßig, auf Wochen- oder Monatsbasis, beladen. Zur Unterstützung der täglichen Geschäftsprozesse und -vorgänge, der sogenannten operativen Entscheidungsfindung, ist die dadurch erreichte Datenaktualität nicht ausreichend. Im Idealfall werden Änderungen sofort propagiert, so dass ein stetiger Strom von Aktualisierungen, zusätzlich zu dem heterogenen Anfrageworkload, das System belastet. Eine weitere Steigerung der operativen Entscheidungsfindung bilden die *Active Data Warehou-*

ses. Diese automatisieren den Entscheidungsprozess, so dass die Latenzzeit, die durch menschliche Interaktion entsteht, entfällt.
Die vorgestellten Anwendungsbereiche ersetzen einander nicht, sondern werden in Koexistenz zueinander betrieben. So besteht der Workload in Anwendungsstufe 4 weiterhin zu einem signifikanten Anteil aus Batch-Anfragen zur Berichtsproduktion sowie aus analytischen Anfragen. Aufgrund der Verschiedenheit der Anwendungen und der Heterogenität der Nutzergruppen sieht sich ein Data-Warehouse-System somit einer Reihe von Dienstgüteanforderungen (engl. service level) gegenüber.

Klassifizierung der Entscheidungsebenen

Die skizzierten Anwendungen unterstützen Geschäftsprozesse innerhalb verschiedener Entscheidungsebenen. Diese lassen sich in eine strategische, eine taktische und eine operative Ebene klassifizieren, deren Charakteristika im Folgenden beschrieben werden (siehe Abbildung 3.6).
Primär unterscheiden sich die drei Ebenen in dem Zeithorizont, über den die Entscheidung wirkt, sowie in der Art bzw. dem Bezug der Entscheidung. Strategische Entscheidungen beziehen sich auf die Definition und das Erreichen von Zielen und sind somit sehr langfristig orientiert. Im Unterschied dazu beziehen sich operative Entscheidungen auf einzelne Prozesse und sind nur innerhalb eines kurzen Zeitraumes von Relevanz. Dies sei an einem Beispiel dargestellt: die Entscheidung über die Vergabe eines Kredits mit einer Laufzeit von 20 Jahren fällt sehr langfristig aus, stellt jedoch für die betreffende Bank lediglich eine operative Entscheidung dar. Die Betrachtung des Zeithorizonts ist somit nicht ausreichend zur Differenzierung der Entscheidungsebenen. Durch Hinzunahme der Art der Entscheidung bzw. des Wirkungsbereiches wird die Trennung jedoch klarer. Die Vergabe eines Kredits hat keinen Einfluss auf die Unternehmenspolitik bzw. deren Zielsetzung, sondern wirkt lediglich auf den Einzelfall im operativen Geschäft. Im Gegensatz dazu ist die Festlegung der Produkt- bzw. Distributionspolitik eines Unternehmens sowohl vom Zeithorizont als auch vom Wirkungsbereich her eine strategische Entscheidung. Zwischen diesen beiden Ebenen befindet sich die taktische Entscheidungsebene, die sich vor allem mit der Kontrolle und Umsetzung von Unternehmenszielen befasst und somit mittelfristig wirkt. Darunter fällt zum Beispiel die Kontrolle der Preisentwicklung und die Anpassung der Preisgestaltung, um etwa eine bestimme Produktpolitik erfolgreich umzusetzen.
Die verschiedenen Entscheidungsebenen bedürfen unterschiedlicher Daten, die ebenfalls in Abbildung 3.6 schematisiert sind. Die den Entscheidungen zugrunde liegenden Daten werden nach ihrem zeitlichen Bezug bzw. der Datenlatenz und der Art der Datenquellen unterschieden. So benötigen strategische Entscheidungen historische bzw. zeitraumbezogene Daten, wohingegen operative Entscheidungen sehr aktuelle und zeitpunktbezogene Daten erfordern. Die Datenlatenz, d.h. die Verzögerung zwischen Eintreffen des Ereignisses bis hin zur Informationsextraktion, muss somit für die operative Entscheidungsfindung sehr gering sein. Demgegenüber sind die Anforderungen an die Datenaktualität für strategische Entscheidungen wesent-

	Zeithorizont	Art der Entscheidung	Entscheidungsebene	Datenlatenz	Datenquellen
strategisch	• langfristig	• Unternehmenspolitik	• breit • Unternehmen	• hoch, historische Daten • zeitraumbezogen	• viele unterschiedliche Quellen • auch unstrukturiert • aggregiert
taktisch	• mittelfristig	• Kontrolle und Umsetzung von Unternehmenszielen	• relativ breit • Unternehmen, Abteilungen	• eher hoch • historisch und zeitpunktbezogen	• viele Quellen • strukturiert und semistrukturiert • schwach aggregiert
operativ	• kurzfristig	• Preisbildung • Vertragskonditionen • teilw. automatisierbar	• fokussiert • einzelne Prozesse	• niedrig, operative Daten • zeitpunktbezogen	• wenige Quellen • hoher Detailgrad • strukturiert

Abbildung 3.6: Vergleich strategische, taktische und operative Entscheidungsfindung

lich niedriger. Bedingt durch den organisationsweiten Wirkungsbereich strategischer Entscheidungen ist die Zahl der zu betrachtenden Datenquellen deutlich höher als für operative Entscheidungen. Die Datenquellen der strategischen Ebene schließen auch unstrukturierte Daten, zum Beispiel Textdokumente, mit ein. Im Gegensatz dazu liegen die Daten auf operativer Ebene in strukturierter Form vor, wodurch sich Entscheidungsprozesse einfach automatisieren lassen. Durch die hohe Menge an Informationen, die zur strategischen Entscheidungsfindung notwendig sind, werden diese in der Regel in aggregierter Form verarbeitet. Auf operativer Entscheidungsebene sind hingegen Detaildaten unerlässlich.
Die drei vorgestellten Entscheidungsebenen stellen unterschiedliche Bedürfnisse an das zugrunde liegende Data-Warehouse-System. In dieser Arbeit sind vor allem der Zeithorizont der Entscheidungsebenen und die daraus folgenden verschiedenen Anforderungen an die Datenlatenz von Interesse.

Veränderung der ETL-Prozesse

Das erweiterte Anwendungsspektrum von Data-Warehouses sowie deren Einbindung auf verschiedenen Entscheidungsebenen zieht auch Veränderungen bei den Prozessen zur Datenversorgung nach sich. In einer Untersuchung des TDWI aus dem Jahr 2005 [88] wurden 672 IT-Fachleute zur Verwendung von ETL, EAI und EII-Werkzeugen befragt. Bestandteil dieser Untersuchung war unter anderem eine Umfrage zum Einsatz von ETL in den Unternehmen. Der ETL-Prozess wurde dazu in drei Kategorien unterschieden: klassische Batch-ETL-Verarbeitung, Change-Data-Capture-ETL (CDC-ETL) und Online-ETL, auch als Real-Time-ETL oder Trickle-Feed-ETL bezeichnet (siehe Abbildung 3.7). Letzteres beschreibt eine push-basierte Propagierung von Änderungen in das Data-Warehouse-System, wohingegen CDC-ETL lediglich die Verwendung von Techniken zur Feststellung von Änderungen meint (zeitstempel-, triggerbasierte Verfahren usw.). Von den Befragten gaben 57 Prozent den Grad der Nutzung von Batch-ETL in ihren Unternehmen als hoch an. Bei der Frage nach

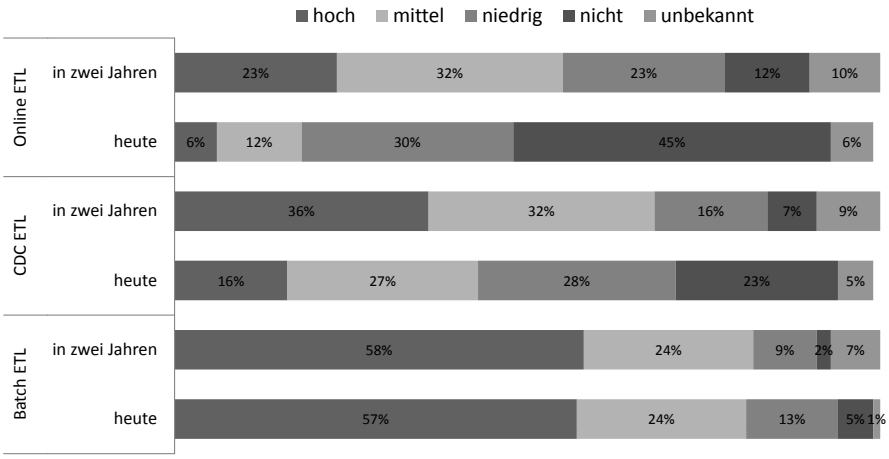


Abbildung 3.7: Die Entwicklung des ETL-Marktes

dem Maß der Nutzung in der Kategorie „hoch“ in zwei Jahren stieg dieser Wert auf lediglich 58 Prozent an. Dies zeigt zum einen die hohe Relevanz von Batch-ETL, zum anderen jedoch auch eine Sättigung dieser Verarbeitungssemantik in den Unternehmen.
Ein anderes Bild ergibt sich für die Kategorien Change-Data-Capture-ETL und Online-ETL: das Maß der hohen Nutzung wurde im Jahr 2005 mit jeweils 16 bzw. 6 Prozent angegeben. Bei der Abschätzung für die Zeit in zwei Jahren stiegen diese Werte auf 36 bzw. 23 Prozent. Die Kombination der Kategorien „mittel“ und „hoch“ für das Online-ETL in der Zeit in zwei Jahren ergibt einen Wert von 55 Prozent. Dies zeigt den klaren Trend hin zu Online-ETL und den damit einhergehenden wachsenden Bedarf nach Echtzeitanalysen in den Unternehmen.

Data-Warehouse-Workload

Die letzten drei Abschnitte haben gezeigt, dass sowohl hinsichtlich der Aktualisierungs- als auch der Anfrageworkloads von Data-Warehouse-Systemen große Veränderungen stattgefunden haben. Immer stärker werdende Datenaktualitätsanforderungen machen eine kontinuierliche Aktualisierung des Data-Warehouses, durch sogenanntes Online-ETL oder Real-Time-ETL, erforderlich. Die klassischen Batch-orientierten Ladeprozesse werden dadurch jedoch nicht verdrängt, sondern haben nach wie vor ihre Existenzberechtigung, da nicht für alle Daten eines Data-Warehouses geringe Latenzzeiten erforderlich sind. Auf Seiten der Anfragen existieren unterschiedliche Anwendungen, die in verschiedene Entscheidungsebenen klassifiziert werden können und somit vielfältige Anforderungen an die Datenaktualität stellen. Eine Gartner-Studie aus dem Jahr 2008 [24] belegt diese Veränderungen noch einmal. In dieser

Untersuchung wird die Heterogenität heutiger Data-Warehouse-Workloads als größte Herausforderung der nächsten drei Jahre für die Datenbankhersteller beschrieben.

Größe der in Data-Warehouses gespeicherten Datenmenge

Neben der steigenden Anzahl der Nutzer und Anfragen sowie der höher werdenden Anfragekomplexität, steigt auch die Menge der in Data-Warehouses gespeicherten Daten. In einer Studie von Richard Winter [90] wird der jährliche Wachstumsfaktor der Data-Warehouse-Datenvolumen mit 1,5 bis 2,5 beziffert. Zum Beispiel hat sich das Datenvolumen des Data-Warehouses von LGR Telecommunications, eine der zehn größten Installationen weltweit, innerhalb von vier Jahren verzehnfacht [89]. Dieses umfasst nunmehr 310 Terabyte und wird jeden Tag um zirka 13 Milliarden Tupel erweitert.
Dieses Wachstum hat mehrere Ursachen: Zum einen sind dies die stark sinkenden Preise bei der Anschaffung von Plattenspeichern, die es Unternehmen mittlerweile ermöglichen, nahezu alles abzuspeichern. So halbieren sich die Preise für Plattenspeicher aller 1,5 Jahre, bei gleichzeitiger Verdoppelung der Festplattengröße aller zwei Jahre. Zum anderen liegt das Datenwachstum in den Data-Warehouse-Anwendungen begründet: Komplexer werdende Analysen benötigen detaillierte Daten, um zuverlässige Aussagen treffen zu können. So ist es für die Bewertung der Kundenprofitabilität notwendig, das gesamte Kundenverhalten zu analysieren, d.h. alle Verkaufstransaktionen, Lieferkosten, Kosten aus Regressansprüchen usw. Die Forderung nach umfassenderen Analysen führt auch zu einem längeren Vorhalten von Daten – mittlerweile durchschnittlich sieben Jahre lang [89]. Dieser Trend wird nochmals verstärkt durch neue Regulierungsvorschriften, wie zum Beispiel Basel II im Bankwesen oder Sarbanes Oxley für börsennotierte Unternehmen, die die Datenvorhaltung für einen bestimmten Zeitraum vorschreiben. Des Weiteren werden neue Arten von Daten verfügbar, die ebenfalls gespeichert und für Analysen aufbereitet werden müssen, wie zum Beispiel GPS-Daten, E-Mail-Kundenkontakte oder Call-Center-Audiomitschnitte.

3.3.2 Entwicklungsprozess der Hardware- und DBMS-Architekturen

In den vorangegangenen Abschnitten wurden die Auswirkungen der zunehmenden Anwendungskomplexität auf den Data-Warehouse-Workload sowie das Datenvolumen dargestellt. Zur Bewältigung dieser Anforderungen müssen sich auch die den Data-Warehouse-Systemen zugrundeliegenden Datenbanksysteme anpassen bzw. weiterentwickeln. Dies gilt zum einen für die verwendeten Hardware-Architekturen als auch für die Architektur der Datenbanksysteme. Für beide sind im Folgenden die aktuellen Entwicklungen und ihre Auswirkungen auf die Architektur der Data-Warehouse-Systeme dargestellt.

Data-Warehouse-Hardware-Architektur

Aus den zunehmenden Anforderungen an Systeme zur Datenanalyse sind in den letzten Jahren eine Reihe von Architekturansätzen entstanden. Die wesentlichen Techniken – Parallelisierung, Einsatz spezieller Hardware und die massive Nutzung vom Hauptspeicher – sowie einige Vertreter werden im Weiteren vorgestellt.

Parallelisierung Ein wichtiger Ansatz, um trotz steigender Anzahl von Anfragen und größer werdenden Datenvolumen gute Antwortzeiten zu erreichen, ist die Parallelisierung von DBMS-Architekturen. Hier werden die folgenden drei Ansätze unterschieden: Shared-Memory, Shared-Disk und Shared-Nothing. Der einfachste aber zugleich auch am wenigsten leistungsfähige Ansatz ist die Shared-Memory-Architektur (z.B. Microsoft SQL Server [MSSQL]): Alle Prozessoren teilen sich gemeinsamen Haupt- und Plattenspeicher, wodurch die Implementierung dieser Systeme vereinfacht wird, da auf verteilte Sperrprotokolle verzichtet werden kann, was aber gleichzeitig auch die größte Beschränkung darstellt. Da sich alle Prozessoren den gleichen Bus für E/A-Operationen und Speicherzugriffe teilen müssen, skalieren diese Systeme nur sehr schlecht. Ähnlichen Beschränkungen unterliegt die Shared-Disk-Architektur. Voneinander unabhängige Verarbeitungsknoten, mit jeweils eigenem Hauptspeicher, greifen gemeinsam auf einen Plattenspeicher zu. Da gemeinsame Hauptspeicherbereiche fehlen, sind verteilte Sperrmechanismen notwendig, die jedoch bei steigender Anzahl von Verarbeitungsknoten zum Engpass für das gesamte System werden. Oracle RAC [RAC] ist ein Beispiel für diese Architektur. Die skalierbarste dieser drei Architekturen ist die Shared-Nothing-Architektur, auch als Massive-Parallel-Processing- oder kurz MPP-Architektur bezeichnet. In dieser besitzen alle Verarbeitungsknoten, die über ein LAN verbunden sind, ihren eigenen Festplattenspeicher. Die Datentabellen werden horizontal partitioniert und auf die Verarbeitungsknoten verteilt. Pufferspeicher und Sperrtabellen werden lokal für jeden Verarbeitungsknoten gehalten und können somit nicht zum Engpass werden. Bekannte Vertreter dieser Architektur sind zum Beispiel Teradata [Teradata], Netazza [Netazza] und Greenplum [Greenplum]. Zusätzlich interessant werden MPP-Systeme beim Einsatz einfacher Commodity-Hardware, wodurch sich kostengünstig sehr leistungsfähige Systeme bauen lassen.

Hardwarebeschleunigung Einen anderen Weg beschreiten Systeme, die durch den Einsatz spezieller und meist proprietärer Hardware Datenbankoperationen beschleunigen. Xtreme Data Appliance [Xtreme] nutzt dazu Hauptplatinen mit zwei Sockeln, wobei ein Standardprozessor, ergänzt um einen FPGA-Chip (Field Programmable Gate Array), zum Einsatz kommt. Der FPGA-Chip dient dabei der Beschleunigung von Kernroutinen bei der Anfrageverarbeitung. Netezza verwendet ebenfalls FPGAs und ergänzt mit diesen den Festplatten-Controller zur Datenvorverarbeitung. Weiterhin existieren Forschungsprojekte zur Nutzung spezieller Mehrkernarchitekturen, wie etwa Grafikkartenprozessoren oder der Cell-Prozessor von IBM [98]. Aus Sicht datenintensiver Data-Warehouse-Anwendungen, die vor allem durch langsame Fest-

plattenzugriffe beschränkt sind, werden jedoch vor allem hauptspeicherorientierte Ansätze relevant, die im folgenden Abschnitt beschrieben werden.

Hauptspeicherdatenbanken Die Optimierung von Data-Warehouse-Anfragen wird vor allem durch die Reduzierung des E/A-Aufwandes erreicht. Auf Seiten der Algorithmen und Software wird dies vor allem durch Materialisierung von Aggregationen, Reduzierung von Zwischenergebnissen durch den Optimierer, spaltenorientierte Speicherung, Kompression usw. erreicht. Auf Hardwareseite ist der E/A-Aufwand durch die extensive Nutzung vom Hauptspeicher reduzierbar, indem die Daten vollständig im Arbeitsspeicher hinterlegt und somit Festplattenzugriffe überflüssig werden. Durch den Einsatz von Kompressionsalgorithmen, die eine Reduzierung des Datenvolumens um den Faktor 5 bis 100 erlauben, ist es möglich, selbst sehr große Datenbanken komplett im Hauptspeicher abzulegen. Für noch höhere Anforderungen ist durch die Verwendung der Shared-Nothing-Architektur für Hauptspeicherdatenbanken eine nahezu beliebige Skalierung möglich (z.B. SAP BI Accelerator [SAPAcc]). Ein weiterer Vorteil der Hauptspeicher-DBMS besteht darin, dass durch den Verzicht auf Zugriffsstrukturen die Anfragezeiten relativ konstant bzw. vorhersagbar sind. Neben den eigenständigen Hauptspeicherdatenbanken existieren des Weiteren noch Hauptspeicher-Caches, wie zum Beispiel IBM solidDB Universal Cache [SolidDB] und Oracle In-Memory Database Cache [OracleInMem], zur Ergänzung relationaler Datenbanken.
Sind klassische Datenbanksysteme vor allem E/A-lastig, ist bei Hauptspeicherdatenbanken die CPU die begrenzende Ressource. Die im vorangegangenen Abschnitt vorgestellten speziellen Hardware-Architekturen sind daher auch zentraler Bestandteil vieler Hauptspeicherdatenbanken.

Datenbanksysteme

Auf Seiten der DBMS-Architekturen bzw. der Datenbank-Engines sind vor allem zwei Entwicklungen hervorzuheben: Im Allgemeinen ist dies die Diversifizierung der Datenbanksysteme in verschiedene Spezialsysteme zur Verarbeitung unterschiedlicher Datenformate und zum anderen der Trend hin zur spaltenorientierten Speicherung, vor allem in Analyseanwendungen.

Diversifizierung in spezialisierte Datenbank- bzw. Datenmanagementsysteme In einer Reihe von Papieren kritisieren Stonebraker et al. [75, 74, 76] die Produktstrategie der kommerziellen Datenbankanbieter, die dem Paradigma „one size fits all“ folgen und ihre Altsysteme stetig erweitern, um neuen Anforderungen gerecht zu werden. Stattdessen wird ein „rewrite from scratch“ gefordert, indem für spezifische Probleme wie Datenstromverarbeitung, Textsuche, Suchmaschinen, Informationsintegration, Data-Warehousing usw. angepasste Lösungen geschaffen werden. In Experimenten haben die Autoren gezeigt, dass spezialisierte Datenbanksysteme, wie zum Beispiel spaltenorientierte Datenbanken, im Bereich von zwei Größenordnungen

schneller sind als die etablierten, generalisierten Systeme. Franklin et al. formulieren das Problem in ihrem Artikel „From Databases to Dataspaces“ [28] in anderer Weise: Der Begriff „Datenbank“ wird als unzureichende Abstraktion beschrieben, der für viele Datenmanagementprobleme zu eng gefasst ist. Anstelle davon proklamieren die Autoren den Begriff der „Dataspaces“, welche Basisoperationen für verschiedene Arten von Datenquellen bereitstellen und je nach Anforderungen erweitert werden können. Als Entwicklungs- und Ausführungsumgebung wird eine DataSpace Support Platform (DSSP) vorgeschlagen, die es Entwicklern ermöglicht, sich auf die spezifischen Problemstellungen zu konzentrieren.
Dass die notwendige Diversifizierung der Datenmanagementsysteme bereits Realität geworden ist, belegt eine Gartner-Studie aus dem Jahr 2008 [24], die 15 Hersteller von Data-Warehouse- bzw. Data-Mart-Lösungen unterschiedlicher Spezialisierung nennt und vergleicht. Als besonders starker Trend sind die spaltenorientierten Datenbanksysteme zu nennen, die im folgenden Abschnitt detailliert beschrieben werden.

Zeilen- versus spaltenorientierte Datenbanksysteme Zur Abbildung von Datenbanktabellen, die auf konzeptioneller Ebene durch eine zweidimensionale Datenstruktur repräsentiert werden, auf die physische Ebene eines eindimensionalen Speicheradressbereichs existieren zwei Möglichkeiten: die zeilen- oder die spaltenorientierte Speicherung. Erfolgt der Zugriff auf Tabelleneinträge meist tupelweise, was inbesondere auch für Einfüge-, Lösch- und Aktualisierungsoperationen gilt, so ist die zeilenorientierte Speicherung im Vorteil. Wird jedoch, wie für OLAP-Anfragen typisch, meist nur auf wenige Spalten einer Tabelle zugegriffen, so reduziert die spaltenorientierte Speicherung den E/A-Aufwand und damit die Kosten der Anfrageverarbeitung erheblich. Änderungsoperationen in spaltenorientierten Systemen sind in der Regel teurer, da die Daten eines Tupels nicht, wie in zeilenorientierten Systemen, hintereinander gespeichert werden und somit mehrere Schreiboperationen notwendig sind. Jedoch fällt diese Einschränkung für OLAP-Anwendungen, bei denen Daten nur selten geschrieben werden, nicht so stark ins Gewicht. Die aufeinanderfolgende Speicherung von Daten gleichen Typs bei den spaltenorientierten Datenbanksystemen erlaubt den Einsatz vielfältiger Kompressionstechniken, wie zum Beispiel Lauflängenkodierung, Delta-Kodierung oder Wörterbuchkompression. Dadurch wird der E/A-Aufwand im Vergleich zu zeilenorientierten Systemen nochmals um ein Vielfaches reduziert. Die Anfrageverarbeitung wird durch den Einsatz von Kompressionstechniken jedoch nicht notwendigerweise verzögert. Stattdessen kann die Datenverarbeitung direkt auf den komprimierten Daten durchgeführt werden. Hierzu sind vor allem die Lauflängenkodierung, die Delta-Kodierung und einige Ansätze der Wörterbuchkompression geeignet [3].
Implementiert wurde die spaltenorientierte Speicherung und Anfrageverarbeitung sowohl durch Forschungsprototypen wie C-Store (mittlerweile in Vertica kommerzialisiert) und Monet-DB sowie in kommerziellen Systemen wie Sybase IQ (festspeicherbasiert) und SAP BI Accelerator (hauptspeicherbasiert).

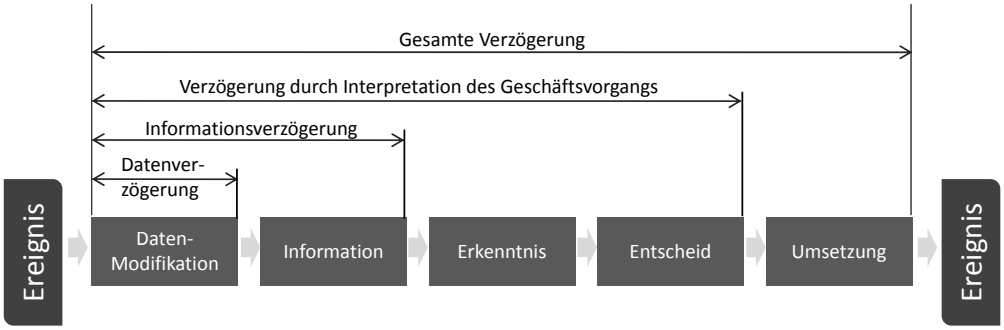


Abbildung 3.8: Lebenszyklus eines Geschäftsentscheids

3.4 Architektur eines Echtzeit-Data-Warehouse

Der in Abschnitt 3.3 dargestellte Trend hin zur Einbindung von Data-Warehouse-Systemen in operative Entscheidungsprozesse führt zu der Forderung nach echtzeitfähigen Data-Warehouses. Auf Grundlage des in Abschnitt 3.1 definierten Data-Warehouse-Begriffs und der vorgestellten Referenzarchitektur wird im Folgenden der Aufbau eines Echtzeit-Data-Warehouse konzipiert. Dem vorangestellt ist jedoch zunächst der Begriff „Echtzeit“ zu konkretisieren. Weiterhin werden die zur Realisierung eines Echtzeit-Data-Warehouse erforderlichen Annahmen in einem Systemmodell zusammengefasst.

3.4.1 Der Echtzeit-Begriff im Data-Warehouse-Umfeld

In Abschnitt 3.3.1 wurde die zunehmende Notwendigkeit nach hochaktuellen Daten in der Data-Warehouse-Nutzung diskutiert. Für Data-Warehouses, die diesen Anspruch adressieren hat sich der Begriff Echtzeit-Data-Warehouse (engl. real-time data warehouse) in der Literatur durchgesetzt, der jedoch wenig mit dem klassischen Echtzeit-Begriff gemeinsam hat. Im Allgemeinen wird ein Computersystem als echtzeitfähig beschrieben, wenn Ergebnisse innerhalb fest definierter Zeitintervalle garantiert berechnet werden können. Dabei steht vor allem die Vorhersagbarkeit, wann ein Ergebnis geliefert werden kann im Vordergrund. Die Performanz des Systems bleibt sekundär. Dies steht im Gegensatz zur Auffassung des Echtzeit-Begriffs im Umfeld von Data-Warehouses. Hier meint der Begriff Echtzeit, Änderungen in der Realwelt bzw. der Diskurswelt zeitnah im Data-Warehouse abzubilden. Um dies zu erreichen, sind sowohl der Prozess der Datenbeschaffung als auch die Einbringung von Aktualisierungen sowie die Anfrageverarbeitung performant umzusetzen. Dies gilt ebenfalls für den Informationsrückfluss im Fall einer Closed-Loop-Architektur, wie sie in *Active Data Warehouses* gegeben ist.
Ein Data-Warehouse-System echtzeitfähig – im klassischen Sinne – zu gestalten,

birgt vielerlei Schwierigkeiten: Zum einen müssen Änderungen sowohl an den Quellsystemen als auch im Data-Warehouse innerhalb einer Transaktion eingebracht werden. Dies ist in lose gekoppelten Data-Warehouse-Systemen nur über verteilte Transaktionsprotokolle möglich, welche jedoch zu langen Schreibsperren an den Quellsystemen führen. Der Prozess der Datenbeschaffung in Data-Warehouses ist daher von den Quellsystemen entkoppelt bzw. asynchron. Ein weiteres Problem stellt die Antwortzeit von Anfragen dar. Die in relationalen Datenbanksystemen verwendeten Anfrageoptimierer liefern nur geschätzte Kosten bzw. Anfragezeiten, die in keinem Fall garantiert werden können. Um dies zu gewährleisten, sind Echtzeitdatenbanken notwendig, die aufgrund der Vielzahl der zu lösenden Problemstellungen einen eigenen Forschungsbereich darstellen [61].
Diese Aspekte spielen in der vorliegenden Arbeit keine Rolle. Das Augenmerk liegt vielmehr in der Integration hochaktueller Daten in das Data-Warehouse-System unter Beibehaltung der in Abschnitt 3.1.1 beschriebenen Eigenschaften. Aufgrund der Mehrdeutigkeit des Echtzeit-Begriffs wird in der Literatur auch von Near-Real-Time-, Right-Time-, On-Time- oder Living-Data-Warehouses gesprochen.

3.4.2 Architektur eines Echtzeit-Data-Warehouses

Ausgehend von der Referenzarchitektur aus Abschnitt 3.1 wird im Folgenden die Architektur eines Echtzeit-Data-Warehouse skizziert (siehe Abbildung 3.9).
Die wesentlichen Unterscheidungsmerkmale im Vergleich zu klassischen Architektur, sind zum einen die kürzeren Aktualisierungszyklen und zum anderen die Anfragezugriffe, die nicht nur auf die Data-Marts beschränkt sind, sondern entlang des Datenverfeinerungsprozess in jeder Prozessstufe ausgeführt werden können. Statt der bisher batch-orientierten Verarbeitungsweise von Änderungen, wodurch Anfragen und Aktualisierungen streng voneinander getrennt wurden, ist im Kontext von Echtzeit-Data-Warehouses eine gleichzeitige Verarbeitung von lesenden und schreibenden Transaktionen notwendig. Die Funktionalität des Datenintegrationsprozess als solches bleibt jedoch davon unberührt. Bill Inmon hat die Bedeutung der Integration in [39] wie folgt formuliert:

„There is no point in bringing data ... into the data warehouse environment without integrating it. If the data arrives at the data warehouse in an unintegrated state, it cannot be used to support a corporate view of data. And a corporate view of data is one of the essences of the architected environment.“

Der in dieser Dissertation gewählte Ansatz besteht somit darin, sowohl die Anforderungen nach hochaktuellen Daten zu bedienen als auch den klassischen Mehrwert eines Data-Warehouse, als eine integrierte, stabilen Datenbasis, zu erhalten. Die Nutzer eines Echtzeit-Data-Warehouse sind sich zum einen der Volatilität der Daten bewusst und können zum anderen gezielt auf noch nicht vollständig integrierte und verfeinerte Daten zugreifen um den Forderungen nach hochaktuellen Daten gerecht zu werden. Data-Warehouse-Anfragen sind um die entsprechenden Anforderungen annotiert und werden von einer Anfrageschicht (siehe Abbildung 3.9) transparent

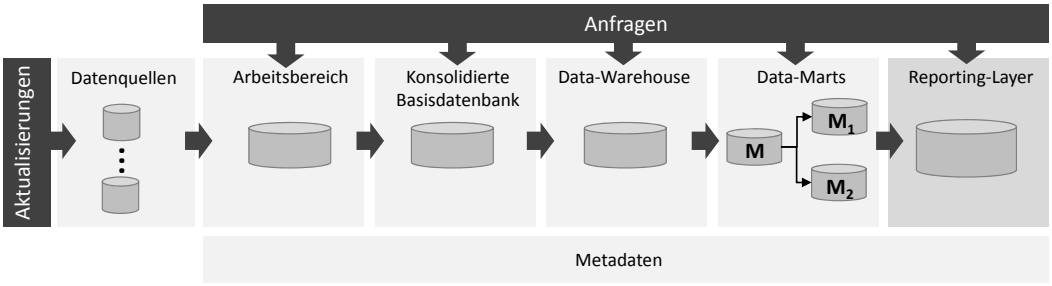


Abbildung 3.9: Architektur eines Echtzeit-Data-Warehouses

auf die einzelnen Prozessstufen umgeleitet. Dies setzt entsprechende Umschreibemechanismen für Anfragen (engl. query-rewrite) voraus.
Um die von einem Data-Warehouse geforderte Datenstabilität zu gewährleisten, ist des Weiteren eine neue Datenschicht erforderlich, die von dem kontinuierlich aktualisierten Echtzeit-Data-Warehouse entkoppelt ist. Diese ist in Abbildung 3.9 als Reporting-Layer dargestellt und wird in Kapitel 6 ausführlich betrachtet. Der Reporting-Layer als letzte Stufe im Datenproduktionsprozess wird ebenfalls von der Anfrageschicht bedient, so dass Anfragen mit den entsprechenden Anforderungen nach Datenstabilität (zum Beispiel für die Berichtsproduktion) automatisch an diese Stufe weitergeleitet werden.
Die Synchronisierung der stetigen Last von Anfragen und Aktualisierung denen ein Echtzeit-Data-Warehouse ausgesetzt ist, wird in den Kapiteln 4 und 5 betrachtet. Kapitel 4 behandelt dabei zunächst nur den einstufigen Fall, d.h. es wird jeweils nur eine, der in Abbildung 3.9 dargestellten, Prozessstufen isoliert betrachtet. In Kapitel 5 wird dieses Problem auf den gesamten Datenproduktionsprozess erweitert, wobei jedoch die Datenübernahme in den Reporting-Layer getrennt zu untersuchen ist.

3.4.3 Systemmodell

Die zur Realisierung eines Echtzeit-Data-Warehouse-Systems erforderlichen Annahmen und Rahmenbedingungen werden in diesem Abschnitt zu einem gemeinsamen Systemmodell (siehe Abbildung 3.10) zusammengefasst. Die Darstellung der einzelnen Modelle erfolgt zunächst nur schematisch und wird in den folgenden Kapiteln detailliert ausgeführt.

Nutzer- und Workloadmodell

Die Analyse der Data-Warehouse-Anwendungen und -Nutzer in Abschnitt 3.3.1 hat gezeigt, dass diese in drei Entscheidungsebenen klassifiziert werden können, die insbesondere den erforderlichen Grad der Datenaktualität spezifizieren. Insofern ist die Notwendigkeit von Echtzeitdaten nicht für jeden Nutzer gegeben. Diese Unterscheidung kann zur Priorisierung des Stroms an Aktualisierungen und damit zum performanten Betrieb eines Echtzeit-Data-Warehouses verwendet werden. Dazu muss

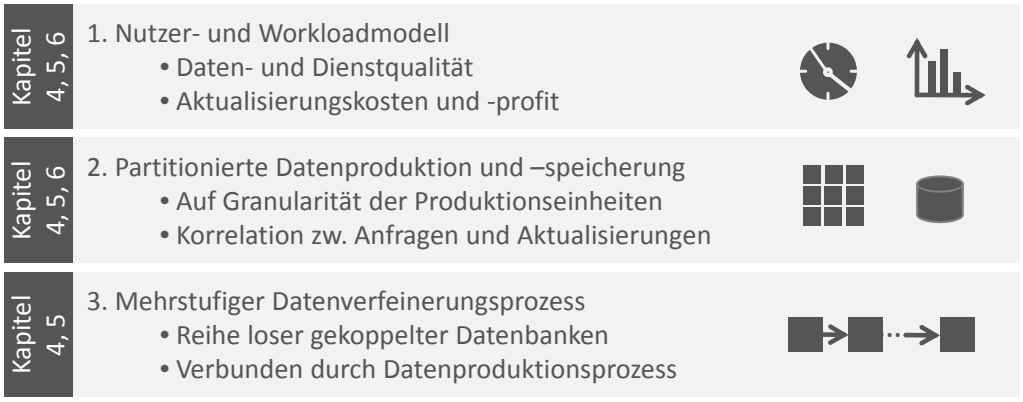


Abbildung 3.10: Systemmodell

der Nutzer in der Lage sein, seine Anforderungen bezüglich der Datenaktualität in Form einer Maßzahl auszudrücken. Dieses Maß wird im Weiteren abstrakt als Datenqualität (engl. Quality of Data, kurz QoD) bezeichnet. Die Datenaktualität ist nur eine mögliche Ausprägung der Datenqualität, welche je nach Anforderung auch durch verschiedene Qualitätsdimensionen besetzt werden kann. Ein Vergleich verschiedener Qualitätsdimensionen und Metriken erfolgt in Kapitel 4.
Die Forderung nach einer hohen Datenaktualität und der damit verbundenen Priorisierung von Schreibtransaktionen führt aufgrund von Sperrkonflikten zur Verzögerung zeitgleich laufender Anfragen. Die erhöhte Antwortzeit führt zur Verminderung der Dienstqualität (engl. Quality of Service, kurz QoS) des Echtzeit-Data-Warehouses, welche den zweiten Nutzerparameter bildet. Beide Parameter, QoS und QoD, stehen in einer wechselseitigen, jedoch nicht linearen Abhängigkeit zueinander. Ein Algorithmus zur Ablaufplanung, der diesen Konflikt adressiert, wird in Kapitel 4 entwickelt.
Der Workload eines Echtzeit-Data-Warehouses teilt sich in zwei Arten von Transaktionen: Anfragen einerseits und Einfüge-, Lösch- bzw. Aktualisierungsoperationen andererseits, die im Weiteren nur noch als Aktualisierungen bezeichnet werden. Aufgrund der push-basierten und somit von Anfragen unabhängigen Propagierung von Aktualisierungen sind gemischte Transaktionen nicht zulässig. Reihenfolgeabhängigkeiten zwischen Aktualisierungen untereinander können jedoch auftreten.
Das Workloadmodell setzt sich aus einer Reihe von Parametern zusammen, die zur Priorisierung der Transaktion benötigt werden: Für Anfragen und Aktualisierungen sind dies die vom Optimierer geschätzten Kosten. Anfragen werden des Weiteren um die vom Nutzer spezifizierten QoS- und QoD-Parameter erweitert. Für Aktualisierungen wird ein Parameter namens Profit ermittelt, der den Nutzen der Aktualisierung aus Sicht der Anfragen spezifiziert. Die Semantik dieses Parameters ist wiederum abhängig von der konkreten Datenqualitätsdimension.

Partitionierte Datenproduktion und -speicherung

Eine weitere für die Realisierung eines Echtzeit-Data-Warehouse-Systems grundlegende Annahme ist die der partitionierten Datenproduktion und -speicherung. Das bedeutet, dass alle Daten, die den Datenproduktionsprozess durchlaufen sowie alle persistierten Daten bezüglich einer gemeinsamen Menge dimensionaler Elemente partitioniert werden. Die Partitionierungsdimensionen bzw. -attribute sind durch die Granularität der Einheiten vorgegeben, in denen die Daten produziert werden. Am Beispiel der Fallstudie B aus Abschnitt 2.2 sind dies etwa die Produktfamilie der Produktdimension, das Land seitens der Geschäftsdimension und Wochen- bzw. Monatsperioden in der Zeitdimension. Somit werden die Datenelemente anhand der durch die Anwendungsdomäne implizit vorgegebenen Dimensionen in logische Einheiten unterteilt. Dimensionsattribute, die für das entsprechende Geschäftsmodell keine treibende Rolle spielen (zum Beispiel das Attribut Farbe in dem GfK-Szenario), qualifizieren sich daher nicht als partitionierendes Kriterium.
Durch die Eigenschaft der Partitionierung sind zwei wichtige Systemfunktionen effizient realisierbar: Zum einen können Korrelationen zwischen Anfragen und Aktualisierungen auf Basis der Partitionen ermittelt werden. Dies ist notwendig, um eine nutzer- bzw. qualitätsgetriebene Ablaufplanung umzusetzen (siehe Kapitel 4 und 5). Zum anderen ermöglicht die Partitionierung die Realisierung eines effizienten Verfahrens zur Ermittlung der Datenvollständigkeit, wie es für die in Kapitel 6 zu entwickelnde Datenschicht benötigt wird.

Mehrstufiger Datenverfeinerungsprozess

Ein Data-Warehouse-System ist, wie die Referenzarchitektur in Abschnitt 3.1.2 sowie die Fallstudien in Kapitel 2 gezeigt haben, per se ein mehrstufiger Prozess. Es wird aus einer Reihe lose gekoppelter Einzelsysteme gebildet, die durch einen gemeinsamen Datenproduktionsprozess miteinander verbunden sind. Der Begriff Datenproduktionsprozess – in Analogie zu klassischen Produktionsprozessen der Sachgüterfertigung – ist dabei bewusst gewählt. Auch Daten erfahren, ausgehend von ihrem Rohzustand, eine schrittweise Verfeinerung bzw. Veredelung, bis sie schließlich in Form von Berichten oder Datenexports verteilt werden. Die Produktionseinheit ist durch die im vorangegangenen Abschnitt vorgestellten Partitionen gegeben.
Die Anzahl der Stufen eines Datenproduktionsprozesses wird vor allem durch zwei Faktoren bestimmt: zum einen durch die in der Referenzarchitektur dargestellten Verarbeitungsschritte zur Datenintegration (Abschnitt 3.1.2), die in Abhängigkeit von den Anwendungen variieren. Zum anderen ist das die Diversifizierung der Datenbanksysteme und damit auch der Data-Warehouse-Landschaft, durch Einführung von Spezialsystemen für unterschiedliche Datenarten und Anwendungsbereiche (Abschnitt 3.3.2).
In Kapitel 4 wird ein Algorithmus zur Ablaufplanung von Aktualisierungen in einem zunächst einstufigen System erarbeitet. Die Evaluierung von Ablaufplanungsalgorithmen in mehrstufigen Systemen ist Inhalt von Kapitel 5. In Kapitel 6 wird durch

Einführung einer zusätzlichen Datenschicht zur Gewährleistung der Datenstabilität der Datenproduktionsprozess um eine weitere Stufe ergänzt.

3.5 Anforderungen an ein Echtzeit-Data-Warehouse

Auf Grundlage des im vorhergehenden Abschnitt erweiterten Data-Warehouse-Begriffs werden im Folgenden die Anforderungen an ein Echtzeit-Data-Warehouse analysiert und damit die Thesen dieser Dissertation formuliert.

3.5.1 Maximierung der Datenaktualität

Unter Echtzeitfähigkeit eines Data-Warehouse-Systems wird die Fähigkeit verstanden, Daten schnell in das Data-Warehouse zu integrieren und diese in für die Datenanalyse geeignete Formate zu überführen. Der Prozess vom Eintreten eines Ereignisses bis hin zu einer daraus folgenden Entscheidung und deren Umsetzung ist in Abbildung 3.8 dargestellt. Dieser ist in fünf Stufen unterteilt, wobei jede Stufe den Prozess um einen gewissen Grad verzögert. Die erste Stufe ist die der Datenmodifikation, welche im Kontext von Data-Warehouse-Systemen den ETL- bzw. Datenproduktionsprozess bezeichnet. Die durch den ETL-Prozess eingeführte zeitliche Verzögerung wird als Datenverzögerung bezeichnet. Eine Reduzierung der Datenverzögerung ist unter anderem durch logische und physische Optimierung der ETL-Prozesse und durch Bereitstellung ausreichender Hardware-Ressourcen zu erreichen. Diese Maßnahmen vorausgesetzt, ist eine entscheidende Verringerung der Datenverzögerung, vor allem durch kürzere Ladezyklen, zu erzielen. Die Batch-orientierten Ladeprozesse werden dabei durch einen kontinuierlichen Strom von Aktualisierungen ersetzt, so dass Änderungen in der Realwelt sofort an das Data-Warehouse propagiert werden.
Diese Aktualisierungssemantik ermöglicht eine hohe Datenaktualität, steht jedoch in Konflikt mit der Anfragelatenz (Abschnitt 3.5.2) und der Datenstabilität (Abschnitt 3.5.3), wie in Abbildung 3.11 dargestellt. Die Anwendungsanalyse in Abschnitt 3.3.1 hat jedoch gezeigt, dass Data-Warehouse-Workloads bezüglich der Anforderung an die Datenaktualität sehr heterogen strukturiert sind. Daraus folgt, dass hinsichtlich der Relevanz einer Aktualisierung eine Abstufung existiert, die zur Priorisierung bzw. zur Ablaufplanung verwendet werden kann. Um den Grad der erforderlichen Datenaktualität seitens der Anwendung auszudrücken, bedarf es eines Nutzer- bzw. Workloadmodells, das in Abschnitt 3.4.3 beschrieben ist.
Die im Abschnitt 3.1.2 vorgestellte Referenzarchitektur verfolgte einen anderen Ansatz, um mit unterschiedlichen Datenaktualitätsanforderungen umzugehen, indem eine Trennung in Data-Warehouse und Operational Data Store vorgenommen wurde. Die historischen Daten werden somit von definierten Datenbereichen hoher Aktualität separiert. Dies setzt voraus, dass zur Entwurfszeit bekannt ist, welche Datenbereiche besonderen Aktualitätsanforderungen genügen müssen. Für beliebige Ad-hoc-Anfragen kann jedoch die Datenaktualität nicht garantiert werden. In dieser Arbeit wird daher ein verallgemeinerter Ansatz eines Echtzeit-Data-Warehouse-

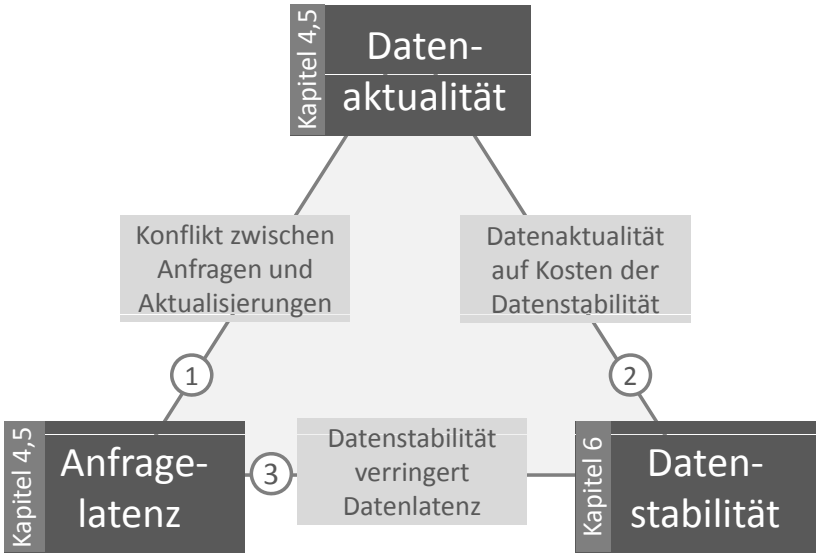


Abbildung 3.11: Anforderungen an ein Echtzeit-Data-Warehouse

Systems vorschlagen, in dem für alle Daten der gleiche Datenproduktionsprozess angewandt wird und erst zur Laufzeit eine Priorisierung erfolgt.

3.5.2 Minimierung der Anfragelatenz

In der zweiten Stufe in Abbildung 3.8 werden durch das Formulieren von Anfragen aus den Daten Informationen abgeleitet. Die Latenz, die bei der Anfrageverarbeitung durch das Datenbanksystem entsteht, wird als Informationsverzögerung bezeichnet (siehe Abbildung 3.8). Sie ist durch die in Abschnitt 3.3.2 vorgestellten Optimierungen bzw. alternativen Hardware- und Datenbanksystemarchitekturen reduzierbar. Jedoch sind die Optimierungen zum großen Teil für den klassischen Data-Warehouse-Einsatz konzipiert und setzen daher dedizierte Aktualisierungsfenster voraus. Für die in Echtzeit-Data-Warehouses vorkommenden gemischten Workloads, bestehend aus Anfragen und Aktualisierungen, ist ein allgemeinerer Ansatz erforderlich. Um den in Abbildung 3.11 dargestellten Konflikt zwischen Datenaktualität und Anfragelatenz bzw. zwischen Aktualisierungen und Anfragen zu lösen, muss das im vorangegangenen Abschnitt vorgeschlagene Nutzermodell erweitert werden. Neben der erforderlichen Datenaktualität muss auch die benötigte Anfragelatenz durch den Nutzer ausdrückbar sein. Dadurch erweitert sich auch das Scheduling-Problem um eine zusätzliche Dimension: Die Aktualisierungen müssen so priorisiert werden, dass zum einen die Datenaktualität maximiert und zum anderen die Anfragelatenz minimiert wird. Da beide Kriterien nicht unabhängig voneinander optimiert werden können, ergibt sich ein multikriterielles Optimierungsproblem, das in Kapitel 4 im Detail betrachtet wird.
Die Stufen 3 bis 5 in Abbildung 3.8 sind im Wesentlichen durch menschliche Inter-

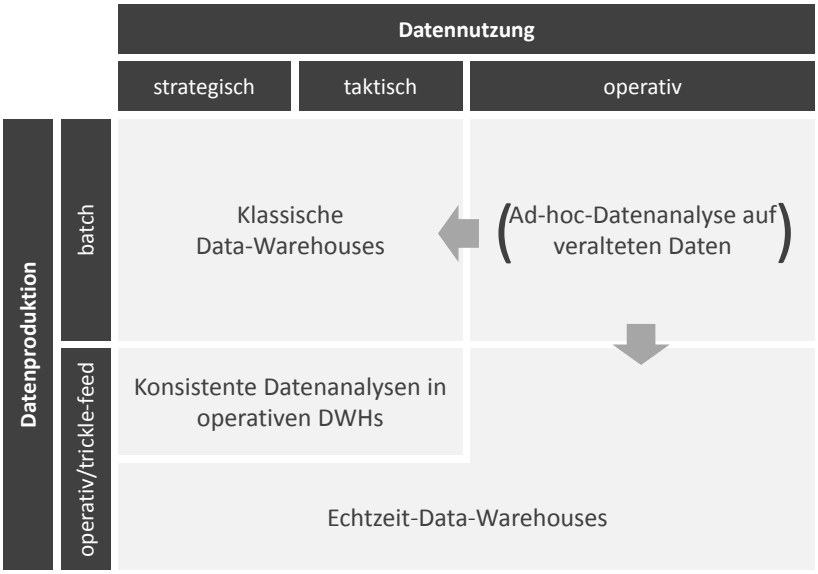


Abbildung 3.12: Klassifikation in Datenproduktion und -nutzung

aktion bestimmt bzw. verzögert und werden daher im Weiteren nicht untersucht.

3.5.3 Erhalt der Datenstabilität

Die Anwendungsanalyse in Abschnitt 3.3.1 hat gezeigt, dass Data-Warehouse-Systeme sowohl in Anwendungen zur operativen als auch zur klassischen, strategischen Entscheidungsfindung eingesetzt werden. Die push-basierte Aktualisierungssemantik eines Echtzeit-Data-Warehouse-Systems erhöht zwar die Datenaktualität, wie sie in operativen Prozessen gefordert wird, stellt jedoch ein Problem für strategische Analysen dar. Die fortwährenden Datenänderungen, die im Widerspruch zum klassischen Data-Warehouse-Begriff (siehe Abschnitt 3.1.1) stehen, führen zu Inkonsistenzen in Analyseprozessen sowie in der Berichtsproduktion. Des Weiteren existieren viele Data-Warehouse-Szenarien, in denen die Datenbasis kontinuierlich verändert wird, da die Datenproduktion aufgrund betrieblicher Abläufe nicht homogenisiert werden kann. Die Datenauswertung jedoch erfolgt weiterhin klassisch, auf Ebene taktischer und strategischer Entscheidungsfindungen, zumeist durch die Erzeugung von Standardberichten. Eine Klassifikation bezüglich der Datenproduktion und der Datennutzung ist in Abbildung 3.12 dargestellt: Im klassischen Data-Warehouse-Betrieb ist die Datenstabilität durch die in Batches organisierten Ladeprozesse garantiert. Dahingegen kann im Fall der kontinuierlichen Aktualisierungen nur durch Einführung einer zusätzlichen, entkoppelten Datenschicht die Stabilität gewährleistet werden. Stabilität heißt insbesondere, dass sich berichtsrelevante Daten nicht unkontrolliert ändern können, sondern definierte Publikationszeitpunkte existieren.

Kapitel 6 beschreibt Aufbau, Organisation und Betrieb dieser Datenschicht.
Die Forderung nach Datenstabilität steht, wie schon in Abbildung 3.11 gezeigt, im Konflikt mit dem Bedürfnis nach Datenaktualität. In Bezug zur Anfragelatenz ist die Datenstabilität jedoch von Vorteil, da sie Möglichkeiten zur Vorberechnung von Aggregationen eröffnet. In Kapitel 6 werden entsprechende Strategien diskutiert.
Es bleibt anzumerken, dass der in Abbildung 3.12 dargestellte Fall der operativen Datennutzung auf in Batches aktualisierten Daten eher pathologischer Natur ist. Dieser ist stattdessen durch den Betrieb eines klassischen oder eines echtzeitfähigen Data-Warehouses zu ersetzen.
In der Referenzarchitektur aus den Abschnitt 3.1.2 ist das Problem der Datenstabilität durch die explizite Trennung von Data-Warehouse und Operational Data Store gelöst, so dass die historischen, nicht veränderlichen Daten von den transienten Daten separat gespeichert sind. Diese Trennung ist aufgrund der oben genannten Nachteile für ein Echtzeit-Data-Warehouse-System nicht adäquat und ist daher in ein allgemeineres, ganzheitliches Konzept zu überführen.

4 Datenproduktionssteuerung in einstufigen Systemen

Data-Warehouse-Systeme sind als zentrale Plattformen für die integrierte Informationsversorgung gesamter Unternehmen in einer Vielzahl von unterschiedlichen Anwendungen und Geschäftsprozessen involviert. In Abschnitt 3.3.1 wurden diese zum einen in Anwendungstypen als auch in Entscheidungsebenen (strategisch, taktisch und operativ) klassifiziert. Besonders die operativen Anwendungen – und damit die Forderung nach hochaktuellen Daten – stellen Data-Warehouse-Systeme vor große Herausforderungen. Zur Gewährleistung einer hohen Datenaktualität muss der klassische Batch-Betrieb zur Beladung des Data-Warehouse durch eine push-basierte Propagierung von Aktualisierungen ersetzt werden. Der daraus resultierende kontinuierliche Strom von Aktualisierungen steht im Konflikt zum Anfrageworkload. Aus Sicht der Data-Warehouse-Nutzer ergeben sich daraus zwei Möglichkeiten. Zum einen können auf Kosten der Aktualität Aktualisierungen verzögert werden, um so die Anfrageverarbeitung zu beschleunigen. Zum anderen können vor der Anfrageausführung stets alle relevanten Aktualisierungen eingebracht werden, wodurch die Aktualität zwar maximiert, die Anfrageverarbeitung jedoch stark verzögert wird. Beide Optimierungsziele stehen offensichtlich im Konflikt zueinander und können nicht getrennt optimiert werden. Diese Art von Optimierungsproblem wird als multikriterielle Optimierung bezeichnet.
Im Folgenden werden zunächst die Dienst- und Datenqualitätskriterien im Allgemeinen betrachtet und für den Anwendungsfall geeignete Metriken ausgewählt. Das multikriterielle Optimierungsproblem wird auf ein bekanntes Problem der Kombinatorik zurückgeführt und es wird ein Algorithmus erarbeitet, der dieses Problem effizient löst. Darauf aufbauend wird dieser in ein Online-Scheduling-Verfahren integriert. Die partitionsbasierte Korrelationsbestimmung zwischen Anfragen und Aktualisierungen wird im Detail untersucht und Optimierungen werden erarbeitet. Eine umfassende Evaluierung der Qualität der Ablaufplanung sowie der Laufzeit- und Speicherkomplexität schließen dieses Kapitel ab.

4.1 Qualitätskriterien und Systemmodell

Unter dem Begriff Qualitätskriterium im Kontext der Datenverarbeitung sind sowohl die Dienst- als auch die Datenqualität subsummiert, die sich wiederum in eine Vielzahl von Dimensionen unterscheiden lassen. Im ersten Teil dieses Abschnitts werden verschiedene Qualitätsdimensionen vorgestellt und aus diesen die zur Beschreibung des Optimierungsziels geeigneten Dimensionen ausgewählt. Diese Qualitätsdimen-

Qualitätdimension	Qualitätskriterien
Dienstqualität	Antwortzeit, Wiederherstellbarkeit, Durchsatz, Skalierbarkeit, Verfügbarkeit, Wartbarkeit, Flexibilität
Datenqualität	Aktualität, Genauigkeit, Vollständigkeit, Zuverlässigkeit, Glaubhaftigkeit, Nachverfolgbarkeit, Objektivität, Reputation, Relevanz, Datenmenge, Mehrwert, Interpretierbarkeit, Verständlichkeit, Konsistenz, Prägnanz, Zugänglichkeit, Sicherheit

Abbildung 4.1: Dienstqualitäts- und Datenqualitätskriterien

sionen spannen ein multikriterielles Optimierungsproblem auf, welches in Abschnitt 4.1.3 zunächst abstrakt formuliert wird. Die zur Optimierung notwendigen Annahmen an das Data-Warehouse-System sowie den Workload werden abschließend in einem Modell in Abschnitt 4.1.4 skizziert.

4.1.1 Dienstqualitätskriterien

Die Dienstqualität eines Systems umfasst die Menge der qualitativen und quantitativen Kriterien, mit denen ein Dienst erbracht werden kann und wird insbesondere zur Bewertung und Analyse von Multimediaanwendungen, Datennetzwerken, Datenstrom- und Datenbanksystemen angewandt. Die wichtigsten und in der Literatur am häufigsten zitierten Dienstqualitätskriterien sind in Abbildung 4.1 dargestellt. Dienstqualitätskriterien die nur qualitativ beschrieben und daher nicht weiter betrachtet werden, sind zum Beispiel die Wiederherstellbarkeit, die Wartbarkeit sowie die Flexibilität. Die Notwendigkeit, Dienstqualitätskriterien auch in Informationssysteme zu integrieren wird in [87] und [11] argumentiert und ist in dem Forschungsprojekt Mariposa [73], das sich mit der verteilten Datenhaltung beschäftigt, konsequent umgesetzt.
Von besonderer Relevanz bei der Bewertung der Dienstgüte eines Information-Management- bzw. eines Datenbanksystems sind vor allem die quantitativen Dienstqualitätskriterien wie der Durchsatz, die mittlere Antwortzeit, die Varianz der Antwortzeiten und die gewichtete mittlere Antwortzeit (engl. mean stretch). Die Frage, welche dieser Metriken die Qualität der Nutzererfahrung am besten widerspiegelt, ist nur schwer zu beantworten und wurde in einer Reihe von Studien untersucht [94, 19]. Wie diese Frage zu beantworten ist, hängt aber insbesondere von der Art der Anwendung ab: In OLTP-Systemen, in denen die Anfragen oft gleichartig strukturiert sind und daher eine ähnliche Verarbeitungszeit für alle Anfragen zu erwarten ist, liefert die mittlere Antwortzeit ein gutes Maß für die Qualität bzw. die Geschwindigkeit der Datenbank. In OLAP- bzw. BI-Anwendungen hingegen kann sich die Verarbeitungszeit einzelner Anfragen um ganze Größenordnungen unterscheiden. Beispielsweise wurden in [53] eine Reihe von Workloads aus der Praxis evaluiert und Unterschiede in der Laufzeit von Anfragen um den Faktor 10.000 festgestellt. Die

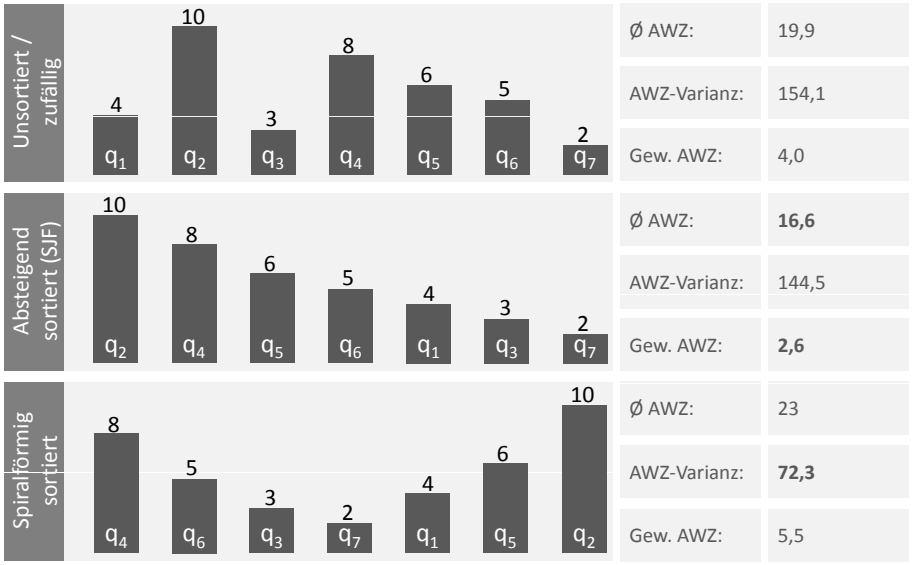


Abbildung 4.2: Auswirkung der Ablaufplanung auf die Dienstqualitätsmetriken

Verteilung der Verarbeitungszeit in solchen Workloads wird daher als endlastig (engl. heavy-tailed) bezeichnet. Da in die Berechnung der Antwortzeit einer Anfrage auch die eigene Verarbeitungszeit mit eingeht, ist für endlastige Workloads die mittlere Antwortzeit zu ungenau bzw. aus Nutzersicht nicht fair. Das heißt, ist sich ein Nutzer der Komplexität – und damit der Laufzeit seiner Anfrage – bewusst, so wird er auch eine proportional höhere Antwortzeit erwarten. Statt der mittleren Antwortzeit sollte daher die mittlere gewichtete Antwortzeit verwendet werden, bei der die Antwortzeit einer Anfrage zusätzlich auf die eigene Ausführungszeit normiert ist. Die Anfrageplanung im Kontext von BI-Systemen, in denen endlastige Workloads häufig zu finden sind, wurde in [45, 46, 14, 53] betrachtet.

Eine weitere wichtige Metrik zur Messung der Dienstqualität ist die Varianz der Antwortzeiten. Untersuchungen in [19] haben gezeigt, dass Nutzer eine höhere mittlere Antwortzeit akzeptieren, wenn im Gegenzug die einzelnen Antwortzeiten untereinander weniger Varianz aufweisen. Mathematisch wird dies durch die Minimierung der maximalen Antwortzeit aller Anfragen eines Workloads erreicht.

Der Durchsatz, als Maß für die Gesamtperformanz eines Systems, ist kein guter Indikator für die Qualität der einzelnen Nutzererfahrungen und wird daher nicht weiter betrachtet.

Um die vorgestellten Metriken in ihrer Anwendbarkeit hinsichtlich der Bewertung der Dienstqualität zu bewerten, ist in Abbildung 4.2 ein Beispiel skizziert. In diesem sind drei Ablaufpläne, bestehend aus sieben Anfragen mit unterschiedlichen Bearbeitungszeiten bzw. Kosten, dargestellt. Jeder der drei Ablaufpläne wird bezüglich der mittleren Antwortzeit (AWZ), der Varianz der Antwortzeiten und der gewichteten

mittleren Antwortzeit bewertet. Die Anfragen im ersten Ablaufplan sind nach keinem Schema geordnet und liegen in der Reihenfolge ihrer Ankunft vor. Entsprechend sind die Werte der einzelnen Metriken nur durchschnittlich bis schlecht. Im zweiten Ablaufplan wurden die Anfragen absteigend nach ihrer Ausführungszeit geordnet. Dieses Schema der Ablaufplanung wird als Shortest-Job-First, kurz SJF, bezeichnet und führt bewiesenermaßen zur minimalen mittleren Antwortzeit und auch zur minimalen gewichteten mittleren Antwortzeit [65, 64]. Um den Effekt des Verhungerns zu vermeiden, der auftreten kann, wenn Anfragen sehr hoher Bearbeitungsdauer fortwährend durch kleinere Anfragen verdrängt werden, kann Shortest-Job-First durch Lottery-Scheduling-Algorithmen approximiert werden [85].
Die Minimierung der Varianz der Antwortzeiten ist, im Gegensatz zu den vorhergehenden Metriken, ein NP-hartes Problem und daher nur für kurze Ablaufpläne optimal lösbar. In [92] werden zwei heuristische Algorithmen – Balanced-Spiral und Verified-Spiral – zur näherungsweisen Lösung dieses Problems vorgestellt. Für die sieben Beispielanfragen ist der optimale Ablaufplan in Abbildung 4.2 dargestellt. Es ist ersichtlich, dass die Varianz der Antwortzeiten im Vergleich zu den anderen Ablaufplänen deutlich verbessert werden konnte.

Dienstqualität und Aktualisierungen

Bisher wurde die Dienstqualität nur in Verbindung mit den Anfragen eines Systems betrachtet. Jedoch haben Aktualisierungen einen ebenso großen Einfluss. In Abschnitt 3.5 wurde der Konflikt zwischen Datenaktualität und Datenlatenz in einem Data-Warehouse-System dargestellt. Die Ausführung einer Aktualisierung verbessert die Aktualität eines Systems (siehe Abschnitt 4.1.2), erhöht jedoch die Datenlatenz, da Anfragen verzögert werden müssen. Somit kann eine Optimierung der Dienstqualität nur unter gleichzeitiger Betrachtung der Anfragen und der Aktualisierungen erfolgen. Um dies zu gewährleisten, ist der Ablaufplan der Anfragen als gegeben zu betrachten. Das zu verwendende Ablaufplanschema ist dabei abhängig von der Wahl des Optimierungskriteriums.
Sollen in den Anfrageablaufplan Aktualisierungen eingebettet werden, so ist zu bewerten, wie sich das auf die Datenlatenz und die Datenaktualität der Anfragen auswirkt. Letzteres wird im darauf folgenden Abschnitt diskutiert. Die Auswirkung auf die Datenlatenz wird an dem in Abbildung 4.2 dargestellten Beispiel diskutiert. Unter Verwendung des Ablaufplanschemas Shortest-Job-First führt das Einfügen von Aktualisierungen, die ebenfalls mit Kosten behaftet sind, zu einer stetigen Verschlechterung der mittleren Antwortzeit sowie der gewichteten mittleren Antwortzeit. Die Auswirkung einer Aktualisierung auf die Antwortzeiten ist umso größer, je weiter rechts die Aktualisierung in den Anfrageablaufplan eingeschoben wird. Weiterhin kann das Einfügen von Aktualisierungen unabhängig voneinander betrachtet werden, da die Auswirkungen auf die Antwortzeiten proportional stets die gleichen sind. Besteht das Optimierungskriterium aus der Minimierung der Varianz der Antwortzeiten, sind diese Eigenschaften nicht gegeben. Aufgrund des V-förmigen Ablaufplanes (siehe Abbildung 4.2), der die Optimalität der Varianz garantiert, beschreibt

die Veränderung der Varianz beim Einfügen einer Aktualisierung keine stetige Funktion in Abhängigkeit der Ausführungsreihenfolge. Des Weiteren ist eine unabhängige Betrachtung beim Einfügen mehrerer Aktualisierungen nicht möglich, da die Varianz je nach Position der Aktualisierungen zueinander sowohl erhöht als auch gemindert werden kann. Für das Dienstqualitätskriterium der Antwortzeitvarianz ist es somit nicht möglich, eine Zielfunktion aufzustellen, anhand derer das Einfügen von Aktualisierungen optimiert werden kann.
Aufgrund der oben dargelegten Eigenschaften der Dienstqualitätsmetriken werden im Folgenden nur noch die mittlere Antwortzeit und die gewichtete mittlere Antwortzeit betrachtet. Da beide Metriken durch die Anwendung desselben Ablaufplanschemas minimiert werden können (Shortest-Job-First), wird im allgemeinen Fall nur von der mittleren Antwortzeit die Rede sein.

4.1.2 Datenqualitätskriterien

Die Datenqualität beschreibt die Güte einer Information aus der Sichtweise der Prozesse oder der Nutzer, die diese Information verwenden (engl. fitness for use). Der Begriff Datenqualität ist komplex und vielschichtig und daher mehrdimensional definiert, wobei jede Dimension einen spezifischen Teilaspekt umfasst. In der Literatur existiert eine Vielzahl von Ansätzen, Qualitätsdimensionen zu klassifizieren, von denen hier nur die wichtigsten genannt werden: Jarke et al. [41], Wang et al. [86], Naumann et al. [56] und Francalanci et al. [27]. Naumann et al. [56] unterscheidet zum Beispiel die technische (Verfügbarkeit), inhaltsbezogene (Genauigkeit, Vollständigkeit), subjektive (Glaubwürdigkeitm Reputation) und instanzbezogene (Zugänglichkeit) Qualitätsdimension. Die in der Literatur am häufigsten genannten Datenqualitätskriterien sind in Abbildung 4.1 zusammengefasst dargestellt. Als besonders relevant können hier insbesondere die Genauigkeit (engl. accuracy), die Aktualität (engl. currency), die Vollständigkeit (engl. completeness) und die Konsistenz (engl. consistency) eingestuft werden.
Zur Optimierung eines Systems bezüglich einer Dimension ist zunächst eine Metrik zu entwickeln bzw. auszuwählen, anhand derer die Qualität messbar gemacht werden kann. Für einige Dimensionen, wie zum Beispiel die Konsistenz und die Interpretierbarkeit, sind nur qualitative Aussagen möglich. Die Güte anderer Dimensionen, wie Genauigkeit oder Aktualität, kann hingegen konkret quantifiziert werden. Die Erhebung der Werte der Datenqualitätsmetriken für relationale Datenbanksysteme im Allgemeinen wurde in [55] betrachtet und für Data-Warehouse-Systeme im Speziellen in [84].
In Abschnitt 3.5.1 wurde dargelegt, dass die Echtzeitfähigkeit eines Data-Warehouse-Systems dessen Fähigkeit bemisst, die Verzögerung der Dateneinbringung zu minimieren bzw. die Datenaktualität zu maximieren. Aus diesem Grund wird in den weiteren Ausführungen ausschließlich die Qualitätsdimension Aktualität fokussiert. Die anderen bereits genannten Datenqualitätsdimensionen spannen getrennte Forschungsgebiete auf und bedürfen daher einer gesonderten Betrachtung.

Datenaktualität

Die Bestimmung der Datenaktualität eines Anfrageergebnisses erfolgt anhand verschiedener Metriken, die in drei Klassen unterschieden werden können: (1) Abstandsbasierte Metriken (engl. lag-based metrics) bemessen die Aktualität einer Anfrage anhand der Anzahl der noch nicht eingebrachten Aktualisierungen. Dazu sind lediglich die Abhängigkeiten zwischen Anfragen und Aktualisierungen zu ermitteln. (2) Divergenzbasierte Ansätze betrachten die Wertdifferenz zwischen dem bereits persistierten Datum und dem Datum nach der Aktualisierung. Diese Differenz ist jedoch im strengen Sinne im Kontext der Anfragen zu sehen. So kann eine Aktualisierung unter Anwendung der divergenzbasierten Metrik als sehr wichtig eingestuft werden, hinsichtlich der Veränderung der Anfrageergebnisse aber kaum einen Einfluss haben. Da eine Betrachtung der Anfragesemantik im Einzelnen nicht durchführbar ist, können an dieser Stelle nur Annahmen getroffen werden. Ein weiterer Nachteil dieses Ansatzes besteht darin, dass zur Berechnung der Wertdifferenz das zu aktualisierende Datum ermittelt werden muss, was eine sehr teure Operation darstellt. (3) Zeitbasierte Aktualitätsmetriken messen die zeitliche Differenz zwischen der letzten Aktualisierung eines Datums und der ältesten noch nicht eingebrachten Aktualisierung. Auch für diesen Ansatz müssen wie für die abstandsbasierte Metrik lediglich die Abhängigkeiten zwischen Anfragen und Aktualisierungen ermittelt werden.
Angesichts ihrer einfachen Berechenbarkeit sowie ihrer klaren Semantik sollen im Folgenden die abstandsbasierte und die zeitbasierte Aktualitätsmetrik zur Bestimmung der Datenqualität verwendet werden.

4.1.3 Multikriterielle Optimierung

Die multikriterielle Optimierung bezeichnet eine Klasse von Optimierungsproblemen, bei denen mehrere in Konflikt stehende Zielsetzungen vorliegen. Ein typisches Beispiel ist die Suche nach Hotels, bei der sowohl der Preis als auch die Nähe zum Strand zu minimieren ist. Es ist leicht zu sehen, dass es sich dabei um zwei widerstreitende Kriterien handelt.
Formal lässt sich ein multikriterielles Minimierungsproblem wie folgt beschreiben:

$$min_x[\mu_1(x), \mu_2(x), ..., \mu_n(x)]$$

$$g_j(x) \leq 0, 1 \leq j \leq r$$

$$h_k(x) = 0, 1 \leq k \leq s$$

$$x_l \leq x_i \leq x_u.$$

Dabei bezeichnet μ_i die i-te Optimierungsfunktion, g und h sind die r Ungleichungs- bzw. die s Gleichheitsbedingungen für das Problem und x_l und x_u sind die Beschränkungen des Suchraums nach unten bzw. nach oben. Die Lösungen dieser Problemdefinition werden als Pareto-effizient bezeichnet. Sie sind dadurch gekennzeichnet, dass eine Verbesserung in einer Dimension nur durch eine Verschlechterung in einer anderen Dimension zu erreichen ist. Für multikriterielle Probleme gibt es daher nicht

nur ein optimales Ergebnis, sondern die Lösung besteht aus einer Menge von Pareto-effizienten Ergebnissen. Aufgrund der geltenden Dualität $max\mu_m(x) = -min\mu_m(x)$ kann jedes Minimierungsproblem in ein Maximierungsproblem transformiert werden und umgekehrt, so dass obiger Formalismus auch im Allgemeinen angewendet werden kann.
Multikriterielle Problemstellungen sind in vielen Anwendungsgebieten präsent, im Umfeld von Data-Warehouse-Systemen jedoch kaum untersucht. Eine der wenigen existierenden Arbeiten ist [68], in der ein sogennantes QoX-Framework für die qualitätsgetriebene Entwicklung von ETL-Prozessen vorgeschlagen wird. Es wurden verschiedenen Qualitätsdimensionen diskutiert und beispielhaft die Abhängigkeiten zwischen den Qualitätsdimensionen in verschiedenen Experimenten untersucht sowie die Konflikte zwischen diesen in Abhängigkeitsgraphen dargestellt. In dieser Dissertation wird ein konkreter Konflikt zwischen den Qualitätskriterien Antwortzeit und Datenaktualität herausgegriffen und es werden multikriterielle Optimierungsansätze erarbeitet.

Lösungsmethoden

Der einfachste und intuitivste Lösungsansatz für multikriterielle Probleme besteht aus der Überführung der Menge der Optimierungsfunktionen in eine einzige aggregierte Funktion. Im Allgemeinen besteht diese Funktion aus der gewichteten linearen Summe der Einzeloptimierungskriterien [37]. Die entstehende Gleichung kann sehr effizient mittels Methoden der nichtlinearen Progammierung gelöst werden. Der Nachteil dieser Ansätze besteht jedoch in der Notwendigkeit von A-priori-Wissen zur Festlegung der Gewichte. Ist dieses Wissen nicht vorhanden, kann die Zuweisung der Gewichte nur subjektiv erfolgen. Des Weiteren ist es mit diesem Ansatz unmöglich, alle Pareto-Punkte zu ermitteln. Andere Ansätze, wie Normal Boundary Intersection [16], Normal Constraint [54] und Successive Pareto Optimization [33], umgehen diese Nachteile, indem statt einer aggregierten Funktion mehrere aggregierte Funktionen erstellt werden, die jeweils unter verschiedenen Annahmen die Pareto-Lösungen liefern. Eine weitere sehr verbreitete Klasse von Lösungsmethoden für multikriterielle Probleme sind die evolutionären Algorithmen. Einen guten Überblick dieser Ansätze liefern Wiley & Sons in [18].
Für das vorliegende Problem, der Ermittlung von Ablaufplänen zur Maximierung der Datenaktualität und zur Minimierung der Antwortzeit, wurde eine spezifische Lösung entwickelt. Diese basiert auf der Beobachtung, dass nicht alle Pareto-Punkte gefunden werden müssen, sondern stets nur derjenige, der die Nutzeranforderung optimal erfüllt. Weiterhin sind nur zwei Qualitätsdimensionen relevant: die Antwortzeit als Dimension der Dienstqualität (siehe Abschnitt 4.1.1) und die Aktualität als Dimension der Datenqualität (siehe Abschnitt 4.1.2). Dadurch konnte die Fragestellung auf ein bekanntes Optimierungsproblem der Kombinatorik, dem Rucksackproblem, mit zusätzlichen Nebenbedingungen abgebildet werden (siehe Abschnitt 4.2.2).

4.1.4 Workload- und Systemmodell

Das zu betrachtende Szenario besteht aus drei Komponenten: 1) einem zentralen Data-Warehouse zur Verarbeitung von Nutzeranfragen und Aktualisierungen, 2) einem Arbeitsbereich, der kontinuierlich Aktualisierungen an das Data-Warehouse propagiert und 3) einer Data-Warehouse-Anwendung, in der nutzerdefinierte Anfragen erzeugt werden. Der Aufbau des Workloads, dessen Parameter, das Partitionierungsmodell sowie die konkreten Dienst- und Datenqualitätskriterien werden in den folgenden Abschnitten vorgestellt.

Workloadmodell

Der Workload W in einem Echtzeit-Data-Warehouse kann in zwei Klassen unterschieden werden: in den Workload W_q, der ausschließlich Anfragen $q_i \in W_q$ beinhaltet und in den Workload W_u, bestehend aus Aktualisierungen $u_j \in W_u$. Der Begriff Aktualisierung wird im Folgenden synonym für alle Änderungsoperationen wie Einfügen, Löschen und Aktualisieren verwendet. Gemischte Transaktionen, bestehend aus Anfragen und Aktualisierungen, können aufgrund der push-basierten und damit von den Anfragen unabhängigen Einbringung von Aktualisierungen nicht auftreten. Zur Unterstützung der Ablaufplanung bei der Optimierung der Datenaktualität und der Antwortzeiten sind eine Reihe von Parametern notwendig, die in einem Vorverarbeitungsschritt für jede Anfrage und jede Aktualisierung ermittelt werden (siehe Abbildung 4.3). Jede Anfrage ist mit den Anforderungen der jeweiligen Nutzer bezüglich der beiden Qualitätsdimensionen – Aktualität als Dimension der Datenqualität und Antwortzeit als Dimension der Dienstqualität – annotiert. Diese sind als Parameterpaar $<qos_{q_i}, qod_{q_i}>$, mit $qos_{q_i} \in [0,1]$ und $qos_{q_i} \in [0,1]$, darstellbar, wobei gilt, dass die Summe beider Parameter immer 1 ergibt ($qos_{q_i} + qod_{q_i} = 1$). Weist ein Nutzer qos_{q_i} einen hohen Wert zu, so votiert er damit für geringe Antwortzeit, wohingegen ein hoher Wert für qod_j für höhere Aktualitätsanforderungen steht. Auch wenn der Zusammenhang zwischen Dienst- und Datenqualität nicht linear ist, bietet dieses Parameterpaar dennoch eine gute Metapher zur Spezifikation der Nutzeranforderungen.
Zur Bewertung des Nutzwertes einer Aktualisierung für die Datenqualität eines Systems wird jede Aktualisierung um einen Profitparameter p_{u_j} erweitert. Die Belegung dieses Profitparameters ist abhängig von der gewählten Datenqualitätsmetrik, kann jedoch einfach bestimmt werden (siehe Abschnitt 4.1.2).
Durch die Kompilierung der Anfragen und Aktualisierungen können näherungsweise die Kosten bestimmt werden (siehe Abbildung 4.3). Der Kostenparameter einer Anfrage q_i wird mit c_{q_i} und der einer Aktualisierung u_j mit c_{u_j} bezeichnet. Die zur Schätzung der Kosten ermittelten Pläne können bei der eigentlichen Ausführung der Anfragen bzw. der Aktualisierungen wiederverwendet werden.
Reihenfolgeabhängigkeiten zwischen Aktualisierungen werden in diesem Workloadmodell nicht berücksichtigt, so dass die Ausführungsordnung beliebig veränderbar ist. Allerdings kann das Modell sehr einfach um diese Eigenschaft erweitert wer-

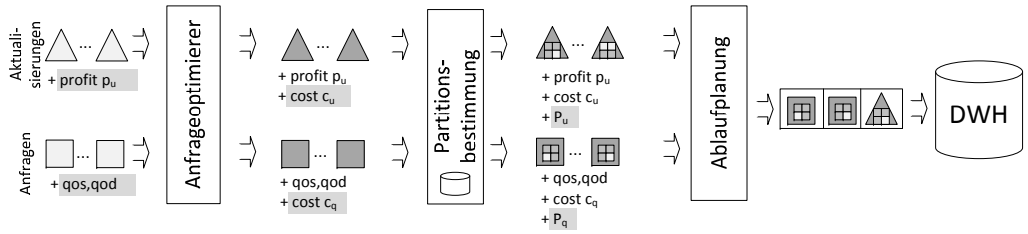


Abbildung 4.3: Workloadmodell

den. Dies hätte positive Auswirkungen hinsichtlich der Laufzeiteigenschaften des in Abschnitt 4.2 entwickelten Scheduling-Algorithmus, da durch den Erhalt von Abhängigkeitsbedingungen der Suchraum zusätzlich eingeschränkt wird.

Partitionierungsmodell

In Abschnitt 3.4.3 wurde die partitionierte Datenproduktion und -speicherung als Grundvoraussetzung für den Betrieb eines Echtzeit-Data-Warehouses eingeführt. Diese ermöglicht die Bestimmung des Wertes einer Aktualisierung, indem korrelierende Anfragen ermittelt werden können. Die Granularität der Partitionierung bestimmt dabei die Genauigkeit sowie die Kosten der Korrelationsbestimmung. Der Aufbau der Datenstruktur zur Partitionsbestimmung wird im Detail in Kapitel 6 betrachtet.
Jede Anfrage q_i und jede Aktualisierung u_j liest bzw. schreibt eine oder mehrere Partitionen ($|P_{q_i}| \geq 1$ und $|P_{u_j}| \geq 1$). Die Bestimmung der Menge der Partitionen kann durch einen effizienten Datenbankzugriff realisiert werden (siehe Abbildung 4.3). Eine Korrelation zwischen einer Anfrage und einer Aktualisierung liegt vor, wenn die Schnittmenge der beiden Partitionsmengen nicht leer ist ($P_{q_i} \cap P_{u_j} \neq \emptyset$).

Bestimmung der mittleren Antwortzeit

Die Antwortzeit einer Anfrage q_i setzt sich zusammen aus der eigenen Verarbeitungszeit e_{q_i} sowie der Wartezeit verursacht durch die Ausführung vorheriger Anfragen und Aktualisierungen. Während die Wartezeit, verursacht durch die Menge an Anfragen, iterativ berechnet werden kann, wird die durch die Aktualisierungen verursachte Wartezeit jeweils zu den Kosten der darauffolgenden Anfrage addiert. Daraus ergibt sich die mittlere Antwortzeit eines Workloads W durch folgende Berechnungsvorschrift:

$$MittlereAntwortzeit(W) = \frac{1}{|W_q|} \sum_{i=0..|W_q|} (|W_q| - i) \cdot (e_{q_i} + c_{u_j}). \qquad (4.1)$$

Bestimmung der Aktualität

Die Bestimmung der Aktualität einer Anfrage bzw. eines gesamten Workloads erfolgt unabhängig von der verwendeten Metrik (abstandsbasiert oder zeitbasiert) allein auf Basis des Profitparamters p_{u_j}. Dazu muss lediglich sichergestellt werden, dass der Profit durch eine positive Zahl ausdrückbar ist und ein höherer Profitwert mit einer verbesserten Datenqualität bzw. Datenaktualität korrespondiert. Wird eine Aktualisierung u_j vor einer Anfrage q_i ausgeführt und besteht eine Abhängigkeit zwischen diesen ($P_{q_i} \cap P_{u_j} \neq \emptyset$), so wird die Datenaktualität der Anfrage q_i um die Größe des Profitwertes p_{u_j} erhöht. Die mittlere Aktualität aller Anfragen eines Workloads ist dann wie folgt berechenbar:

$$Aktualitaet(W) = \frac{1}{|W_q|} \sum_{q_i \in W_q,\ u_j \in W_u,\ P_q \cap P_j \neq \emptyset} p_{u_j}. \tag{4.2}$$

Die Festlegung des Profitparameters soll am Beispiel der abstandsbasierten Metrik illustriert werden. Diese misst die Aktualität einer Anfrage anhand der Anzahl an noch nicht eingebrachten Aktualisierungen und stellt somit ein Minimierungsproblem dar. Durch Vergabe eines negativen Profitwertes für alle Aktualisierungen ($\forall u_j,\ p_j = -1$) kann das Minimierungsproblem in ein Maximierungsproblem überführt werden. Die Ausführung einer Aktualisierung vor einer korrelierenden Anfrage würde somit die Aktualität dieser Anfrage um 1 verbessern.

4.2 Multikriterielle Ablaufplanung

Das Problem der multikriteriellen Optimierung wurde in Abschnitt 4.1 bereits abstrakt dargelegt und die zwei Qualitätsdimensionen Aktualität und Antwortzeit eingeführt. Darauf aufbauend wird in den folgenden Abschnitten zunächst der Begriff Pareto-Effizienz für Ablaufpläne definiert und ein Algorithmus entwickelt, der sowohl im statischen als auch im dynamischen Fall den optimalen Ablaufplan ermittelt.

4.2.1 Pareto-effiziente Ablaufpläne

Die Lösungen eines multikriteriell definierten Problems wurden in Abschnitt 4.1.3 als Pareto-effizient bezeichnet. Ein einleitendes Beispiel veranschaulicht den Begriff der Pareto-Effizienz im Kontext der Ablaufplanoptimierung.

Einleitendes Beispiel

Der zu betrachtende Beispielworkload besteht aus sieben Transaktionen: fünf Anfragen q_1 bis q_5 und zwei Aktualisierungen u_1 und u_2. Der vollständige Lösungsraum für diesen Workload setzt sich aus $7! = 5.400$ möglichen Permutationen bzw. Ablaufplänen zusammen, die in Abbildung 4.4 dargestellt und bezüglich der Antwortzeit (x-Achse) und Aktualität (y-Achse) bewertet sind. Es sollen zunächst nur die optimalen Ablaufpläne hinsichtlich einer Dimension betrachtet werden: Die Minimierung der

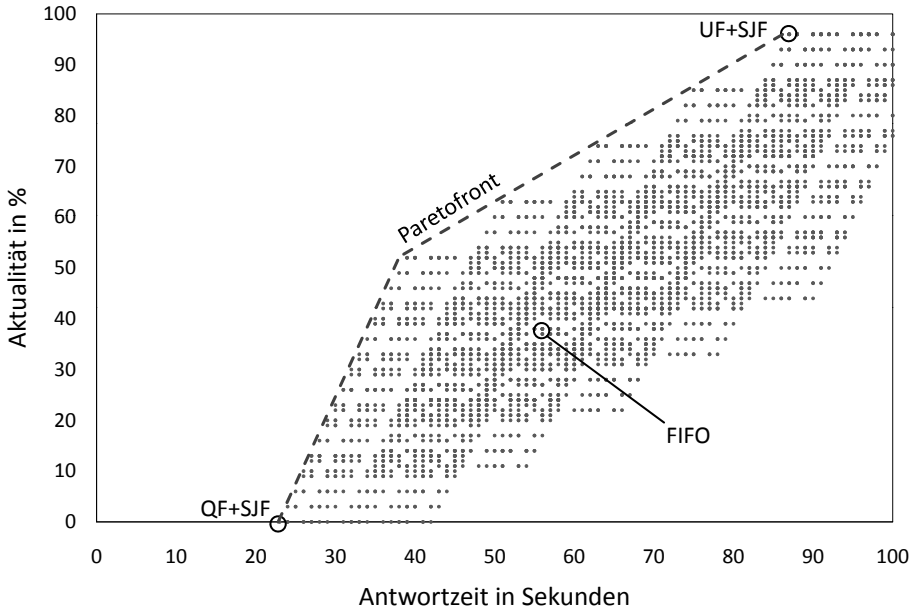


Abbildung 4.4: Alle 7! möglichen Ablaufpläne eines Workloads bestehend aus fünf Anfragen und zwei Aktualisierungen

Antwortzeit wird durch das Scheduling-Verfahren Shortest-Job-First (SJF) garantiert, welches eine absteigende Sortierung der Anfragen nach ihrer Verarbeitungszeit vornimmt. Da zusätzlich auch Aktualisierungen in den Ablaufplan integriert werden müssen, ergibt sich der optimale Ablaufplan durch Ausführung aller Anfragen vor allen Aktualisierungen (engl. Queries-First, QF) und der Anwendung von SJF. Die Maximierung der Aktualität wird durch die Priorisierung der Aktualisierungen vor den Anfragen erreicht. Werden die Anfragen zusätzlich wieder nach dem SJF-Schema geordnet, kann der in Abbildung 4.4 dargestellte Ablaufpan $UF + SJF$ ermittelt werden. Beide Ablaufpläne sind Pareto-effizient, da keine Verbesserung in einer Dimension möglich ist, ohne in der anderen Dimension ein schlechteres Ergebnis zu erzielen. Bei gleichzeitiger Betrachtung beider Optimierungsziele sind alle Ablaufpläne berechenbar, wobei besonders die Pareto-effizienten Ablaufpläne auf den verbindenden Linien zwischen $QF + SJF$ und $UF + SJF$ von Interesse sind. Diese werden als Pareto-Front bezeichnet. Ziel ist es, für gegebene Nutzeranforderungen einen Ablaufplan auf der Pareto-Front zu finden, der diesen Anforderungen am nächsten kommt. Beispielhaft ist noch ein weiterer Ablaufplan dargestellt, der nach dem FIFO-Prinzip geordnet ist, wodurch keine Optimalität gewährleistet wird.

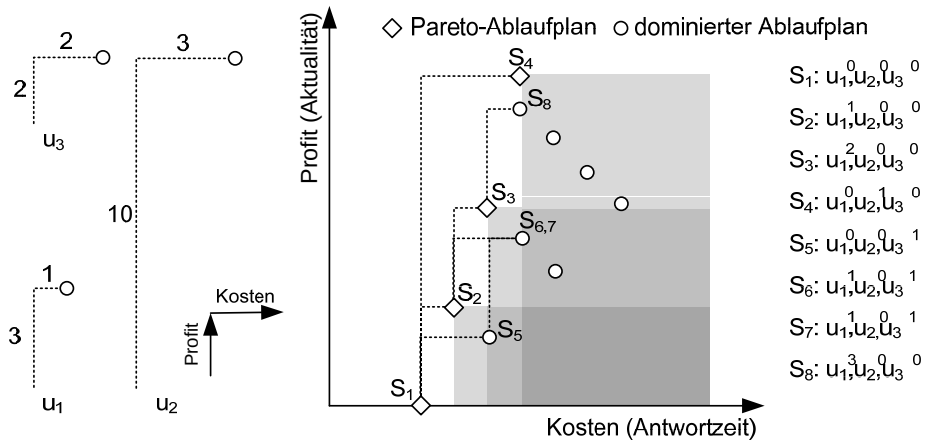


Abbildung 4.5: Pareto-effiziente Ablaufpläne des Workloads: $q_1, q_2, q_3, u_1, u_2, u_3$

Definitionen

In dem vorhergehenden Beispiel wurden die Ablaufpläne $QF + SJF$ und $UF + SJF$ als Pareto-effizient identifiziert. Im folgenden Abschnitt wird dieser Begriff im Kontext der Ablaufplanung definiert und eine Metapher zur Beschreibung von Aktualisierungen vorgestellt.
In Abbildung 4.5 sind drei Aktualisierungen unterschiedlicher Kosten und Profite dargestellt, welche in einen gegebenen Ablaufplan, bestehend aus drei Anfragen, einzufügen sind. Jede der drei Aktualisierungen kann durch zwei Vektoren repräsentiert werden: in x-Richtung durch einen Kostenvektor und in y-Richtung durch einen Profit-Vektor. Zur Veranschaulichung wird angenommen, dass alle Anfragen von allen Aktualisierungen profitieren. In späteren Abschnitten wird auch der allgemeinere Fall betrachtet. Die Reihenfolge der Anfrage sei durch ein beliebiges Scheduling-Verfahren oder durch Nutzerpriorisierungen [45, 66, 17, 78, 79] fest vorgegeben. Die Position einer Aktualisierung u_j^n in einem Ablaufplan S sei beschrieben durch n und initial mit dem Wert 0 besetzt, d.h. die Aktualisierung wird nach allen Anfragen ausgeführt. Der Positionswert n ist durch die Anzahl der Anfragen im Workload begrenzt, d.h. eine Aktualisierung kann maximal um die Anzahl der im Workload vorhandenen Anfragen verschoben werden. Die Menge aller möglichen Ablaufpläne – und damit der Lösungsraum für das Optimierungsproblem – ist dann wie folgt definiert:

Definition 4.1 (Lösungsmenge P) *P ist die Menge der Ablaufpläne, die sich durch die Verschiebung aller Aktualisierungen u_j^n um 0 bis maximal $|W_q|$ Positionen ergeben: $P = \{u_j^n | \forall j, 0 \leq n \leq |W_q|\}$.*

Der Ablaufplan S_1 in Abbildung 4.5 ist damit durch die Notation (u_1^0, u_2^0, u_3^0) beschreibbar. Zur Bestimmung Pareto-effizienter Ablaufpläne wird der Begriff der Dominanz benötigt, der wie folgt definiert ist:

Definition 4.2 (Dominanz $\triangleleft$) *Gilt für zwei Ablaufpläne S und S', dass S in mindestens einem Kriterium (Minimierung der Kosten, Maximierung des Profits) besser und in keinem Kriterium schlechter als S' ist, so wird S' von S dominiert: $S' \triangleleft S$, falls $c(S) \leq c(S')$, $p(S') \geq p(S)$ und $c(S) \neq c(S')$ oder $p(S) \neq p(S')$.*

Im Beispiel in Abbildung 4.5 wird S_5 von S_2 in beiden Kriterien und S_8 von S_4 bezüglich des Aktualitäts-Kriteriums dominiert. Aufbauend auf der Dominanz ist ein Pareto-effizienter Ablaufplan wie folgt definiert:

Definition 4.3 (Pareto-effizienter Ablaufplan) *Seien S und $S' \in P$, dann ist S Pareto-effizient, falls kein S' mit $S \triangleleft S'$ existiert. $P^* = \{S \in P | \nexists S' \in P, S \triangleleft S'\}$.*

Ein Pareto-effizienter Ablaufplan garantiert somit, dass eine Verbesserung in einem Kriterium nicht möglich ist, ohne dass sich das Ergebnis in einem anderen Kriterium verschlechtert. Im Beispiel ist dies für die Ablaufpläne S_1, S_2, S_3 und S_4 erfüllt.
Das Problem der Ablaufplanoptimierung unter Berücksichtigung zweier Kriterien gleichzeitig wurde bereits 1956 in [71] erstmalig betrachtet (mit den Optimierungszielen Minimierung der Job-Verarbeitungszeit und Minimierung der verspäteten Jobs). Jedoch sind keine Arbeiten zur Ablaufplanung von Aktualisierungen unter Berücksichtigung der Qualitätsdimensionen Antwortzeit und Aktualität bekannt.

4.2.2 Abbildung auf das Rucksackproblem

Das Problem der Auswahl von Aktualisierungen, die die Datenaktualität maximieren und gleichzeitig die dafür notwendige zusätzliche Antwortzeit minimieren, kann auf das 0-1-Rucksackproblem zurückgeführt werden (0-1 bedeutet, dass jedes Element bzw. jede Aktualisierung höchstens einmal in der Lösungsmenge vorkommen darf). Die Abbildung auf dieses Problem, das heisst die Erzeugung der Rucksackelemente sowie die Festlegung der Rucksackgröße und zusätzlicher Nebenbedingungen, ist Inhalt der folgenden Abschnitte.

Definition des Rucksackproblems

Die in den Rucksack einzufügenden Aktualisierungen werden jeweils durch einen Profitparameter p_j und einen Kostenparameter c_j beschrieben. Das 0-1-Rucksackproblem besteht darin, eine Menge von Aktualisierungen auszuwählen, so dass die Summe der Profite maximiert wird, ohne dass die Summe der Kosten eine festgelegte Rucksackgröße B überschreitet:

$$maximize \sum_{j \in |U|} p_j u_j \tag{4.3}$$

$$subject\ to \sum_{j \in |U|} c_j u_j \leq B \tag{4.4}$$

$$u_j \in \{0, 1\},\ j = 1, ..., |U|. \tag{4.5}$$

Da die mehrmalige Ausführung einer Aktualisierung semantisch nicht sinnvoll ist, gilt zusätzlich die Einschränkung, dass jede Aktualisierung nur einmal in der Lösungsmenge vorhanden sein darf (Gleichung (4.5)).

Bestimmung der Rucksackgröße

Zur Berechnung der Rucksackgröße B, d.h. der zur Verfügung stehenden Zeit zur Ausführung von Aktualisierungen, werden die minimale und die maximale Antwortzeit des gesamten Workloads benötigt. Die minimale Antwortzeit ergibt sich aus der Ausführung aller Anfragen vor allen Aktualisierungen, durch Anwendung der Query-First-Strategie QF (siehe Abbildung 4.6). Analog wird die maximale Ausführungszeit durch Anwendung der Update-First-Strategie UF erreicht, d.h. der Ausführung aller Akualisierungen vor allen Anfragen. Die Differenz beider Werte ergibt das Zeitfenster B_{Total}, welches notwendig ist, um alle Aktualisierungen zuerst auszuführen und damit garantiert die höchste Datenaktualität zu erreichen:

$$B_{Total}(W) = MittlereAntwortzeit(W_{UF}) - MittlereAntwortzeit(W_{QF}). \quad (4.6)$$

Die Größe des für die Aktualisierungen zur Verfügung stehenden Zeitfensters wird durch die Nutzer mittels der in Abschnitt 4.1.4 eingeführten Parameter $<qos_{q_i}, qod_{q_i}>$ vorgegeben. Dazu wird der qos_{q_i}-Parameter aller Anfragen q_i mit dem Zeitfenster B_{Total} aus Gleichung (4.6) multipliziert (siehe Abbildung 4.6), so dass sich die Rucksackgröße B wie folgt ergibt:

$$B(W) = \frac{B_T}{|W_q|} \sum_{q_i \in W_q} 1 - qos_{q_i}, \quad qos_{q_i} \in [0,1]. \quad (4.7)$$

Die Rucksackgröße B bildet somit die Beschränkung für das in den Formeln (4.3)-(4.5) formulierte Rucksackproblem. Ein hoher Wert für B bzw. eine schwache Beschränkung erlaubt das Einfügen vieler Aktualisierungen, wodurch die Datenqualität bzw. Datenaktualität verbessert wird. Ein kleines B führt dazu, dass nur wenige Aktualisierungen eingefügt werden können, wodurch die Dienstqualität verbessert und die Datenqualität verschlechtert wird.
Die in Formel (4.7) ermittelte Rucksackgröße B ist nur ein rechnerisch ermittelter Wert, was zur Folge hat, dass die Antwortzeit des damit errechneten Pareto-effizienten Ablaufplans von B abweichen bzw. darunter liegen kann. Dieser Zusammenhang ist in Abbildung 4.6 dargestellt: Die Rucksackgröße B liegt zwischen zwei Pareto-effizienten Ablaufplänen S_i und $S_i + 1$, wobei S_i die Lösung des Rucksackproblems darstellt. Die Differenz zwischen der vorgegebenen Rucksackgröße B und der Antwortzeit des Ablaufplans S_i hängt von der Dichte der Pareto-Front ab. Diese wird durch die Anzahl der Pareto-effizienten Pläne $|P^*|$ bestimmt und ist nach unten durch $n + 1$ und nach oben durch 2^n beschränkt. Erstgenannter Wert ergibt sich, wenn jeweils die Profite bzw. die Kosten aller Aktualisierungen identisch sind, d.h. $(p_{u_j} = p_{u_k} \wedge c_{u_j} = c_{u_k} \ \ \forall u_j, u_k \in W_u)$. Sind jeweils die Werte der Kosten gleich den Werten der Profite für jede Aktualisierung, d.h. $(p_{u_j} = c_{u_j} \ \ \forall u_j \in W_u)$, so ergibt sich

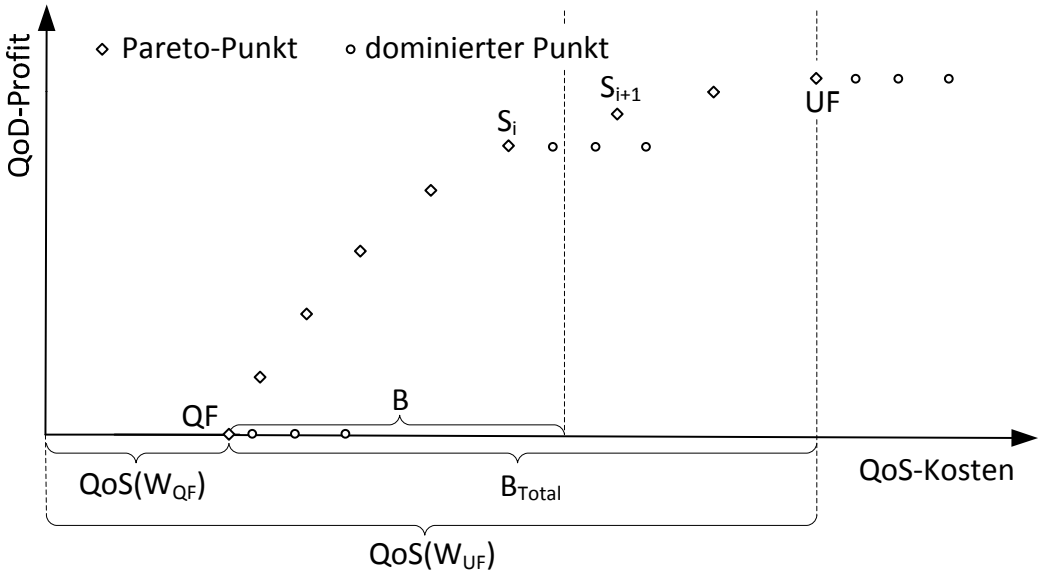


Abbildung 4.6: Ermittlung der Rucksackgröße B

die obere Schranke. In praktischen Szenarien liegt die Anzahl der Pareto-effizienten Lösungen zwischen diesen zwei Werten, ist aber in jedem Fall hoch genug, dass der ermittelte Ablaufplan sehr nah an der vorgegebenen Rucksackgröße liegt.

Erzeugung von Eingabelementen

Die Eingabeelemente des Standardrucksackproblems sind durch Kosten (bzw. Gewicht) und Profit bestimmt. Auch für Aktualisierungen sind Kosten- und Profitparameter bekannt, jedoch variieren diese aus Sicht der Anfragen in ihrer Position innerhalb eines Anfrageablaufplanes. Um die Kosten und Profite der Aktualisierungen an den jeweiligen Positionen zu bestimmen, wird eine Abhängigkeitsmatrix D der Größe $|Q| \times |U|$ angelegt, die angibt, welche Anfrage von welcher Aktualisierung profitiert (siehe Abbildung 4.7). Existiert eine solche Abhängigkeit, wird das entsprechende Matrixelement u_j^k mit dem Wert 1 versehen. Die Kosten und Profite einer Aktualisierung u_j^k können damit als Funktionen der Position k formuliert werden:

$$cost(u_j^k) = c_j \cdot k \tag{4.8}$$

$$profit(u_j^k) = p_j \cdot \sum_{i=1}^{k} u_j^i. \tag{4.9}$$

Bei der Verschiebung einer Aktualisierung von Position i auf Position $i+1$ wachsen die Kosten stets um einen festen Faktor, der Profit wächst jedoch nur, wenn eine Abhängigkeit mit Anfrage q_{i+1} besteht, d.h. das Matrixelement u_j^{i+1} gesetzt ist. Durch

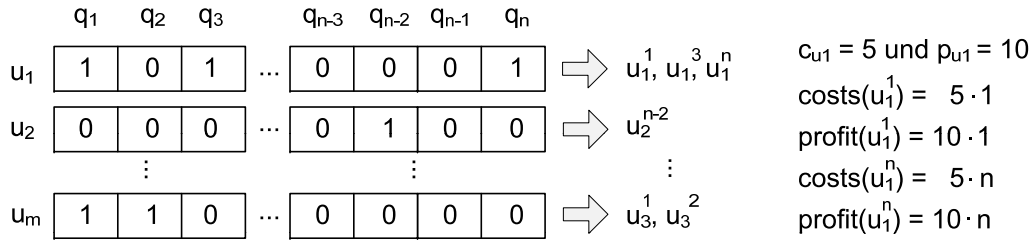


Abbildung 4.7: Beispiel einer Abhängigkeitsmatrix

Anwendung der Abhängigkeitsmatrix kann für jede Aktualisierung eine Menge von Aktualisierungselementen mit variierenden Profiten und Kosten bestimmt werden (siehe Abbildung 4.7). Dabei müssen nur die Aktualisierungselemente betrachtet und erzeugt werden, für die eine Abhängigkeit besteht. Aktualisierungselemente an einer Position k, an der keine Abhängigkeit vorhanden ist ($u_j^k = 0$), brauchen nicht betrachet werden, da sie von Aktualisierungselementen u_j^i ($i < k$) dominiert werden (siehe Definition 4.2). Die Anzahl der möglichen Aktualisierungselemente einer Aktualisierung u_j wird durch den Parameter N_j definiert. Um zu gewährleisten, dass sich von jeder Aktualisierung maximal ein Aktualisierungselement für die Lösungsmenge qualifiziert, wird das Rucksackproblem (Gleichungen 4.3-4.5) um folgende Bedingung erweitert:

$$\sum_{i=0}^{N_j} u_j^i \leq 1. \tag{4.10}$$

Diese Problemstellung ist eine Variation des Multiple-Choice-Rucksackproblems (engl. Multiple-Choice Knapsack Problem, MCKP) [57], dessen Bedingung darin besteht, genau ein Element jeder Klasse (Aktualisierung) für die Lösungsmenge zu ermitteln. Die Anzahl der Aktualisierungselemente ist zum einen durch die Anzahl der Anfragen und Aktualisierungen und zum anderen durch die Dichte der Abhängigkeitsmatrix bestimmt. Bei einer Abhängigkeitsdichte von 100% wäre die Anzahl der Aktualisierungselemente gleich $|Q| \cdot |U|$. In praktischen Szenarien ist jedoch von einer weitaus geringeren Abhängigkeitsdichte auszugehen, so dass die Anzahl der Aktualisierungselemente entsprechend verringert wird.

4.2.3 Lösung mittels dynamischer Programmierung

Das Rucksackproblem gehört zur Liste der 21 klassischen NP-vollständigen Probleme, die 1972 von Richard Karp aufgestellt wurde [44]. Jedoch existieren Algorithmen, die nicht das gesamte Problem, sondern nur Unterprobleme lösen und die nach Garey and Johnson [29], „[..] will display 'exponential behavior' only when confronted with instances containing 'exponentially large' numbers.“ Diese Art der Algorithmen werden als pseudopolynomiell bezeichnet. Die Laufzeit pseudopolynomieller Algorithmen ist durch ein Polynom in Abhängigkeit der Eingabelänge und

der größten in der Kodierung vorkommenden Zahl beschränkt. Ist diese Zahl relativ klein, sind diese Algorithmen sehr effizient lösbar, wie auch die Experimente in 4.5.3 bestätigen.
Ein pseudopolynomieller Algorithmus für das 0-1-Rucksackproblem, basierend auf den Ideen dynamischer Programmierung, wurde in [82] entwickelt und wird unter Berücksichtigung der Einschränkung in Gleichung (4.6) erweitert (*MultiOptScheduling*-Algorithmus 1). Die Eingabe besteht aus der Anzahl der zu bearbeitenden Elemente N (die Anzahl der Aktualisierungselemente), der Rucksackgröße B (das zur Verfügung stehende Zeitfenster für Aktualisierungen) sowie den Profit- und Kostenvektoren der Aktualisierungselemente. Zusätzlich sind alle Aktualisierungselemente u_j^i innerhalb ihrer Klasse nach dem Profit aufsteigend sortiert und der Rang innerhalb der Klasse ist in einem Array $classpos[N+1]$ gespeichert. Weiterhin wird angenommen, dass alle Eingabewerte sowie alle Zwischenergebnisse ganzzahlig sind.
In der dynamischen Programmierung werden zunächst für einfachere Teilprobleme Zwischenlösungen ermittelt, dann für auf diesen aufbauenden immer komplexer werdende Probleme und schließlich für das gesamte Problem selbst. Zur Speicherung der Zwischenlösungen wird eine Matrix der Größe $N+1 \times B+1$ P angelegt, deren Elemente mit 0 initialisiert werden. Für alle Variablen $1 \leq n \leq N$ und $1 \leq c \leq B$ speichert $P[n][c]$ die optimale Lösung des entsprechenden Teil-Rucksackproblems. Dabei gilt: sind die Kosten des n-ten Aktualisierungselements kleiner gleich der Kostenschranke des betrachteten Teilproblems (Zeile 3), so trägt dieses Element potenziell zur Teillösung bei, ansonsten kann dies übersprungen werden. Gilt Ersteres, so berechnet sich der Profit dieses Matrixelements mittels $P[n][c] = profit[n] + P[n - classpos[n]][c - cost[n]]$ (Zeile 4), d.h. aus dem aktuell betrachteten Element und dem Element der Vorgängerklasse. Durch einen Vergleich mit dem Profit des Vorgängerelements ($P[n][c] = P[n-1][c]$) wird die bessere Lösung mittels $P[n][c] = max(P[n-1][c], p)$ errechnet (Zeile 6). Eine zweite Matrix R, ebenfalls der Größe $N+1 \times B+1$, speichert, ob die einzelnen Elemente Bestandteil einer Teillösung waren oder nicht (Zeile 7).
Nachdem beide Iterationen abgeschlossen sind, wird die Gesamtlösung mittels Rückverfolgung aus den Teillösungen in absteigender Komplexität berechnet. Dazu wird, beginnend mit dem letzten Element $R[N+1][B+1]$, jeweils die Zugehörigkeit zu einer Teillösung überprüft und so die Ergebnismenge sukzessiv aufgebaut (Zeilen 10 bis 20). Der Rücksprung zum letzten Element der Vorgängerklasse ($n - classpos[n]$ in Zeile 4 und 16) ist wichtig, um die Bedingung in Gleichung (4.6) zu garantieren, dass die Lösung nur ein Aktualisierungselement jeder Aktualisierung enthält. Das Ergebnis des Algorithmus sind diejenigen Aktualisierungselemente, welche die Aktualität maximieren ohne die Kostenschranke zu verletzen. Der so entstandene Ablaufplan ist Pareto-effizient, da die Aktualität nicht weiter verbessert werden kann, ohne die Kostenschranke zu verletzen. Ist für eine Aktualisierung u_j kein Aktualisierungselement u_j^k in der Ergebnismenge vertreten und die Lösung daher unvollständig, so wird das Element u_j^0 der Lösungsmenge hinzugefügt. Diese Aktualisierungen werden nach allen Anfragen ausgeführt und verursachen damit weder Kosten noch erhöhen

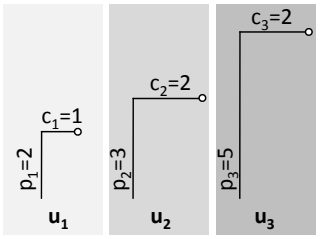
Algorithmus 1 MultiOptScheduling(N, B, profit[N], cost[N])

Benötigt: $P[N+1][B+1]$ // mit 0 initialisieren
$R[N+1][B+1]$ // mit 0 initialisieren
$classpos[N+1]$ // Position der Akt.-Elements in seiner Klasse
$result[N]$ // mit Falsch initialisieren

1: **for** $n = 1$ to N **do** // für jedes Aktualisierungselement
2: **for** $c = 1$ to B **do** // für jeden Schrankenwert $c \leq B$
3: **if** $cost[n] \leq c$ **then** // wenn Elementkosten kleiner sind
4: // der Profit des aktuellen Akt.-Elements und der vorhergehenden Klasse
5: $p = profit[n] + P[n - classpos[n]][c - cost[n]]$
6: **end if**
7: $P[n][c] = max(P[n-1][c], p)$ // die Alternative mit dem höchsten Profit wählen
8: $R[n][c] = (p > P[n-1][c])$ // Akt.-Elementals Element einer Teillösung markieren
9: **end for**
10: **end for**
11: // Berechnung der Lösungsmenge aus den Teillösungen
12: $c = B,\ n = N$ // die Zähler mit der Größe der Matrix initialisieren
13: **while** $n > 0$ **do** // vom letzten zum ersten Element
14: **if** $R[n][c]$ **then** // wenn Element in Teillösung enthalten
15: $result[n] = true$ // das Akt.-Element in der finalen Lösungsmenge speichern
16: $c = c - cost[n]$ // c um die Kosten des Aktualisierungselements reduzieren
17: $n = n - classpos[n]$ // zum letzten Akt.-Element der vorhergehenden Klasse
18: **else**
19: $n = n - 1$ // zum vorhergehenden Akt.-Element springen
20: **end if**
21: **end while**

sie den Profit. Die Eigenschaft der Pareto-Effizienz wird somit nicht verletzt.
Die Laufzeit- und Speicherkomplexität des Algorithmus ist durch die Parameter N und B gegeben und beträgt $O(N \cdot B)$ bzw. $\Omega(N \cdot B)$. Ob der Algorithmus effizient gelöst werden kann, hängt daher maßgeblich von der Größe der beiden Paramter ab: N ist durch die Anzahl der sich gleichzeitig im System befindlichen Anfragen und Aktualisierungen und deren Abhängigkeitsdichte stark nach oben begrenzt, was auch in den Experimenten in Abschnitt 4.5.3 bestätigt wird. Die Größe des Parameters B ist durch die Größe des Zeitfensters, aber vor allem durch die Genauigkeit der verwendeten Zeiteinheiten bestimmt. Ist eine Genauigkeit im Bereich einiger Millisekunden gefordert, so wird B entsprechend groß. Da die Kosten der Anfragen und Aktualisierungen jedoch auf geschätzten Werten des Optimierers beruhen und daher mit Ungenauigkeit behaftet sind, kann die Zeiteinheit dahingehend gröber gewählt werden.

Beispiel Die Funktionsweise des *MultiOptScheduling*-Algorithmus wird an einem Beispielworkload, dargestellt in Abbildung 4.8a, illustriert. Dieser besteht aus drei Aktualisierungen u_1, u_2 und u_3 sowie zwei Anfragen q_1 and q_2, deren Abhängigkeiten in der Matrix D in Abbildung 4.8b dargestellt sind. Ausgenommen des Paares q_1 und u_3 sind alle Anfragen von allen Aktualisierungen abhängig, so dass sich



(a) Drei Aktualisierungen u_1, u_2 und u_3 als Kosten-Profit-Vektor

	q_2	q_1	Aktualisierungs-elemente
u_1	1	1	u_1^1, u_1^2
u_2	1	1	u_2^1, u_2^2
u_3	1	0	u_3^2

(b) Abhängigkeitsmatrix D

	Kosten	Profit	classPos	n
u_1^1	1	2	1	1
u_1^2	2	4	2	2
u_2^1	2	3	1	3
u_2^2	4	6	2	4
u_3^2	4	5	1	5

(c) Datenstruktur der Aktualisierungselemente

n\c	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	**2**	**2**	**2**	**2**	**2**
2	0	2	**4**	**4**	**4**	**4**
3	0	2	4	**5**	**7**	**7**
4	0	2	4	5	7	**8**
5	0	2	4	5	7	8

(d) Lösungsmatrix P: graue Kreise markieren den Pfad der Rückverfolgung, schwarze Kreise die Lösungselemente

Abbildung 4.8: Beispiel für den *MultiOptScheduling*-Algorithmus

fünf Aktualisierungselemente ($N = 5$) aus der Abhängigkeitsmatrix ableiten lassen, die in der rechten Spalte in Abbildung 4.8b skizziert sind. Die Datenstruktur der Aktualisierungselemente ist in Abbildung 4.8c dargestellt: Zur Berechnung der Kosten und Profite der Aktualisierungselemente werden die Funktionen (4.8) bzw. (4.9) verwendet. Weiterhin wird jedes Aktualisierungselement innerhalb der Klasse aufsteigend sortiert und der Rang in der Spalte *classPos* gespeichert. Klassenübergreifend werden alle Aktualisierungselemente in Spalte n aufsteigend nummeriert. Die resultierende Datenstruktur in Abbildung 4.8c bildet die Eingabe für den *MultiOptScheduling*-Algorithmus.
Die Teillösungen werden in einer Matrix P der Größe $N + 1 \times B + 1$, dargestellt in Abbildung 4.8d, gespeichert. Der Parameter B sei in diesem Beispiel mit dem Wert 5 belegt. In Fettschrift dargestellte Matrixelemente kennzeichnen Teillösungen (korrespondierend zur Matrix R in Algorithmus 1). Die jeweils erste Zeile und Spalte der Lösungmatrix ist mit 0 initialisiert und wird für den Vergleich mit dem Vorgängerelement benötigt.
Der Algorithmus beginnt mit dem Aktualisierungselement u_1^1 ($costs(u_1^1) = 1$), das in jeden Rucksack der Größe 1 bis 5 hineinpasst und daher als Teillösung in jeder Spalte markiert wird. Das nächste Aktualisierungselement u_1^2 kann in jeden Rucksack der

Größe 2 und höher aufgenommen werden. Da der Profit von u_1^2 höher als der Profit des Vorgängerelements u_1^1 ist $(4 > 2)$, wird dieses Bestandteil der Teillösung in den Spalten 2 bis 5. Das Element u_2^1 der Aktualisierung u_2 bildet für die Rucksackgröße 3 eine Teillösung zusammen mit dem Element u_1^1 und für $c = 4$ und $c = 5$ zusammen mit u_1^2. Für das nächste Aktualisierungselement u_2^2, mit Kosten in Höhe von 4, gilt dies für $c = 5$ zusammen mit u_1^1 der Vorgängerklasse. Die Rücksprungvariable $n - classpos[n]$ in Zeile 4 in Algorithmus 1 stellt sicher, dass mehrelementige Teillösungen stets nur aus Aktualisierungselementen verschiedener Klassen gebildet werden. Das letzte zu bearbeitende Element u_3^2 ist in keiner Teillösung repräsentiert. In einem Rückverfolgungsverfahren, beginnend bei Element u_3^2, wird die Gesamtlösung zusammengesetzt. Da dieses Element nicht als Teillösung markiert ist, wird das Vorgängerelement u_2^2 ($n = n - 1$ der gleichen Spalte ($c = 5$)) betrachtet. Dies ist als Teillösung markiert und wird daher der Ergebnismenge hinzugefügt. Die nächste zu betrachtende Spalte wird aus der Differenz der aktuellen Spalte und den Kosten von u_2^2 ermittelt, d.h. $5 - costs(u_2^2) = 5 - 4 = 1$). Die nächste Zeilennummer in der Lösungsmatrix ist $n = n - classpos[n] = 5 - 2 = 2$. Das Element u_1^2 ist auch nicht in einer Teillösung enthalten, so dass im nächsten Schritt u_1^1 gefunden wird und der Algorithmus beendet ist.
Zur Vervollständigung der Lösung wird das Element u_3^0 dem Ergebnis hinzugefügt, so dass sich folgender Pareto-effizienter Ablaufplan für dieses Beispiel ergibt: $S = \{u_1^1, u_2^2, u_0^3\}$.

Gesamtprozess der Ablaufplanung

Die für die Bestimmung Pareto-effizienter Ablaufpläne notwendigen Verarbeitungsschritte sind in Abbildung 4.9 zu einem vollständigen Prozess zusammengefügt. Die beiden kontinuierlichen Anfrage- bzw. Aktualisierungsströme werden zunächst unabhängig voneinander prozessiert. In Schritt (1) wird der optimale Ablaufplan für die Anfragen (SJF), unter Verwendung des Dienstqualitätkriteriums Antwortzeit, bestimmt. Im nächsten Schritt wird durch Anwendung der Abhängigkeitsmatrix D die Menge der Aktualisierunselemente ermittelt. Parallel dazu wird in Schritt (3) die Rucksackgröße B aus den Nutzeranforderungen der Anfragen berechnet. In Schritt (4) werden die optimalen Positionen der Aktualisierungselemente, unter Einhaltung der Schranke B, bestimmt und im letzten Schritt wird der finale Ablaufplan erstellt.

4.3 Dynamische Ablaufplanung zur Laufzeit

Die meisten Publikationen behandeln ausschließlich die Berechnung von Pareto-effizienten Lösungen für das statische Ablaufplanungsproblem, bei dem alle Jobs und ihre Verarbeitungsinformationen a priori bekannt sind. Eine empfehlenswerte Übersicht bietet [81]. Nach bestem Kenntnisstand existiert keine Arbeit zur Pareto-effizienten Ablaufplanung in dynamischen Szenarien.

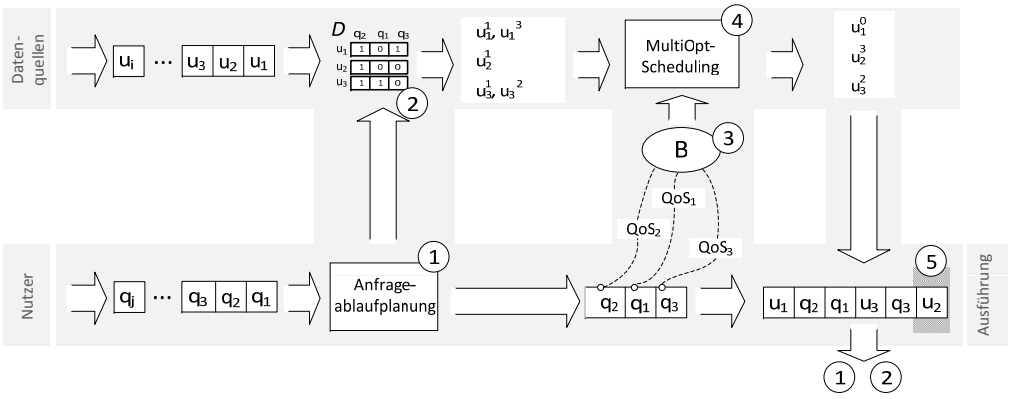


Abbildung 4.9: Gesamtprozess der Ablaufplanung

Im Gegensatz zu der bisher betrachteten statischen Ablaufplanung, ist im dynamischen Fall die Menge der Anfragen und Aktualisierungen nicht a priori bekannt. Stattdessen werden diese kontinuierlich dem Scheduler bzw. dem Data-Warehouse zugeführt. Dies hat zur Folge, dass ein Pareto-effizienter Ablaufplan S_1 vom Zeitpunkt t_1 zu einem späteren Zeitpunkt t_2, nachdem neue Anfragen und Aktualisierungen hinzugefügt wurden, sehr wahrscheinlich von einem anderen Ablaufplan S_2 dominiert wird (siehe Definition 4.2). Daher kann in der dynamischen Ablaufplanung von Pareto-Effizienz nur zu einem konkreten Ausführungszeitpunkt gesprochen werden. Dynamische Ereignisse sind unter anderem das Eintreffen neuer Anfragen und Aktualisierungen sowie deren Ausführung. Nach jedem Eintreten eines solchen Ereignisses muss der Ablaufplan bis zu einem gewissen Grad neu berechnet werden. Bei dieser Neuberechnung sind vier Fälle zu unterscheiden, die jeweils unterschiedlichen Berechnungsaufwand benötigen:

1. Ausführung einer Anfrage q_i: Die Ausführung einer Anfrage macht die Neuberechnung der Abhängigkeitsmatrix D notwendig (Schritt (2) in Abbildung 4.9), falls Aktualisierungen existieren, für die gilt $\exists u_i^j = 1$, für $1 \leq j \leq |U|$. Da sich der QoS-Wert durch Ausführung der Anfrage q_i ändert, müssen auch die Schranke B (Schritt (3)) und demzufolge auch die Aktualisierungselemente neu berechnet werden (Schritt (4)).

2. Ausführung einer Aktualisierung u_j: Bei Ausführung einer Aktualisierung u_j müssen die betreffenden Einträge u_j^i, für $1 \leq i \leq |Q|$, der Abhängigkeitsmatrix D gelöscht und in Konsequenz die Aktualisierungselemente neu berechnet werden.

3. Hinzufügen einer Anfrage q_i: Eine neue Anfrage q_i resultiert in der Neuberechnung der Abhängigkeitsmatrix D, falls $\exists u_i^j$, für $1 \leq j \leq |U|$. Zusätzlich muss die Schranke B aktualisiert und der Ablaufplan neu berechnet werden.

4. Hinzufügen einer Aktualisierung u_j: Nach dem Hinzufügen einer neuen Aktualisierung u_j muss die Abhängigkeitsmatrix D aktualisiert werden, und falls $\exists u_j^i$ für $1 \leq i \leq |Q|$ gilt, muss auch der Ablaufplan neu berechnet werden. Andernfalls ist der vorhergehende Ablaufplan immer noch Pareto-effizient und kann daher wiederverwendet werden.

In einigen der oben beschriebenen Fälle ist es möglich, zu einem Zeitpunkt i den vorhergehenden Ablaufplan S_{i-1} bzw. die vorhergehende Matrix P_{i-1} wiederzuverwenden und damit den Gesamtaufwand der Neuberechnung zu reduzieren: Bei der Ausführung einer Aktualisierung in Fall (2) werden Aktualisierungselemente bzw. Zeilen in Matrix P_{i-1} entfernt. Wenn keines dieser gelöschten Aktualisierungselemente u_j^k Teil einer partiellen Lösung war, d.h. $R[j][c] = false \ \ \forall c, \ k$, kann der vorhergehende Ablaufplan S_{i-1} wiederverwendet werden. Durch das Hinzufügen neuer Aktualisierungen (Fall (4)) ist die Matrix P um neue Aktualisierungselemente zu ergänzen. Der Algorithmus zur Ermittlung des Pareto-effizienten Ablaufplans muss dann nur noch auf diese neuen Einträge ΔP angewendet werden. Für die Anfragen betreffenden Fälle (1) und (3) ist stets eine Neuberechnung der Matrix P notwendig, da durch das Hinzukommen bzw. Wegfallen von QoS-Werten die Schranke B geändert wird.

Stabilitätsmaß Bisher wurde gezeigt, dass die Berechenbarkeit der Pareto-effizienten Ablaufpläne auch im dynamischen Fall möglich ist. Dabei ist zu erwarten, dass – in Abhängigkeit von der Anzahl neuer Ereignisse – sich aufeinanderfolgende Ablaufpläne verschieden stark voneinander unterscheiden. In dynamischen Systemen wird die Stärke des Unterschieds zwischen einer Lösung zum Zeitpunkt t_1 und einer Lösung zu einem späteren Zeitpunkt t_2 als Änderungsstabilität bezeichnet [10]. Ist die Änderungsstabilität zweier Lösungen bzw. Ablaufpläne besonders gering, so deutet dies darauf hin, dass die jeweiligen Probleminstanzen gar nicht oder nur schwach miteinander korrelieren. Um zwei Ablaufpläne miteinander vergleich zu können, soll basierend auf Definition 3.1 die folgende Distanzfunktion eingeführt werden:

$$d(S_1, S_2) = \frac{1}{|S_1 \cap S_2|} \sum_{u_i^{p_1} \in S_1, u_j^{p_2} \in S_2, i=j} |p_1 - p_2|.$$

Die Distanz ist somit als Mittelwert der Differenzen aller Aktualisierungspositionen definiert, die in den beiden zu vergleichenden Ablaufplänen vorkommen. Die Distanz $d(S_1, S_2)$ zwischen zwei aufeinanderfolgenden Ablaufplänen $S_1 = (u_1^6, u_2^2, u_3^4)$ und $S_2 = (u_2^1, u_3^5)$, nach der Ausführung von u_1, beträgt somit $(|2-1|+|4-5|)/2 = 1$. Eine geringe Distanz steht demzufolge für eine hohe Änderungsstabilität und umgekehrt.
Eine Analyse der Einflussfaktoren auf die Änderungsstabilität ist Inhalt des Abschnitts 4.5.4.

4.4 Selektionsbasierte Ausnahmebehandlung

Die Ermittlung der Abhängigkeiten zwischen Anfragen und Aktualisierungen basiert auf der Annahme, dass die Daten in partitionierter Form vorliegen und dass diese Partitionen stets von jeder Transaktion adressiert werden. Eine Anfrage ist abhängig von einer Aktualisierung, wenn beide eine gemeinsame Menge an Partitionen selektieren, d.h. $P_{q_i} \cap P_{u_j} \neq \emptyset$ (siehe Abschnitt 4.1.2). Die Wahl der Partitionierungsattribute bestimmt dabei die Granularität der entstehenden Partitionen und beeinflusst somit die Genauigkeit der ermittelten Abhängigkeiten bzw. Korrelationen. Da eine Partition mehrere hundert oder gar tausende Tupel repräsentiert, ist die partitionsbasierte Bestimmung der Korrelationen nur als näherungsweises Verfahren zu bezeichnen. Dies führt zu folgenden Problemen: 1) Wenn eine Abhängigkeit zwischen einer Anfrage q_i und einer Aktualisierung u_j fälschlicherweise ermittelt wurde und dies zu einer Priorisierung dieser Aktualisierung u_j führt, so wird dadurch die Anfrage q_i unnötig verzögert und damit die Datenlatenz erhöht. 2) Des Weiteren werden durch fälschlicherweise positiv ermittelte Abhängigkeiten andere Aktualisierungen niedriger priorisiert und damit wird die Datenaktualität gemindert. Ein analytisches Verfahren zur Bewertung der Genauigkeit des partitionsbasierten Ansatzes sowie eine darauf basierende Ausnahmebehandlung werden in diesem Abschnitt diskutiert.

Korrelationswahrscheinlichkeit Die Prädikate in den WHERE-Klauseln einer Anfrage und einer Aktualisierung können, wie in Abbildung 4.10 dargestellt, in zwei Typen unterschieden werden: zum einen in Prädikate, die Attribute des Partitionierungsschemas PS enthalten und zum anderen in Prädikate, die den Datenraum innerhalb der Partition weiter einschränken. Die in Letzteren vorkommenden Attribute werden als Residuum R bezeichnet, wobei gilt, dass $PS \cap R = \emptyset$. Die bisher nur einstufige Betrachtung der Übereinstimmung auf Ebene der selektierten Partitionen (Attributmenge PS) kann damit um eine zweite Stufe erweitert werden (Attributmenge R). Dazu wird die Selektivität der Prädikate, die die Attributmenge R enthalten, betrachtet. Die Wahrscheinlichkeit der Überlappung zwischen einer Anfrage q mit der Selektivität s_q und einer Aktualisierung u mit der Selektivität s_u ist durch Anwendung der hypergeometrischen Verteilung ermittelbar. Diese gibt die Wahrscheinlichkeit dafür an, dass eine bestimmte Anzahl von Elementen, die eine definierte Eigenschaft erfüllen, aus einer Stichprobe der Größe n gezogen werden. Die Elemente der Stichprobe werden dabei aus einer Basispopulation ohne Zurücklegen gezogen. Die hypergeometrische Verteilung wird mit den Parametern N (Grundgesamtheit), M (Anzahl der Elemente, die eine bestimmte Eigenschaft erfüllen), n (Anzahl der Elemente in der Stichprobe) und k (Anzahl der Elemente mit der zu prüfenden Eigenschaft) beschrieben und errechnet sich wie folgt:

$$h(k; N, M, n) = \binom{M}{k}\binom{N-M}{n-k} \Big/ \binom{N}{n}.$$

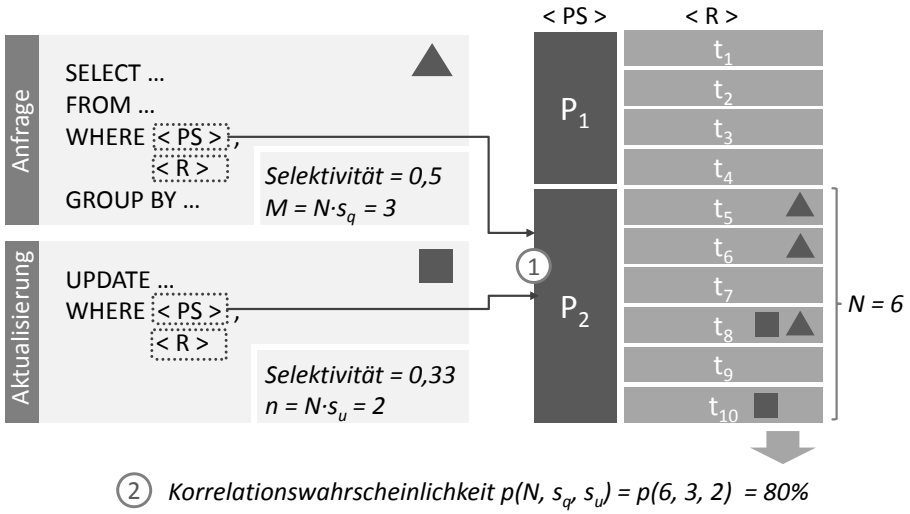
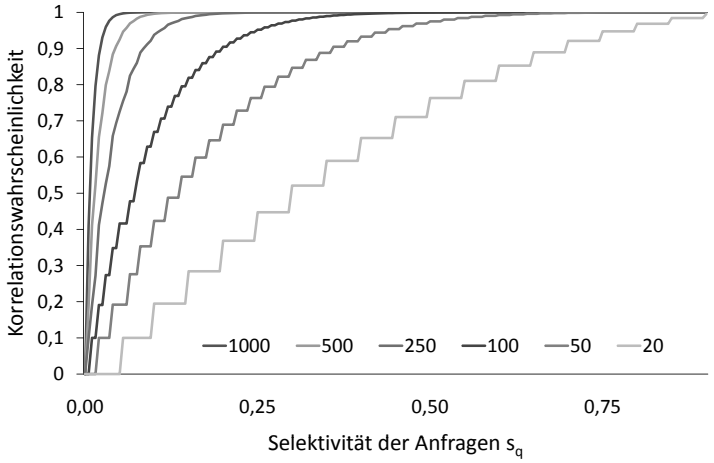


Abbildung 4.10: Schematische Übersicht der selektionsbasierten Ausnahmebehandlung

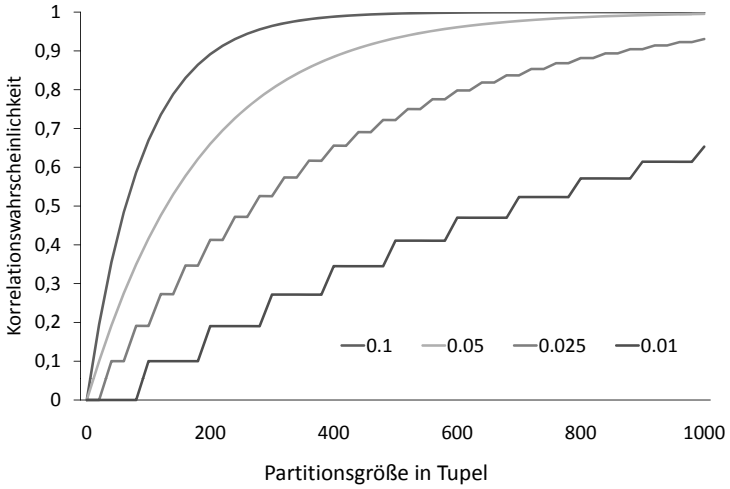
Die Belegung der Parameter für den vorliegenden Anwendungsfall ergibt sich wie folgt: Die Anzahl der Elemente der Grundgesamtheit N ist durch die Kardinalität der betrachteten Partition gegeben. Aus diesem Parameter kann, zusammen mit der Selektivität der Anfrage s_q, M mit $s_q \cdot N$ berechnet werden. Die Größe der Stichprobe n wird anhand der Selektivität der Aktualisierung s_u durch $s_u \cdot N$ ermittelt. Zur Vereinfachung des Problems wird des Weiteren der Fall betrachtet, dass keine Elemente in der Stichprobe gefunden werden können, d.h. $k = 0$. Daraus ergibt sich die Berechnungsvorschrift der Korrelationswahrscheinlichkeit $p(N, s_q, s_u)$ für eine Anfrage q und eine Aktualisierung u bei gegebener Partition der Größe N:

$$p(N, s_q, s_u) = \begin{cases} 1 - \binom{N - s_q \cdot N}{s_u \cdot N} \Big/ \binom{N}{s_u \cdot N} & , s_q + s_u < 1 \\ 1 & , \text{anderenfalls.} \end{cases}$$

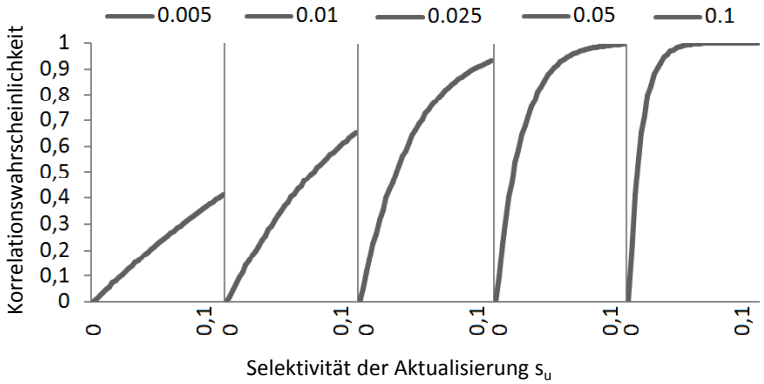
Für den Fall, dass die Summe der Selektivitäten größer als 1 ist, ergibt sich eine Korrelationswahrscheinlichkeit von 100%.
Das Beispiel in Abbildung 4.10 illustriert die Zusammenhänge: Gegeben sind eine Anfrage mit der Selektivität $s_q = 0,5$, eine Aktualisierung der Kardinalität $s_q = 0,33$ sowie ein Datensatz bestehend aus zehn Tupeln und unterteilt in zwei Partitionen P_1 und P_2. Auf Ebene der Partitionen kann eine Korrelation festgestellt werden, da sowohl die Anfragen als auch die Aktualisierung dieselbe Partition P_2 adressieren. Die tatsächliche Korrelation ist durch zwei Symbole innerhalb der einzelnen Tupel hervorgehoben. Für dieses Beispiel ist eine Überlappung in Tupel t_8 erkennbar. Die Wahrscheinlichkeit solch einer Überlappung kann mit Hilfe der oben beschriebenen



(a) Steigende Anfrageselektivität s_q, feste Aktualisierungsselektivität $s_u = 0,1$, verschiedene Partitionsgrößen N



(b) Steigende Kardinalität N, Aktualisierungsselektivität $s_u = 0,1$, verschiedene Anfrageselektivitäten s_q



(c) Partitionsgröße 1.000, verschiedene Anfrageselektivitäten s_q und variierende Aktualisierungsselektivität s_u zwischen 0% und 10%

Abbildung 4.11: Evaluierung der Korrelationswahrscheinlichkeit

Vorschrift errechnet werden. Die Bestimmung der dazu notwendigen Parameter für die Funktion $p(N, s_q, s_u)$ ist in Abbildung 4.10 dargestellt. Im Beispiel ergibt sich somit eine Korrelationswahrscheinlichkeit von 80%.

Evaluierung Der Einfluss der Partitionsgröße N und der Selektivitäten s_q und s_u auf die Korrelationswahrscheinlichkeit eines Anfrage-Aktualisierungs-Paares ist in Abbildung 4.11 dargestellt. Im ersten Experiment wurde zu diesem Zweck die Aktualisierungsselektivität s_u auf 10% festgelegt; die Partitionsgrößen (zwischen 20 und 1000) und Anfrageselektivitäten variieren (siehe Abbildung 4.11a). Es ist erkennbar, dass die Korrelationswahrscheinlichkeit für größere Partitionen selbst schon bei hohen Anfrageselektivitäten sehr schnell gegen 100% strebt. Abbildung 4.11b stellt diesen Zusammenhang nochmals detaillierter für sehr hohe Anfrageselektivitäten und wachsende Partitionsgrößen dar. Den Zusammenhang zwischen variierenden Anfrage- und Aktualisierungsselektivitäten zeigt abschließend Abbildung 4.11c. Bei hohen Anfrageselektivitäten von 0.5% und 1% steigt die Korrelationswahrscheinlichkeit mit kleiner werdender Aktualisierungsselektivität nur sehr langsam an, steigt aber kontinuierlich mit abnehmender Anfrageselektivität.
Aus der Analyse der Korrelationswahrscheinlichkeit sind folgende Schlüsse ableitbar: Ist für ein System die Gültigkeit des bisherigen Korrelationskriteriums $P_{q_i} \cap P_u \neq \emptyset$ zu überprüfen, so ist eine Analyse des Workloads, d.h. der Korrelationswahrscheinlichkeit der Anfrage-Aktualisierungs-Paare, notwendig. Ist die Korrelationswahrscheinlichkeit gleichbleibend hoch und kann davon ausgegangen werden, dass sich der Workload auch in Zukunft wenig verändert, so ist das bisherige Kriterium ausreichend. Andernfalls muss das Korrelationskriterium um eine Ausnahmebehandlung erweitert werden, die jeweils die Wahrscheinlichkeit einer Überlappung prüft:

$$P_{q_i} \cap P_u \neq \emptyset \wedge p(N, s_{q_i}, s_u) > MPT.$$

Der Parameter MPT (engl. matching probability threshold) ist ein vom Systemadministrator festzulegender Schwellwert, der angibt, wie hoch die Wahrscheinlichkeit sein muss, damit eine Korrelation festgestellt wird.

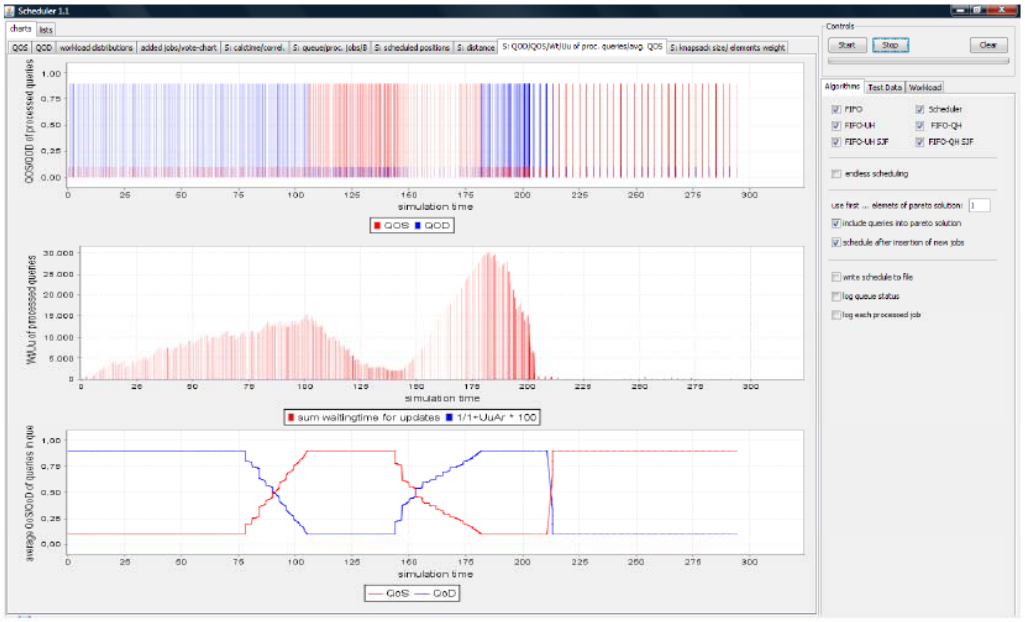
4.5 Evaluierung

In einer Reihe ausgewählter Experimente wurden 1) die Leistung und Adaptivität, 2) die Laufzeit- und Speicherkomplexität und 3) die Änderungsstabilität der multikriteriellen Ablaufplanung unter verschiedenen Workloads und Nutzeranforderungen evaluiert.

4.5.1 Experimentierumgebung

Die Experimentierumgebung besteht aus zwei Komponenten: einem Workload-Generator und einer Scheduling-Komponente, die verschiedene Scheduling-Verfahren sowie die multikriterielle Ablaufplanung implementiert. Beide Komponenten sind durch

(a) Leistungsvergleich der verschiedenen Ablaufplanstrategien



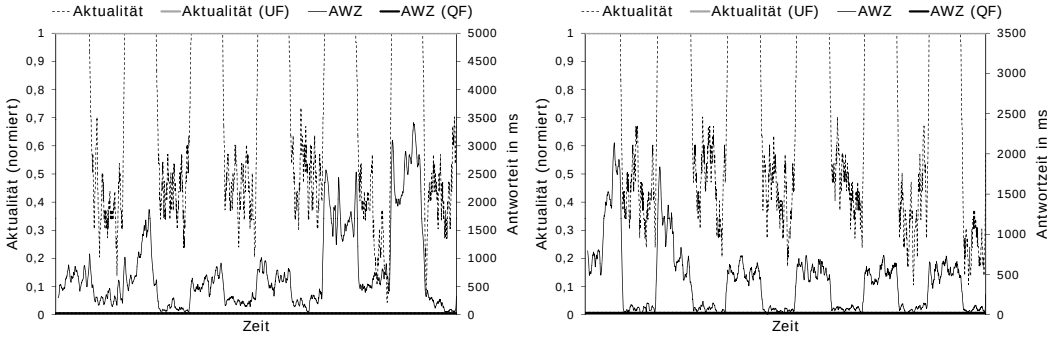
(b) Adaptivität der multikriteriellen Ablaufplanung

Abbildung 4.12: Screenshots der Simulationsumbegung

eine zentrale Simulationsumgebung, die in den Abbildungen 4.12a und 4.12b dargestellt ist, parametrier- und steuerbar. Der durch den Generator erzeugte Workload bietet vielfältige Parameteroptionen, die in der Simulationsumgebung verändert werden können: Anzahl der Anfragen und Aktualisierungen sowie deren Kosten, Profit der Aktualisierungen, zeitlicher Abstand zwischen dem Hinzufügen neuer Transaktionen (Simulation der Last) und Nutzeranforderungen bezüglich Aktualität und Antwortzeit. Die Werteverteilung aller Parameter folgt wahlweise einer Normal- oder einer Zipfverteilung. Weiterhin ist der Grad der Abhängigkeit zwischen Anfragen und Aktualisierungen parametrierbar, so dass verschiedene Anfragetypen erzeugt werden können: große Bereichsanfragen, die von vielen Aktualisierungen abhängig sind und Punktanfragen, die nur wenige oder keine Abhängigkeit vorweisen.
Die durch eine festgelegte Parametrierung erzeugten Workloads können des Weiteren aufgezeichnet werden, so dass eine Wiederholbarkeit und Vergleichbarkeit der Experimente gewährleistet ist. Mit Hilfe eines aufgezeichneten Workloads ist es ebenfalls möglich eine sogenannte „Endlossimulation“ durchzuführen, in welcher der Workload in einer Schleife ausgeführt wird und so die Möglichkeit gibt, verschiedenen Parametrierungen zu testen. Weiterhin wurden verschiedene Sichten implementiert um die Leistungsfähigkeit der in diesem Kapitel vorgestellten Verfahren zu analysieren. Zwei dieser Sichten sind in den Abbildungen 4.12a und 4.12b dargestellt: In Abbildung 4.12a ist ein Vergleich zwischen der multikriteriellen Ablaufplanung und den Referenzalgorithmen QF (queries first) und UF (updates first) dargestellt. Es wird jeweils die für einen Workload erzielte Aktualität und das Mittel der Antwortzeiten berechnet. Die Adaptivität der multikriteriellen Ablaufplanung kann in der in Abbildung 4.12b gezeigten Sicht nachvollzogen werden. In dieser sind die Anforderungen der Nutzer an das System (in der Abbildung oben) sowie die tatsächliche Umsetzung dargestellt.
Die Ausführung der Transaktionen erfolgt ebenfalls simuliert ohne Verwendung einer Datenbank. Die Ausführungzeiten sind dabei durch die vom Workload-Generator erzeugten Kosten vorgegeben. Durch den Datenbankoptimierer verursachte Fehler, resultierend aus schlechten Kostenabschätzungen, wurden somit nicht betrachtet. Es ist jedoch offensichtlich, dass die Qualität der Ablaufplanung direkt durch die Qualität der Kostenschätzungen beeinflusst wird.

4.5.2 Leistungsvergleich und Adaptivität

Im ersten Experiment wurde zunächst die Qualität der multikriteriellen Ablaufplanung im Vergleich zu zwei Basisalgorithmen, QF und UF, evaluiert sowie die Adaptivität bei sich ändernden Nutzeranforderungen gezeigt. QF favorisiert stets Anfragen vor Aktualisierungen und minimiert somit die Antwortzeit. UF priorisiert stets Aktualisierungen vor Anfragen und maximiert dadurch die Datenaktualität. Beide Algorithmen liefern in dem jeweiligen Qualitätskriterium das optimale Ergebnis, wie in Abbildung 4.13 dargestellt.
Die Adaptivität der multikriteriellen Ablaufplanung wurde an zwei Workloads bewertet, W_{GAUSS} und W_{ZIPF}. Beide bestehen aus jeweils 5.000 Anfragen und 5.000



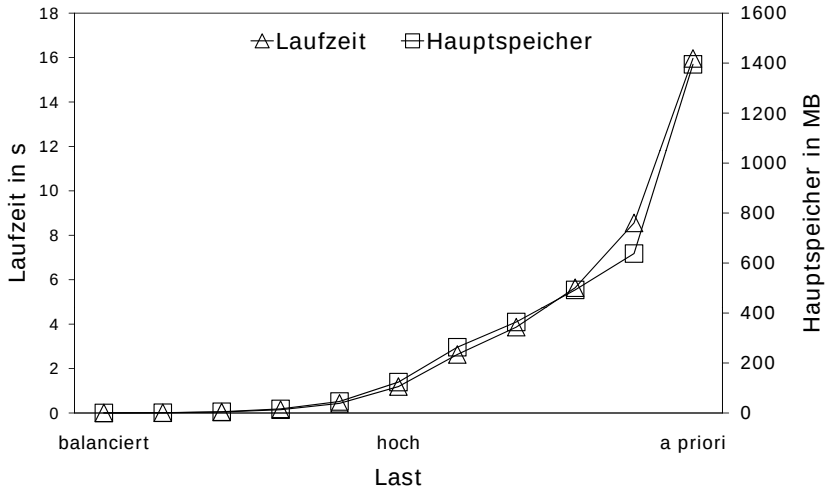
(a) Normalverteilung **(b)** Zipfverteilung

Abbildung 4.13: Adaptivität und Leistung der multikriteriellen Ablaufplanung

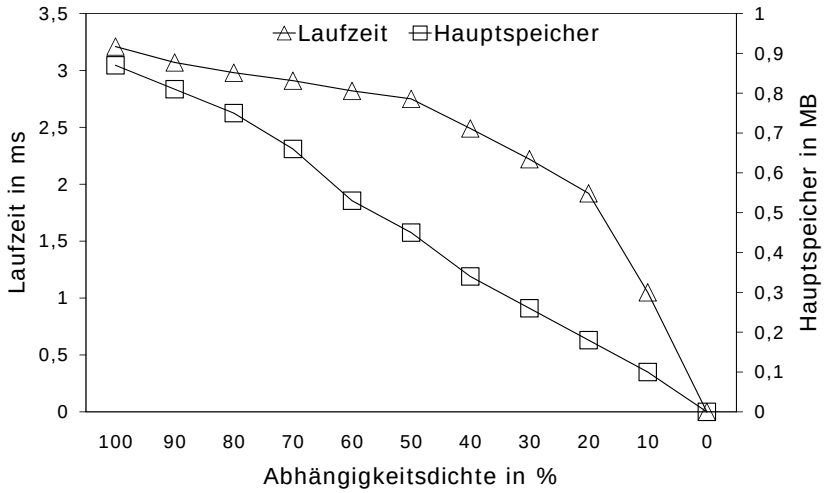
Aktualisierungen, deren Kosten und Profit aus einer Normalverteilung (mit $\mu_{c_q} = 5000\ ms$, $\mu_{c_u} = 500\ ms$, $\mu_{p_u} = 50\ rows$ und $\sigma = 1$)) bzw. aus einer Zipfverteilung gezogen wurden (mit $c_q = 10 - 20000\ ms$, $c_u = 5 - 500\ ms$ und $p_u = 1 - 500\ rows$). Während der Ausführung beider Workloads wurde das Nutzerverhalten sechsmal zwischen den Extremen $qos_{q_i} = 0$ und $qos_{q_i} = 1$ verändert. In den Abbildungen 4.13a und 4.13b sind die Antwortzeit und die Aktualität der Anfragen über die Laufzeit der Workloads hinweg dargestellt. Zur Glättung der Darstellung wurde der Durchschnitt der Werte über ein gleitendes Fenster der Größe 30 gebildet. Es ist zu sehen, dass mit jeder Änderung des Nutzerverhaltens von einem Extrem (hohe Datenaktualität) in das andere (kurze Antwortzeit) auch die Ablaufplanung angepasst wird. Hohe Aktualitätsanforderungen resultieren in einem höheren Wert der Rucksackgröße, wodurch mehr Aktualisierungen stärker priorisiert werden. Bei der Forderung nach kurzen Antwortzeiten wird der Rucksack verkleinert, wodurch weniger oder gar keine Aktualisierungen vor Anfragen ausgeführt werden.
Des Weiteren wird ersichtlich, dass die multikriterielle Ablaufplanung in jeder Phase – mit kleinen Abstrichen – genauso gut ist wie die jeweilige optimale Ablaufplanung QF bzw. UF (siehe die grauen und scharzen dicken Linien in den Abbildungen 4.13a and 4.13b). Die marginal schlechteren Ergebnisse resultieren aus Überlastsituationen, in denen Anfragen aus der vorherigen Phase noch im System sind und somit gegenteilige Anforderungen enthalten.

4.5.3 Laufzeit- und Speicherkomplexität

Die in Abschnitt 4.2.3 analytisch ermittelte Laufzeit- und Speicherkomplexität, $O(N \cdot B)$ bzw. $\Omega(N \cdot B)$, wurde in einer Reihe von Experimenten untersucht. Von besonderem Interesse war dabei das pseudopolynomielle Verhalten des *MultiOptScheduling*-Algorithmus.
Für das erste Experiment wurde die Abhängigkeitsdichte zwischen Anfragen und Aktualisierungen auf 10% festgesetzt, was einem eher hohen bzw. pessimistischen



(a) Steigende Last



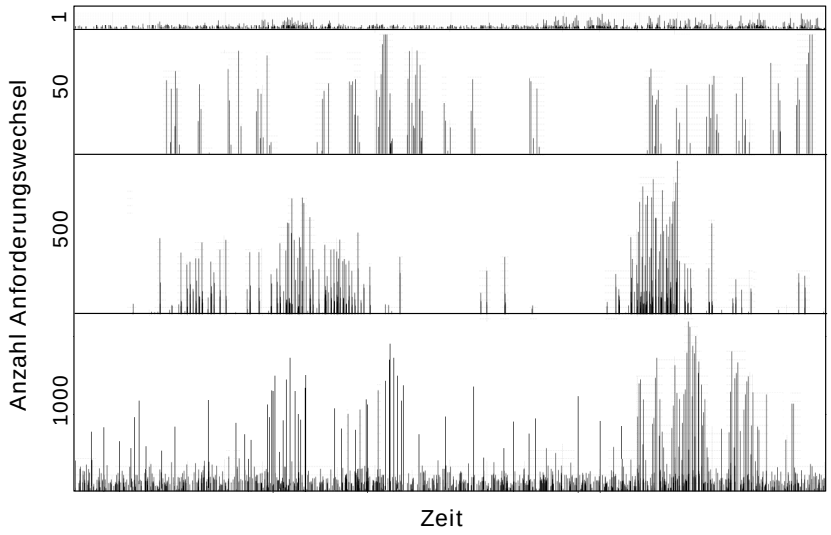
(b) Steigende Abhängigkeitsdichte

Abbildung 4.14: Laufzeit und Speicherbelegung

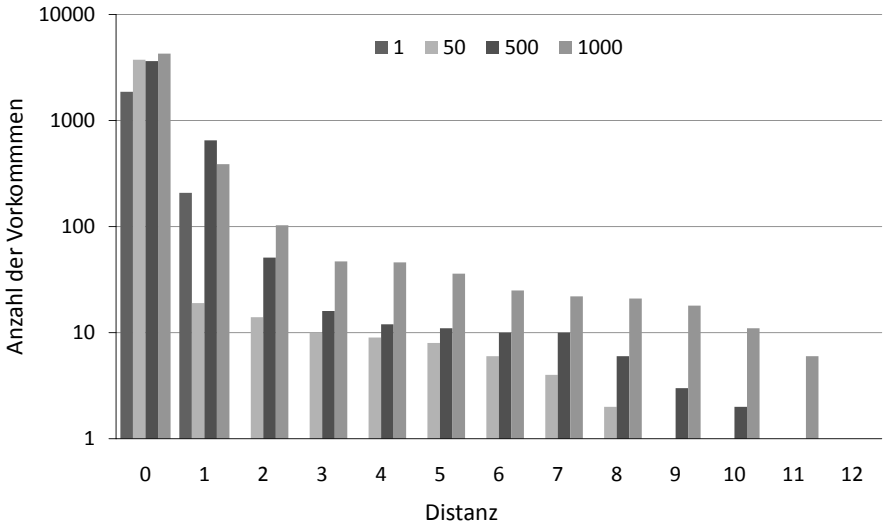
Wert entspricht. Die durch den *MultiOptScheduling*-Algorithmus zu verarbeitende Last wurde zunehmend erhöht, indem die Transaktionen (1.000 Anfragen und 1.000 Aktualisierungen) zunehmend schneller dem System hinzugefügt wurden. Unter geringster Last war die Ankunftsrate der Transaktionen gleich der Bedienrate des Systems (*balanced* in Abbildung 4.14a). Bei höchster Last wurden alle Transaktionen gleichzeitig in das System gegeben, d.h. die Ablaufplanung musste 2.000 Transaktionen gleichzeitig verarbeiten (*a priori*). In Abbildung 4.14a sind Laufzeit und Speicheranforderung unter steigender Last dargestellt. Für praxisnahe Szenarien mit wenigen dutzend bzw. hundert zu verarbeitenden Transaktionen gleichzeitig beträgt die Laufzeit 2,5 bis 150 Millisekunden und der Speicherverbrauch liegt zwischen 0,1 und 15 MB. Im Fall der A-priori-Verarbeitung der 2.000 Transaktionen zur gleichen Zeit benötigt der *MultiOptScheduling*-Algorithmus 16 Sekunden und 1.400 MB Speicher. In Echtzeit-Data-Warehouse-Systemen, in denen die Aktualisierungen kontinuierlich verarbeitet werden, sind 1.000 Aktualisierungen zur gleichen Zeit jedoch eine sehr pessimistische Annahme. Auf der anderen Seite können bei sehr großen Aktualisierungsfenstern, bestehend aus 1.000 Aktualisierungen und mehr, 16 Sekunden im Vergleich zur Laufzeit der Transaktionen vernachlässigt werden.
In einem zweiten Experiment wurde das Laufzeit- und Speicherverhalten unter abnehmender Abhängigkeitsdichte, von 100% bis 0%, für 1.000 Anfragen und 1.000 Aktualisierungen und balancierte Last untersucht (siehe Abbildung 4.14b). Es ist zu sehen, dass mit fallender Abhängigkeitsdichte sowohl die Laufzeit als auch der Speicherverbrauch immer stärker abnehmen. Dies liegt in der bereits in Abschnitt 4.2.3 analytisch ermittelten Komplexität begründet, die insbesondere von der Anzahl der Aktualisierungselemente N abhängig ist. Diese wird aus dem Produkt der Anzahl der sich gleichzeitig im System befindlichen Anfragen und Aktualisierungen sowie der Abhängigkeitsdichte berechnet, die somit maßgeblich die Laufzeit- und Speicheranforderungen beeinflusst.
Beide Experimente zeigen die Anwendbarkeit des *MultiOptScheduling*-Algorithmus für praxisnahe Workloads, die aus wenigen hundert Transaktionen zur gleichen Zeit bestehen und eine mittlere Abhängigkeitsdichte von 20% und weniger aufweisen. Der durch die Ablaufplanung verursachte Mehraufwand kann somit für diese Szenarien im Vergleich zur Verarbeitungszeit komplexer BI-Anfragen sowie hinsichtlich des verbesserten Antwortzeitverhaltens durch die Optimierungen (siehe vorheriger Abschnitt) vernachlässigt werden.

4.5.4 Änderungsstabilität

In einem letzten Experiment wurde die Änderungsstabilität der Pareto-effizienten Ablaufpläne unter Anwendung des Distanzmaßes aus Abschnitt 4.3 untersucht. Der verwendete Workload bestand aus 5.000 Anfragen und 5.000 Aktualisierungen und die Nutzeranforderungen wurden 1, 50, 500 bzw. 1.000 mal zwischen den beiden Extremen geändert (von $qos_{q_i} = 0$ nach $qos_{q_i} = 1$ und vice versa). In Abbildung 4.15a sind die Distanzen aller paarweisen Ablaufpläne über die Simulationszeit hinweg dargestellt. Für gleichbleibende Nutzeranforderungen sind die Distanzen zwischen



(a) Distanzen während der Simulation



(b) Histogramm des Distanzmaßes

Abbildung 4.15: Änderungsstabilität unter veränderlichen Nutzeranforderungen ($qos_{q_i} = 0 \leftrightarrow 1$)

den Ablaufplänen sehr gering, nehmen jedoch mit wachsender Frequenz der Anforderungswechsel stark zu. In Abbildung 4.15b sind die gerundeten Distanzwerte in einer alternativen Darstellungsform als Histogramm zu sehen.
Die dargestellten Ergebnisse entsprechen der Anforderung, dass stabile Umgebungsverhältnisse in ähnlichen Ablaufplänen resultieren. So wird bei gleichbleibenden Nutzeranforderungen ein Pareto-effizienter Ablaufplan zum Zeitpunkt t_1 mit hoher Wahrscheinlichkeit auch zu einem späteren Zeitpunkt noch die Eigenschaft der Pareto-Effizienz besitzen. Bei wechselnden Nutzeranforderungen ist zu erwarten, dass auch die aufeinanderfolgenden Pareto-effizienten Ablaufpläne stark voneinander abweichen. Dies wird durch die Ergebnisse in Abbildung 4.15b bestätigt. Für häufige Anforderungswechsel wird auch die Gewichtung der zu optimierenden Qualitätskriterien verändert, was zu hohen Distanzwerten zwischen den Ablaufplänen führt. Dies unterstreicht nochmals die Ergebnisse aus Abschnitt 4.5.2, in welchem die Adaptivität des Ansatzes gezeigt wurde; außerdem verdeutlicht dies die Anwendbarkeit der multikriteriellen Ablaufplanung auch für das Online-Scheduling (siehe Abschnitt 4.3).

4.6 Zusammenfassung

Echtzeit-Data-Warehouse-Systeme müssen kontinuierliche Ströme aus Anfragen und Aktualisierungen, unter Berücksichtigung widerstreitender Anforderungen, verarbeiten. Basierend auf der Anforderungsanalyse aus den Abschnitten 3.5.1 und 3.5.2 wurden die Datenaktualität sowie die Anfrageantwortzeit als zentrale Qualitätsdimensionen herausgerarbeitet. Für beide Qualitätskriterien wurden verschiedene Metriken vorgestellt und gegeneinander verglichen sowie Maximierungs- und Minimierungsprobleme formuliert. Es wurde ein Workloadmodell erarbeitet, mit welchem Aktualisierungen und ihre Position in der Anfragewarteschlange als Kosten-Profit-Vektor darstellbar sind. Darauf aufbauend, konnte ein Algorithmus zur Lösung der multikriteriellen Optimierung, auf Grundlage des Rucksackproblems, entwickelt werden. Die so ermittelten Ablaufpläne besitzen die Eigenschaft der Pareto-Effizienz, d.h. sie sind in keiner Qualitätdimension verbesserbar, ohne sich in der anderen Dimension zu verschlechtern. Die Ablaufpläne sind somit, hinsichtlich der im Data-Warehouse-System vorliegenden Nutzeranforderungen, optimal. Zur Bewertung der Stabilität von Pareto-effizienten Ablaufplänen unter dynamischen Aspekten wurde ein entsprechendes Distanzmaß entwickelt. Dieses wurde in Experimenten für den dynamische Fall untersucht und die Anwendbarkeit der multikriteriellen Ablaufplanung auch für das Online-Scheduling bestätigt. Des Weiteren wurden die Laufzeit- und Speicheranforderungen evaluiert und die Adaptivität der vorgeschlagenen Ablaufplanung bezüglich sich ändernden Nutzeranforderungen erfolgreich nachgewiesen.
Zur Ermittlung von Abhängigkeiten zwischen Anfragen und Aktualisierungen wurde die Annahme gemacht, dass die Daten in partitionierter Form vorliegen und dass diese Partitionen stets von jeder Transaktion adressiert werden. Die Genauigkeit

dieser grobgranulare Korrelationsbestimmung wurde analytisch evualiert, sowie um ein statistisches Verfahren, basierend auf den Selektivitäten der Anfragen und Aktualisierungen, erweitert.

Fallstudiendiskussion

Mit der zunehmenden Forderung nach der Integration hochaktueller Daten in das Data-Warehouse wird eine intelligente Ablaufsteuerung immer wichtiger. Dies gilt sowohl für das Data-Warehouse-Szenario der UBS WM&SB (Fallstudie A, siehe Abschnitt 2.1) als auch für die GfK Retail&Technology (Fallstudie B, siehe Abschnitt 2.2).
Die UBS WM&SB besitzt mit dem Data Sourcing Framework sowie dem Job Dependency Manager bereits die Infrastruktur, um zentral die Aktualisierungs- und Anfragelast zu überwachen und zu steuern. Dazu wird einem Aktualisierungsvorgang bzw. einem Job eine Priorität zugewiesen sowie die Menge an Ressourcen festgelegt, die der Job bei der Ausführung verwenden darf (Grad der Parallelität). Des Weiteren können für die Parameterzuweisung unterschiedliche Zeitfenster unterschieden werden. So ist zum Beispiel der Grad der Parallelität für Jobs, die tagsüber eingebracht werden stark begrenzt, um nebenläufige Anfragen nur wenig zu verzögern. Nimmt die Anfragelast in den Feierabendstunden und in der Nacht ab, so gelten andere Parameterbelegungen. Allerdings erfolgt die Parameterzuweisung nur statisch anhand fest definierter Regeln, was zur Folge hat, dass die Ablaufpläne nicht optimal sind und zur Verfügung stehende Ressourcen nicht voll ausgenutzt werden, d.h. ein Job wird nur soviele Ressourcen verwenden wie ihm zugewiesen wurden, auch wenn die aktuelle Last sehr gering ist. In der momentanen Situation, in der Aktualisierungen außerhalb der Geschäftszeiten eingebracht werden, ist die vorhandene statische Ablaufplanung ausreichend. Bei zunehmendem Bedarf nach Echtzeitdaten ändern sich jedoch die Anforderungen, da sowohl Aktualisierungen als auch Anfragen kontinuierlich um Ressourcen konkurrieren. Eine statische Ablaufplanung, in der die Parameter nur drei Belegungszustände kennen (Geschäftszeit, Abendstunden und Nacht), ist im Echtzeitumfeld nicht adäquat. Stattdessen kann die in diesem Kapitel entwickelte multikriterielle Ablaufplanung angewendet werden. Der Profit- und der Kosten-Parameter ist dazu mit der notwendigen Semantik zu belegen. So werden zum Beispiel Aktualisierungen an den Stammdaten im Data-Warehouse der WM&SB stets priorisiert verarbeitet und müssen daher einen höheren Profit zugewiesen bekommen.
Das Data-Warehouse der GfK Retail&Technology wurde in Abschnitt 2.2 als operativ klassifiziert, da die Daten ständigen Änderungen unterworfen sind. Bereits eingebrachte Daten können aufgrund von Qualitätsprüfungen noch nachträglich geändert werden. Des Weiteren ist der Zeitpunkt für die Datenlieferungen infolge der räumlichen Datenproduktion nur schwer vorhersagbar. Eine Ablaufplanung könnte helfen, die Datenproduktion und die Datennutzung aufeinander abzustimmen. Jedoch sind Ad-hoc-Anfragen in diesem Szenario relativ selten. Stattdessen werden für die Berichtsproduktion die Anfragen stapelweise verarbeitet (bis zu 100.000 Anfra-

gen pro Batch), so dass die Freiheitsgrade für eine Ablaufplanung relativ gering sind. Für zukünftige Online-Analysen (siehe Abschnitt 2.2.3) könnte die multikriterielle Ablaufplanung jedoch mit Gewinn eingesetzt werden.

5 Bewertung von Ladestrategien in mehrstufigen Datenproduktionsprozessen

Die Anforderungsanalyse in Kapitel 3 hat gezeigt, dass Datenproduktionsprozesse in der Regel mehrstufiger Natur sind, was auch durch die Fallstudien aus Kapitel 2 bestätigt werden konnte. Der mehrstufige Charakter des Datenproduktionsprozesses sowie die lose Kopplung der involvierten Datenbanksysteme führen zu der Beobachtung, dass die Daten innerhalb des Data-Warehouse-Systems mehrfach, zum Teil in verschiedenen Aggregationsstufen, repliziert werden. Die Granularität, in welcher die Datenverarbeitung stattfindet, ist durch das Partitionierungsschema des Data-Warehouse-Systems vorgegeben (siehe Abschnitt 4.1.4). Zu einem bestimmten Zeitpunkt betrachtet weisen die gleichen Partitionen – in verschiedenen Produktionsstufen – unterschiedliche Datenaktualität auf. Beginnend bei den Quellsystemen nimmt der Grad der Datenaktualität in Richtung des Data-Warehouses und der Data-Marts kontinuierlich ab, wie in Abbildung 5.1 beispielhaft dargestellt. Die Aktualität der Daten im Vergleich zu den Quellsystemen (Δt_i) ist dabei farblich kodiert. Der Fortschritt der technischen Integration innerhalb des Datenproduktionsprozesses entscheidet über die Anwendbarkeit des Partitionierungsschemas, was in Abbildung 5.1 durch Struktur und Verteilung der einzelnen Partitionen dargestellt ist. Sind die Daten in der entsprechenden Stufe des Datenproduktionsprozesses soweit integriert, dass das Partitionierungsschema angewandt werden kann (in der Darstellung ist dies für den Arbeitsbereich und alles darüber der Fall), so ermöglicht dies den direkten Vergleich der Partitionsaktualität über mehrere Stufen hinweg.
Im Kontext des Echtzeitbetriebs eines Data-Warehouse-Systems ist es somit möglich, Nutzeranfragen, die eine hohe Aktualität erfordern, an diejenige Stufe des Datenproduktionsprozesses zu leiten, die diese Aktualität zur Verfügung stellen kann. Dies setzt in den meisten Fällen ein Umschreiben der Anfragen voraus, was im Folgenden jedoch als gegeben betrachtet wird. Des Weiteren ist eine prozessübergreifende Ablaufplanung der Aktualisierungen erforderlich, um stets die notwendige Aktualität in allen Prozessstufen zur Verfügung stellen zu können. In Kapitel 4 wurde ein Algorithmus zur Ablaufplanung vorgestellt, der stets den optimalen bzw. den Pareto-effizienten Ablaufplan hinsichtlich der Qualitätsdimensionen Antwortzeit und Aktualität liefert. Der so ermittelte Ablaufplan kann auch als minimalintrusiv bezeichnet werden, da stets nur so viele Aktualisierungen eingebracht werden wie zum Erhalt der Datenqualität notwendig sind. Der Untersuchungsgegenstand beschränkte sich dabei jedoch auf lediglich einstufige Prozesse und wird daher im Folgenden

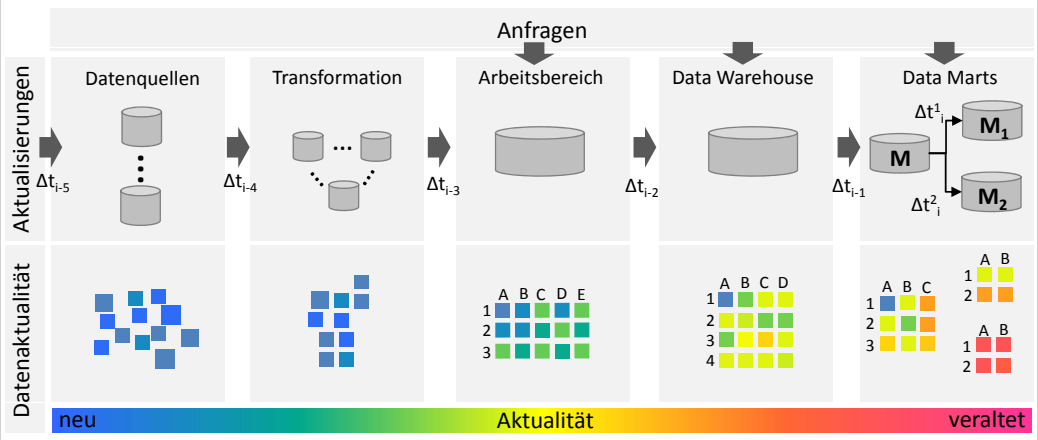


Abbildung 5.1: Push-basiertes Laden eines Data-Warehouses und Illustration der Aktualität

auf mehrstufige Prozesse ausgeweitet.
Daraus ergeben sich zwei zentrale Fragestellungen, die in diesem Kapitel adressiert werden: Zum einen, wie ist die multikriterielle Ablaufplanung aus Kapitel 4 in mehrstufige Prozesse zu integrieren, um sowohl innerhalb der einzelnen Prozessstufen als auch auf Ebene des gesamten Prozesses eine optimale Datenaktualität zu erreichen? Und zum anderen, wie können komplexe mehrstufige Datenproduktionsprozesse überwacht, d.h. wie kann die Entwicklung der Datenaktualität visualisiert werden, um so Ladevorgänge zu optimieren? Die Fragestellungen können nach dem Optimierungszeitraum in Online- bzw. Offline-Ansätze unterschieden werden. Während die Ablaufplanung die Online-Optimierung zur Laufzeit adressiert, unterstützt die Visualisierung der Aktualitätszustände von Datenproduktionsprozessen die Offline-Optimierung.

5.1 Ablaufplanung in mehrstufigen Datenproduktionsprozessen

In Kapitel 4 wurde ein minimal-intrusiver Algorithmus zur Ablaufplanung vorgestellt, der stets nur diejenigen Aktualisierungen priorisiert, die in Hinblick auf die Erfüllung der Nutzeranforderungen notwendig sind. Die erforderliche Aktualität der Daten war somit zu jedem Zeitpunkt gewährleistet, ohne dass eine definierte Kostenschranke überschritten wurde. Für mehrstufige Datenproduktionsprozesse ist die Ablaufplanung jedoch wesentlich komplexer, da durch die vielen Verarbeitungsschritte die Aktualisierungen zusätzlichen Verzögerungen bzw. Verweilzeiten ausgesetzt sind. Dies hat zur Folge, dass die Nutzeranforderungen, aufgrund der zeitlichen Verschiebung, nur sehr verzögert oder gar nicht erfüllt werden können.
Zur Adressierung dieses Problems können zwei wesentliche Ansätze unterschieden

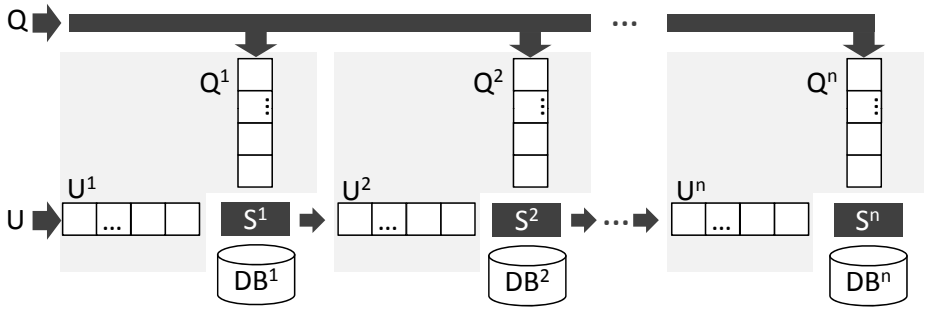


Abbildung 5.2: Systemarchitektur

werden: die lokale und die globale Ablaufplanung. Der erste Ansatz implementiert für jede Prozessstufe eine individuelle Ablaufplanung, die die Anfragen und Aktualisierungen dieser Stufe berücksichtigt; dies ist aus lokaler Sicht damit optimal. Jedoch können die so entstehenden Ablaufpläne aus Sicht des gesamten Produktionsprozesses nie die beste Aktualität erreichen. Daher ist dieser Ansatz der globalen Ablaufplanung gegenüber zu stellen, welche den Workload des gesamten Datenproduktionsprozesses kennt. Die Vor- und Nachteile dieser beiden Ansätze sind Untersuchungsgegenstand der folgenden Abschnitte.
Es bleibt anzumerken, dass lediglich Ad-hoc-Anfragen betrachtet werden und somit kein zusätzliches Wissen zu wiederkehrenden Zugriffsmustern vorhanden ist.

5.1.1 Ladestrategien und Problemstellung

Grundlage der Untersuchungen ist eine Data-Warehouse-Landschaft, bestehend aus einer Reihe entkoppelter Datenbanken, die durch einen gemeinsamen Datenproduktionsprozess miteinander verknüpft sind. Die lose Kopplung der Systeme wird durch Warteschlangen realisiert. Pro Datenbank existieren jeweils zwei Warteschlangen, eine für Anfragen und eine für Aktualisierungen (siehe Abbildung 5.2). Alle Anfragen an das Data-Warehouse-System werden über eine gemeinsame Middleware an die verschiedenen Prozesstufen verteilt. Hierbei sind verschiedene Strategien denkbar, welche jedoch nicht Gegenstand dieser Arbeit sein sollen. Aktualisierungen werden in Reihenfolge ihres Aufretens aufsteigend propagiert und zur Laufzeit priorisiert. Zur Vereinfachung der Untersuchung wird angenommen, dass sich der Datenproduktionsprozess nicht verzweigt.
Pro Prozessstufe existiert eine Komponente zur Ablaufplanung S_i, welche die Datenbanken entsprechend der Nutzeranforderungen mit Anfragen bzw. Aktualisierungen versorgt. Die Ablaufplanung kann lokal oder global erfolgen. Im ersten Fall hat der Scheduler S_i nur Wissen über den Workload der eigenen Stufe i; im zweiten Fall hat er Informationen über die Anfragen in allen Stufen. Zwischen diesen Extrema sind beliebige Abstufungen denkbar.
Sowohl die lokale als auch die globale Ablaufplanung basieren auf dem Algorithmus zur multikriteriellen Ablaufplanung, der in Kapitel 4 vorgestellt wurde. Beide

Verfahren sind in Teilen ähnlich und daher in Algorithmus 2 gemeinsam skizziert. Der wesentliche Unterschied besteht in der Erzeugung der Eingabeelemente für den Rucksackalgorithmus. Während in der lokalen Ablaufplanung lediglich die Anfragen Q^k und die Aktualisierungen U^k der betreffenden Prozessstufe k betrachtet werden, sind in der globalen Ablaufplanung alle Anfragen $q_i \in \bigcup_{l=k..n} Q^l$ zu betrachten. Das heisst, zur Berechnung der Aktualisierungselemente für die Ablaufplanung in Prozessstufe k werden die Anfragen aller folgenden Stufen ebenfalls berücksichtigt.
Daraufhin wird in beiden Verfahren der Pareto-effiziente Ablaufplan bestimmt, wie in Abschnitt 4.2 dargelegt. Das Ergebnis sind jeweils für die einzelnen Prozessstufen gültige Ablaufpläne. Zusätzlich kann auch eine Gewichtung der Anfragen und damit auch der korrelierten Aktualisierungen durch die Einführung von Service-Leveln erfolgen. Bei der Ausführung einer Anfrage wird deren Datenaktualität bestimmt und für spätere Auswertungen gespeichert.

Problemstellung Bei der nutzergetriebenen Ablaufplanung in einstufigen Prozessen wird auf Basis der Kosten und Profite der Aktualisierungen der Pareto-effiziente Ablaufplan für alle sich im System befindlichen Transaktionen erzeugt. Dies ist im mehrstufigen Prozess jedoch nicht ohne Weiteres möglich, da durch jede Prozessstufe eine zusätzliche Verzögerung eingeführt wird, durch welche die zeitliche Abhängigkeit zwischen Anfragen und mit diesen korrelierten Aktualisierungen gestört wird. Im konkreten Fall führt dies dazu, dass die Aktualität von Anfragen gemindert wird, da Aktualisierungen erst nach deren Ausführung eingebracht werden. Die beiden Strategien – die lokale und die globale Ablaufplanung – sind dahingehend zu untersuchen, inwieweit sie die zeitliche Lokalität zwischen Anfragen und Aktualisierungen erhalten und damit zu guten Ablaufplänen führen.
Ein ähnliches Problem ist in der Cache-Speicherverwaltung mehrstufiger Cache-Hierarchien anzutreffen [95, 31, 15]. Auch hier nimmt die zeitliche Lokalität in der Cache-Hierarchie zunehmend ab. Die typischen Lösungsansätze bestehen darin, zusätzliches Anwendungswissen [52, 91] zu verwenden und so die Lokalität zu erhöhen. Dies ähnelt im Vorgehen der globalen Ablaufplanung. Auch hier wird zur Verbesserung der Datenaktualität zusätzliches Workloadwissen verwendet. Der Unterschied zwischen der Cache-Speicherverwaltung und den Datenproduktionsprozessen liegt jedoch in der Zugriffsstruktur. Während Cache-Zugriffe streng entlang der Cache-Hierarchie erfolgen, ist es in Datenproduktionsprozessen möglich, Anfragen direkt an eine beliebige Prozessstufe zu propagieren.

5.1.2 Evaluierung und Diskussion

Der experimentelle Vergleich zwischen der lokalen und der globalen Ablaufplanung ist Inhalt der folgenden Abschnitte. Ziel ist es, die Eigenschaften der Datenproduktionsprozesse und Workloads herauszustellen, unter denen die globale Ablaufplanung die Datenaktualität im Vergleich zur lokalen Ablaufplanung signifikant verbessern kann. Im Zuge dessen sind Richtlinien zu entwickeln, die die Entscheidung, welche Ablaufplanung zu verwenden ist, unterstützen. Unter Anwendung dieser Ergebnis-

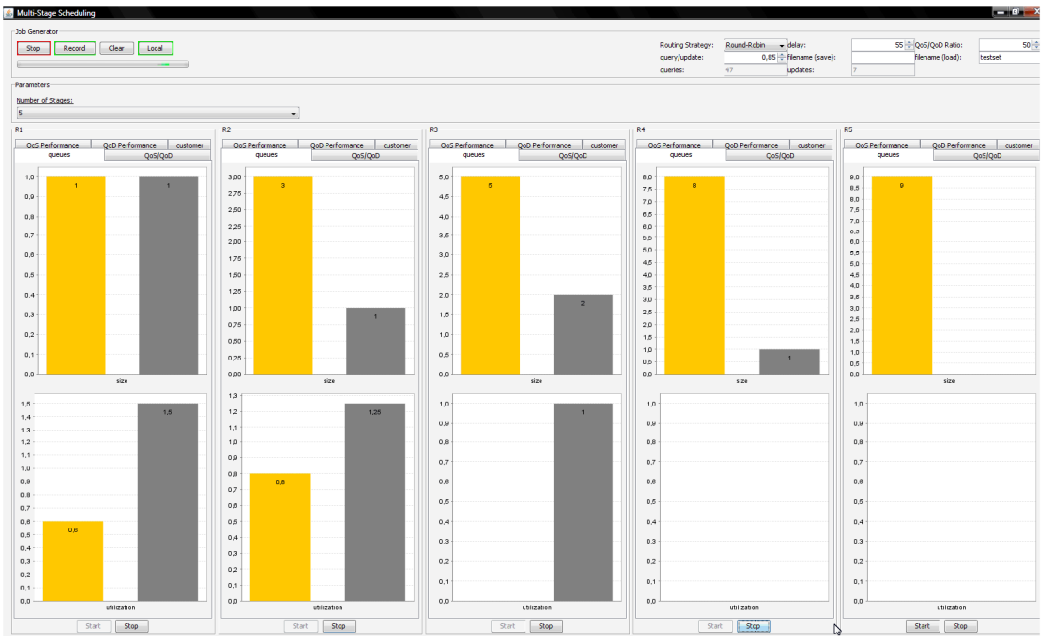
Algorithmus 2 Schematische Beschreibung der Ablaufplanung

```
1: if modus=lokal then
2:    Berechnung der Anfrage-Aktualisierungskorrelation zwischen U_k und allen q_i ∈ Q^k
3: else if modus=global then
4:    Berechnung der Anfrage-Aktualisierungskorrelation zwischen U_k und allen q_i ∈
      ⋃_{j=k..n} Q_j
5: end if
6: Berechnung des Pareto-effizienten Ablaufplans
7: Ausführung der Transaktion mit der höchsten Priorität
8: if transaktion=Anfrage then
9:    Ausführung der Anfrage q_i
10:   Ermittle die Anzahl der nicht eingebrachten Aktualisierungen für q_i
11: else if modus=Aktualisierung then
12:   Ausführung der Anfrage u_i
13: end if
```

se werden die Schwächen der globalen Ablaufplanung ermittelt und Verbesserungen vorgeschlagen.
Da der Aufbau einer echten verteilten Umgebung sehr aufwendig und teuer ist, wurde eine entsprechende Simulationsumgebung entwickelt. Diese erlaubt es, Datenproduktionsprozesse unterschiedlicher Größe anhand einer Vielzahl von Experimenten effizient gegeneinander zu vergleichen.

Experimentierumgebung

Die Experimentier- bzw. Simulationsumgebung zur Untersuchung der Ablaufplanung in mehrstufigen Prozessen ist in den Abbildungen 5.3a und 5.3b dargestellt. Sie setzt sich zusammen aus einem Workload-Generator, einer Komponenten zur Ablaufplanungen, die auf Datenproduktionsprozessen beliebiger Länge instanziiert werden können sowie zahlreichen Sichten zur Unterstützung der Auswertung. Die Implementierung ermöglicht es Datenproduktionsprozessen bis zu einer Länge von 15 Stufe zu überwachen und bezüglich verschiedener Leistungsparameter auszuwerten. In Abbildung 5.3a ist die Online-Auswertung eines fünfstufigen Datenproduktionsprozess dargestellt. Die Stufe sind von links beginnend aufsteigend sortiert. Für jede Produktionsstufe ist eine separate Ansicht verfügbar, in welcher zum einen die Belegung der Anfrage- und Aktualisierungswarteschlangen und zum anderen die Last dargestellt ist. In einer weiteren Ansicht (siehe Abbildung 5.3b) sind die nicht eingebrachten Aktualisierungen je Prozessstufe dargestellt. Dafür steht ein Histogramm-Sicht (unterer Teil) und eine erweiterte Sicht zur Verfügung (oberer Teil), in welcher für jede Anfrage die Anzahl der fehlenden Aktualisierungen sowie deren Position im Datenproduktionsprozess dargestellt ist (farblich kodiert).
Im oberen Teil der Simulationsumgebung kann der Workload bezüglich verschiedener Parameter variiert werden: Anzahl der Anfragen und Aktualisierungen, Last, Aktualitätsanforderungen, Anfrage und Aktualisierungskosten und Service-Level der Anfragen (jeweils aus einer Normalverteilung bzw. einer Pareto-Verteilung gezogen).



(a) Darstellung der Arbeitslast und der Warteschlangenlänge

(b) Darstellung der nicht eingebrachten Aktualisierungen

Abbildung 5.3: Simulationsumgebung zur Analyse mehrstufiger Datenproduktionsprozesse

Weiterhin ist es möglich einen erzeugten Workload aufzuzeichnen und somit auf Datenproduktionsprozessen verschiedener Länge und Konfigurationen zu testen.

Bewertungskennzahlen Die Entwicklung der Datenaktualität in mehrstufigen Datenproduktionsprozessen ist der primäre Untersuchungsgegenstand der folgenden Evaluierungen. Zu deren Bewertung werden zwei Kennzahlen, basierend auf der in Abschnitt 4.1.2 vorgestellten abstandsbasierten Metrik, verwendet. Diese bemisst die Aktualität eines Anfrageergebnisses nach der Anzahl nicht eingebrachter Aktualisierungen (engl. unapplied updates, kurz *uu*). Die Aktualität einer Anfrage q^i, ausgeführt an Prozessstufe i, berechnet sich aus allen Aktualisierungen, die noch in den Prozessstufen $j \leq i$ verweilen.

$$uu(q^i) = \sum_{u \in U^j, j \leq i, P_q \cap P_u \neq \emptyset} 1. \tag{5.1}$$

Der Index im Exponent bezeichnet die entsprechende Prozessstufe. Zur Bewertung einer Ablaufplanung für einen Workload wird in jeder Prozessstufe die mittlere Anzahl der nicht eingebrachten Aktualisierungen betrachtet:

$$AvgUu^i(W) = \frac{1}{|W_q^i|} \sum_{q \in W_q} uu(q). \tag{5.2}$$

Für detailliertere Analysen sind die nicht eingebrachten Aktualisierungen in Form eines Histogramms zu visualisieren. Die Anzahl der Histogrammklassen ergibt sich aus der maximalen Zahl nicht eingebrachter Aktualisierungen für einen Workload; die Klassenbreite beträgt 1. Die Anzahl der Vorkommen der Anfragen, die eine bestimmte Zahl fehlender Aktualisierungen vorweisen, ist auf der y-Achse abgetragen. Durch den Vergleich der allgemeinen Kurvenverläufe, der Streuung und der Zentrierung der Histogramme sind ausführliche Analysen möglich.

Einfluss der Prozesslänge

In einem ersten Experiment wurde die Auswirkung der Länge n des Datenproduktionsprozesses auf den Grad der Verzögerung der Aktualität untersucht. Der dazu verwendete Workload bestand aus 2.500 Anfragen und 1.000 Aktualisierungen, deren Kosten aus einer Normalverteilung stammten ($\mu = 100ms$, $\sigma = 20ms$). Unter Anwendung der *lokalen Ablaufplanung* wurde der Workload 20 mal ausgeführt und die Resultate wurden gemittelt. Die Länge des Datenproduktionsprozesses variierte dabei zwischen einer und fünf Stufen.
Das Ergebnis ist in Abbildung 5.4 in Form von fünf Histogrammen – eines für jede Stufe – dargestellt. Die Klasse eines Histogramms bezeichnet die für eine Anfrage vorkommende Anzahl nicht eingebrachter Aktualisierungen, deren Häufigkeiten in der y-Achse dargestellt sind. Es ist zu erkennen, dass in Stufe 1 der mehrheitliche Teil der Anfragen aktuelle Ergebnisse lieferte (0 uu) und nur relativ wenige Aktualisierungen nicht rechtzeitig propagiert werden konnten (bis zu 5 uu). Dies entspricht

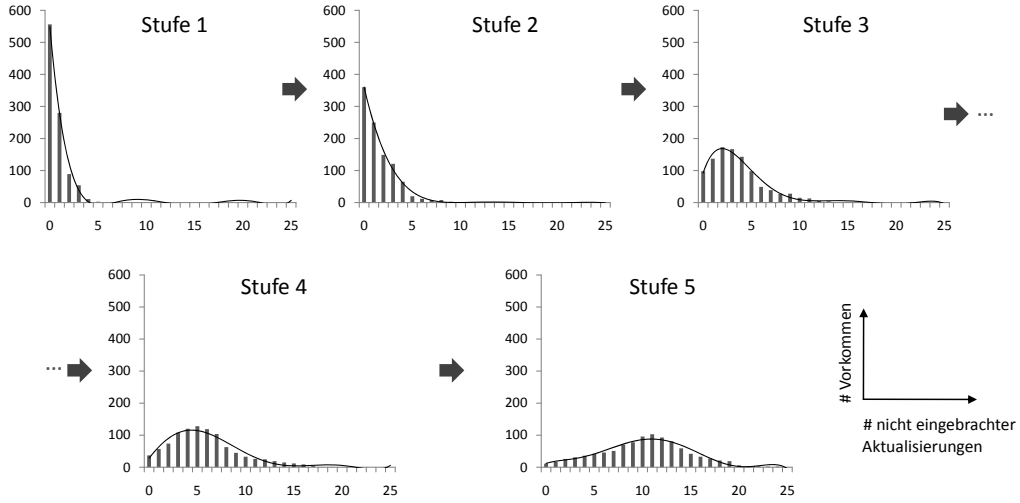


Abbildung 5.4: Anzahl der nicht eingebrachten Aktualisierungen in Abhängigkeit von der Länge des Datenproduktionsprozesses

den Ergebnissen aus Kapitel 4, in dem gezeigt werden konnte, dass die multikriterielle Ablaufplanung in einem einstufigen System bezüglich der Aktualität das optimale Ergebnis liefert.
Je später allerdings die einzelnen Stufen im Datenproduktionsprozess angesiedelt sind, desto schlechter wird die Datenaktualität. Für Anfragen, die in der letzten Stufe ausgeführt werden, können jeweils bis zu 20 Aktualisierungen nicht rechtzeitig eingebracht werden. Im Mittel schwankt dieser Wert um den Wert zehn. Der Grund für die Verminderung der Datenaktualität liegt in den zunehmenden Verweilzeiten der Aktualisierungen begründet, die durch jede weitere Prozessstufe erhöht werden. Aktualisierungen, die mit Anfragen in späteren Stufen korrelieren, werden durch frühere blockiert, welche nur auf Basis lokaler Nutzeranforderungen priorisieren. Die Datenaktualität sinkt somit mit zunehmender Länge des Datenproduktionsprozesses.

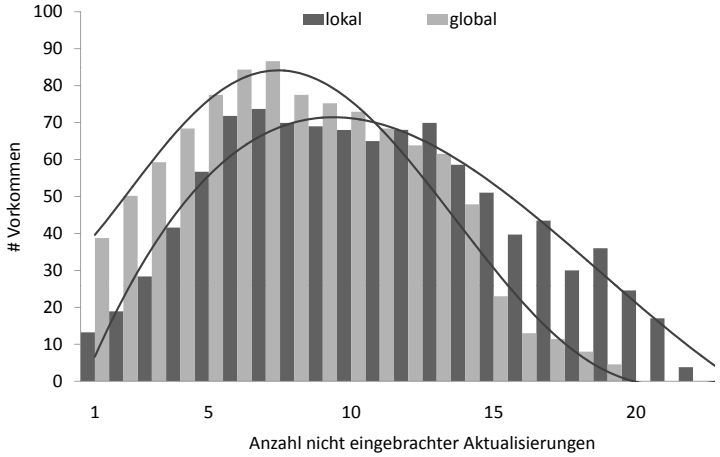
Vergleich zwischen lokaler und globaler Ablaufplanung

In dem vorangegangenen Experiment wurde die Schwäche der *lokalen Ablaufplanung* bei zunehmender Länge des Datenproduktionsprozesses nachgewiesen. Das folgende Experiment vergleicht dieses mit der *globalen Ablaufplanung*. Dazu wurde für beide Verfahren die Datenaktualität an Stufe 5 eines insgesamt fünfstufigen Prozesses untersucht. Alle an Stufe 5 geleitete Anfragen bekamen das höchste Service-Level zugewiesen. In Abbildung 5.5a ist das Ergebnis dargestellt. Es ist zu sehen, dass die *globale Ablaufplanung* zu einer besseren Verteilung der nicht eingebrachten Aktualisierungen führt als die *lokale Ablaufplanung*. Im Mittel hatte jede Anfrage unter

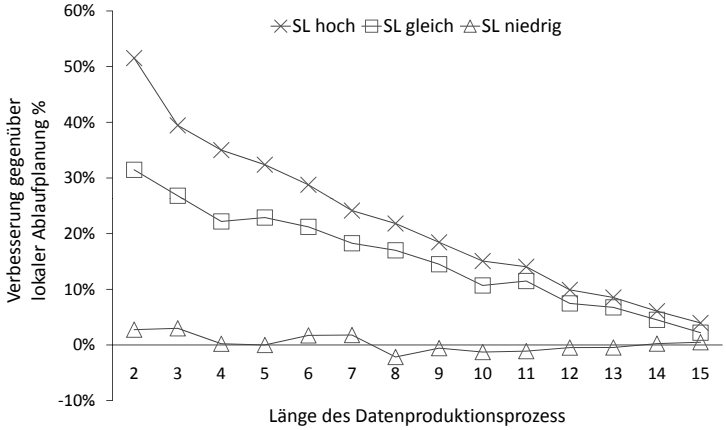
Verwendung der *lokalen Ablaufplanung* 10,7 uu, was durch die *globale Ablaufplanung* auf 8,1 uu und somit um zirka 30% verbessert werden konnte. Unter dem Gesichtspunkt jedoch, dass den Anfragen und damit auch den korrelierten Aktualisierungen die höchste Priorität zugewiesen war, ist eine Verbesserung um 30% vergleichsweise wenig. Welche konkreten Einflussgrößen hier eine Rolle spielen, wird in den Abschnitten 5.1.2 und 5.1.2 im Detail untersucht.
In einem zweiten Schritt wurde dieses Experiment generalisiert und die prozentuale Verbesserung der *globalen Ablaufplanung* gegenüber der *lokalen Ablaufplanung* in der n-ten Stufe eines n-stufigen Prozesses gemessen. Bei der Zuweisung der Service-Level wurden folgende Fälle unterschieden: 1) das höchste Service-Level an Anfragen der letzten Stufe (*SL hoch*), 2) das gleiche Service-Level für alle Anfragen (*SL gleich*) und 3) das höchste Service-Level für Anfragen der ersten Stufen (*SL niedrig*).
In Abbildung 5.5b sind die Ergebnisse für einen Datenproduktionsprozess steigender Prozesslänge dargestellt. Für einen zweistufigen Prozess kann eine Verbesserung um 50% und 30% bzw. keine Verbesserung für die Konfiguration *SL niedrig* beobachtet werden. Diese nimmt allerdings kontinuierlich mit der Länge des Prozesses ab und beträgt in einem 15-stufigen Prozess in der 15. Stufe nur noch 3%. Des Weiteren verlieren mit zunehmender Prozesslänge auch die Service-Level ihren Einfluss. In Stufe 5 ist durch Zuweisung des höchsten Service-Levels an Anfragen der fünften Stufe (*SL hoch*) nur noch eine Verbesserung von 7% im Vergleich zur Zuweisung des gleichen Service-Levels an alle Anfragen (*gleiche SL*) zu erzielen. Da durch die Priorisierung einer bestimmten Prozessstufe andere Stufen des Produktionsprozesses gleichzeitig niedriger priorisiert werden, ist der Einsatz von Service-Leveln ab einer bestimmten Prozesslänge nicht zu empfehlen. Der Grund dafür liegt in der zunehmenden Verzögerung mit steigender Prozesslänge. Die Aktualisierungen, die mit Anfragen in der 15. Stufe korrellieren, werden zwar priorisiert behandelt, können jedoch durch die Verzögerung in den vorhergehenden 14 Stufen nicht rechtzeitig propagiert werden.
Beide Experimente haben gezeigt, dass der Zuwachs an Aktualität, der durch Anwendung einer *globalen Ablaufplanung* erzielt werden kann, begrenzt ist, d.h. die *globale Ablaufplanung* lohnt nur für relativ kurze Datenproduktionsprozesse. Auch der Einfluss unterschiedlicher Service-Level ist durch die Länge des Prozesses beschränkt. In den folgenden Experimenten werden die begrenzenden Faktoren betrachtet und Empfehlungen abgeleitet, welche Ablaufplanung unter welchen Bedingungen einzusetzen ist.

Einfluss stufenkonkurrierender und langlaufender Aktualisierungen

Eine Annahme der bisherigen Untersuchungen bestand darin, die Anfrage- und Aktualisierungskosten gleichzusetzen. Daher ist im Folgenden zu untersuchen, wie die Qualität der Ablaufplanung sich bei steigenden Aktualisierungskosten verändert. Die Abbildungen 5.6a, 5.6b, 5.6c und 5.6d fassen die Ergebnisse durch eine Visualisierung der Anfrageaktualitäten zusammen. Die Experimentierumgebung wird anhand der Abbildungen erläutert: Dargestellt sind 100 Anfragen in 5 Warteschlangen bzw. Prozessstufen. Der Kopf der Warteschlange befindet sich am unteren Ende der Ab-



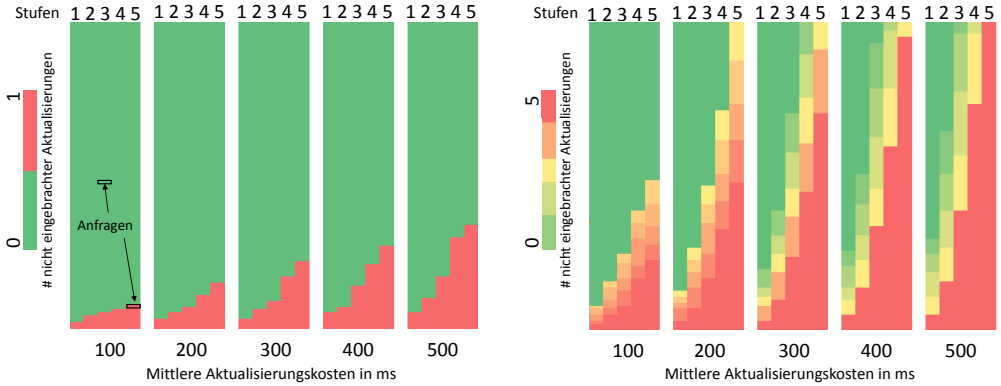
(a) Lokale vs. globale Ablaufplanung



(b) Abnehmende Verbesserung mit zunehmender Prozesslänge

Abbildung 5.5: Vergleich der lokalen und globalen Ablaufplanung

bildung. Jede Anfrage ist farblich kodiert (die entsprechende Legende dazu befindet sich jeweils links der Abbildungen). Die Färbung einer Anfrage repräsentiert die Anzahl der nicht eingebrachten Aktualisierungen und somit die Aktualität, die diese Anfrage bei ihrer Ausführung im besten Fall haben wird. Die Service-Level der Anfragen wurden aufsteigend vergeben, d.h. Anfragen der fünften Stufe haben das höchste und Anfragen der ersten Stufe das niedrigste Service-Level. Aktualisierungen, die mit Anfragen höherer Stufen korrellieren, sind daher höher priorisiert. Die Anfragekosten wurden, wie in den vorhergehenden Experimenten, aus einer Normalverteilung gezogen ($\mu = 100ms$, $\sigma = 20ms$), wohingegen die Aktualisierungskosten variieren (μ =100ms bis 500ms).
Im ersten Experiment (siehe Abbildung 5.6a) korreliert eine Anfrage stets genau mit einer Aktualisierung. Kann diese Aktualisierung vor Ausführung der Anfrage



(a) Jede Anfrage korreliert mit einer Aktualisierung

(b) Jede Anfrage korreliert mit fünf Aktualisierung

(c) Jede Anfrage korreliert mit zehn Aktualisierungen

(d) Jede Anfrage korreliert mit 20 Aktualisierungen

Abbildung 5.6: 100 Anfragen in 5 Stufen mit steigenden Aktualisierungskosten

bis zur entsprechenden Stufe propagiert werden, so ist die Aktualität für die Anfrage maximal (grüne Markierung in Abbildung 5.6a). Die Korrelation der Anfragen der einzelnen Stufen mit der Menge der jeweiligen Aktualisierungen ist disjunkt, d.h. alle Anfragen an Stufe 1 korrelieren mit einer Menge von Aktualisierungen U_1, alle Anfragen an Stufe 2 mit U_2 aber nicht mit U_1 usw. Werden die Aktualisierungskosten von 100 ms auf 500 ms erhöht, so ist zu sehen, dass für einen zunehmenden Teil der Anfragen (rot markiert) die korrelierte Aktualisierung nicht rechtzeitig propagiert werden kann. Dies betrifft im konkreten Fall bei Aktualisierungskosten von 500 ms in Stufe 5 rund ein Drittel aller Anfragen.
In den weiteren Experimenten ist jede Anfrage mit 5, 10 bzw. 20 Aktualisierungen korreliert, die stufenübergreifend ebenfalls wieder disjunkt sind. Die Aktualisierungskosten werden wiederum schrittweise von 100 ms auf 500 ms erhöht. Es ist zu sehen, dass in Stufe 5 bei Aktualisierungskosten von 500 ms stets alle Anfragen bei ihrer

Ausführung die geringste Aktualität besitzen, d.h. 5, 10 bzw. 20 fehlende Aktualisierungen aufweisen. In den Experimenten mit 10 bzw. 20 korrelierten Aktualisierungen (Abbildungen 5.6c und 5.6d) gilt das sogar bereits für Aktualisierungskosten von nur 200 bzw. 300 ms.
Zusammengefasst lässt sich sagen, dass für alle Anfragen, die in den Abbildungen 5.6a bis 5.6d rot dargestellt sind, es nicht möglich ist, die Datenaktualität zu verbessern, ohne diese Anfragen zusätzlich verzögern zu müssen. Dieser Anfragetyp wird als *nicht optimierbar* bezeichnet. Die Anzahl der nicht optimierbaren Anfragen erhöht sich 1) mit dem sich verschlechternden Kostenverhältnis zwischen Anfragen und Aktualisierungen, 2) mit zunehmender Anzahl stufenübergreifender disjunkter Aktualisierungen und 3) wie in Abschnitt 5.1.2 bereits gezeigt, mit zunehmender Länge des Datenproduktionsprozesses. Die Datenaktualität der nicht optimierbaren Anfragen verhält sich somit invariant gegenüber dem verwendeten Scheduling-Verfahren, d.h. die *globale Ablaufplanung* kann die Aktualität dieser Anfragen nicht weiter verbessern. Mit zunehmender Anzahl nicht optimierbarer Anfragen in einem Workload verschlechtert sich auch die Leistung der *globalen Ablaufplanung* im Vergleich zur *lokalen Ablaufplanung*. In Abschnitt 5.1.2 wird dieser Aspekt detaillierter betrachtet und die *globale Ablaufplanung* dahingehend erweitert, dass die nicht optimierbaren Anfragen bei der Ablaufplanung nicht berücksichtigt werden.

Anteil der stufenkonkurrierenden Aktualisierungen

Die Annahme im vorherigen Experiment bestand darin, dass die Menge der Aktualisierungen, die in die einzelnen Stufen eingebracht werden, völlig disjunkt voneinander sind. Dies ist in praktischen Szenarien oft nicht der Fall, sondern stattdessen werden Anfragen unterschiedlicher Stufen oft gemeinsam von einer Aktualisierung profitieren.
In diesem Experiment wurden die Kosten der Anfragen und Aktualisierungen gleichgesetzt, jedoch jeweils die Überlappungen bzw. Disjunktheit der Aktualisierungsmengen variiert. In Abbildung 5.7 ist dies symbolisch durch die Venn-Diagramme dargestellt. Bei einem Anteil stufenübergreifender Aktualisierungen von 0% korrelieren die Anfragen aller Stufen mit der gleichen Aktualisierungsmenge. Dieser Anteil nimmt in 5%-Schritten zu, bis schließlich die Anfragen jeder Stufe mit einer jeweils disjunkten Aktualisierungsmenge korrelieren. Die Länge des Datenproduktionsprozesses wurde auf 5 festgesetzt. Den Anfragen an Stufe 5 wurde das höchste Service-Level zugewiesen. Für jede Abstufung der Disjunktheit wurde die lokale und die globale Ablaufplanung angewendet und die mittlere Anzahl der nicht eingebrachten Aktualisierungen an Stufe 5 gemessen (siehe Abbildung 5.7). Bei einer Disjunktheit von 0% erzielen beide Scheduling-Algorithmen das gleiche Ergebnis. Da die Anfragen aller Stufen mit der gleichen Menge an Aktualisierungen korrellieren, ist durch die gesamtheitliche Betrachtung kein Mehrwert zu erzielen. Mit zunehmender Disjunktheit kann sich allerdings die *globale Ablaufplanung* gegenüber der *lokalen Ablaufplanung* sukzessiv verbessern. Bei völlig disjunkten Aktualisierungsmengen ist das Ergebnis der *globalen Ablaufplanung* um 40% besser im Vergleich zur *lokalen*

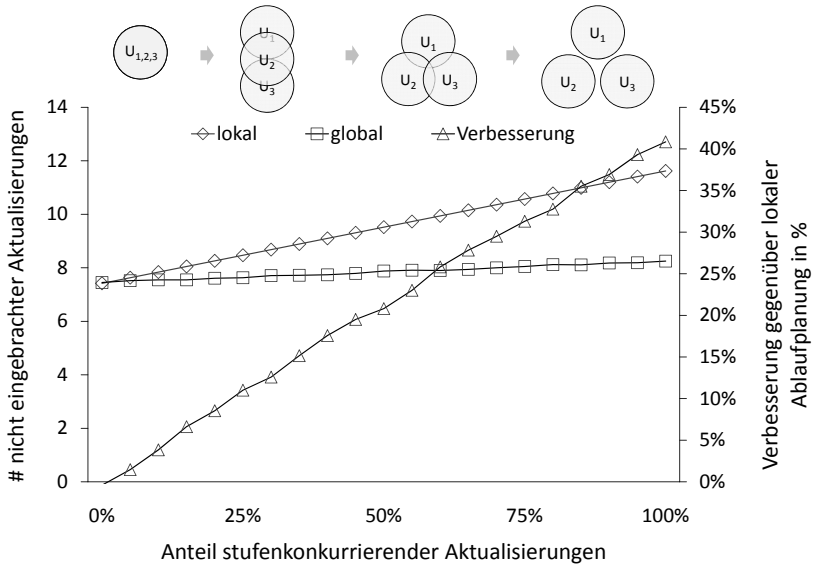


Abbildung 5.7: Variierung des Anteils stufenkonkurrierender Aktualisierungen

Ablaufplanung. Die Berücksichtigung aller Nutzeranforderungen innerhalb des Datenproduktionsprozesses ist also nur bei wenig überlappenden bis hin zu disjunkten Aktualisierungsmengen gewinnbringend.

Optimierungsansatz

In einem vorangegangenen Abschnitt wurde nachgewiesen, dass es für die sogenannten nicht optimierbaren Anfragen nicht möglich ist, die Datenqualität zu verbessern. Die Priorisierung der mit diesen Anfragen korrellierten Aktualisierungen verursacht daher Kosten, erzeugt jedoch keinen Profit. Diese Eigenschaft ist für eine Optimierung der *globalen Ablaufplanung* ausnutzbar. Dazu werden bei der Priorisierung von Aktualisierungen Korrelationen mit diesen Anfragetypen unberücksichtigt gelassen. Dadurch wird der Anteil der Aktualisierungen verringert, die hoch priorisiert werden, die jedoch aufgrund der in 5.1.2 beschriebenen Verzögerung nicht rechtzeitig propagiert werden können. Um die Auswirkung der Optimierung zu evaluieren, wurde in einem fünfstufigen Prozess die Datenaktualität in Stufe 1 und 5 sowohl mit als auch ohne Optimierung ermittelt (siehe Abbildung 5.8). Es ist zu sehen, dass durch die Optimierung die Datenaktualität in Stufe 5 nur sehr geringfügig verschlechtert wird, die in Stufe 1 aber um fast 60% verbessert werden kann. Der Effekt, dass eine Verbesserung der Datenaktualität in einer Stufe i zu einer Verschlechterung der Datenaktualität in allen vorhergehenden Stufen kleiner i führt, kann somit durch die Optimierung verringert werden. Im Kontext der Pareto-effizienten Ablaufplanung aus Kapitel 4 ist der vorgeschlagene Optimierungsansatz als Vorverarbeitungsschritt

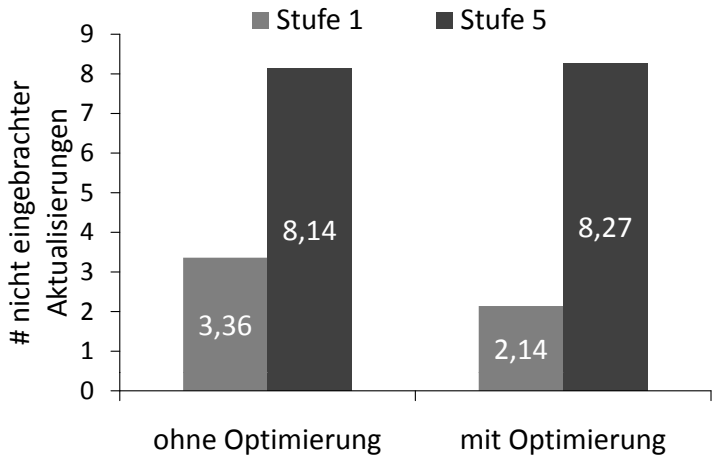
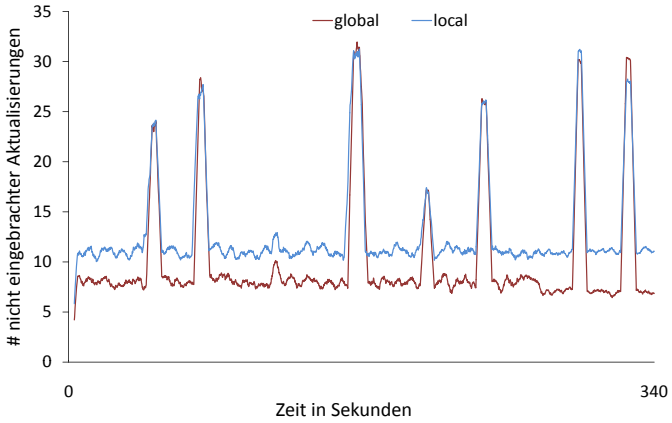


Abbildung 5.8: Optimierung der globalen Ablaufplanung

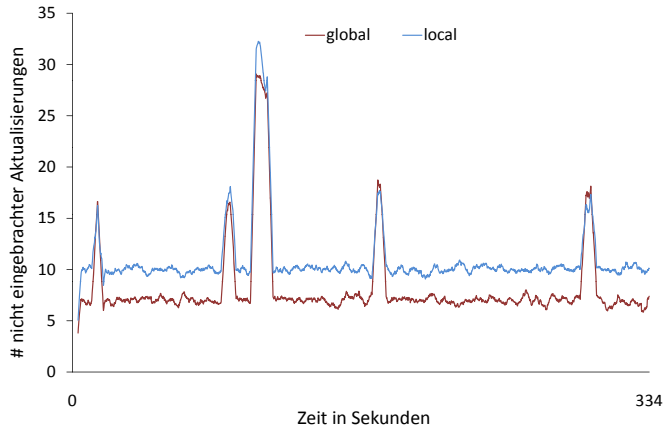
in die Erzeugung der Aktualisierungselemente (siehe Abschnitt 4.2.2) zu integrieren. Die dort angegebene Abhängigkeitsmatrix zwischen Aktualisierungen und Anfragen verkleinert sich durch den Optimierungsansatz, was in weniger Aktualisierungselementen und somit in einer geringeren Laufzeit des Rucksackalgorithmus resultiert (siehe Abschnitt 4.5.3).

Auswirkung langlaufender Anfragen und Aktualisierungen zur Laufzeit

Der negative Effekt hoher Aktualisierungskosten auf die Ergebnisse der *globalen Ablaufplanung* wurde in den letzten Experimenten evaluiert. In den folgenden Experimenten wurde das Laufzeitverhalten für Workloads mit langlaufenden Anfragen und Aktualisierungen untersucht.
Im ersten Experiment wurden die Kosten der 2.500 Anfragen aus einer Normalverteilung gezogen ($\mu = 100ms$, $\sigma = 20ms$) und die Kosten der 1.000 Aktualisierungen aus einer Pareto-Verteilung mit einem Minimalwert von 50 ms und einem Alpha von 0,8, wobei die Kosten nach oben auf 5.000 ms beschränkt waren. In Abbildung 5.9 ist das Ergebnis über die Simulationszeit hinweg dargestellt. Die Anzahl der nicht eingebrachten Aktualisierungen wurde dabei über ein Fenster bestehend aus 25 Anfragen gemittelt. Wie durch die vorherigen Experimente bestätigt, ist für ein ausgeglichenes Kostenverhältnis die Aktualität der Anfragen für die *globale Ablaufplanung* konstant besser als für die *lokale Ablaufplanung*. Werden jedoch langlaufende Aktualisierungen ausgeführt, so verschlechtern sich die lokale und die globale Ablaufplanung im gleichen Maße und nähern sich in ihrer Leistung einander an. Der Grund hierfür ist, dass durch die Ausführung teurer Aktualisierungen die Verarbeitung anderer Transaktionen stark verzögert und damit die zeitliche Lokalität gestört wird. Die Betrachtung der Korrelationen über alle Stufen hinweg ist in solchen Fällen nicht gewinnbringend.



(a) Pareto-verteilte Aktualisierungskosten



(b) Pareto-verteilte Anfragekosten

Abbildung 5.9: Auswirkung langlaufender Aktualisierungen und Anfragen zur Laufzeit

In einer weiteren Untersuchung wurde das gleiche Experiment, jedoch mit konstant gehaltenen Aktualisierungskosten und Pareto-verteilten Anfragekosten durchgeführt. Das Ergebnis ist in Abbildung 5.9b dargestellt. Im Mittel ist die *globale Ablaufplanung* analog zum ersten Experiment konstant besser als die *lokale Ablaufplanung*. Werden jedoch langlaufende Anfragen verarbeitet, so führt dies zum gleichen Ergebnis wie oben. Die teuren Anfragen blockieren den Datenproduktionsprozess, wodurch der zeitliche Bezug zwischen einer Anfrage und der mit ihr korrelierten Aktualisierungsmenge unterbrochen wird.

5.2 Visualisierung der Datenqualität in mehrstufigen Datenproduktionsprozessen

Wie bereits in der Einleitung dieses Kapitels aufgeführt, durchlaufen die Daten in einem Data-Warehouse-System einen komplexen Prozess aus Lade-, Transfer- und Transformationsoperationen. Die Teilprozesse sind zeitlich und organisatorisch meist voneinander entkoppelt, so dass durch jede Operation die Verweilzeit der Daten an den Zwischenstationen erhöht und damit die Aktualität gemindert wird. Zu einem gegebenen Abfragezeitpunkt stehen dem Data-Warehouse-Nutzer somit unter Umständen nur veraltete Informationen zur Verfügung, da noch nicht alle Daten den Produktionsprozess vollständig durchlaufen haben und freigegeben wurden.
Zur Gewährleistung bzw. zur Verbesserung der Datenaktualität und anderer Qualitätsdimensionen muss eine Visualisierung zur Überwachung und Analyse der Datenproduktion entwickelt werden. Anhand der Analyseergebnisse kann eventuelles Potenzial zur Optimierung des Datenproduktionsprozesses aufgedeckt und damit können Zwischenverweilzeiten minimiert werden.

5.2.1 Erfassung und Speicherung

Zur ganzheitlichen Analyse der Datenproduktionsprozesse müssen in jeder Prozessstufe Änderungen der Datenaktualität protokolliert und gespeichert werden. Setzt man die Prozessstufen in Beziehung zueinander, wird so ein direkter Vergleich der Stufen möglich. Die Granularität, in der die Prozessstatusdaten gespeichert werden, ist durch das Partitionierungsschema des Data-Warehouse-Systems vorgegeben (siehe Abschnitt 4.1.4). Dies ist am folgenden Beispiel, einem dreidimensionalen Datenschema, illustriert:

$$Zeit \Rightarrow \{Jahr \mapsto Quartal \mapsto Monat\}$$

$$Filiale \Rightarrow \{Region \mapsto Land \mapsto Stadt \mapsto Name\}$$

$$Produkt \Rightarrow \{Kategorie \mapsto Familie \mapsto Marke \mapsto Name\}.$$

Ist bekannt, dass die eintreffenden Daten zeitlich nach Quartalen sowie nach Produktfamilie prozessiert und eingepflegt werden, so erfolgt eine Partitionierung der Daten bezüglich dieser beiden Dimensionsebenen:

$$Zeit \Rightarrow \{Jahr \mapsto Quartal\}$$

$$Produkt \Rightarrow \{Kategorie \mapsto Produktfamilie\}.$$

Im vorangegangenen Kapitel konnten durch diese Vereinfachung Korrelationen zwischen Anfragen und Aktualisierungen festgestellt werden. Im vorliegenden Kontext werden die Datenaktualitätsinformationen aggregiert und auf Ebene der Partitionierung gespeichert. Dazu wird für jede Prozessstufe eine Tabelle angelegt, die das Partitionierungsschema implementiert und als Fakten die Datenqualitätsinformationen speichert (Schritt (1) in Abbildung 5.10). Am Beispiel der Aktualitätsmetriken

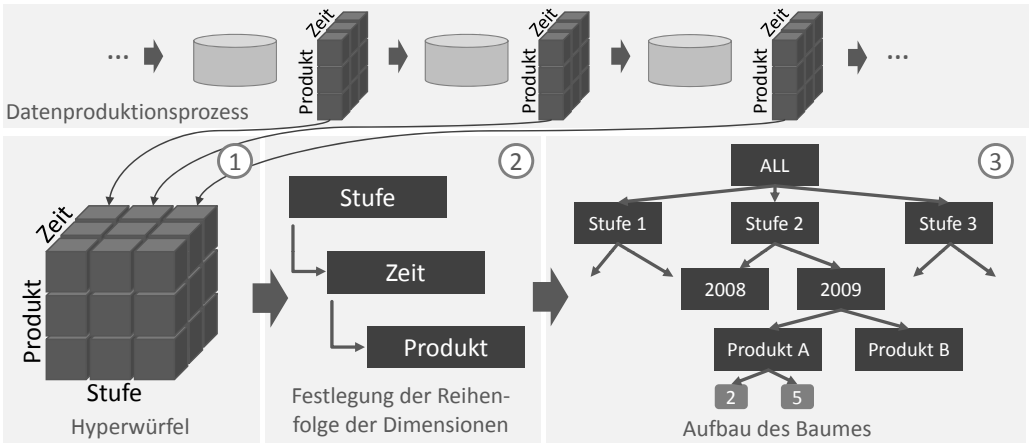


Abbildung 5.10: Vorgehensweise der Visualisierung von Datenproduktionsprozessen

aus Abschnitt 4.1.2 ist dies entweder die Anzahl der eingebrachten Aktualisierungen (unter Verwendung der abstandsbasierten Metrik) oder der Zeitstempel der letzten Aktualisierung (unter Nutzung der zeitbasierten Metrik).
Es bleibt anzumerken, dass die Granularität der Partitionierung stets ein Kompromiss zwischen dem Detailgrad der Kennzahlen und dem Aufwand zur Speicherung und Wartung dieser Datenstruktur ist. ´

5.2.2 Visualisierung der Datenqualität

Ausgehend von der Tatsache, dass das visuelle Wahrnehmungssystem des Menschen hocheffiziente Fähigkeiten zur Mustererkennung und zur Erfassung komplexer, ganzheitlicher Zusammenhänge bildlicher Strukturen besitzt, ist es naheliegend, diese zu nutzen und die Daten graphisch darzustellen. Die Visualisierung unter Verwendung herkömmlicher Darstellungsmittel, wie Balken-/Säulen-/Tortendiagramme etc., ist jedoch ab drei und mehr Dimensionen nicht mehr adäquat, so dass zumeist die Metapher eines Baums verwendet wird, dessen Knoten die Daten und dessen Kanten die Beziehung der Dimensionshierarchien repräsentieren.
Die Anforderungen bei der Darstellung von Baumstrukturen sind zum einen Kompaktheit und zum anderen visuelle Skalierbarkeit. Kompaktheit meint vor allem die platzsparende Repräsentation und die optimale Ausnutzung des rechteckigen Bildschirms. Die Anforderung der Skalierbarkeit bringt zum Ausdruck, dass mit zunehmender Datenmenge bei der Navigation von der Baumwurzel zu den Blättern die Darstellungsform immer noch zweckmäßig sein muss. Des Weiteren ist es aus Gründen der Komplexität nicht förderlich, alle Daten zugleich und parallel in ihrer feinsten vorliegenden Granularität darzustellen. Vielmehr sollte die graphische Darstellung des Baums bzw. des Datenwürfels den Prinzipien des Online Analytical Processing (OLAP) folgen: Dem Analysten ist also die Möglichkeit zu geben, zwischen

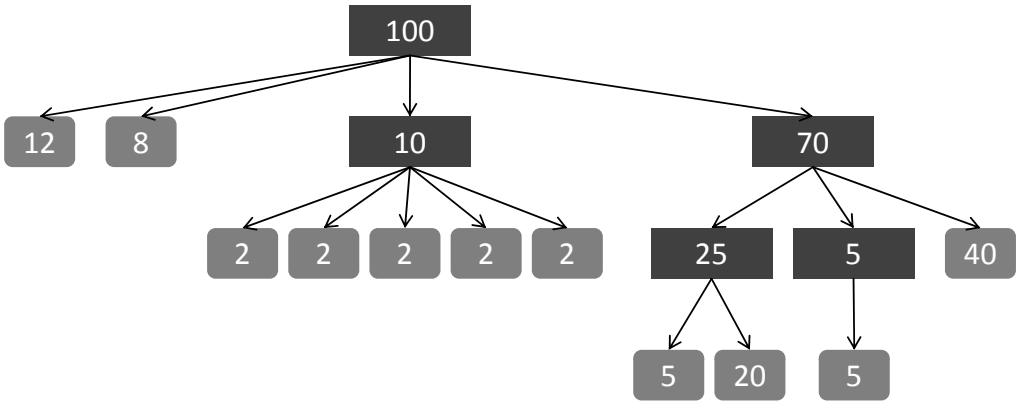


Abbildung 5.11: Beispiel einer Baumstruktur nach [67]

verschiedenen Detailstufen zu wechseln, sich zuerst einen aggregierten Überblick über die Daten zu verschaffen und explorativ entlang der Dimensionshierarchien auf detailliertere Ebenen der graphischen Darstellung zu navigieren. Weiterhin muss die Visualisierung die Darstellung von Kennzahlen (z.B. Aktualität) ermöglichen und den Anwender bei deren Unterscheidung durch Verwendung verschiedener Farben, Flächen oder Formen unterstützen. ´

Tree-Map

Eine Visualisierungsform, die all diese Forderungen adressiert, ist die Tree-Map, vorgestellt in [67]. Diese erlaubt die Darstellung einzelner Dimensionsebenen als auch die gleichzeitige Ansicht verschiedener Granularitäten. Durch die rechteckige Form ist sie besonders kompakt und skaliert auch mit großen Datenmengen. Des Weiteren unterstützt sie viele der aus dem OLAP-Bereich bekannten Navigationsoperationen. Kernpunkt des Verfahrens ist die platzausfüllende Darstellung einer Baumstruktur in einem rechteckigen, zweidimensionalen Zeichenbereich. Die Blätter des Baums bestehen aus einem numerischen Attribut oder sind auf ein solches abbildbar; die Knoten beinhalten die Aggregationswerte der jeweiligen Kinder, mit der Wurzel als Superaggregat (Beispiel siehe Abbildung 5.11). Der Algorithmus beginnt bei der Wurzel des Baums und unterteilt den Zeichenbereich in Teilrechtecke, die in ihrer Größe proportional zu den Attributwerten der Kinder sind. Diese werden so in den Zeichenbereich eingefügt, dass der zur Verfügung stehende Platz vollständig ausgenutzt wird. Anschließend steigt der Algorithmus rekursiv entlang der Zweige bis zu den Blättern ab und partitioniert die lokalen Teilrechtecke wiederum jeweils proportional zu den Attributwerten der jeweiligen Kinder. Die Größe der Tree-Map-Flächen ist somit äquivalent zu den Werten der zugrundliegenden Kennzahlen und Abweichungen sind leicht erkennbar. Zusätzlich kann dies zur besseren Unterscheidung noch durch Farb- oder Grauabstufungen unterstützt werden. Dazu wird der Wertebereich des

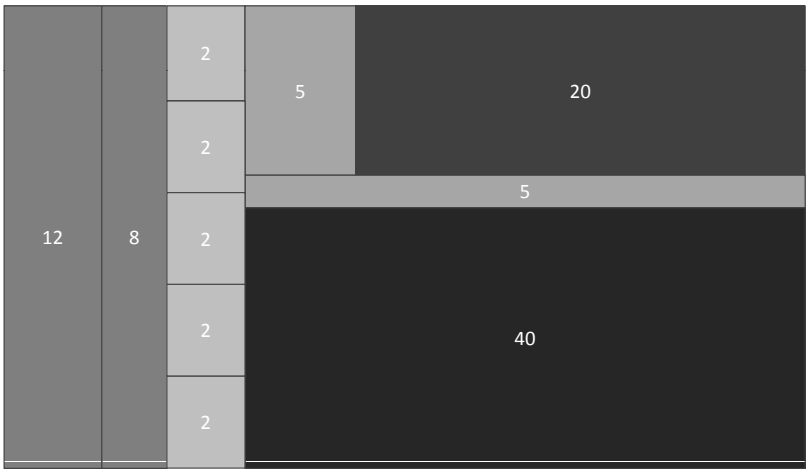


Abbildung 5.12: Tree-Map für o.g. Baumstruktur [67]

zur Einfärbung genutzten Attributs auf den Wertebereich der gewünschten Farbskala abgebildet. Alternativ kann die farbliche Repräsentation auch zur Darstellung eines zweiten Attributs verwendet werden. Dieses wird ebenfalls nach denselben Aggregationsvorschriften wie das Partitionierungsattribut in den Blättern und Knoten der Baumstruktur abgelegt und mitgeführt (siehe Abbildung 5.12).

Abbildung des Hyperwürfels auf die Tree-Map

Zur Visualisierung des Hyperwürfels muss dieser zunächst in eine Baumstruktur transformiert und anschließend dem Tree-Map-Algorithmus übergeben werden. Dazu wird eine Sicht auf den Datenwürfel erzeugt, welche die Hierarchiefolge der Dimensionen zueinander festlegt (Schritt (2) in Abbildung 5.10). Die Reihenfolge der Dimensionen ist nicht starr, sondern kann auch während der Analyse verändert werden, um eine andere Sicht auf die Daten zu erlangen.
Der Würfel in seiner Gesamtheit bildet die Wurzel des Baumes (Schritt (3)). Daran schließen sich als Kindknoten die Ausprägungen der ersten Dimension gemäß der gewählten Hierarchiereihenfolge an. An jedem dieser Knoten werden stufenweise die Ausprägungen der nächsten Dimension als Kindknoten angehängt. Die Ausprägungen der untersten Hierarchiedimension bilden schließlich die Blätter, welche die Kennzahlen zur Beschreibung der Datenqualität beinhalten. Die Partitionierung sowie die Einfärbung der Zeichenfläche geschieht wahlweise mit denselben oder verschiedenen Kennzahlen.
Die Aggregationsfunktion, welche für die Werte in den inneren Knoten verwendet wird, ist abhängig von der Semantik der darzustellenden Kennzahl. Wird zur Bewertung der Datenqualität eine zeitbasierte Metrik verwendet (siehe Abschnitt 4.1.2), so ist zur Aggregation die Maximum-Funktion geeignet, d.h. das Alter des Vaterknotens entspricht dem des ältesten Kindknotens. Wird eine abstandsbasierte Metrik

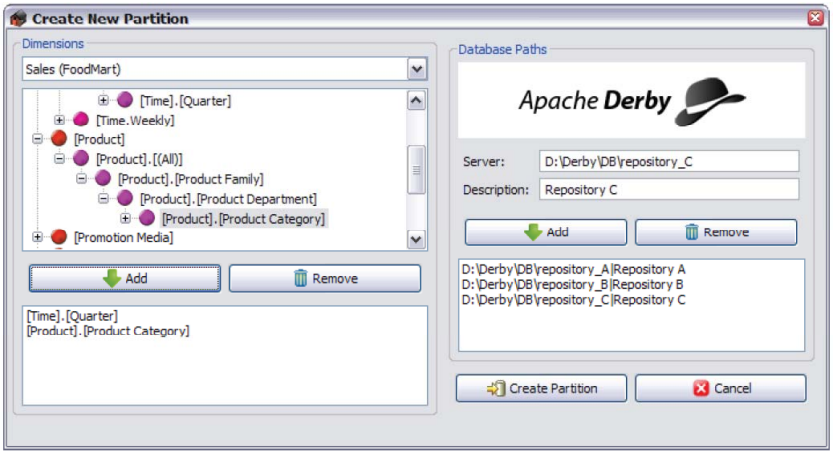
verwendet, qualifizieren sich sowohl der Mittelwert als auch das Maximum als Aggregationsfunktion.

OLAP-Operationen

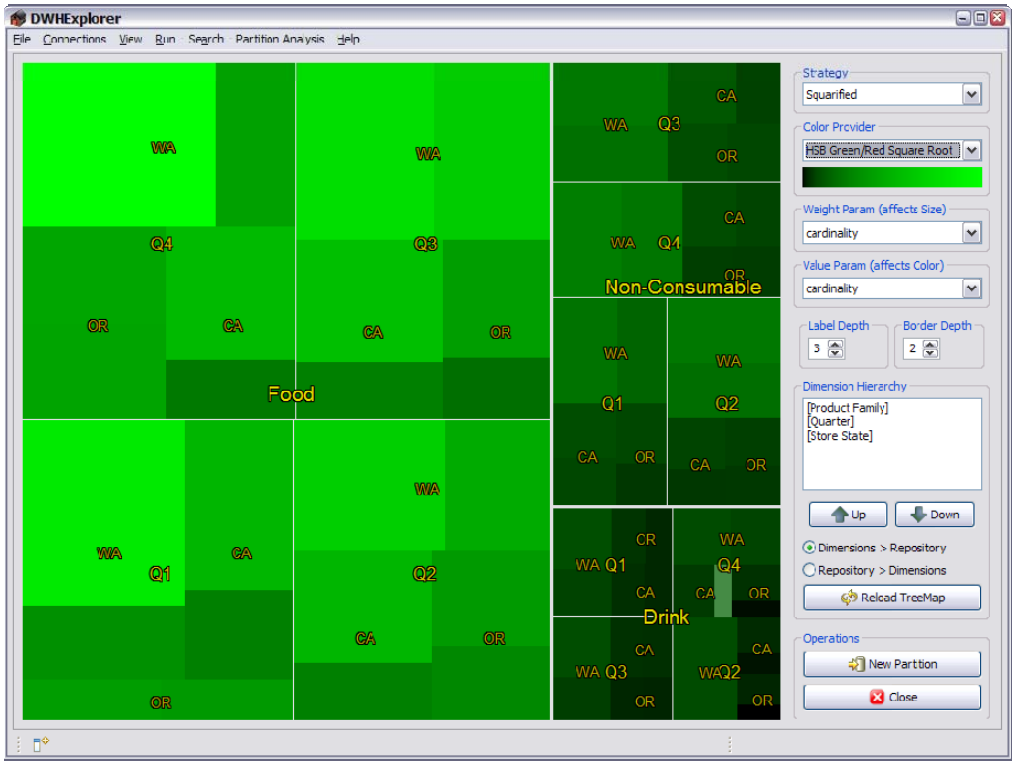
Die Tree-Map bzw. das in Abbildung 5.10 dargestellte Vorgehensmodell unterstützen eine Reihe von OLAP-Operationen: Die Pivotierung, d.h. die Rotation der Ansicht, kann durch eine Vertauschung der Dimensionsreihenfolge in Schritt (2) erfolgen. Ebenso kann durch das Weglassen einer Dimension eine Slice-Operation realisiert werden. Beim Drill-down, d.h. der Auswahl einer bestimmten Fläche, wird der entsprechende innere Knoten in Schritt (3) selektiert und der darunter liegende Teilbaum, mit diesem inneren Knoten als neue Wurzel, dem Tree-Map-Algorithmus übergeben. Ein Roll-up erfolgt analog durch Aufhebung der Selektion.

5.2.3 Prototypische Umsetzung

Das vorgestellte Verfahren wurde prototypisch, unter Verwendung der Eclipse Rich Client Platform (RCP), implementiert. Zur Abbildung des Datenproduktionsprozesses kamen mehrere Apache-Derby-Datenbanken [100] zum Einsatz, welche zentral an einem Mondrian-ROLAP-Server [Mondrian] registriert wurden. Für die Tree-Map-Visualisierung wurde die SWT-Komponente JTreeMap von ObjectLab [JTMap] verwendet. Diese ist quelloffen und steht unter der Apache Software License sowie der Eclipse Public License. Die JTreeMap-Klassen wurden für die vorliegende Anwendung teilweise abgeleitet und erweitert. In dem zu Illustationszwecken verwendeten Schema sind Verkäufe nach den Dimensionen Produkt, Zeit, Geschäft, Promotion usw. hinterlegt. Das Partitionierungsschema bzw. die Dimensionsreihenfolge, unter deren Verwendung die Tree-Map erstellt werden soll (Schritt (2) in Abbildung 5.10), wird in einem Dialogfenster (siehe Abbildung 5.13a) festgelegt. In diesem Beispiel ist dies in absteigender Ordnung das Quartal der Zeitdimension, die Produktfamilie der Produktdimension, der Bundesstaat in der Geschäftsdimension sowie die Produktionsstufen in der untersten Dimensionsebene (zwei Stufen in diesem Beispiel). Das Ergebnis der Visualisierung ist in Abbildung 5.13b dargestellt: Die Daten sind in drei Produktfamilien partitioniert, „Food“, „Drink“ und „Non-Consumable“, die jeweils in vier Quartale und drei Bundesstaaten unterteilt sind. Für die Ausprägung „Food“, „Q4“ und „WA“ ist gut zu sehen, dass die Datenqualitätsmetriken der beiden Stufen stark voneinander abweichen. Dies ist zum einen durch die verschieden großen Flächen als auch durch die unterschiedliche Einfärbung hervorgehoben. Ebenso sind andere Partitionen zu erkennen, deren Datenqualität nicht oder nur kaum differiert. Bei der Ebenenbeschriftung sind die kompakte und die informative Variante zu unterscheiden. Bei Ersterer sind nur die Blätter beschriftet, bei Zweiterer alle Dimensionsebenen, d.h. auch innere Knoten. Je nach Umfang der darzustellenden Daten können Beschriftungsebenen hinzugefügt bzw. weggelassen werden.
Die graphische Darstellung als Tree-Map bietet somit eine geeignete Visualisie-



(a) Bestimmung der Hierarchiereihenfolge



(b) Treemap-Darstellung

Abbildung 5.13: Prototypische Umsetzung der Visualisierung von Datenproduktionsprozessen

rung der Kennzahlen eines Datenproduktionsprozesses, unterstützt deren explorative Analyse und lässt Rückschlüsse auf mögliche Engpässe und Optimierungen zu.

5.3 Zusammenfassung

In diesem Kapitel wurden zwei zentrale Fragestellungen im Kontext mehrstufiger Datenproduktionsprozesse untersucht: Die erste befasste sich mit der Integration der in Kapitel 4 entwickelten multikriteriellen Ablaufplanung in mehrstufige Prozesse. Insbesondere wurde untersucht, ob und unter welchen Bedingungen eine Ablaufplanung in Datenproduktionsprozessen anwendbar ist. Zur Beantwortung der Frage wurden zwei Extremfälle – die lokale und die globale Ablaufplanung – definiert und in einer Reihe von Experimenten untersucht und miteinander verglichen. Die mit der globalen Ablaufplanung zu erreichende Aktualität war nicht oder nur wenig besser als die Ablaufplanung auf Basis lokaler Informationen. Nur unter sehr extremen Annahmen, wie sehr kurzen Datenproduktionsprozessen, disjunkten Aktualisierungsmengen und ungleich verteilten Service-Leveln, konnte durch die globale Ablaufplanung ein signfikanter Mehrwert erzeugt werden. Dies ist vor allem durch die Störung der zeitlichen Lokalität, zwischen Anfragen und Aktualisierungen in Prozessen ab einer bestimmten Länge begründet. Im Allgemeinen lässt sich daher die Empfehlung aussprechen, für jede Prozessstufe eine individuelle Ablaufplanung zu verwenden. Die für Datenproduktionsprozesse typische lose Kopplung der einzelnen Prozessstufen kann somit beibehalten werden. Lediglich in Einzelfällen ist zu untersuchen, ob ein hybrider Ansatz, d.h. die gemeinsame Ablaufplanung eines Teils des Produktionsprozesses, von Nutzen sein kann.
Die zweite Fragestellung befasste sich mit der Analyse komplexer Prozesse durch eine Visualisierung der Datenqualität im Allgemeinen bzw. der Aktualität im Speziellen. Durch Anwendung des Partitionierungsschemas auf die Datenbanken der einzelnen Prozessstufen können die Datenqualitätsinformationen in einer vereinheitlichten Datenstruktur erfasst und gesammelt werden. Das Partitionierungsschema ist auch Ausgangspunkt für die graphische Darstellung der Prozessinformationen. Die partitionierenden Dimensionen werden gemeinsam mit den Faktattributen, welche die Datenqualität beschreiben, in eine Baumstruktur überführt und anschließend in Form einer Tree-Map visualisiert. Diese ist aufgrund ihrer Kompaktheit und Skalierbarkeit besonders für die Darstellung der vorliegenden multidimensionalen und hierarchischen Daten qualifiziert. Des Weiteren wurde die Tree-Map um Standard-OLAP-Operationen erweitert, so dass dem Nutzer ein Werkzeug zur Verfügung steht, mit dem explorativ Datenproduktionsprozesse auf ihr Optimierungspotenzial hin untersucht werden können.

Fallstudiendiskussion

Wie bereits dargestellt, unterscheiden sich die Datenproduktionssprozesse der GfK Marketing Services und der UBS WM&SB wesentlich in der Anzahl der Verarbeitungsstufen. Im Allgemeinen muss zwar im Einzelfall entschieden werden, inwiefern

eine globale Ablaufplanung lohnend eingesetzt werden kann, jedoch ist diese für das Szenario der GfK Marketing Services wahrscheinlich meist nicht zweckmäßig. Zum einen ist die Anzahl der Verarbeitungsstufen – mit über 60 an der Zahl – den Experimenten aus Abschnitt 5.1.2 zufolge zu hoch. Zum anderen sind in vielen Prozessstufen manuelle Verarbeitungsschritte bzw. Qualitätsüberprüfungen notwendig, wodurch die Verzögerung zusätzlich stark erhöht wird (siehe Abschnitt 5.1.2). Diese beiden Eigenschaften stören den zeitlichen Bezug zwischen Anfragen und Aktualisierungen, so dass eine globale Ablaufplanung keinen Nutzen bringt. Im Vergleich dazu ist der Datenproduktionsprozess der UBS WM&SB – bestehend aus fünf bis sechs Verarbeitungsschritten – wesentlich kürzer. Auch sind die Datentransformationsprozesse viel stärker automatisiert, wodurch Verzögerungen durch manuelle Eingriffe vermieden werden. Da die Mehrheit der Aktualisierungen erst zum Tagesabschluss eingebracht werden, ist ihr Datenvolumen entsprechend groß, was in langlaufenden Aktualisierungsjobs resultiert. Aktualisierungen dieser Art stören, wie in Abschnitt 5.1.2 gezeigt, die zeitliche Lokalität und führen zur Verbesserung der lokalen gegenüber der globalen Ablaufplanung. Im Hinblick auf steigende Echtzeitanforderungen werden die langlaufenden Jobs jedoch zunehmend in kleinere Aktualisierungen unterteilt, was für eine globale Ablaufplanung spricht. Über die Anzahl der stufenkonkurrierenden Aktualisierungen (siehe Abschnitt 5.1.2) in den beiden Szenarien kann an dieser Stelle keine Aussage gemacht werden. Stattdessen sind zusätzliche Untersuchungen notwendig.

Die Visualisierung der Datenqualität im Verlauf des Datenproduktionsprozesses kann in beiden der in Kapitel 2 vorgestellten Szenarien gewinnbringend eingesetzt werden. Insbesondere für die Datenwertschöpfungskette der GfK Marketing Services, die aus bis zu 60 Verarbeitungsschritten besteht, ist eine visuelle Überwachungskomponente sehr hilfreich. Allerdings können die 60 Verarbeitungsstufen aufgrund der daraus resultierenden Komplexität der Darstellung nicht gleichzeitig angezeigt werden. Stattdessen ist eine Klassifizierung der Verarbeitungsstufen sowie eine Unterteilung in Hierarchieebenen erfordlich. Die Datenqualität kann dann explorativ durch die bereits integrierten OLAP-Operationen, wie zum Beispiel Drill-Down, untersucht werden. Neben der Untersuchung der Datenaktualität ist im Szenario der GfK Marketing Services vor allem die Vollständigkeit der Daten für die Berichtsproduktion von Interesse. Um diese zu überwachen, müssten je nach Berichtskontext und Kunden entsprechende Vollständigkeitsmetriken definiert werden. Die UBS WM&SB besitzt mit dem Data Sourcing Framework bereits die notwendige Infrastruktur zur Fortschrittskontrolle der Datenproduktion. Die dafür notwendigen Schnittstellen könnten für die Umsetzung einer Visualisierungskomponente wiederverwendet werden.

6 Konsistente Datenanalyse in operativen Datenproduktionsprozessen

Aktuelle Data-Warehouse-Systeme werden, wie in Abschnitt 3.3.1 dargestellt, sowohl in Anwendungen zur operativen als auch zur klassischen, strategischen Analyse eingesetzt. Die push-basierte Aktualisierungssemantik eines Echtzeit-Data-Warehouse-Systems erhöht zwar die Datenaktualität, wie sie in operativen Prozessen gefordert wird, stellt jedoch ein Problem für die auf der Berichtsproduktion basierende strategische Entscheidungsfindung dar. In klassischen Data-Warehouse-Szenarien werden die Daten zu definierten Zeitpunkten aktualisiert, so dass die Datenbasis zwischen den Aktualisierungen stabil bleibt. Die Invalidierung bereits erzeugter Berichte ist daher ausgeschlossen. Im Gegensatz dazu sind die Daten eines Echtzeit-Data-Warehouse-Systems zu keinem Zeitpunkt stabil, sondern ständigen Änderungen unterworfen. Dies hat vor allem Auswirkungen auf die Berichtskonsistenz: Zwei Berichte, die zeitlich kurz nacheinander erzeugt wurden, können ein widersprüchliches Bild auf die Daten erzeugen. Diese Inkonsistenzen sind den Anwendern eines Data-Warehouse-Systems nur schwer zu vermitteln und verringern das Nutzervertrauen. Ein weiteres Problem der push-basierten Aktualisierung besteht darin, dass die Vollständigkeit der Daten, aufgrund der kontinuierlichen Datenproduktion, nicht garantiert werden kann. Daher gilt es insbesondere zu vermeiden, Berichte auf unvollständigen Daten zu produzieren.
Die für die Berichtsproduktion notwendige Stabilität kann durch Einführung einer zusätzlichen und vom Echtzeitbetrieb des Data-Warehouse-Systems entkoppelten Datenschicht erzielt werden, die im Weiteren als Reporting-Layer bezeichnet wird. Stabilität heißt insbesondere, dass sich berichtsrelevante Daten nicht unkontrolliert ändern können, sondern definierte Publikationszeitpunkte existieren. Die Einordnung des Reporting-Layer in die Architektur eines Echtzeit-Data-Warehouses (siehe Abschnitt 3.4.2) ist in Abbildung 6.1 skizziert. Der Reporting-Layer als letzte Stufe im Datenproduktionsprozess wird ebenfalls von der zentralen Anfrageschicht bedient, so dass Anfragen mit den entsprechenden Anforderungen bezüglich der Datenstabilität an diese Stufe weitergeleitet werden.
Der Aufbau, die Organisation und der Betrieb dieser Datenschicht wird im ersten Teil dieses Kapitels dargelegt. Der zweite Teil befasst sich mit der Komprimierung von Nullwerten, deren explizite Speicherung für die Vollständigkeitsbestimmung bei der Berichtserzeugung notwendig ist. Weiterhin wird die Anfrageverarbeitung auf nullwertkomprimierten Daten dargestellt.

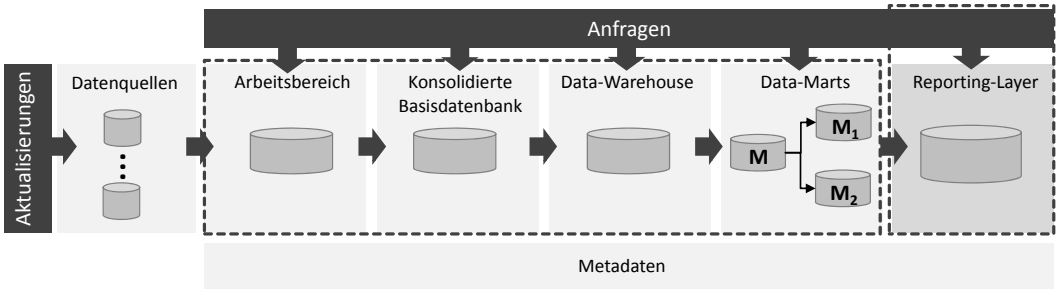


Abbildung 6.1: Einordnung des Reporting-Layer in die Architektur eines Echtzeit-Data-Warehouses

6.1 Der Reporting-Layer als Basis einer stabilen Berichtsproduktion

Im Gegensatz zu Ad-hoc-Anfragen, bei denen nur der aktuelle Zustand von Interesse ist und die Ergebnisse schnell wieder verworfen werden, spielt die Konsistenz bei der Berichtsproduktion eine große Rolle. Dies liegt vor allem darin begründet, dass Berichte selbst eine persistierte Sicht auf die Daten bilden, somit einen längeren Lebenszyklus haben und daher die Konsistenz zueinander gewährleistet werden muss. Aufgrund der stetigen Aktualisierungen in Echtzeit-Data-Warehouse-Systemen kann Stabilität nur durch Einführung einer zusätzlichen, vom Echtzeitbetrieb entkoppelten, Datenschicht garantiert werden. Diese wird im Folgenden als Reporting-Layer bezeichnet und ist in Abbildung 6.2 schematisch dargestellt. Die Idee des Reporting-Layer ist in gewisser Hinsicht mit der eines klassischen Data-Warehouses zu vergleichen (siehe Abschnitt 3.1.1). Beide haben die Aufgabe der organisatorischen Trennung der operativen Daten von den zur Analyse verwendeten Daten. Die Trennung geschieht durch Kopieren der Daten aus den Quellsystemen in das Data-Warehouse bzw. aus dem Echtzeit-Data-Warehouse in den Reporting-Layer. Weitere Aspekte, wie die Datenaufbereitung und -integration, sind bereits durch das Echtzeit-Data-Warehouse-System abgedeckt und spielen in diesem Kontext keine weitere Rolle.

6.1.1 Stabilität durch Entkopplung

Für die Übernahme der Daten aus dem Echtzeit-Data-Warehouse in die Basistabelle des Reporting-Layer werden die Daten zunächst in definierte Verwaltungseinheiten auf Basis ausgewählter Dimensionsattribute aufgeteilt (siehe Abschnitt 6.1.2), welche dann zu festgelegten Publikationszeitpunkten in den Reporting-Layer eingebracht werden (Schritt (A1) in Abbildung 6.2). Die sukzessive Bereitstellung der Verwaltungseinheiten wird in einer Publish-Tabelle protokolliert (Schritt (A2)). Während der Gültigkeitsdauer einer Verwaltungseinheit sind alle darauf berechneten Berichte zueinander konsistent. Die Datenpublikation kann durch den Verwalter einer bestimmten Datendomäne oder automatisch mittels definierter Regeln erfolgen. Trotz festgeschriebener Publikationszeitpunkte muss eine Berichtigung der Daten

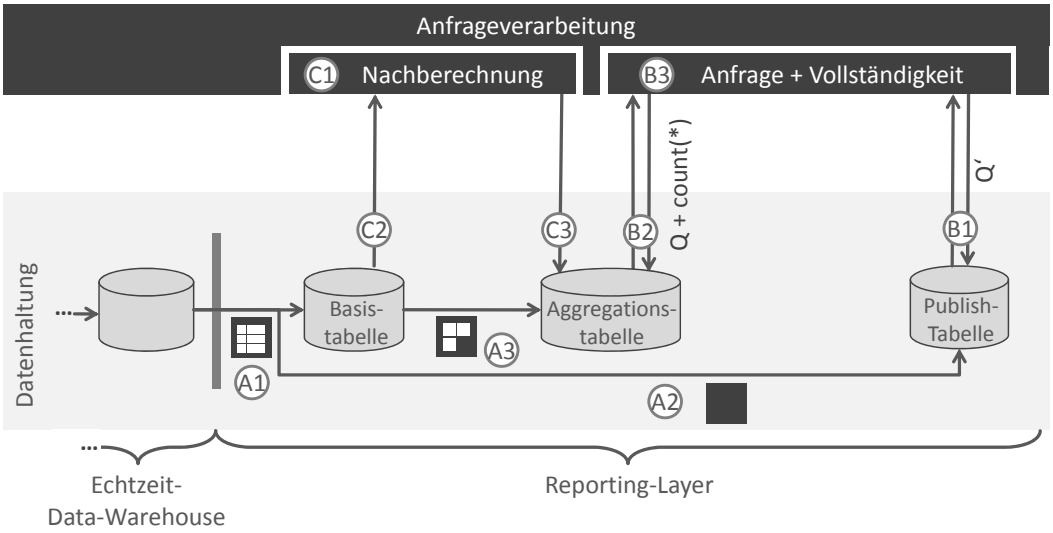


Abbildung 6.2: Reporting-Layer: Datenhaltung und Anfrageverarbeitung

des Reporting-Layer jederzeit möglich sein. Nachträgliche Änderungen erhöhen jedoch den Wartungsaufwand, da Berichte eventuell invalidiert werden müssen, und sind dementsprechend teure Operationen. Das Ziel ist es, für jede Datendomäne ein sinnvolles ökonomisches Maß an Stabilität und Aktualität zu erreichen. Das heißt, Daten sind zum einen zeitnah zu publizieren, damit frühzeitig Berichte berechnet werden können. Zum anderen sollte die Datenfreigabe erst stattfinden, wenn die Wahrscheinlichkeit für nachträgliche Änderungen nur noch sehr gering ist.

6.1.2 Vorberechnung von Basisaggregaten

Mit dem Fokus auf der Verwendung zur Berichtserzeugung sind die folgenden Anforderungen an den Reporting-Layer zu stellen: Aggregate müssen vorberechenbar sein, um so die Erzeugung von Standardberichten, deren Berichtsstruktur a priori bekannt ist, zu beschleunigen. Zusätzlich muss es möglich sein, Ad-hoc-Berichte auf dem Reporting-Layer zu berechnen. Dafür sind bereits vorhandene Aggregate wiederzuverwenden sowie fehlende Aggregate zu identifizieren, nachzuberechnen und nachzuladen. Die Prüfung auf Vollständigkeit der Daten des Reporting-Layer muss zudem ohne viel Mehraufwand möglich sein. Dies wird durch die Partitionierung der Daten erreicht, wodurch zum einen der Ladezustand des Reporting-Layer protokollierbar ist und zum anderen fehlende Aggregate effizient identifiziert werden können. Vor der Darstellung der Datenhaltung und Anfrageverarbeitung auf Basis des Reporting-Layer wird im folgenden Abschnitt zunächst das Partitionierungsschema sowie dessen Verwendung zur Vollständigkeitsbestimmung formal beschrieben.

Partitionierungsschema

Das multidimensionale Datenmodell weist einige interessante Eigenschaften auf, die im Folgenden für eine Partitionierung der Daten verwendet werden. Ein multidimensionales Schema besteht aus einer Menge von Dimensionshierarchien, die sich in Klassifikationsattribute und Eigenschaftsattribute unterteilen sowie aus einer Menge von Kennzahlen. Beide Konstrukte werden nachfolgend im Detail und am Beispiel der in Abbildung 6.3 dargestellten Produktionsdimension betrachet.
Eine Dimension besteht aus einer Menge von Attributen, die bezüglich einer funktionalen Abhängigkeit geordnet vorliegen. Die Dimensionsattribute können in drei Kategorien unterschieden werden:

- Primärattribut, das – in Analogie zum Primärschlüssel des relationalen Datenmodells – alle anderen Attribute einer Dimension bestimmt und die Feinheit der Granularität nach unten begrenzt. In Abbildung 6.3 ist dies die *ProduktID*.
- Klassifikationsattribute CA_i, welche die Dimensionen in verschiedenen Ebenen kategorisieren, wie die *Produktfamilie* und die *Produktgruppe* im Beispiel in Abbildung 6.3. Ausprägungen der Klassifikationsattribute werden als Klassifikationsknoten bezeichnet, etwa die Ausprägung *DVD-Spieler* des Klassifikationsattributs *Produktfamilie*.
- Eigenschaftsattribute FA_j, die funktional durch die Primär- und Klassifikationsattribute bestimmt werden und die Instanzen einer Dimension zusätzlich beschreiben. Nicht alle Eigenschaftsattribute müssen, wie zum Beispiel die Farbe oder die Marke der Produktdimension, für alle Ausprägungen einer Dimension semantisch gültig sein. Dies gilt etwa für die Eigenschaftsattribute *Bildschirmauflösung* oder *Auflösung*, die nur Produkte des Klassifikationsknotens *Monitore* oder *Fernseher* näher beschreiben. Im Gegensatz zu Klassifikationsattributen sind hierarchische Aggregationen auf Eigenschaftsattributen nicht möglich.

Zusätzlich ist das generische Dimensionsattribut *TOP* mit der einzigen Ausprägung *ALL* Teil jeder Dimensionshierarchie.
In Abschnitt 3.4.3 wurden die partitionierte Datenproduktion und -speicherung bereits abstrakt eingeführt und Empfehlungen ausgesprochen, nach welchen Dimensionsattributen die Partitionierung durchzuführen ist. Dafür qualifizieren sich insbesondere diejenigen Attribute, die die Einheiten vorgeben, nach welchen die Daten produziert werden. Die Unterscheidung in Klassifikations- und Eigenschaftsattribute erlaubt eine begriffliche Präzisierung des Partitionierungsschemas:

Partitionierungsschema: *Ein Partitionierungsschema PS besteht aus einem n-Tupel* $\{D_1.CA_1, ..., D_n.CA_n\}$*), wobei jedes Element durch ein Klassifikationsattribut sowie die Dimension als Präfix beschrieben ist. Des Weiteren gilt, dass für jede Dimension des multidimensionalen Schemas ein Klassifikationsattribut Element des Partitionierungsschemas sein muss.*

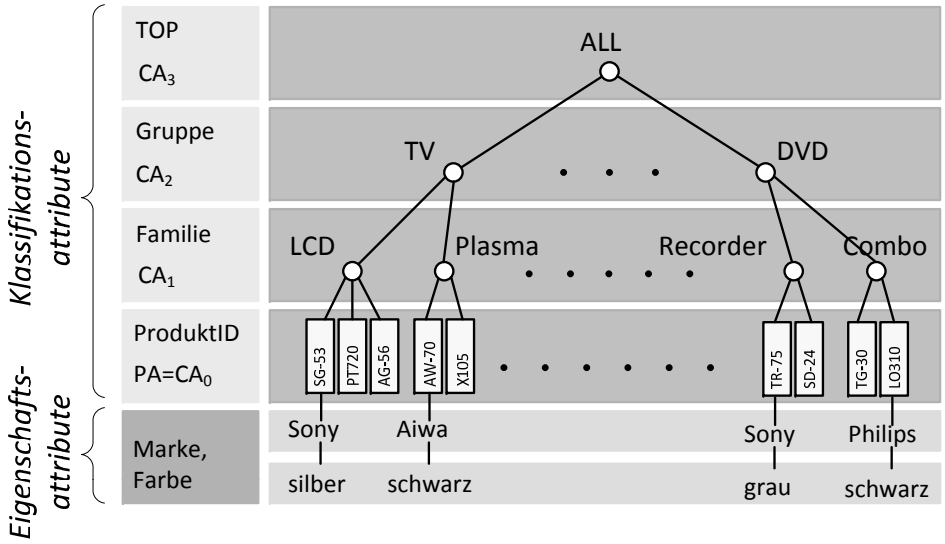


Abbildung 6.3: Klassifikations- und Eigenschaftsattribute am Beispiel einer Produktdimension

Bezogen auf das Beispiel in Abbildung 6.3 ist $PS = \{Produkt.Gruppe\}$ ein gültiges Partitionierungsschema. Durch die Partitionierung wird somit der multidimensionale Datenraum in gröbere Verwaltungseinheiten zerlegt. Die Festlegung des Partitionierungsschemas beschränkt den Detailgrad der Daten nach unten und muss dementsprechend sorgfältig gewählt werden. Jedoch kann beobachtet werden, dass für viele Datenanalysen ohnehin meist mit Aggregaten gerechnet wird, so dass dadurch keine Einschränkungen entsteht. Der für die Auswertung notwendige Detailgrad wird vielmehr durch die Eigenschaftsattribute sowie deren Kombination bestimmt. Diese qualifizieren sich nicht als Teil des Partitionierungsschemas, da sie andere Dimensionsattribute nicht funktional bestimmen und auch nicht für alle Ausprägungen einer Dimension semantisch sinnvoll sind.

Aggregationsmodell

Zur Beschleunigung der Berichtsproduktion müssen Aggregate vorberechnet und in der Aggregationstabelle abgelegt werden (Schritt (A3) in Abbildung 6.2). Die Struktur und Granularität der Aggregate ist dabei durch das Partitionierungsschema bzw. die Unterscheidung in Klassifikationsattribute und Eigenschaftsattribute vorgegeben, wodurch die Wiederverwendbarkeit und Wartbarkeit der Aggregate erhöht wird. Die Intuition dahinter ist, dass Berichte durch die Klassifikationsattribute strukturiert und durch die Eigenschaftsattribute lediglich verfeinert werden, so dass Berichtkennzahlen direkt aus der Aggregationstabelle entnommen bzw. abgeleitet werden können.
Dieser Zusammenhang ist in einem Beispiel in Abbildung 6.4 illustriert. Dazu wurde

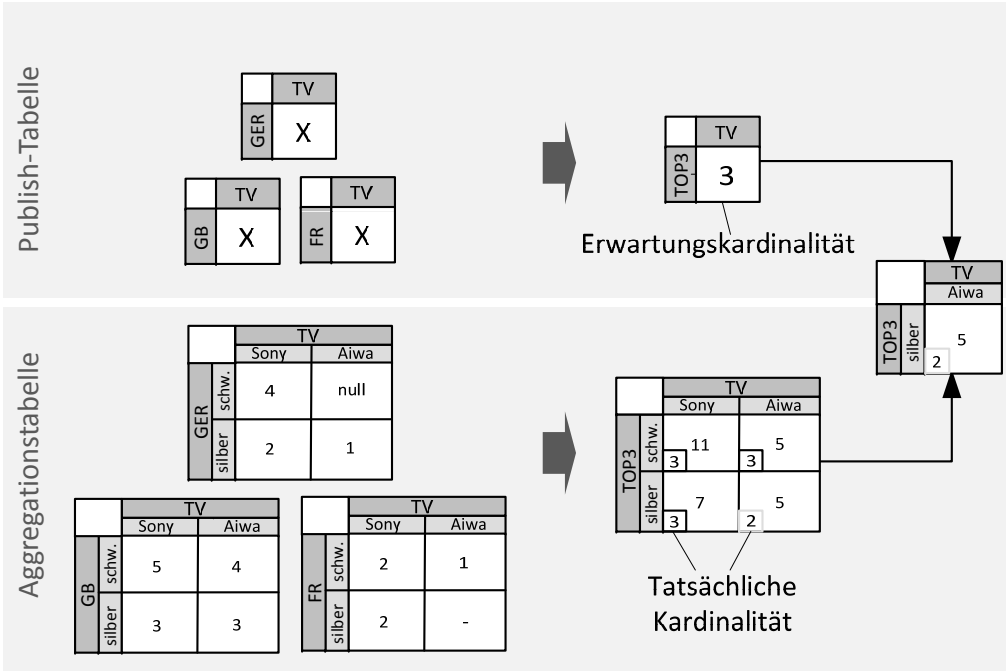


Abbildung 6.4: Aggregationsmodell und Bestimmung der Vollständigkeit

die Produktdimension aus dem vorhergenden Beispiel um eine Geschäftsdimension erweitert. In Abbildung 6.4 sind drei Berichte dargestellt, die Verkaufskennzahlen für eine bestimmte Produktgruppe, ein Land sowie Marke und Farbe ausweisen. Zur Berechnung der Kennzahlen existieren zwei Möglichkeiten: zum einen die Verwendung von Detaildaten der Basistabelle, was in Abhängigkeit des Umfangs der Basisdaten sehr teuer sein kann, und zum anderen die Wiederverwendung bereits berechneter und in der Aggregationstabelle gespeicherter Aggregate, deren Aggregationsniveau durch das Partitionierungsschema fest vorgeben ist. Die Aggregationstabelle kann für jeden Standardbericht mit allen notwendigen Aggregaten auf Basis der Klassifikationsattribute des Partitionierungsschemas (in diesem Beispiel $PS = (Produkt.Gruppe, Geschaeft.Land)$) und der Menge der Eigenschaftsattribute vorberechnet werden. Die in Abbildung 6.4 dargestellten drei Berichte sind dann durch den direkten Abruf der zwölf Aggregate berechenbar. Des Weiteren sind auch höherwertige Aggregationen auf Basis der Aggregationstabelle ableitbar. Im Beispiel ist dies durch eine Aggregation der Länder auf einen Klassifikationsknoten „TOP3“ dargestellt. Auch für diese Berechnung werden lediglich die zwölf Basisaggregate benötigt.

6.1.3 Vollständigkeitsbestimmung und Nullwertsemantik

Durch das standardisierte Berichtswesen, wie es in den meisten Unternehmen vorzufinden ist, kann die Aggregationstabelle bei der Publikation der Daten gezielt mit allen für den Standardbericht notwendigen Aggregaten gefüllt werden. Kennzeichnend für das beschriebene Verfahren ist, dass Aggregate nur in der Basisgranularität des Partitionierungsschemas gehalten und höherwertige Aggregate grundsätzlich aus diesen Teilaggregaten synthetisiert werden. Dies führt zwar bei wiederkehrendem Ausweis derselben abgeleiteten Aggregate im Berichtswesen zu einem leicht erhöhten Berechnungsaufwand, hält allerdings den Speicherbedarf für die Aggregationstabelle unter Kontrolle und führt überdies zu einer erhöhten Wiederbenutzungswahrscheinlichkeit der materialisierten Aggregate.
Jedoch können auch Ad-hoc-Anfragen von der Aggregationstabelle profitieren, falls die notwendigen Basisaggregate schon im Kontext von Standardberichten materialisiert wurden. Um zu ermitteln ob eine beliebige Anfragen mit Hilfe der in dem Reporting-Layer gespeicherten Aggregate beantwortet werden kann, bedarf es einer Vollständigkeitsbestimmung. Dazu ist eine Publish-Tabelle notwendig, in welcher hinterlegt ist, zu welchen Verwaltungseinheiten Daten für die Berichtsproduktion freigegeben wurden. Das Schema der Publish-Tabelle ist dabei identisch mit dem Partitionierungsschema. Die Einträge der Publish-Tabelle werden dazu verwendet, die Vollständigkeit einer Anfrage zu bestimmen. Dafür werden zwei Maße benötigt: Zum einen die *Erwartungskardinalität*, welche durch Umschreiben der Anfragen auf die Publish-Tabelle des Reporting-Layer bestimmt werden kann. Und zum anderen die *Ergebniskardinalitäten* einer Anfrage. Ist die Ergebniskardinalität bzw. die tatsächliche Kardinalität des Aggregats einer Anfrage niedriger als der Wert der Erwartungskardinalität, so ist die Aggregationstabelle für diese Anfrage unvollständig. Die Identifizierung der fehlenden Aggregate, deren Nachberechnung sowie die Bestimmung der Kardinalitäten wird in Abschnitt 6.1.5 betrachtet.
Sind in einer Verwaltungseinheit für eine bestimmte Kombination von Eigenschaftsattributen keine Basisdaten vorhanden, so wird dieser Wert mit NULL besetzt und physisch abgelegt. Die Speicherung von solchen NULL-Werten ist eine notwendige Voraussetzung, um diese von Basisaggregaten zu unterscheiden, die noch nicht berechnet wurden, da sie bisher nicht berichtsrelevant waren. Auf den Gegenstand bezogen ist die explizite Speicherung von Nullwerten notwendig um die Ergebniskardinalitäten einer Anfragen und damit die Vollständigkeit dieser korrekt zu bestimmen. Die explizite Speicherung von Nullwerten ist vor allem für dünnbesetzte Datenräume ein Problem, da zusätzlicher Speicheraufwand ensteht. Ein Komprimierungsverfahren, welches den Speicherbedarf erheblich reduziert, wird in Abschnitt 6.2 im Detail betrachtet.

Beispiel für die Vollständigkeitsbestimmung

Das Beispiel in Abbildung 6.4 illustriert diese Zusammenhänge. Das Ergebnis einer Berichtsanfrage auf der Aggregationstabelle, welche die Länder in einen Klassifika-

tionsknoten „TOP3“ zusammen aggregiert, ist in der Abbildung unten rechts dargestellt („TOP3“ steht für Deutschland, Frankreich und Großbritannien). Erwartungsgemäß werden die Einzelverkäufe zu gröberen Aggregaten verdichtet. Zusätzlich sind die Ergebniskardinalitäten, d.h. die Anzahl der Basisaggregate, aus denen sich das gröbere Aggregat berechnet, mit angegeben. Diese haben, mit einer Ausnahme, den Wert 3. Lediglich für das Tupel *(FR, TV, silber, Aiwa)* ergibt sich eine Kardinalität von 2, da ein Basisaggregat noch nicht berechnet wurde.
Zur Bestimmung der Vollständigkeit der so ermittelten Aggregate müssen die Ergebniskardinalitäten mit der zu erwartenden Kardinalität verglichen werden. Die Erwartungskardinalität wird auf Basis der in der Publish-Tabelle gespeicherten Verwaltungseinheiten ermittelt. In dieser sind Daten für Deutschland, Frankreich und Großbritannien in der Produktgruppe „TV“ protokolliert. Das Bilden des „TOP3“-Aggregats auf Basis der in der Publish-Tabelle gespeicherten Daten führt zu einer Erwartungskardinalität von 3. Ein Vergleich mit den vorher ermittelten Ergebniskardinalitäten beweist, dass das Aggregat *(FR, TV, silber, Aiwa)* des TOP3-Berichts nicht vollständig ist.

6.1.4 Datenhaltung

Die Grundlage der Datenhaltung der Reporting-Layer-Infrastruktur bilden vier Tabellen: die Basis-, die Aggregations-, die Publish- und die Feature-Dictionary-Tabelle. Die in Abschnitt 6.1.1 bereits eingeführte Basistabelle hat die Aufgabe, den Reporting-Layer von den operativen Datenänderungen des Echtzeit-Data-Warehouses zu trennen. Das Schema der Basistabelle entspricht dabei dem des Data-Warehouses, wobei Voraggregationen ebenfalls zulässig sind. Die Publish-Tabelle protokolliert alle in den Reporting-Layer bzw. in die Basistabelle eingebrachten und somit stabilen Daten. Die Granularität, in der die publizierten Daten protokolliert werden, entspricht dem angewendeten Partitionierungsschema. Die Kardinalität sowie die Anzahl der Spalten der Publish-Tabelle ist sehr gering, so dass die Zugriffskosten für die Bestimmung der Erwartungskardinalität vernachlässigbar sind (vgl. Abschnitt 6.1.7). Die Aggregationstabelle speichert die für die Berichtsproduktion notwendigen Aggregate. Deren Granularität ist ebenfalls durch das Partitionierungsschema vorgegeben, kann aber durch Hinzunahme beliebiger Eigenschaftsdimensionsattribute verfeinert werden.
Da die Anzahl der Eigenschaftsattribute sehr unterschiedlich sein kann und daher die Ausprägungen dieser Attribute sehr dünn besetzt sind, werden diese durch eine Menge von Attribut-Wert-Paaren, zum Beispiel *(Farbe:silber)* oder *(Farbe:Silber, Marke:Sony)*, repräsentiert und in einer Spalte gespeichert. Die Dimensionalität, die durch eine Dimension mit n Eigenschaftsdimensionsattributen verursacht wird, verringert sich damit auf zwei: ein Attribut als Teil des Partitionierungsschemas und ein Attribut, welches die Ausprägungen der Eigenschaftsdimensionen repräsentiert. Zur Vermeidung teurer Zeichenkettenoperationen bei der Anfrageverarbeitung auf Eigenschaftsattributen wird jede Menge von Attribut-Wert-Paaren zusätzlich über eine injektive Funktion auf einen eineindeutigen numerischen Schlüssel, der *Eigenschaft-*

sID, abgebildet. Die Zuordnung zwischen den *EigenschaftsIDs* und den Ausprägungen ist in der Tabelle Feature-Dictionary hinterlegt. Die *EigenschaftsID* wird von der Aggregationstabelle referenziert, so dass auch hochdimensionale Eigenschaftskombinationen kompakt gespeichert werden können. Diese Art der Speicherung hat des Weiteren den Vorteil, dass mit einer Anfrage eine Menge von Aggregaten unterschiedlicher Granularitäten ermittelt werden kann.

6.1.5 Prozess der Anfrageverarbeitung mit Vollständigkeitsbestimmung

Die Anforderungen an die Anfrageverarbeitung des Reporting-Layer sind zweigeteilt: Zum einen müssen Standardberichte schnell und effizient auf Basis vorberechneter Aggregationen ableitbar sein. Zum anderen müssen auch Ad-hoc-Berichte unter Nutzung der Aggregationstabelle schnell beantwortet werden können. Dazu müssen fehlende Aggregate erkannt und unter Nutzung der Basistabelle nachberechnet und in die Aggregationstabelle geladen werden. Zur Erkennung fehlender Aggregationen ist für jede Anfrage die Vollständigkeit zu bestimmen. Die Vollständigkeitsbestimmung ist so umzusetzen, dass die erste Anforderung nicht verletzt wird. Das heisst, Anfragen zur Berechnung von Standardberichten, die immer vollständig beantwortet werden können, dürfen durch die Vollständigkeitsprüfung nicht verzögert werden.
Der gesamte Prozess der Anfrageverarbeitung ist in Abbildung 6.2 dargestellt. In den Schritten (B1) und (B2) wird eine Anfrage Q parallel sowohl an die Aggregationstabelle als auch an die Publish-Tabelle gestellt. Die Anfrage Q wird dabei um ein $COUNT(*)$ ergänzt, um so die Ergebniskardinalität zu bestimmen. Zur Bearbeitung der Anfrage Q auf der Publish-Tabelle wird diese in eine Anfrage Q' umgeschrieben, die lediglich die Attribute des Partitionierungsschemas adressiert. Zur Bestimmung der Erwartungskardinalität wird Q' ebenfalls um eine Aggregationsfunktion $COUNT(*)$ erweitert. Die Mehrkosten durch die zusätzliche Aggregationsfunktion sind dabei vernachlässigbar (siehe Abschnitt 6.1.7). Durch einen Vergleich der Menge der Ergebnis- und der Erwartungskardinalitäten wird in Schritt (B3) die Vollständigkeit der Aggregationstabelle für die Anfrage Q bestimmt.
Ist die Aggregationstabelle für eine Anfrage unvollständig, d.h. sind Erwartungskardinalität und Ergebniskardinalität verschieden, so folgt die zweite Phase der Anfrageverarbeitung, bestehend aus den Schritten (C1) bis (C3), dargestellt in Abbildung 6.2. In Schritt (C1) wird das unvollständige Aggregate in die Basisaggregate zerlegt und die fehlenden Basisaggregate werden durch einen Vergleich mit dem Inhalt der Aggregationstabelle bestimmt. Die fehlenden Aggregate werden unter Verwendung der Basistabelle in Schritt (C2) nachberechnet und anschließend in Schritt (C3) in die Aggregationstabelle geladen. Das Ergebnis der Anfrageverarbeitung besteht aus der Ergebnismenge aus Phase 1 und den nachberechneten Aggregate aus Phase 2.

6.1.6 Verwandte Arbeiten und Techniken

Das Vorberechnen von Aggregaten, die Vollständigkeitsprüfung sowie das dynamische Nachladen, wie es in dem Reporting-Layer zum Einsatz kommt, teilen viele

Eigenschaften mit bekannten Techniken wie materialisierten Sichten und Caching. Diese werden im Folgenden vorgestellt sowie gegen den Reporting-Layer-Ansatz abgegrenzt.
Das allgemeine Problem der Anfrageverarbeitung unter Verwendung materialisierter Sichten wurde in [34] behandelt und in [72] zusätzlich um den Aspekt der Aggregationen erweitert. Ein grundlegendes Problem bei der Verwendung materialisierter Sichten, die deskriptiv durch Anfragen beschrieben werden, besteht in der Prüfung auf Enthaltensein sowie der notwendigen Umformulierung von Anfragen, die bewiesenermaßen NP-hart ist [43, 51]. Die Suche nach Umformulierungen für Aggregationsanfragen verursacht zusätzliche Komplexität im Vergleich zu konjunktiven Anfragen ohne Aggregation [4, 5]. Dies ist ein wesentliches Unterscheidungsmerkmal zum Reporting-Layer, in der durch die Verwendung eines Partitionierungsschemas das Anfrageraster fest vorgegeben ist, wodurch Anfrageumformulierungen entfallen. Die Vollständigkeit bzw. Ableitbarkeit wird stattdessen auf Ausprägungsebene durch einen Vergleich der Kardinalitäten überprüft. Des Weiteren werden die zu materialisierenden Ergebnisse im Reporting-Layer zwar ebenfalls durch Anfragen bzw. Berichtsstrukturen vorgeben, die Aggregationen jedoch auf ein einheitliches Niveau festgelegt. Durch die Abbildung der Eigenschaftsattribute einer Dimension auf eine Spalte ist es außerdem möglich, mit einer Anfrage eine Mengen von Aggregaten unterschiedlicher Granularitäten zu ermitteln. Bei der Verwendung materialisierter Sichten sind dazu mehrere Sichten sowie Anfragen notwendig.
Das Problem, die Vollständigkeit zu prüfen bzw. zu garantieren findet sich auch beim Caching wieder. Ein bekannter Vertreter ist DBCache [6, 35], das verschiedene Techniken zum Caching in mehrstufigen Infrastrukturen bereitstellt. DBCache verwendet sogenannte Cache-Gruppen und führt das Konzept von Cache-Schlüsselbedingungen und referenziellen Cache-Bedingungen ein, um die Vollständigkeit von Wert und Domaine zu garantieren. Sobald diese Bedingungen durch den Datenbankadministrator festgelegt wurden, kann DBCache asynchron und je nach Bedarf die Cache-Tabellen füllen. Allerdings ist DBCache nicht in der Lage, Aggregationen zwischenzuspeichern und die Verwendung von Cache-Bedingungen im multidimensionalen Modell kann schnell zu riesigen in der Cache-Datenbank zu speichernden Datenmengen führen. In dem hier vorgestellten Reporting-Layer wird die Vollständigkeit duch die Ermittlung der Kardinalitäten überprüft und ist somit stets garantiert. Der Effekt des unkontrollierten Nachladens ist im Reporting-Layer nicht gegeben, da stets nur die fehlenden Basisaggregate nachberechnet werden. Auch ist der Reporting-Layer keinen Speicherplatzbeschränkungen unterworfen, wie das bei klassischem Caching der Fall ist, so dass ein Großteil der benötigten Aggregate vorberechnet werden kann.
Eine sehr einfache und in bestimmten Anwendungsgebieten sehr effiziente Cache-Strategie bietet der Result-Cache in Oracle 11g. In diesem werden Anfrageergebnisse im Speicher vorgehalten, so dass spätere Anfragen davon profitieren können. Allerdings muss eine Anfrage, die vorberechnete Ergebnisse wiederverwenden möchte, identisch zu der Anfrage sein, deren Ergebnis im Cache abgelegt wurde. Dies schränkt den Nutzen des Result-Caches stark ein. In Anwendungen, in denen die Anfragen sehr homogen strukturiert sind und sich oft wiederholen, kann dieser An-

satz jedoch lohnend sein.
Im Bereich der OLAP-Anwendungen ist eine Arbeit zur schemaspezifische Pufferung bekannt [21]. Bei diesem Ansatz wird der multidimensionale Datenraum ebenfalls in Blöcke unterteilt. Für eingehende Anfragen werden diese Blöcke berechnet und in zwei Gruppen kategorisiert: zwischengespeicherte Blöcke und nicht zwischengespeicherte Blöcke. Um eine Anfrage zu verarbeiten, berechnet das System die fehlenden Blöcke aus den Rohdaten. Dieser Ansatz wurde in [20] weiterentwickelt, wo auch Blöcke in unterschiedlichen Aggregationsstufen betrachtet werden. Dieses blockbasierte Zwischenspeichern ähnelt dem Ansatz der einheitlichen Verwaltungseinheiten des Reporting-Layer. Dieser geht jedoch noch weiter und nutzt den Partitionierungsansatz als Vollständigkeitskriterium.
Eine notwendige Voraussetzung für die Wiederverwendung von Ergebnissen aus vorberechneten Kennzahlen ist die Summierbarkeit, welche in [50] definiert ist.

6.1.7 Evaluierung

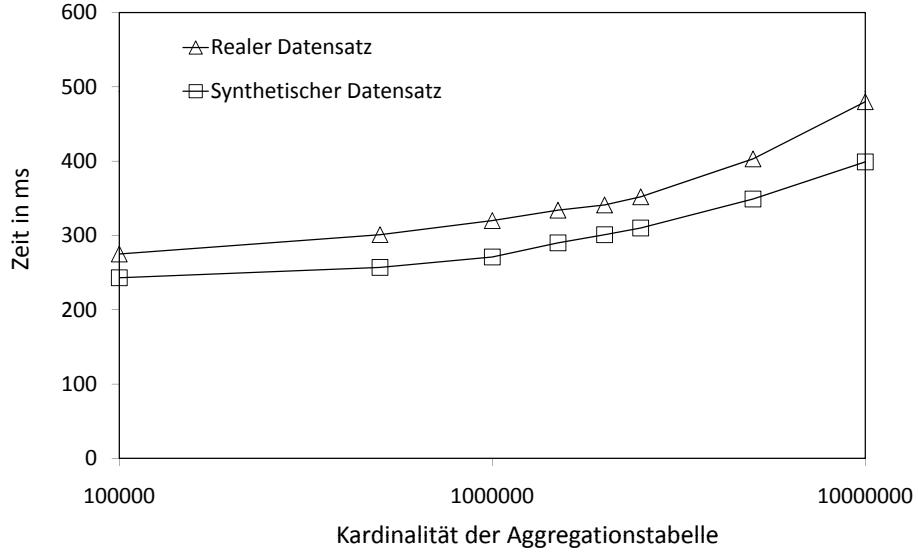
Die Zielstellung bei der Entwicklung des Reporting-Layer lag in der Schaffung einer stabilen Datenbasis zur effizienten Berichtserzeugung. Die Stabilität wurde dabei durch die Entkopplung von den operativen Datenänderungen erreicht. Dies ermöglicht den Einsatz vormaterialisierter Aggregationen zur Beschleunigung der Berichtsproduktion. Die Skalierbarkeit der Anfrageverarbeitung auf voraggregierten Daten sowie die Bewertung des Aufwands zur Bestimmung der Vollständigkeit sind Inhalt der folgenden Abschnitte.

Experimentierumgebung

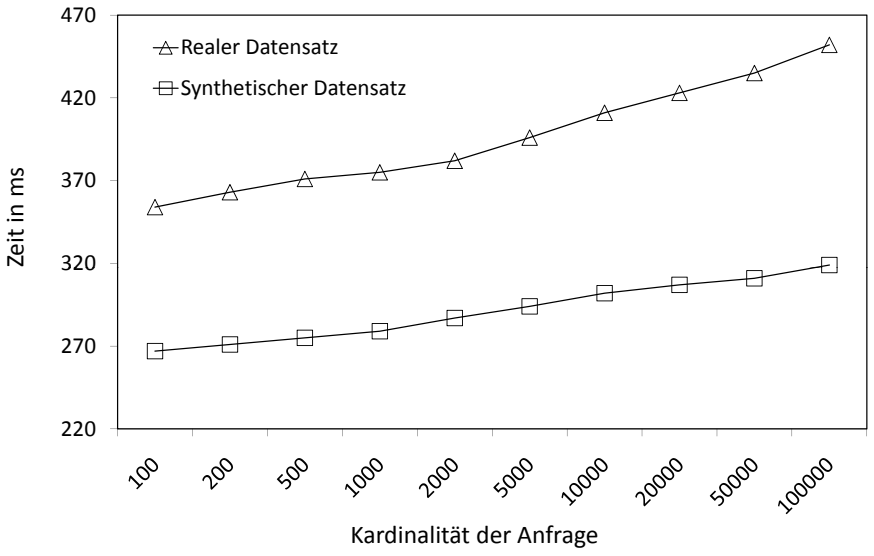
Zur Evaluierung des Reporting-Layer wurden sowohl synthetische als auch reale Datensätze aus der Marktforschung verwendet. Der synthetische Datensatz bestand aus 1 Mio., der reale Datensatz aus zirka 3 Mio. Basisaggregaten. Die Ausführung aller Experimente erfolgte auf einem Intel Celeron mit 2,66 GHz, Windows-Betriebssystem und 2 GB Speicher. Die in Abschnitt 6.1.5 beschriebene Anfrageausführung wurde mittels *Stored Procedures* in DB2 Version 8.5 umgesetzt.

Skalierbarkeit

Im ersten Experiment wurde die Skalierbarkeit der Anfrageausführung auf der Reporting-Ebene untersucht. Dazu wurden verschiedene Anfragen erzeugt, die sich vollständig auf Basis der Aggregationstabelle beantworten lassen und im Mittel 200 Basisaggregate benötigen. Die mittlere Laufzeit der Anfragen auf den realen und den synthetischen Datensätzen bei ansteigender Kardinalität ist in Abbildung 6.5a dargestellt. Für beide Datensätze skaliert die Anfrageverarbeitung nahezu linear. Ein ähnliches Verhalten ist auch für die ansteigende Anzahl notwendiger Basisaggregate beobachtbar, bei gleichbleibender Kardinalität der Aggregationstabelle von 750.000 sowohl für den synthetischen als auch den realen Datensatz (siehe Abbildung



(a) Wachsende Kardinalität der Aggregationstabelle



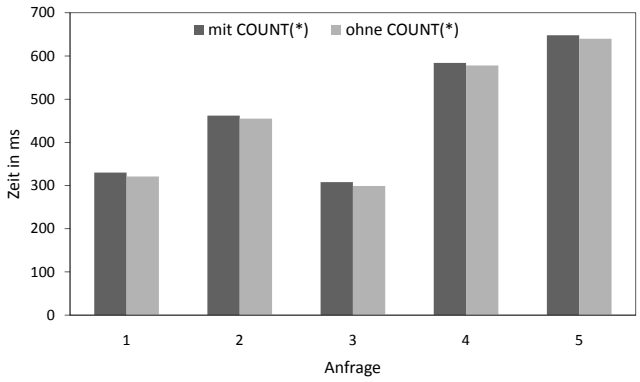
(b) Wachsende Kardinalität der Anfragen

Abbildung 6.5: Evaluierung der Skalierbarkeit der Anfrageverarbeitung

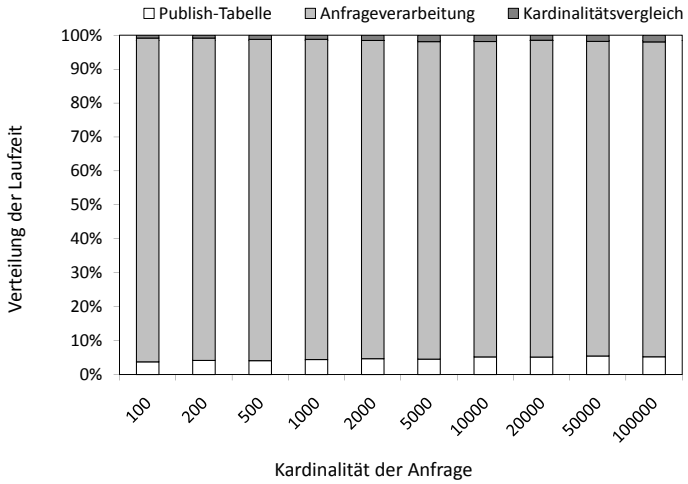
6.5b). Auch hier wächst die mittlere Laufzeit der Anfragen linear mit der Anzahl der Basisaggregate.

Aufwand der Vollständigkeitsüberprüfung und der Nachberechnung

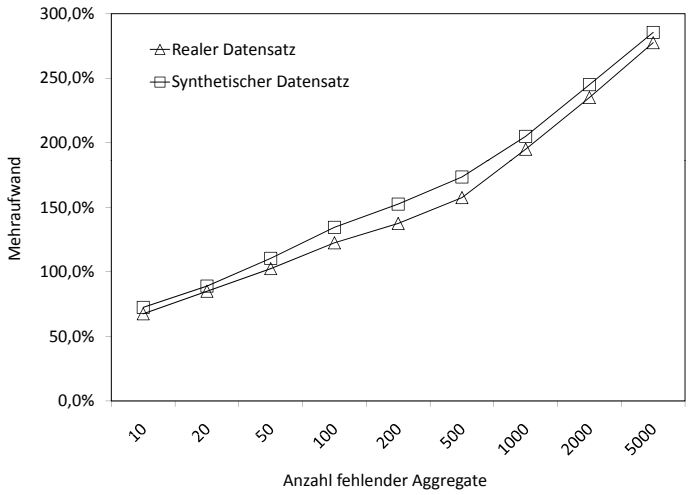
In einer zweiten Reihe von Experimenten wurde der Mehraufwand der Vollständigkeitsbestimmung und der Nachberechnung von Aggregaten untersucht. Die Kosten für die Vollständigkeitsprüfung sind vor allem durch die Ermittlung der Erwartungskardinalität, das zusätzliche $COUNT(*)$ zur Berechnung der Ergebniskardinalität und deren Vergleich bestimmt. Der Aufwand für diese Einzeloperationen sowie ihr Anteil an den Gesamtkosten der Anfrageverarbeitung wird im Folgenden untersucht. Zunächst wurden die Kosten der zusätzlichen Aggregationsfunktion $COUNT(*)$, um die jede Anfrage erweitert werden muss, evaluiert. Die mittleren Laufzeiten von fünf Anfragen, mit und ohne $COUNT(*)$, jeweils auf dem realen Datensatz ausgeführt, sind in Abbildung 6.6a dargestellt. Die Anfragen sind nach aufsteigender Komplexität geordnet, wobei diese durch die Anzahl adressierter Eigenschaftsdimensionen variiert wurde. Im Mittel erzeugte die zusätzliche Aggregationsfunktion zur Ermittlung der Ergebniskardinalität einen vernachlässigbaren Mehraufwand von 1,7%.
Die Laufzeit der Anfrage zur Ermittlung der Erwartungskardinalität ist primär durch die Kardinalität der Publish-Tabelle bestimmt. Diese ist jedoch im Vergleich zur Aggregationstabelle vernachlässigbar klein. Für den realen Datensatz, bestehend aus 3 Mio. Tupeln, und das angewendete Partitionierungsschema (Land, Monat und Produktgruppe) beträgt die Kardinalität der Publish-Tabelle nur zirka 1.000. Für den Vergleich der Ergebnis- und der Erwartungskardinalitäten ist ein Verbund der materialisierten Ergebnisse der Anfrage an die Aggregationstabelle und der Anfrage an die Publish-Tabelle notwendig. Da besonders letzteres Anfrageergebnis von geringer Kardinalität ist, sind die zu erwartenden Kosten des Verbunds relativ niedrig (zum Beispiel durch Anwendung eines Hash-Verbunds). Die Aufschlüsselung der Kosten der Anfrageverarbeitung in Abbildung 6.6b bestätigt dies. Es ist zu sehen, dass die Kosten der Anfrageverarbeitung vorwiegend durch die eigentliche Ausführung bestimmt sind. Mit zunehmender Kardinalität des Anfrageergebnisses steigen auch die Kosten des Verbunds, bleiben jedoch im Verhältnis relativ gering. Der Zugriff auf die Publish-Tabelle geht in konstantem Verhältnis in die Gesamtkosten mit ein.
Zuletzt wurde das Laufzeitverhalten der Anfrageverarbeitung bei wachsender Anzahl fehlender Aggregate untersucht. Die Kardinalität des synthetischen Datensatzes betrug 1 Mio., die des realen Datensatzes 3 Mio. Tupel. Die für das Experiment notwendigen Anfragen benötigten im Mittel 10.000 Basisaggregate. Von diesen wurde eine ansteigende Zahl stetig gelöscht, so dass eine Nachberechnung der Aggregate notwendig wurde (siehe Abschnitt 6.1.5). Der durch die Nachberechnung verursachte Mehraufwand gegenüber der Anfrageausführung auf einer vollständigen Aggregationstabelle ist in Abbildung 6.6c dargestellt. Bereits für zehn fehlende Basisaggregate steigt der Mehraufwand um über 70% und wächst linear weiter mit der Anzahl der fehlenden Aggregate. Aufgrund der Nutzung des Reporting-Layer zur Vorberechnung von Aggregaten für die Standardberichtsproduktion ist die Notwendigkeit von



(a) Laufzeitvergleich von Anfragen mit und ohne $COUNT(*)$



(b) Verteilung der Kosten auf die Einzelprozessschritte



(c) Wachsende Anzahl fehlender Aggregate

Abbildung 6.6: Evaluierung der Anfrageverarbeitung mit Vollständigkeitsprüfung

Nachberechnungen nur selten gegeben.
Die Experimente haben gezeigt, dass der durch die Vollständigkeitsbestimmung verursachte Mehraufwand vernachlässigbar gering ist. Der Reporting-Layer erfüllt somit die Anforderungen der performanten Ausführung von Anfragen zur Erzeugung von Standardberichten. Zusätzlich können auch Ad-hoc-Berichte von vorberechneten Aggregaten profitieren und mit minimalen Zusatzkosten auf Vollständigkeit überprüft werden.

6.2 Nullwertkomprimierung

In Abschnitt 6.1.2 wurde ein Verfahren zur Vollständigkeitsbestimmung von Berichtsanfragen vorgestellt, welches die explizite Speicherung von Nullwerten voraussetzt. Diese sind notwendig, um zu unterscheiden, ob ein Aggregat noch nicht vorberechnet wurde, weil bisher kein Bericht das Aggregat benötigte oder ob das Aggregat bereits berechnet wurde, jedoch keine Basisdaten vorlagen. Die Speicherung von Nullwerten wird auch als negatives Caching bezeichnet, d.h. das Nichtvorhandensein von Daten wird explizit persistiert.
In hochdimensionsalen Daten, die typischerweise dünn besetzt sind, da auf Grund der kombinatorischen Explosion für viele Ausprägungen keine Werte exisitieren, sind Nullwerte sehr häufig vorzufinden. Es kann gezeigt werden, dass in realen Datensätzen der Anteil der Nullwerte an der gesamten Aggregationstabelle leicht 95% und mehr beträgt (siehe Evaluierung in Abschnitt 6.2.5). Der so entstehende zusätzliche Speicherbedarf übersteigt somit den der Aggregationstabelle um ein Vielfaches. Da diese mit Nullwerten besetzten Aggregate nichts zum Ergebnis beitragen, jedoch zur korrekten Bestimmung der Vollständigkeit benötigt werden, wird im Folgenden eine Nullwertkomprimierung betrachtet, die den Speicheraufwand erheblich reduziert aber gleichzeitig die notwendigen Eigenschaften der Nullwerte erhält.
Das Vorgehen der Nullwertkomprimierung, die Anfrageverarbeitung sowie die Wartung der komprimierten Daten sind in Abbildung 6.7 skizziert. Der aus Nullwerten bestehende Anteil R_{null} einer Tabelle R wird im ersten Schritt analysiert. Sogenannte Abhängigkeitsregeln werden identfiziert (SDT) und im Anschluss zur Komprimierung der Nullwerte eingesetzt (siehe Abschnitt 6.2.2). Das Ergebnis ist eine Tabelle R', deren Anteil an Nullwerten R'_{null} erheblich reduziert ist. Der Mehraufwand zur Speicherung der Abhängigkeitsregel SDT fällt dabei nicht ins Gewicht. Zur Laufzeit einer Anfrage werden in Schritt (2) die zur Bestimmung der Vollständigkeit notwendigen Nullwerte dekomprimiert (siehe Abschnitt 6.2.3). Des Weiteren ist die Wartung der komprimierten Daten bei Einfüge-, Lösch- und Aktualisierungsoperationen zu berücksichtigen (Schritt (3), siehe Abschnitt 6.2.3). Die Leistungsfähigkeit der Nullwertkomprimierung und der Anfrageverarbeitung auf komprimierten Daten wird in Abschnitt 6.2.5 in einer Reihe von Experimenten untersucht.

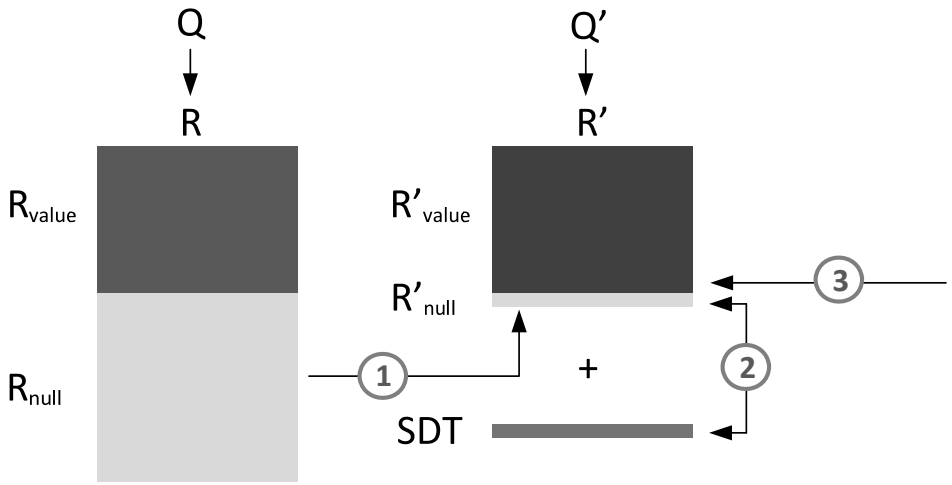
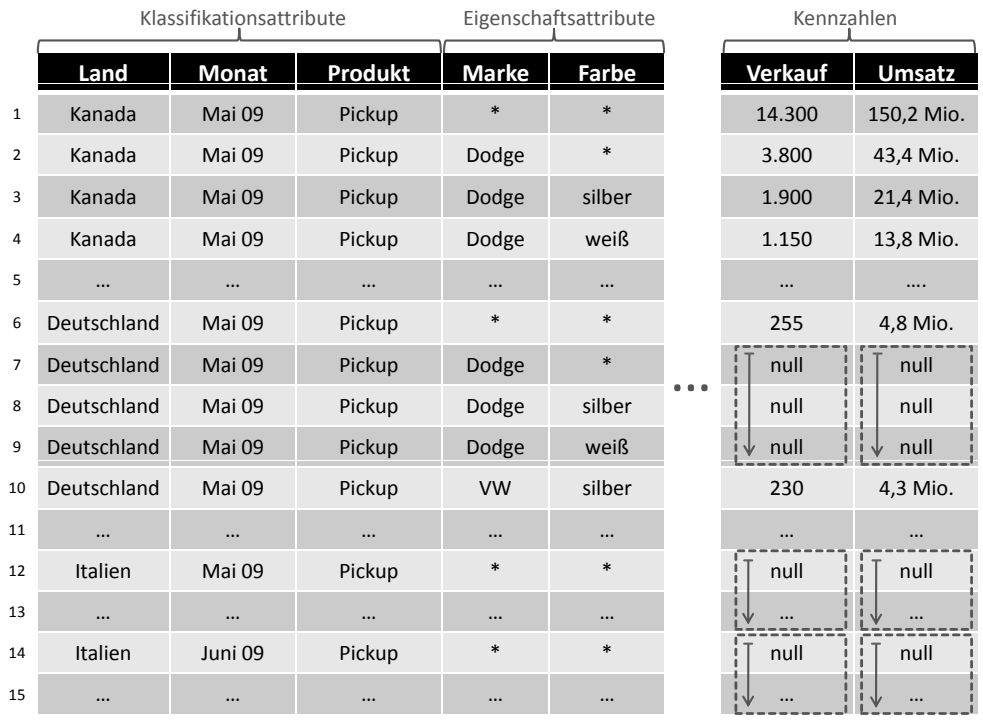


Abbildung 6.7: Vorgehensmodell zur Nullwert-Komprimierung

6.2.1 Einleitendes Beispiel und Vorbetrachtungen

Die grundlegende Idee des Ansatzes wird an einem Beispiel illustriert. Hierzu sei eine Tabelle mit PKW-Verkaufsdaten gegeben (siehe Abbildung 6.8). Diese enthält die Anzahl der verkauften PKWs sowie die erzielten Umsätze, unterteilt nach Land, Monat, Produkt, Marke und Farbe. Land, Monat und Produkt sind in diesem Beispiel Klassifikationsattribute (siehe Abschnitt 6.1.2). Marke und Farbe sind Eigenschaftsattribute, die die Produktdimension weiter verfeinern. Der Attributwert „*“ steht für eine Generalisierung aller Werte dieses Attributs.
Zunächst wird das Produkt „Pickup“ betrachtet, welches häufig in Kanada aber eher selten in Europa vorzufinden ist. Anhand des Beispieldatensatzes ist ersichtlich, dass wenige Verkäufe dieses PKW-Typs in Deutschland und keine Verkäufe in Italien getätigt wurden. Weiterhin ist zu sehen, dass für die Marke „Dodge“, welche in Deutschland kaum etabliert ist, keine Verkäufe im „Mai 2009“ vorhanden sind (siehe Tupel 7). Daraus kann geschlossen werden, dass auch für alle abgeleiteten Tupel in diesem Zeitraum – in Deutschland – keine Verkäufe existieren können. Von Tupel 7 („Dodge,“ null) abgeleitete Datensätze sind in diesem Fall das Tupel 8 („Dodge,“ „silber“) und das Tupel 9 („Dodge,“ „weiß“). Durch den Erhalt dieser Abhängigkeit bzw. Ableitbarkeit können diese Nulltupel, die keine zusätzlichen Informationen beinhalten, gelöscht werden. Im Beispiel Italien können alle Datensätze, die von den Tupeln 12 und 14 ableitbar sind, gelöscht werden.
In der Praxis wird jeweils nur eine kleine Teilmenge aller PKW-Typen oder Automarken in den einzelnen Ländern verkauft, so dass mit hoher Wahrscheinlichkeit die Anzahl der Nulltupel den größten Teil der Daten ausmachen wird.
Wie bereits in Abschnitt 6.1.2 beschrieben, werden multidimensionale Daten durch eine Reihe von Dimensionen aufgespannt, deren Attribute in Klassifikations- und Eigenschaftsattribute unterschieden werden. Klassifikationsattribute CA_i $(i = 0, ..., n)$



	Land	Monat	Produkt	Marke	Farbe	Verkauf	Umsatz
1	Kanada	Mai 09	Pickup	*	*	14.300	150,2 Mio.
2	Kanada	Mai 09	Pickup	Dodge	*	3.800	43,4 Mio.
3	Kanada	Mai 09	Pickup	Dodge	silber	1.900	21,4 Mio.
4	Kanada	Mai 09	Pickup	Dodge	weiß	1.150	13,8 Mio.
5	...	...	...	...	...	...	....
6	Deutschland	Mai 09	Pickup	*	*	255	4,8 Mio.
7	Deutschland	Mai 09	Pickup	Dodge	*	null	null
8	Deutschland	Mai 09	Pickup	Dodge	silber	null	null
9	Deutschland	Mai 09	Pickup	Dodge	weiß	null	null
10	Deutschland	Mai 09	Pickup	VW	silber	230	4,3 Mio.
11	...	...	...	...	...	...	...
12	Italien	Mai 09	Pickup	*	*	null	null
13	...	...	...	...	...	...	...
14	Italien	Juni 09	Pickup	*	*	null	null
15	...	...	...	...	...	...	...

Abbildung 6.8: Multidimensionaler Datensatz mit PKW-Verkaufsdaten

beschreiben eine Hierarchie und sind bezüglich ihrer funktionalen Abhängigkeit geordnet. Instanzen der Klassifikationsattribute werden als Klassifikationsknoten beschrieben, z.B. „Pickup“ als Instanz des Klassifikationsattributes „Produkt“. Den Wurzelknoten der Klassifikationshierarchie bildet das Attribut $CA_j = TOP$ mit der einzigen Ausprägung „ALL“. Im Gegensatz zu Klassifikationsattributen bestehen zwischen Eigenschaftsattributen FA_i $(i = 0, ..., n)$ keine funktionalen Abhängigkeiten. Jedoch werden Eigenschaftsattribute funktional durch Klassifikationsattribute bestimmt.
Als durchgehendes Beispiel für die folgenden Abschnitte wird die Tabelle $R(A, B, C, D, E, F, M)$, bestehend aus sechs Eigenschaftsattributen A bis F und einem Faktenattribut M (siehe Abbildung 6.9a), verwendet. Da die Komprimierung ausschließlich durch Abhängigkeit auf Basis der Eigenschaftsattribute definiert ist, werden die Klassifikationsattribute einer zu komprimierenden Tabelle hier zunächst nicht betrachtet. Sie werden erst wieder bei der Durchführung der Komprimierung in Abschnitt 6.2.2 relevant. Die Tabelle ist mit einer Menge von Beispieldaten gefüllt, wobei die Ausprägung „ALL“ durch das Zeichen „*“ repräsentiert wird. Die Kombinationen der verschiedenen Wertausprägungen der Eigenschaftsattribute in R bilden einen direkten azyklischen Graphen $(\Gamma, \prec)$, dargestellt in Abbildung 6.9b. Die Kanten des Graphen sind wie folgt definiert: Ist $F(FA_1, ..., FA_n) \prec F'(FA'_1, ..., FA'_n)$, dann muss für $FA'_i \neq *$ gelten, dass für $\forall i$ $FA_i = FA'_i$. Der Graph $(\Gamma, \prec)$ ist keine

Repräsentation des vollständigen multidimensionalen Würfels, da nur die für den Nutzer interessanten Kombinationen von Eigenschaften Teil der Tabelle R sind. So fehlt zum Beispiel die Eigenschaftsmenge $(*, B, *, *, *, *)$, abgekürzt als B.

6.2.2 Nullwertkomprimierung

Aufbauend auf dem Beispiel des vorangegangenen Abschnitts, wird im Folgenden die Extraktion von Abhängigkeitsregeln aus den Nullwerten einer zu komprimierenden Tabelle beschrieben, die im Anschluss zur Komprimierung der nullwertigen Daten verwendet werden. Weiterhin wird eine Datenstruktur zur effizienten Speicherung und Abfrage der komprimierten Daten entwickelt.

Nullwertverhältnis als Kennzahl

Der Grad der Komprimierbarkeit einer Tabelle, bestehend aus Eigenschaftsattributen F und einem Fakt M, wird durch das Nullwertverhältnis der einzelnen Eigenschaftsattributkombinationen bestimmt. Dies ist wie folgt definiert:

Definition 6.1 (Nullwertverhältnis) *Gegeben sei eine Menge von Eigenschaftsattributen F einer Tabelle R. $C_v(F)$ ist die Kardinalität der Tupel, die diese Eigenschaftsattribute instanziieren und deren Faktattribut M kein Nullwert ist, d.h. $M \neq null$. Analog dazu ist $C_n(F)$ die Kardinalität der Tupel, deren Faktattribut durch einen Nullwert besetzt ist, d.h. $M = null$. Aus diesen beiden Werten ergibt sich das Nullwertverhältnis ρ einer Eigenschaftsmenge wie folgt:*

$$\rho(F) = \frac{C_n(F)}{C_v(F) + C_n(F)}.$$

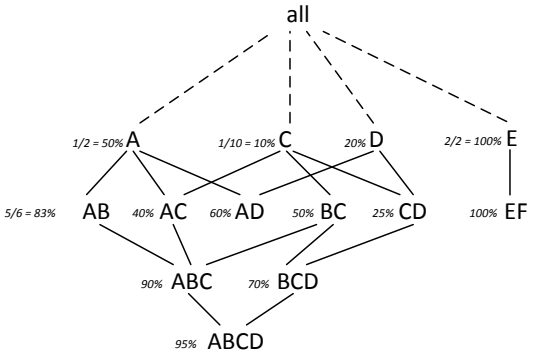
Die Reduzierung auf lediglich ein Faktattribut dient der Vereinfachung der folgenden Beschreibung. Sie ist jedoch zulässig, da aus der Nullwertigkeit eines Faktes, d.h. dem Nichtvorhandensein von Basisdaten, auch die Nullwertigkeit aller anderen Fakten geschlossen werden kann. Die Intuition hinter dem Nullwertverhältnis wird an dem folgenden Beispiel illustriert: Das Eigenschaftsattribut $F = (A)$ aus Abbildung 6.9b besitzt zwei Ausprägungen: zum einen den Wert 600 für das Faktattribut M und zum anderen den Nullwert. Daraus ergibt sich für A, nach obiger Definiton, ein Nullwertverhältnis von $\rho(A) = 0,5$. Aus diesem Wert lässt sich unter Annahme einer Gleichverteilung der Eigenschaftskombinationen ableiten, dass jeder Nachfolger von A ein Nullwertverhältnis größer oder gleich dem Nullwertverhältnis von A haben muss, z.B. $\rho(AB) = 0,83 > \rho(A) = 0,5$. Die Gleichverteilung von Eigenschaftsattributen ist wie folgt definiert:

Eigenschaft 6.1 (Gleichverteilung von Eigenschaftsattributen) *Die Gleichverteilung von Eigenschaftsattributen beschreibt die Eigenschaft, dass jede Ausprägung von Eigenschaftsattributen F_i auch in allen Nachfolgern $F_j \prec F_i$ enthalten ist.*

Am Beispiel des Eigenschaftsattributs A bedeutet dies, dass a_1 und a_2 auch in allen

A	B	C	D	E	F	M
a_1	*	*	*	*	*	null
a_2		*	*	*	*	600
*	*	c_1	*	*	*	400
*	*	...	*	*	*	null
*	*	c_{10}	*	*	*	null
*	*	*	d_1	*	*	150
...	...	...	...	...	...	null
*	*	*	d_5	*	*	150
a_1	b_1	*	*	*	*	null
a_1	b_2	*	*	*	*	null
a_1	b_3	*	*	*	*	null
a_2	b_1	*	*	*	*	400
a_2	b_2	*	*	*	*	null
a_2	b_3	*	*	*	*	null
...	...	...	...	...	...	null
*	*	*	*	e_1	*	null
*	*	*	*	e_2	*	null
*	*	*	*	e_1	f_1	null
*	*	*	*	e_1	f_1	null
...	...	...	...	...	...	null

(a) Beispielrelation R



(b) Graph der Eigenschaftsmenge

Abbildung 6.9: Beispieldatensatz

Nachfolgern von A, AB, AC, AD, ABC usw. vorkommen müssen. Da die Navigation durch multidimensionale Daten meist beginnend von gröberen und hin zu verfeinerten Kennzahlen geschieht (Drill-down), indem schrittweise immmer neue Eigenschaftsattribute hinzugenommen werden, ist die Annahme der Gleichverteilung in den meisten Datensätzen korrekt. Der hier vorgestellte Ansatz ist jedoch nicht ausschließlich auf gleichverteilte Daten beschränkt. Im Evaluierungskapitel 6.1.7 wird der Algorithmus zur Komprimierung auch für andere Verteilungen untersucht.
Neben dem Nullwertverhältnis wird zum Vergleich von Eigenschaftsattributen noch die Größe einer Eigenschaftsattributmenge benötigt, die folgendermaßen definiert ist:

Definition 6.2 (Größe einer Eigenschaftsattributmenge) *Die Größe einer Eigenschaftsattributmenge F_i ist die Anzahl der in F_i vorkommenden Eigenschaftsattribute.*

Eigenschaften von Basistupeln

Mit den bisher beschriebenen Methoden ist es möglich, für eine Tabelle R, bestehend aus Eigenschafts- und Faktattributen, eine Graphstruktur aufzubauen, deren Knoten die Eigenschaftsattribute von R und ihre Nullwertverhältnisse ρ repräsentieren. Ausgehend von dieser Struktur, kann die Nullpartition $R_{null} \subseteq R$ komprimiert werden. Der Komprimierungsansatz basiert auf sogenannten Basis- bzw. Seed-Tupeln, deren Eigenschaften in diesem Abschnitt beschrieben werden.

Das Eigenschaftsattribut A, mit den zwei Ausprägungen a_1 ($M = null$) und a_2 ($M = 600$), hat, wie im vorigen Abschnitt hergeleitet, ein Nullwertverhältnis von $\rho_A = 0,5$. Unter der Annahme, dass die Eigenschaft 3.1 für R gegeben ist, kann geschlossen werden, dass für alle Nachfolger, $t \in F_i \prec F_A$, mindestens 50% der Tupel ebenfalls einen Nullwert enthalten müssen. Am Beispiel in Abbildung 6.9b sind dies die Tupel 12-14, da sie ebenfalls die Ausprägung a_1 des Eigenschaftsattributs A enthalten. Durch Erhalt des Tupels 1 können somit die Tupel 12-14 abgeleitet bzw. neu erzeugt werden. Tupel $(a_1, *, *, *, *, *)$ ist somit ein Basistupel, welches in dem Beispiel alle Tupel $(a_1, b_1, *, *, *, *)$ abdeckt. Die Nulltupel einer Relation werden in Basistupel sowie zwei andere Arten unterschieden:

- Die Menge der Basistupel, $R_{seed} \subset R_{null}$, besteht aus den Tupeln, aus deren Existenz andere Tupel ableitbar sind und welche somit explizit gespeichert werden müssen, d.h. $R_{seed} \subset R'_{null}$.

- Redundante Tupel, $R_{redundant} \subseteq R_{null}$, sind die Tupel, welche aus der Menge der Basistupel R_{seed} abgeleitet sind und die somit aus R_{null} entfernt werden können, z.B. $(a_1, b_1, *, *, *, *)$. Durch das Löschen der redundanten Tupel wird R_{null} zu R'_{null} komprimiert.

- Die Ausreißertupel, $R_{outlier} \subseteq R_{null}$, sind weder Basistupel noch werden sie durch ein Basistupel abgedeckt und müssen daher ebenfalls explizit gespeichert werden, d.h. $R_{outlier} \subset R'_{null}$.

Um die Basistupel einer Nullpartition bestimmen zu können, ist noch eine weitere Kennzahl notwendig: der *Abdeckungsgrad* einer Eigenschaftsattributmenge. Als Beispiel sind die Eigenschaftsattributmengen A and D aus Abbildung 6.9b zu betrachten. Aus diesen ist ableitbar, dass 50% aller Nulltupel des Nachfolgers AD durch A und 20% durch D abgedeckt sind. Im Allgemeinen kann der Abdeckungsgrad einer Eigenschaftsattributmenge F_n wie folgt bestimmt werden:

Definition 6.3 (Abdeckungsgrad) *Der Abdeckungsgrad $\omega(F_n, C)$ einer Eigenschaftsattributmenge F_n, welche durch eine Menge von Eigenschaftsattributmengen C abgedeckt ist, wobei $F_n \prec F_i$ für jedes $F_i \in C$ gilt, ist definiert als*

$$\omega(F_n, C) = 1 - \prod(1 - \rho(F_i)) \qquad \forall F_i \in C$$

unter der Annahme, dass Eigenschaft 3.1 erfüllt ist.

Für das vorhergehende Beispiel ergibt sich somit $\omega(F_{AD}, \{F_A, F_D\}) = 1 - (1 - 0,5)(1 - 0,2) = 0,6$, d.h. 60% aller Nulltupel in AD werden durch A und D abgedeckt. Das Maß der Abdeckung wird im Folgenden dazu verwendet, die Basistupel zu identifizieren, von denen möglichst viele andere Nulltupel ableitbar sind. Umso höher das Nullwertverhältnis und der Abdeckungsgrad, desto mehr redundante Tupel können gelöscht werden, woraus sich eine stärkere Komprimierung ergibt.

Algorithmus 3 Abhängigkeitssuche

Benötigt: Tabelle R und minimaler Abdeckungsgrad *mincover*.

```
1: for all Tupel ∈ R do
2:    zähle C_n und C_v für jede Eigenschaftsattributkombination
3: end for
4: Erzeuge leeren Graphen G
5: for all Eigenschaftsattributkombination do
6:    Erzeuge Knoten N = (S = ∅, C_n, C_v) in G mit
7: end for
8: level = 1
9: for all Knoten N ∈ G, traversiert in absteigender Ordnung der Nullwertverhältnisse
   und in aufsteigender Ordnung der Größe der Eigenschaftsattributkombination do
10:   for all Knoten N do
11:      if N.width = level then
12:         Trace-Down(N)
13:      end if
14:   end for
15:   level++
16: end for
17: SDT=leer
18: for all Knoten N do
19:   if N.seedset ≠ NULL then
20:      füge N und N.seedset in SDT ein
21:   end if
22: end for
23: return SDT
```

Algorithmus zur Suche von Basistupeln

Wie im vorherigen Abschnitt dargelegt, sind von Basistupeln ableitbare Tupel redundant und können somit entfernt werden. Unter Ausnutzung der bereits eingeführten Kennzahlen – dem Nullwertverhältnis, der Größe einer Eigenschaftsattributmenge und dem Abdeckungsgrad – wird im Folgenden ein Algorithmus zur Identifizierung der Abhängigkeitsregeln in einer Relation R entwickelt. Dieser wird im Weiteren mit *Abhängigkeitssuche* bezeichnet und ist in den Algorithmen 3 und 4 dargestellt.
Im ersten Schritt sind die Nullwertverhältnisse aller Eigenschaftsattributkombinationen zu bestimmen (siehe Zeilen 1 bis 3 in Algorithmus 3). Diese können in einem Lauf über die Basistabelle R, durch Zählen der Häufigkeiten der Null- bzw. Nicht-Null-Ausprägungen (C_n bzw. C_v), ermittelt werden. In einem zweiten Schritt ist der Graph, bestehend aus den Eigenschaftsattributkombinationen, zu erzeugen. Jeder Knoten des Graphen wird repräsentiert durch ein Tripel, bestehend aus C_n, C_v und einer Seed-Menge, die zunächst leer initialisiert wird. Die Knoten werden in einer Tabelle *FeatureSetNode(featureset, C_v, C_n)* und die Seed-Mengen der einzelnen Knoten in einer zweiten Tabelle *FeatureSetEdge(featureset,parent)* gespeichert.
Nach Erstellung des Graphen wird dieser in aufsteigender Ordnung der Größe der Eigenschaftsattributmenge und in absteigender Ordnung der Nullwertverhältnisse

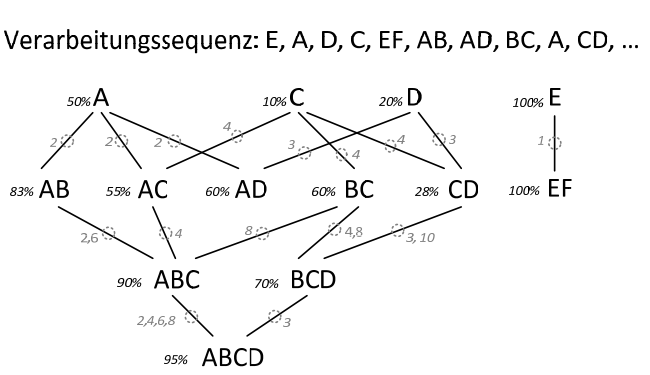
Algorithmus 4 Trace-Down

Benötigt: Knoten N, *mincover*.

```
1: for all c ∈ N.nachfolger do
2:   if c.coverratio < mincover ∧ N ∉ c.seedset then
3:     füge N in c.seedset ein
4:     c.coverratio = cover(c.seedset)
5:     trace-down(c)
6:   else
7:     return
8:   end if
9: end for
```

abgearbeitet. Somit werden Knoten mit wenigen Attributen und hohem Nullwertverhältnis bevorzugt, da die entsprechenden Ausprägungen viele Nulltupel abdecken. Die Traversierung der Knoten ist in ihren Einzelheiten in Abbildung 4 dargestellt. Für jeden Nachfolgerknoten eines Knoten N wird überprüft, ob dessen Abdeckungsgrad bereits ein definiertes Limit *mincover* überschreitet. Falls dies der Fall ist, wird dieser Knoten nicht weiter betrachtet, da er bereits ausreichend durch eine Menge von Eigenschaftsattributmengen abgedeckt ist. Falls der Abdeckungsgrad geringer als die definierte Grenze ist, so wird N der Basistupelmenge des betrachteten Knoten hinzugefügt und der Grad der Abdeckung dieses Knoten wird neu berechnet. Die Berechnungsvorschrift des Abdeckungsgrad ist in Definition 6.3 dargestellt. Die Traversierung endet, falls alle Knoten die minimale Abdeckung *mincover* erreicht haben oder alle Knoten verarbeitet wurden.
Abschließend wird der Graph ein zweites Mal durchlaufen, um die ermittelten Eigenschaftsattributkombinationen und deren Seed-Mengen zu extrahieren und persistent zu machen. Für jeden Knoten, dessen Seed-Menge nicht leer ist, wird ein entsprechender Eintrag in die Basistupelabhängigkeitstabelle, kurz SDT, eingefügt. Jeder Eintrag in dieser Tabelle bildet eine sogenannte Abhängigkeitsregel.

Beispiel Der Algorithmus zur Suche von Abhängigkeitsregeln wird an einem Beispiel (siehe Abbildung 6.10) illustriert. Der minimale Abdeckungsgrad wurde für das Beispiel auf 90% gesetzt. Durch die Verarbeitung der Knoten in absteigender Ordnung der Nullwertverhältnisse qualifiziert sich E als Erstes. Der einzige Nachfolger von E ist der Knoten EF, in dessen Seed-Menge E eingefügt wird. Im nächsten Verarbeitungsschritt folgt A mit seinen Nachfolgern AB, AC, AD, ABC und $ABCD$, die jeweils A in ihre Seed-Menge aufnehmen. Analog wird in den Verarbeitungsschritten 3 bis 5 verfahren. In Schritt 6 wird der Knoten AB verarbeitet. Dessen Nachfolger ist der Knoten ABC, welcher bereits durch die Knoten A und C aus vorangegangenen Verarbeitungsschritten abgedeckt wird. Der Abdeckungsgrad des Knoten ABC ermittelt sich aus $1-(1-0,5)(1-0,1) = 0,55$ und ist daher geringer als der minimale Abdeckungsgrad von 90%. Somit wird AB ebenfalls zur Basistupelmenge von ABC hinzugefügt. Um den neuen Abdeckungsgrad des Knoten ABC zu ermitteln, ist es nicht ausreichend, $(1-0,83)$ an die Berechnung in Definition



Eigenschafts-attributmenge	Seed-Menge
AB	A
AC	A, C
AD	A, D
BC	A
CD	C, D
EF	E
ABC	AB, BC, A, C
BCD	BD, CD, C, D
ABCD	AB, BC, A, C, D

Abbildung 6.10: Beispiel zur Identifizierung von Basistupeln und die daraus resultierende Abhängigkeitstabelle (SDT)

6.3 anzufügen, da die 83% des Nullwertverhältnisses von AB wiederum durch A ableitbar sind, welches Element der Basistupelmenge von AB sowie ABC ist. Aus diesem Grund muss zunächst das Nullwertverhältnis von AB bezüglich der eigenen Basistupelmenge $S = \{A, C\}$ wie folgt berechnet werden:

$$\rho(AB, \{A, C\}) = 1 - \frac{(1 - 0,83)}{(1 - 0,5)} = 0,66.$$

Durch Anwendung dieses Nullwertverhältnis kann der Abdeckungsgrad des Knoten ABC neu berechnet werden: $1 - (1 - 0,5)(1 - 0,1)(1 - 0,66) = 0,85$.
Die resultierenden Abhängigkeitsregeln für dieses Beispiel sind in der Abbildung 6.10 rechts dargestellt. Die Abhängigkeitsregeln können dabei durchaus rekursiv definiert sein. So kommt zum Beispiel die Eigenschaftsattributmenge AB zweimal auf der rechten Seite einer Regel vor und wird in einer dritten Abhängigkeitsregel durch A abgedeckt. Des Weiteren können auch redundante Seed-Mengen, wie zum Beispiel $\{CD, C, D\}$ für BCD, auftreten. Unter der Annahme, dass die Eigenschaft 6.1 erfüllt ist, könnten entweder C und D oder CD aus der Seed-Menge gelöscht werden. Jedoch ist für reale Datensätze die Eigenschaft 6.1 nicht immer erfüllt, so dass die redundante Speicherung zu höheren Komprimierungsraten führen kann und daher beibehalten wird.

Nullwertkompression einer Tabelle

Im vorherigen Abschnitt wurden die Charakteristika der Basistupel beschrieben sowie ein Algorithmus zu deren Identifizierung von Abhängigkeitsregeln vorgestellt, der die Zuordnung zwischen Eigenschaftsattributkombinationen und den Seed-Mengen speichert. Anhand dieser kann für jedes Tupel einer zu komprimierenden Datentabelle die Menge der potenziellen Basistupel bestimmt werden. Die Abhängigkeitstabelle beschränkt demzufolge den Suchraum bei der Bestimmung von Basistupeln und ermöglicht dadurch eine effiziente Komprimierung. Zur Bestimmung der Basistupel

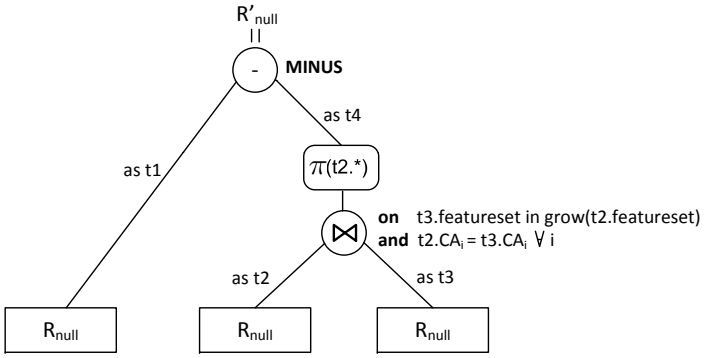


Abbildung 6.11: Anweisung zur Komprimierung der Partition R_{null}

für ein gegebenes Tupel t_i, bei vorhandener Abhängigkeitstabelle SDT, wird eine Funktion *Grow* benötigt.

Definition 6.4 (Grow) *Grow erhält als Eingabe ein Tupel t_i sowie die Abhängigkeitstabelle SDT und erzeugt eine Tupelmenge gröberer Granularität entsprechend der in SDT definierten Abhängigkeiten: $Grow(t_i, SDT) = T_j$, mit $F_{t_i} \prec F_{T_j}$.*

Die Funktionsweise von *Grow* wird an einem Beispiel erläutert: Gegeben sei das Eingabetupel $\{A = a_1, B = b_4, C = c_9\}$ und die Abhängigkeitstabelle aus Abbildung 6.10. Dieses Tupel ist eine Ausprägung der Eigenschaftsattributkombination ABC, für die ein Eintrag in der Abhängigkeitstabelle existiert. Die Anwendung der Funktion *Grow* gibt eine Menge aus vier Basistupeln, $\{A = a_1, B = b_4\}$, $\{B = b_4, C = c_9\}$, $\{A = a_1\}$ und $\{C = c_9\}$, zurück. Ohne das Vorhandensein der Abhängigkeitstabelle müssten zur Erzeugung potenzieller Basistupel alle Teilmengen einer Eigenschaftsattributkombination überprüft werden. Für die Eigenschaftsattributkombination $ABCD$ müssten in diesem Fall alle 14 ableitbaren Teilmengen überprüft werden.
Die zu komprimierende Tabelle R ist in zwei Partitionen, R_{value} und R_{null}, unterteilt. Erstere enthält alle Tupel, deren Faktattribute eine nicht nullwertige Ausprägung besitzen; zweitere enthält alle nullwertigen Tupel. Alle weitergehenden Betrachtungen beziehen sich ausschließlich auf die Partition R_{null}. Die Komprimierung der Partition R_{null} mit den zu ermittelnden Basistupeln wird durch die in Abbildung 6.11 dargestellte Anweisung realisiert. Durch einen Verbund der Partition R_{null} mit sich selbst wird ermittelt, ob ein Tupel von einem Basistupel ableitbar und somit redundant ist ($t2$ und $t3$ in Abbildung 6.11). Dazu wird die Funktion *Grow* auf alle Tupel in $t2$ angewendet, um die potenziellen Basistupel zu ermitteln. Durch einen Verbund mit $t3 = R_{null}$ wird berechnet, ob die potenziellen Basistupel existieren und somit die entsprechenden Tupel in $t2$ redundant sind oder nicht. Alle redundanten Tupel qualifizieren sich für das Teilergebnis $t4$ und werden über die Mengenoperation $MINUS$ aus der Ursprungsrelation $t1 = R_{null}$ entfernt. Durch die Mengenoperation werden gleichzeitig auch Duplikate eliminiert, die dadurch auftreten, dass ein

redundantes Tupel von mehreren Basistupeln ableitbar ist. Die resultierende komprimierte Tabelle R'_{null} enthält demzufolge nur noch Basistupel oder Ausreißer.
Das Verbundprädikat zwischen $t2$ und $t3$ enthält noch eine weitere Bedingung, durch die sichergestellt wird, dass nur Tupel innerhalb derselben Ausprägung der Klassifikationsdimensionen verglichen werden. Ein Tupel mit den Eigenschaftsattributwerten $\{Farbe =' wei', Marke =' VW'\}$ der Produktfamilie „Pickup“ kann somit nicht von einem Tupel $\{Farbe =' wei'\}$ der Produktfamilie „SUV“ abgeleitet werden. Aus diesem Grund muss jeweils die Gleichheit der Klassifikationsattribute CA_i in dem Verbundprädikat überprüft werden.

6.2.3 Anfrageverarbeitung auf nullwertkomprimierten Daten

In Kapitel 6.1 wurde der von operativen Änderungen entkoppelte Reporting-Layer vorgestellt. Die Anfrageverarbeitung mit Vollständigkeitsprüfung auf dem Reporting-Layer macht die Speicherung von Nullwerten notwendig. Die Realisierung der Anfrageverarbeitung auf nullwertkomprimierten Daten sowie deren Wartung bei Änderungsoperationen ist Inhalt dieses Abschnitts.

Partielle Dekomprimierung

Zur inhaltlich korrekten Beantwortung von Anfragen, insbesondere unter dem Aspekt der Vollständigkeitsprüfung, müssen die Nullwerte wieder rekonstruiert bzw. die Daten dekomprimiert werden. Dies geschieht zur Laufzeit der Anfragen und nur für den Teil der Daten, der zur Beantwortung der Anfragen notwendig ist. Die Anweisung, die diese partielle Dekomprimierung leistet und stets vor der eigentlichen Anfrageverarbeitung ausgeführt werden muss, ist in Abbildung 6.12 dargestellt. Zur Beantwortung einer Anfrage Q müssen zunächst die in dieser Anfrage vorkommenden Klassifikations- und Eigenschaftsattribute analysiert werden. Da nur der Teil der Daten dekomprimiert wird, der für die Anfrage relevant ist, erfolgt eine Selektion über die in Q vorkommenden Klassifikationsattribute und Ausprägungen. Dies wird durch die Menge der Prädikate $CA_i = Pred_i$, mit $Pred_i \in Q$, realisiert. Das Ergebnis dieser Operation erhält das Alias $t1$.
Die in der Anfrage Q vorkommenden Instanzen der Eigenschaftsattribute werden in einer temporären Tabelle *Temp_Feature*, alias $t2$, gespeichert. Durch Anwendung der Funktion *Grow* auf die Tabelle *Temp_Feature* und einen Verbund mit $t1$ kann ermittelt werden, ob ein Tupel der potenziellen Ergebnismenge aus den in $t1$ gespeicherten Basistupeln ableitbar ist. Wird ein angefragtes Tupel durch mindestens ein Basistupel aus R'_{null} abgedeckt, so folgt daraus, dass dieses Tupel auf Basis von R'_{null} beantwortbar und somit Teil der Ergebnismenge ist. Neben den ableitbaren bzw. redundanten Tupeln sind dies auch die Basis- und die Ausreißertupel, die das zweite Verbundprädikat, $t1.featureset = t2.featureset$, erfüllen. Zur Eliminierung von Duplikaten, die durch mehrfache Ableitbarkeit oder durch Aktualisierungen (siehe nächster Abschnitt) entstehen können, ist eine abschließende *GROUP-BY*-Operation erforderlich. Das Ergebnis ist eine partielle Dekomprimierung von R'_{null}.

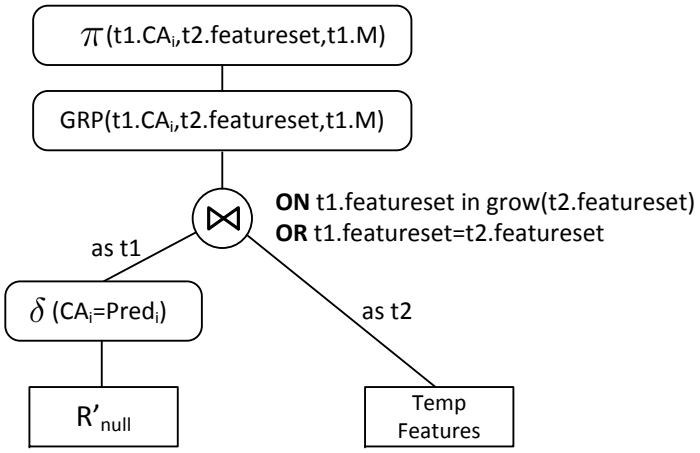


Abbildung 6.12: Partielle Dekomprimierung der Partition R'_{null}

Es bleibt zu bemerken, dass die durch die Dekomprimierung erzeugten Tupel nicht zwingend in der Originalrelation R_{null} vorhanden sein mussten. So kann zum Beispiel eine Anfrage, die das Tupel $\{A = a_{11}, D = d_7\}$ selektiert, beantwortet werden, falls die Basistupel $\{A = a_{11}\}$ oder $\{D = d_7\}$ Teil der komprimierten Tabelle R'_{null} sind. Das Tupel $\{A = a_{11}, D = d_7\}$ muss dazu nicht notwendigerweise Teil der Originalrelation R_{null} gewesen sein.

Inkrementelle Wartung der komprimierten Daten

Das Identifizieren der Basistupel sowie die Komprimierung einer Datentabelle ist – abhängig von deren Größe und der Anzahl der Eigenschaftsattribute – ein aufwendiger Prozess (siehe Abschnitt 6.2.5). Dieser kann somit nicht nach jeder Änderung an der Basistabelle neu ausgeführt werden, sondern muss inkrementell duchführbar sein.

Einfügen Zunächst wird das Einfügen neuer Tupel in die komprimierte Tabelle R' betrachtet. Das einzufügende Tupel wird mit t_{new} bezeichnet. Zwei Fälle sind beim Einfügen zu unterscheiden:

- Fall 1: Ist $M(t_{new}) <> null$, wird t_{new} in R'_{value} eingefügt und $FeatureSetNode$ aktualisiert, d.h. C_v um eins inkrementiert.

- Fall 2: Ist $M(t_{new}) = null$, muss bestimmt werden, ob t_{new} von einem bereits vorhandenen Tupel ableitbar ist:
  - Ist t_{new} ableitbar, wird es nicht in R'_{null} eingefügt und $FeatureSetNode$ wird aktualisert, d.h. C_n wird inkrementiert.

– Ist t_{new} nicht ableitbar, wird das Tupel in R'_{null} eingefügt, $FeatureSetNode$ aktualisiert und alle Tupel in R'_{null}, die von t_{new} ableitbar sind, werden gelöscht.

Durch das Vorgehensmodell wird garantiert, dass keine redundanten Tupel in die komprimierte Tabelle R' eingefügt werden. Dadurch ist zu jedem Zeitpunkt eine optimale Komprimierung gewährleistet (*Sofortiger Modus*). Zur Überprüfung der Redundanz ist jedoch ein Test auf Ableitbarkeit und somit eine partielle Dekomprimierung notwendig. Alternativ kann die Prüfung auf Ableitbarkeit eines Tupels t_{new} entfallen und das Einfügen dahingehend vereinfacht werden, dass jedes Tupel in R' eingefügt und $FeatureSetNode$ entsprechend aktualisiert wird. Dieses Vorgehen wird als *Verzögerter Modus* bezeichnet und ermöglicht durch das Einführen von Redundanz ein schnelleres Einfügen. Duplikate, die beim Einfügen redundanter Tupel entstehen, werden durch die partielle Dekomprimierung (siehe vorheriger Abschnitt) kompensiert.

Löschen Wie auch beim Einfügen von Tupeln, sind beim Löschen zwei Fälle zu unterscheiden. Mit t_{del} wird das zu löschende Tupel bezeichnet.

- Fall 1: Ist $M(t_{del}) <> null$, wird t_{del} aus R'_{value} gelöscht und $FeatureSetNode$ aktualisiert, d.h. C_v um eins dekrementiert.

- Fall 2: Ist $M(t_{del}) = null$, muss bestimmt werden, ob t_{del} ein Basistupel ist:
  – Ist t_{del} ein Basistupel, dann impliziert das Löschen von t_{del} auch das Löschen aller Tupel, die ausschließlich von t_{del} abhängig sind. Wird zum Beispiel das Basistupel $\{A = a_1\}$ gelöscht, so ist nach dieser Semantik auch das Tupel $\{A = a_1, D = d_7\}$ zu entfernen. Da durch das Löschen eines Basistupels viele abhängige Nulltupel implizit mit entfernt werden, ist es notwendig, die Anzahl der Nullwertvorkommen im Graph zu aktualisieren. Dies ist nur heuristisch mittels *Eigenschaft 3.1* möglich, da keine exakten Informationen über die Anzahl der Abhängigkeiten existieren.
  Das aktualisierte Nullwertverhältnis C'_n einer Eigenschaftsattributkombination F_j, welche von dem Basistupel t_{del} mit der Eigenschaftsattributkombination F_i abhängig ist ($F_j \prec F_i$), wird folgendermaßen bestimmt:

$$C'_n(F_j) = C_n(F_j) - \frac{C_n(F_j)}{C_n(F_i) \cdot \rho(F_i)}.$$

  Am Beispiel der Eigenschaftsattributkombination AB mit $C_n(AB) = 20.000$ und A mit $C_n(A) = 50$ und einem Nullverhältnis $\rho(A) = 0,5$ ergibt sich $C'_n(AB)$ zu

$$C'_n(AB) = C_n(AB) - \frac{C_n(AB)}{C_n(A) \cdot \rho(A)} = 20.000 - \frac{20.000}{50 \cdot 0,5} = 19.200$$

Das bedeutet, dass durch Entfernen des Basistupels $\{A = a_1\}$ implizit 800 weitere redundante Tupel gelöscht werden und somit aus der Zählung der Nulltupel entfallen. Weiterhin muss die $FeatureSetNode$-Tabelle aktualisiert, d.h. C_n dekrementiert werden.

- Ist t_{del} kein Basistupel, dann kann t_{del} gelöscht und C_n um eins dekrementiert werden.

Aktualisieren Die Aktualisierung eines Tupels wird durch das Ausführen der Operationen Löschen und Einfügen umgesetzt.

Inkrementelle Rekomprimierung

Die oben beschriebenen Aktualisierungsschritte führen zu Änderungen der Kardinalitäten der Null- und Wertetupel (C_n und C_v), welche in der $FeatureSetNode$-Tabelle gespeichert sind. Als Folge verändern sich auch die Nullwertverhältnisse der Eigenschaftsattributkombinationen, wodurch die Seed-Mengen und damit die Abhängigkeitstabelle invalidiert werden. Bisher gültige Abhängigkeitsregeln veralten bzw. neue Abhängigkeitsregeln kommen hinzu. Um ein optimales Speicherplatzverhalten zu gewährleisten, muss die Komprimierung daher regelmäßig ausgeführt werden. Die Veränderungen der Abhängigkeitsregeln sind dazu transparent zu machen, indem die Abhängigkeitstabelle um eine Spalte „Status“, mit den Ausprägungen *fortwährend*, *neu* und *veraltet*, erweitert wird (siehe Abbildung 6.13). Des Weiteren werden die Seed-Mengen in ihre Elemente zerlegt und einzeln abgespeichert. Bei der Neuausführung des Algorithmus zur Ermittlung der Abhängigkeitsregeln wird jeweils eine neue Abhängigkeitstabelle SDT' angelegt. Eine Abhängigkeitsregel, die in der neuen Tabelle SDT' vorhanden ist, nicht aber in der vorhergehenden Version SDT ($r \in SDT' \wedge r \notin SDT$), wird in SDT eingefügt und als „neu“ gekennzeichnet. Abhängigkeitsregeln, die in beiden Tabellen vorhanden sind, d.h. $r \in SDT' \wedge r \in SDT$, werden mit „fortwährend“ markiert. Überholte Regeln, $r \notin SDT' \wedge r \in SDT$, werden mit „veraltet“ gekennzeichnet.

Die Statusinformation einer Abhängigkeitsregel wird bei der Komprimierung wie folgt berücksichtigt: Unter Anwendung des *Sofortigen Modus* sind jeweils nur die mit nneummarkierten Regeln zu verwenden, da als „fortwährend“ markierte Regeln bereits bei der Aktualisierung angewendet wurden. Im *Verzögerten Modus* hingegen müssen sowohl die mit „neu“ als auch mit „fortwährend“ markierten Regeln berücksichtigt werden. Veraltete Regeln werden nicht mehr angewendet, da sie zu einem schlechteren Komprimierungsverhältnis im Vergleich zu den aktuellen Regeln führen. Jedoch können sie nicht aus der Abhängigkeitstabelle entfernt werden, solange noch Basistupel in R'_{null} vorhanden sind, die durch diese Regeln bestimmt sind.

6.2.4 Verwandte Arbeiten und Techniken

Klassische Techniken zur Datenkompression basieren meist auf Statistiken oder Wörterbüchern [96, 36]. Diese Methoden sind syntaktischer Natur, da sie eine Tabelle

Eigenschafts-attributmenge	Seed	Status
AB	A	current
AC	A	current
AC	C	out-dated
AD	A	current
AD	D	current
CD	C	current
…	…	…
ABC	AB	current
ABC	BC	out-dated
ABC	AC	new
…	…	…

Abbildung 6.13: Erweiterte Abhängigkeitstabelle

als eine einzige große Byte-Kette verstehen. Zur Vermeidung der Dekompression der kompletten Datenbank während eines Tabellenzugriffs werden üblicherweise einzelne Tupel oder Attribute komprimiert. Für kleine Byte-Sätze eingesetzt, ist die syntaktische Kompression jedoch für gewöhnlich nicht sehr effizient.
Im Umfeld von MOLAP sind vollständige Kompressionsmethoden, wie die sogenannte *Header Compression* [23] oder die *Chunk-Offset Compression* [93] weit verbreitet; bei diesen Methoden müssen Nachbarschaften in den multidimensionalen Daten erhalten bleiben.
Modernere Ansätze, die auch die Semantik bei der Kompression berücksichtigen, sind [7, 47, 40, 48, 1]. Die aktuelleren Arbeiten zu *Condensed Cubes* [1] und *Quotient Cubes* [47, 48] gruppieren Tupel mit ähnlichen Aggregationswerten in gemeinsame Partitionen. Dies ist im vorliegenden Ansatz nur bedingt möglich, da die Mehrzahl der Tupel mit Nullwerten besetzt ist, die gemeinsam eine sehr große Partition bilden. Die *Condensed Cubes* extrahieren spezielle Tupel und erzeugen ein zusätzliches Dimensionsattribut, wodurch eine Menge von Tupeln repräsentiert wird. Durch Anwendung einer einfachen Operation kann ein solches Tupel expandiert und somit können alle ursprünglichen Tupel wieder hergestellt werden. Dies ähnelt der Idee der *Basistupel*, durch die abgedeckte Tupel entfernt bzw. rekonstruiert werden können. Die *Quotient Cubes* unterteilen die Würfel in Klassen, so dass jede Zelle einer Klasse denselben Aggregationswert enthält. Die Kompression ergibt sich dadurch, dass nur die Zellen an der unteren und oberen Grenze pro Klasse gespeichert werden müssen.
Einen anderen Ansatz verfolgt das Konzept der „Eisbergwürfel“ (engl. iceberg cubes), welches nur diejenigen Group-by-Kombinationen vorberechnet, deren Aggregationswert über einem bestimmten Schwellwert für den minimalen Support liegt. Dieser Ansatz wurde zuerst durch [9] vorgestellt. In der Praxis ist es oftmals schwierig, diesen Schwellwert festzulegen, insbesondere auch bei dem vorliegenden Ansatz, in dem auch sehr dünnbesetzte Datenräume gespeichert werden müssen.

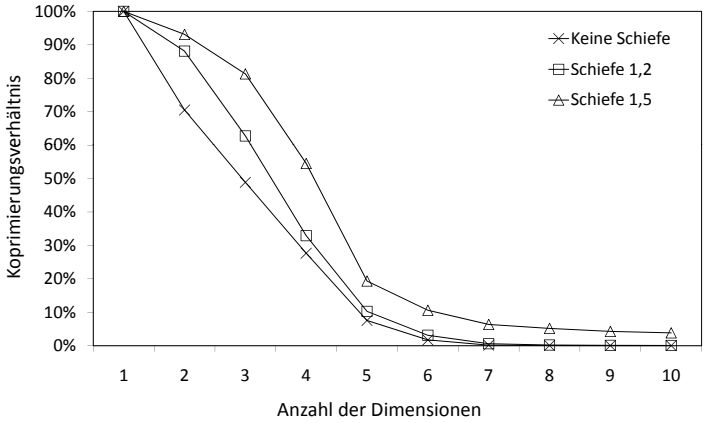
Dwarf Cubes sind komprimierte Datenstrukturen, welche die Größe der Datenwürfel verringern können, indem sie Präfix- und Suffixredundanzen beseitigen. Der Grad der Kompression basiert auf der Dichte der multidimensionalen Rohdaten [69, 70].
Die minimale Beschreibungslänge (engl. *Minimum Description Length*, MDL) ist eine informationstheoretische Methode zur Beschreibung der Regelmäßigkeit bzw. der Strukturiertheit in Daten. Je höher die Regelmäßigkeit, umso stärker ist der zu erreichende Kompressionsgrad. Im OLAP-Umfeld wird diese Methodik in [13] zur Komprimierung verwendet. Dazu werden Zusammenfassungen der Form $S\theta H$ für k-dimensionale Würfel erzeugt, wobei S die Anfrageergebnisse generalisiert und H alle Ausnahmen dieser Generalisierung beschreibt.
In [32] wird die Datenbankkompression mit Data-Mining-Techniken vorgeschlagen. Durch Anwendung des *Apriori-Algorithmus* werden Regeln extrahiert, anhand derer die Daten mit geringem Speicheraufwand abgelegt werden. Mit Hilfe der Regelmenge ist der Originalzustand wiederherstellbar.

6.2.5 Evaluierung

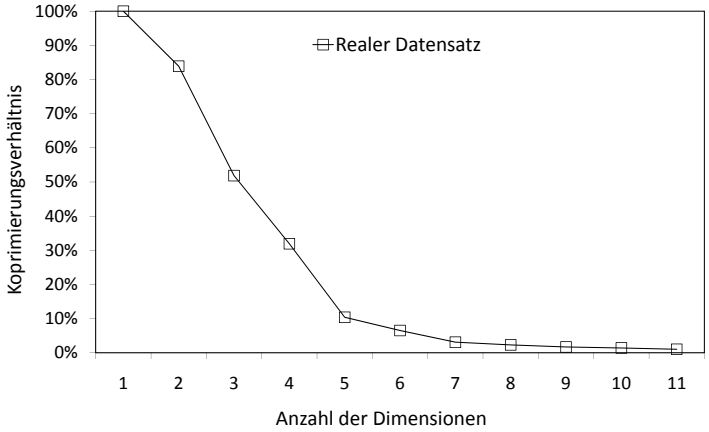
Die Leistungsfähigkeit des Kompressionsalgorithmus, die Anfrageperformanz auf den komprimierten Daten sowie deren Wartung wurde in einer Reihe von Experimenten untersucht.

Experimentierumgebung

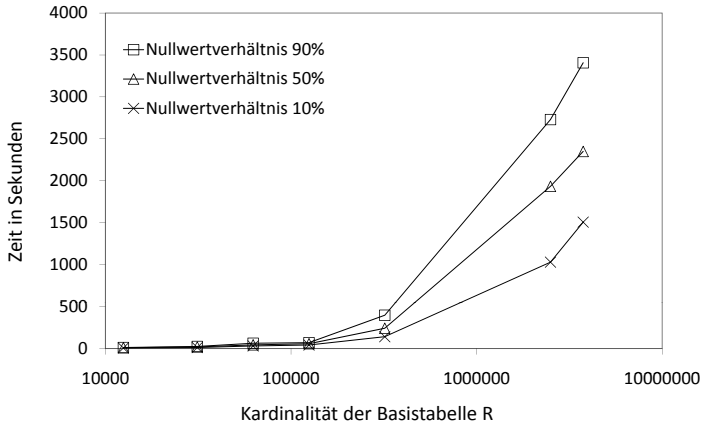
Zur Untersuchung der vorgestellten Algorithmen wurden sowohl synthetische als auch reale Datensätze verwendet. Die synthetischen Daten können bezüglich des Nullwertverhältnisses, der Anzahl der Dimensionen und Tupel sowie der Verteilung variiert werden. Die Verteilung bzw. die Schiefe der Daten kann über einen Parameter S veändert werden. Die in *Eigenschaft 6.1* definierte Gleichverteilung ist bei einem Wert $S = 1,0$ gegeben. Für diese Verteilung ist auch die zu erreichende Komprimierungsrate maximal. Jedoch können auch alternative Verteilungen gewählt werden. Eine Schiefe von $S = 1,2$ bedeutet zum Beispiel, dass 20% aller Ausprägungen einer Eigenschaftsattributkombination F_i nicht in den Nachfolgeknoten, $F_j \prec F_i$, vorkommen. Dies ist in Anwendungsszenarien der Fall, in denen die Nutzer nicht alle Aggregate feinster Granularität selektiert haben. Weiterhin wurde ein realer Datensatz aus der Marktforschung betrachtet, in dem die Verkaufsinformationen zu bestimmten Produkten hinterlegt sind. Dieser enthält ca. 3 Mio. Tupel (340 MB) in 11 Dimensionen. Das Nullwertverhältnis des gesamten Datensatzes beträgt aufgrund der Dünnbesetztheit des Datenraums cirka 99%.
Die Ausführung aller Experimente erfolgte auf einem Intel Celeron mit 2,66 GHz, Windows-Betriebssystem und 2 GB Speicher.



(a) Komprimierungsverhältnis für synthetische Daten



(b) Komprimierungsverhältnis für reale Daten



(c) Laufzeitverhalten für unterschiedliche Tabellengrößen und Nullwertverhältnisse

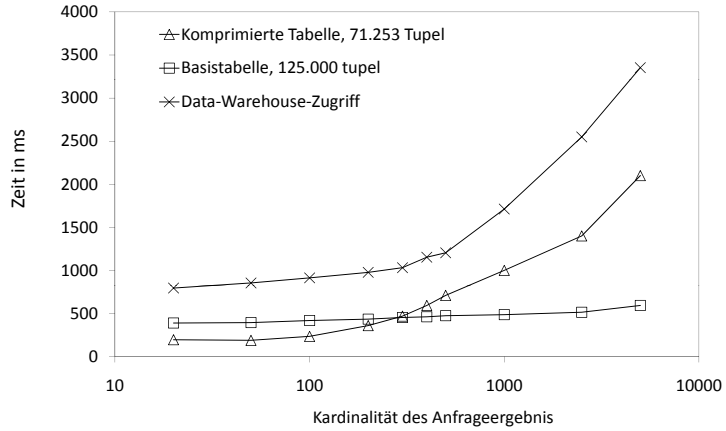
Abbildung 6.14: Evaluierung der Nullwertkomprimierung

Komprimierung

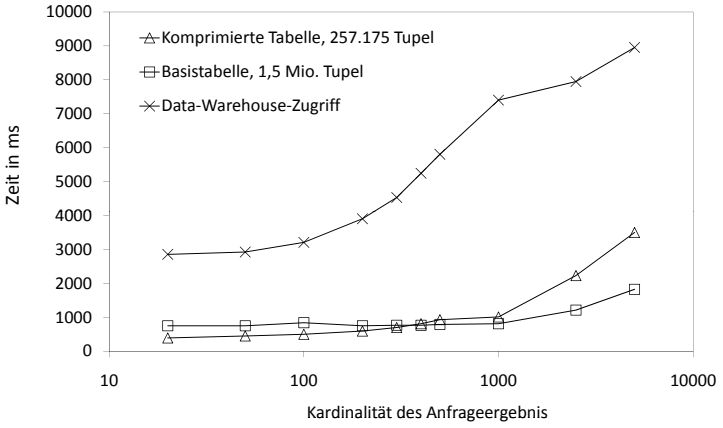
In einer ersten Reihe von Experimenten wird die Güte der Komprimierung untersucht. Die entsprechenden Ergebnisse sind in den Abbildungen 6.14a für den synthetischen bzw. in 6.14b für den realen Datensatz dargestellt.
Die Anzahl der Tupel für den synthetischen Datensatz war auf 2,5 Mio. fesgelegt, wohingegen die Schiefe von 1,0 auf 1,25 und 1,5 verändert wurde. Abbildung 6.14a stellt das Komprimierungsverhältnis für diese drei Datensätze in Abhängigkeit von der Dimension der Daten, zwischen 1 und 10 variiert, dar. Es ist festzustellen, dass für eine wachsende Anzahl von Dimensionen das Komprimierungsverhältnis kontinuierlich verbessert werden kann. Je höher die Zahl der Dimensionen, umso komplexer wird der Graph aus Eigenschaftsattributkombinationen, was zu besseren, d.h. für den Datensatz allgemein gültigeren Abhängigkeitsregeln führt. Für eine 10-dimensionale Basistabelle R_{null} kann der gleichverteilte Datensatz sogar bis auf 0,05% seiner Ursprungsgröße reduziert werden. Für nicht gleichverteilte Daten ist das Komprimierungsverhältnis wie erwartet schlechter, da nicht alle Ausprägungen gleich häufig vorkommen und daher unterschiedlich stark von den Basistupeln abgedeckt werden. Jedoch wird auch für einen 10-dimensionalen Datensatz der Schiefe $S1,5$ immer noch ein Komprimierungsverhältnis von 4% erreicht.
Auch für den in Abbildung 6.15b dargestellten realen Datensatz wurde das Komprimierungsverhältnis mit steigender Anzahl der Dimensionen kontinuierlich verbessert. Bei Betrachtung aller 11 Dimensionen kann der Datensatz auf 1% seiner Ursprungsgröße reduziert werden.
Der Mehraufwand, welcher zur Speicherung der Abhängigkeitsregeln notwendig ist, wurde in den Experimenten bereits berücksichtigt und ist vernachlässigbar gering. Am Beispiel des synthetischen Datensatzes mit 2,5 Mio. Tupeln und 10 Dimensionen sind dies nur 226 zu speichernde Abhängigkeitsregeln.
Weiterhin wurde das Laufzeitverhalten der Nullwertkomprimierung für verschiedene Tabellengrößen und Nullwertverhältnisse untersucht. Das Ergebnis ist in Abbildung 6.14c dargestellt. Es ist zu beobachten, dass die Ausführungszeit der Nullwertkomprimierung sowohl mit wachsender Kardinalität der Basistabelle als auch mit steigendem Nullwertverhältnis ansteigt. Beide Faktoren führen zu einer größer werdenden Nullwertpartition R_{null}, wodurch die zur Komprimierung notwendigen Operationen verteuert werden (siehe Abbildung 6.11).

Anfrageverarbeitung auf nullwertkomprimierten Daten

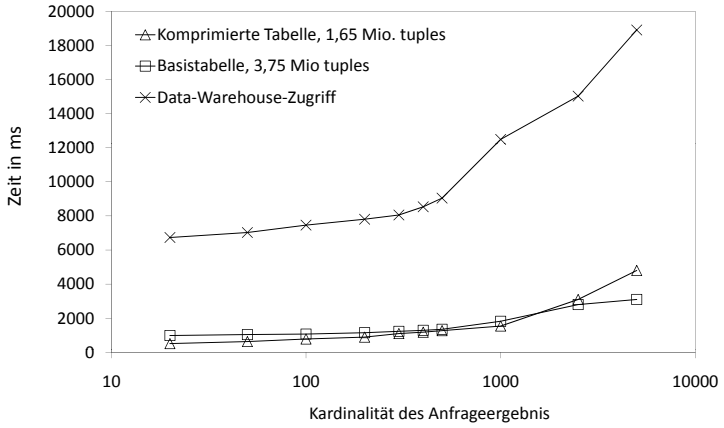
Die Auswirkungen der partiellen Dekomprimierung auf die Performanz der Anfrageverarbeitung sind Inhalt der folgenden Experimente. Dazu wurden drei verschiedene Datensätze unterschiedlicher Kardinalitäten und Nullwertverhältnisse erzeugt (siehe Abbildung 6.15). Der erste Datensatz R_1 umfasst 125.000 Tupel und beinhaltet Nulltupel im Umfang von 50%. Datensatz R_2 besteht aus 1,5 Mio. Tupeln, wovon 75% mit Nullwerten besetzt sind, und Datensatz R_3 umfasst 3,75 Mio. Tupel, mit einem Nullwertanteil von 50%. Zur Bewertung der Anfrageverarbeitung wurden drei



(a) $|R_1| = 125.000$



(b) $|R_2| = 1,5\ Mio$



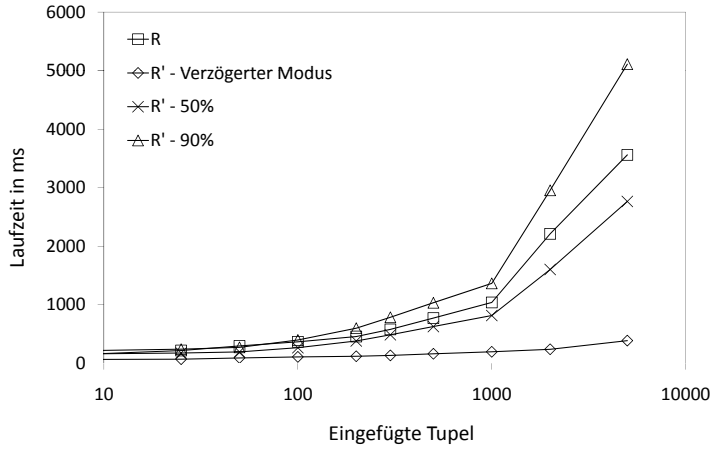
(c) $|R_3| = 3,75\ Mio$

Abbildung 6.15: Evaluierung der Anfrageverarbeitung auf nullwertkomprimierten Daten

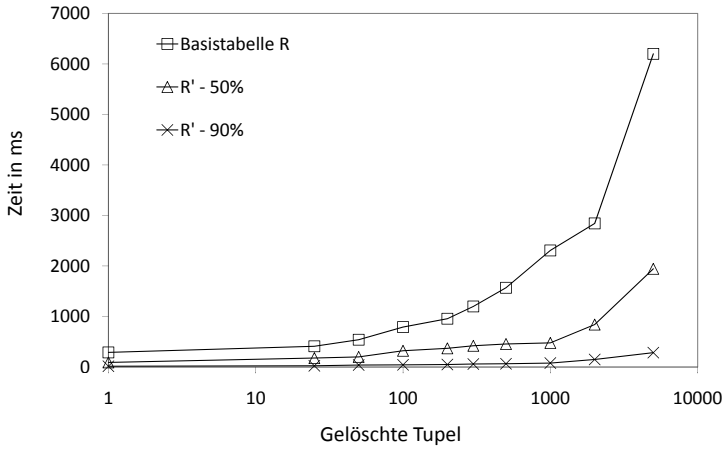
verschiedene Zugriffsarten unterschieden: die Beantwortung der Anfragen auf Basis der Originalrelation R und der komprimierten Relation R' sowie auf Basis einer Relation $R_{\neg null}$, die keine Nullwerte enthält, so dass zur Bestimmung der Vollständigkeit Zugriffe auf die Basisrelation im Data-Warehouse notwendig sind.
Für alle drei Datensätze und die verschiedenen Modi der Anfrageverarbeitung wurde die Kardinalität der von den Anfragen benötigten Tupel variiert. Greifen die Anfragen – im Sinne einer Punktanfrage – nur auf wenige Tupel zu, so ist die Anfrageverarbeitung auf den komprimierten Daten am schnellsten. Bei steigender Tupelanzahl fällt zunehmend die Operation der partiellen Dekomprimierung ins Gewicht, so dass ab einem gewissen Punkt die Anfragen auf den unkomprimierten Daten schneller verarbeitet werden können. Die Ausführung von Anfragen auf einer Relation $R_{\neg null}$, die keine Informationen über Nullwerte enthält, ist in jedem Fall schlechter, da weitere Zugriffe auf das Data-Warehouse notwendig werden.

Wartung der komprimierten Daten

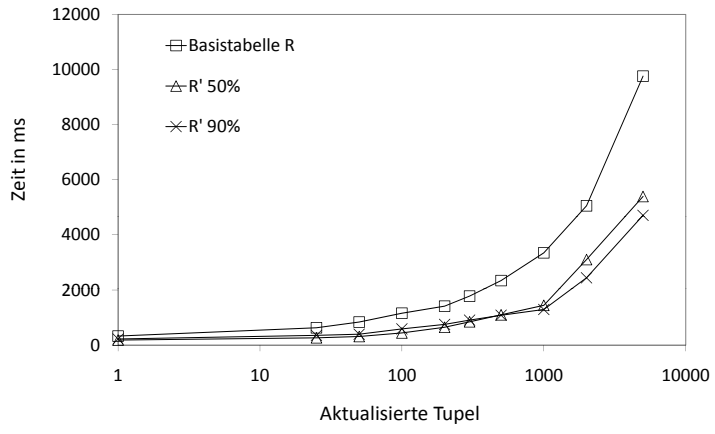
Der Aufwand der inkrementellen Wartung bei Änderungen an der komprimierten Relation wird in einer letzten Reihe von Experimenten untersucht. Die Größe der Relation R war auf 2,5 Mio. Tupel und das Nullwertverhältnis auf 50% bzw. 90% festgelegt.
Die Ergebnisse für die Einfügeoperation sind in Abbildung 6.16a dargestellt. Das Einfügen im *Verzögerten Modus* erzielt die besten Ausführungsgeschwindigkeiten, da die komprimierte Relation R' klein im Vergleich zur Originalrelation R ist und der teure Dekomprimierungsschritt entfällt. Unter Anwednung des *Sofortigen Modus* ist die Laufzeit der Einfügeoperation vor allem von dem Nullwertverhältnis der Ursprungsrelation abhängig. Bei einem geringen Nullwertverhältnis von 50% erfolgt das Einfügen noch schneller als auf der Originalrelation R, während bei einem Nullwertverhältnis von 90% das Einfügen bereits deutlich langsamer ist. Der Grund hierfür ist der im Dekomprimierungsschritt erzeugte Datenraum, welcher mit steigendem Nullwertverhältnis zunehmend größer wird und dadurch die zur Dekomprimierung notwendige Verbundoperation verteuert (siehe Anfrageplan in Abbildung 6.12).
Die Kosten der einzelnen Operationsschritte für das Einfügen von Tupeln und der inkrementellen Wartung sind in Abbildung 6.17 aufgeschlüsselt. Es ist zu sehen, dass die Bestimmung, ob ein Tupel abgedeckt ist, den Anteil an den Gesamtkosten mit zunehmender Anzahl einzufügender Tupel dominiert. Für 5.000 einzufügende Tupel beträgt der Anteil dieser Operation 85% an den Gesamtkosten. Im Gegensatz dazu bleibt die Bestimmung, ob die einzufügenden Tupel Basistupel sind, auch mit steigender Tupelanzahl konstant. Die dafür notwendige Anfrage an die Abhängigkeitstabelle, die im Vergleich zur komprimierten Relation R' nur sehr wenige Tupel enthält, kann für größere Einfügeoperationen vernachlässigt werden.
Abschließend wurde der Wartungsaufwand der komprimierten Daten unter Lösch- und Änderungsoperationen betrachtet (siehe Abbildungen 6.16b and 6.16c). Die Größe der Relation R beträgt wiederum 2,5 Mio. Tupel und das Nullwertverhältnis ist auf 50% bzw. 90% festgelegt. Für die Löschoperation kann gezeigt werden, dass ein



(a) Einfügen



(b) Löschen



(c) Aktualisieren

Abbildung 6.16: Wartung der komprimierten Daten ($|R| = 2.5\ Mio$)

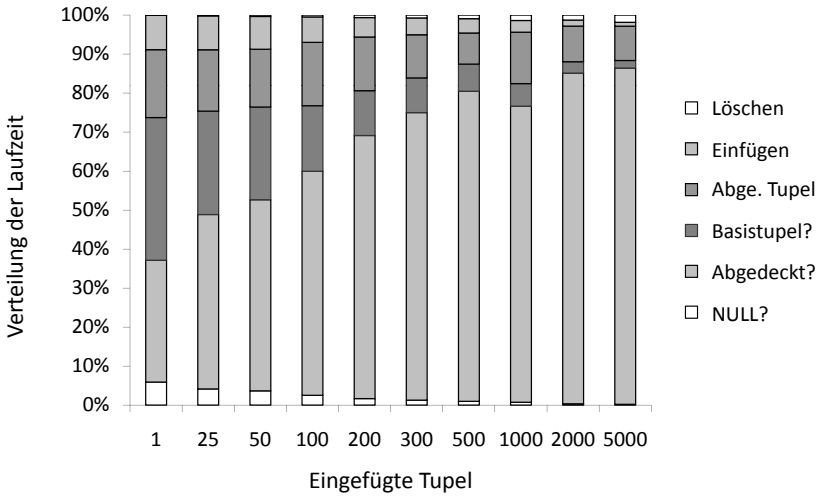


Abbildung 6.17: Kostenverteilung der einzelnen Schritte der Einfügeoperation

Löschen auf den komprimierten Relationen schneller durchgeführt werden kann als auf der Ursprungsrelation R. Die Ursache liegt zum einen in der geringeren Größe der Relation selbst, aber vor allem auch in dem verminderten Aufwand bei der inkrementellen Wartung im Vergleich zur Einfügeoperation. Beim Löschen eines Tupels muss lediglich festgestellt werden, ob dieses Tupel ein Basistupel ist. Dies ist eine vergleichsweise günstige Operation, wie im vorhergehenden Experiment gezeigt.
Aktualisierungen werden durch die kombinierte Ausführung einer Lösch- und einer Einfügeoperation umgesetzt. Dementsprechend ist zu erwarten, dass die Evaluierung der Laufzeit für die Aktualisierungen eine Mischung aus den vorhergehenden Ergebnissen für Laufzeiten der Lösch- und Einfügeoperationen ergibt. Das Ergebnis ist in Abbildung 6.16c dargestellt. Für kleinere Mengen zu aktualisierender Tupel ist das Aktualisieren auf dem komprimierten Datensatz am schnellsten. Mit zunehmender Tupelanzahl verstärkt sich das Gewicht der teuren Einfügeoperation, so dass der Datensatz mit einem geringeren Nullwertverhältnis von 50% die besseren Ergebnisse liefert.

6.3 Zusammenfassung

Die in Echtzeit-Data-Warehouse-Systemen geforderte Datenaktualität wird vor allem durch die push-basierte Progagierung von Aktualisierungen erreicht. Operative Prozesse, die lediglich einen aktuellen Zustand benötigten, profitieren von dieser Aktualisierungssemantik. In der Berichtsproduktion führt die fortwährende Aktualisierung zu Inkonsistenzen. Der im ersten Teil dieses Kapitels vorgestellte Reporting-Layer entkoppelt das, von operativen Datenänderungen unterworfene Echtzeit-Data-Warehouse-System von der Berichtsproduktion und vermeidet unkontrollierte Än-

derungen berichtsrelevanter Daten. Dazu werden die Daten zu definierten Publikationszeitpunkten für die Berichtsproduktion freigegeben. Zwischen den Publikationszeitpunkten ist garantiert, dass sich die Daten mit sehr hoher Wahrscheinlichkeit nicht mehr ändern. Die somit erreichte Datenstabilität erlaubt den Einsatz spezieller Aggregationstechniken, um die Produktion von Standardberichten zu beschleunigen. Dazu werden alle Aggregate in einer festgelegten Granularität vorgehalten, um so eine hohe Wiederverwendbarkeit zu erreichen. Des Weiteren sind auch Ad-hoc-Berichte auf dem Reporting-Layer berechenbar. Diese können – sofern vorhanden – ebenfalls von bereits materialisierten Aggregaten profitieren. Die Vollständigkeit der Berichtsanfragen ist durch die Protokollierung der publizierten Daten sowie durch spezielle Anfragetechniken zu jeder Zeit garantiert. Der durch die Vollständigkeitsprüfung verursachte Mehraufwand ist dabei minimal, so dass die Standardberichtsanfragen nicht verzögert werden.
Der zweite Teil dieses Kapitels befasste sich mit der Komprimierung von Nullwerten, die zur korrekten Vollständigkeitsbestimmung notwendig sind. Die explizite Speicherung von Nullwerten erlaubt die Unterscheidung in Aggregate, die noch nicht vorberechnet wurden, weil bisher kein Bericht das Aggregat benötigte und in Aggregate, die bereits berechnet wurden, für die jedoch keine Basisdaten vorlagen. Experimente haben gezeigt, dass aufgrund der Dünnbesetztheit multidimensionaler Datensätze der Anteil der Nullwerte leicht 95% des gesamten Datensatzes ausmachen kann. Zur Reduzierung des Speicheraufwands von nullwertbesetzten Daten wurde ein Verfahren entwickelt, das durch Extraktion und Anwendung sogenannter Abhängigkeitsregeln die Daten komprimiert. In Experimenten mit synthetischen und realen Daten konnte gezeigt werden, dass die komprimierten Daten nur noch bis zu 0,05% der Ursprungsgröße ausmachen. Die für die Anfrageverarbeitung notwendigen Eigenschaften der Nullwerte bleiben durch Anwendung partieller Dekomprimierungstechniken erhalten.

Fallstudiendiskussion

Das Data-Warehouse der UBS WM&SB wird in seiner jetzigen Form lediglich für strategische Analysen, zumeist durch die Erzeugung von Berichten, verwendet. Jedoch lässt sich, wie in Abschnitt 2.1.3 dargestellt, prognostizieren, dass der Bedarf nach hochaktuellen Daten in der Zukunft steigen wird. Dessen ungeachtet, wird die periodische Berichtsproduktion kaum an Relevanz verlieren. Daraus resultiert ein Konflikt zwischen der Forderung nach Aktualität bzw. kurzen Aktualisierungszyklen auf der einen Seite und dem Bedürfnis nach Stabilität für die Berichtserzeugung auf der anderen Seite. Ein pragmatischer Lösungsansatz besteht darin, die bereits eingebrachten Echtzeitdaten zum Tagesabschluss zu verwerfen und die Daten wie bisher in langlaufenden Batch-Prozessen zu laden. Somit wäre die für die Berichtsproduktion benötigte Datenqualität gewährleistet, allerdings würde Datenintegrationsarbeit doppelt geleistet werden. Eine zusätzliche Datenschicht zur Trennung der Echtzeitanalysedaten von den Daten zur Berichtserzeugung würde dieses Problem beseitigen. Die notwendige Stabilität ist dabei durch definierte Veröffentlichungszeitpunkte ge-

währleistet.
Im Data-Warehouse der GfK Marketing Services ist der hier vorgestellte Ansatz bereits heute anwendbar. Durch nachträgliche Datenkorrekturen bzw. durch die Aufnahme verspäteter Datenlieferungen sind die Daten der GfK Marketing Services nicht stabil, was zu Problemen bei der Berichtsproduktion führt. Marktberichte für einzelne Länder können sofort produziert werden, sobald die Daten auf nationaler Ebene verfügbar sind. Internationale Berichte, wie zum Beispiel ein Europabericht, können erst erstellt werden, wenn alle Länderdaten vorhanden sind. Zwischenzeitliche Fehlerkorrekturen, die unter Umständen erst nach Auslieferung der Länderberichte durchgeführt wurden, haben zur Folge, dass die im Europabericht für ein Land ausgewiesenen Zahlen von den Länderberichten abweichen können. Derartige Inkonsistenzen sind dem Kunden nur schwer zu vermitteln und müssen vermieden werden. Durch die Einführung einer zusätzlichen Datenschicht, dem Reporting-Layer, kann dies gewährleistet werden. Ein weiterer Vorteil des Reporting-Layer besteht darin, dass keine materialisierten Sichten mehr angelegt werden müssen, sondern stattdessen Aggregate in einer einheitlichen Granularität erzeugt werden. Die Granularität wird so gewählt, dass alle Anfragen von den vorberechneten Aggregaten profitieren können.

7 Zusammenfassung und Ausblick

Data-Warehouse-Systeme haben sich im Laufe der letzten Jahre zur zentralen Plattform der Informationsversorgung von Unternehmen und Organisationen entwickelt. Dies schließt sowohl strategische und taktische Analysen basierend auf vorberechneten Berichten oder multidimensionalen Würfeln als auch die Unterstützung operativer Unternehmensprozesse mit ein. Letzteres macht die Integration von Daten in Echtzeit notwendig, was jedoch im Konflikt zu klassischen Data-Warehouse-Anforderungen steht. Im Detail konnten für ein Echtzeit-Data-Warehouse-System (i) die Maximierung der Datenaktualität, (ii) die Minimierung der Anfragelatenz und (iii) der Erhalt der Datenstabilität trotz operativer Änderungen als zentrale Optimierungsziele herausgestellt werden. In dieser Dissertation wurden neue Methoden und Techniken entwickelt die diese konfligierenden Ziele adressieren und die in Summe den Betrieb eines Echtzeit-Data-Warehouse-System ermöglichen.

Datenaktualität und Anfragelatenz Der von Echtzeit-Data-Warehouse-Systemen zu verarbeitende kontinuierliche Strom aus Anfragen und Aktualisierungen erzeugt einen Konflikt zwischen der Datenaktualität und der Anfragelatenz. Für beide Qualitätsdimensionen wurden in Kapitel 4 verschiedene Metriken evaluiert und getrennte Maximierungs- und Minimierungsprobleme formuliert. Die gemeinsame Betrachtung dieser beiden Probleme führte zu einem multikriteriellen Optimierungsproblem. Es wurde gezeigt, dass die Position der Aktualisierungen in der Anfragewarteschlange durch einen Kosten-Profit-Vektor dargestellt werden kann, wodurch das multikriterielle Optimierungsproblem auf ein erweitertes Rucksackproblem zurückgeführt werden konnte. Dieses ließ sich dann mittels dynamischer Programmierung effizient lösen. Die resultierenden Ablaufpläne sind hinsichtlich der Anforderungen der Anwender eines Echtzeit-Data-Warehouse-Systems optimal. In einer Reihe von Experimenten konnte weiterhin gezeigt werden, dass die multikriterielle Ablaufplanung auch für das Online-Scheduling anwendbar ist. Die beiden ersten Ziele – die Maximierung der Datenaktualität und die Minimierung der Anfragelatenz – wurden somit im zunächst einstufigen Fall gelöst.

Ablaufplanung in mehrstufigen Datenproduktionsprozessen Die mehrstufige Ablaufplanung wurde in Kapitel 5 betrachtet. In einer Reihe von Experimenten wurde untersucht, unter welchen Bedingungen eine Ablaufplanung in komplexen Datenproduktionsprozessen anwendbar ist. Dazu wurden die lokale und die globale Ablaufplanung als Extremfälle gegenübergestellt und deren Qualität unter variierenden Bedingungen miteinander verglichen. Das Ergebnis war, dass nur unter relativ extremen Annahmen, wie zum Beispiel sehr kurzen Datenproduktionsprozessen, dis-

junkten Aktualisierungsmengen und ungleich verteilten Service-Leveln, die globale Ablaufplanung einen signifikanten Mehrwert erzeugen konnte. Diese Annahmen können jedoch zum Teil in konkreten Szenarien erfüllt sein, wie die Fallstudie in Abschnitt 2.1 gezeigt hat. Die Art der anzuwendenden Ablaufplanung für mehrstufige Datenproduktionsprozesse ist daher fallbasiert zu entscheiden. Die erarbeiteten Empfehlungen können dabei als Leitfaden verwendet werden.

Analyse von Datenproduktionsprozessen Zur Analyse komplexer Datenproduktionsprozesse wurde ein Visualisierungskonzept erarbeitet, mit dem die Entwicklung der Datenqualität über den Prozessverlauf hinweg dargestellt und untersucht werden kann. Dafür wurden die prozessierten Daten durch Anwendung eines Partitionierungsschemas in eine einheitliche Datenstruktur überführt. Diese kann in Kombination mit den partitionierenden Dimensionen in eine Baumstruktur transformiert werden. Aufgrund der Menge der darzustellenden Daten wurde die Tree-Map als Darstellungsform gewählt, die besonders kompakt und skalierbar ist. Zur Exploration der Daten wurde die Tree-Map um Standard-OLAP-Operationen erweitert, so dass dem Anwender ein Werkzeug zur Verfügung steht, mit dem explorativ Datenproduktionsprozesse auf ihr Optimierungspotenzial hin untersucht werden können.

Datenstabilität für die Berichtsproduktion Die operativen Datenänderungen in Echtzeit-Data-Warehouse-Systemen erhöhen zwar die Datenaktualität, wie sie zur Unterstützung operativer Prozesse notwendig ist, führen aber in der Berichtsproduktion sehr schnell zu Inkonsistenzen. Zur Lösung des Konflikts zwischen Datenaktualität und -stabilität wurde in Kapitel 6 eine zusätzliche Datenschicht, die Reporting-Ebene, eingeführt, die das Echtzeit-Data-Warehouse-System von der Berichtsproduktion entkoppelt. Datenänderungen in dieser Schicht sind nur zu definierten Publikationszeitpunkten möglich, wodurch die Datenstabilität zwischen den Publikationszeitpunkten garantiert ist. Dies ermöglicht auch den Einsatz vorberechneter Aggregationen zur Beschleunigung der Standardberichtsproduktion. Die Granularität wurde, anders als bei der Verwendung materialisierter Sichten, vereinheitlicht, wodurch ein hoher Grad an Wiederverwendbarkeit erreicht werden konnte. Sowohl Berichts- als auch Ad-hoc-Anfragen profitierten von den vorberechneten Aggregaten. Die Überprüfung auf Vollständigkeit bei der Anfrageverarbeitung wird durch die Protokollierung der publizierten Daten sowie durch spezielle Anfragetechniken garantiert.

Komprimierung von Nulltupeln Die korrekte Vollständigkeitsbestimmung der Reporting-Ebene macht die Speicherung von Nulltupeln notwendig. Aufgrund der Dünnbesetztheit der multidimensionalen Daten beträgt der Anteil der Nullwerte am Gesamtdatensatz jedoch 95% und mehr. Zur Minimierung des Speicheraufwands wurde daher ein Komprimierungsverfahren entwickelt, das im ersten Schritt sogenannte Abhängigkeitsregeln identifiziert und diese im zweiten Schritt zur Komprimierung der Daten verwendet. Zur Laufzeit der Anfrage kann durch Anwendung partieller

Dekomprimierungstechniken der Datensatz rekonstruiert werden. Dadurch können die Daten der Reporting-Ebene auf einen Bruchteil ihrer Ursprungsgröße reduziert werden, ohne dass die Anfrageverarbeitung bzw. die Vollständigkeitsbestimmung beeinträchtigt wird.
Durch den Einsatz der Reporting-Ebene in Verbindung mit der Nullwertkomprimierung kann der Erhalt der Datenstabilität trotz der operativen Änderungen in Echtzeit-Data-Warehouse-Systemen gewährleistet werden.

Ausblick

In der Praxis existiert bisher nur eine sehr begrenzte Anzahl von Data-Warehouse-Architekturen, die Daten in Echtzeit integrieren. Jedoch sind zahlreiche Anwendungen bekannt, in denen durch die Einbindung des Data-Warehouses in operative Prozesse ein Mehrwert generiert werden kann. Laut einer Gartner-Studie [30] werden bis 2010 30% aller BI-Funktionalitäten Daten mit einer Aktualität von 15 Minuten und höher benötigen. Die Ursache für die Zurückhaltung bei der Entwicklung ist vielfältig. Zum einen sind Data-Warehouse-Systeme extrem komplex, wodurch Änderungen an vielen Komponenten, wie den Quellsystemen, den ETL-Systemen, den Analysewerkzeugen, der Datenrepräsentation usw., notwendig werden, um von der Integration hochaktueller Daten zu profitieren. Unter Berücksichtigung der hohen Entwicklungskosten im Data-Warehouse-Bereich ist die Hürde damit hoch angelegt. Zum anderen existieren Befürchtungen, dass durch notwendige Änderungen und Weiterentwicklungen die Datenqualität und -integrität leidet. Hier ist es die Aufgabe der Unternehmen, die nächsten Schritte zu gehen und ihre Data-Warehouse-Systeme entsprechend dem aktuellen Forschungsstand weiterzuentwickeln. Diese Dissertation bildet hierfür einen guten Ausgangspunkt.

Literaturverzeichnis

[1] Condensed Cube: An Effective Approach to Reducing Data Cube Size. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 155, Washington, DC, USA, 2002. IEEE Computer Society.

[2] UBS Geschäftsbericht, 2009.

[3] D. Abadi, S. Madden, and M. Ferreira. Integrating compression and execution in column-oriented database systems. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 671–682, New York, NY, USA, 2006. ACM.

[4] F. Afrati and R. Chirkova. Selecting and using views to compute aggregate queries. In *In Proc. ICDT*. Morgan Kaufmann, 2005.

[5] F. N. Afrati, R. Chirkova, S. Gupta, and C. Loftis. Designing and using views to improve performance of aggregate queries (extended abstract). In *DASFAA*, pages 548–554, 2005.

[6] M. Altinel, C. Bornhövd, S. Krishnamurthy, C. Mohan, H. Pirahesh, and B. Reinwald. Cache tables: paving the way for an adaptive database cache. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 718–729. VLDB Endowment, 2003.

[7] S. Babu, M. Garofalakis, and R. Rastogi. Spartan: a model-based semantic compression system for massive data tables. volume 30, pages 283–294, New York, NY, USA, 2001. ACM.

[8] A. H. Bauer and H. H. Günzel. *Data-Warehouse-Systeme*. dpunkt, Heidelberg, 2009.

[9] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cube. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 359–370, New York, NY, USA, 1999. ACM.

[10] J. Branke, E. Salihoğlu, and Şima Uyar. Towards an analysis of dynamic environments. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1433–1440, New York, NY, USA, 2005. ACM.

[11] R. Braumandl, A. Kemper, and D. Kossmann. Quality of service in an information economy. *ACM Trans. Interet Technol.*, 3(4):291–333, 2003.

[12] S. Brobst and J. Rarey. Five stages of data warehouse decision support evolution. DSSResources.COM, Juni 2003.

[13] S. Bu, L. V. S. Lakshmanan, and R. T. Ng. Mdl summarization with holes. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 433–444. VLDB Endowment, 2005.

[14] S. Chaudhuri, R. Kaushik, A. Pol, and R. Ramamurthy. Stop-and-restart style execution for long running decision support queries. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 735–745. VLDB Endowment, 2007.

[15] Z. Chen, Y. Zhang, Y. Zhou, H. Scott, and B. Schiefer. Empirical evaluation of multi-level buffer cache collaboration for storage systems. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 145–156, New York, NY, USA, 2005. ACM.

[16] I. Das and J. E. Dennis. Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[17] D. L. Davison and G. Graefe. Dynamic resource brokering for multi-user query execution. *SIGMOD Rec.*, 24(2):281–292, 1995.

[18] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1 edition, Juni 2001.

[19] B. G. C. Dellaert and B. E. Kahn. How tolerable is delay? consumers evaluations of internet web sites after waiting. working paper. *Journal of Interactive Marketing*, 13:41–54, 1999.

[20] P. Deshpande and J. F. Naughton. Aggregate aware caching for multi-dimensional queries. In *EDBT '00: Proceedings of the 7th International Conference on Extending Database Technology*, pages 167–182, London, UK, 2000. Springer-Verlag.

[21] P. Deshpande, K. Ramasamy, A. Shukla, and J. F. Naughton. Caching multidimensional queries using chunks. In *In Laura M. Haas and Ashutosh Tiwary, editors, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA*, pages 259–270. ACM Press, 1998.

[22] W. W. Eckerson and R. P. Sherman. Strategies for managing spreadmarts. *TDWI Research Report*, 13, 2008.

[23] S. J. Eggers, F. Olken, and A. Shoshani. A compression technique for large statistical data-bases. In *VLDB '1981: Proceedings of the seventh international conference on Very Large Data Bases*, pages 424–434. VLDB Endowment, 1981.

[24] D. Feinberg and M. A. Beyer. Magic quadrant for data warehouse database management systems. Gartner RAS Core Research Note G00163473, Dezember 2008.

[25] M. Fiedler, J. Albrecht, T. Ruf, J. Görlich, and M. Lem. Pre-caching hochdimensionaler aggregate mit relationaler technologie. In *Datenbanksysteme in Business, Technologie und Web (BTW 2009), 13. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme*, pages 498–507, 2009.

[26] R. Finger. Bi-betriebsmodelle auf dem prüfstand. *BI-Spektrum*, pages 21–25, 2006.

[27] C. Francalanci and B. Pernici. Data quality assessment from the user's perspective. In *IQIS '04: Proceedings of the 2004 international workshop on Information quality in information systems*, pages 68–73, New York, NY, USA, 2004. ACM.

[28] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, 2005.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[30] B. Gassman, K. Schlegel, and M. A. Beyer. Survey shows bi users want fresher data. Gartner Research, September 2006.

[31] B. S. Gill. On multi-level exclusive caching: offline optimality and why promotions are better than demotions. In *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–17, Berkeley, CA, USA, 2008. USENIX Association.

[32] C.-L. Goh, K. Aisaka, M. Tsukamoto, and S. Nishio. Database compression with data mining methods. pages 177–190, 2000.

[33] D. Gritschneder, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded pareto front of practical multiobjective optimization problems. *SIAM Journal on Optimization*, 20:915–934, 2009.

[34] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[35] T. Härder and A. Bühmann. Value complete, column complete, predicate complete. *The VLDB Journal*, 17(4):805–826, 2008.

[36] D. A. Huffmann. A Method for the Construction of Minimum-Redundancy Codes. *Proc. Inst. Radio Eng. 40*, pages 1098–1101, 1952.

[37] E. J. Hughes. Evolutionary many-objective optimisation: many once or one many? In *Congress on Evolutionary Computation*, pages 222–227, 2005.

[38] W. H. Inmon. *Building the Operational Data Store*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[39] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, Inc., New York, NY, USA, 4 edition, 2005.

[40] H. V. Jagadish, R. T. Ng, B. C. Ooi, and A. K. H. Tung. Itcompress: An iterative semantic compression algorithm. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 646, Washington, DC, USA, 2004. IEEE Computer Society.

[41] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses: An extended repository approach. *Inf. Syst.*, 24(3):229–253, 1999.

[42] T. Jörg and S. Deßloch. Formalizing etl jobs for incremental loading of data warehouses. In *Datenbanksysteme in Business, Technologie und Web (BTW 2009), 13. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme*, pages 327–346, 2009.

[43] H. Karloff and M. Mihail. On the complexity of the view-selection problem. In *In PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 167–173, New York, NY, USA, 1999. ACM Press.

[44] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations. New York: Plenum*, 1972.

[45] S. Krompass, H. Kuno, U. Dayal, and A. Kemper. Dynamic workload management for very large data warehouses: juggling feathers and bowling balls. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1105–1115. VLDB Endowment, 2007.

[46] S. Krompass, H. Kuno, J. L. Wiener, K. Wilkinson, U. Dayal, and A. Kemper. Managing long-running queries. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 132–143, New York, NY, USA, 2009. ACM.

[47] L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient Cube: How to Summarize the Semantics of a Data Cube. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 778–789. VLDB Endowment, 2002.

[48] L. V. S. Lakshmanan, J. Pei, and Y. Zhao. Qc-trees: an efficient summary structure for semantic olap. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 64–75, New York, NY, USA, 2003. ACM.

[49] W. Lehner. *Datenbanktechnologie für Data-Warehouse-Systeme : Konzepte und Methoden*. dpunkt-Verlag, 1. Auflage edition, 2003.

[50] H.-J. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *SSDBM '97: Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*, pages 132–143, Washington, DC, USA, 1997. IEEE Computer Society.

[51] A. Y. Levy, A. O. Mendelzon, and Y. Sagiv. Answering queries using views (extended abstract). In *In PODS '95: Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 95–104, New York, NY, USA, 1995. ACM Press.

[52] X. Li, A. Aboulnaga, K. Salem, A. Sachedina, and S. Gao. Second-tier cache management using write hints. In *FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 9–9, Berkeley, CA, USA, 2005. USENIX Association.

[53] A. Mehta, C. Gupta, S. Wang, and U. Dayal. rfeed: A mixed workload scheduler for enterprise data warehouses. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1455–1458, Washington, DC, USA, 2009. IEEE Computer Society.

[54] A. Messac and C. A. Mattso. Normal constraint method with guarantee of even representation of complete pareto frontier. *AIAA Journal*, 42:2101–2111, 2004.

[55] A. Motro and I. Rakov. Estimating the quality of data in relational databases. In *In Proceedings of the 1996 Conference on Information Quality*, pages 94–106. MIT, 1996.

[56] F. Naumann. *Quality-driven query answering for integrated information systems*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[57] R. M. Nauss. The 0-1 knapsack problem with multiple choice constraints. *European Journal of Operational Research*, 2(2):125–131, March 1978.

[58] C. Pettey and H. Stevens. Gartner reveals five business intelligence predictions for 2009 and beyond, Januar 2009.

[59] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N.-E. Frantzell. Meshing streaming updates with persistent data in an active data warehouse. *IEEE Trans. Knowl. Data Eng.*, 20(7):976–991, 2008.

[60] P. Ponniah. *Data Warehousing Fundamentals*. John Wiley & Sons, Inc., New York, NY, USA, 2001. Foreword By-Reddy, Pratap P.

[61] K. Ramamritham, S. H. Son, and L. C. Dipippo. Real-time databases and data services. *Real-Time Syst.*, 28(2-3):179–215, 2004.

[62] G. Redwitz. Gfk handbuch der panelforschung - retail and technology. GfK Marketing Services GmbH & Co. KG, 2005.

[63] A. Schanzenberger. System design for periodic data production management, December 2006.

[64] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678690, 1968.

[65] L. E. Schrage and L. W. Miller. The queue m-g-1 with the shortest remaining processing time discipline. *Operations Research*, 14:670684, 1966.

[66] B. Schroeder, M. Harchol-Balter, A. Iyengar, and E. Nahum. Achieving class-based qos for transactional workloads. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 153, Washington, DC, USA, 2006. IEEE Computer Society.

[67] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.

[68] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal. Qox-driven etl design: reducing the cost of etl consulting engagements. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 953–960, New York, NY, USA, 2009. ACM.

[69] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis. Dwarf: shrinking the petacube. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 464–475, New York, NY, USA, 2002. ACM.

[70] Y. Sismanis and N. Roussopoulos. The dwarf data cube eliminates the high dimensionality curse. Technical Report TR-CS4552, University of Maryland, 2003.

[71] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[72] D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 318–329, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[73] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1):048–063, 1996.

[74] M. Stonebraker, C. Bear, U. Çetintemel, M. Cherniack, T. Ge, N. Hachem, S. Harizopoulos, J. Lifter, J. Rogers, and S. B. Zdonik. One size fits all? part

2: Benchmarking studies. In *Third Biennal Conference on Innovative Data Systems Research (CIDR 2007)*, pages 173–184, 2007.

[75] M. Stonebraker and U. Cetintemel. One size fits all: An idea whose time has come and gone. pages 2–11, 2005.

[76] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of an architectural era: (it's time for a complete rewrite). In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1150–1160. VLDB Endowment, 2007.

[77] K. Strange and B. Hostmann. Bi competency center is core to bi success. *Gartner Research*, 2003.

[78] M. Thiele, U. Fischer, and W. Lehner. Partition-based workload scheduling in living data warehouse environments. In *DOLAP '07: Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 57–64, New York, NY, USA, 2007. ACM.

[79] M. Thiele, U. Fischer, and W. Lehner. Partition-based workload scheduling in living data warehouse environments. *Information Systems*, 34:1–5, 2009.

[80] C. Thomsen, T. B. Pedersen, and W. Lehner. Rite: Providing on-demand data for right-time data warehousing. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 456–465, Washington, DC, USA, 2008. IEEE Computer Society.

[81] V. T'Kindt and J.-C. Billaut. *Multicriteria Scheduling - Theory, Models and Algorithms*. Springer Verlag, 2006.

[82] P. Toth. Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, 25:29–45, 1980.

[83] V. Tziovara, P. Vassiliadis, and A. Simitsis. Deciding the physical implementation of etl workflows. In *DOLAP '07: Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 49–56, New York, NY, USA, 2007. ACM.

[84] P. Vassiliadis, M. Bouzeghoub, and C. Quix. Towards quality-oriented data warehouse usage and evolution. In *CAiSE*, pages 164–179. Springer, 1999.

[85] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: flexible proportional-share resource management. In *OSDI '94: Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, page 1, Berkeley, CA, USA, 1994. USENIX Association.

[86] R. Y. Wang and D. M. Strong. Beyond accuracy: what data quality means to data consumers. *J. Manage. Inf. Syst.*, 12(4):5–33, 1996.

[87] G. Weikum. Towards guaranteed quality and dependability of information systems. In *Datenbanksysteme in Business, Technologie und Web (BTW 1999), 8. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme*, pages 379–409. Springer Verlag, 1999.

[88] C. White. Data integration: Using etl, eai, and eii tools to create an integrated enterprise. *A 101communications Publication*, 2005.

[89] R. Winter. Scaling the data warehouse, Oktober 2008.

[90] R. Winter. Why are data warehouses growing so fast? BeyeNetwork, April 2008.

[91] T. M. Wong and J. Wilkes. My cache or yours? making storage more exclusive. In *ATEC '02: Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference*, pages 161–175, Berkeley, CA, USA, 2002. USENIX Association.

[92] N. Ye, X. Li, T. Farley, and X. Xu. Job scheduling methods for reducing waiting time variance. *Comput. Oper. Res.*, 34(10):3069–3083, 2007.

[93] Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 159–170, New York, NY, USA, 1997. ACM.

[94] M. Zhou and L. Zhou. How does waiting duration information influence customers' reactions to waiting for services. *J. of Applied Social Psychology*, 26:1702–1717, 1996.

[95] Y. Zhou, Z. Chen, and K. Li. Second-level buffer cache management. *IEEE Trans. Parallel Distrib. Syst.*, 15(6):505–519, 2004.

[96] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

Online-Quellenverzeichnis

[97] Norwich Union:
http://www.aviva.co.uk/media-centre/story/2840/norwich-union-launches-innovative-pay-as-you-drive/.
Zuletzt besucht am: 25.12.2009

[98] CELL Project (IBM Research):
http://www.research.ibm.com/cell/.
Zuletzt besucht am: 19.12.2009

[99] Cognos TM1 (IBM):
http://www-142.ibm.com/software/products/de/de/cognostm1.
Zuletzt besucht am: 19.12.2009

[100] Derby (Apache):
http://db.apache.org/derby/.
Zuletzt besucht am: 19.12.2009

[101] E-Government (Deutsche Bundesregierung):
http://www.cio.bund.de/DE/E-Government/e-government_node.html.
Zuletzt besucht am: 19.12.2009

[102] BIAF (Fraport AG):
http://www.isreport.de/business-intelligence/near-real-time-reporting-fuer-den-betrieb-des-flughafen-frankfurt.html.
Zuletzt besucht am: 25.12.2009

[103] Greenplum:
http://www.greenplum.com/.
Zuletzt besucht am: 19.12.2009

[104] i2010 (Europäischen Union):
http://europa.eu/legislation_summaries/employment_and_social_policy/job_creation_measures/c11328_de.htm.
Zuletzt besucht am: 19.12.2009

[105] iCharts:
http://www.icharts.net.
Zuletzt besucht am: 19.12.2009

[106] Infonea Cube (Comma Soft):
http://www.comma-soft.com/.
Zuletzt besucht am: 19.12.2009

[107] InfoZoom (humanIT):
`http://www.infozoom.com`.
Zuletzt besucht am: 19.12.2009

[108] JTreeMap (ObjectLab):
`http://jtreemap.sourceforge.net/`.
Zuletzt besucht am: 19.12.2009

[109] ManyEyes (IBM):
`http://manyeyes.alphaworks.ibm.com/manyeyes/`.
Zuletzt besucht am: 19.12.2009

[110] Mondrian (Pentaho):
`http://mondrian.pentaho.org/`.
Zuletzt besucht am: 19.12.2009

[111] SQL Server (Microsoft):
`http://www.microsoft.com/germany/sql/2008/default.mspx`.
Zuletzt besucht am: 19.12.2009

[112] Mycrocosm:
`http://mycro.media.mit.edu/`.
Zuletzt besucht am: 19.12.2009

[113] Netazza:
`http://www.netezza.com/`.
Zuletzt besucht am: 19.12.2009

[114] OpenCalais (Thomsen Reuters):
`http://www.opencalais.com`.
Zuletzt besucht am: 19.12.2009

[115] Open Mashup Alliance:
`http://www.openmashup.org/`.
Zuletzt besucht am: 21.12.2009

[116] OpenSocial (Google):
`http://code.google.com/intl/de-DE/apis/opensocial/`.
Zuletzt besucht am: 19.12.2009

[117] Oracle In-Memory Database Cache:
`http://www.oracle.com/database/in-memory-database-cache.html`.
Zuletzt besucht am: 19.12.2009

[118] PANOsight (Panoratio):
`http://www.panoratio.com`.
Zuletzt besucht am: 19.12.2009

[119] PivotLink:
`http://www.pivotlink.com/`.
Zuletzt besucht am: 19.12.2009

[120] Powerpivot (Microsoft):
http://www.powerpivot.com/.
Zuletzt besucht am: 19.12.2009

[121] QlikView (QlikTech):
http://www.qlikview.com/.
Zuletzt besucht am: 19.12.2009

[122] RAC (Oracle):
www.oracle.com/lang/de/database/rac_home.html.
Zuletzt besucht am: 19.12.2009

[123] SAP BI Accelerator:
http://www.sap.com/germany/plattform/netweaver/fsc.epx.
Zuletzt besucht am: 19.12.2009

[124] SolidDB (IBM):
http://www-01.ibm.com/software/data/soliddb/universal-cache/.
Zuletzt besucht am: 19.12.2009

[125] Swivel:
http://www.swivel.com.
Zuletzt besucht am: 19.12.2009

[126] Tableau Desktop (Tableau Software):
http://www.tableausoftware.com/products/desktop.
Zuletzt besucht am: 19.12.2009

[127] Teradata:
http://www.teradata.com.
Zuletzt besucht am: 19.12.2009

[128] Track-n-Graph:
http://www.trackngraph.com.
Zuletzt besucht am: 19.12.2009

[129] WolframAlpha:
http://www.wolframalpha.com/.
Zuletzt besucht am: 19.12.2009

[130] Xtreme Data Appliance (XtremeData, Inc.):
http://www.xtremedata.com/.
Zuletzt besucht am: 19.12.2009

Abbildungsverzeichnis

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Maik Thiele
Dresden, 23. März, 2010