

Diplomarbeit

Konzeption und Realisierung eines Warehouse-Konzepts für das Content Management

Stefan Eli

11. Juni 2003

1. Gutachter: Prof. Dr. Jürgen Nehmer, Universität Kaiserslautern
 2. Gutachter: Prof. Dr. Joachim W. Schmidt, TU Hamburg-Harburg
- Betreuer: Dipl.-Inform. Rainer Müller, TU Hamburg-Harburg

FB Informatik • AG Systemsoftware • Prof. Dr. Jürgen Nehmer
Universität Kaiserslautern • Postfach 3049 • 67653 Kaiserslautern

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe, und dass keine anderen als die angegebenen Quellen und Hilfsmittel von mir verwendet wurden.

Hamburg, 11. Juni 2003

.....

Inhaltsverzeichnis

1	Einleitung	1
1.1	Ziel der Arbeit	1
1.2	Gliederung	1
2	Data Warehouse-Konzepte	3
2.1	Was ist Data Warehousing?.....	3
2.2	Datenbankstrukturen für ein Data Warehouse.....	5
2.2.1	Relationale, multidimensionale und objektorientierte DBMS	5
2.2.2	Das <i>Sternschema</i>	8
2.3	Datenimport	10
2.4	Anwendungen	11
2.4.1	Abfrage- und Berichts-Werkzeuge	12
2.4.2	Multidimensionale Analyse	13
2.4.3	Statistische Analyse	15
2.4.4	Data Mining	16
3	Konzeption eines Warehouse-Modells für das Content Management	18
3.1	Content Management-Systeme	18
3.2	Selektionskriterien für die Datenauswahl	22
3.3	Transformation der Daten	24
4	Content Management am Beispiel des INFOASSET BROKERS	26
4.1	Basisfunktionalität	26
4.2	Systemarchitektur	27
4.2.1	Die Konzepte des INFOASSET BROKER	30
4.2.2	Die Kernfunktionen des INFOASSET BROKER	30
4.2.3	Die INFOASSET BROKER Erweiterungen	31
4.3	Die Alerting Extension	32
5	Realisierung des Content Warehouse-Modells	35
5.1	Die <i>Tracking Extension</i>	36
5.1.1	Aufzeichnen von Änderungen in der Inhalts-Komponente	36
5.1.2	Aufzeichnen von Änderungen in der Layout-Komponente	37
5.1.3	Aufzeichnung von Suchanfragen.....	38
5.1.4	Schnittstelle zwischen Broker und <i>Tracking Server</i>	39
5.2	Realisierung des <i>Tracking Servers</i>	40
5.2.1	Pufferung und Verarbeitung der Daten	41
5.2.2	Die Datenbank Schnittstelle	43
5.3	Ergebnisse.....	44
6	Zusammenfassung und Ausblick	47
6.1	Zusammenfassung	47
6.2	Ausblick.....	48
	Literaturverzeichnis	49

Abbildungsverzeichnis

Abbildung 2.1: Bestandteile eines Data Warehouse	4
Abbildung 2.2: Dreidimensionaler <i>Hypercube</i> [Vitt02]	6
Abbildung 2.3: Ausschnitt aus einer Datenbankstruktur für <i>OLTP</i> [Fuhr02]	7
Abbildung 2.4: Versicherungs-Policen als <i>Sternschema</i> [Fuhr02]	9
Abbildung 2.5: Veranschaulichung von Slicing und Dicing	14
Abbildung 2.6: Drill Down und Roll Up	15
Abbildung 2.7: Drill Across	16
Abbildung 3.1: Content Management-Funktionen zweiter Ordnung	19
Abbildung 3.2: Datenfluss vom CMS zum Data Warehouse	24
Abbildung 3.3: Erhobene Daten in Form eines <i>Sternschemas</i>	25
Abbildung 4.1: Screenshot der INFOASSET BROKER Benutzeroberfläche	27
Abbildung 4.2 : Die Schichtenarchitektur des INFOASSET BROKER	28
Abbildung 4.3: Typischer Ablauf beim Erzeugen eines <i>Goals</i>	33
Abbildung 4.4: Sequenzdiagramm für die An- und Abmeldung eines <i>Goals</i>	33
Abbildung 5.1: Datenfluss vom Broker zum Data Warehouse	35
Abbildung 5.2: Erweiterungen des Brokers durch die Tracking Extension	36
Abbildung 5.3: Das Interface <i>EventService</i>	37
Abbildung 5.4: Erhobene Daten bei Änderung eines <i>Assets</i>	37
Abbildung 5.5: Benutzte Daten beim Anfordern einer Vorlage	38
Abbildung 5.6: Daten der Suchanfrage	38
Abbildung 5.7: Kapselung der RMI-Funktionalität in <i>BasicTracking</i>	39
Abbildung 5.8: Klassendiagramm des <i>Tracking Servers</i>	41
Abbildung 5.9: Sequenzdiagramm eines Aufrufs von <i>sendDataToServer</i>	41
Abbildung 5.10: Zusammenfassen von Ereignissen	42
Abbildung 5.11: Sequenzdiagramm des nebenläufigen Prozesses	43
Abbildung 5.12: Zusammenfassen von zwei Ereignissen	44
Abbildung 5.13: Datenbankschema in Form eines <i>Sternschemas</i>	44
Abbildung 5.14: Summe der aufgesuchten Seiten im Juni 2003	45
Abbildung 5.15: Besuchte Seiten im Juni abhängig vom Tag	45
Abbildung 5.16: Besuchte Seiten im Juni, aufgeschlüsselt nach Benutzer	46
Abbildung 5.17: Besuchte Vorlagen-Typen für einen Benutzer	46

1 Einleitung

Die zunehmende Menge digitaler Daten in vielen Unternehmen stellt hohe Anforderungen an Computer- und Speichersysteme. Um diese Daten zu verwalten sind leistungsfähige Systeme nötig. Mit der Absicht, diese großen Datenmengen zusammen mit den leistungsfähigen Computersystemen für strategische Unternehmensentscheidungen zu nutzen, entstanden Anfang der 90er Jahre die Data Warehouses. Die Aufgabe eines Data Warehouse ist die Vereinfachung von Entscheidungsprozessen in Unternehmen. Dazu wird eine zentrale, unternehmensweite Informationsbasis angestrebt, aus der man mit Hilfe von geeigneten Werkzeugen eigenständig zeitnahe, problemspezifische Informationen gewinnen kann. Üblicherweise setzen sich die Daten in einem Unternehmen aus Finanz-, Umsatz- und Produktionsdaten zusammen. Es stellt sich die Frage, ob man die Infrastruktur eines Data Warehouse nicht auch für eine gänzlich andere Datenbasis nutzen könnte. Ebenfalls sehr große Mengen an Daten werden von Content Management-Systemen verwaltet. Die Form der Inhalte kann sehr unterschiedlicher Natur sein, wie z.B. Texte (sowohl strukturiert als auch unstrukturiert), Bilder aller Art etc. Eine nähere Untersuchung von Content Management-Systemen führt zu dem Ergebnis, dass sich die eigentlichen Inhalte eines Content Management-Systems weniger für ein Warehouse-Modell eignen. Vielmehr ist es von Interesse, auf welche Inhalte Benutzer zugreifen, und welche Inhalte sie dabei erzeugen, ändern oder löschen. Diese Informationen, entsprechend aufbereitet und in ein für Data Warehouses nutzbares Datenbankschema konvertiert, lassen viele interessante Möglichkeiten zur Analyse zu.

1.1 Ziel der Arbeit

Ziel dieser Arbeit ist die Konzeption und Realisierung eines Modells, das sowohl die umfangreiche Datenbasis eines Content Management-Systems, als auch deren Nutzungscharakteristik durch den Anwender einem Data Warehouse zugänglich macht. Zu diesem Zweck wird eine generische Schnittstelle zwischen dem verwendeten INFOASSET BROKER und dem Data Warehouse entworfen. Mit Hilfe dieser Schnittstelle werden Daten über das Benutzerverhalten und die Zugriffe auf Inhaltsobjekte des INFOASSET BROKERS gewonnen, die anschließend zusammengefasst und in einer relationalen Datenbank in einem passenden Data Warehouse-Datenbankschema abgelegt werden.

1.2 Gliederung

Kapitel 2 führt den Leser in die Grundlagen von Data Warehousing ein und gibt einen Überblick über verwendete Datenbankstrukturen und mögliche Anwendungen. In Kapitel 3 werden die grundlegenden Anforderungen und Komponenten eines Content Management-Systems vorgestellt und auf deren Tauglichkeit für ein Data Warehouse-Modell geprüft. Die Kernkomponenten und die Struktur des INFOASSET

BROKER werden in Kapitel 4 vorgestellt. Das entworfene Warehouse-Modell wird anschließend in Kapitel 5 in den INFOASSET BROKER integriert und die prototypische Realisierung erläutert. Kapitel 6 schließt mit einem Überblick über mögliche Erweiterungen des Content Warehouse-Ansatzes.

2 Data Warehouse-Konzepte

Das folgende Kapitel gibt dem Leser einen Überblick über die Thematik des Data Warehousing. Die Unterschiede zwischen einem Data Warehouse und operativen Systemen sowie die Grundlagen zu den beim Data Warehousing verwendeten Datenbankstrukturen werden in Kapitel 2.2 vorgestellt. Die weiteren Abschnitte präsentieren Werkzeuge zur Datenmigration und die möglichen Einsatzbereiche für Data Warehouse-Anwendungen.

2.1 Was ist Data Warehousing?

Der Begriff Data Warehouse wurde durch W.H. Inmon geprägt, der mit seinem 1993 erstmals erschienenen Buch „Building the Data Warehouse“ den Grundstein für diesen Bereich der Informationssysteme legte. [Inmo96]

Ein Data Warehouse stellt ein Informationssystem dar, das verschiedenste Analysemöglichkeiten für die in einem Unternehmen anfallenden Daten bieten soll. Die dabei gewonnenen Informationen dienen in der Regel der Entscheidungsunterstützung, womit ein Data Warehouse auch als Entscheidungsunterstützungssystem dient.

Daraus ergibt sich eine klare Abgrenzung zu den operativen Systemen, die meist als sogenannte *Online Transaction Processing-Systeme (OLTP-Systeme)* realisiert sind. Diese Systeme haben die Aufgabe, die im Geschäftsbetrieb anfallenden Daten zu speichern und benötigte Daten zur Verfügung zu stellen. Die Notwendigkeit, dass konkurrierende Schreib- und Lesezugriffe auf die Datenbank in eine geeignete Abfolge gebracht werden müssen, stellt damit ein zentrales Hauptproblem dieser Anwendungen dar. Gelöst wird dieses Problem durch die so genannten Transaktionen, die jeweils einen Block von Datenbankanweisungen zu einer atomaren Einheit zusammenfassen, die dann entweder komplett erfolgreich ausgeführt oder in einem Fehlerfall komplett verworfen wird.

Ein zweites Problem der operativen Systeme ist die Sicherung der Konsistenz der Daten. Es muss sichergestellt werden, dass gleiche Daten im System auch gleich bleiben, d.h. wenn ein Datum geändert wird, das an anderer Stelle ein weiteres Mal gespeichert wurde, muss es auch an dieser anderen Stelle geändert werden. Die gängige Lösung dieses Problems ist jedes Datum möglichst nur einmal zu speichern. Bei den relationalen Datenbanken hat dies zu den so genannten Normalformen geführt.

Beim Data Warehouse ist es so, dass schreibende Zugriffe auf das Data Warehouse idealerweise nur zu festgelegten Zeitpunkten erfolgen, an denen die Aktualisierung der Daten erfolgt, z.B. am Monatsende, am Wochenende oder über Nacht. Das Problem des konkurrierenden Zugriffs stellt sich hier in der Regel nicht, da die Aktualisierung der Daten zu einer Zeit stattfindet, in der die Anwender nicht arbeiten und damit auch keine Abfragen ausführen. In extremen Fällen sind die Aktualisierungsintervalle heute aber auch schon so kurz (z.B. stündlich), dass sich die Problematik des konkurrierenden Zugriffs auch für das Data Warehouse ergibt.

Die Speicherung der Daten in einer normalisierten Form kommt für ein Data Warehouse in der Regel nicht in Frage, da diese Form in Zusammenhang mit den typischerweise komplexen Abfragen und teilweise sehr großen Datenmengen zu erheblichen Geschwindigkeitseinbußen führt. Daher wurden für Data Warehouses spezielle Datenbankkonzepte entwickelt, die im Kapitel 2.2 näher erläutert werden. Da das Data Warehouse in der beschriebenen Form ein eigenständiges System darstellt, müssen die benötigten Daten aus den operativen Systemen und ggf. weiteren externen Systemen, die z.B. zu Tochterfirmen gehören können, in das Data Warehouse übertragen werden. Während dieser Übertragung oder in nachfolgenden Schritten müssen diese Daten dann in eine für das Data Warehouse geeignete Form gebracht werden. Einen Überblick über die an einem Data Warehouse beteiligten Bestandteile gibt die Abbildung 2.1. Kapitel 2.3 geht näher auf diese Werkzeuge ein. Über entsprechende Abfragen greifen nun die Anwendungen auf die vorbereiteten

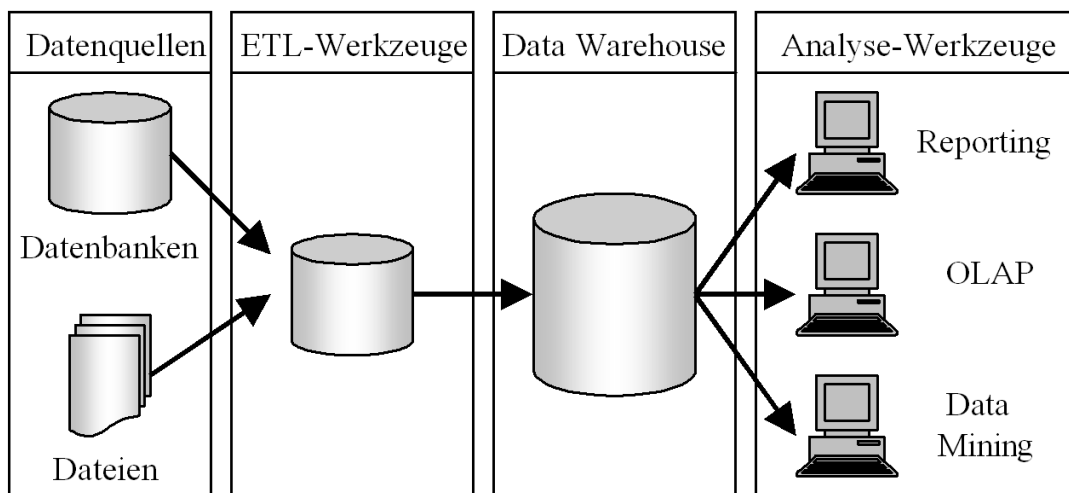


Abbildung 2.1: Bestandteile eines Data Warehouse

Daten zu, die es den Anwendern des Data Warehouse ermöglichen sollen, die gewünschten Analysen durchzuführen und neue Entscheidungen zu treffen. Aus den beschriebenen Aspekten eines Data Warehouse lassen sich Anforderungen ableiten. Ralph Kimball [Kim96] beschreibt sechs Anforderungen an ein Data Warehouse: Der Zugriff auf die Daten muss dabei, laut Kimball, von den Rechnern der Manager und Analysten aus möglich sein. Der Zugriff muss schnell erfolgen, d.h. kleinere Anfragen müssen in weniger als einer Sekunde ablaufen. Weiterhin müssen die Abfragewerkzeuge einfach zu bedienen sein, wobei idealerweise ein Mausklick genügen sollte, um einen Bericht zu erhalten.

Konsistenz meint, dass zwei Personen, die unabhängig voneinander denselben Bericht anfordern, auch dieselben Informationen geliefert bekommen, egal zu welchem Zeitpunkt sie dies tun. Außerdem meint Konsistenz, dass zu allen gespeicherten Daten Angaben abgefragt werden können, die genau beschreiben welche Daten dort jeweils abgelegt sind. Schließlich heißt Konsistenz auch, dass die Anwender eine Warnung erhalten, wenn sie auf Daten zugreifen wollen, die aufgrund eines Problems oder weil sie gerade erst eingefügt werden noch nicht vollständig sind.

Die Werkzeuge zur Anwendung des Data Warehouse machen etwa 40 Prozent des Data Warehouse aus. Sie sollen vor allem dem Grundsatz „Zeige mir was wichtig ist!“ folgen. Veröffentlichung der benutzten Daten meint dabei nicht nur eine Sammlung und Ablage der Daten, sondern eine sorgfältige und bereinigte Zusammenstellung aus den vielfältigen Quellen des Unternehmens, die damit auch eine gewisse Qualität gewährleisten.

2.2 Datenbankstrukturen für ein Data Warehouse

In den letzten Jahren wurde zum Teil heftig diskutiert, welche Datenbanktypen für ein Data Warehouse am besten geeignet seien. Dabei wurden insbesondere, die inzwischen sehr weit verbreiteten *relationalen Datenbankmanagementsysteme (DBMS)*, die speziell für die multidimensionale Analyse ausgelegten multidimensionalen *DBMS* und die noch relativ neuen objektorientierten *DBMS* betrachtet. Im folgenden Kapitel sollen die Vor- und Nachteile dieser Systeme dargestellt werden. Da bei den meisten heute im Einsatz oder in der Entwicklung befindlichen Data Warehouses ein relationales *DBMS* verwendet wird, wird in Kapitel 2.2.2 das dafür entwickelte *Sternschema* vorgestellt.

2.2.1 Relationale, multidimensionale und objektorientierte DBMS

Am weitesten verbreitet sind derzeit sicherlich *Relationale DBMS* [Voss94]. In nahezu jedem Unternehmen, vom großen Konzern bis hin zu kleinen Firmen mit nur wenigen Mitarbeitern, werden solche Systeme für die Datenverarbeitung eingesetzt. Da die zu verarbeitenden Daten sich bei einem derart weiten Anwenderspektrum in ihrer Art und ihrem Umfang teilweise erheblich unterscheiden, befinden sich eine Vielzahl von *relationalen DBMS* mit unterschiedlichem Leistungsumfang auf dem Markt. Allerdings eignen sich die sonst bei *relationalen Datenbanken* üblichen normalisierten Datenbankschemata nicht für die typischen Anfragen, die ein Data Warehouse verarbeiten muss. Das im folgenden Kapitel vorgestellte spezielle Datenbankschema für ein relationales Data Warehouse erlaubt auch auf dieser Basis sehr große und leistungsfähige Lösungen.

Bei den *Multidimensionalen DBMS* versucht man bereits in der Architektur des Systems einer typischen Anwendung im Data Warehouse, der *multidimensionalen Analyse*, gerecht zu werden. Diese häufig mit dem Schlagwort *Online Analytical Processing (OLAP)* bezeichnete Anwendung stellt sicherlich eine der wichtigsten Anwendungsarten im Data Warehouse-Bereich dar. Genau dies wird allerdings auch zu einem Hauptproblem dieser speziellen Datenbankart. Sie eignet sich kaum für andere Anwendungen und bleibt hinsichtlich des verkraftbaren Datenvolumens auch eher auf kleine Datenbanken beschränkt. Damit kommen sie allerdings durchaus als Ergänzung zu einem anderen *DBMS* in Frage. Multidimensionale Daten lassen sich am besten in einer mehrdimensionalen Matrix, auch häufig als Kreuztabelle bezeichnet, darstellen. Im Rahmen der technischen Realisierung werden eher die Begriffe *Array* oder *Hypercube* verwendet. Die darin abgelegten Daten werden als Fakten oder Faktdaten bezeichnet, wobei deren Semantik durch die Dimensionen

bestimmt wird. Jede Dimension stellt dabei ein notwendiges Charakteristikum dar, das die Matrixwerte beschreibt.

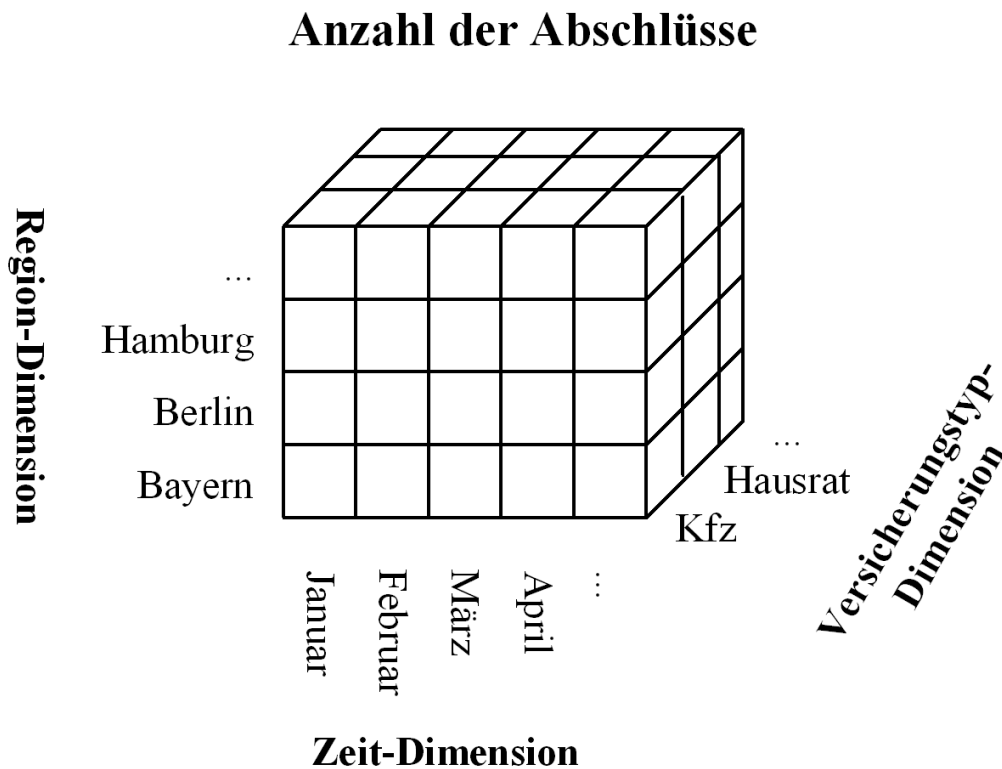


Abbildung 2.2: Dreidimensionaler *Hypercube* [Vitt02]

Ein Beispiel für einen dreidimensionalen *Hypercube* zeigt Abbildung 2.2. Dort wird die Anzahl der Versicherungsabschlüsse nach der Region, dem Monat und dem Versicherungstyp aufgeschlüsselt. In diesem Beispiel werden typischerweise alle Zellen des Würfels mit Zahlenwerten belegt sein, womit die Matrix voll besetzt wäre. Würde aber z.B. in einem zweidimensionalen Feld das Alter der Angestellten anhand deren Nachnamen und Sozialversicherungsnummern aufgeschlüsselt, so wäre dieses Feld nur sehr dünn besetzt. Eine direkte Speicherung dieser dünn besetzten Matrix würde zu erheblicher Verschwendung von Speicherplatz führen. Der Grund dafür liegt darin, dass diese Daten eigentlich keinen multidimensionalen Charakter haben. Dies lässt sich auch leicht dadurch überprüfen, dass eine zeilenweise Summenbildung über diesem Feld in keiner Richtung sinnvoll wäre. Im Beispiel aus Abbildung 2.2 könnte dagegen über jede beliebige Zeile eine Summe gebildet werden, die einen sinnvollen Wert ergeben würde. Die Daten sind damit hochgradig multidimensional. Diese Beispiele belegen die Aussage, dass ein *multidimensionales DBMS* auch nur für die Speicherung multidimensionaler Daten eingesetzt werden sollte. Da ein Data Warehouse aber in der Regel nicht ausschließlich multidimensionale Daten enthält, kommt ein solches System als alleiniges *DBMS* meist nicht in Frage. Zur Optimierung des Speicherplatzbedarfs und der Abfragegeschwindigkeit wird ein großer *Hypercube* mit vielen Dimensionen

gelegentlich auch in kleinere Multi-Cubes aufgeteilt, die dann jeweils die tatsächlich benötigten Dimensionskombinationen repräsentieren.

Die *objektorientierten DBMS* basieren auf dem Paradigma der Objektorientierung und versuchen die zugehörigen Konzepte auf Datenbanken zu übertragen. Für das Data Warehousing kommen *objektorientierte DBMS* prinzipiell in Frage, da sie gegenüber relationalen Systemen einige Eigenschaften besitzen, die für ein Data Warehouse von Vorteil sein können. So können in einem *objektorientierten DBMS* erheblich leichter auch komplexe, unstrukturierte Daten, wie Texte, Bildinformationen oder multimediale Daten abgelegt werden. Da die Analyse solcher unstrukturierter Daten heutzutage noch als nicht abgeschlossenes Forschungsgebiet betrachtet werden muss und die Verwendung solcher Daten schon eher in den Bereich des Knowledge Warehouse als des Data Warehouse fallen, dürfte deren Relevanz erst in den nächsten Jahren zunehmen. Die objektorientierten Modelle bieten darüber hinaus mächtige Konstrukte, wie z.B. die Bildung von Klassenhierarchien und Vererbung, die den Relationstypen des relationalen Modells überlegen sind. Leistungsmäßig sind die *objektorientierten DBMS* den relationalen in vielen Aspekten etwa gleichwertig, wenn nicht teilweise sogar überlegen. Allerdings sind *relationale DBMS* in Bezug auf skalierbare Parallelität oder Abfrageoptimierung heutzutage den *objektorientierten DBMS* noch überlegen. Die Hauptprobleme der *objektorientierten DBMS* für die Verwendung als Data Warehouse bestehen in dem derzeit noch geringen Reifegrad dieser Systeme, dem Fehlen von Standards und der bisher geringen Verbreitung dieser Systeme. Hier wird die zukünftige Entwicklung zeigen, ob und wann *objektorientierte DBMS* einen breiteren Einsatz im Data Warehouse-Bereich erfahren.

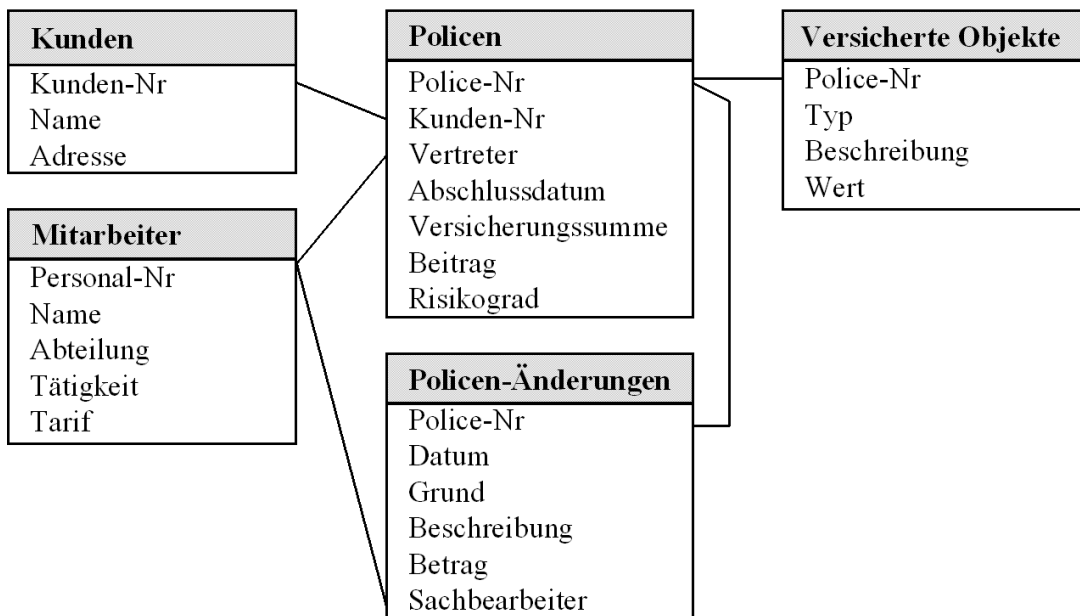


Abbildung 2.3: Ausschnitt aus einer Datenbankstruktur für OLTP [Führ02]

2.2.2 Das Sternschema

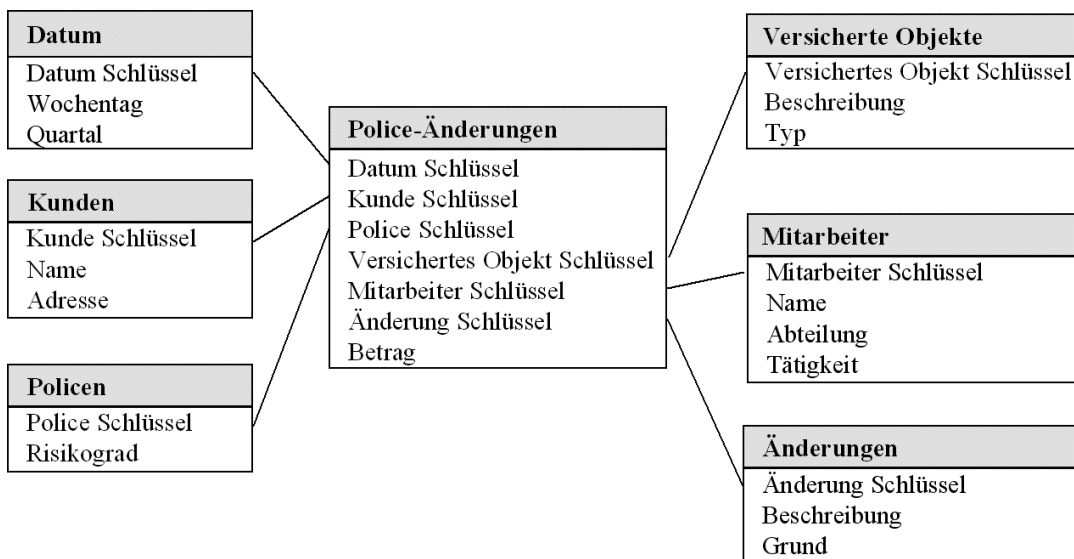
Wie bereits in Kapitel 2.1 erwähnt, werden im Rahmen des Tagesgeschäfts *OLTP* Systeme eingesetzt, die für eine schnelle und sichere Speicherung und Verarbeitung der anfallenden Daten sorgen sollen. Ein Ausschnitt aus der Datenbankstruktur einer Versicherung könnte z.B. so aussehen wie in Abbildung 2.3.

Man sieht deutlich, dass eine Vielzahl von Beziehungen zwischen den einzelnen Entitäten bestehen und in dieser Abbildung auch Beziehungen zwischen Entitäten dargestellt sind, die sich in der Praxis möglicherweise auf unterschiedlichen Systemen befinden. So werden die versicherten Objekte wahrscheinlich in einer anderen Datenbank abgelegt sein als die der Mitarbeiter, die wohl in der Personalabteilung zu finden sein wird. Dennoch werden auch die Sachbearbeiter Zugriff auf eine Mitarbeiter-Tabelle haben müssen. Eine komplexere Abfrage, die mehrere dieser Tabellen benötigt, wird damit schwierig und aufwendig, da sehr viele Tabellen verknüpft werden müssen, die zudem ggf. auf mehrere Systeme verteilt sind. Dazu kommen möglicherweise noch erheblich komplexere gemeinsame Schlüssel als sie in der Abbildung dargestellt sind.

Ein Data Warehouse soll solche komplexeren Abfragen ermöglichen, ohne dabei lange Wartezeiten zu verursachen. Das bei den *OLTP*-Systemen verwendete Entity Relationship Model ist dafür in der Regel nicht geeignet. Ralph Kimball [Kimb96] schreibt dazu:

“Entity relation data models are a disaster for querying because they cannot be understood by users and they cannot be navigated usefully by DBMS software. Entity relation models cannot be used as the basis for enterprise data warehouses.”

Heutzutage muss diese Aussage allerdings ein wenig relativiert werden, da die verfügbare Hardwareleistung und entsprechend optimierte Software es durchaus ermöglichen, auch auf relationalen Daten umfangreichere Analysen durchzuführen. Insbesondere Mischformen, bei denen häufig benötigte verdichtete Daten im *Sternschema* oder verwandten Schemata abgelegt werden, die meist seltener benötigten Detaildaten aber in einem relationalen Schema verbleiben, sind gelegentlich anzutreffen. Um auf einem *relationalen DBMS* ein Data Warehouse aufzubauen, wurde das *Sternschema* entwickelt. Ein Beispiel für die Repräsentation der Veränderungen einer Versicherungs-Police als *Sternschema* zeigt die folgende Abbildung 2.4.

Abbildung 2.4: Versicherungs-Policen als *Sternschema* [Führ02]

Das *Sternschema* ist charakterisiert durch eine einzelne *Fakttabelle*, deren Inhalt die zu analysierenden Daten sind. Häufig handelt es sich dabei um historische Daten. Diese Tabelle bildet die Mitte des Sterns und ist über je genau ein Schlüsselfeld mit den *Dimensionstabellen*, welche die Spitzen des Sterns bilden, verbunden. Diese Schlüsselfelder bilden zusammen den Primärschlüssel der Fakttabelle. Alle Schlüsselfelder sind generierte Schlüssel, d.h. es werden keine tatsächlichen Informationen wie Name, Datum oder dergleichen verwendet, sondern sie werden durchnummeriert. Für jeden zu analysierenden Bereich wird dann ein solcher Stern angelegt. Das Beispiel in Abbildung 2.4 enthält in der zentralen Fakttabelle die Änderungsbeträge für Versicherungs-Policen und die Schlüssel zu den Dimensionstabellen. Die Dimensionstabelle Datum repräsentiert die Zeitdimension. Dieser Dimension kommt eine besondere Bedeutung zu, da sie naturgemäß in jedem Stern auftritt, der historische Daten beinhaltet und die Wahl der zeitlichen Granularität entscheidenden Einfluss auf eine Vielzahl von Faktoren im Data Warehouse hat. Je kleiner die betrachteten Zeitabstände sind, desto größer werden die zu verwaltenden Datenmengen und umso kleiner wird in der Regel auch der Abstand zwischen den Aktualisierungen des Data Warehouse sein.

Die weiteren Dimensionen charakterisieren den Kunden, die Police, das versicherte Objekt, den Sachbearbeiter, der die Veränderung vorgenommen hat, und die Änderung, die auch Aufschluss darüber gibt, ob eine Aufstockung oder Reduzierung des Versicherungsbetrags stattfand. Zu beachten ist hierbei, dass jeweils nur solche Attribute in den Dimensionstabellen stehen, die als relevant für mögliche Analysen angesehen werden. So kann im Beispiel hinsichtlich der Police nur anhand des Risikogrades differenziert werden, da keine weiteren Informationen über die Police abgelegt werden. Die Fakttabelle enthält die Werte, die Gegenstand der durchzuführenden Analysen sind. Die Dimensionstabellen haben eher beschreibenden Charakter und repräsentieren die analyserelevanten Geschäftsdimensionen. In diesem Beispiel enthält die Fakttabelle nur einen numerischen Wert, es ist aber durchaus üblich dort mehrere relevante Werte

abzulegen. Ein *Sternschema* für die Umsatzanalyse eines Warenhauses könnte in der Faktttabelle z.B. den Verkaufsumsatz, die verkaufte Menge und die gelagerte Menge für jedes Produkt enthalten. In seltenen Fällen kann es vorkommen, dass die Faktttabelle gar keine Werte, sondern nur Schlüsselfelder enthält. Ralph Kimball bezeichnet solche Faktttabellen als *faktenlose Faktttabellen*. Kimball nimmt auch eine weitere Klassifizierung der Dimensionstabellen vor. So spricht er von *degenerierten Dimensionen*, wenn die Faktttabelle einen Dimensionsschlüssel enthält, zu dem es keine Dimensionstabelle gibt. Ein Beispiel dafür würde man erhalten, wenn man die Police-Dimension aus obigem Beispiel entfernen würde. Man könnte dann keine Analysen betreffend der Police mehr durchführen, würde aber, um ggf. noch auf das Versicherungsdokument zurückgreifen zu können, den Police-Schlüssel in der Faktttabelle behalten. Solche Dokumentnummern sind auch der häufigste Fall für eine degenerierte Dimension. Als *große Dimensionen* bezeichnet Kimball Dimensionstabellen, die selbst eine erhebliche Größe erreichen. Ein Beispiel hierfür ist die Produkt-Dimension in einem handelsbezogenen Data Warehouse, die häufig über einige Tausend Einträge verfügt und meist auch eine große Anzahl von Attributen aufweist.

Die Kundendimension in unserem Beispiel würde schnell zur großen Dimension anwachsen, wenn wir sie um demographische Informationen, wie Altersklasse, Familienstand, Einkommen, Geschlecht, Zahlungsmoral, etc. erweitern würden. Um dies zu vermeiden, könnte eine separate Demographiedimension angelegt werden, die dann über einen zusätzlichen Demographieschlüssel mit der Faktttabelle verbunden wird. Als *unsaubere Dimension* bezeichnet Kimball eine Dimension, die viele mehrfache Einträge oder unwesentliche Einträge enthält. Unter mehrfachen Einträgen werden solche verstanden, bei denen unter unterschiedlichen Schlüsseln ansonsten identische Daten abgelegt sind.

Unwesentliche Einträge sind solche, zu denen keine darauf verweisenden Fakten mehr existieren. Auch hierfür ist die Kundentabelle ein typisches Beispiel, da sie laut Kimball z.B. bei Banken und Versicherungen meist nur etwa zu 80 Prozent korrekt sind.

Werden mehrere Sterne angelegt, die evtl. auch über gemeinsame Dimensionstabellen verfügen, so wird auch gelegentlich von einem *Multisternschema* oder einem *Galaxy-Schema* gesprochen. Weitere Informationen zum *Sternschema* können z.B. folgenden Quellen entnommen werden: [Kimb96], [Muck98], [Voss94], [Conn01], [Anah97]. Aus der beschriebenen Grundform des *Sternschemas* haben sich inzwischen einige mehr oder weniger starke Abwandlungen wie das Snowflake- oder das Starflake-Schema ergeben, auf die an dieser Stelle nicht weiter eingegangen wird.

2.3 Datenimport

Der Datenimport besteht aus den Phasen *Datenextraktion* und *Transformation* und bezeichnet die Migration der Quelldaten in das Data Warehouse. Dazu werden sogenannte *ETL-Werkzeuge* (Extraction, Transformation, Loading) benutzt.

Die größten Schwierigkeiten werden in dieser Phase durch Unterschiede in den einzelnen Quellsystemen verursacht. Häufig basieren die Quellsysteme z.B. auf

unterschiedlichen Datenbanksystemen, es werden unterschiedliche Datenformate verwendet oder es gibt unterschiedliche Ausprägungen bzw. Bezeichnungen für dieselbe Semantik. Die Datenmigration beschreibt den Prozess der Datenübernahme aus unternehmensinternen oder –externen Quellen in das Data Warehouse. Hierbei muss sichergestellt werden, dass die übernommenen Daten folgende Kriterien erfüllen:

- Es sollten nur Daten in das Data Warehouse übernommen werden, die an für den Unternehmenserfolg relevanten Sachverhalten ausgerichtet sind. Des Weiteren muss die Möglichkeit bestehen, diese Daten nach den unterschiedlichsten Kriterien (Dimensionen) auswerten zu können.
- Die Daten werden bei ihrer Übernahme ins Data Warehouse mit einer Zeitmarke versehen. Diese Zeitmarken können sowohl Zeitpunkte, als auch Zeitintervalle widerspiegeln (z.B. Preis von Produkt A zum Zeitpunkt x, Umsatz des Tochterunternehmens T in den Jahren 1980 bis 1990).
- Die Daten sollten nur im Ausnahmefall nach ihrer Einspielung verändert werden.
- Die Struktur und die Formate der Daten sind so zu wählen, dass diese in einer einheitlichen Form in der Datenbasis vorliegen (z.B. Festlegung des Datumsformats, entweder englische oder deutsche Darstellungsform).

Ein Beispiel sind die vielen verschiedenen Möglichkeiten, ein Kalenderdatum zu speichern. Im Gegensatz zu einer vollständigen Speicherung des Datums wird z.B. häufig nur eine Zahl gespeichert, die relativ zu einem festgelegten Referenzdatum die Anzahl vergangener Tage angibt. Dazu kommen dann noch alle weiteren denkbaren Möglichkeiten bis hin zur Speicherung des Datums als Zeichenkette, die dann wieder in einer Unzahl von national und international gebräuchlichen Formaten aufgebaut sein kann. Im Data Warehouse müssen all diese unterschiedlichen Formate in ein einheitliches Format konvertiert werden.

Wenn die Daten nicht bereits in der Extraktionsphase in ein für das Data Warehouse geeignetes Format gebracht werden, schließt sich dann die Transformationsphase an, die ggf. in mehreren Schritten die Daten geeignet aufbereitet und formatiert. Dabei kann auch bereits eine Vorverdichtung von Daten stattfinden. So könnten z.B. häufig benötigte Monatswerte jeweils zum Monatsende aus den wöchentlich anfallenden Daten vorberechnet werden. Entsprechende Abfragen können dann auf die vorverdichteten Daten zugreifen, was zu einer Verbesserung der Abfragegeschwindigkeit führt. Am Ende der Transformationsphase werden die Daten schließlich auf Integrität und Konsistenz geprüft. Es folgt die letzte Phase, in der die Daten in das Data Warehouse übertragen werden. Abhängig vom Umfang der Daten ist an dieser Stelle zwischen einem kompletten Neuladen (*data refresh*), oder einem partiellen Laden (*data update*) des Data Warehouse zu entscheiden.

2.4 Anwendungen

Damit die im Data Warehouse abgelegten Daten zur Informationsgewinnung eingesetzt werden können, sind Anwendungen erforderlich, mit denen die

gewünschten Analysen durchgeführt werden können. Da diese Anwendungen ein sehr breites Spektrum abdecken ist es sinnvoll, Klassifizierungen vorzunehmen. An dieser Stelle soll zunächst eine Klassifizierung anhand des Anwendungszwecks vorgenommen werden.

Bei den Data Warehouse-Anwendungen handelt es sich um Analysewerkzeuge, die dem Anwender auf unterschiedliche Art und Weise und unter Einsatz verschiedener Verfahren bei der Suche nach entscheidungsrelevanten Informationen helfen sollen oder ihn sogar automatisch auf bestimmte Ereignisse aufmerksam machen.

Zusammengefasst werden diese Werkzeuge unter dem Oberbegriff *Online Analytical Processing (OLAP)*. Im Grunde ist dies auch die korrektere Verwendung des Begriffs *OLAP*, da er keinerlei Aussage über das konkrete Verfahren enthält. In der Data Warehousing-Literatur hat sich aber weitestgehend eine Verwendung dieses Begriffs für die unten beschriebene multidimensionale Analyse eingebürgert.

Deshalb wird in dieser Arbeit allgemein von Anwendung oder Analyse-Werkzeug gesprochen, wenn keine spezielle Art der Data Warehouse-Anwendung gemeint ist. Rick Tanler nennt in [Tan97] fünf Basisfunktionen, die in einem Analyse-Werkzeug integriert werden:

- Anwendungsschnittstelle: Die Bildschirmdialoge und Methoden, die dazu dienen die Benutzereingaben an interne Funktionen weiterzuleiten und diese auszuführen.
- Abfrage: Eine Applikationslogik, die SQL-Abfragen erzeugt.
- Verarbeitung: Eine Applikationslogik, die die Abfrageergebnisse analysiert.
- Formatierung: Eine Applikationslogik, die für eine geeignete Formatierung (z.B. Benennung von Zeilen und Spalten in einer tabellarischen Darstellung) sorgt sowie für eine möglichst standardisierte Speicherung (z.B. als HTML Seite).
- Anzeige: Die Anzeige der formatierten Daten, als Bericht oder Grafik auf dem Rechner des Anwenders.

Die folgende Klassifizierung in vier Bereiche, stellt keine strenge Unterteilung dar. Die Praxis erfordert meist eine Mischung verschiedener Analyseverfahren, was sich auch in den entsprechenden Entwicklungen und Produkten bemerkbar macht. In der Regel findet sich jedoch immer ein gewisser Schwerpunkt, der durch Anteile anderer Analyseverfahren ergänzt wird.

2.4.1 Abfrage- und Berichts-Werkzeuge

Abfrage- und Berichts-Werkzeuge stellen den einfachsten Typ von Analyse-Werkzeugen dar. Dabei reicht ihre Funktionalität zunächst kaum über das hinaus, was durch das zugrunde gelegte *relationale Datenbanksystem* ohnehin an Funktionalität zur Verfügung steht. Allerdings sollen diese Werkzeuge die Erstellung von Abfragen erleichtern, z.B. durch die Möglichkeit QBE¹ statt SQL zu

¹ QBE steht für Query By Example und bezeichnet ein Verfahren, Datenbankabfragen visuell durch Verbinden von Tabellen und Anlegen eines abstrakten Berichtsbeispiels zu definieren. Es findet z.B. im MS Access Verwendung.

verwenden und für eine ansprechende, durch den Anwender beeinflussbare Darstellung der Ergebnisse auf dem Rechner des Anwenders sorgen. Die meisten dieser Werkzeuge folgen dem Microsoft Windows Modell für eine einfach bedienbare Benutzungsschnittstelle und nutzen MICROSOFTS *ODBC-Schnittstelle* (open database connectivity) für die Anbindung an unterschiedliche *relationale DBMS* der führenden Datenbankhersteller. Die Funktionalität der Werkzeuge nimmt dabei von Version zu Version weiter zu.

Abfrage- und Berichts-Werkzeuge eignen sich insbesondere für die Erstellung von Listen, Zählungen oder Statusberichten, die keine aufwendigen Berechnungen erfordern. Dazu zählen beispielsweise folgende Abfragen:

- Statusberichte: Wie sahen die Zahlen zu den Neuabschlüssen im letzten Monat aus?
- Zählungen: Wie viele Kunden nutzen ein spezielles Angebot?
- Listen: Welche Kunden überschreiten eine bestimmte Versicherungssumme?

Neben der so genannten *Ad-hoc-Anfrage*, die kurzfristig und spontan erstellt wird, besteht auch die Möglichkeit, Abfragen zu automatisieren. So könnte beispielsweise monatlich ein Bericht mit den Verkaufszahlen des letzten Monats erstellt werden und den entsprechenden Mitarbeitern übermittelt werden oder im Intranet bereitgestellt werden.

Obwohl sich die Anwendung auf eher einfache Analysemöglichkeiten beschränkt, können viele Fragen hinsichtlich des Geschäfts bereits durch diese grundlegenden Informationen beantwortet werden. Es besteht dann häufig der Bedarf eine speziellere Analyse durchzuführen, für die dann die fortgeschritteneren Verfahren der Datenanalyse zum Einsatz kommen können.

2.4.2 Multidimensionale Analyse

Die multidimensionale Analyse stellt ein komplexeres Analyse-Verfahren des *OLAP* dar. Wenn dabei auch physisch ein *multidimensionales DBMS* eingesetzt wird, ist auch die Abkürzung *MOLAP* (*Multidimensional Online Analytical Processing*) gebräuchlich. Bei einer Realisierung auf Basis eines *relationalen DBMS* wird auch die Abkürzung *ROLAP* (*Relational Online Analytical Processing*) verwendet.

Die Daten werden dabei logisch in einem *Hypercube* organisiert. Ein Beispiel für einen dreidimensionalen Würfel wurde bereits in Abbildung 2.2 gegeben. Jede verwendete Dimension bildet eine Kante des *Hypercube* und jede Zelle des *Hypercube* enthält einen Wert für die sich dort überschneidenden Dimensionen. In der Regel sollten dabei über alle Zeilen, Spalten und denkbaren Ausschnitte oder Teilwürfel des Würfels sinnvolle Aggregationen, wie z.B. eine Summenbildung, erfolgen können.

Die einzelnen Dimensionen sollten in der Darstellung, die häufig in Form von *Pivottabellen* erfolgt, beliebig angeordnet werden können, was in etwa einem Drehen des *Hypercubes* entspricht und auch als *Rotation* bezeichnet wird. Dabei sollte es möglich sein, Schnitte durch den *Hypercube* zu legen, die nur einen Teil der Dimensionen des *Hypercube* enthalten. Dieses Verfahren wird dann auch als

Schneiden oder engl. *Slicing* bezeichnet. Im Beispiel aus Abbildung 2.2 könnte z.B. nur der zweidimensionale Bereich für einen bestimmten Monat ausgewählt werden. Wird entlang einer oder mehrerer Dimensionen nur ein Teilbereich ausgewählt, so spricht man von *Dicing*, so wie in Abbildung 2.5 dargestellt.

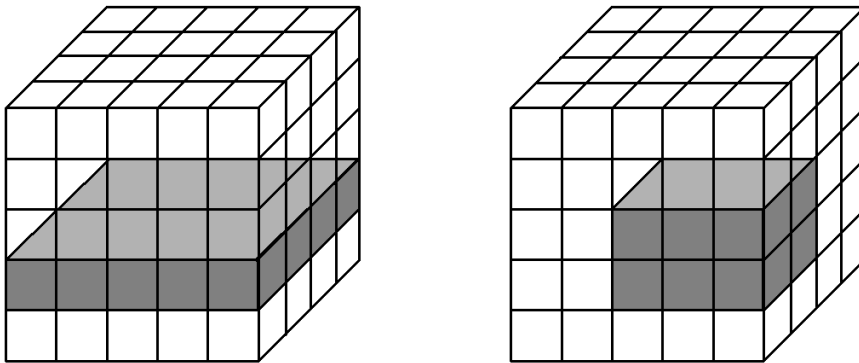


Abbildung 2.5: Veranschaulichung von Slicing und Dicing

Insbesondere bei Dimensionen, die eine Hierarchie bilden, kommen weitere Verfahren zum Einsatz, für die die Bezeichnung *drill down* und *drill up* bzw. *roll up* gebräuchlich sind. Unter *drill down* versteht man dabei das schrittweise Verfeinern einer Analyse durch Einbeziehung immer weiterer Detailstufen, z.B. aus einer hierarchischen Dimension.

Drill up bzw. *roll up* ist gerade der umgekehrte Weg, bei dem man von einer feinen Detailstufe zu immer größeren Zusammenfassungen aufsteigt. Abbildung 2.6 veranschaulicht diese beiden Verfahren. Bewegt man sich auf einer bestimmten Detailstufe in Querrichtung durch die Sichten auf den *Hypercube*, so spricht man von einem *drill across*, wie es in Abbildung 2.7 dargestellt ist.

Die wahre Stärke der multidimensionalen Analyse ist, dass diese Verfahren dabei in beliebiger Reihenfolge und über beliebige Dimensionen stattfinden können, ohne dass dabei die Integrität der Analyseberechnungen beeinträchtigt wird. Die Flexibilität der multidimensionalen Analyse ermöglicht solche Anfragen wie:

- Wie haben sich Werbemaßnahmen auf die Verkäufe ausgewirkt?
- Welche Angebote sollten geändert oder aus dem Programm genommen werden?
- Welches sind die treuesten Kunden?

Die multidimensionale Analyse benötigt zur Bereitstellung dieser Flexibilität eine zusätzliche Berechnungsebene gegenüber den einfachen Abfrage-Werkzeugen. Man führe sich dazu vor Augen, dass jede einzelne Zelle eines *Hypercube* normalerweise bereits einen Wert enthält, der sich aus einer Verdichtung der im Data Warehouse abgelegten Daten ergibt.

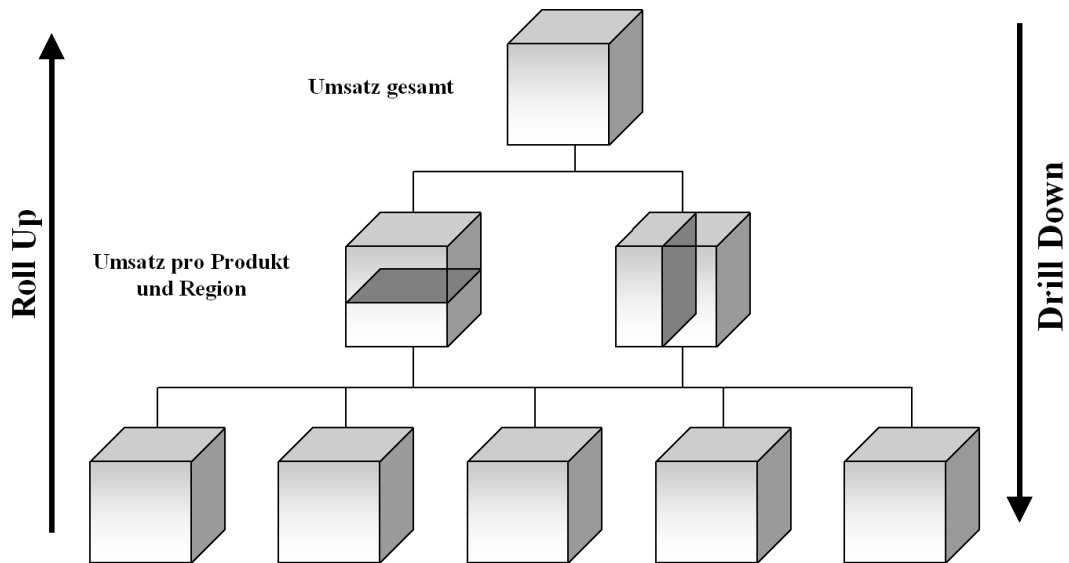


Abbildung 2.6: Drill Down und Roll Up

Die Operationen des Anwenders erfordern dann die Ermittlung weiterer Aggregationen, um z.B. Zwischen- und Gesamtwerte zu ermitteln. Zur Optimierung der multidimensionalen Analyse werden die Daten häufig auch physisch in einer multidimensionalen Struktur abgelegt. Dies kann einerseits temporär erfolgen, d.h. der *Hypercube* wird z.B. explizit für eine Analysesitzung aufgebaut und anschließend wieder verworfen. Dies ist natürlich nur bis zu einer gewissen Größe, die auch stark von der verfügbaren Hard- und Software abhängt, praktikabel. Andererseits könnten die *Hypercubes* auch ständig auf einem Server bereitgehalten werden und dann auch automatisch in der Aktualisierungsphase des Data Warehouses aktualisiert werden.

2.4.3 Statistische Analyse

Auf der nächsten Komplexitätsstufe der Analyse-Werkzeuge stehen die Werkzeuge zur statistischen Datenanalyse. Sie dienen dazu, große Datenmengen auf einfache Beziehungen oder Formeln, wie z.B. Mittelwert, Median, Standardabweichung, Regressionen, Korrelationen, usw. zu reduzieren.

Techniken der statistischen Analyse kommen insbesondere bei der Modellbildung für Verkaufsvorhersagen und Marktanteile zum Einsatz. Die typischen Fragestellungen bei der Anwendung dieser Modelle haben „was-wäre-wenn“-Charakter. Die Modelle sollen also eine Simulation alternativer zukünftiger Ereignisse ermöglichen. So liefert die Regressionsanalyse z.B. mögliche Beziehungen zwischen abhängigen Variablen von unabhängigen Variablen. Das Regressionsmodell kann dann dazu benutzt werden, erwartete Werte der abhängigen Variablen für gegebene unabhängige Variablen zu ermitteln. Wäre die Anzahl der Versicherungsabschlüsse die zu untersuchende abhängige Variable und die Beitragshöhe eine unabhängige Variable, dann könnten Fragen beantwortet werden, wie z.B.: „Was wäre, wenn wir die Beiträge erhöhen oder senken?“

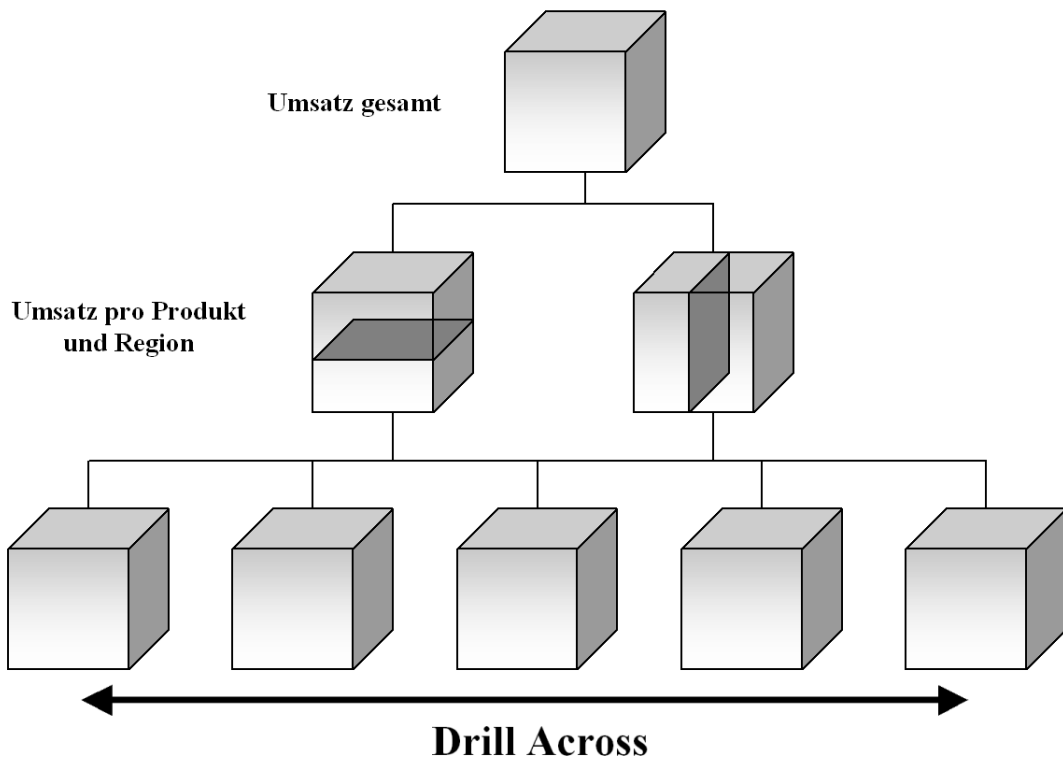


Abbildung 2.7: Drill Across

Bei der statistischen Analyse muss, im Gegensatz zu den Abfrage- und Berichtswerkzeugen und den multidimensionalen Analysewerkzeugen, bei denen häufig der Berichtsteller auch der Anwender ist, klar zwischen dem Analysten und den Anwendern der Modelle unterschieden werden. Es gibt im Normalfall immer nur einige Analysten, die statistische Modelle erstellen, während eine Vielzahl von Anwendern versucht, mit Hilfe des Modells Informationen zu gewinnen und Entscheidungen zu treffen. Dies ergibt sich zwangsweise aus der Komplexität der statistischen Analysen, die eine gewisse Präzision erfordern, um gültige Modelle liefern zu können. Folglich wird man entsprechend ausgebildete Spezialisten mit der Analyse und Modellbildung beschäftigen, während die eigentlichen Anwender nur noch mit dem fertigen Modell konfrontiert werden, in das sie dann jeweils aktuelle Werte einfügen können. Wie bei einfachen Berichten können auch hier bestimmte Analysen, z.B. eine monatliche Verkaufsprognose, automatisch zu festgelegten Zeitpunkten erstellt werden.

2.4.4 Data Mining

Data Mining verwendet viele der Techniken aus der statistischen Analyse und erweitert sie mit noch komplexeren Funktionen, um z.B. Muster und Beziehungen in analysierten Datenmengen zu finden. Dabei kommen fortgeschrittene Verfahren der Mustererkennung, Lernalgorithmen und neuronale Netze zum Einsatz, um

entsprechende Vorhersagemodelle zu erstellen. Data Mining ist besonders nützlich für die Analyse nichtlinearer Probleme mit einer großen Zahl von Variablen.

Während die statistische Analyse direkt auf den Benutzer ausgerichtet ist, werden Data Mining Anwendungen eher als im Hintergrund laufende Agenten implementiert, die den Anwender auf versteckte Muster oder Auffälligkeiten hinweisen sollen, die diesem sonst vermutlich nicht auffallen würden.

Obwohl der Markt für Data Mining-Anwendungen noch relativ jung ist, tummeln sich dort inzwischen eine Vielzahl von Anbietern und Produkten, so dass bei der Erwägung einer Anschaffung solcher Werkzeuge eine entsprechend sorgfältige Auswahl erfolgen sollte. Eine eigenständige Realisierung solch fortgeschrittener Techniken dürfte in der Regel nur für große Unternehmen in Frage kommen. Einen Überblick über das Thema Data Mining findet man in [Han00].

3 Konzeption eines Warehouse-Modells für das Content Management

Nachdem im vorherigen Kapitel Data Warehousing eingeführt wurde, werden in diesem Kapitel zuerst die allgemeinen Eigenschaften von *Content Management-Systemen* (CMS) vorgestellt. Danach folgt eine Übersicht über die Funktionen von *Content Management-Systemen*, die für das Konzept eines Warehouse-Modells besonders relevant sind. Abschließend werden die Schritte dargestellt, die notwendig sind, um die ausgewählten Daten des CMS in ein Warehouse-Modell zu überführen.

3.1 Content Management-Systeme

Zur Definition von Content Management-Systemen finden sich in [Roth01] zwei Charakterisierungen. Die erste definiert Content Management in einem engeren Sinne:

„Software-basiertes Content Management befasst sich mit der systematischen Sammlung und Verwaltung von Informationsbausteinen in einem einzigen (logischen) Bestand. Es stellt Anfragemethoden und Mechanismen für die sichere Arbeit ganzer Nutzergruppen mit diesem Inhaltsbestand (‘Content Base’) bereit.“

Unter dieser sehr allgemeinen Definition lassen sich auch Dokumenten-Management-Systeme und bei entsprechender Auslegung alle Datenbanken und Informationssysteme subsumieren.

Die zweite, weiter gefasste Definition, lautet:

„Software-basiertes Content Management befasst sich mit der systematischen Sammlung, Erstellung, Speicherung und Veredelung von strukturierten Inhalten und Mediendaten aller Art in einem einzigen, fein granulierten (logischen) Bestand. Es unterstützt gezielt die sichere Aggregation, Veredelung, Verarbeitung, Auswertung und Wiederverwendung dieser Content Base durch ganze Benutzergruppen.“

Es werden also gegenüber dem Dokumenten-Management neue Schwerpunkte gesetzt. Gerade die letzte Definition beschreibt alle Tätigkeiten, die zu Wertschöpfungsprozessen in der Informationsverarbeitung gehören, insbesondere Erstellung und Veredelung.

Zusammenfassend kann man in Anlehnung an die letzte Definition Content Management als die systematische Verwaltung von beliebig strukturierten Inhalten in einem zentralen Bestand zum Zwecke der Erstellung, Veredelung, Fortschreibung und Publikation verstehen. Der entscheidende Unterschied zum reinen Dokumenten-Management ist der Zweck, nämlich die Publikation.

Mehrplatz-fähigkeit	Check-In Check-Out	Bearbeitung Verifikation	Anfrage-funktionen	Fremd-formate
Protokoll-funktionen	Strukturierte Speicherung von typisiertem Inhalt			Daten-sicherung Rollback
Verarbeitung Verwaltung				Mehr-sprachigkeit
Zugangs-kontrolle	Meta-Information			Massen-operationen
Workflow				Aggregierung/ Beziehungen

Abbildung 3.1: Content Management-Funktionen zweiter Ordnung

Es folgt eine Übersicht der Anforderungen, die an ein Content Management zweiter Ordnung gestellt werden. Wie in Abbildung 3.1 dargestellt, sind dies:

Zugangskontrolle: Zugangskontrollen sind in arbeitsteiligen Umgebungen unverzichtbar. Sie basieren auf einer Benutzerverwaltung, welche die Rechte der einzelnen Benutzer kennt. Während der Anmeldung weist sich der Benutzer aus und das System überprüft, welche Zugangsberechtigungen der Benutzer hat. Die Leistungsfähigkeit der Zugangskontrolle wird bestimmt durch die Merkmale

- Sicherheit der Kontrolle
- Einteilung in Benutzergruppen, -rollen und -profile
- Granularität der Benutzerrechte

Protokollfunktionen: Protokollfunktionen erfassen Bestandsänderungen in der Content Base und können ebenso granular implementiert werden wie Zugangskontrollen. Im Extremfall ist die Protokollierung jeder einzelnen Interaktion mit der Content Base, zusammen mit Angabe zu Datum, Zeit und Nutzer, vorstellbar

Datensicherung: Professionelle Datensicherung ist unerlässlich, da Informationen die wichtigsten Aktivposten in einem Content Management-System darstellen. Erleichtert wird die Sicherung durch CMS, die die relevanten Inhalte an wenigen physischen Orten konzentrieren und dadurch Sicherungsstrategien erleichtern. Datenbankbasierte CMS können durch eine Replikation der Datenbank leicht gesichert werden.

Rollback: Rollback nennt man das Rückgängigmachen von Änderungen an Inhaltsbeständen mit der Bedingung, dass sich die Inhalte nach einem Rollback wieder in einem definierten, in sich konsistenten Zustand befinden.

Mehrplatzfähigkeit: CMS sind in der Regel mehrplatzfähig, um mehreren Benutzern eine gleichzeitige Bearbeitung von Daten zu ermöglichen. Datenbankbasierte Systeme sind in der Lage, die datenbankseitig bereitgestellten Funktionen, wie z.B. das Sperren von Tabellenzeilen oder -seiten oder das Transaktionsmanagement, zu nutzen.

Check-In und Check-Out: Mehrplatzfähigkeit lässt sich auch mit so genanntem Check-In und Check-Out-Mechanismen realisieren. Beim Check-Out teilt der Benutzer dem CMS mit, dass er bestimmte Inhalte bearbeiten möchte. Falls die Inhalte verfügbar sind, d.h. nicht gerade von einem anderen Benutzer „ausgecheckt“ sind, werden sie dem Benutzer zur Verfügung gestellt und als „checked-out“ markiert. Der Zustand der Inhalte wird im CMS gespeichert und ist damit persistent. Die Daten sind vor Änderungen anderer Benutzer geschützt, bis sie wieder „eingchecked“ werden.

Metainformation und Verwaltungsfunktionalität: Eines der wesentlichen Merkmale eines CMS ist seine Fähigkeit zur Anreicherung der verwalteten Objekte, also insbesondere der eigentlichen Inhalte, mit zusätzlicher Information. Von diesen zusätzlichen Daten, genannt Metainformation, existieren zwei Ausprägungen:

- Sie können bereits vom Hersteller des CMS vorgesehen sein. Dann spricht man von Klassifizierungsinformationen, wie z.B. Objekttyp, Thema oder Schlagworte und Verwaltungsinformationen, wie z.B. Autor, Bearbeiter, Tag der letzten Änderung oder Status.
- Sie können vom Benutzer frei nach seinen Bedürfnissen angelegt werden. Diese Informationen können vielfältiger Natur sein.

Anfragefunktionen: CMS können umfangreiche Informationsmengen verwalten, so dass die Notwendigkeit besteht, gezielt Teilmengen zu selektieren, wie z.B. alle Inhalte, die nach einem Stichtag hinzugefügt wurden.

Massenoperationen: Massenoperationen sind nützlich, wenn man große Mengen an Inhalten modifizieren möchte. So wäre es ohne eine solche Massenoperation sehr aufwändig, wenn man z.B. alle Inhalte löschen möchte, die von einem bestimmten Autor geschrieben wurden.

Bearbeitung und Verifikation: Einfache Content Management-Lösungen übergeben Inhalte zur Bearbeitung an externe Applikationen. Welche Bearbeitungsschritte der Benutzer in den Applikationen durchführt, sind dem CMS unbekannt. Ein einfaches CMS behandelt solche Inhalte als *black box*. Komplexere Systeme, wie z.B. Redaktionssysteme, bemühen sich, die Widersprüche der flexiblen Wahl eines Bearbeitungsprogramms einerseits und der guten Integration der Bearbeitungsfunktionalität in das Gesamtsystem andererseits zu lösen. Die möglichen Lösungen sind:

- Fernsteuerung von externen Applikationen
- Nutzung eines gemeinsamen Datenformats (z.B. HTML, XML oder RTF)
- Implementierung eines Editors im CMS

- Aufzeichnen atomarer Änderungen:** Für fein granulierte Protokoll- und Rollbackfunktionen müssen auch atomare Änderungen von Inhalten registriert werden können. Je feiner die gewünschte Granulierung, desto enger muss der Editor an das CMS angebunden sein.
- Aggregation und Beziehungen:** Als Vorbedingung für höhere Funktionen wie Mehrfachverwendung und Versionierung müssen sich Inhalte zu größeren Einheiten zusammensetzen lassen. Eine wichtige Funktion von CMS ist auch die Herstellung von Beziehungen zwischen Bestandteilen von komplexen Inhalten. Bereits ein einfaches HTML-Dokument enthält Verweise auf andere HTML-Dokumente oder auf Grafiken. Um dieses Leistungsmerkmal bereitstellen zu können, bedarf es einer leistungsfähigen Sprache zur Formulierung von Beziehungen zwischen Inhaltsobjekten.
- Versionsverwaltung:** Ein versionsfähiges CMS ermöglicht die parallele Verwaltung von verschiedenen Versionen derselben Objekte. Je nach gewünschter Granularität der Versionsverwaltung (komplettes Dokument oder Teile eines Dokuments), werden leistungsfähige Komponentenverwaltungen benötigt.
- Mehrsprachenfähigkeit:** Die Mehrsprachenfähigkeit eines CMS, also die parallele Verwaltung desselben Inhalts in mehr als einer Sprache, ist letztendlich eine spezialisierte Form der Versionierung. Der Umfang einer Mehrsprachenfähigkeit bestimmt dann, ob verschiedene Sprachversionen des Inhalts verschieden strukturiert sein können oder ob spezialisierte Programme zum Übersetzungsmanagement unterstützt werden.
- Workflow:** Ein Workflow modelliert die zugrunde liegenden Geschäftsprozesse und hilft bei der Beherrschung des Arbeitsflusses. Technisch gesehen funktionieren Workflows nach dem gleichen Prinzip, dass Objekte mehrere Zustände haben können und Interaktionen mit diesen Objekten Zustandsänderungen auslösen, die dann wiederum weitere Verarbeitungsschritte anstoßen.
- Gestaltung:** Leistungsfähige CMS arbeiten überwiegend nach dem Modell der Trennung von Struktur, Inhalten und Gestaltung. Struktur und Inhalte werden in einem strukturierten Datenbeschreibungsformat aufbewahrt, während die Gestaltung mit Hilfe von so genannten Vorlagen (Templates) in einer losgelösten Formatierungssprache realisiert wird. Die Leistungsfähigkeit dieses Verfahrens hängt neben der Ausdrucksstärke der Formatierungssprache auch davon ab, wie tief Inhalte strukturiert bzw. typisiert sind.
- Verarbeitungsfunktionen:** Kein CMS kann alle Verarbeitungsfunktionen bereitstellen, die ein Benutzer irgendwann einmal benötigen könnte. Verbreitete Verarbeitungsfunktionen sind Funktionen zur Formatierung von Inhalten und zur Rechtschreibkorrektur, weitergehende Funktionalitäten, insbesondere Automatisierungsprozesse, die von Makrorekordern bis hin zu ausgewachsenen Programmiersprachen reichen. Zumindest höherwertige CMS stellen außerdem eine Programmierschnittstelle (API) zur Verfügung,

mit deren Hilfe Drittanbieter zusätzliche Funktionen implementieren oder externe Programme anbinden können.

Fremdformatwandlung: Zuletzt wäre noch der Im- und Export von Daten zu erwähnen, weil kein CMS alle existierenden Datenformate beherrscht. Die Leistungsfähigkeit eines Imports bzw. Exports wird neben der Qualität des Filters entscheidend dadurch bestimmt, ob die Zielsprache alle Eigenheiten der Quellsprache wiedergeben kann.

3.2 Selektionskriterien für die Datenauswahl

Klassische Unternehmensdaten in einem Data Warehouse bestehen aus Finanz-, Umsatz- oder Produktionsdaten. Diese haben gemein, dass sie überwiegend aus numerischen Informationen bestehen. Im Gegensatz dazu verfügen Content Management-Systeme über eine Datenbasis, die aus den unterschiedlichsten Inhaltsobjekten, wie z.B. Textdokumenten, Bildern oder Videos besteht. Damit entziehen sich diese Objekte einer direkten Nutzung in einem Data Warehouse. Die Alternative besteht im Erheben von so genannten Metadaten, d.h. Informationen über die Nutzung der Informationen in einem Content Management-System. Die eigentliche Information im Warehouse-Modell besteht also aus dem Zugriff auf Daten.

Es werden nun einige der eben vorgestellten Komponenten eines Content Management-Systems näher daraufhin untersucht, inwiefern sie sich für eine Gewinnung von Metadaten eignen.

Zugangskontrolle: Durch Gewinnung von Daten in der Komponente „Zugangskontrolle“ kann man feststellen, wer sich wann und von welchem Rechner aus am System angemeldet, und eventuell auch wieder abgemeldet hat. Diese Daten haben für sich genommen nur eine geringe Aussagekraft und lassen keine umfangreiche Analyse zu.

Gestaltung: Die Trennung von Inhalt und Gestaltung führt in CMS zu einer vorlagenbasierten Benutzeroberfläche. Zu allen Inhalten, die auf dem Bildschirm dargestellt werden sollen, muss eine passende Vorlage existieren. Anhand der Reihenfolge, in der diese Vorlagen vom Benutzer, der z. B. auf einen Hyperlink klickt, aufgerufen werden, lassen sich folgende Informationen extrahieren:

- Von welcher Vorlage kam der Benutzer zur aktuellen Vorlage
- Welche Vorlage wird aktuell benutzt
- Zu welcher Vorlage geht der Benutzer

Wenn man in diesem Zusammenhang auch von der Komponente „Zugriffskontrolle“ Gebrauch macht, kann man beim Aufruf einer Vorlage zusätzlich feststellen, welcher Benutzer diese angefordert hat. Damit lässt sich lückenlos dokumentieren, welche Abfolge von Vorlagen (und implizit auch Inhalten, die sich allerdings durch die Vorlagen nicht eindeutig bestimmen lassen) der Benutzer auf der Oberfläche genommen hat.

Workflow: Jede Vorlage beinhaltet eine Menge von Verweisen auf andere Vorlagen. Durch die Abfolge der Vorlagen, die der Benutzer wählt, legt er, durch die Verweise in den Vorlagen geleitet, implizit den Workflow fest. Der Workflow ist also in den Daten, die durch die Komponente „Gestaltung“ erhoben werden, implizit enthalten.

Protokollfunktionen: Die Protokollfunktionen, die sich im Gegensatz zur Gestaltung lediglich auf den Inhalt beschränken, erfassen laut Definition Bestandsänderungen in der Content Base. Dazu gehören das Erstellen, Modifizieren oder Löschen von Inhalten. Zusammen mit den Daten Benutzer, Zeit und Datum ist auch diese Komponente eines CMS eine umfangreiche Datenquelle.

Wenn man die oben herausgestellten Datenquellen aus Layout, Inhalt und Zugangskontrolle kombiniert, erhält man folgende Daten zusammen mit deren jeweiligem Kontext:

- Benutzer-ID und damit Zugriff auf Informationen wie z.B. Gruppenmitgliedschaft, Profil etc.
- Name der aktuell benutzten Vorlage
- Zugriff auf ein Inhaltsobjekt (Art des Zugriffs und eindeutiger Verweise auf das Inhaltsobjekt)
- Datum und Zeit

Die Benutzer-ID ist eine eindeutige, anonymisierte Identifikation eines Benutzers, die es erlaubt, einen Wechsel der Vorlage oder einen Zugriff auf Inhalte einem Benutzer eindeutig zuzuordnen. Aus Gründen des Datenschutzes werden weder persönliche Daten noch die Internet-Adresse des Benutzers verwendet.

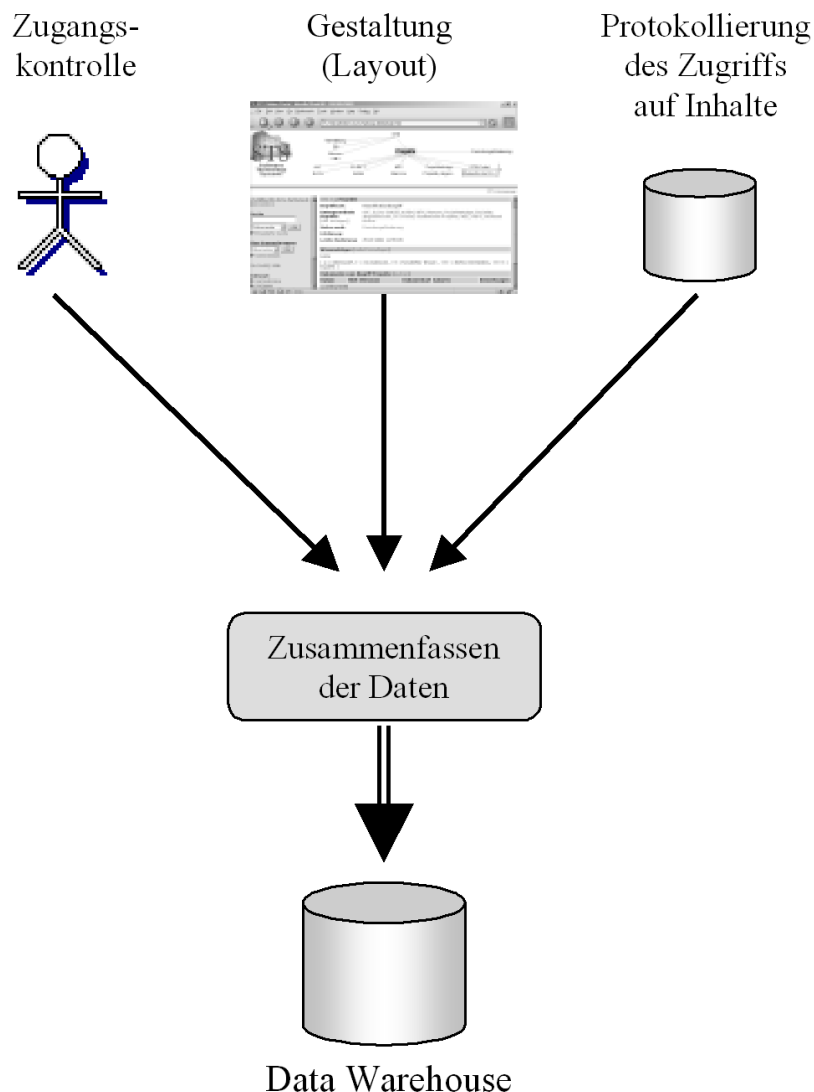


Abbildung 3.2: Datenfluss vom CMS zum Data Warehouse

3.3 Transformation der Daten

Nachdem im vorigen Abschnitt festgelegt wurde, welche Daten für ein Content Warehouse-Modell in Frage kommen, müssen diese Daten nun aufbereitet werden. Dazu wird ein Modul benötigt, das die drei Datenströme aus den Komponenten „Gestaltung“, „Protokollierung“ und „Zugangskontrolle“ zu einem Datenstrom zusammenfügt (siehe Abbildung 3.2). Nach dem Bereinigen und Hinzufügen eines Zeitstempels, werden die Daten in Form eines *Sternschemas*, das in Kapitel 2.2.2 vorgestellt wurde, in einer *relationalen Datenbank* abgelegt. Es fällt auf, dass die Faktentabelle leer ist. Im Gegensatz zu herkömmlichen Data Warehouse-Modellen, die in der Faktentabelle die zu analysierenden numerischen Daten enthalten, besitzt die Faktentabelle im Content Warehouse-Modell lediglich Fremdschlüssel für die

Dimensionstabellen. Allerdings ist die Information, dass der Benutzer z.B. ein Inhaltsobjekt adressiert hat, im Warehouse-Modell die eigentliche Information.

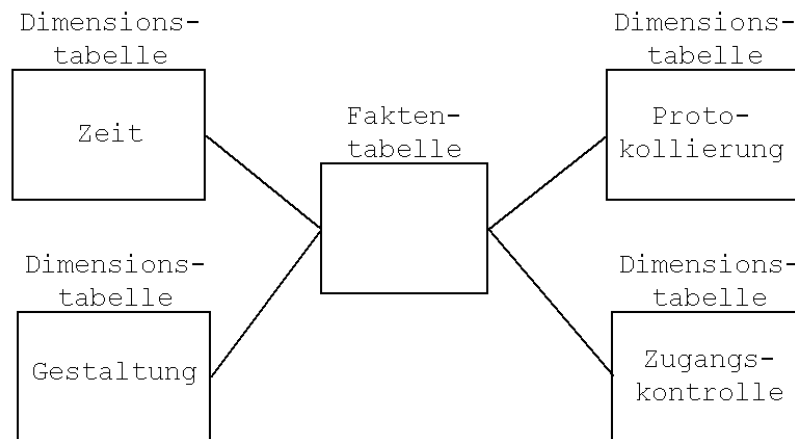


Abbildung 3.3: Erhobene Daten in Form eines *Sternschemas*

4 Content Management am Beispiel des INFOASSET BROKERS

Der INFOASSET BROKER [Info03] ist eine bewährte Standardsoftware für den Aufbau von Informations- und Wissensportalen im Intranet, Extranet und Internet. Hervorgegangen ist er 1999 aus einem Projekt der Technischen Universität Hamburg-Harburg. Die folgenden Kapitel beschreiben die Architektur des INFOASSET BROKER und speziell eine Erweiterung, die in Kapitel 5 benötigt wird.

4.1 Basisfunktionalität

Der INFOASSET BROKER ist eine Wissensmanagementlösung für mittelständische und große Unternehmen sowie Behörden zur Nutzung heterogener Wissensquellen im Unternehmen. Mit seinem Unified Asset Management integriert er Dokumente, Personen, Konzepte, Bewertungen, Tests u.v.m. unter einer Oberfläche (Abbildung 4.1). Dafür greift er auf unterschiedliche Datenquellen, wie z.B. Dateisysteme, Datenbanken oder Content Management-Systeme zu.

Um umfangreiche Informationsbestände nutzbar zu machen, bedarf es einer thematischen Kategorisierung des Wissens. Mit dem INFOASSET BROKER kann eine unternehmensweite *Taxonomie*² erstellt und gepflegt werden, die in einer grafischen, interaktiven Wissenslandkarte dargestellt wird. Die Wissenslandkarte dient der Strukturierung und transparenten Darstellung des Unternehmenswissens und erlaubt eine direkte Navigation durch die Konzepte zu Informationsobjekten und Know-how-Fragen.

Der Großteil des unternehmensweiten Wissens ist typischerweise in Form von Office-Dokumenten abgelegt. Die integrierten *Dokumenten-Managementfunktionen* des INFOASSET BROKERS umfassen u.a. Versionierung, Publizierung und automatische Verschlagwortung anhand der Wissenslandkarte.

Über die Verlinkung der Dokumente mit ihren Autoren stellt der INFOASSET BROKER eine direkte Verbindung zwischen dem *expliziten Wissen* und den menschlichen Erfahrungsträgern (*implizites Wissen*) her. Für erfolgreiches Wissensmanagement ist nicht nur die Wissensspeicherung entscheidend, sondern auch das schnelle und effektive Auffinden von Wissen. Der rollenbasierte Zugriffsschutz sichert die Integrität aller Informationsobjekte und erlaubt die Zusammenarbeit auch mit externen Partnern, die für bestimmte Bereiche des Enterprise Information Portals freigeschaltet werden können und damit direkt in die Kollaborationsprozesse integriert werden.

Zur Bedienung genügt ein marktgängiger Web-Browser. Die Installation von Client-Software oder spezieller Plug-ins ist nicht erforderlich.

² Taxonomie: Einordnung in ein System

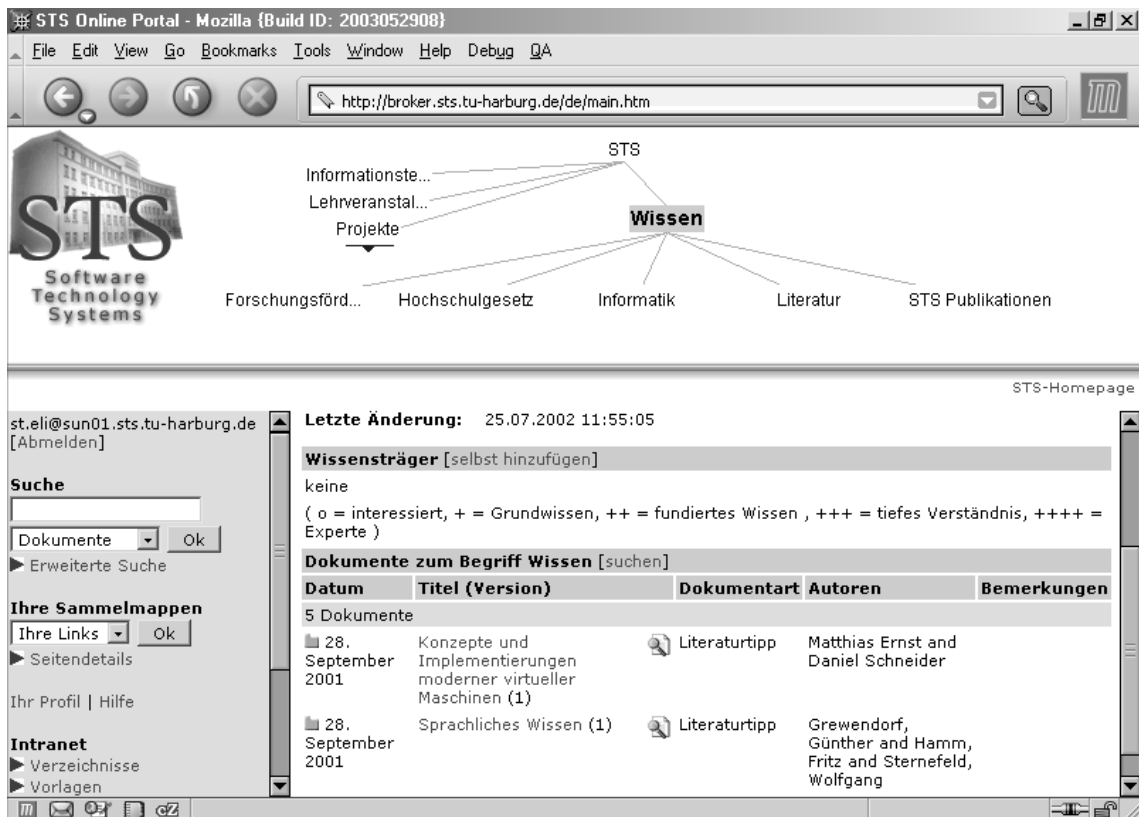


Abbildung 4.1: Screenshot der INFOASSET BROKER Benutzeroberfläche

4.2 Systemarchitektur

Der INFOASSET BROKER ist eine vollständig in Java-2 realisierte plattform-unabhängige Server-Software. Die medienneutrale Informationshaltung im INFOASSET BROKER ermöglicht den gleichzeitigen Zugriff für Benutzer mit verschiedenen Endgeräten auf dieselben Informationen. Mögliche Endgeräte umfassen Web-Browser, WAP-Handys, PDAs mit WAP/Web-Browser etc.

Als Informationsspeicher dienen wahlweise Dateisysteme, Datenbanken oder auch Content Management-Systeme. Je nach Einsatzgebiet können auch Kombinationen aus diesen Informationsspeichern verwendet werden. Der INFOASSET BROKER wurde konsequent als Client-Server-System entworfen, um hohe Skalierbarkeit zu gewährleisten. Die folgende Abbildung 4.2 zeigt den schematischen Aufbau der Schichtenarchitektur.

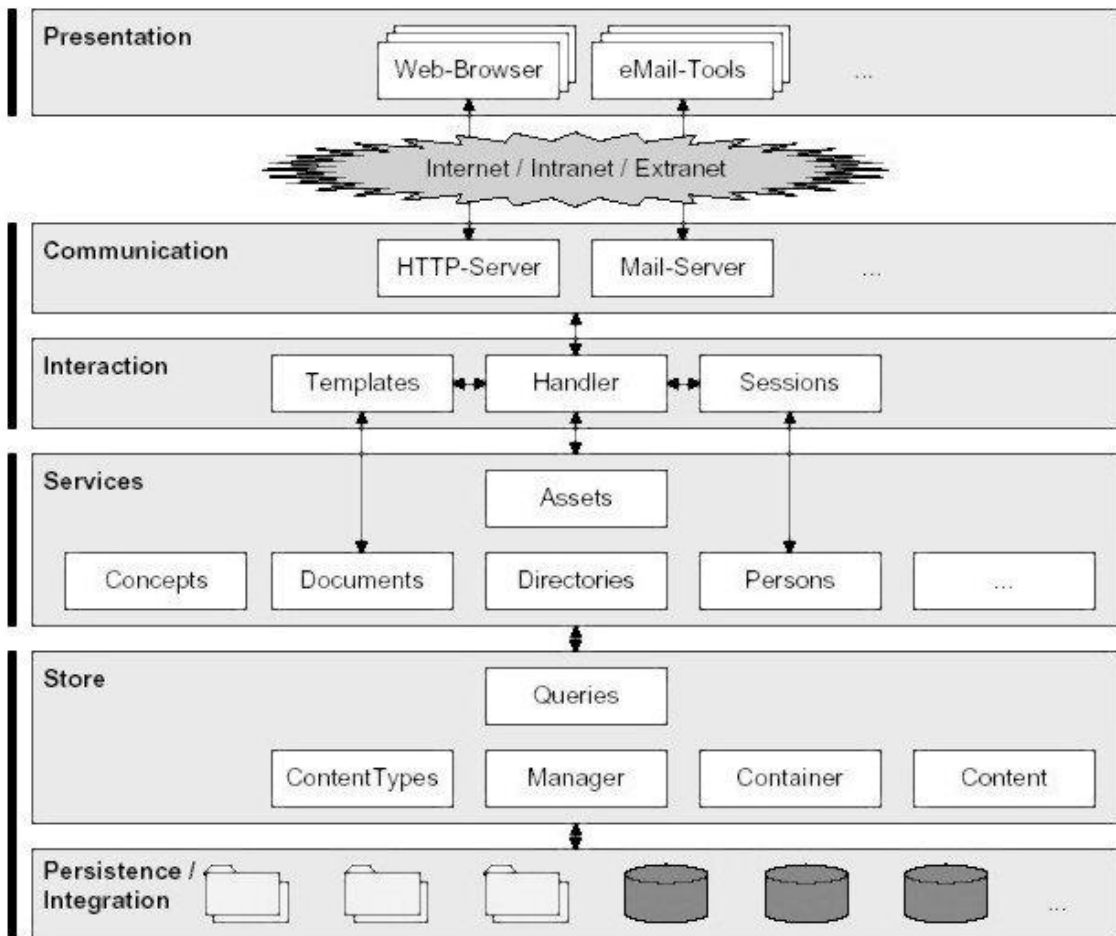


Abbildung 4.2 : Die Schichtenarchitektur des INFOASSET BROKER

Die sechs Schichten setzen sich zusammen aus [Wegn02]:

1. **Presentation:** Zur Darstellung der Präsentationsschicht (Graphical User Interface, GUI) wird wie bei Internet-basierten Systemen üblich als Benutzeroberfläche primär ein Web-Browser und für bestimmte Aufgaben eine e-Mail-Anwendung benutzt. Auch spezielle mobile Geräte wie WAP-fähige Mobiltelefone oder Personal Digital Assistents (PDA) können unterstützt werden. Im Sinne der geforderten Medien- und Aktorenunabhängigkeit kann potentiell eine Vielzahl von Endgeräten bedient werden. Für die geforderten Portalfunktionen sind zunächst Web-Browser und e-Mail zwingend zu unterstützen. Sie kommunizieren über die üblichen Internet-Protokolle (HTTP, SMTP, POP, FTP, etc.) mit der Kommunikationsschicht.
2. **Communication:** Diese Schicht dient der Kommunikation mit der client-seitigen Präsentationsschicht und der Umsetzung der jeweiligen Protokolle und Interaktionsmuster auf die darunterliegende Schicht. Hier sind insbesondere ein Web-Server zur Kommunikation über HTTP sowie ein e-Mail-Server zur Generierung und zum Empfang von e-Mail erforderlich. Weiterhin können auf dieser Schicht bereits Maßnahmen zur Lastverteilung realisiert werden, falls dazu mehrere Instanzen der darunterliegenden Schichten existieren.

3. **Interaction:** Diese Schicht realisiert die medien- und aktorenunabhängigen Abstraktionen des zugrunde liegenden Modells. Der Kommunikationsschicht obliegt die Übersetzung der medienabhängigen Protokolle und Interaktionsformen in ein generelles, auf dieser Schicht verstandenes Schema. Dazu existiert als grundlegender Mechanismus die Regel (Handler), die Anfragen aller Art in den Rollen von Dienstleister oder Kunde bearbeitet bzw. stellt. Zur Generierung der Antworten auf Anfragen wird ein vorlagenbasierter Ansatz gewählt, der es unabhängig von der Funktionalität der Regeln erlaubt, Gestaltungsvorlagen (Templates) mit dem aktuell benötigten Inhalt zu füllen und über die Kommunikationsschicht auszuliefern. Dieses Konzept der Templates kommt der Forderung nach Trennung von Struktur, Inhalt, Layout und Funktionalität nach. In den Regeln wird lediglich die Umsetzung der Präsentation und Funktionalität der Dienste geleistet. Zudem wird in dieser Schicht eine Sitzungsverwaltung (Sessions) realisiert, die insbesondere von dem Web-Server der darüberliegenden Schicht benötigt wird, um die Mängel des zustandslosen Protokolls HTTP auszugleichen, die aber auch von den Regeln als Kontext benötigt wird. Die Gestaltungsvorlagen und Sitzungsinformationen werden teilweise auf die semantischen Objekte (*Assets*) der Dokumente und Personen der darunterliegenden Dienstsicht abgebildet. Die Sitzungsschicht ist die Komponente, in der, wie im vorherigen Kapitel beschrieben, die Daten der verwendeten Vorlagen entnommen werden können.
4. **Services:** Diese Schicht enthält die eigentliche Anwendungslogik des Portalsystems. Diese umfasst die objektorientierte Realisierung der *Assets* der semantischen Objekte wie Begriffe, Dokumente, Verzeichnisse, Personen und Verknüpfungen, sowie aller weiteren Klassen. Diese Dienste werden von allen Regeln der darüberliegenden Schicht in Anspruch genommen. Zur Sicherstellung der Persistenz der erzeugten *Assets* wird transparent für die darüberliegenden Schichten die nächst tiefere Schicht in Anspruch genommen.
5. **Store:** Diese Schicht stellt eine einheitliche Abstraktionsebene dar, die einerseits die transparente Nutzung verschiedenartiger und verschiedener Persistenzmechanismen wie Datenbanken oder Dateisysteme erlaubt und andererseits zur Integration bestehender Anwendungen wie Content Management-Systeme, betrieblicher Informationssysteme oder sonstiger beliebiger Systeme dienen kann. Jede konkrete Anbindung basiert auf einer ‚Manager‘ genannten Komponente, die Container und Content-Objekte verwaltet.
6. **Persistence/Integration:** Diese letzte Schicht enthält die konkreten, von der darüberliegenden Store-Schicht verwendeten Persistenzdienste und integrierten Systeme, die üblicherweise von Drittherstellern stammen. Hier sind *relationale* oder *objektorientierte Datenbanken*, Dateisysteme, Content Management- und Dokumenten-Management-Systeme, bestehende Verzeichnisdienste (LDAP, NIS etc.), News-Server, externe e-Mail-Server, externe Web-Server, News-Ticker und anderes mehr denkbar.

4.2.1 Die Konzepte des INFOASSET BROKER

Das elementare Konzept ist das Unified Asset Management. Dies bedeutet, dass alle Informationsobjekte, wie z.B. Begriffe, Texte, Bilder, Linktipps, Literaturtipps, Personen, Gruppen oder Aufgaben, einheitlich als *Assets* geführt werden. Für den Anwender hat dies den Vorteil, dass sämtliche generischen Funktionen auf allen *Assets* zur Verfügung stehen, wie z. B. Suche, Verlinkung, persönliche Bewertungen, automatische Empfehlungen, automatische Verschlagwortung, Klassifikation usw.

Der INFOASSET BROKER unterstützt die Vernetzung beliebiger *Assets*. Wissensnetze und semantische Zusammenhänge können direkt durch Hyperlinks dargestellt werden. Unified Link Management ermöglicht heterogene Linksammlungen und das Aufspüren häufig und weniger häufig genutzter Informationen.

Die Template-Technologie des INFOASSET BROKERS unterstützt eine eindeutige Trennung zwischen Ablauflogik, Content und Layout. Neben der Vereinheitlichung des Layouts der dynamisch generierten WML/HTML-Seiten eröffnet die Verwendung von Templates die Möglichkeit kundenspezifischer Anpassungen. Ein Beispiel dafür ist das EURIFT-Portal [EURIO3], das ebenfalls auf der INFOASSET BROKER-Technologie basiert. Durch Anpassen der Vorlagen an den jeweiligen Kunden können so die unterschiedlichsten Benutzeroberflächen erstellt werden. Es ist auch möglich, die Benutzeroberflächen von der Benutzergruppe abhängig zu machen. So kann z.B. die Redaktion zum Erstellen und Bearbeiten von Informationen eine andere Oberfläche (Vorlagensatz) benutzen als die Kunden, die sich die publizierten Informationen anschauen.

Die Informationsobjekte zur Modellierung von *Assets* unterliegen im INFOASSET BROKER keinem starr vordefinierten Schema, sondern können in ihrer Struktur an die Erfordernisse des Kunden angepasst werden.

Die Schnittstellen der INFOASSET BROKER Standardsoftware sind über die Schichten der Architektur hinweg weitgehend offen gelegt. Hierdurch wird erreicht, dass das System in vielen Bereichen um Dienste, Werkzeuge und Content-Manager erweitert und an die spezifischen Wünsche einzelner Kunden angepasst werden kann. Der INFOASSET BROKER ist eine generische Standardsoftware die "out of the box" einsetzbar ist. Aufgrund seiner Anpassungsmechanismen bietet er eine effiziente Plattform für die Erstellung kundenspezifischer Portale.

4.2.2 Die Kernfunktionen des INFOASSET BROKER

Es folgt eine Übersicht über die Kernfunktionalität des Brokers. Dokumente sind ein wichtiger Bestandteil zum Erfassen und Verteilen des expliziten Wissens. Multimediale Dokumente, wie z.B. MS Word-Dokumente, Bilder oder Videoclips, können durch Hochladen im INFOASSET BROKER zugänglich gemacht werden. Die integrierten Dokumentenmanagementfunktionen beinhalten u.a. Volltextsuche, Metadatensuche, Versionierung, Publikation und automatische Verschlagwortung anhand der Wissenslandkarte.

Personen- und Gruppenprofile beschreiben die Träger des impliziten Wissens durch Verwaltung einheitlicher und detaillierter Personenprofile für Benutzer, anonyme Gastnutzer, Mitarbeiter, Kunden etc. Die Sichtbarkeit und der Detaillierungsgrad der

Personenprofile ist für unterschiedliche Gruppen getrennt einstellbar. Die Gruppenverwaltung unterstützt die Darstellung von Bereichen, Abteilungen, Projektgruppen, Verteilern, etc. Die Rechtevergabe ist gruppenbezogen, d.h. ein Benutzer besitzt die Rechte aller Gruppen, deren Mitglied er ist. Personendaten, Anmeldeinformationen und Gruppenzugehörigkeiten können optional über LDAP (lightweight directory access protocol) mit anderen Systemen ausgetauscht werden. Die Unternehmensterminologie wird durch Begriffe strukturiert erfasst. Begriffe und ihre Beziehung bilden die Wissenslandkarte, eine verbindliche Taxonomie für die Klassifikation von Informationsobjekten. Der Benutzer kann, ausgehend von einem Begriff, systematisch an weitere relevante Informationen, z.B. Wissensträger und Dokumente etc., herangeführt werden. Zur Unterstützung von Teamarbeit stehen für den direkten Wissensaustausch themenspezifische Diskussionsforen und Online-Chats zur Verfügung. Weitere Informationen zum Broker sind zu finden unter: [INF01], [INF02a], [INF02b].

4.2.3 Die INFOASSET BROKER Erweiterungen

Neben den oben genannten Basisfunktionen enthält der Broker wahlweise zusätzliche Funktionen, sogenannte *Extensions*. Jede *Extension* kann einzeln hinzugenommen werden. Dazu ist lediglich eine Rekonfiguration und ein Neustart des Systems erforderlich. Die verfügbaren INFOASSET BROKER *Extensions* im Überblick:

Asset Discovery Extension: Die selbstlernende Asset Discovery Extension stellt eine Fuzzy-Ähnlichkeitssuche sowie die automatische Klassifikation von Dokumenten zur Verfügung.

Recommendation Extension: Die Recommendation Extension dient der Bewertung von Dokumenten durch Nutzer mit ähnlichem Interessenprofil. So werden Informationsobjekte, die von Nutzern mit ähnlichen Interessen als besonders hilfreich eingestuft wurden, empfohlen und in Trefferlisten bevorzugt angezeigt.

Competitor Information Extension: Die Competitor Information Extension dient der strukturierten Sammlung und Aufbereitung von Daten über Wettbewerber aus unterschiedlichen Quellen.

Online Test Extension: Mit der Online Test Extension können Web-basierte Tests zusammengestellt, automatisch durchgeführt und ausgewertet werden. Die Online Test Extension dient typischerweise dem *Recruiting* oder wird im Zusammenhang mit *E-Learning* verwendet.

Shop Extension: Die Shop Extension dient dem Verkauf (digitaler) Güter im Internet. Im Intranet kann sie zur kostenstellenbasierten Abrechnung von Online-Informationen genutzt werden.

Alerting Extension: Die Alerting Extension stellt Benachrichtigungsdienste bereit, mit der Benutzer gemäß ihren persönlichen Interessen informiert werden, wenn für sie relevante Dokumente verändert oder neue Dokumente

innerhalb ihres Interessengebietes hinzugefügt wurden. Auf diese Extension wird im folgenden Kapitel näher eingegangen.

Über diese Extensions hinaus ist der INFOASSET BROKER flexibel an die in der Praxis anzutreffenden Kundenwünsche anpassbar bzw. erweiterbar. So bestehen vorgeplante Erweiterungspunkte an den folgenden Stellen der Gesamtarchitektur:

- Vollständige Anpassbarkeit der multilingualen Oberfläche (z.B. Layout, rollenspezifische Sichte, Sortier- und Selektionskriterien) gänzlich ohne Programmierkenntnisse
- Sehr weitgehende Anpassbarkeit der interaktiven Dialogfolgen und Systemdienste durch Parameterdateien ohne Programmierung
- Framework-Unterstützung und klar definierte Java-APIs für programmierte Systemerweiterungen:
 - Ereignis- und vorlagengesteuerte Dialogprogrammierung als unifizierender Mechanismus zur Definition und Anpassung von Push- und Pull-Diensten
 - Sitzungsmanagement und Authentisierung (rollenbasiert und zertifikatbasiert) mit Möglichkeiten zur Integration mit Sitzungsmanagement-Lösungen von Drittsystemen

4.3 Die Alerting Extension

Die bereits im vorigen Abschnitt erwähnte Alerting Extension soll nun ausführlicher dargestellt werden, da die Realisierung des Warehouse-Modells (vgl. Kapitel 5.1.1) von dieser Erweiterung Gebrauch macht.

Unter Notifikation (*Alerting*) im Allgemeinen versteht man jegliche Form der Benachrichtigung eines Benutzers in Softwaresystemen. Sie kann z.B. durch Zusenden einer e-Mail oder durch das automatische Aufzeigen von bestimmten Inhaltsänderungen erfolgen. Notifikation beschreibt also die gezielte elektronische Benachrichtigung interessierter Personen über für sie relevante Vorgänge.

Die Alerting Extension stellt Benachrichtigungsdienste bereit, mit der Benutzer gemäß ihrer persönlichen Interessen informiert werden, wenn die für sie relevanten Informationsobjekte (*Assets*), wie z.B. Dokumente, Ordner, Themen, Kontaktinformationen, oder gespeicherte Suchergebnisse verändert oder neue Informationsobjekte innerhalb ihres Interessengebietes hinzugefügt wurden. Dabei bestimmt der Benutzer selbst die Granularität und Aktualität der Änderungen, indem er die Informationsobjekte, über die er informiert werden möchte, in seine Sammelmappen einfügt. Im Falle einer Änderung wird er dann darüber informiert, dass eine Änderung eines *Assets* stattgefunden hat. Eine Benachrichtigung kann aus einer personalisierten e-Mail (ASCII-Mail, HTML-Mail) oder einer SMS bestehen.

Die Vorteile der Alerting Extension bestehen in einer Beschleunigung arbeitsteiliger Prozesse (Information-Push ergänzt Information-Pull) und einer flexiblen Berücksichtigung persönlicher Informationsbedürfnisse und Arbeitsstile.

Eine wichtige Rolle spielen in diesem Zusammenhang sogenannte *Goals*. Der Benutzer verfolgt während der Interaktion mit dem System verschiedene Ziele, wie z.B. Dokumente anlegen oder Personendaten ändern. *Goals* repräsentieren diese

Ziele in der Dialog- und Workflow Schicht des Systems und existieren für die wichtigsten *Assets*, wie Dokumente, Verzeichnisse, Konzepte, Personen, Gruppen, und ihre Beziehungen untereinander.

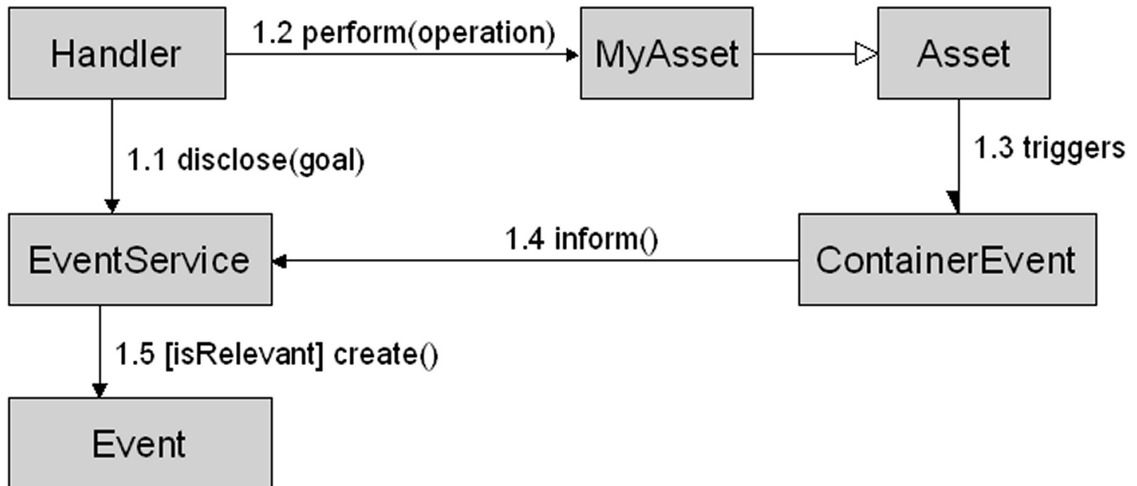


Abbildung 4.3: Typischer Ablauf beim Erzeugen eines Goals

Im Beispiel in Abbildung 4.3 sieht man, dass der Handler, der die an den Web-Server des Brokers gestellten Anfragen beantwortet, mit dem Aufruf `disclose(goal)` zuerst das zum jeweiligen Handler passende *Goal* erzeugt. Dann führt der Handler die eigentliche Operation aus, die zu einer Veränderung eines *Assets* führen kann. Dadurch ausgelöst wird ein `ContainerEvent`, das den `EventService` über die Änderung informiert. Dieser fragt dann das *Goal*, ob die Änderung des *Assets* zum Ziel des *Goals* passt. Wenn ja, wird das Event an alle

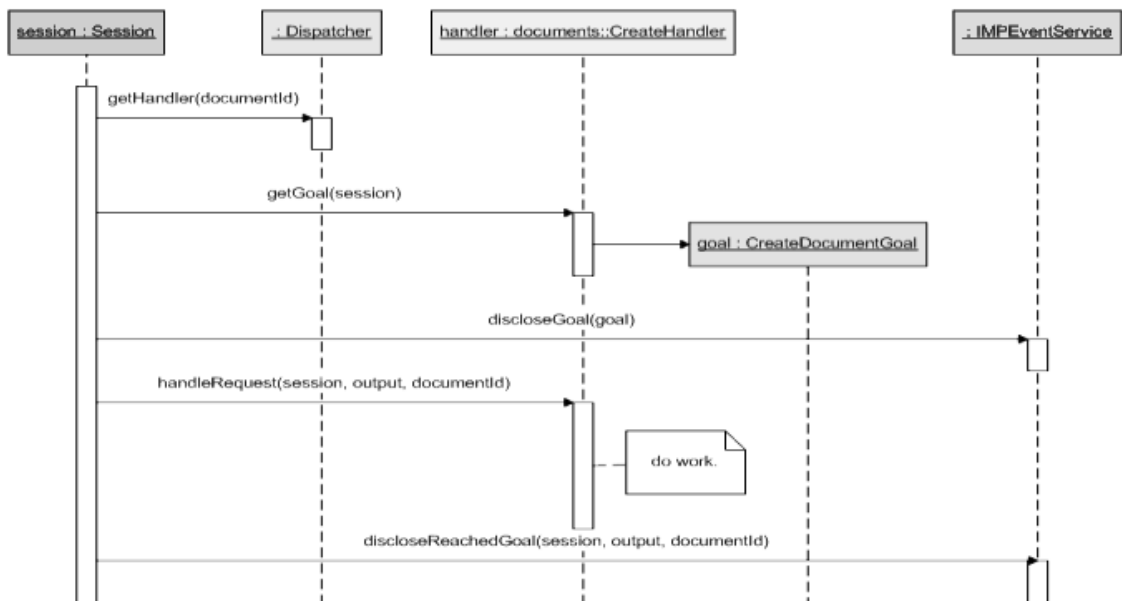


Abbildung 4.4: Sequenzdiagramm für die An- und Abmeldung eines Goals

registrierten `EventListener` geschickt, und wenn nein, wird das *Goal* wieder gelöscht. Durch Anmeldung beim `EventListener` kann man sich über das Auftreten von der `AlertingEvents` informieren lassen. Dieser Mechanismus wird in Kapitel 5.1.1 benutzt, um Änderungen an *Assets* festzustellen.

In Abbildung 4.4 ist dieser Ablauf noch einmal in einem Sequenzdiagramm für das *Goal CreateDocumentGoal* dargestellt.

5 Realisierung des Content Warehouse-Modells

Nachdem sich Kapitel 3 allgemein mit der Eignung von Content Management-Systemen für das Data Warehousing beschäftigt hat, werden die dort vorgestellten Verfahren für das Wissensmanagement-Portal INFOASSET BROKER implementiert. In Kapitel 5.1 wird zunächst die Erweiterung des BROKERS um die *Tracking Extension* erläutert. Der *Tracking Server*, der die von der *Tracking Extension* erhobenen Daten entgegennimmt und aufbereitet, wird in Kapitel 5.2 behandelt. Die Ergebnisse der Realisierung werden anhand eines Beispiels in Kapitel 5.3 dargestellt.

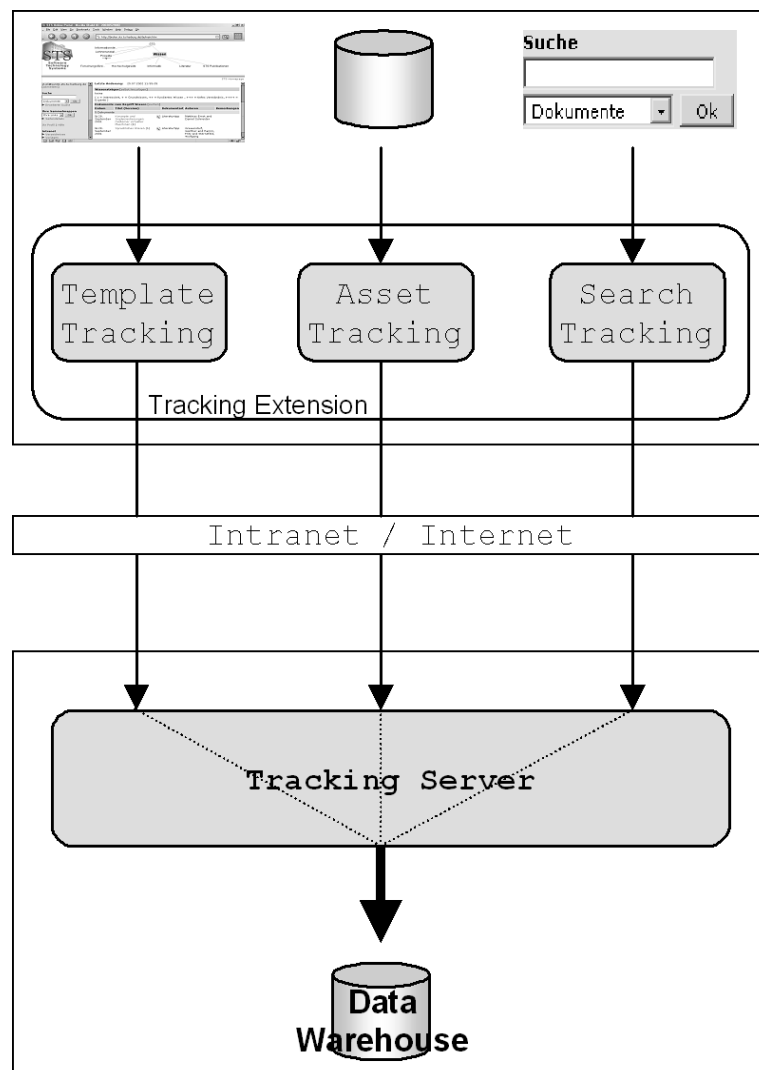


Abbildung 5.1: Datenfluss vom Broker zum Data Warehouse

5.1 Die *Tracking Extension*

Das Warehouse-Modell in Kapitel 3.2 unterscheidet zwischen Layout und Inhalt. Die *Tracking Extension* greift diese Zweiteilung auf und erweitert den INFOASSET BROKER um die *Tracking Extension*. Der Datenfluss von Broker zum Data Warehouse wird in Abbildung 5.1 gezeigt. Die *Tracking Extension* umfasst folgende Klassen: `AssetTracking` [Ecke98] nutzt die `Alerting Extension` zur Überwachung von Änderungen an *Assets*, `TemplateTracking` überwacht den Wechsel der Vorlagen in der Layout-Komponente und `SearchTracking` zeichnet Suchanfragen der Benutzer auf. Abbildung 5.2 zeigt, in welchen Schichten des Brokers die *Tracking Extension* ansetzt. Der folgende Abschnitt befasst sich mit dem Aufzeichnen von Änderungen an Inhaltsobjekten, den *Assets*.

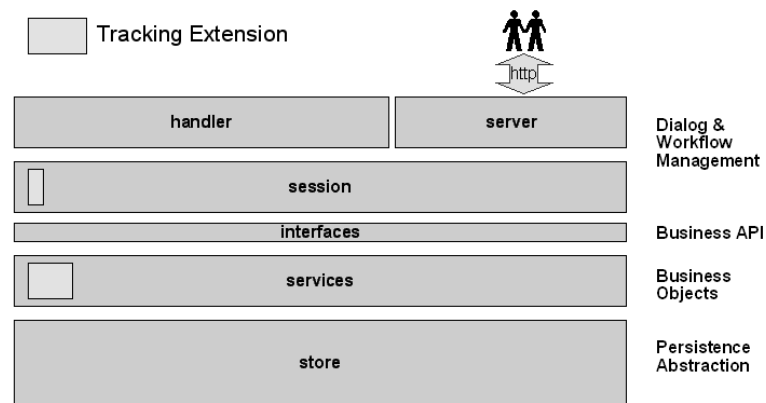


Abbildung 5.2: Erweiterungen des Brokers durch die *Tracking Extension*

5.1.1 Aufzeichnen von Änderungen in der Inhalts-Komponente

Um eine transparente Einbindung der *Tracking Extension* in den INFOASSET BROKER zu ermöglichen, wurde die in Kapitel 4.3 vorgestellte `Alerting Extension` benutzt. Diese stellt das Interface `EventService` (Abbildung 5.3) zur Verfügung, das mit der Funktion `addListener` eine Anmeldung erlaubt. Daraufhin erfolgen Benachrichtigungen über alle Änderungen an *Assets*.

Die Klasse `AssetTracking` erhält also neben der Benachrichtigung über eine Änderung zusätzlich einen Verweis auf das Ziel des eingetretenen Ereignisses (*Goal*). Über die Klasse `Goal` erhält man folgende Daten:

- `actorId`: Die `UserId` des Benutzers.
- `targetId`: Die `Id` des geänderten *Assets*.
- `targetName`: Der vom Benutzer vergeben Name des *Assets*.
- `targetType`: Der Typ des geänderten *Assets*.
- `goalName`: Der symbolische Name des *Goals*

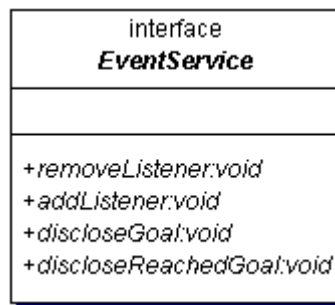


Abbildung 5.3: Das Interface EventService

Abbildung 5.4 zeigt, aus welchen Daten ein Ereignis besteht, das durch eine Änderung eines *Assets* ausgelöst wurde. Die Spalte Attributwerte stellt in dieser Abbildung beispielhaft einige Werte dar. Das Attribut *UserId* identifiziert eindeutig den Benutzer. Der *Timestamp* speichert den genauen Zeitpunkt, wann die Benachrichtigung eingegangen ist. *AssetModification* gibt an, wie das *Asset* verändert wurde (Erstellen, Modifizieren, Löschen). Die beiden Attribute *TargetType* und *TargetId* stellen zusammen eine eindeutige Kennzeichnung des *Assets* innerhalb des Brokers dar. *TargetName* ist der vom Benutzer vergebene Name für das *Assets* und *GoalName* speichert den symbolischen Namen des *Goals*. In diesem Beispiel hat der Benutzer ein *Asset* aus dem

Asset	
Attributname	Attributwert
UserId	9z1ha7odl9k
Timestamp	20030601224848
AssetModification	Modified
TargetType	concept
TargetName	UML diagrams
TargetId	gnsj6ncryo7r
GoalName	ModifyConcept

Abbildung 5.4: Erhobene Daten bei Änderung eines *Assets*

Bereich *Concept*, das den Namen „UML diagrams“ trägt, verändert. Nachdem alle Daten zusammengestellt sind, werden sie an den *Tracking Server* gesendet. Nähere Informationen dazu finden sich in Kapitel 5.1.4.

5.1.2 Aufzeichnen von Änderungen in der Layout-Komponente

Die Benutzeroberfläche besteht, wie bereits in Kapitel 3 erwähnt, aus einer Menge von Vorlagen, die von unterliegenden Schichten mit Inhalt gefüllt werden. Jede Vorlage besitzt eine Menge von Verweisen (Hyperlinks) auf andere Vorlagen, die in

den jeweiligen Kontext passen und somit den Workflow vorgeben. Bereits die symbolischen Namen der Vorlagen, wie z.B. Verzeichnis, Dokument, Gruppe, Begriff oder Person, lassen erkennen, in welchem Bereich der Benutzer sich aufhält. Dies wird spezifiziert durch den genauen Namen der Seite, der z.B. create, modify, login oder edit lauten kann.

TemplateTracking entnimmt die Daten der Session-Komponente, da jede Änderung einer Vorlage durch die Session-Komponente angestoßen wird. Für jeden Benutzer wird eine eindeutige SessionId generiert, sodass sich jede Anforderung einer neuen Vorlage einem Benutzer zuordnen lässt. Dies ist insofern nützlich, da sich dadurch die Daten Art der Vorlage und Name der Vorlage einem Benutzer zuordnen lassen.

Template	
Attributname	Attributwert
UserId	9z1ha7odl9k
Timestamp	20030601224839
TemplateKind	concepts
TemplatePage	createSubConcept
TemplateAssetName	UML diagrams

Abbildung 5.5: Benutzte Daten beim Anfordern einer Vorlage

In dem Ereignis, das bei Wechsel der Vorlage generiert wird, sind wie im Beispiel in Abbildung 5.5 wieder die Attribute UserId und Timestamp enthalten. Zusätzlich werden die Attribute TemplateKind und TemplatePage erhoben, die, wie oben erwähnt, die Art und den Namen der Vorlage angeben. Das Attribut TemplateAssetName enthält nur einen Wert bei einem Asset, das der Benutzer selbst angelegt und benannt hat, wie z.B. Verzeichnisnamen, Dokumentnamen oder Begriffe.

5.1.3 Aufzeichnung von Suchanfragen

Ebenfalls in der Session-Komponente verankert ist das Aufzeichnen von Suchanfragen. Dabei wird der komplette vom Benutzer eingegebene Text,

Search	
Attributname	Attributwert
UserId	9z1ha7odl9k
Timestamp	20030601224839
SearchText	UML Java
SearchConcept	documents
SearchResultSize	4

Abbildung 5.6: Daten der Suchanfrage

die Art der zu suchenden Dokumente wie Dokumente, Personen oder Begriffe, sowie die üblichen Attribute `UserId` und `Timestamp`. Im obigen Beispiel in Abbildung 5.6 wird nach Dokumenten gesucht, welche die Zeichenfolgen „UML“ und „Java“ enthalten. Gefunden wurden vier Ergebnisse. Zusammen mit den Daten aus der Layout-Komponente könnte man nun feststellen, ob der Benutzer eines der Suchergebnisse auswählt oder ob er später wieder zum Suchergebnis zurückkehrt, weil er die gewünschte Information nicht gefunden hat.

5.1.4 Schnittstelle zwischen Broker und *Tracking Server*

Die Schnittstelle sollte folgende Eigenschaften erfüllen:

- Mögliche Erweiterungen der Datenquellen sollten ohne oder nur mit geringen Änderungen am Quellcode möglich sein.
- Aus Gründen der Performance sollte die Schnittstelle „schlank“ sein, d.h. es sollten möglichst einfache Datentypen statt komplexer Objekte übertragen werden.

Deswegen wurde die Schnittstelle so implementiert, dass die Klassen, die das Sammeln von Daten als Aufgabe haben, keine Kenntnis von der Übertragung der eigentlichen Daten zum *Tracking Server* haben müssen. Dazu erben die drei oben genannten Klassen, wie in Abbildung 5.7 dargestellt, von der Basisklasse `BasicTracking`. Diese benutzt das RMI-Protokoll [SUN01a] um eine Verbindung zu dem *Tracking Server* auf einem entfernten Rechner herzustellen. Mit RMI (*Remote Method Invocation*) stellt das JDK seit der Version 1.1 einen Mechanismus zur Verfügung, der es ermöglicht, Objekte auf einfache Weise im Netz zu verteilen und ihre Dienste anderen Arbeitsplätzen zur Verfügung zu stellen.

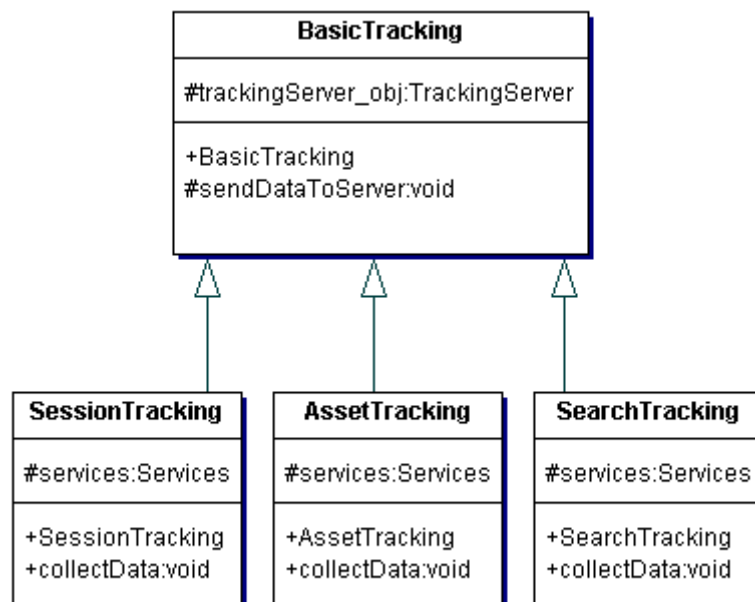


Abbildung 5.7: Kapselung der RMI-Funktionalität in `BasicTracking`

Der Konstruktor der Klasse `BasicTracking` stellt beim Aufruf eine Verbindung zum *Tracking Server* her. Falls der *Tracking Server* nicht läuft oder aus anderen Gründen eine Verbindung fehlschlägt, werden die Tracking Daten wahlweise in eine lokale Log-Datei geschrieben. Die drei erbenenden Klassen rufen zum Versenden der Daten die Funktion `sendDataToServer` auf. Die Deklaration der Funktion lautet:

```
sendDataToServer(  
    String eventType,  
    String userId,  
    long timestamp,  
    Vector attributeNames,  
    Vector attributeValues);
```

Neben dem `eventType`, der je nach Herkunft der Daten einen der Werte `Template`, `Asset` oder `Search` enthält, sind die `userId` sowie der `timestamp` bei jedem Aufruf obligatorisch. Die Anzahl der Attributnamen und -werte wurde variabel gewählt und hängt vom jeweiligen Typ des Ereignisses ab. Dies wurde bewusst so gewählt, um Erweiterungen ohne Änderungen der eigentlichen Schnittstelle zu ermöglichen.

5.2 Realisierung des *Tracking Servers*

Zum Zusammenfassen der drei Datenströme wurde ein eigenständiger Server implementiert. Der Start auf dem gleichen Rechner, auf dem der `INFOASSET BROKER` läuft, hätte aufgrund der Masse der zu verarbeitenden Daten zu einer deutlichen Verschlechterung der Antwortzeiten des Brokers geführt. Die Verwendung des RMI-Protokolls hingegen erlaubt die Anbindung des *Tracking Servers* an den Broker auf einem beliebigen Rechner.

Die Notwendigkeit eines *Tracking Servers* bestand aus folgenden Gründen:

Die Klassen im Broker, die das Sammeln der Daten realisieren, sind zeitkritisch. Eine Verarbeitung der Daten im Broker schied aus, da dies eine Beeinträchtigung der Antwortzeiten zur Folge gehabt hätte.

Ein weiterer Grund ist, dass sich in den drei Datenströmen größtenteils Ereignisse befinden, die durch dieselbe Benutzeraktion angestoßen wurden. Um also eine konsistente und vollständige Datenbasis zu erhalten, müssen zuerst alle Ereignisse, die auf dieselbe Benutzeraktion zurückgehen, zu einem Ereignis zusammengefasst werden. Ein Beispiel erläutert die Problematik: Der Benutzer ändert den Namen eines Dokuments und klickt zum Bestätigen der Änderung auf „Übernehmen“. Durch diesen Klick wird zum einen ein Wechsel der Vorlage ausgelöst, was ein Ereignis im Vorlagen-Datenstrom erzeugt. Zum anderen wird fast zur gleichen Zeit die Änderung eines *Assets* gemeldet, da der Benutzer den Namen eines Dokuments geändert hat. Dies wiederum zieht ein Ereignis im Inhalts-Datenstrom nach sich.

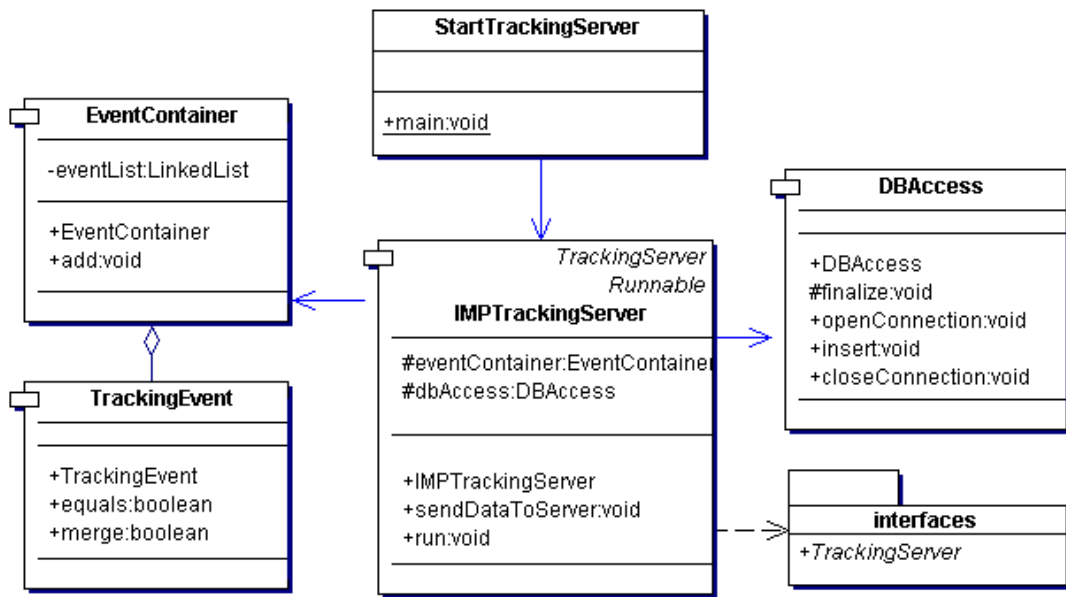


Abbildung 5.8: Klassendiagramm des *Tracking Servers*

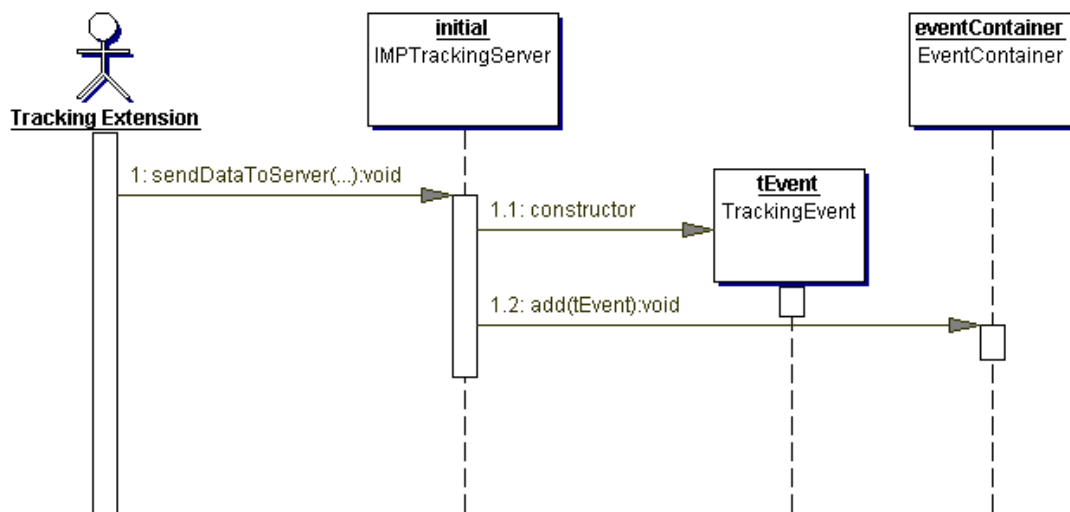


Abbildung 5.9: Sequenzdiagramm eines Aufrufs von `sendDataToServer`

5.2.1 Pufferung und Verarbeitung der Daten

Die Notwendigkeit, die Daten zusammenzufassen, erfordert eine vorübergehende Speicherung. Dazu wurde die Klasse `EventContainer` entworfen. Sie ermöglicht ein einfaches Hinzufügen von neuen Ereignissen, `TrackingEvent` genannt. Ein `TrackingEvent` besteht aus einem Konstruktor, der genau auf die

Schnittstelle zwischen Broker und *Tracking Server* zugeschnitten ist. Er übernimmt die Parameter `eventType`, `userId`, `timestamp` und die beiden Vektoren mit den Attributpaaren. Ein auf diese Weise initialisiertes `TrackingEvent` wird dem `EventContainer` hinzugefügt, wie in Abbildung 5.9 in einem Sequenzdiagramm dargestellt ist. Das Hinzufügen erfordert besondere Beachtung, da an dieser Stelle nach Ereignissen gesucht wird, die zusammengehören. Die Bedingungen zweier `TrackingEvents`, die dazu erfüllt sein müssen, sind:

- die gleiche `UserId`,
- nur wenige Zehntelsekunden Differenz im Zeitstempel und
- unterschiedliche Typen der Ereignisse, wie z.B. `Template` und `Asset`.

Sind alle diese Kriterien erfüllt, werden zwei Ereignisse zu einem zusammengefasst. Ein Beispiel für das Zusammenfassen zweier Ereignisse ist in Abbildung 5.12, und ein Sequenzdiagramm dieses Prozesses in Abbildung 5.10 dargestellt.

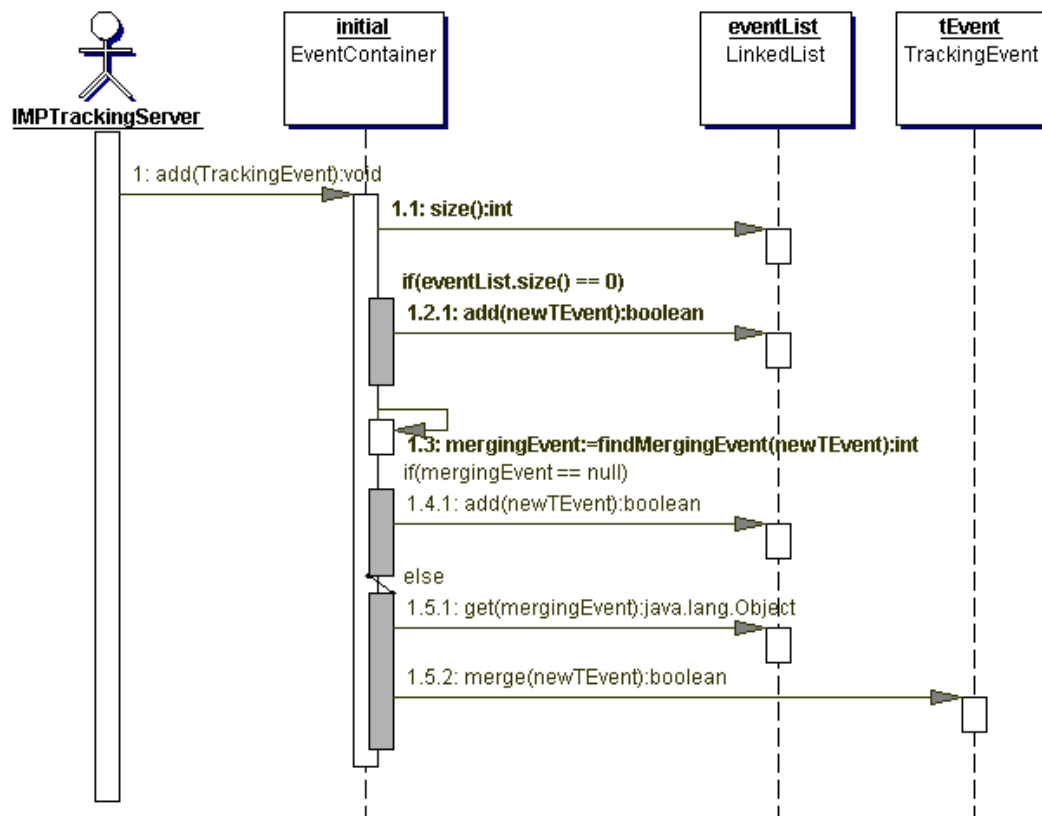


Abbildung 5.10: Zusammenfassen von Ereignissen

Der `EventContainer` dient, wie oben erwähnt, einer vorübergehenden Aufbewahrung der `TrackingEvents`. Die Aufbewahrung eines `TrackingEvents` endet zum einen, falls es sich bereits für die Dauer einer festgelegten Zeitspanne darin befindet und nicht mehr mit Ereignissen gerechnet

werden muss, die zusammengefügt werden könnten. Zum anderen kann es aus dem Container entnommen werden, falls es bereits aus allen möglichen Teilereignissen, d.h. aus Template-, Asset- und Search-Ereignis, zusammengesetzt ist. Dann ist das Ereignis komplett und kann in die Datenbank eingefügt werden. Zuständig dafür ist ein nebenläufiger Prozess in `IMPTrackingServer`, der, wie in Abbildung 5.11 dargestellt, nach einer festgelegten Zeitspanne aufwacht, die fertigen Ereignisse aus dem `EventContainer` entnimmt und an die Datenbankschnittstelle übergibt.

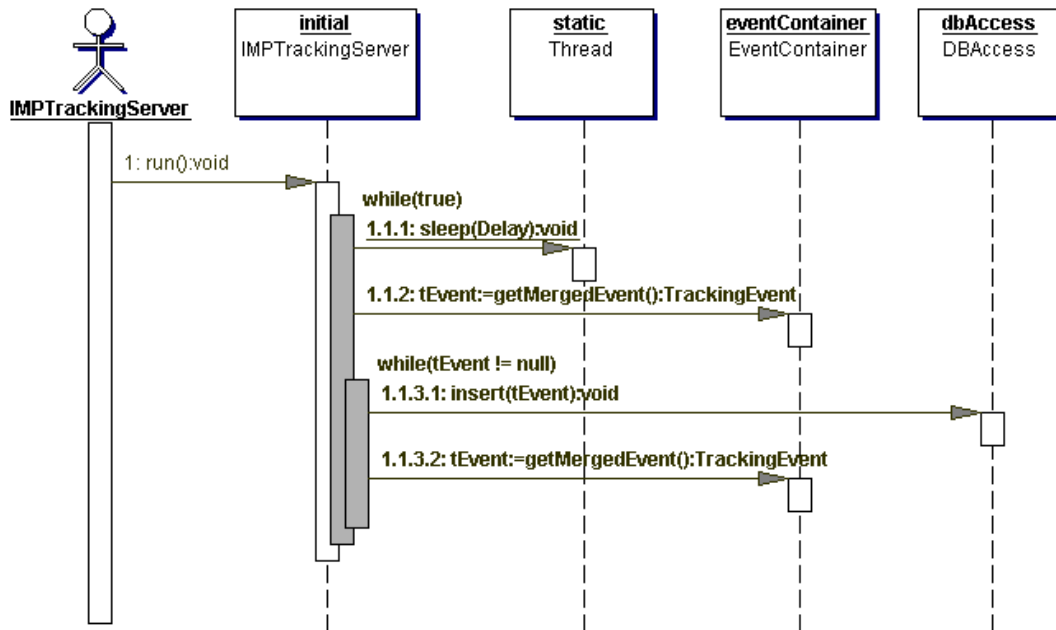


Abbildung 5.11: Sequenzdiagramm des nebenläufigen Prozesses

Zu beachten ist bei der Klasse `EventContainer` zusätzlich, dass durch den oben erwähnten nebenläufigen Prozess, der Ereignisse entnimmt und durch die neu eintreffenden Ereignisse, die dem `EventContainer` hinzugefügt werden, konkurrierende Zugriffe auf diese Klasse stattfinden können. Daher ist es notwendig, den Zugriff auf diese Klasse zu synchronisieren.

5.2.2 Die Datenbank Schnittstelle

Die Klasse `DBAccess` ist für Schnittstelle [SUN01b] zur Datenbank verantwortlich. Ein Aufruf der Methode `insert` überträgt die Daten der Klasse `TrackingEvent` in die jeweiligen Tabellen der Datenbank. Wie bereits in Kapitel 3.3 erwähnt, wird das *Sternschema* verwendet. Abbildung 5.13 zeigt ein solches Schema, angepasst an die Daten der *Tracking Extension*.

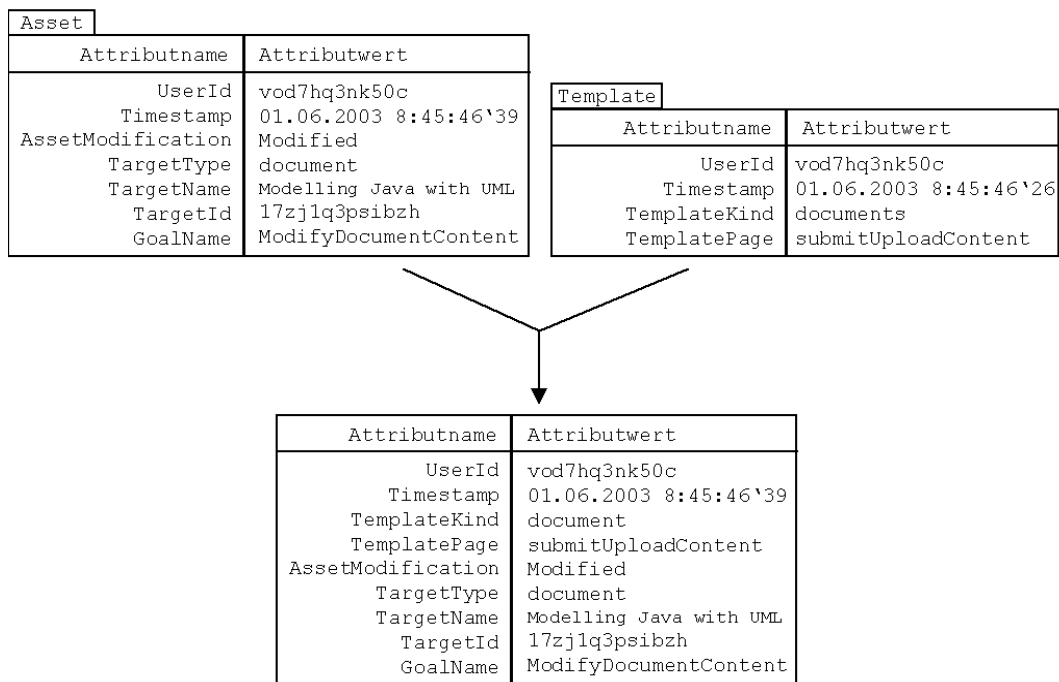


Abbildung 5.12: Zusammenfassen von zwei Ereignissen

5.3 Ergebnisse

Ein Beispiel soll an dieser Stelle die umfangreichen Möglichkeiten zur Analyse durch das Content Warehouse-Modell veranschaulichen.

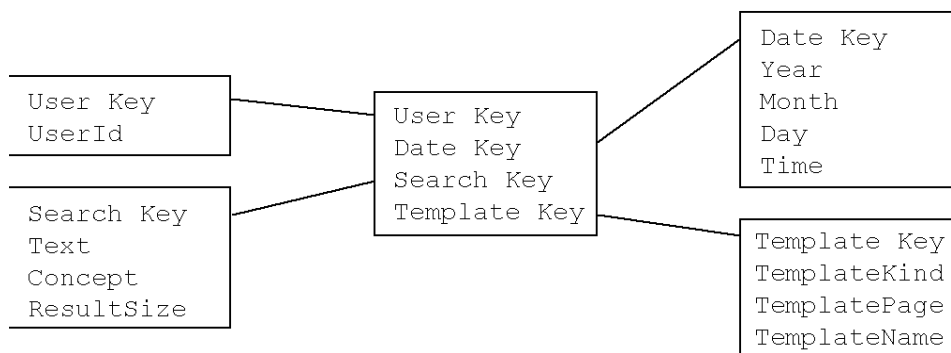


Abbildung 5.13: Datenbankschema in Form eines Sternschemas

Die folgenden Grafiken wurden mit der PivotTable-Funktion von MICROSOFT EXCEL 2000 erstellt. Diese Funktion stellt neben einer Anbindung an externe Datenbanken, wie die für diese Arbeit benutzte MYSQL-Datenbank [MYSQL03], auch elementare Data Warehouse-Funktionalität wie *Drill Down* und *Roll Up* (vgl. Kapitel 2.4.2) zur Verfügung. In diesem Beispiel zeigt Abbildung 5.14 die Anzahl

der Zugriffe auf Vorlagen innerhalb des INFOASSET BROKERS für den Juni 2003. In der folgenden Abbildung 5.15 wurde die Auswahl verfeinert (*Drill Down*) und die Zugriffe sind nun nach Tagen aufgeschlüsselt. Offensichtlich wurde der Broker am 1., 7. und 9. Juni benutzt. Um die Daten weiter aufzuschlüsseln werden in Abbildung 5.16 zusätzlich die UserIds pro Tag dargestellt.

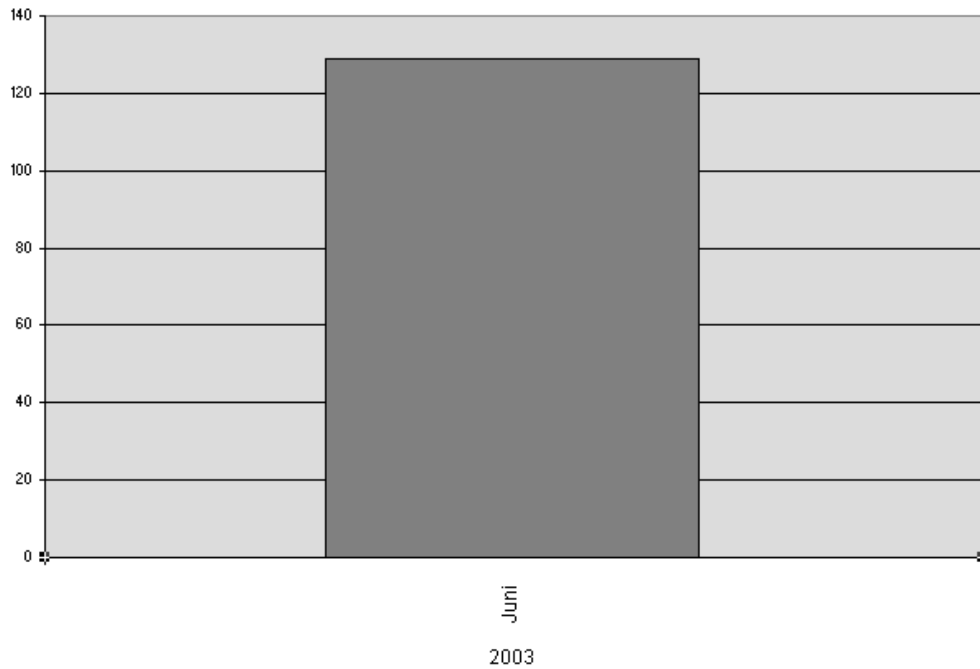


Abbildung 5.14: Summe der aufgesuchten Seiten im Juni 2003

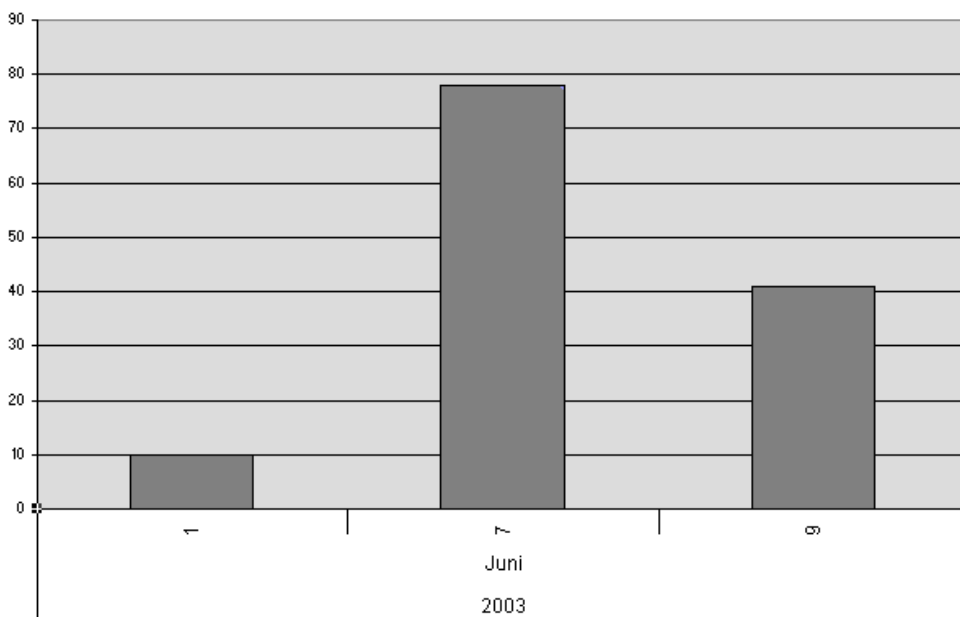


Abbildung 5.15: Besuchte Seiten im Juni abhängig vom Tag

Man erkennt, dass der Benutzer mit der eindeutigen Bezeichnung ‚vod7hq3nk50c‘ am 7.Juni 2003 ca. 58 Seiten des Brokers angeschaut hat. Eine Darstellung der benutzten Vorlagen dieses Benutzers am 7. Juni zeigt schließlich Abbildung 5.17.

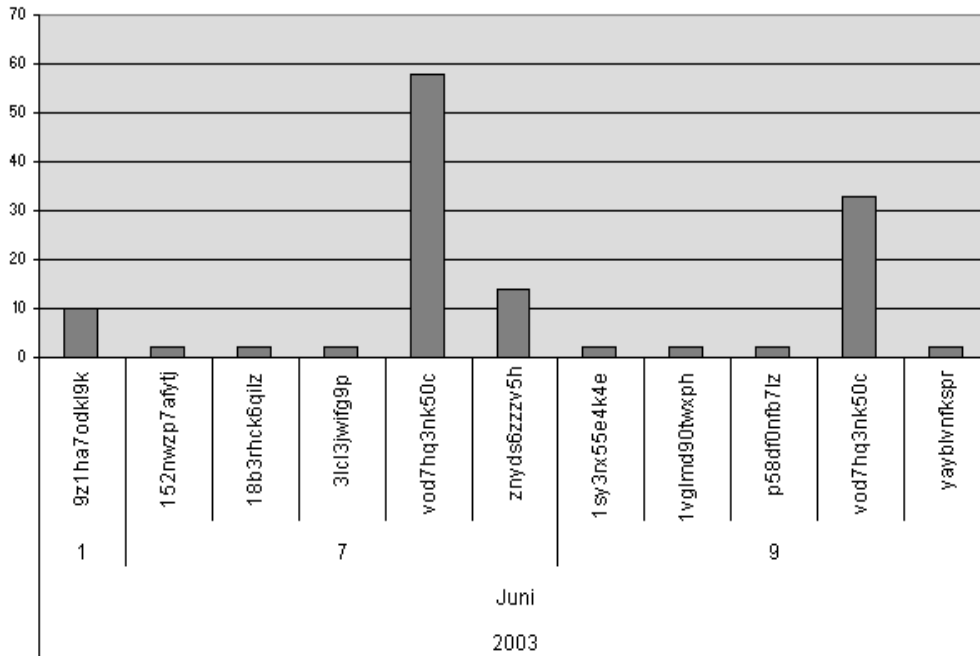


Abbildung 5.16: Besuchte Seiten im Juni, aufgeschlüsselt nach Benutzer

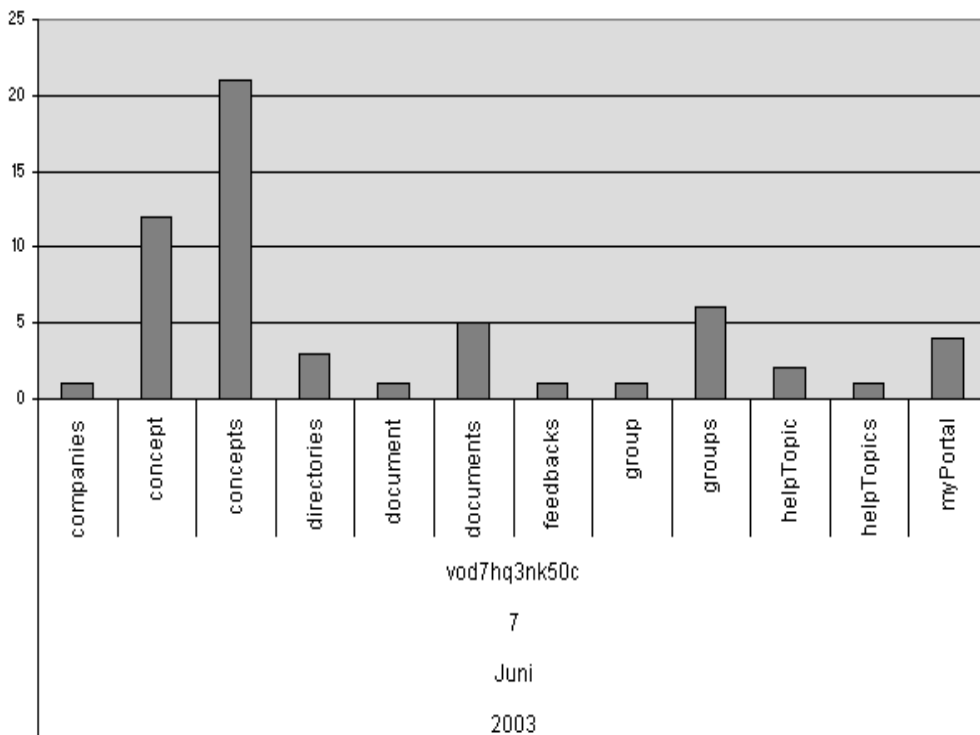


Abbildung 5.17: Besuchte Vorlagen-Typen für einen Benutzer

6 Zusammenfassung und Ausblick

In diesem Kapitel wird zunächst eine Zusammenfassung dieser Arbeit gegeben. Als Abschluss folgt ein Ausblick auf mögliche Erweiterungen der *Tracking Extension*.

6.1 Zusammenfassung

Zur Speicherung und Bereitstellung großer Datenmengen sind Wissensmanagement- und Content Management-Systeme sehr gut geeignet. Für Benutzer ist es jedoch oft nicht leicht, gewünschte Informationen schnell zu erreichen und sich in der Menge an Informationen zurechtzufinden. Intuitiv zu bedienende Benutzeroberflächen können den Anwender dabei unterstützen. Objektive Maßstäbe, wie intuitiv eine Benutzeroberfläche zu bedienen ist, gibt es aber nicht. Eine mögliche Lösung ist das Sammeln und Auswerten von Benutzeraktionen. Durch Erheben dieser Daten im Broker und der Überführung in ein Content Warehouse-Modell kann man z.B. folgende Fragestellungen beantworten:

- Hat der Benutzer mit seiner Suchanfrage die gewünschte Information gefunden, verfeinert er seine Suchanfrage oder verlässt er die Seite, weil die Suche nicht erfolgreich war?
- Wie viele Seiten, beginnend bei der Einstiegsseite, hat sich der Benutzer angeschaut, bis er die gewünschte Information gefunden hat?
- Was sind die häufigsten Suchanfragen und wie viele relevante Suchergebnisse liefern sie?
- Gibt es eine Korrelation zwischen Benutzergruppen und häufig besuchten Seiten?
- Welche Seiten werden oft und welche selten besucht?
- Wie viele Anwender nutzen den Broker? (Auswertung nach Minuten, Stunden, Tagen, Monaten, etc)

Die Beantwortung dieser Fragen durch das Content Warehouse-Modell kann unter anderem dabei helfen, die Bedienoberfläche zu optimieren oder statistische Aussagen über die Nutzung des Brokers zu erstellen.

Im Rahmen dieser Arbeit wurde also eine Schnittstelle zwischen dem INFOASSET BROKER und Data Warehouse-Software entworfen und implementiert. Diese Schnittstelle erlaubt es, Daten über die Nutzung des Brokers in einem Data Warehouse zu analysieren. Dazu wurde ein Content Warehouse-Modell entwickelt, das die Nutzungsdaten in ein für das Warehouse verständliches Datenbankschema bringt.

6.2 Ausblick

Zum Abschluss wird ein Ausblick auf mögliche Weiterentwicklungen gegeben. Wünschenswert wäre eine universelle Konfiguration sowohl der Tracking Extension, als auch des Tracking Servers durch externe Text- oder XML-Dateien. Die Konfiguration könnte folgende Bereiche umfassen:

1. Auswahl der Daten, die innerhalb des Brokers erhoben werden sollen.
2. Freie Zuordnung der Daten zu Fakten- und Dimensionstabellen im *Tracking Server* und Anlegen eines Datenbankschemas.
3. Freie Auswahl einer SQL-Datenbank zur Speicherung der Daten.
4. Erweiterung der *Alerting Extension* um eine Möglichkeit, über *Goals* benachrichtigt zu werden, auf die nur lesend zugegriffen wurde. Zurzeit werden nur die drei Änderungen Erstellen, Modifizieren und Löschen berücksichtigt. Durch Einrichten von *Goals*, die einen Lesezugriff melden, ist diese Erweiterung problemlos möglich. Die *Tracking Extension* muss für diese Erweiterung nicht modifiziert werden.
5. Erweiterung der *Alerting Extension* zur Unterstützung sogenannter *Multi-Asset-Views*, d.h. der Darstellung von mehreren *Assets* in einer Vorlage. Dazu müssten sowohl *Alerting Extension* als auch *Tracking Extension* um die Verarbeitung von *Goal*-Listen erweitert werden.

Literaturverzeichnis

- [Anah97] Sam Anahory, Dennis Murray: *Data Warehousing in the Real World: A Practical Guide for Building Decision Support Systems*, Addison Wesley, 1997
- [Conn01] T.M. Connolly, C.E. Begg: *Database systems: a practical approach to design, implementation and management*; 3rd. Edition, Addison-Wesley, 2001
- [Ecke98] Bruce Eckel: *Thinking in JAVA, The Definitive Introduction To Object-Oriented Programming In The Language Of The World-Wide Web*, Prentice Hall, 1998
- [EURI03] Forschungsprojekt des Arbeitsbereichs Softwaresysteme der Technischen Universität Hamburg-Harburg, *European Reference Center for Intermodal Freight Transport*, <http://www.eurift.org>, 2003
- [Fuhr00] Dirk Fuhrmann: *Konzeption eines Data Warehouse Systems: Prototypische Realisierung einer Anwendung*, Diplomarbeit, Universität Dortmund, 2000
- [Han00] Jiawei Han, Micheline Kamber: *Data Mining – Concepts and Techniques*, Morgan Kaufmann Publishers, 2000
- [INF01] INFOASSET AG, *The infoAsset Broker – Technical White Paper*, 2001
- [INF02a] INFOASSET AG, *infoAsset Broker – Administrationshandbuch für Windows und Unix-Plattformen*, 2002
- [INF02b] INFOASSET AG, *infoAsset Broker – Benutzerhandbuch*, 2002
- [INF03] infoAsset, *infoAsset Broker Homepage*, <http://www.infoasset.de>, 2003
- [Inmo96] W.H. Inmon: *Building the Data Warehouse*, 2nd Edition, Wiley, 1996
- [Kimb96] Ralph Kimball: *The Data Warehouse Toolkit, Practical Techniques for Building Dimensional Data Warehouses*, John Wiley & Sons, 1996
- [Muck98] Harry Mucksch, Wolfgang Behme (HRSG.): *Das Data Warehouse-Konzept, Architektur – Datenmodelle –Anwendungen; mit Erfahrungsberichten*, 3. Auflage, Wiesbaden: Gabler, 1998
- [MYSQ03] MYSQL, *Open Source Database*, <http://www.mysql.com>, 2003
- [Roth01] Gunter Rothfuss, Christian Ried: *Content Management mit XML*, Springer, 2001

- [SAP02] SAP AG, *Data Warehousing with MySAP Business Intelligence*, White Paper, 2002
- [SUN01a] SUN MICROSYSTEMS, *Java Remote Method Invocation (RMI)*, <http://java.sun.com/products/jdk/rmi/index.html>, 2001
- [SUN01b] SUN MICROSYSTEMS, *JDBC Data Access API*, <http://java.sun.com/products/jdbc/>, 2001
- [Vitt02] Elizabeth Vitt, Michael Luckevich, Stacia Misner: *Business Intelligence*, Microsoft Press, 2002
- [Voss94] Gottfried Vossen, *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme*, 2. Auflage, Addison-Wesley, 1994
- [Wegn02] Holm Wegner. *Analyse und objektorientierter Entwurf eines integrierten Portalsystems für das Wissensmanagement*, Dissertation, Technische Universität Hamburg-Harburg, Hamburg, 2002.