

Diplomarbeit

Michael Maretzke

**Evaluation einer Suchmaschine und Integration in eine
existierende Internet-Shoppingapplikation**

Erstprüfer: Prof. Dr. Lore Kern-Bausch

Abgabesemester: Sommersemester 2000

Aufgabenstellende Firma: Conrad.com AG
Klaus-Conrad-Straße 1
92240 Hirschau

1.	Einleitung.....	1
1.1.	Die Firma "Conrad Electronic GmbH".....	1
1.2.	Die Firma "Conrad.com AG".....	2
1.3.	Das Thema der Diplomarbeit.....	2
1.4.	Die Ziele der Diplomarbeit.....	4
1.5.	Die Struktur der Diplomarbeit.....	4
2.	Ist-System Analyse.....	5
2.1.	Grober Gesamtüberblick.....	5
2.2.	Hardwareüberblick.....	8
2.3.	Softwareüberblick.....	9
2.4.	Verwendete Datenbank.....	10
2.5.	Übergreifende Problempunkte Hardware / Software / Datenbank.....	12
2.6.	Analyse der Suche im bestehenden System.....	14
2.6.1	Ergebnisse der Black Box* Analyse.....	15
2.6.2	Ergebnisse der White Box Analyse.....	18
2.6.3	Schwachpunkte des existierenden Suchmodells.....	22
3.	Ziele des Soll-Modells.....	30
3.1.	Schnelle Lösung.....	30
3.2.	Skalierbare Lösung.....	31
3.3.	Leicht wartbare Lösung.....	31
3.4.	Technologisch zukunftsorientierte Lösung.....	32
3.5.	Funktionsorientierte Lösung.....	32
3.5.1	Minimalfunktionalität.....	33
3.5.2	gewünschte Funktionalität.....	33
4.	Suchmaschinen im Allgemeinen.....	35
4.1.	Überblick über Suchmaschinen.....	35
4.1.1	Geschichte und Einsatzbereiche.....	35
4.1.2	Grundlegende Suchkonzepte auf einem Computer.....	36
4.1.3	Arten von Suchmaschinen.....	38
4.1.4	Generelle Aufgaben von Suchmaschinen.....	40
4.2.	Überblick über die Aufgaben und Einsatzgebiete von Agenten.....	40
4.3.	Überblick über verschiedene Methoden und Techniken von Suchmaschinen.....	41
4.3.1	Boolean Retrieval Methoden.....	41
4.3.2	Vector Space Retrieval Methoden.....	43
4.3.3	Probabilistic Retrieval Methoden.....	46
4.3.4	Extended Boolean Retrieval Methoden.....	49
4.3.5	Expertensystembasierte Retrieval Methoden.....	51
4.3.6	Verwendung von fortgeschrittenen mathematischen Modellen.....	52
4.4.	Ein exemplarisches Beispiel: http://www.webcrawler.com	56
4.4.1	Allgemeiner Überblick.....	57
4.4.2	Architektur* WebCrawler.....	57
4.4.3	SuchEngine.....	58
4.4.4	Agents.....	59
4.4.5	Datenbank.....	59
4.4.6	QueryEngine.....	59
4.4.7	Fazit.....	60
5.	Mögliche Lösungswege.....	61
5.1.	Nutzdaten- und Suchdatenbestand in einer gemeinsamen Datenbank.....	62
5.2.	Nutzdaten- und Suchdatenbestand getrennt in unterschiedlichen Systemen.....	63
5.2.1	Verwendung kommerzieller Systeme.....	64
5.2.2	Eigenentwicklung.....	95
6.	Integration des bestmöglichen Systems.....	99
6.1.	Abschließende Bewertung der evaluierten Systeme.....	99
6.1.1	Software AG – Tamino.....	99
6.1.2	Autonomy – Knowledge Server.....	100
6.1.3	Eigenentwicklung.....	101

6.1.4	Entscheidungsmatrix.....	101
6.1.5	Entscheidung für ein System.....	103
6.2.	Integrationsschritte ins existierende System	104
6.3.	Jetziger Status – prototypische Implementierung.....	104
6.3.1	Datenmodelloptimierung.....	104
6.3.2	prototypische PERL Applikation.....	107
7.	Ausblicke auf neuere / alternative Techniken.....	111
7.1.	Einsatz von Oracle Datenbank in Verbindung mit Parallel Query Execution.....	111
7.2.	Einsatz von SUN-Clustern* / Oracle Parallel Server	115
7.2.1	SUN-Cluster*	115
7.2.2	Oracle Parallel Server	116
7.3.	Verwendung phonetischer Algorithmen für eine bessere Treffermenge.....	117
7.3.1	Algorithmus "Soundex"	118
7.3.2	Algorithmus "phonet"	119
7.3.3	Änderungen am Suchprototypen	120
8.	Schlusswort und Erstellungserklärung	123
8.1.	Schlusswort.....	123
8.2.	Erstellungserklärung.....	124
9.	Glossar	125
10.	Literaturverzeichnis.....	133
11.	Verzeichnisse.....	138
11.1.	Abbildungsverzeichnis.....	138
11.2.	Tabellenverzeichnis	138
11.3.	Ausschnittverzeichnis.....	139

1. Einleitung

Bevor auf das Thema der Diplomarbeit genauer eingegangen wird, wird die Firma und das nähere Umfeld der Diplomarbeit vorgestellt und erläutert. Daran anschließend werden neben den Zielen auch die Grundstruktur der Diplomarbeit vorgestellt.

1.1. Die Firma "Conrad Electronic GmbH"

"Wer hier nichts findet, ist kein Mann !" ist ein Slogan, der in einer Marketingaktion auf einem Plakat unter dem Conrad Electronic Katalog steht. Im Hauptkatalog der Conrad Electronic findet sich so ziemlich alles von Alarmanlagen über Bausätzen für Kleinelektronikanlagen, Computer, Diskobeleuchtung, Elektronikbauteile, Modellbau, ... bis hin zu Zangen für den Hobbyelektroniker. Das Produktsortiment umfasst ca. 65.000 Produkte mit eindeutig steigender Tendenz. Die Conrad Electronic ist ein Familienunternehmen und blickt auf mehr als 75 Jahre Erfahrung im Versandhandel im Bereich "Electronics" zurück. Das Unternehmen wird in der 4. Generation geführt und wurde im Jahre 1923 gegründet. Der Firmensitz ist aus kriegshistorischen Gründen die Stadt Hirschau in der Oberpfalz ("Stadt der weißen Erde" – Hirschau ist auch bekannt durch den Kaolin-Berg direkt am Ortseingang). Die Firmengruppe Conrad Electronic umfasst derzeit etwa 2.500 Mitarbeiter in Europa und Asien. Täglich werden über das eigene Logistikzentrum in Wernberg/Köblitz ca. 25.000 Pakete und Päckchen versendet. Im neuen Logistikzentrum in Wernberg werden pro Jahr ca. 130.000 Paletten mit Waren aus aller Welt bearbeitet. Im Jahr 1999 machte die gesamte Conrad Gruppe einen Gesamtumsatz von einer Milliarde DM. Conrad Electronic ist mittlerweile Marktführer und Europas größter Electronic Spezialversender. Die Hauptvertriebswege sind noch der Katalog mit seinen saisonalen Nebenkatalogen und die 18 Verkaufsfilialen. Die Väter der Conrad Electronic Gruppe sind von je her Innovationen aufgeschlossen. "Heute ist für die Familie Conrad schon immer morgen gewesen." Daher führte Klaus Conrad bereits im Jahre 1976 eine EDV-Anlage ein. Damit wurde ein Informationssystem für Absatz, Beschaffung, Finanzierung, Investition, Personal und Kontrolle und eine Kundenkartei mit 20.000 Adressen realisiert. Zum damaligen Zeitpunkt eine gewagte, innovative und, wie sich herausstellen sollte, richtige Entscheidung. Die Firmengruppe Conrad Electronic will durch weiteres Wachstum, Neukundenakquise und Ausbau des Produktsortimentes die Marktführerschaft sichern bzw. weiter ausbauen. Für die weitere Kundengewinnung sind auch neue Vertriebswege

wichtig. Das Internet wurde von der Conrad Gruppe bereits im Jahre 1997 erfolgreich erobert. Hier kommt nun die Tochterfirma "CoMedia GmbH" bzw. die daraus erwachsene "Conrad.com AG" ins Spiel [CON98, CIN97].

1.2. Die Firma "Conrad.com AG"

Die Conrad.com AG existiert seit dem April 2000 und zeichnet sich verantwortlich für alle Aspekte des Online-Handels der Conrad Electronic GmbH. Bis zum April 2000 wurde E-Business* von der 1997 gegründeten Conrad-Tochter CoMedia GmbH abgewickelt. Seit der Gründung 1997 entwickelte sich Conrad zu einem führenden europäischen Online-Anbieter für Elektronik und Technik im Consumerbereich. Bereits fünf Monate nach dem Start des ersten Conrad Online-Shops*, www.conrad.de, bestellten täglich bis zu 250 Kunden über das Internet. Heute laufen täglich rund 3000 Bestellungen via Internet in der Zentrale in Hirschau ein. Im Jahr 1999 verzeichnete man bei der Conrad.com AG ca. sechs Millionen Besucher auf allen Ländersites. Die Conrad.com AG betreibt mittlerweile neben dem Hauptshop www.conrad.de noch weitere länderspezifische Shops (.at, .fr, .nl) und viele themenabhängige Shops (www.modellbau.conrad.de, www.topshop.conrad.de, www.conradkom.de, ...). Neben den Conrad Shops wird auch der Völkner Shop von der Conrad.com AG betrieben. Völkner wurde von der Conrad Gruppe aufgekauft und ist somit Teil der Firmenfamilie. Im Jahr 1999 ist allein durch den Vertriebskanal Internet ein Gesamtumsatz von 55 Millionen Mark erwirtschaftet worden. Nur im Januar 2000 registrierte man bei der Conrad.com AG allein für die Domain www.conrad.com etwa eine Millionen Visits* und nahezu 13 Millionen Seitenabrufe*. Der Erfolg der Plattform reflektiert sich auch durch zahlreiche Preise und Auszeichnungen, wie der "Online Today Award" (1999), der "IT Business Award Platin" (1998) oder die Auszeichnung "eShopping-Company of the Year" (1999) um nur einige zu nennen. Das auserkorene Ziel der Conrad.com AG ist Wachstum. Das derzeitige Kundenprofil ist dominiert durch einen 97%-igen Männeranteil. Durch Änderung der Zielgruppen (weibliche Kunden, andere soziale Schichten) sollen weitere Neukunden generiert werden. Derzeit ist die Zuwachsrate mit 30.000 Neukunden pro Quartal angesiedelt. Soweit zur Firma [CCM00, CNM99].

1.3. Das Thema der Diplomarbeit

Das Thema der Diplomarbeit ? "Evaluation* einer Suchmaschine* und Integration in eine existierende Internet-Shoppingapplikation". Gerade das Thema "Suchen & Finden" ist

in einer Shoppingapplikation* von zentraler Bedeutung. Ein Surfer kommt mit einem Wunsch im Internetshop an und möchte ein Produkt kaufen, das seine Wünsche erfüllt – sein Zielprodukt. Das Ziel der Shoppingapplikation ist es Umsatz durch Verkauf zu generieren. Der Wunsch bzw. die Wünsche des Surfers und die Angebote des Shops müssen irgendwie zur Überdeckung kommen. Hier kommt die Suche ins Spiel. Je besser und cleverer die Suche ist, desto eher findet der Surfer sein Zielprodukt. Je schneller die Suche zum Ziel führt, desto eher wird der Surfer zum Käufer. Der Kunde und der Shop haben dann die jeweiligen Aufgaben erfüllt. Dauert die Suche zu lange oder liefert unbefriedigende Ergebnisse, so wird der Surfer den Kaufvorgang aus eigenem finanziellen Interesse – surfen kostet in Deutschland ja immer noch Geld – abbrechen. Die aktuell implementierte Applikation belastet gerade beim Suchprozess den Datenbank- und den Applikationsrechner über Gebühr. Warum dies so ist soll eine genauere Analyse der verwendeten Konzepte ergeben. Für diesen Misstand müssen Auswege skizziert und geprüft werden. Dafür kommen auch kommerzielle Suchprodukte in die engere Auswahl und werden systematisch untersucht. Neben der Wahl des Produktes, um die Suchproblematik zu entschärfen, müssen Wege für die Integration des Produktes in die existierende Shoppingapplikation gefunden werden. Vielleicht ist ja auch die Optimierung der Applikation eine Lösung. Dann müssten Wege für die Optimierung und die anschließende Integration bei gleichbleibendem Leistungsumfang erdacht werden. Als problematisches Beiwerk zur Diplomarbeit bleibt noch zu erwähnen, dass die komplette Shoppingapplikation extern programmiert wurde. Zur Applikation gibt es keine Spezifikation, keine Dokumentation, keinen oder nur schlecht kommentierten Quellcode und zudem ist der Wille des Dienstleisters die "Karten offen auf den Tisch zu legen" nicht sehr groß. Es existiert eine Abhängigkeit, die ich bisher nicht gekannt habe. Als Kunde muss man als Bittsteller dem Dienstleister gegenüber auftreten. Zu dem Unwillen des Dienstleisters kommen noch einige Probleme seitens der Conrad.com AG. In der Conrad.com AG ist bis jetzt noch keinerlei IT-Know-How vertreten. Der Dienstleister ist in der Lage gewesen, technisch nicht einwandfreie Konzepte und Realisierungen für teures Geld zu verkaufen. Projekte wurden ohne Kontrollorgane oder Strukturierung durchgeführt. Ein Scheitern oder ein Gelingen eines Projektes konnte zu keiner Zeit festgestellt werden. Soviel zum Umfeld der Diplomarbeit.

1.4. Die Ziele der Diplomarbeit

- klar umrissenes und eingegrenztes Themengebiet
- Evaluation möglicher Lösungsansätze
- prototypische Implementierung* des Systems mit den meisten Vorteilen
- für Conrad.com AG verwendbares (in der bestehenden Systemplattform*) bzw. richtungsweisendes (für eine neue Systemplattform) Ergebnis

1.5. Die Struktur der Diplomarbeit

Der Aufbau der Diplomarbeit orientiert sich stark an der Herangehensweise an diese Aufgabe. Zunächst wird eine Ist-System* Analyse durchgeführt, um ein Bild der Probleme und Konzepte in der aktuellen Implementierung entstehen zu lassen. Dabei versucht man von einem groben Überblick des Gesamtsystems bis hin zur Suchapplikation – dem eigentlichen Thema – zu verfeinern. Nach der Eingrenzung und Beschreibung der Problempunkte in der derzeitig realisierten Shoppingapplikation wird anhand eigener Aspekte und auch Benutzeranforderungen das Soll-Modell* definiert. Das Soll-Modell stellt die optimale Lösung dar – der Maßstab für die Probanden. Die Anforderungen an das Soll-Modell leiten sich im Wesentlichen von den Schwächen des Ist-Modells ab. Nach der Darstellung des Zieles für die Suche, werden Suchmaschinen und deren Technologien im Allgemeinen vorgestellt. Dabei werden die verschiedenen Einsatzbereiche und Aufgaben im Einzelnen beleuchtet und auch auf technische Aspekte einzelner Suchmethoden eingegangen. Abschließen wird das Kapitel ein Beispiel einer Internetsuchmaschine. Nach der Darstellung der Technik wird wieder das eigentliche Problem fokussiert. Mögliche Lösungswege für die Erfüllung des Soll-Modells und die Eliminierung der Schwächen der implementierten Lösung sollen aufgefunden und evaluiert werden. Kandidaten werden hierbei der Informationsserver "Tamino" von der Software AG und der "Knowledge Server" von Autonomy sein. Auch die Möglichkeit, die bestehende Applikation zu optimieren wird als Lösungsmöglichkeit genauer evaluiert. Daran anschließend werden Integrationswege der Lösung, die die meisten Vorteile für das Gesamtsystem bietet, aufgezeigt. Hierfür wird eine abschließende Bewertung der möglichen Lösungskandidaten durchgeführt. Daran anschließend werden noch alternative Lösungsmöglichkeiten vorgestellt, die aus verschiedensten Gründen nicht genauer evaluiert worden sind. Ganz zum Schluss der Diplomarbeit werde ich noch "Danke" sagen und die Erstellungserklärung anfügen.

2. Ist-System Analyse

Das bestehende System wird eingehender untersucht und beschrieben. Ausgehend vom allgemeinen Teil wird anschließend auf die Hauptkomponente der Diplomarbeit – die Suche – verfeinert. Das System wird wegen der fehlenden Dokumentation des Dienstleisters bis auf wenige Ausnahmen als eine Black Box* betrachtet.

Die Durchführung des Projektes "WWW-Auftritt" des Dienstleisters wurde von Seite Conrad.com AG ohne tiefe IT-Kenntnisse beauftragt und auch durchgeführt. Aus diesem Grunde liegen weder Pflichtenheft*, Spezifikationen*, Abnahmen*, Dokumente, Sourcen* und ähnliche Projektdokumentationen bei Conrad.com AG im Hause vor. Die aktuelle Web-Präsenz als Projekt betrachtet, zeigt die Gefahren auf, wenn man sich als Kunde auf "Gedeih und Verderb" einem Dienstleister ausliefert. Besonders schlimm ist die Tatsache, dass der durchführende Dienstleister mit der selbst erstellten Applikation starke Performanz- und Verfügbarkeitsprobleme hat. Eigentlich nötige Marketingaktionen können wegen einer potentiellen Überlastung des Systems und damit verbundenen Serverausfällen* nicht durchgeführt werden. Um das System zu stabilisieren werden ständig Änderungen am System durchgeführt und eine nachträgliche Verschlinkung der Applikation versucht.

Während der Erstellung der Diplomarbeit wird für aktuelle und zukünftige Projekte ein fundamentales IT-Know-How in Form eines IT-Teams aufgebaut. Eine der ersten Aufgaben des IT-Teams wird die Projektbegleitung bei der Konzeption und Realisierung einer neuen Shoppingapplikation sein.

2.1. Grober Gesamtüberblick

Hauptbestandteil der ganzen Shopping-Landschaft der Conrad.com AG ist der deutsche Hauptshop "www.conrad.de". Dort sind alle Produkte des gesamten Printkatalogs über das Internet zugänglich. Um möglichst viele Interessen der einzelnen potentiellen Kunden anzusprechen, wurden zudem einige Spezialshops installiert. Die Spezialshops heißen im Sprachgebrauch der Conrad.com AG "Advanced Small Shops" (Ass*). Diese Spezialshops basieren im Wesentlichen auf der Datenbasis des Hauptshops. Für die einzelnen Themenshops wurde lediglich das Warenangebot auf die jeweilige Thematik begrenzt.

Maschine	URL	Bemerkung
demdwu88	www.conrad.de	deutscher Hauptshop mit einer Abbildung des gesamten verfügbaren Produktsortiment
	www.conrad-electronic.com	s.o.
	www.digitalkamera.conrad.de	deutscher Themenshop mit Schwerpunkt auf Digitalkameras
demdwu92	www.computer.conrad.de	deutscher Themenshop mit Schwerpunkt auf Computer und Computerzubehör
	www.entertainment.conrad.de	deutscher Themenshop mit Schwerpunkt auf Unterhaltungselektronik, Audio, Video, DVD, Sound & Light
	www.topshop.conrad.de	deutscher Themenshop mit Schwerpunkt auf Restposten, Sonderangeboten, Neuheiten
	www.profi.conrad.de	deutscher Geschäftskundenshop mit Fokus auf "Business-to-Business"
	www.conrad.at	österreichischer Hauptshop mit einer Abbildung des Produktsortiments für Österreich; das Sortiment ist das selbe, wie in Deutschland; die Texte werden aus dem deutschen Hauptshop übernommen
	www.conrad.ch	Schweizer Hauptshop mit einer Abbildung des Produktsortiments für die Schweiz; das Sortiment ist das selbe, wie in Deutschland; die Texte werden aus dem deutschen Hauptshop übernommen
	www.conrad.nl	niederländischer Hauptshop mit einer Abbildung des länderspezifischen Produktsortiments für Niederlande; die Produkttexte sind hier niederländisch
	www.conrad.fr	französischer Hauptshop mit einer Abbildung des länderspezifischen Produktsortiments für Frankreich; das Produktsortiment ist um einige Produkte erweitert; die Funktionalität des Shops selber unterscheidet sich vom deutschen Hauptshop; die Produkttexte sind hier nicht die deutschen
	www.produktinfo.conrad.de	ftp-Zugang zu den Produktbeschreibungen und technischen Spezifikationen; die URL wird nur für die Verlinkung von den jeweiligen Shops aus verwendet
demdwu94	www.microsoft.conrad.de	deutscher Themenshop mit Hauptschwerpunkt auf Microsoft Produkten
	www.kommunikation.conrad.de	deutscher Themenshop mit Hauptschwerpunkt auf Kommunikation, Handys, Funkgeräten
	www.modellbau.conrad.de	deutscher Themenshop mit Schwerpunkt auf Modellbau, RC-Modelle, Zubehör
...

Tabelle 2-1 Übersicht über die Shops der Conrad.com AG

Die Tabelle 2-1 ist bei weitem nicht komplett. Internen Informationen zufolge sind im Auftrag der Conrad.com AG über 200 URLs belegt. Die ausländischen Shops nehmen hier z.T. Sonderrollen ein. Zunächst unterscheiden diese sich vom eigentlichen Hauptshop durch die Sprache und gerade in Frankreich, wo Conrad Electronic ein eigenes Logistikcenter betreibt, durch ein länderspezifisches Produktangebot. Funktionell unterscheiden sich die einzelnen Themenshops nicht und lassen sich alle durch den Hauptshop abbilden. Problematischer sind hier wiederum die ausländischen Shops. In Frankreich beispielsweise ist eine Bezahlung mit Kreditkarte möglich, in den

anderen Ländern ist diese Zahlungsform nicht vorgesehen. Ein ebenso interessanter Aspekt am WWW-Auftritt ist die Backend*-Integration. Während die deutschen Shops bereits an das hausinterne SAP Warenwirtschaftssystem* angebunden sind, müssen die ausländischen Shops für einen Artikeldaten- und Bestelldatenabgleich das selbstentwickelte Warenwirtschaftssystem COFI (COnrad FInanzbuchhaltung) heranziehen. Die Datenabgleiche passieren derzeit bei den inländischen und ausländischen Shops im Offline-Batchbetrieb*. Allein die Integration des Internetauftritts in das Backend* ist ein extrem komplexer Prozess, daher wird an dieser Stelle nicht weiter auf die Feinheiten und Problemen der einzelnen Shops eingegangen. Im Nachfolgenden wird der Shop "www.conrad.de" behandelt. Dieser ist mit ca. 75% Kundenzahl und damit auch ca. 75% Umsatzanteil der wichtigste Shop. Jedoch sind prinzipiell die Probleme in allen Shops die gleichen, da alle URLs* letztlich auf die gleiche Applikation zugreifen bzw. durch die gleiche Applikation abgebildet werden.

Das oben skizzierte Businessmodell* zeigt die typischen Charakteristiken eines Internet-Shops auf. Der Kunde kann durch die Produktpalette navigieren, sich Informationen zu den einzelnen Produkten beschaffen, sich registrieren, Waren merken und letztlich auch Waren kaufen. Wenn man diese doch recht simple Aufgabe im Hinterkopf behält und den hardwareseitigen Maschinenpark anschaut, kommen Zweifel an der implementierten Lösung auf. Die Anforderungen an die Lösung sind zwar nicht gerade gering – ca. 6 Millionen Seitenabrufe pro Monat, entsprechen ungefähr 1 Millionen Benutzer pro Monat – aber eigentlich ist der Hardwarepark ziemlich groß skaliert. Die Architektur* der Lösung ist zentral für die nötige Rechenleistung verantwortlich.

Um den groben Überblick über das installierte System zu vertiefen hier eine visuelle Aufstellung des Systems [PIP99].

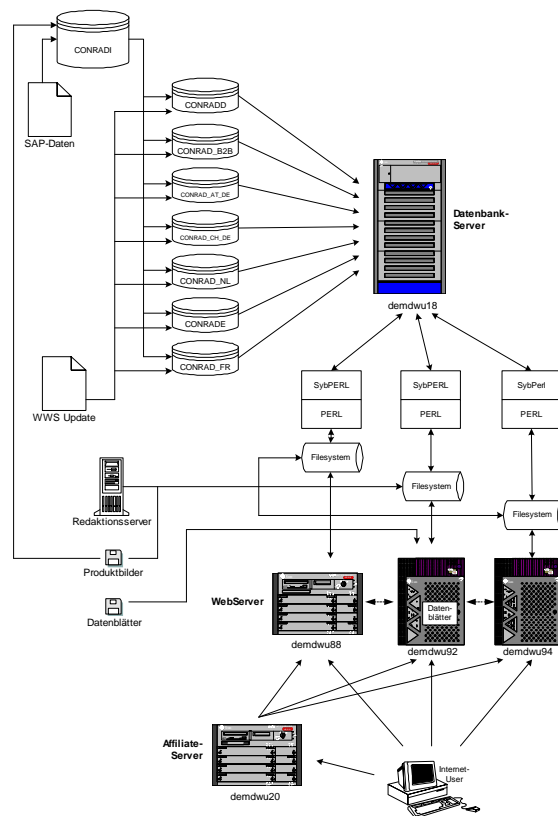


Abbildung 2-1 Architektur der Conrad Online Plattform

Wie das Bild Abbildung 2-1 erkennen lässt, ist der Datenbankserver* ein zentraler Punkt im ganzen System. Um so interessanter ist es anzumerken, dass kein Notfallszenario* seitens Dienstleister existiert. Die Datenbank wird zwar via Backup* gesichert, aber fällt die Datenbank aus, so steht das gesamte System.

2.2. Hardwareüberblick

Hier wird kurz die verwendete Hardware des Systems vorgestellt. Das eigentliche WWW-System basiert auf SUN Solaris. Lediglich für Verwaltungsaufgaben werden NT-Server verwendet.

Name	Server	Hardware		Software		Aufgabe
		CPU	RAM	Betriebssystem	Andere	
demdwu18	SUN6500	18CPU 400MHz	16GB	Solaris 7 64bit	Sybase ASE 11.9.3	Datenbank-Server
demdwu20	SUN4000	8CPU 400MHz	5.5GB	Solaris 7 32bit	Oracle 7.3.1	Affiliate-Server
demdwu88	SUN4500	8 CPU 400MHz	12GB	Solaris 7 32bit	Apache 1.3.9 ModPERL 1.21 PERL 5.005_03	Webserver
demdwu92	SUN450	2 CPU 400MHz	4GB	Solaris 7 32bit	Apache 1.3.9 ModPERL 1.21 PERL 5.005_03	Webserver
demdwu94	SUN450	2 CPU 400MHz	4GB	Solaris 7 32bit	Apache 1.3.9 ModPERL 1.21 PERL 5.005_03	Webserver
InfoOffice (Server ist Pixelpark Eigentum)	PC	PII 233MHz	64MB	NT4 SP5	InfoOffice Server 99- 1-D	Redaktions-Server

Tabelle 2-2 Hardwareaufstellung der Conrad Online Plattform

Wie die Aufstellung Tabelle 2-2 vermuten lässt, sind hardwareseitig keinerlei Bottlenecks* vorhanden. Einziger Schwachpunkt in der obigen Konfiguration ist die feste Zuordnung der URLs auf die drei Webserver. Eine echte Lastverteilung ist so nicht möglich. Ideal wäre hier der Einsatz eines Hardware Loadbalancers auf Routingebene*. Leider sind solche Maßnahmen wegen der Architektur* der Applikation nicht möglich.

2.3. Softwareüberblick

Als Software werden für Standardaufgaben wie z.B. Webserver oder Datenbank Standardsoftwarepakete eingesetzt. Wie bereits in Tabelle 2-2 zu sehen ist, wird als Webserver Apache in Kombination mit ModPERL verwendet. Für die Ausführung der Applikation wird PERL eingesetzt. Als Redaktionstool wird InfoOffice verwendet. Das Redaktionstool wird für die Anpassung der Inhalte in den Shops benutzt. Hierbei werden lediglich redaktionelle Inhalte bearbeitet. Als Datenbankserver* für die Kernapplikation wird der Sybase ASE (Adaptive Server Enterprise) eingesetzt, für die Verwaltung der Affiliate* Shops kommt eine Oracle-Datenbankinstanz zum Einsatz. Affiliate* Shops sind Shops, die andere WWW-Site-Betreiber in Ihre WWW-Präsenz einbinden können und je nach Umsatz eine Provision von der Conrad.com AG erhalten. Soweit zu den Standardkomponenten.

Die größte Eigenentwicklung ist die Shopping-Applikation als solches. Die Applikation ist in PERL* entwickelt und wird durch eine Ansammlung von CGI-Skripten* repräsentiert. Die PERL-Skripten interagieren stark mit der Sybase-Datenbank. In der Datenbank werden Stored Procedures* verwendet, um definierte Aufgaben performant datenbankseitig abzuarbeiten. Um die Performanz der Ausführung von PERL-Skripten als CGI-Skripte* zu verbessern, wird im Webserver ModPERL* verwendet. ModPERL ist ein PERL Interpreter, der als Modul in den Webserver* einkompiliert wird. Wesentlicher Vorteil dieses Moduls ist, dass die Aufrufzeiten des PERL Interpreters wegfallen. Problematisch an PERL ist, dass die Sprache eine interpretierte Sprache mit all ihren Vor- und Nachteilen ist. Als API* zwischen Datenbank und PERL wird SybPerl verwendet. SybPerl ist eine Sammlung von Funktionen, die die Connectivity* zu Sybase bilden.

2.4. Verwendete Datenbank

Hier wird die Installation der Sybase Datenbank untersucht, da diese einen zentralen Bestandteil der Applikation darstellt. Auf der Maschine "demdwu18" zeigt sich bezüglich der Datenbankinstallation folgendes Bild.

Dateisystem	Gigabytes	verwendet	verfügbar	Kapazität	mounted
/dev/vx/dsk/EMAPL2091/OPT.DATABASE	18	15,41	2,54	86%	/opt/database
/dev/vx/dsk/EMAPL2091/OPT.BACKUP.SYBASE	36	17,70	18,15	50%	/opt/backup/sybase
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_1	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_1
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_2	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_2
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_3	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_3
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_4	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_4
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_5	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_5
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_6	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_6
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_7	0,39	0,34	0,04	89%	/opt/dev/sybase/tmp_7
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_8	0,39	0,27	0,10	73%	/opt/dev/sybase/tmp_8
/dev/vx/dsk/EMAPL2091/SYBASE.TMP_9	0,39	0	0,39	1%	/opt/dev/sybase/tmp_9

Tabelle 2-3Datenbankinstallation auf der Maschine demdwu18 - Größenangaben in GigaBytes

Als Datenaufkommen sind produktseitig ca. 65.000 Produkte in der Datenbank zu verwalten. Dazu kommen noch Daten von ca. acht Millionen Benutzern. Zudem sind noch weitere Strukturen für die Applikation in Form von redundant gehaltenen Informationen, Informationen über den Warenkorb, das Sessionhandling* und auch Merktzettelfunktionalitäten in der Datenbank abgelegt.

[RAN96a] Die Sybase-Datenbank ist strukturell so aufgebaut, dass einige umfangreiche Operationen in der tempdb* ablaufen. Dazu gehören z.B. große oder mehrstufige Abfragen, Sortierung (ORDER BY) und Gruppierung (GROUP BY). In der tempdb können auch programmatisch Tabellen für kurzfristige Verwendung erzeugt werden. Diese Tabellen werden entweder explizit vom Nutzer gelöscht, oder beim Abbau der Verbindung. Bei der Shoppingapplikation kommt es häufig dazu, dass der tempdb-Bereich überläuft und die Datenbank nicht mehr performant arbeiten kann. Einzige Lösung in so einem Fall ist das Herunterfahren der Datenbank und anschließende Löschung der tempdb-Bereiche. Warum diese temporären Bereiche der Datenbank Problempunkte darstellen kann nur geraten werden. Allerdings geben tiefere Einblicke in die Code-Struktur den ein oder anderen Hinweis.

Die Datenbank kann nur von "außen" betrachtet werden. Einen direkten Zugang zum System verwehrt uns der Dienstleister. Daher dienen als Informationsquelle lediglich die Daten des Dienstleisters was z.B. die Konfiguration und die Datenmodelle* angeht. Die implementierten Datenmodelle (siehe Anhang A) geben durchaus die gewünschte Realität wieder. Probleme kommen erst auf, wenn man das nationale Datenbankmodell mit dem internationalen Datenbankmodell vergleicht. Hier sieht man, dass das ursprünglich einsprachige Datenbankmodell um das Handling mehrerer Sprachen erweitert wurde. Solche grundlegenden Konzepte wie Mehrsprachigkeit sollten aber bereits im Vorfeld der Modellierung geklärt sein. Da die Abbildungen der implementierten Datenmodelle etwas komplexer sind, soll an dieser Stelle auf den Anhang A verwiesen werden. Dort finden sich die Datenmodellbilder in maximal möglicher Größe. Die Abbildungen der Datenmodelle sind allesamt direkt vom Dienstleister übernommen worden.

Durch den nichtvorhandenen Zugang auf die Datenbank konnte der Grad der Denormalisierung* in der Produktivdatenbank nicht geprüft werden. Im Sinne einer Geschwindigkeitsoptimierung wäre es durchaus sinnvoll das erstellte Modell auszudünnen. Dadurch entstehen zwar Redundanzen, aber bei vielen Leseoperationen, wie das in einem Katalogsystem der Fall ist, spielt die Updatezeit keine herausragende Rolle. Problematisch am momentan verwendeten Konzept ist auch, dass bei einem Textupdate auch nur eines einzelnen Produktes derzeit immer (!) ein Vollreplikat* der

internationalen Datenbank in die nationalen Modelle eingespielt wird. Für die Änderung eines Produkttextes werden also immer ca. 45.000 Produkttexte kopiert. Diese Vorgehensweise könnte man vielleicht noch mit zwei zuge-drückten Augen akzeptieren, würde dieses Replikat nicht gegen 11:00 Uhr angestoßen. Die Datenbank wird durch dieses Replikat natürlich stark belastet.

2.5. Übergreifende Problempunkte Hardware / Software / Datenbank

Vor der Analyse des Gesamtsystems war es fast offensichtlich, dass es speziell zu diesem Punkt jede Menge Fakten zu beschreiben gäbe. Letztlich stellte sich jedoch heraus, dass weder die Hardware noch die Datenbank prinzipiell zu den Schwach- bzw. Problempunkten zählen.

Die Hardwarelandschaft ist ziemlich groß skaliert und wird laut Monitoring meist im einstelligen Prozentbereich belastet. Die Datenbankmaschine – eine SUN Enterprise 6.500 mit 18 Prozessoren und 16 Gigabyte Hauptspeicher – wird höchstens zu 1/3 belastet.

Die Datenbank ist ein Standardprodukt mit seinen Stärken und Schwächen. Einzig die Konfiguration der Datenbank könnte ein Flaschenhals darstellen. Diese soll aber angeblich durch einen Sybase Consultant geprüft worden sein. Einziger wirklicher Schwachpunkt stellt die fehlende Clusterfähigkeit* der Sybase Datenbank dar. Durch die Parallelisierung der Anfragen kann die Leistung noch in gewissen Grenzen erhöht werden.

Bleibt als letzter Problempunkt die Software. Dabei werden nicht die eingesetzten Produkte wie Apache, ModPERL oder ähnliche verwendete Software angesprochen. Hier steht allein die Applikation im Mittelpunkt. Warum die Applikation mittlerweile als Problempunkt identifiziert wurde, wird im Punkt 2.6 näher beschrieben. An dieser Stelle werden nur die suchunabhängige Punkte untersucht. Die Applikation ist eine stark datenbankabhängige Anwendung. Dazu kommt noch, dass mehrere hundert gleichzeitige Benutzer auf die Applikation bzw. Datenbank zugreifen. In vielen funktionellen Zweigen der Anwendung werden Verbindungen zur Datenbank aufgebaut und anschließend nach der Abfrage wieder abgebaut. Der Verbindungsauf- und -abbau ist aber ein ressourcen- und zeitintensiver Vorgang. Die Datenbank wird je nach aufgerufener Funktionalität und Anzahl der gleichzeitigen Benutzer stark belastet. In der Applikation wurde bereits versucht den Auf- und Abbau der Verbindungen

dahingehend zu minimieren, dass beim Start des Apache HTTP-Servers* eine Anzahl Verbindungen zur Datenbank aufgebaut werden und diese von sämtlichen CGI-Skripten*, die innerhalb dieses HTTP-Prozesses ablaufen benutzt werden. Diese Vorgehensweise wurde aber leider nur recht halbherzig verfolgt, wie Blicke in den PERL Code beweisen. Ein anderes konzeptionelles Problem stellt die Verwaltung der Sessions dar. Sessions identifizieren einen Benutzer eindeutig während seines Besuchs auf dem Internet Shop. HTTP als Übertragungsprotokoll ist ein zustandsloses Protokoll. Damit ist jedes Kommando vom Browser* an den WWW-Server mit der Lieferung der Seite abgeschlossen. Die Sessionnummer wird in die URL encodiert. Die Sessions werden hier datenbankseitig verwaltet. Damit muss für jede Prüfung, ob eine Sessionnummer gültig ist, oder nicht die Datenbank angefragt werden. Das geschieht bei jeder Seite bzw. bei jeder Überprüfung der URL. Diese Vorgehensweise belastet die Datenbank durch unzählige Verbindungsauf- und -abbauvorgänge. Neben dieser Verfahrensweise werden oft Tabellen im temporären Datenbankbereich tempdb erzeugt und gelöscht. Ob diese Vorgehensweise des Tabellen anlegen, benutzen und anschließend wieder löschen im Mehrbenutzerumfeld als performant eingestuft werden kann oder eher nicht, kann nicht mit Gewissheit gesagt werden. Problematisch ist auf jeden Fall, dass Abfragen auf neu generierte Tabellen bei Sybase immer wenig performant sind, da der Query-Optimizer* vor der ersten Abfrage die Tabelle und deren Inhalt systematisch auf Optimierungsmöglichkeiten untersucht [RAN96b]. Bei temporären Tabellen frisst schon allein diese Tatsache Performanz.

Bei der Conrad.com AG werden derzeit mehrere Varianten der Performanzsteigerung durchleuchtet. Stichwortartig sollen hier einige dieser Optionen aufgezeigt werden.

Optimierung	Vorteile	Nachteile	Aufwand / Auswirkungen
Optimierungen an der gegenwärtigen DB-Installation (Index / Konfiguration)	kaum Vorteile zu erwarten, da angeblich bereits alle Maßnahmen ausgeschöpft sind	-	Sybase Consultant
Verminderung der Zugriffe auf tempdb	spürbare Entlastung der Datenbank	umfangreiche Änderungen an der Kernapplikation sind nötig	kaum abschätzbar, da die Kernapplikation wie eine große Black-Box ist – selbst für den Dienstleister
dateibasierte Sessionverwaltung	Entlastung der Datenbank	Änderungen an der Kernapplikation sind an einigen dedizierten Punkten nötig	schwer abschätzbar, da der Dienstleister selbst nicht definieren kann, an welchen Schnittstellen von Nöten wären
Statisierung der Kataloginhalte	Entlastung der Datenbank – derzeit sind ca. 13 SQL-Statements pro erstellte Ausgabeseite nötig	Änderungen an der Kernapplikation sind an einigen dedizierten Punkten nötig	erhöhter Pflegeaufwand für die statischen Katalogteile
Aufteilung der Datenbank in Kunden- und Artikeldatenbank durch Verwendung einer zweiten DB-Instanz	Entlastung der Datenbank durch Teilung der Last	Änderungen an der Applikationslogik nötig; Vorteile kaum quantifizierbar	zweifelhafter Erfolg, da die Kundendaten zwar mengenmäßig überwiegen, aber die Zugriffe auf Produktdaten wesentlich häufiger sind
Auslagerung der Suche	Entlastung der Datenbank	erhöhter Pflegeaufwand für die Produktdaten; Änderung an der Kernapplikation sind nötig, aber nicht umfangreich	der Aufwand hält sich stark in Grenzen, die Schnittstellen sind genau spezifizierbar
...			

Tabelle 2-4 performanzsteigernde Maßnahmen für die Conrad Online Plattform

Aufgabenstellung der Diplomarbeit ist es nun die Vor- und Nachteile der ausgelagerten Suche näher zu untersuchen und wenn möglich passende Produkte zu untersuchen.

2.6. Analyse der Suche im bestehenden System

Bei der Analyse der Suche im bestehenden System sah es eine lange Zeit so aus, dass die Suche genauso wie das restliche System als eine Black Box* betrachtet werden muss. Ausgehend von diesem Ansatz wurden die Schwachpunkte der Suche aus Benutzersicht herausgearbeitet. Dabei ist auch ein Graph entstanden, der den Suchprozess und seine Verzweigungen beispielhaft darstellt. Die Verschlagwortung, also die Zuordnung von verschiedenen Indexbegriffen zu Produkten, wird von Hand von einem Dienstleister geliefert.

Erst im Verlauf der Arbeit ist es gelungen, Teile des Sourcecodes der Suche zu bekommen. Ab diesem Zeitpunkt wurde die Suche als White Box* betrachten und untersucht. Anhand der Ergebnisse bei der Black Box* Analyse konnten gleich die PERL-Dateien *AssPowerSuche.pl*, *AssDisplay.pl* ("Ass"* steht hier für Advanced Small

Shops) und *ConShop.pl* fokussiert werden. Diese stellen die zentralen CGI-Skripten* der Suche dar. Sie rufen diverse Stored Procedures in der Datenbank auf. Unter anderem sind dies die Procedures *sp_search* und *sp_text_browse*. Die im Folgenden genannten und untersuchten Stored Procedures und PERL Sourcen können im Anhang C nachgeschlagen werden.

Die Suche ist eine Kombination aus Volltext- und Schlagwortsuche*. Die Volltextsuche* sucht nach Vorkommnissen des eingegebenen Wortes in den Produktbeschreibungen. Die Schlagwortsuche sucht dem Suchwort zugeordnete Produkte in der Datenbank. Die Zuordnung der Schlagworte zu den einzelnen Produkten wird von einem Dienstleister bezogen.

Die Vorgehensweise dieser Methodik kommt bildlich bei der Black Box* Analyse gut hervor, tieferen Einblick gibt anschließend die White Box Analyse.

2.6.1 Ergebnisse der Black Box* Analyse

Während der beispielhaften Analyse der Suche als Benutzer ist der folgende Graph entstanden, der nachfolgend genauer beschreiben wird.

Die Kanten des Graphen sind mit der URL beschriftet, die als Transition* von der Ausgangsseite zur Zielseite gewertet werden kann. Gelesen wird der Graph von links oben nach rechts unten.

In der Abbildung 2-2 sind die Seiten und eine Referenznummer fett hervorgehoben. Die jeweiligen Nummern können im Anhang B wiedergefunden werden. Dort ist für jede Nummer ein Bildschirmabzug beigelegt.

Die Analyse wurde anhand der exemplarischen Suche nach "Nokia" durchgeführt. Prinzipiell gibt es zwei Möglichkeiten im Internet Shop der Conrad.com AG eine Suche durchzuführen. Die Schnellsuche und die Produktsuche. Beide Suchmöglichkeiten sind im Anhang auf der Hauptseite [1] markiert. Beide Suchmöglichkeiten enden letztlich in der Produkt-Suche-Seite [2]. Bei der Produktsuche wird dem Nutzer lediglich die Möglichkeit gegeben die Suche durch die Eingabe mehrerer Suchwörter näher zu spezifizieren [8]. Nach dem Klick auf den "GO"-Button auf der Hauptseite [1] oder der Wahl von "suchen" auf der Seite [8] erhält der Benutzer erste Suchergebnisse präsentiert [2]. Auf dieser Seite kann der Benutzer das erste Suchergebnis näher einschränken und bekommt die passenden Suchergebnisse präsentiert [21], [22], [23]. Werden mehrere Produkte als Ergebnis gefunden, so wird unter der Ergebnisanzeige eine Navigation ("weiter" / "zurück") angezeigt.

Ist der Benutzer noch nicht mit seinem Suchergebnis auf der Seite [2] zufrieden, so kann er sein Suchergebnis verfeinern [24], eine komplett neue Suche anstoßen [25] oder eine Volltextsuche durchführen [26]. Gehen wir davon aus, dass der Benutzer sich dazu entschließt, sein Suchergebnis zu verfeinern, so hat er auf der Seite [24] die Möglichkeiten durch weitere Eingaben die Suche zu verfeinern. Klickt er letztlich auf "suchen", so wird die Seite [32] angezeigt. Diese Seite [32] hat eine starke Ähnlichkeit mit der Seite [2]. Der Benutzer hat auf Seite [32] die gleichen Wahlmöglichkeiten wie auf Seite [2]. Daher werden diese Links nicht weiter verfolgt. Von der Seite [24] ausgehend hat der Benutzer noch die Möglichkeit weitere Eingabefelder für die Suchverfeinerung anzufordern. Er landet letztlich auf Seite [31], die prinzipiell der Seite [24] entspricht – nur eben mit mehr Eingabefeldern.

Wählt der Benutzer auf Seite [2] die Option "neue Suche", so kommt er auf die Seite [25], die der Seite [8] entspricht.

Bleibt letztlich auf Seite [2] nur die Option "Volltextsuche", die zur Seite [26] führt. Auf dieser Seite kann der Benutzer weitere Wörter spezifizieren, die in der Volltextsuche berücksichtigt werden sollen. Der Klick auf "suchen" führt den Benutzer zur Seite [41]. Auf der linken Seite der Seite [41] werden die gefundenen Produkte dargestellt, die der Abfrage entsprechen. Werden mehr Produkte gefunden als links dargestellt werden können, so wird unterhalb der gefundenen Produkte eine Navigation "weiter" / "zurück" eingeblendet. Die Navigation mit den beiden Buttons "weiter" und "zurück" wird exemplarisch in den Seiten [51], [61] und [62] dargestellt.

2.6.2 Ergebnisse der White Box Analyse

Für die White Box Analyse wurden die vom Dienstleister gelieferten Skripten und Stored Procedures analysiert und versucht Konzepte zu extrahieren. Für die Suche sind zwar veraltete, aber immerhin einige Dokumentationen vorhanden. Darauf und auf eigenen Ergebnissen stützen sich die folgenden Ausführungen.

2.6.2.1. Schlagwortsuche

Die Schlagwortsuche im Internetshop wird laut internen Auswertungen zu ca. 95% vom Anwender gewählt. Sie basiert auf einem manuell erstellten Index [EFE99]. Der Index muss manuell erstellt werden, da nicht alle Schlagworte, die einem Produkt zugeordnet werden aus der Produktbeschreibung extrahiert werden können. Das Handy "Siemens C25" beispielsweise enthält im Produkttext nicht explizit das Wort "Mobiltelefon". Ohne manuell aufgebautem Index würde das "Siemens C25" nicht als Mobiltelefon angezeigt. Mit der Verschlagwortung führt die Suche nach "Mobiltelefon" auch zur Anzeige des Handy "Siemens C25". Die Verschlagwortung und auch die Kategorisierung der Produkte wird von einem externen Dienstleister durchgeführt. Das erstellte Verschlagwortungskonzept ist prinzipiell schnell erläutert. Ein Such- oder Schlagwort wird Produktgruppen zugeordnet. Der Begriff "Nokia" wird beispielsweise eindeutig der Produktgruppe "Handy" zugeordnet. Produkte wiederum werden Produktgruppen zugeordnet. Damit verläuft die Schlagwortsuche aus Benutzersicht immer nach dem Schema "Suchwort eingeben – Produktgruppe wählen – Artikel ansehen" ab.

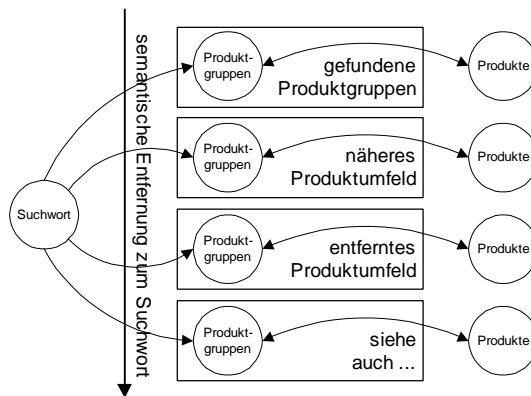
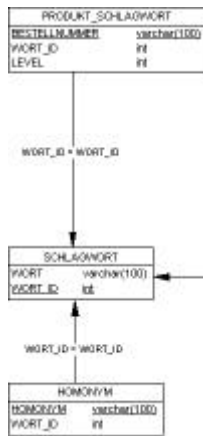


Abbildung 2-3 Prinzip des Verschlagwortungskonzeptes

Bei der Verschlagwortung ist vom Dienstleister auch eine semantische Entfernung der Produktgruppen zum Suchwort berücksichtigt worden. Bei der Eingabe von "Nokia" meint der Benutzer in den meisten Fällen das "Handy" (gefundene Produktgruppen). Manche Benutzer assoziieren auch "Handyakkus" (näheres Produktumfeld), "Handyantennenzubehör" (entferntes Produktumfeld) oder aber auch "Handyakkus" (siehe auch ...) mit dem Begriff "Nokia". Diesem Umstand trägt der Dienstleister mit seinem Verschlagwortungskonzept Rechnung. Das Konzept insgesamt ist natürlich komplexer, aber für das Verständnis an dieser Stelle sei dieser kurze Einblick genug.

Der verschlagwortende Dienstleister liefert die Produktbestellnummern und die jeweils passenden Schlagwörter in einem definierten Datenformat. Diese Daten werden vom shopbetreibenden Dienstleister in die Datenbank eingepflegt [PSW99]. Das Datenmodell*, dass der Dienstleister zur Verschlagwortung verwendet ist ein normalisiertes*. Sehr leicht zu verstehen, aber leider unperformant für die Suche. Der Dienstleister war auch nicht beauftragt ein optimiertes Datenmodell zu liefern, sondern hatte als Auftrag die Verschlagwortung in konzeptueller Form zu erstellen. Der shopbetreibende Dienstleister ist an dieser Stelle für die Umsetzung dieses Konzeptes in einen Algorithmus verantwortlich. Der Verdacht, dass dieser Dienstleister das normalisierte Datenmodell für den Suchalgorithmus nicht angepasst hat, kann leider nicht verifiziert werden, da keine Zugangsberechtigung auf die Datenbank erteilt wird. Die Analyse der SELECT – Statements deutet jedoch darauf hin, dass das normalisierte Datenmodell* für die Suche verwendet wird.

Übergabeformat



Suchtabellen

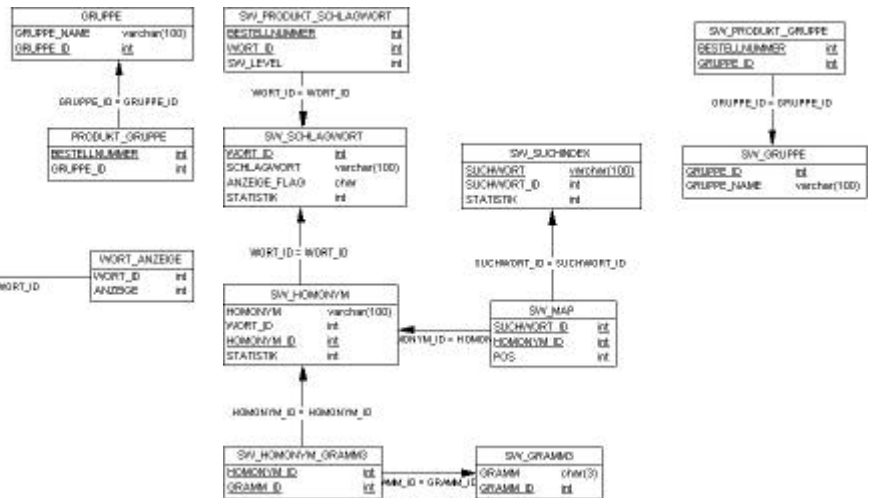


Tabelle 2-5 Datenmodelle für Schlagwortsuche links vom verschlagwortenden Dienstleister, rechts vom shopbetreibenden Dienstleister

Das Einpflegen der gelieferten Daten passiert mit einem PERL Skript namens *sw_indexer.pl*. Die angelieferten Daten werden mittels dieses Skripts in eine Serie von SQL-Statements* übersetzt, die nach vollständiger Löschung des Verschlagwortungsindex in der Datenbank eingespielt werden. Die beiden Einzelschritte der Suche "Bestimmung der Artikelgruppe" und "Bestimmung der Artikel" wird durch die Stored Procedures *sp_schlagwortsuche* und *sp_schlagwortprodukt* abgebildet.

Für die Bestimmung der Artikelgruppe werden zur Ein- und Ausgabe temporäre Tabellen in der tempdb erzeugt. Der Inhalt dieser Tabellen wird von *sp_schlagwortsuche* modifiziert (#SCHLAGWORT) bzw. gefüllt (#SCHLAGWORT_RESULT).

#SCHLAGWORT		
WORT_ID	int	null
SCHLAGWORT	varchar(100)	null
BOOL_MOD	char(1)	null

#SCHLAGWORT_RESULT		
SW_LEVEL	int	
ID	int	
NAME	varchar(100)	null
ANZAHL	int	null

Abbildung 2-4 temporäre Tabellen, die zur Resultatübergabe der Produktgruppensuche verwendet werden

Die Rückgabe in #SCHLAGWORT_RESULT:

LEVEL	ID	NAME	ANZAHL
Level < 1: Vorschlag für nicht gefundenes Schlagwort: -n	WORT_ID	HOMONYM	POS
Level = 0: Vorschlag Schlagworte (Einschränkung Ergebnis): 0	WORT_ID	WORT	null
Level > 0: Rückgabe der Gruppen, Produkte pro Level: LEVEL	GRUPPE_ID	GRUPPE_NAME	ANZAHL

Tabelle 2-6 Rückgabewerte der Produktgruppensuche

Für die Bestimmung der Artikel aus der Produktgruppe ist die Stored Procedure *sp_schlagwortprodukt* zuständig: Zu den Schlagworten werden die Produkte bestimmt. Eine optionale Filterung nach semantischer Entfernung und Gruppe (AND / NOT) ist möglich. Zur Ein- (#SCHLAGWORTID) und Ausgabe (#SCHLAGWORT_PRODUKT) werden auch hier temporäre Tabellen erzeugt.

#SCHLAGWORTID		
WORT_ID BOOL_MOD	int char(1)	null

#SCHLAGWORT_PRODUKT		
BESTELLNUMMER POS	int numeric (5,0)	

Abbildung 2-5 temporäre Tabellen, die zur Resultatübergabe der Produktsuche verwendet werden

Rückgabe in #SCHLAGWORT_PRODUKT:

BESTELLNUMMER	POS
eindeutige Artikelnummer des Produkts	Anzeiger für Relevanz der jeweiligen Produktgruppe / Produkt – Positionszähler

Tabelle 2-7 Rückgabewerte der Produktsuche

2.6.2.2. Volltextsuche

Aus der Datenbank werden die Inhalte mehrerer relevanter Tabellen extrahiert. Aus den einzelnen Inhalten der Tabellen werden Wort → Bestellnummer-Relationen gebildet [PVT99]. Dabei werden Wortvorkommen in Titel, Kategorie und Beschreibung unterschiedlich stark gewichtet. Die gewonnenen Daten werden über einen Bulkload* in die Datenbank geschrieben. Die alten Indexes (Tabellen VTS_INDEX und VTS_ABSATZ) werden dabei gelöscht.

Die Stored Procedure *sp_search* wird aus der Stored Procedure *sp_text_browse* mit den Parametern Suchstring (getrennt durch Leerzeichen) und optionalen Kategorie-IDs und Preisbereich aufgerufen. Der Suchstring wird in Einzelworte zerlegt und auf boole'sche Operatoren* überprüft. Die Suchworte werden in VTS_SUCHINDEX, in VTS_MAP

eine Relation zwischen dem Suchwort und den Worten des Wort-Index (VTS_INDEX) mit einem Flag für exakte Übereinstimmung eingetragen. Nach der Filterung (boole'sche Operatoren*, Kategorie-ID, Preisbereich) wird das Suchergebnis geordnet in die von *sp_text_browse* erzeugte temporäre Tabelle #SEARCH.TEMP geschrieben. Die Ordnung richtet sich nach folgenden Kriterien: Anzahl der Worte (relativ) mit Priorisierung exakter Übereinstimmungen vor Teilmatches, Gewichtung (entsprechend der absoluten Anzahl der Worte).

Durch die Begrenzung der Suchergebnismenge mit den Parametern "Start" und "Count" in *sp_text_browse* und einem Join auf die Tabelle ABSATZ_INFO wird die Information zur Darstellung des Ergebnisframes gewonnen. Die Parameter "Start" und "Count" bilden ein Fenster, das die jeweils anzuzeigenden Produkte in der Tabelle #SEARCH.TEMP definiert.

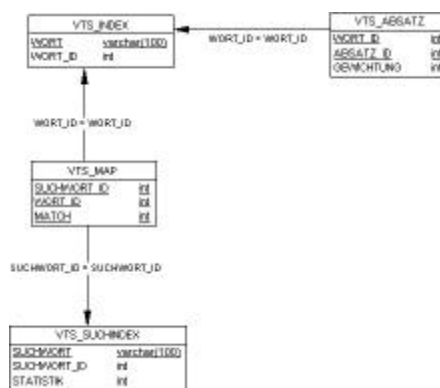


Abbildung 2-6 Datenmodell für die Volltextsuche

2.6.3 Schwachpunkte des existierenden Suchmodells

Während der Analyse des Ist-Systems wurden auch die Schwachpunkte des Suchmodells herausgearbeitet. Diese Schwachpunkte wurden kategorisiert. Ausgehend vom Black Box* Ansatz wurden die Problem- und Schwachpunkte der Suche analysiert, die ohne Wissen der inneren Struktur erkennbar sind. Anschließend ist in der White Box Analyse die Dokumentation des Dienstleisters verwendet worden. Hierbei wurden die internen Problempunkte fokussiert.

2.6.3.1. Sichtbare Schwachpunkte – Black Box* Analyse

Bei der Black Box* Analyse wird nachfolgend auf die Punkte Bedienerfreundlichkeit, Performanz und Funktionalität eingegangen.

Das Thema Bedienerfreundlichkeit ist eine heikle Angelegenheit, da kein Tester den Anspruch der Objektivität für sich erheben kann. Bedienerfreundlichkeit ist immer ein subjektives Thema.

Die Aufteilung Suche im Hauptframe, Anzeige der Treffer im linken Navigationsframe ist etwas gewöhnungsbedürftig. Die Anzeige auf der linken Seite erscheint oftmals zu klein und unübersichtlich. Ebenfalls nicht hilfreich und irreführend ist die Anzahl der Treffer im Ergebnisframe. Manchmal werden pro Seite zehn Treffer angezeigt, manchmal nur vier und wiederum ein anderes Mal sechs Treffer. Speziell hier entsteht der Eindruck, dass die Suche nicht ganz ausgereift ist. Ebenfalls nicht schön ist, dass für die Suche nur eine HTML-Seite verwendet wird, auf der im oberen Drittel die Categoriesuche abgebildet ist, im mittleren Drittel die Möglichkeit zu verfeinern und abschließend die Volltextsuche. Ist die Bildschirmauflösung höher als 800x600 Bildpunkte eingestellt oder scrollt der Benutzer, so werden die einzelnen Funktionalitäten sichtbar. Die Volltextsuche und die Categoriesuche stellen für den Benutzer aber getrennte Aufgabenbereiche dar. Diese Unlogik macht sich auch auf der Seite [24] in Abbildung 2-2 bemerkbar, wenn man auf "zusätzliche Eingabefelder" klickt. Dem Benutzer werden zwar mehr Eingabefelder angeboten, aber die Suchseite wird nicht bei "Suche verfeinern" angezeigt, sondern einfach das obere Drittel der HTML-Seite – die Categoriesuche. Der Benutzer muss also erst zu den Eingabefeldern hinscrollen. Bei geringer Bildschirmauflösung springt der Benutzer also von der Funktionalität "Suche verfeinern" zur Funktionalität "Categoriesuche". Während der Analyse der Suche ist der Eindruck entstanden, dass die Suche konzeptuell zwar gut durchdacht, aber nicht optimal umgesetzt worden ist.

Nach der konzeptuellen Analyse der Suche wurde festgestellt, dass die Antwortzeit der Suche stark von der Tageszeit und dem gesuchten Begriff abhängt. Eine Suche kurz nach dem Mittagessen nach dem Begriff "Handy", ergibt Antwortzeiten von ca. sieben Sekunden. Wurde zur gleichen Tageszeit nach dem Begriff "Farbfernseher" gesucht, so lag die Antwortzeit bei knapp zwei Sekunden. Suchen nach den gleichen Begriffen am frühen Morgen waren um ca. 30-40% schneller. Das Verhalten der Applikation ist nachvollziehbar, da um die Mittagszeit mehr Surfer auf den WWW-Shop zugreifen, als frühmorgens. Problematisch jedoch ist die Antwortzeit von sieben Sekunden. Der

durchschnittliche User wartet laut Statistik höchstens sieben Sekunden auf eine Antwort eines Webserver. Schlecht ist zudem, dass die Wartezeit bei jeder Seite auftritt, die mit der Suche zu tun hat. Warum die Suche z.T. so langsam ist, wird in 2.6.3.2 näher erläutert. Nur ein kleiner Hinweis an dieser Stelle. Die Applikation beansprucht die Datenbank in sehr hohem Maße und für die Belastung sind maßgeblich die im Hintergrund ablaufenden Prozesse verantwortlich, die das nichttriviale Suchkonzept realisieren.

Die Suche in der Applikation ist vom Dienstleister als eine der ressourcenintensivsten Prozesse in der Applikation identifiziert worden. Aus diesem Grunde wird die Suche immer, wenn der Status der Applikation als kritisch eingestuft wird, einfach abgeschaltet. Der Benutzer kann zwar die Suchfelder ausfüllen, bekommt aber keine Antwort auf die Suchanfrage. Bei den Analysen hat die Volltextsuche auch nicht zufriedenstellend funktioniert. Navigiert man über die Navigation zum Handy "Siemens S25", so kann man feststellen, dass das Wort "Siemens" in der Produktbeschreibung auftaucht. Anschließend sollte die Volltextsuche bei der Eingabe von "Siemens" mindestens dieses Handy als Treffer ergeben. Leider liefert die Volltextsuche jedoch überhaupt keine Treffer.

2.6.3.2. Interne Schwachpunkte – White Box Analyse

Bei der Betrachtung der Dokumentation und der diversen Skripten wurden vor allem die Punkte Wartbarkeit, Systembelastung, Effizienz und Zukunftsorientierung betrachtet.

Zunächst werden die zur Verfügung stehenden Skripte hinsichtlich der Wartbarkeit untersucht. Aus der Abbildung 2-2 kann man ablesen, dass das PERL Skript *AssPowerSuche.pl* zentraler Bestandteil der Suche ist. Wegen der großen Relevanz für die Suche wurde zunächst dieses Skript untersucht.

Der Sourcecode ist trotz seiner Länge von knapp 1.500 Zeilen insgesamt in nur vier Funktionen aufgeteilt, wobei zwei der Funktionen jeweils knapp 20 Zeilen umfassen. Der Code ist unstrukturiert. Die verwendeten Variablen haben nichtssagende Namen und werden öfters im Verlauf des Codes in verschiedenen Kontexten verwendet. Der Code ist extrem spärlich mit Kommentaren versehen. Optische Strukturierungshilfsmittel wie Tabulatoren, um Hierarchiestufen anzudeuten sind per se

nicht vorhanden. Dadurch ist der Code ohne Überarbeiten nur sehr schwer lesbar. Das Skript macht den Eindruck, dass der Code in kurzer Zeit schnell gewachsen ist. Schnelles Wachstum an sich ist sicherlich kein Problem. Leider kann jedoch aus dem Sourcecode kein Konzept extrahiert werden. Die Implementierung ist "straight-ahead"* ohne ein Fein*- oder Grobkonzept* passiert. Hier entartet das Wachstum, da unkontrolliert und ohne Planung qualitativ suboptimaler Code entsteht. Der Eindruck, den der Code vermittelt wird noch durch den Umstand verstärkt, dass keine Dokumentation vorliegt. Wegen der mangelnden Transparenz des Codes scheiterten mehrere Versuche, Konzepte oder in einem Reengineeringprozess* Strukturen zu erkennen. Die Logik des Codes ist sehr schwer zu durchblicken. Auch der Versuch einen Eingabe – Verarbeitung – Ausgabe – Fluss zu verfolgen ist gescheitert. Da keine Schnittstellendokumentation vorliegen ist es unmöglich die Ausgaben abhängig von den Eingaben vorherzubestimmen. Letztlich entsteht der Eindruck, dass das Modul nur vom Autor weiterentwickelt und gepflegt werden kann. Durch die fehlende Dokumentation und Konzeption ist es kaum möglich als Außenstehender einen Über- oder Einblick in das Modul zu bekommen. Leider muss dieser Schluss, der sich nur auf ein Modul bezieht auf das gesamte System ausgeweitet werden. Sicherlich sind an einigen Punkten Dokumente vorhanden, aber ein Konzept für das gesamte System liegt leider nicht vor. Das System stellt sich als schwer bzw. gar nicht wartbar heraus.

Nach dem Aspekt der Wartbarkeit wurde der Code hinsichtlich der Systembelastung untersucht. Die Gesamtbelastung auf das System kann nur geschätzt werden. Es liegen zum einen nicht alle Sourcen vor und andererseits müsste sonst das Zusammenspiel der gesamten PERL Skripten, Stored Procedures und auch der restlichen Systemkomponenten untersucht werden. Bei der Analyse des Codes kommt der Verdacht auf, dass einige Funktionalitäten kaum oder nur rudimentär optimiert sind. Viele Schleifen und letztlich auch die extensive Verwendung von regulären Ausdrücken beim Stringersatz belasten das System. Durch das (wahrscheinlich) fehlende Konzept als roter Faden hinter dem Source wirken die einzelnen Funktionalitäten eher schnell hinprogrammiert, als durchdacht entworfen und anschließend implementiert. Wirklich interessant werden letztlich die abgesetzten SQL-Statements für die Datenbankabfrage. SELECT-Statements, die mehrere Tabellen "joinen" sind eher die harmloseren Fälle.

Sicherlich ist es nicht ganz einfach, performante SQL-Statements zu bauen, aber letztlich sind die SQL-Statements einfacher, je besser das Datenmodell* durchdacht ist.

```
select distinct
    SA.SEITEN_ID,
    AI.ABSATZ_ID,
    AI.ABSATZ_TITEL,
    AI.THUMBNAIL,
    AI.PROD_ANZ,
    AI.BESTELLNUMMER
from
    ABSATZ_INFO AI,
    ABSATZ_SEITEN SA,
    #SCHLAGWORT_PRODUKT T,
    PRODUKT_ABSATZ PA
where
    AI.ABSATZ_ID = PA.ABSATZ_ID
and
    AI.ABSATZ_ID = SA.ABSATZ_ID
and
    PA.BESTELLNUMMER in (SELECT BESTELLNUMMER FROM #SCHLAGWORT_PRODUKT)
and
    PA.ABSATZ_ID > 200000000
```

Ausschnitt 2-1 SQL-Anweisungen aus dem PERL Skript AssPowerSuche.pl

```
select h.HOMONYM_ID, count(*), H_LEN=(
    select count(*)
    from SW_HOMONYM_GRAMM3
    where HOMONYM_ID = h.HOMONYM_ID)
from SW_HOMONYM_GRAMM3 h, #GRAMM3 g
where g.GRAMM_ID = h.GRAMM_ID
group by h.HOMONYM_ID
having count(*) >= @REQ
...
insert into #RESULT
select f.HOMONYM_ID, f.ANZAHL, f.H_LEN
from #FOUND f, SW_HOMONYM h
where f.HOMONYM_ID = h.HOMONYM_ID
order by f.ANZAHL desc, f.H_LEN
...
insert into SW_MAP
select @SID, HOMONYM_ID, POS
from #RESULT
where POS <= (
    select max(POS)
    from #RESULT
    where ANZAHL = (
        select ANZAHL
        from #RESULT
        where POS = @MAX_RES)
    and H_LEN = (select H_LEN
    from #RESULT
    where POS = @MAX_RES))
```

Ausschnitt 2-2 SQL-Anweisungen aus der Stored Procedure "sp_schlagworte"

Auch beim Datenmodell entsteht leicht der Eindruck, dass nicht zu viel Zeit darauf verwendet wurde. Als besonders interessante Vorgehensweise soll hier die Erzeugung von Tabellen, die kurzzeitige Verwendung und die anschließende Löschung dienen. Im PERL Skript *AssPowerSuche.pl* allein werden zwei Tabellen nach dieser Art in der tempdb von Sybase genutzt. Temporäre Tabellen werden bei Sybase mit einem

vorangestellten "#" gekennzeichnet. *AssPowerSuche.pl* ruft im Verlauf der Ausführung die Stored Procedures *sp_schlagwortsuche* und *sp_schlagwortprodukt* auf. Die erste Procedure erzeugt fünf Tabellen nach diesem Muster, die zweite Procedure zwei Tabellen. Ein anderes Skript *sp_search* erzeugt sogar sechs dieser Tabellen während der Ausführung. Die Tabellen werden zur Ergebnisübermittlung von intern ablaufenden Procedures verwendet. Problematisch an diesen temporären Tabellen ist, wie in 2.5 bereits erwähnt, der Query Optimizer. Der Query Optimizer untersucht jede neu angelegte Tabelle auf mögliche Optimierungen bei der Abarbeitung von Anfragen. Dieser Vorgang verbraucht natürlich Performanz.

Im gesamten Code findet man immer wieder Funktionalitäten, die auf die Datenbank verlagert wurden. Um die Datenbank von den PERL Skripten aus zu bedienen, muss für jede Kommunikation eine Verbindung zur Datenbank aufgebaut werden. Allein im Skript *AssPowerSuche.pl* konnten vier Verbindungsauf- und -abbauten aufgefunden werden. Je nach Ablaufpfad werden allein in diesem einen Skript bis zu vier dieser Kommunikationen erzeugt. Der Aufbau der Kommunikation kommt einem Anmelden an der Datenbank gleich, der Abbau einer Abmeldung. Bei jedem Aufbau wird also der gleiche Verwaltungsprozess angestoßen, wie beim Anmelden. Ein User-Record wird angelegt, die Zugriffsrechte geprüft, der Benutzer authentifiziert, ...

Zum Thema Effizienz wurde ebenso hauptsächlich das PERL Skript *AssPowerSuche.pl* untersucht. Auffallend zum Thema Effizienz ist die Verwendung von mehrfach geschachtelten "if"-Konstrukten und die Verwendung von Stringvergleichen. Allein die Verwendung von Strings als Parameter sollte bei einem solch komplexen Stück Codefunktionalität in der Planungs- und Konzeptionsphase eliminiert werden. Im Code werden sehr viele Stringvergleiche durchgeführt. Die starke Verschachtelung der "if"-Konstrukte könnte auch im Vorfeld der Implementierung anhand eines Struktogrammes oder eines Ablaufplanes eliminiert werden. Diese beiden beispielhaft herausgestellten Aspekte können auf den gesamten Code ausgedehnt werden. Der Source und auch der Ansatz könnten bei entsprechenden Überlegungen im Vorfeld effizienter gestaltet werden.

Auch dem Benutzer gegenüber präsentiert sich die Suche als nicht effizient. Der grundsätzliche Anspruch des Benutzers an die Suchfunktionalität wird nicht immer

erfüllt. Entweder die Suche dauert dem Benutzer zu lange – die Antwortzeit ist zu hoch – oder sie funktioniert gar nicht, da die Server überlastet sind. Die Suche könnte – intern und auch nach extern – effizienter sein.

Nach den Punkten Wartbarkeit, Systembelastung und Effizienz wird nun noch ein sehr wichtiges Thema – Zukunftsorientierung – betrachtet. Wenn die Suche als eine eigenständige Applikation angesehen wird, so vereint die Implementierung sehr viele Punkte, die nicht von zukunftsorientierter Denkweise zeugen. Die Suche ist aus obigen Gründen für Außenstehende kaum wartbar. Die Applikation Suche ist nicht oder nur unzureichend dokumentiert. Wie soll eine solche Anwendung erweitert oder gepflegt werden ? Die Suche ist unperformant und uneffizient. Eigentlich sind schon die beiden letzten Punkte gute Gründe für eine komplette Neuimplementierung. Zudem wäre mit einem starken Fokus auf Zukunft keine Interpretersprache mit all ihren Vor- und Nachteilen die Sprache der Wahl, sondern eine Compilersprache. Noch besser wäre die Wahl eines kommerziell erprobten Systems. Gerade bei der Funktionalität "Suche" ist die Gefahr groß, das Rad neu zu erfinden. Da aber gerade die Suche in einem Katalogsystem eine zentrale Funktionalität darstellt, muss diese schnell, zuverlässig und einfach sein. Viele Anbieter haben sich auf die Suche oder neudeutsch "Knowledge Management" spezialisiert. Die Verwendung eines kommerziellen Systems hat zudem den Vorteil, dass das System meist einen eigenen Index in einem proprietären Datenbankformat aufbaut und sich somit vom eigentlichen Datenbanksystem abkapselt. Die Belastung der Suche kann so leichter skaliert und verteilt werden. Bei der jetzigen Applikation, die ein einfacher Aufsatz auf die Datenbank darstellt, wird die Datenbank von der Suchapplikation förmlich mit SQL-Statements und Stored Procedure Aufrufen "vergewaltigt". Schätzungen zufolge entspricht die Anzahl der Anfragen an die Datenbank den Anfragen an das grafische Frontend – multipliziert mit einem Faktor X. Diesen Faktor zu beziffern ist relativ schwer, aber Aussagen des Dienstleisters zufolge bewegt sich dieser im Bereich zwischen 15 und 25. Jede Anfrage an den HTTP-Server zieht also ca. 15-25 Anfragen an die Datenbank nach sich. Ein kommerzielles Produkt, das speziell für die Suche ausgelegte Datenmodelle und optimierte Routinen für die Abfrage implementiert, stellt eine homogene, harmonische Applikation dar. Die PERL Applikation stellt vielmehr eine heterogene, kantige Lösung für die Suche dar.

Trotz allen Widrigkeiten, gerade hinsichtlich der PERL Applikation, entsteht der Eindruck, dass eine gut designde Applikation – selbst auf PERL Skripten basierend – die jetzige Serverlast gut tragen könnte. Mit Verwendung eines Application Servers wäre die Problematik sicherlich ganz beseitigt.

3. Ziele des Soll-Modells

Nach den Schwachpunkten in der implementierten Suche sollen die Kriterien definiert werden, die erfüllt sein müssen, um eine Verbesserung der Suchfunktionalität im WWW-Auftritt der Conrad.com AG zu erreichen. Ein guter Ansatzpunkt ist es die Ziele für das neue Suchmodell aus den Schwachpunkten des implementierten Suchmodells abzuleiten. Diese abgeleiteten Ziele wurden um weitere, von Benutzerwünschen geleiteten, Zielen ergänzt [PFP00].

Neben den definierbaren Zielen sind auch fixe Rahmenparameter zu berücksichtigen. Eine unverrückbare Tatsache ist die Wahl der Betriebssystemplattform – Sun Solaris. Der Umfang der Suche muss zudem die Funktionalität der existierenden Suche mindestens erreichen, im besten Fall sogar erweitern.

Nachfolgend sollen einzelne Kriterien näher erläutert und die Motivation, warum die einzelnen Punkte wichtig sind, aufgezeigt werden.

3.1. Schnelle Lösung

Die Suchapplikation muss die Antwort auf die Anfrage des Kunden in kurzer Zeit liefern, da ansonsten die Gefahr besteht, dass der Kunde die Suche und damit den möglichen Einkauf abbricht. Die Antwortzeit der Applikation muss sieben Sekunden auf jeden Fall unterschreiten. Sieben Sekunden ist die durchschnittliche Zeit, die ein Besucher laut Untersuchungen bereit ist auf eine Antwort im WWW zu warten. Diese Zeit muss unterboten werden, um wettbewerbsfähig zu bleiben bzw. wieder zu werden. Aus technischer Sicht sollte die Datenbank nicht über allgemeine Datenbankschnittstellen wie ODBC* o.ä. angesprochen werden. Diese Schnittstellen haben zwar den Vorteil, dass viele Datenbankhersteller adressiert werden können, jedoch leidet die Performance unter diesem verallgemeinernden Ansatz sehr. Die Suchapplikation sollte native Treiber* für verschiedene Datenbanken (mindestens Oracle und Sybase) anbieten. Idealerweise sollte die Suchapplikation eine Einheit mit der Datenbank bilden und dabei die Datenbank wenig belasten. Eine Einheit kann die Applikation eigentlich nur dann bilden, wenn die Suchapplikation aus dem gleichen Hause kommt, wie die zugrundeliegende Datenbank. Diese Vorgehensweise würde auch eher sicherstellen, dass die Datenbank von der Suchapplikation nicht überbelastet wird. Für die Suchapplikation würde es bedeuten, dass sie ein kommerzielles Produkt ist und keine Eigenentwicklung. Für den Fall einer Eigenentwicklung muss sich in

jedem Fall die Interaktion mit der Datenbank auf ein absolutes Mindestmass reduzieren. SQL Statements müssen dann optimiert und das Datenmodell* eventuell verändert werden. Zudem muss das Konzept der Suche optimiert und damit der Code der Applikation verschlankt werden. Diese Kernpunkte sind gute Ansätze für eine performante Lösung.

3.2. Skalierbare Lösung

Um einen "Single Point of Failure" ausschließen zu können, sollte die Suchapplikation nicht aus einem monolithischen* Block bestehen, sondern verteilt auf mehreren Rechnern laufen können. Wenn die Applikation verteilbar ist, sollte sie möglichst linear skalieren*. Eine verteilte Applikation würde an dieser Stelle auch eine höhere Performance bedeuten. Ist die Applikation grundsätzlich nicht verteilbar, so sollten doch mehrere Prozesse der gleichen Applikation gestartet werden können. Es darf nicht vorkommen, dass nur wegen der Starrheit der Suchapplikation Kunden und somit Umsatz verloren geht. Darum muss die Suchapplikation redundant* aufgebaut sein. Im Sinne einer gewissen Offenheit sollte die Suche auf mehrere gängige Datenbanken (z.B. Oracle, Sybase) aufsetzbar sein. Aufsetzbar bedeutet hier, dass sich die Applikation nahtlos mit der Datenbank verquickt, oder aber dass die Applikation zwar einen eigenen proprietären Index aus Performanzgründen verwendet, aber trotzdem auf die Daten dieser Datenbanken zugreifen kann.

3.3. Leicht wartbare Lösung

Ausgehend von der Erfahrung mit der existierenden, selbstentwickelten Applikation zur Suche ist es ein absolutes Muss, dass die neue Applikation eine leicht wartbare Lösung ist. Zur Minimierung des Wartungsaufwandes müssen Standards verwendet werden, weil weithin bekannte Konzepte und Vorgehensweisen zum Einsatz kommen. Vorteilhaft an dieser Stelle ist auch, dass die Verwendung von Standards die Abhängigkeit von speziellen Wissensträgern reduziert. Menschen, die sich mit Standards auskennen und diese umsetzen können, sind leichter zu finden, als Menschen, die proprietäre Eigenlösungen verstehen und letztlich auch warten können. Sollte die Applikation eine Eigenentwicklung sein, so muss der Quellcode strukturiert und intern wie auch extern dokumentiert werden. Bei der Eigenentwicklung sollte auf jeden Fall der Minimalstworkflow Konzeption – Implementierung – Test – Wartung berücksichtigt, befolgt und dokumentiert werden. Durch die Dokumentierung soll

erreicht werden, dass die neue Applikation eine tatsächliche White Box ist. Die Vorgänge in der Applikation, Eingabe – Verarbeitung – Ausgabe sollen anhand der Dokumentation ohne Unterbrechung nachvollziehbar sein.

Neben diesen technischen Voraussetzungen müssen auch operative Aspekte bei der Wartung berücksichtigt werden. Der Aufwand für Dateneinspielungen, Aktualisierungen des Index, personeller als auch maschineller Mehraufwand müssen im Zusammenhang mit der neuen Suche minimiert werden. Ideal wäre ein Szenario, in dem die Artikeldaten ohne manuelle Zuarbeit, also ohne Pflegeaufwand, verwaltet werden können. Zudem sollte die Migration und Installation der neuen Applikation ohne Probleme vonstatten gehen.

3.4. Technologisch zukunftsorientierte Lösung

Die Suchapplikation sollte eine möglichst offene Plattform darstellen, die auf Standards beruht und damit leicht und flexibel erweiterbar ist. Die Zukunftsorientierung basiert zentral auf dem Argument des Investitionsschutzes. Wenn sich die Applikation an neue Anforderungen anpassen kann, so ist ein optimaler Investitionsschutz gewährleistet. Muss bei minimalen Änderungen am Suchprozess bereits die gesamte Applikation geändert bzw. neuimplementiert werden, so ist dieses Ziel gescheitert. Im Sinne der Leistungsmessung sollte die Transparenz in der Applikation auf höchstmöglichem Level angesiedelt werden. Ist die Transparenz gegeben, so kann die Anzahl der Anfragen, die an die Datenbank gesendet werden genau definiert werden. Sind diese internen Strukturen durchschaubar, so ist es unproblematisch ein Ursache – Wirkungsmodell aufzubauen, in dem genauere Aussagen über die Leistungsfähigkeit der Applikation getroffen werden können. Neben diesen technisch orientierten Aspekten muss in jedem Fall auch die Leistungsfähigkeit und Glaubwürdigkeit des jeweiligen Anbieters geprüft werden. Eine kommerzielle Applikation, deren Herstellerfirma schlechte Leistung bietet, oder schlechte Beratungsleistung bietet, disqualifiziert sich schon im Vorfeld. Selbst dann, wenn die technischen Voraussetzungen erfüllt sind.

3.5. Funktionsorientierte Lösung

Die Suche als Applikation muss sich an bestimmten Forderungen nach der zu implementierenden Funktionalität orientieren. Nachfolgend werden die Mindestkriterien definiert und daran anschließend die von Benutzern gewünschten Funktionalitäten, die

in weiteren Ausbaustufen realisiert werden sollten. Prinzipiell gilt hier bei der Suche aber auch der Grundsatz erfolgreicher Software "Je einfacher, desto erfolgreicher".

3.5.1 Minimalfunktionalität

Als absolutes Muss ist die Schlagwortsuche zu definieren. Dabei steht das manuell erstellte Konzept zur Verschlagwortung und Kategorisierung im Vordergrund.

Aufgrund dieser Verschlagwortung ist die Benutzerakzeptanz extrem gestiegen. Rund 95% aller Suchen gehen über die Verschlagwortung.

Als weitere Funktionalität muss die Suche per Artikelnummer bei Eingabe der ganzen Artikelnummer, oder auch nur Teilen davon funktionieren. Ebenso die verknüpfte Suche, also die Kombination von Suchbegriffen mit boole'schen Operatoren*. Eine Volltextsuche soll ebenso verfügbar sein. Gemeinsames Ziel aller Funktionalitäten ist die Ermittlung einer möglichst optimalen Treffermenge.

3.5.2 gewünschte Funktionalität

Die folgenden Funktionalitäten stellen alles Kann-Anforderungen dar und können erst in späteren Phasen realisiert werden. Ein Hauptziel sollte die Ähnlichkeit zu bestehenden Suchmaschinen – optisch und auch funktionell – sein. Vorteil daran ist, dass der Kunde den Umgang gewöhnt ist und keine Umstellung von Nöten ist. Das Suchsystem muss dem Kunden bei zu wenigen oder zu vielen Treffern Hilfestellungen für die Suche anbieten. Die Suchapplikation muss die Möglichkeit bieten, die Suche bestehend auf der bereits gefundenen Treffermenge zu verfeinern oder zu vergrößern. Die Verfeinerung soll auch über die logische Verknüpfung von zwei bzw. mehreren Begriffen möglich sein. Auch der Preis eines Artikels sollte in die Suche miteinbezogen werden. Beim Preis sollte der Kunde die Möglichkeit haben eine Preisspanne einzugeben. Kunden wissen oft relativ genau, wie viel ein Artikel ungefähr kostet. Die Applikation soll eine sogenannte Suchhistorie haben. Das heißt, dass die Suchmaschine sich merkt, welche bisherigen Suchworte ein Kunde eingegeben hat, bzw. aus dem Hilfeangebot gewählt hat. Der Kunde erhält eine Liste, mit der er an die bisherige Suche anderer Kunden anknüpfen kann. Er kann einen neuen Suchversuch ab einem bestimmten Punkt in der Suche des vorigen Kunden wählen. Die Suchhistorie ist bis zum kundenspezifischen Cookie ausbaubar. Zusätzlich wäre eine phonetische Suche, d.h. eine Suche, die Eingaben hinsichtlich lautsprachlicher Aspekte bewertet, denkbar.

Eine unscharfe Suche, die Schreibfehler in einem gewissen Toleranzbereich ausgleicht würde die Hauptfunktionalitäten abrunden.

4. Suchmaschinen im Allgemeinen

Bevor im nächsten Kapitel mögliche Lösungen für die beschriebene Problematik der Conrad.com AG behandelt werden, soll dieses Kapitel Informationen allgemeiner Natur rund um Suchmaschinen und deren verwendeten Hintergrundtechnologien bieten. Hierbei wird der Fokus nicht auf unendliche Tiefe gelegt, sondern nur ein breiter Überblick über die einzelnen Technologien gegeben. Für tiefergreifendere Informationen möge der geneigte Leser im Literaturverzeichnis nachschlagen.

4.1. Überblick über Suchmaschinen

Nach einem kurzen Ausflug in die Geschichte der Suchmaschinen und den möglichen Einsatzbereichen folgen technische Möglichkeiten der Suche auf einem Computer. Dort werden Vor- und Nachteile der einzelnen Varianten aufgezeigt. Anschließend kommt ein Überblick über Arten von Suchmaschinen.

4.1.1 Geschichte und Einsatzbereiche

Die Geburtsstunde der modernen Suchmaschinen ist nahe an der Geburtsstunde der eigentlichen elektronischen Datenverarbeitung und der Möglichkeit Massendaten zu speichern. Mit dem Anwachsen von Datenmengen kam auch immer öfters das Problem auf, Daten schnell und zuverlässig zu finden. Anwachsende Quellcodedateien und auch Textdateien verlangten bald nach effektiven Volltextsuchmöglichkeiten. Von der Suche innerhalb einer Datei erfolgte ein Übergang von der Suche innerhalb von Dateien hin zur Suche nach Dateien und die Untersuchung deren Inhalts. Hier wurde bald die dateisystembasierte Suche durch indexunterstützte Suchmethoden ersetzt. Langsam nähern wir uns den Konzepten heutiger Suchmaschinen an. Heutzutage basieren die meisten Suchmaschinen auf großen Indexdatenbanken, die nach bestimmten Kriterien speziell für die performante Suche optimiert sind.

Suchmaschinen werden heutzutage vor allem dort eingesetzt, wo dynamisch dezentrale Inhalte verwaltet werden und schnell aufgefunden werden sollen. Dies ist beispielsweise im Internet- oder Intranetumfeld der Fall. Suchmaschinen, die dort eingesetzt werden, gleichen fortlaufend die tatsächlich gespeicherten Dokumente mit den indexierten Dokumenten ab. Änderungen an Dokumenten oder neue Dokumente werden so schnell und sauber erfasst und der Datenbestand der Suchmaschine fortlaufend aktualisiert. Mit anwachsender Menge an Dokumenten steigt auch das Alter des Indexdatenbestandes der Suchmaschine. Die Suchmaschinen haben dann Probleme die anwachsende

Datenmenge zu indexieren. Dieses Problem tritt ganz extrem im Internet und in großen Intranet-Netzen auf.

Andere Einsatzgebiete von Suchmaschinen sind sogenannte OPACs (Online Public Access Catalogues). Das sind z.B. Datenbestände von Bibliotheken oder Verzeichnisse von Magazinen. Hier zeichnet sich die Anwendung durch extrem viele Suchabfragen auf einem statischen Datenbestand aus. Suchmaschinen, die auf derart statischen Datenbeständen agieren sind anders konzipiert, als Suchmaschinen, die mit der Schnelligkeit des Internets umgehen müssen. [SUF99a]

4.1.2 Grundlegende Suchkonzepte auf einem Computer

Mit dem Fortschreiten der Geschichte wurden natürlich auch unterschiedliche Konzepte der Suche auf einem Computersystem durchlaufen.

Nach der Volltextsuche innerhalb einer Datei wurde die Suche auf das Dateisystem des Computers ausgedehnt. Speicherressourcen waren plötzlich nicht mehr so knapp und Dateien und darin gespeicherte Daten konnten leichter abhandelt werden. Die Suche im Dateisystem zeichnet sich im Wesentlichen dadurch aus, dass keinerlei Vorbereitungen für die Suche an sich notwendig sind. Die Suchmaschine muss keine Indexdatei oder – datenbank aktualisieren bzw. aufbauen. Die Daten sind immer aktuell, weil eben keine Indexdatei existiert. Neben diesen Vorteilen fallen aber vor allem die starke Belastung für das Gesamtsystem auf. Für jede Suchabfrage müssen alle Dateien einzeln angesprochen und evtl. auch noch durchsucht werden. Logischerweise wird die Suche auf dem Dateisystem je nach Rechner und Zahl der zu durchsuchenden Dateien immer langsamer. Außerdem hat die Suche im Dateisystem auch noch das Problem, dass Mehrbenutzertauglichkeit nur in gewissem Umfang gegeben ist. Wollen beispielsweise zwei Suchprozesse die gleiche Datei durchsuchen, so wird einer der beiden Prozesse blockiert.

Nach der Suche in vielen Dateien auf Dateisystemebene hat sich die indexbasierte Suche etabliert. Hier wird die Suche in einer einzigen Datei mit der Suche im Dateisystem kombiniert. Eine Indexsoftware speichert in einer (Index-)Datei Verweise auf die zu durchsuchenden Dateien ab. Die Indexsoftware sammelt Informationen über und in den Dateien auf und schreibt diese kombiniert in eine Indexdatei. Die Suchsoftware durchsucht die Indexdatei hinsichtlich der Relevanz des eingegebenen Suchtextes mit dem Index. Bei Treffern verweist die Suchsoftware auf die eigentliche

(Quell-)Datei. Vorteilhaft wirkt sich auf die Systembelastung aus, dass lediglich eine Datei durchsucht werden muss. Auch wird die Suche schneller sein, als bei der reinen dateisystembasierten Suche. Schlechter hingegen ist die Erhöhung der Komplexität der Suchsoftware. Jetzt reichen nicht allein die Hilfsmittel des Betriebssystems zur Dateisystemverwaltung für die Suche. Die Funktionalität des Suchens und Indexierens wird in zwei getrennte Prozesse gespalten. Auch der Ressourcenverbrauch für die Indexdatei kann ziemlich hohe Ausmaße erreichen. Je nach Detaillierungs- und Kompressionsgrad kann die Indexdatei gleich groß, oder sogar noch größer werden, als die Summe der eigentlich zu durchsuchenden Dateien. Bei hohen Kompressionen kann der Datenbestand in der Indexdatei aber wieder bis auf ca. 4-6% der ursprünglichen Größe reduziert werden. Problematisch an der Indexdatei ist die Tatsache, dass der Index immer veraltet ist. Die Indexsoftware muss den Datenbestand periodisch nach Änderungen und Neuerungen durchsuchen und den Index aktualisieren. Bei relativ kleinen Datenbeständen kann die Aktualisierung nahe Realtime erfolgen, aber bei großen Datenbeständen (wie im Internet) kann die Überarbeitung teilweise bis zu mehreren Wochen dauern. Diese Problematik wächst mit der Zahl der Dateien und der Häufigkeit der Änderungen an diesen Dateien.

Problematisch an der Speicherung von Daten in der Indexdatei ist nun noch der Verlust der Semantik der Daten. In der Indexdatei werden die Daten unstrukturiert gespeichert. Um die Strukturierung der Indexinformationen nicht zu verlieren, wurde die Indexdatei im Laufe der Zeit in eine Datenbank verlegt. In der Datenbank muss der Indexdatenbank auf einzelne Datenattribute verteilt werden, um überhaupt sinnvoll abgespeichert zu werden. In der strukturierten Indexdatenbank kann bei der Suche jetzt zwischen der Berufsbezeichnung „Müller“ und dem Nachnamen „Müller“ unterschieden werden. Bei der Suche in der Indexdatei geht diese semantische Unterscheidung verloren. Ebenso von Vorteil ist, dass eine Datenbank speziell auf Datensuch- und -einfügeoperationen optimiert ist. Die Datenbank an sich stellt also die derzeit bestmögliche Performance bei Suchoperationen zur Verfügung. Aus diesem Grund operieren die meisten Suchmaschinen heutzutage auf Datenbankbasis. [SUF99a]

Suche im Dateisystem	
die Funktionen des Betriebssystems zur Verwaltung von Dateien werden ausgenutzt;	
Vorteile	Nachteile
Keine Vorbereitungen notwendig (keine Indexdatei notwendig)	jede einzelne Datei muss einzeln adressiert werden
Daten sind immer aktuell	starke Belastung für das Gesamtsystem
	Suche wird je nach Rechner und Zahl der Dateien langsamer
	mehrere Sucher greifen auf eine Datei zu (zwangweise Sequentialisierung)

Tabelle 4-1 Vor- und Nachteile der Suche im Dateisystem

Suche in einer Indexdatei	
Kombination der Suche in einer einzigen Datei und der Suche im Dateisystem; Indexsoftware sammelt alle Informationen über und in den Dateien und schreibt alles in eine (Index-)Datei; die Suchsoftware durchsucht die Indexdatei und verweist auf die eigentliche (Quell-)Datei	
Vorteile	Nachteile
Höhere Suchgeschwindigkeit (nur eine Datei durchsuchen)	Suchsoftware wird komplexer (Suchsoftware und Indexsoftware nötig)
Geringere Systembelastung	große Indexdatei (enthält alle Informationen der indexierten Dateien)
	Indexsoftware muss periodisch gestartet werden
	Index ist immer veraltet (Problem wächst mit der Zahl der Dateien und der Häufigkeit der Änderungen)

Tabelle 4-2 Vor- und Nachteile der Suche in einer unstrukturierten Indexdatei

Suche in einer Indexdatenbank	
Strukturierung der Indexinformationen; kein Verlust der Semantik der einzelnen Indexinformationen; optimierte Suche möglich; kommerzielle, spezialisierte Produkte einsetzbar; die Vor- und Nachteile sind denen der Suche in einer Indexdatei ähnlich	
Vorteile	Nachteile
Semantik der einzelnen Informationen bleibt erhalten (Name „Müller“ != Beruf „Müller“)	eigene spezialisierte Software für Indexverwaltung (Datenbanksystem) nötig
Spezialisiertes System optimiert für Datenverwaltungsaufgaben (Suche, Einfügen, Ändern)	ebenfalls Indexsoftware nötig

Tabelle 4-3 Vor- und Nachteile der Suche in einer strukturierten Indexdatenbank

4.1.3 Arten von Suchmaschinen

Hinter Suchmaschinen verstecken sich vielerlei Konzepte. Am einfachsten ist die Unterscheidung in Volltextsuchmaschinen, Kataloge und Metasuchmaschinen.

Metasuchmaschinen sind Kombinationen mehrerer Suchsysteme. Sie übergeben den eingegebenen Suchstring an andere Suchmaschinensysteme und verarbeiten die Gesamttreffermenge nach eigenen Maßgaben. Die Suchergebnisse der einzelnen angefragten Systeme werden vor der eigentlichen Präsentation hinsichtlich eigener Relevanzmaßgaben sortiert und nochmals ausgewertet. Der Vorteil für den Sucher ist die parallele Anfrage an mehrere Suchsysteme und die kanalisierte Ergebnisbündelung.

Neben den Metasuchmaschinen existieren Kataloge. Ein ganz bekanntes Katalogsystem ist Yahoo. Kataloge unterscheiden sich im Wesentlichen von Volltext- und Metasuchmaschinen dadurch, dass die zugrundeliegenden Indexdaten manuell erstellt

werden. Die Kategorisierung der Daten wird manuell von Redakteuren erledigt. Eingängigster Vorteil der manuellen Verarbeitung der zu indexierenden Daten ist die hohe Qualität des Suchergebnisses. Die zu indexierenden Daten werden von Redakteuren gesichtet und gewertet. Dabei werden die Daten kategorisiert und im Index platziert. Für den Aufbau, die Suche und die Datenvorhaltung werden keine so großen Rechenressourcen benötigt wie bei der Volltextsuchmaschine. Ebenso entfällt die Netzbelastung durch die Agenten und Roboter, die den Volltextsuchindex aufbauen.

Neben den obigen Varianten an Suchmaschinen gibt es noch die eigentlichen Volltextsuchmaschinen. Volltextsuchmaschinen unterscheiden sich im Wesentlichen durch die Art der Informationssammlung und –kategorisierung. Was bei Katalogen hauptsächlich manuell passiert, wird bei Volltextsuchmaschinen automatisiert. Agenten, Roboter oder Crawler durchsuchen ständig die Datenbasis (z.B. das Internet oder das Intranet) und melden neue oder aktualisierte Daten an die Suchmaschine. Diese kategorisiert die Neudaten und aktualisiert den Index. Die Agenten durchstreifen ständig den Datenbestand und erzeugen dadurch eine nicht unerhebliche Netzlast. Im Internet sollen laut neuerer Untersuchungen bis zu 7% des Gesamtdatenverkehrs durch Agenten und Roboter entstehen. Trotz dieser ständigen Aktualisierungen der Agenten ist der Index ab einer bestimmten Größe des Datenbestandes immer veraltet. Die Aktualität des Index hängt ebenso stark von der Änderungsfrequenz am Datenbestand ab. Für den Aufbau, die Pflege und die Suche im Indexdatenbestand sind erhebliche Rechnerressourcen erforderlich. Zum einen braucht der automatisch aufgebaute Index viel Plattenspeicher und zum anderen müssen die Agenten im Netz gesteuert und die Daten indexiert werden. Dafür sind leistungsfähige Rechner von Nöten. Zudem muss der Rechner noch die ganzen Anfragen der Sucher verarbeiten. Volltextsuchmaschinen durchsuchen den Datenbestand oft nur stur nach der eingegebenen Zeichenkette. Meistens werden hier noch boole'sche Verknüfungsoperationen ("UND" / "ODER") berücksichtigt. Bei der Suche auf einem festgelegten Datenbestand werden mittlerweile auch schon fortgeschrittenere Suchmethodiken angewendet. Dabei kommen z.B. die phonetische Suche (Suche nach bestimmten Sprachmustern) oder die Mustersuche (Suche nach bestimmten Wort-/ Buchstabenmustern) zum Einsatz. Diese Methodiken

sind jedoch durch extrem große Indexdateien nur für einen eingeschränkten Datenbestand einsetzbar.

4.1.4 Generelle Aufgaben von Suchmaschinen

Die Aufgaben der Suchmaschinen im Allgemeinen kann man in vier Phasen unterteilen: Akquise der Daten, Indexierung der Daten, Aktualisierung des Index und die Anfragenbearbeitung. Gemäß dieser Aufteilung existieren für die verschiedenen Arten von Suchmaschinen auch verschiedene Softwarekomponenten für die Realisierung dieser Aufgaben.

	Metasuchmaschine	Katalog	Volltextsuchmaschine
Akquise der Daten	-	Manuell / Redakteur und Kunde	Maschinell / Agent
Indexierung der Daten	-	Manuell / Redakteur	Maschinell / Datenbank
Aktualisierung des Index	-	Manuell / Redakteur	Maschinell / Datenbank
Anfragenbearbeitung	Maschinell / Suchmaschine	Maschinell / Suchmaschine	Maschinell / Suchmaschine

Tabelle 4-4 Arten von Suchmaschinen und deren Aufgaben

4.2. Überblick über die Aufgaben und Einsatzgebiete von Agenten

Agenten, oder auch Roboter, Crawler, Spider, ... genannt, sind für die Datenakquise aus dem Datenbestand zuständig. Sie wandern ständig über den gesamten bekannten Datenbestand und melden Neudaten und Änderungen an bestehenden Daten an die Suchmaschine zurück. Dort wird der Index den Meldungen entsprechend aktualisiert. Die Agenten folgen Stück für Stück den Verknüpfungen (in HTML-Dokumenten: Links) in einem Dokument. Als Startpunkt verwenden Agenten entweder bereits vorhandene Daten oder stützen sich auf neu gemeldete Seiten. Treffen diese Agenten auf Verknüpfungen in Dokumenten entscheiden die Agenten für jeden Link, wie mit diesem weiterverfahren werden soll. Die weitere Vorgehensweise hängt von der jeweiligen Strategie der Suchmaschine bzw. des Agenten ab. Manche Agenten folgen beispielsweise maximal 3 Linkebenen auf einem Server und senden den jeweiligen Dokumenteninhalt an die Suchmaschine. Andere verfolgen nur eine Linkebene und verzweigen dann auf andere Server. Agenten im Internet verfolgen zumeist nur explizit gekennzeichnete Links, dem HTML-Element <A>. Mit Frames*, Imagemaps* oder anderen Verweismöglichkeiten haben die meisten Agenten Probleme und ignorieren derart gestaltete Verweise. Durch die Arbeitsweise der Agenten entsteht eine nicht zu verachtende Netzlast. Die Agenten fordern ständig Dokumente über das Netz an und verarbeiten die empfangenen Dokumenten ihrer Strategie entsprechend. Die Zeit

zwischen den Besuchen des Datenbestandes variiert zwischen einigen Tagen und mehreren Wochen. Auch hier existieren verschiedene Konzepte und Strategien. Manche Suchmaschinen steuern die Besuchsfrequenz des Roboters danach, wie häufig die zu indexierenden Dateien geändert werden. Im Internet kann sich ein Serverbetreiber auch davor schützen, von einem Agenten besucht zu werden oder dem Agenten spezielle Informationen mitgeben. Dadurch können Dokumente geschützt werden oder aber auch der Server entlastet werden. Derzeit funktioniert diese Art Steuerung über Meta-Tags innerhalb der HTML-Dokumente oder eine Datei *robots.txt*, die an einer festgelegten Stelle auf dem Server hinterlegt wird. Viele Agenten werten diese Informationen aus. [SUF99b]

4.3. Überblick über verschiedene Methoden und Techniken von Suchmaschinen

Suchmaschinen basieren je nach Anwendungsgebiet auf verschiedensten Technologien. Die einzelnen Suchmaschinen verwenden zur optimalen Erfüllung ihrer Aufgaben die jeweils optimalste Technologie. Andere Suchmaschinen wiederum verwendeten in frühen Versionen einfache Methoden und verfeinerten die dahinterliegenden Konzepte. Mittlerweile verwenden sie komplexe mathematische Modelle, um die Suche zu optimieren. In diesem Abschnitt soll ein Überblick über die Methoden und Techniken verschiedener Suchmethoden gegeben werden, ohne dabei jedoch zu tief in die dahinterliegenden mathematischen Konzepte und Formeln einzugehen.

4.3.1 Boolean Retrieval Methoden

[OAR99, HIL95, CAW99a] Beim Boolean Retrieval werden vor der Indexierung der Dokumente Indexbegriffe definiert. Die einzelnen Dokumente werden hinsichtlich der Existenz dieser Begriffe im Inhalt untersucht und eine Indextabelle erstellt. In der Indextabelle wird die Relevanz der einzelnen Dokumente hinsichtlich der Indexbegriffe vermerkt. Das Dokument und die darin enthaltene Information wird also durch eine Menge von Schlüsselworten oder Indexbegriffen beschrieben (z.B. Geld, Macht, Bauern, ...). Es erfolgt eine Abstraktion des Inhaltes des Dokumentes auf einige wenige Begriffe. Dabei ist generell ein Informationsverlust zu erwarten. Der Grad des Verlustes ist stark von der Wahl der Indexbegriffe abhängig. Werden Begriffe für die Indexierung gewählt, die wenig Information repräsentieren und dabei auch noch häufig vorkommen

(z.B. "die" oder "macht"), so leidet die Qualität des Indexes darunter. Anfragen, die derlei Begriffe enthalten, produzieren eine extrem große Treffermenge [SUF99c].

Dokumentenname / Inhalt	Indexiertes Wort und gleichzeitig Suchwort					
	Geld	Macht	Bauern	Kuchen	Schrank	Schränke
d ₁ = "Geld allein macht glücklich"	X	X				
d ₂ = "Bauernmöbel und Schränke"			X			X
d ₃ = "Kuchen backen für Singles"				X		
d ₄ = "Die Macht der Könige"		X				
d ₅ = "Gebäck im Kühlschrank"					X	
d ₆ = "Macht Kuchen dick ?"		X		X		

Tabelle 4-5 Beispiel einer Indextabelle

Die Indextabelle hat den Vorteil, dass sich die Menge der zu speichernden Informationen pro Dokument stark verringert und somit das zu speichernde Datenvolumen drastisch einschränkt.

Als Suchabfrage wird ein boole'scher Ausdruck verwendet, der Indexbegriffe mit boole'schen Operatoren verknüpft.

Dazu ein kleines Beispiel anhand Tabelle 4-5.

Anfrage	Ergebnismenge
"Geld AND (Macht OR Erfolg) AND Reichtum"	leer
"Geld OR Macht"	d ₁ , d ₄ , d ₆
"Kuchen AND (Schrank OR Macht)"	d ₆

Tabelle 4-6 Beispielabfragen auf der obigen Indextabelle

Das boole'sche Suchverfahren ist eine Methode nach dem "Exact-Match" Verfahren. Dokumente, die exakt der Suchabfrage entsprechen werden der Treffermenge zugeordnet. Problematisch an dieser Vorgehensweise ist, dass ein Dokument, das bei einer Abfrage wie "A OR B OR ... OR Z" nur eine Bedingung erfüllt genauso in der Treffermenge ist, wie ein Dokument, das alle Bedingungen erfüllt. Andererseits ist ein Dokument, das bei einer Abfrage wie "A AND B AND ... AND Z" nur eine Bedingung nicht erfüllt, genauso wenig in der Treffermenge, wie ein Dokument, das keine der Bedingungen erfüllt [SAL82a].

Der Benutzer erhält also immer eine exakte Antwort auf seine Anfrage. Dokumente, die sich in der Ergebnismenge befinden, sind alle gleichwertig. Ein Ranking nach der Relevanz bezüglich der Anfrage wird nicht vorgenommen. Die Größe der Antwortmenge ist vom Benutzer kaum kontrollierbar. Sie ist abhängig von den

festgelegten Indexbegriffen und den Begriffen in der Abfrage. Meistens erhält der Anfragende entweder extrem viele Ergebnisse oder überhaupt keine.

Ein zentrales Problem der boole'schen Suche ist der Anspruch an den Benutzer seinen komplexen und unscharfen Informationsbedarf in eine formale Frageformulierung unter Verwendung logischer Operatoren zu übersetzen. Diese Umsetzung wird immer näherungsweise und grob passieren. Bei diesem Prozess geht je nach Erfahrung des Benutzers Information verloren. Die eigentlich nicht korrekte Anfrage des Benutzers wird von der boole'schen Suche allerdings exakt ausgewertet. Spätestens hier stellt sich allgemein die Frage, ob die Methodik der boole'schen Suche dem Problem "Informationen finden" angemessen ist [KNO94a].

Trotzdem ist die boole'sche Suche das am weitesten verbreitete Verfahren und kommt bei großen Suchdiensten fast ausschließlich zur Verwendung. Die Suche mit der boole'schen Suchoperatorenverknüpfung ist eine der ersten optimierten Suchmethoden überhaupt und ist einfach und schnell zu implementieren. Die Suchabfrage ist aufgrund des erstellten Indexes performant.

4.3.2 Vector Space Retrieval Methoden

[KNO94b, CAW99b] Das Vector Space Retrieval Modell ist das älteste - und trotzdem ein aktuelles - Modell zur strukturierten Suche nach Informationen. Das Vector Space Verfahren ist ein mathematisch einfaches und gut handhabbares Modell mit starker Analogie zum 3-D Raum. Ausgangspunkt des Verfahrens ist ein Vektorraum, der für jeden Indexbegriff eine eigene Koordinatenachse – also Dimension – besitzt. Bei N Indexbegriffen wird ein N-dimensionaler Vektorraum aufgespannt. Jedes einzelne zu indexierende Dokument wird über N Koordinaten im N-dimensionalen Vektorraum platziert.

Dazu ein kleines Beispiel im 3-D Raum.

Für $N=3$ mit den Indexbegriffen t_1 = automatisch, t_2 = manuell und t_3 = Indexierung besitzt das Dokument "Alles über automatische Indexierung" die Darstellung (1 / 0 / 1). In diesem Beispiel wurden keine differenzierten Gewichte benutzt, sondern "1" zeigt die Anwesenheit und "0" die Abwesenheit eines Indexbegriffes an [KNO94b].

Eine an das System gestellte Abfrage wird wie ein Dokument als Punkt im Vektorraum dargestellt. Berechnet wird nun die Distanz des Punktes im Raum, der die Abfrage darstellt und den umliegenden Punkten, die Dokumente darstellen. Die Distanz zwischen den Punkten wird als Maß für die Ähnlichkeit zwischen Abfrage und Dokument angesehen. Die Ausgabe der Ergebnisse an den Benutzer erfolgt nun nach steigendem Abstand bzw. fallender Ähnlichkeit. Der Benutzer erhält also alle Dokumente der Ergebnismenge nach Relevanz gewichtet. Bei der Gewichtung der Dokumente wird implizit die Annahme getroffen, dass die Ähnlichkeit zwischen Abfrage und Dokument ein Indikator für die Relevanz bezüglich der Abfrage darstellt. Das Vector Space Modell ist ein heuristisches Modell, das nach dem "Partial Match" Verfahren arbeitet. Durch die Ähnlichkeit zum 3-D Raum ist es ein anschauliches Verfahren. Im Unterschied zum boole'schen Verfahren müssen die Anfragen an das System nicht strukturiert sein. Dadurch ist es jedoch nicht mehr möglich, satzähnliche Konstrukte oder synonyme Begriffe durch boole'sche Operatoren* zu verknüpfen. Wie auch beim Boolean Retrieval Modell ist es von zentraler Bedeutung die Indexbegriffe zu definieren. Bei der Entscheidung, welche Indexbegriffe den Vektorraum aufspannen sollen, kann die Berechnung eines "term discrimination value" hilfreich sein. Der "term discrimination value" misst für jeden Indexbegriff, ob dieser dem Ziel dienlich ist, oder ob seine Verwendung negative Folgen auf die Verteilung der Dokumente hätte [KNO94b].

Beim Aufstellen der Indexbegriffe sollte beachtet werden, dass die Verwendung zu häufiger Begriffe als Indexes die Dokumente im Vektorraum zusammenrücken lässt, denn sehr viele Dokumente gleichen sich dann in der Hinsicht, dass sie diesen Begriff enthalten. Analoges, wenn auch mit umgekehrtem Sinn, gilt für den Fall zu seltener Begriffe. Ziel der Indexbegriffwahl ist es, die Dokumente so zu repräsentieren, dass sie möglichst gut unterscheidbar, also möglichst verschieden sind. Neben der Optimierung der Indexbegriffen, die den Vektorraum aufspannen, kann das Verfahren noch durch weitere Optionen verbessert werden. Zunächst kann durch eine Gewichtung der Indexbegriffe eine Verbesserung der Retrievalergebnisse erreicht werden. Durch die Gewichtung der einzelnen Begriffe wird die Abstandsmessung der Punkte im N-dimensionalen Vektorraum beeinflusst. Durch geeignete Gewichtung der Begriffe kann die Abstandsmessung und damit die Ähnlichkeitsberechnung optimiert werden. Zur

Gewichtung der Indexbegriffe kann beispielsweise das "Document Frequency" [KNO94c] Verfahren herangezogen werden. Das Document Frequency Verfahren (auch bekannt als "Inverse Dokumenthäufigkeit") ist eine einfache, plausible und effektive Formel zur Termgewichtung, die in verschiedenen Varianten vorkommt. Beispielhaft soll hier auf die einfachste Form eingegangen werden. Betrachtet man ein bestimmtes Dokument d , so kann man für einen darin vorkommenden Indexbegriff i folgende Gewichtung definieren:

$$\text{inverse Dokumentenhäufigkeit}(i, d) = \frac{\text{Auftretenshäufigkeit von } i \text{ in } d}{\text{Dokumenthäufigkeit von } d}$$

Das Gewicht eines Indexbegriffes ist hoch, wenn es nur wenige Dokumente gibt, in denen er auftaucht und wenn er gleichzeitig im fraglichen Dokument häufiger vorkommt.

Dazu ein kleines Beispiel:

Betrachtet werden 3 Dokumente d_1 , d_2 und d_3 aus einer Kollektion von 10.000 Dokumenten.

d_1 = "Anwendung von VectorSpace Modellen zum Information Retrieval"

d_2 = "Zusammenhang zwischen Information, Daten und Information Retrieval"

d_3 = "VectorSpace Modelle im Vergleich"

Damit mit der obigen Formel gearbeitet werden kann, wird die Anzahl der Dokumente benötigt, in denen der jeweilige Indexbegriff gefunden wurde.

Indexbegriff	Anzahl Dokumente
Anwendung	3.000
Information	2.000
Modell	500
Retrieval	200
VectorSpace	1.200
Zusammenhang	1.800
Daten	700

Tabelle 4-7 beispielhafte Indexierung von Dokumenten

Aus der Tabelle 4-7 kann für die einzelnen Dokumente folgende Indexierung erstellt werden:

d_1	d_2	d_3
Anwendung (1/3000); VectorSpace (1/1200); Modellen (1/500); Information (1/2000); Retrieval (1/200);	Zusammenhang (1/1800); Information (2/2000); Daten (1/700); Retrieval (1/200);	VectorSpace (1/1200); Modelle (1/500);

Tabelle 4-8 beispielhafte Indexierung für drei Beispieldokumente

Für die Anfrage "VectorSpace und Information Retrieval" ergibt sich dann folgende Gewichtung für die 3 Beispieldokumente:

	Gewicht	übereinstimmende Indexbegriffe
d_1	$1/1200 + 1/2000 + 1/200 = 77/1500 = \mathbf{0,051}$	VectorSpace, Information, Retrieval
d_2	$2/2000 + 1/200 = 3/500 = \mathbf{0,006}$	Information, Retrieval
d_3	$1/1200 = \mathbf{0,0008}$	VectorSpace

Tabelle 4-9 beispielhafte Gewichtung der drei Beispieldokumente

Eine weitere Optimierung im Zusammenhang des Vector Space Retrieval Modells ist die mögliche Clusterung von Dokumenten. Dabei wird aus einer Gruppe von Dokumenten ein virtueller Repräsentant berechnet, der letztlich diese Dokumente bei der Ähnlichkeitsberechnung vertritt. Wird der virtuelle Vertreter der Gruppe als relevant angesehen, so werden für weitere Berechnungen die geclusterten Dokumente verwendet.

4.3.3 Probabilistic Retrieval Methoden

[KNO94d, CAW99c] Wie der Name dieser Kategorie von Techniken schon sagt, basiert diese Technologiefamilie auf Wahrscheinlichkeiten. Im Gegensatz zu Vector Space Modellen, die an die Anschauung appellierende, heuristische Verfahren sind, versuchen probabilistische Modelle normativ* zu arbeiten. Grundlage der Methode ist der mathematische Beweis, dass dasjenige Ranking optimal ist, das die Dokumente nach der Wahrscheinlichkeit ihrer Relevanz für die Frage anordnet. Diese Wahrscheinlichkeit gilt es – unter Berücksichtigung vereinfachender Annahmen – abzuschätzen. Grundlage für die Wahrscheinlichkeitsabschätzung ist die Verteilung der Indexbegriffe auf die Dokumente.

Durch die vereinfachenden Annahmen bezüglich der Wahrscheinlichkeit der Relevanz der Dokumente werden die Modelle erst rechnerisch oder vom Datenaufkommen her handhabbar. Die Annahmen sind meist sehr realitätsfremd, haben aber den Vorteil, dass sie letztlich eine optimale Lösung liefern und explizit offengelegt sind – also nachvollziehbar sind.

Jedem probabilistischem Verfahren liegt ein spezifisches Modell zu Grunde, das die Daten und Wahrscheinlichkeiten definiert, die benötigt werden, um die gesuchte Wahrscheinlichkeit (nämlich die, dass ein Dokument, das gewisse Begriffe enthält, auf eine Frage, die bestimmte andere Begriffe enthält, relevant ist) zu errechnen. Wahrscheinlichkeiten, die vom Modell als nötig definiert werden, müssen abgeschätzt werden.

Dazu ein Beispiel:

Gibt es in einer Datenbank mit N Dokumenten r relevante Dokumente auf eine Frage, so errechnet sich die Wahrscheinlichkeit, dass ein beliebiges Dokument der Datenbank "nicht relevant" ist als

$$p(\text{nicht relevant}) = \frac{1}{N - r}$$

Da r in der Praxis aber nicht bekannt ist und weil N normalerweise sehr viel größer ist als r , kann man statt des korrekten Wertes ohne großen Fehler auch die einfache Abschätzung

$$p(\text{nicht relevant}) = \frac{1}{N}$$

verwenden.

Von der Qualität der Schätzungen hängt in hohem Maße die Brauchbarkeit des Verfahrens ab. Um das spezifische Modell zu verfeinern, können vereinfachende Annahmen eliminiert werden. Das Modell nähert sich mit Reduktion der vereinfachenden Annahmen der Realität an. Mit der Erhöhung der Komplexität des Modells steigt aber auch die Zahl der zu schätzenden Parameter. Da bei jeder Einzelschätzung immer ein Fehler akzeptiert wird, summieren sich die Einzelfehler bei mehreren Einzelschätzungen zu einem größeren Gesamtfehler. Aus diesem Grund sind einfachere Modelle mit wenigen Abschätzungen den komplexeren, realitätsnäheren überlegen [KNO94d].

Anfragen an probabilistische Suchmethoden sind genauso unstrukturiert, wie Anfragen an das Vector Space Modell.

Als Beispiele für probabilistische Verfahren werden hier kurz drei Verfahren skizziert.

4.3.3.1. Grad der Wortübereinstimmung

[CIS, ROB76] Bei diesem Verfahren wird die Wahrscheinlichkeit p , dass ein Dokument auf die Anfrage f passt, an der Anzahl der übereinstimmenden Wörter gemessen.

f = "Informationen Datenbanken finden"

Dokument	Grad der Übereinstimmung
"Moderne Datenbanken"	1/3
"Speichern von Informationen in Datenbanken"	2/3
"Suchen und Finden in Datenbanken"	2/3
"Informationen über's Glücklichein finden"	2/3
"Die Kunst, Informationen in Datenbanken zu finden"	3/3

Tabelle 4-10 beispielhafte Anwendung des Verfahrens "Grad der Wortübereinstimmung"

4.3.3.2. Anwendung verschiedener Verfahren zur Gewichtung von Indexbegriffen

[CIS, ROB76] Ein anderer Ansatz ist es jeden einzelnen Indexbegriff – oder auch Kombinationen von Indexbegriffen – zu gewichten. Das Wahrscheinlichkeitsmaß für die Relevanz eines Dokumentes auf eine Anfrage ist dann die Summe aller Gewichte. Damit wird das Ranking dahingehend optimiert, dass Dokumente, die eine gleiche Anzahl an Wörtern mit der Anfrage gemeinsam haben bezüglich der Relevanz der jeweiligen Indexbegriffe bewertet werden.

Dazu drei Beispiele:

- "Collection Frequency" Indexbegriffe, die nur in wenigen Dokumenten auftauchen sind eher nützlich, als solche, die in vielen Dokumenten vorkommen.
- "Term Frequency" Je häufiger ein Indexbegriff in einem Dokument vorkommt, desto wichtiger erscheint dieser Begriff für das gesamte Dokument zu sein.
- "Document Length" Ein Indexbegriff, der gleich oft in einem kurzen Dokument vorkommt, wie in einem langen Dokument, ist für das kurze Dokument charakteristischer, als er das für das lange Dokument ist.

Für ein Dokument/Indexbegriff-Paar werden alle obigen Gewichte berechnet und anschließend kombiniert. Daraus ergibt sich ein Gesamtgewicht, das für das Maß der Relevanz der einzelnen Dokumente hinsichtlich der Anfrage steht.

4.3.3.3. "Relevance Feedback"

[CAW99d, KNO94d, POO99a] Die Wahrscheinlichkeit p , dass ein Dokument auf eine Frage mit dem Frageterm f relevant ist, wird in diesem Verfahren berechnet als

$$p(\text{relevant}) = \frac{r(1-u)}{u(1-r)}$$

r ist dabei die Wahrscheinlichkeit, dass f im Dokument vorkommt, unter der Voraussetzung, dass es relevant ist.

u ist analog die Wahrscheinlichkeit, dass f im Dokument vorkommt und es nicht relevant ist.

4.3.4 Extended Boolean Retrieval Methoden

[CAW99e, SAL82b] Die Extended Boolean Retrieval Modelle ergänzen die herkömmliche boole'schen Suchmodell durch Verwendung eines Rankingmechanismus, um die Ähnlichkeit zwischen Dokument und Anfrage zu berechnen. Den Erweiterten boole'schen Suchmodellen gemeinsam ist, dass sie durch Austausch der logischen Operatoren AND und OR versuchen die Relevanz der Treffer zu erhöhen. Für die Ersetzung der Operatoren verwenden die Methoden die unterschiedlichsten Ansätze. Durch die Verwendung dieser komplexen Operatoren wird die einfache boole'sche Suche durch Rankingmechanismen ergänzt. Dadurch wird das einfache Modell mit seinen Unzulänglichkeiten hinsichtlich der Suchergebnisse wesentlich verbessert.

[LEE95a] Extended Boolean Retrieval Modelle können generell durch ein Quadrupel $\langle T, D, Q, F \rangle$ charakterisiert werden.

- T ist eine Menge von Indexbegriffen, die Anfragen und Dokumente repräsentieren
- D ist eine Menge von Dokumenten. Jedes Dokument $d \in D$ wird dargestellt durch $\{(t_1, w_1), \dots, (t_n, w_n)\}$, wobei w_i die Gewichtung des Begriffes t_i im Dokument d ist und w_i jeden Wert zwischen 0 und 1 annehmen kann ($0 \leq w_i \leq 1$)
- Q ist eine Menge von Abfragen, die vom System erkannt werden können. Jede Abfrage $q \in Q$ ist ein gültiger boole'scher Ausdruck, zusammengesetzt aus Indexbegriffen und den boole'schen Operatoren AND, OR und NOT.

- F ist eine Rankingfunktion

$$F : D \times Q \rightarrow [0,1]$$

die einem jeden Paar (d,q) eine Zahl im geschlossenen Intervall $[0,1]$ zuweist.

Diese Zahl ist ein Maß für die Ähnlichkeit zwischen Dokument d und der Abfrage q und wird der "Dokumentenwert" für das Dokument d hinsichtlich der Abfrage q genannt. Die Funktion F ist wie folgt definiert:

1. Für jeden Begriff t_i in der Abfrage q definiert die Funktion $F(d, t_i)$ das Gewicht des Begriffes t_i im Dokument d , z.B. w_i
2. Boole'sche Operatoren* wie AND, OR und NOT werden bewertet, indem die zugehörigen mathematischen Formeln benutzt werden. Diese Berechnungsformeln der Operatoren sind ein wichtiger Faktor für die Qualität des Ranking.

In der Funktion $F()$ werden beim extended boole'schen Suchen die Operatoren definiert. In der einschlägigen Literatur wurden bereits viele Operatoren analysiert und bewertet. Die einzelnen Operatoren sind unter Namen wie "Fuzzy-Set", "Waller-Kraft", "Paice", "p-norm", "infinite-one", "network Boolean", "INQUERY Boolean", ... bekannt [LEE95b]. Die einzelnen Operatoren sind durch unterschiedlichste Problematiken ausgehend vom zugrundeliegenden mathematischen Modell geprägt. Zur Übersicht hier einige ausgewählte Operatoren.

Operator	AND	OR
Fuzzy-Set	$MIN(w_1, w_2)$	$MAX(w_1, w_2)$
INQUERY Boolean	$w_1 \cdot w_2$	$w_1 + w_2 - w_1 w_2$
infinite-one	$b \cdot MIN(w_1, \dots, w_n) + \frac{(1-b)(w_1 + \dots + w_n)}{n}$	$b \cdot MAX(w_1, \dots, w_n) + \frac{(1-b)(w_1 + \dots + w_n)}{n}$

Tabelle 4-11 verschiedene Operatoren des Extended Boole'schen Modells

Beim letzten Operator gilt $0 \leq b \leq 1$.

Die Berechnungsvorschriften für die unterschiedlichen Operatoren können beliebig komplex werden. Eine Abhandlung über die Qualität der einzelnen Operatoren im Vergleich, deren Vor- und Nachteile kann in [LEE95b] nachgelesen werden.

Problematisch an der Vorgehensweise, bekannte Operatoren mit neuer Semantik zu überladen, ist die Auffassung des Menschen. Manche der neueren Operatoren liefern

komplett von der Vorstellung des Menschen abweichende Resultate, die zwar mathematisch korrekt, aber nur sehr schwer nachvollziehbar bzw. steuerbar sind. Durch die Verwendung dieser komplexeren Operatoren wird aus dem Boole'schen Suchmodell ein "Partial-Match" Verfahren.

4.3.5 Expertensystembasierte Retrieval Methoden

[POO99b] Während obige Suchmethoden immer davon ausgehen, dass der Index über die Dokumente neu erstellt werden kann, stellen die "expertsystem based" Methoden den Investitionsschutz bereits erstellter Indexe in den Vordergrund.

Expertensystembasierte Suchmodelle werden darum hauptsächlich in OPAC's oder stark themenbezogenen Suchsystemen verwendet.

Bei diesen Systemen werden Forschungsergebnisse im Bereich der neuronalen Netze und Expertensysteme umgesetzt. Ziel der Verfahren ist es die vom Benutzer eingegebene Suchanfrage unter Verwendung von Agentenwissen bzw.

Expertensystemen in eine boole'sche Anfrage zu wandeln. Die Umformung soll dabei so geschehen, dass der Inhalt der Benutzeranfrage möglichst ohne semantische Verluste umgesetzt wird. Bei der Umsetzung wird ein Expertensystem eingesetzt, das das jeweils fachbezogene Wissen eines menschlichen Experten und dessen

Entscheidungsgrundlagen bzw. Lösungsverfahren codiert. Es wurden bereits mehrere Systeme entwickelt, die auf Basis von Produktionsregeln und semantischen Netzen arbeiten [POO99b].

- "Gauch's Query Reformulation" System verwendet Produktionsregeln, um die Qualität der Anfrage zu verbessern. Die boole'schen Operatoren werden verändert, oder verwandte Begriffe zur Abfrage hinzugefügt, oder verschiedene Begriffe durch breiter oder enger gefasste Begriffe aus einem Wörterbuch ersetzt. [GUA99]
- "PLEXUS" ist ein expertenbezogenes System, das Wörter aus dem Bereich "Garten" auf semantische Primitive abbildet. Es beinhaltet einige Suchstrategien, die als Produktionsregeln implementiert sind und baut ein temporäres Benutzerprofil über dessen Wissen zum Thema "Garten" auf. Basierend auf diesem Profil erhält der Benutzer seinem Kenntnisstand entsprechende Hilfestellung. [VIC87]

- "Rule Based Retrieval Information by Computer" kurz RUBIC, benutzt Produktionsregeln, um eine ganze Hierarchie an Suchstrategien bzw. Vorgehensweisen zu definieren. Die jeweilige Wissensbasis wird durch eine Ansammlung von Konzepten abgebildet. Jedes Konzept beinhaltet eine Beschreibung, die Beziehung zu anderen Konzepten und die Regeln, die das Textmuster beschreiben, die vorhanden sein müssen, um das gespeicherte Suchkonzept anzuwenden. Dieses System verlangt vom Benutzer, dass er systemkonforme Anfragen stellt. [TON87]
- Drabenstott und ihre Kollegen entwickelten eine prototypische Implementierung eines Online-Kataloges, der Such- oder Entscheidungsbaume verwendet, um abzubilden, wie erfahrene Bibliothekare eine Suchstrategie auswählen und formulieren davon abhängig einen Suchausdruck. Der Entscheidungsbaum wird durch ein Flussdiagramm dargestellt. [DRA96a, DRA96b, DRA96c]

4.3.6 Verwendung von fortgeschrittenen mathematischen Modellen

[AUT00] In diesem Kapitel werden die vom Autonomy Knowledge Server verwendeten Technologien und die Hintergründe näher erläutern.

Autonomy's Vorgehensweise unterscheidet sich fundamental von den vorig genannten. Autonomy will den Computer in die Lage bringen, Bedeutungen aus Texten zu extrahieren und anhand dieser Informationen die Einteilung in Kategorien zu verbessern. Der Computer soll Zusammenhänge und den Kontext "verstehen", von den Worten auf Ideen abstrahieren und die Kerngedanken zwischen den Zeilen aufnehmen. Autonomy's Unternehmensziel: "Oracle der unstrukturierten Daten" werden. Die verwendete Technologie adressiert vornehmlich die Verwaltung unstrukturierter Massendaten. [ATW00a, WIR00a] Autonomy will die Explosion unstrukturierter Information in Form von Texten handhabbar machen. Um dieses Ziel zu erreichen werden neuronale Netzwerke* und fortgeschrittene Mustervergleichsalgorithmen (non-linear adaptive digital signal processing) kombiniert verwendet. Diese Algorithmen fassen Dokumente digital zusammen und legen die Merkmale fest, die dem Text eine digital erfassbare Bedeutung geben. Bei Autonomy ist die Dynamic Reasoning Engine* (DRE*) für genau diese Aufgaben zuständig. Sie ist Grundlage aller Produkte von Autonomy.

Für eine Standardtextsuche gibt der Benutzer einfach eine boole'sche Abfrage, oder natürliche Sprache ein. Diese Abfrage wird an die DRE* zur Analyse übermittelt. Die DRE bearbeitet die Anfrage und gibt eine nach Relevanz geordnete Liste von Dokumenten zurück. Neben der "normalen" Suche unterstützt die DRE aber auch die Suche nach Konzepten und Ideen. Dadurch unterscheidet sich die Technologie zentral von den sonst üblichen Methoden. Der Benutzer umschreibt seine Idee, verwendet seine natürliche Sprache um die zu suchende Informationen zu definieren. Diese umfangreichere Abfrage wird von der DRE als Dokument interpretiert und digital erfasst. Anschließend sucht die DRE Dokumente mit dem höchsten Grade an Relevanz. Diese werden – geordnet nach der Relevanz – als Treffermenge zurückgegeben. Durch die Konzentration auf die Mustersuche ist die verwendete Technologie komplett sprachunabhängig. Wörter werden vom Computer als abstrakte Symbole mit Bedeutung angesehen. Die Bedeutung wird aus dem Vorkommen im Kontext abgeleitet, nicht aus einer einfachen Grammatikdefinition. Dialekte oder Abänderungen in der Sprache stören die verwendete Technologie nicht.

[ATW00b] Die verwendete Technologie grenzt sich stark von herkömmlichen Suchmethoden ab. Die reine Schlüsselwortsuche nach der boole'schen Methode kann Dokumente nur anhand der darin auftauchenden Wörter identifizieren, aber keine Aussage über die Relevanz zur Suche treffen. Genauso wenig kann eine Aussage getroffen werden, ob das Dokument thematisch etwas mit dem gesuchten Wort zu tun hat. Eine Suche nach Konzepten oder Ideen, die in natürlicher Sprache die gesuchte Information charakterisieren, ist nicht möglich, da in der boole'schen Suche mehr Suchwörter in der Eingabe zumeist mehr Dokumente als Treffer liefern. Zudem verwendet die boole'sche Suche einfache Relevanzkriterien, die jedoch oft fehlerhafte Resultate ergeben. Ein Romantext beispielsweise

" ... er ging auf der linken Seite der Straße die Straße hinunter ... die Straße war nass ... seine Schritte hallten auf der Straße ... er hörte Schritte auf der Straße ... er rannte die Straße hinab ... der Gegner erwischte ihn und ermordete ihn kaltblütig ..."

Im Text kommt das Wort "Straße" häufig vor, der Text handelt jedoch von einem Mord. Bei einer boole'schen Suche nach dem Wort "Straße" würde dieser Text sicherlich als relevant eingestuft. Bei der Suche nach "Mord" wohl eher nicht.

Ein anderer Ansatz Ideen und Konzepte aus Texten zu extrahieren ist die Verwendung von Parsern*, um natürliche Sprache zu "verstehen". Die unstrukturierte Information wird um Grammatikregeln und Lexikons ergänzt – wird also strukturiert. Mit diesen Hilfsmitteln wird versucht textuelle Information zu verstehen. Problematisch an diesem Ansatz ist die schlechte Performanz von Parsern mit komplexen Regeln. Zudem kann die Wichtigkeit von Ideen nicht bewertet werden. Der Parser trifft in seinem Entscheidungsbaum binäre Entscheidungen (wahr / falsch). Trifft der Parser aufgrund von Fehlern im Regelwerk oder im Parser einmal eine falsche Entscheidung, so ist der ganze Entscheidungsprozess gefährdet. Als größtes Manko ist die extreme Sprachabhängigkeit zu sehen. Für jede zu parsende Sprache muss ein eigenes Regelwerk und ein Lexikon generiert werden.

[ATW00, WIR00a] Nach diesen Bemerkungen allgemeiner Natur soll die Technologie von Autonomy mehr hinterleuchtet werden. Autonomy's Technologieansatz basiert auf relativ alten Annahmen und bereits anerkannten Gesetzmäßigkeiten. Claude Shannon's Prinzip der Informationstheorie, Thomas Bayes' Annahmen über Wahrscheinlichkeitsmodelle bei mehreren Unbekannten und Kenntnissen im Feld der neuronalen Netze ermöglichen die schnelle Musteridentifikation in Texten und anderen Quellen. Die Kombination dieser drei Technologien ermöglicht es aus einem Text die Schlüsselkonzepte zu extrahieren, weil Autonomy "versteh", wie die Häufigkeit und Beziehung zwischen Begriffen mit deren Bedeutung korreliert.

Shannon's Informationstheorie ist die mathematische Grundlage aller digitaler Kommunikationssysteme. Laut Shannon ist "Information ein quantifizierbarer Wert in der Kommunikation". In seinem Werk "The Mathematical Theory of Communication" geht Shannon genauer auf die Aspekte der Kommunikation und die Möglichkeit Information zu berechnen ein. Prinzipiell verfolgt Shannon jedoch die Theorie, dass je geringer das Auftreten einer Kommunikationseinheit (Wort, Satz) ist, desto informativer ist diese. Ideen, die seltener im Umfeld einer Kommunikation sind, tendieren daher eher

dazu, relevanter für die Bedeutung der Kommunikation zu sein. Unter Berücksichtigung dieser Aussagen ist es möglich die wichtigsten oder informativsten Konzepte in einem Dokument zu finden. Weiter soll an dieser Stelle nicht auf die Theorien von Claude Shannon eingegangen werden, da diese mathematisch fundiertes Wissen voraussetzen und trotz der Einfachheit der obigen Aussage doch nicht ganz einfach zu verstehen sind.

[ROU00] Thomas Bayes Theorem ist eine zentrale Säule moderner statistischer Wahrscheinlichkeitsmodelle. Seine Arbeiten konzentrierten sich auf die Berechnung von Wahrscheinlichkeitsbeziehungen zwischen mehreren Variablen und die Abhängigkeiten der Variablen untereinander. Bayes Theorem setzt Ereignisse der Gegenwart unter dem Wissen vergangener Ereignisse mit der in Zukunft zu erwartenden Ereignisse in Beziehung. Erst nach seinem Tod entdeckte ein Freund in seinen Aufzeichnungen die Arbeit "An Essay Towards Solving a Problem in the Doctrine of Chances". Dort skizziert Bayes ein Modell für die Vorhersage von Ereignissen unter Unsicherheitsbedingungen.

Die mathematische Formel

$$P(t | y) = \frac{P(y | t)P(t)}{P(y)}$$

drückt die Annahmen von Thomas Bayes exakt aus. Um die Formel auch von pragmatischer Sicht zu verstehen soll sie an folgendem Beispiel erläutert werden. Ein Koch arbeitet in einem belebten Straßencafe. Kellner rufen dem Koch Bestellungen zu. Dies geschieht vor einer lauten Geräuschkulisse aus Straßenlärm, klappernden Tellern und redenden Gästen. Was ein Gast tatsächlich bestellt hat ist in der obigen Gleichung t . Die verstümmelte Bestellung, die der Koch hört ist y . Ein bayes'scher Entscheidungsprozess würde dem Koch ermöglichen so viele Bestellungen wie möglich korrekt abzuarbeiten. Er hat noch ein hilfreiches Hintergrundwissen: Wenige Kunden bestellen Lachsschnitten, wohingegen Steak mit Pommes Frites ein richtiger Renner ist. Der Koch kann diese Art von Information verwenden, um die vergangene Wahrscheinlichkeit zu berechnen, dass ein geliefertes Essen bestellt wurde: $P(t)$ oder die Wahrscheinlichkeit von t . Zusätzlich weiß der Koch noch, dass ein Kellner Gerichte mit französischem Namen immer falsch ausspricht, ein Bus alle 10 Minuten vorbeifährt

und brutzelnde Bratpfannen den Unterschied zwischen z.B. Hammel und Semmel verwaschen. Nimmt er nun alle diese Faktoren zusammen, kann er ein komplexes Entscheidungsmodell erstellen, wie Bestellungen durcheinanderkommen können: $P(y/t)$ – die Wahrscheinlichkeit von y unter der Voraussetzung t . Durch Bayes' Theorem kann der Koch die Wahrscheinlichkeit $P(t/y)$ berechnen, dass unter Berücksichtigung aller äußeren Einflussfaktoren das, was er verstanden hat, auch dem entspricht, was der Gast bestellt hat. Der Koch jedoch verwendet in der Realität nicht mathematisch komplexe Modelle, sondern baut auf seine Erfahrung.

Bayes hatte damals, als er seine Theorien aufgestellt hat das Problem, dass er durch seine bescheidenen Mittel – Stift, Notizblock und die Zeit, die er für die Berechnungen benötigte – stark eingeschränkt wurde. Durchgesetzt haben sich seine Theorien erst, als Computer Millionen von Wahrscheinlichkeiten in einem Augenblick berechnen konnten.

Die Technologie wird für Aufgaben aus der Chaostheorie und Bedeutungsfindung eingesetzt. Dort beispielsweise in der Mustererkennung, Signalverarbeitung, Verbesserung der MP3-Qualität, Bilddatenbanken, Fingerabdruck-, Gesichts- und Handschrifterkennung, ... Die Computer können aus der Erfahrung der Vergangenheit lernen und dadurch Rückschlüsse auf zu bewertende Situationen ziehen.

4.4. Ein exemplarisches Beispiel: <http://www.webcrawler.com>

[PIN94] Einige der erwähnten Konzepte und Vorgehensweisen sollen an dieser Stelle durch die Internet-Suchmaschine „WebCrawler“ beispielhaft verdeutlicht werden. Der Autor und Betreiber dieser Suchmaschine geht mit den realisierten Konzepten weniger restriktiv um, als andere Suchmaschinenbetreiber. Die Vorgehensweise einer Suchmaschine und die Realisierung der Qualität der zurückgelieferten Abfragen sind streng gehütete Betriebsgeheimnisse von so bekannten Suchmaschinen wie „FireBall“, „HotBot“ oder anderen. Um so angenehmer ist es, dass Brian Pinkerton umfangreiche Informationen über die Struktur des WebCrawlers öffentlich zur Verfügung stellt. Leider beschreibt das Dokument eine ältere Version des WebCrawlers. Anschließend wird kurz die Problematik beschrieben, die der WebCrawler zu lösen versucht und darauf folgend die Suchmaschine genauer beschrieben.

4.4.1 Allgemeiner Überblick

Benutzer des Internets finden Informationen, indem sie Hypertextlinks* folgen. Mit steigender Größe der Datenbasis muss der Benutzer immer mehr Links folgen, um sein Ziel zu erreichen. Der WebCrawler hilft dem Benutzer dadurch, dass er automatisch durch das Internet wandert und Seiten indexiert. Der WebCrawler bietet einen öffentlichen Zugang zum Index und eine Realtiesuche für autorisierte Benutzer. Ein WebRobot wartet den Index durch laufende Aktualisierung. Der WebCrawler bildet durch eine inhaltsbasierte Indexierung einen hochqualitativen Index.

4.4.2 Architektur* WebCrawler

Der WebCrawler folgt grundsätzlich dem gleichen Designansatz, wie das Internet – ein dezentralisiertes Design.

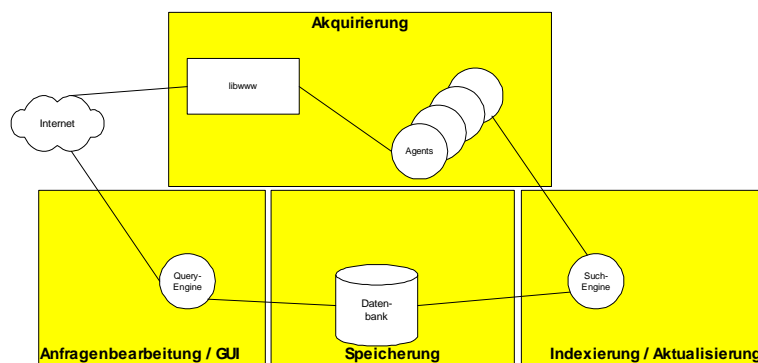


Abbildung 4-1 Architektur* des WebCrawlers

Die ganze Architektur* orientiert sich an den Aufgaben, die grundsätzlich von einer Suchmaschine zu erfüllen sind. Die Aufgaben sind folgende:

- Akquirierung der Dokumente
- Indexierung der Dokumente und Aktualisierung des Indexes
- Speicherung der Informationen
- Anfragenbearbeitung und grafische Benutzerschnittstelle

Für die einzelnen Aufgaben sind jeweils verschiedene Kernkomponenten zuständig. Die SuchEngine steuert die WebCrawler-Aktivitäten und ist zuständig dafür, welche neuen Dokumente untersucht werden sollen. Die SuchEngine stößt auch die Akquirierung der Dokumente durch die Agents an. In der SuchEngine wird die weitere Vorgehensweise für jedes einzelne Dokument entschieden. Die Datenbank speichert Dokumenten-Metadaten, die Verbindungen zwischen den Dokumenten und dem Volltextindex

darstellen. Die Agents sind dafür verantwortlich, die von der Suchmaschine beauftragten Dokumente aus dem Netz zu holen. Die QueryEngine bietet den Abfrageservice im Netz an.

4.4.3 SuchEngine

Die SuchEngine ist für die Indexierung der Dokumente und die Aktualisierung des Indexes zuständig. Die SuchEngine verfolgt dabei folgende Vorgehensweise.

Ausgehend von einer bestehenden Dokumentenmenge werden die nach extern führenden Links* extrahiert und diese Links verfolgt, die zu einem neuen Dokument führen. Neue Dokumente werden zum Index hinzugefügt und anschließend wiederholt sich der gesamte Prozess. Die Suchmaschine entscheidet nicht nur, welche Dokumente besucht werden sollen, sondern auch die Arten von Dokumenten. Nicht indexierbare Dokumente – z.B. Bilder oder nicht unterstützte Dateiformate - werden ignoriert und erst gar nicht abgerufen. Die SuchEngine unterstützt generell zwei Arbeitsweisen. Den Indexing Modus und die Realtime Suche.

4.4.3.1. Indexing Mode

Ziel der Indexierung ist es einen möglichst umfassenden Webindex aufzubauen. Die verfolgte Strategie ist ein "breadth-first"-Algorithmus, um sicherzustellen, dass jeder Server mindestens ein Dokument im Index hat. Dabei wird jedes Mal, wenn ein "neuer" Server gefunden wurde dieser Server* in die Liste der zu besuchenden Server gestellt. Bevor weitere Dokumente geholt werden, wird von dem neuen Server ein Dokument geholt und indexiert. Wurden alle bekannten Server besucht, durchsucht der WebCrawler sequentiell die Liste der zu besuchenden Server nach neuen Servern und die Prozedur wiederholt sich.

4.4.3.2. Realtime Suche

Bei der Realtime Suche will der WebCrawler den Benutzer dahingehend bei der Informationsfindung unterstützen, indem er roboterartig das Internet nach relevanten Informationen durchsucht. Das geschieht dann nicht im eigenen Index, sondern "online" im Internet. Zunächst wird eine Abfrage gegen den eigenen Index gestartet, um eine Anfangsdokumentenliste oder auch nur ein Dokument zu finden. Aus der Liste werden die relevantesten markiert und die unerforschten verfolgt. Werden neue Dokumente gefunden, so werden diese zum Index dazugefügt und die Abfrage wird wiederholt.

Dieser Prozess wird solange wiederholt, bis der WebCrawler für den Benutzer genügend Dokumente gefunden hat, oder ein Zeitlimit erreicht wurde. Problematisch ist, dass der WebCrawler jedem Link folgt. Dabei kann eventuell ein irreführender Pfad entstehen, da als Maß für die Relevanz die Ähnlichkeit zwischen dem "Anchor-Text"* und der Abfrage angenommen wird. Anchor-Texte* beschreiben aber äußerst kurz, wenn überhaupt, wohin sie führen und werden dann gegen einen Volltextindex abgeglichen.

4.4.4 Agents

Die Agents sind Prozesse* mit der festgelegten Aufgabe, die Dokumente aus dem Internet zu holen, die von der SuchEngine zur weiteren Bearbeitung angefordert werden. Zur Realisierung dieser Aufgabe verwenden die Agents die "libwww"* vom W3C-Konsortium als Basistechnologie. Der WebCrawler verwendet maximal 15 solcher Agenten.

4.4.5 Datenbank

Die Datenbank ist der Informationsspeicher schlechthin. Sie enthält den Volltextindex und eine Repräsentation des Internet als Graph. Der Index wird nach dem Vector Space Modell* gespeichert. Um Anfragen an den Index zu beschleunigen wird der Index invertiert gespeichert. Indexiert werden der Dokumententitel und auch der Dokumenteninhalt. Bei der Indexierung unterteilt ein lexikalischer Analyzer* das Dokument in einen Wortstrom, der Tokens* aus dem Titel und dem Dokumenteninhalt enthält. Dieser Strom wird durch eine sogenannte "Stop-List"* gefiltert. Die Stop-List beinhaltet gewöhnliche und gebräuchliche Wörter, die keinen Informationsgehalt haben. Die übrigen "wichtigen" Wörter werden nach der inversen Dokumentenhäufigkeit – wie in 4.3.2 bereits beschrieben – gewichtet. Neben dem Volltextindex werden in der Datenbank auch Metainformationen über die indexierten Dokumente gespeichert. Zu den einzelnen Dokumenten werden Informationen über Links, Server und URL gespeichert. Die URL wird nicht textuell gespeichert, sondern aufgespalten in Objekte. Jedes URL Objekt wird letztlich als btree* gespeichert um den Zugriff auf unbesuchte Server zu vereinfachen.

4.4.6 QueryEngine

Die QueryEngine stellt die Schnittstelle zum Index im WWW dar. Der Benutzer kann eine Anfrage eintippen und überlässt der QueryEngine die weitere Bearbeitung. Das

Abfragemodell entspricht der Vorgehensweise beim Vector Space Modell auf einer Volltextdatenbank. Als Relevanzkriterium wird ein einfaches, leistungsstarkes Modell verwendet. Die Relevanz r wird nach der folgenden Formel ermittelt

$$r = \frac{(\text{alle Wörter in der Abfrage}) + (\text{Wortgewicht im Dokument}) + (\text{Gewichtung in der Abfrage})}{\text{Anzahl der Wörter in der Abfrage}}$$

Die Bestimmung der Relevanz erfolgt nach einem einfachen Modell. Wie aber die Erfahrung zeigt, sind einfache Modelle im Endergebnis meist mindestens genauso gut, wie komplexere Modelle.

4.4.7 Fazit

Zentraler Ansatzpunkt des WebCrawlers ist die inhaltsbasierte Indexierung. Es stellt sich heraus, dass dieser Ansatz der einzig gangbare ist, um einen guten Index zu erhalten. Viele andere Suchmaschinen verfolgten nur die Indexierung der Titel von Dokumenten. Problematisch daran ist, dass 20% aller Seiten entweder keinen Titel haben, oder nur einen nichtssagenden. Die Qualität des Indexes leidet an dieser Stelle natürlich erheblich. Andererseits ist es erstaunlich, dass der verfolgte "breadth-first"-Ansatz einen brauchbaren, qualitativ hochwertigen Index ergibt. Es werden ja eigentlich pro Server im Internet nur eine Seite gespeichert und die Durchdringung bei Gelegenheit intensiviert. Ebenfalls aufgefallen ist, dass die Realtime Suche eine extrem ressourcenfressende Angelegenheit ist. Für jede Anfrage wird das Internet mit Anfragen an die einzelnen Server überhäuft. Das ist auch der Grund dafür, dass die Realtime Suche nicht mehr angeboten wird.

5. Mögliche Lösungswege

Nachdem das bestehende System in der Ist-System-Analyse genauer untersucht und die Schwachpunkte aufgezeigt und die Ziele des Soll-Modells definiert wurden, sollen nun mögliche Lösungswege aufgezeigt werden.

Grundsätzlich bleibt an dieser Stelle nur die Wahl zwischen einem Standardsoftwareprodukt oder einer Eigenentwicklung der Suche. Um die kommerziellen Produkte zu finden, die prinzipiell in der Lage sind, die in 3 definierten Forderungen zu erfüllen, muss der Markt sondiert und genauer durchleuchtet werden. Schwierig an der Fixierung kommerzieller Produkte ist die extern konzipierte Verschlagwortung. Das Konzept für die Verschlagwortung stellt einen Spezialfall dar. Kommerzielle Produkte decken als Standardsoftware in diesem Bereich "nur" allgemeine Fälle ab. Dadurch disqualifizieren sich natürlich viele Softwareprodukte als Lösungsmöglichkeiten.

Als prinzipielle Lösungsmöglichkeiten stellen sich folgende Szenarien dar:

- Suchdaten- und Nutzdatenbestand gemeinsam in einer Datenbank
- Suchdaten- und Nutzdatenbestand in unterschiedlichen Systemen

Diese grundsätzlichen Ansätze lassen sich mit kommerziellen Produkten, als auch mit einer Eigenentwicklung abdecken. Die Eigenentwicklung hat in jedem Fall den Vorteil, dass das extern erstellte Verschlagwortungskonzept zu 100% umgesetzt werden kann. Zudem steckt in der aktuellen Implementierung der Suche noch enormes Verbesserungspotential. Kommerzielle Produkte haben auch Probleme mit dem extrem heterogenen Produktbestand der Conrad.com AG. Die Eigenschaften der einzelnen Produkte unterscheiden sich oft sehr stark voneinander. Viele kommerzielle Produkte gehen zwar von einem großen Artikelaufkommen aus, können aber meist nur ähnlich strukturierte Artikel verwalten.

Grundsätzlich stellt sich die Frage, wie sich der Weg zur Lösung, die den Anforderungen entspricht, darstellt. Eine Vorgehensweise ist die Ausarbeitung einer Matrix (siehe dazu Tabelle 6-1), die die Anforderungen an das Produkt enthält. Diese wird nach der Ausarbeitung sukzessive ausgefüllt und je nach Deckungsgrad das geeignetste Produkt ausgewählt. Bleibt noch die Frage nach dem Vorgehen. Für jedes der einzelnen Produkte muss eine Installation vorgenommen und Daten bereitgestellt

werden. Dabei muss jeweils ein funktionsfähiger Prototyp implementiert werden. Dieser kann anschließend bewertet und geprüft werden.

Nachfolgend werden die einzelnen prinzipiellen Möglichkeiten näher erläutert und bewertet. Daran anschließend werden die gewählten kommerziellen Systeme vorgestellt und bewertet. Den kommerziellen Produkte werden verschiedene Varianten von Eigenentwicklungen gegenüber gestellt und letztlich versucht die möglichst beste Lösung zu finden.

5.1. Nutzdaten- und Suchdatenbestand in einer gemeinsamen Datenbank

Grundsätzlich ist es ein gangbarer Weg die bestehende Datenbank für die Nutzdatenhaltung und die Suchdatenhaltung zu verwenden. Dieser Weg könnte sogar auf Basis der verwendeten Sybase Datenbank beschriftet werden. Zwingend notwendig wäre aber an dieser Stelle eine Änderung der Applikation und auch des zugrunde liegenden Datenmodells. Diese Änderungen sind unabhängig davon, ob ein kommerzielles System zum Einsatz kommt, oder aber die Applikation im Sinne einer Eigenentwicklung angepasst wird. Als kommerzielles System kommt eigentlich nur der Datenbankserver* der Firma Sybase in Frage. Dieser bietet für die Umsetzung der Schlagwortsuche aber keinerlei Ansatzmöglichkeiten an. Da dies aber ein sehr zentraler Aspekt der Neuimplementierung ist, fällt diese Möglichkeit sofort unter den Tisch. Andere Systeme auf die Sybase-Datenbank auf den gleichen Datenbestand aufzusetzen erscheint anhand der damit verbundenen Migrations- und Integrationsschwierigkeiten nicht als ratsam.

Neben den kommerziellen Systemen bleibt noch die Reimplementierung der Suche bzw. das Tuning der bestehenden Suche. Problematisch an diesem Ansatz bleibt jedoch weiterhin, dass grundsätzlich auf diese Weise keinerlei Lastverteilung erreicht werden kann. Skalierungsmöglichkeiten existieren mit diesem Modell ebenfalls nicht. Sieht man die Applikation getrennt in Shoppingapplikation und Suchapplikation und belässt beide auf einem Rechner, so existiert immer noch ein "Single Point of Failure". Fällt die Suche aus, so entfällt die komplette Suchfunktionalität aus der Shoppingapplikation. Eine Skalierung im Sinne der Erreichbarkeit bzw. Zuverlässigkeit wird also nicht erreicht. Bestehende Redundanzen, die zu Sicherungs- bzw. Backupzwecken sowieso existieren (sollten) werden nicht genutzt. Die Suche könnte durchaus auf den Backupserver der Datenbank ausgelagert werden. Die redundante Datenbankmaschine,

die eigentlich nur "vor sich hinläuft" kann für die Auslagerung der Suche verwendet werden.

Ein ganz gewichtiges Argument gegen diese Konstellation ist die aktuell gängige Vorgehensweise bei Systemüberlast. Ist der Datenbankserver* überlastet, so wird die Suchfunktionalität einfach abgeschaltet. Diese Vorgehensweise hat sich "bewährt" und wird wohl auch in Zukunft bei Systemüberlast angewendet werden. Damit wäre allerdings keine grundlegende Verbesserung hinsichtlich der Erreichbarkeit und Zuverlässigkeit erreicht.

5.2. Nutzdaten- und Suchdatenbestand getrennt in unterschiedlichen Systemen

Ganz wesentlicher Vorteil dieses Ansatzes ist es, dass die Last auf den Datenbankserver* durch den Einsatz dedizierter Server für die Suchapplikation und die Shoppingapplikation besser skalierbar ist. Bei der Aufteilung der Last auf dedizierte Server wird der aktuell überbelastete Applikationsdatenbankserver entlastet. Dadurch wird die insgesamt angespannte Situation entschärft. Neben der Entlastung des Datenbankservers* wird auch der momentan existierende "Single Point of Failure"* entschärft. Die Suchapplikation wird auf einen oder mehrere dedizierte Server ausgelagert. Fällt nun ein Suchserver aus, so können weitere Server die Aufgabe des einzelnen übernehmen. Außerdem ist das Ausfallen des Suchservers nicht direkt mit dem Ausfallen des Shopservers gekoppelt. Fällt nur die Suche aus, so ist die restliche Applikation eigentlich nicht betroffen. Ganz entscheidend ist die Tatsache, dass die Suchapplikation rein von der Technikbasis nichts mehr mit der existierenden Shoppingapplikation zu tun hat und auf andere Technologien gesetzt werden kann. Der Einsatz eines spezialisierten Suchsystems wird dadurch möglich. Die Suchapplikation ist durch den Einsatz dedizierter Server unabhängig von der alten Shoppingapplikation. Durch die Auslagerung der Suche wird die vorhandene Hardware optimaler ausgenutzt, als es bis jetzt der Fall ist. Wird ein kommerzielles System für die Auslagerung der Suche verwendet, so ist zu bedenken, dass eventuell zusätzlich Hardware und sicherlich Softwarelizenzen zu beschaffen sind.

Nachfolgend werden nacheinander die Verwendung kommerzieller Produkte und die Möglichkeiten einer Eigenentwicklung erläutert.

5.2.1 Verwendung kommerzieller Systeme

Die Titelüberschrift dieses Kapitels hört sich einfacher an, als es sich schließlich in die Praxis umsetzen lässt. Die Evaluierung eines kommerziellen Produktes ist eine extrem zeitaufwendige Angelegenheit. Nachdem die Soll-Kriterien an das neue Zielsystem definiert sind, können von verschiedensten Unternehmen Informationen zu möglichen Lösungskandidaten beschafft werden. Hierbei ist das Internet eine gute Informations- und Impulsquelle. Durch den Einsatz diverser Suchmaschinen, aber auch durch Mund-zu-Mund-Propaganda in der Conrad.com AG wurden letztlich zwei mögliche Kandidaten aus dem Überangebot an Möglichkeiten extrahiert. Nach der Marktsondierung muss ein erster Kontakt zur Vertriebsmannschaft der Zielfirma – hier die Firma "Autonomy" und die "Software AG" – geknüpft werden. Nach mehreren Terminen mit den verschiedenen Vertriebsleuten der Firmen wurde uns zu Evaluationszwecken jeweils eine Kopie der Software überlassen. Weder die Firma Autonomy, noch die Software AG haben es sich nehmen lassen einen Prototypen für die Conrad.com AG zu bauen. Bei der Beschaffung von Hintergrundinformationen über die jeweils verwendete Technologie sind die jeweiligen Vertriebsbeauftragten und auch das Internet eine wertvolle Informationsquelle. Neben der Beschaffung der kommerziellen Produkte ist auch die Datenbeschaffung für die zu bauenden Prototypen eine nicht zu vernachlässigende Aufgabe. Ebenso die Analyse für die spätere mögliche Integration des kommerziellen Produktes in die bestehende Applikation stellt ein Problempunkt dar.

Zunächst sollen die wesentlichen Vor- und Nachteile der Verwendung eines kommerziellen Standardproduktes herausgestellt werden.

Kommerzielle Produkte konzentrieren starkes Entwicklungs-Know-How auf eine dedizierte Aufgabe. Das bedeutet, dass technologische und konzeptuelle Erfahrung auf dem Gebiet der Suche in das Standardprodukt einfließt und damit jeder Eigenentwicklung in Bezug auf Fortschritt, Performanz und wohl auch Fehlerfreiheit überlegen ist. Bei Standardprodukten sind die Probleme und die Leistungsfähigkeit schon im Vorfeld einer Implementierung präzise vorherzusagen. Referenzkunden und –installationen geben wichtige Informationen über das Verhalten des Produktes in Grenzbereichen oder auch nur im tagtäglichen Einsatz. Diese Punkte sind bei der Eigenentwicklung erst im Nachhinein hinreichend bekannt.

Die obigen Vorteile bedingen natürlich auch einige Nachteile. Mit dem Know-How im Bereich der Suche können nicht alle Spezialfälle erschlagen werden. Das meist starre Konzept im Bereich der Suche deckt ungefähr 80% aller Standardfälle ab. Vielen Shopbetreibern reicht eine performante Volltextsuche. Eine manuelle Verschlagwortung wird meist gar nicht abgedeckt. Die Funktionalität ist durch das Standardprodukt fix vorgegeben und bietet eigentlich kaum Spielraum für Varianten. Diese Starrheit verbietet technologische Vorsprünge im Hinblick auf den Markt. Ein Alleinstellungsmerkmal kann mit einem Standardprodukt selten erreicht werden. Innovationen müssen eigentlich immer selbst implementiert werden.

5.2.1.1. Autonomy Knowledge Server

[ATW00a, ATW00c, ADC99a, ADC99b] Nachdem in 4.3.6 die Technik, die hinter den Produkten von Autonomy steckt, vorgestellt wurde, soll hier das Vorgehen bei der Evaluierung im Vordergrund stehen. Im Umfeld der Evaluierung wurden folgende Produkte von Autonomy benutzt und getestet:

- Knowledge Server
- Autoindexer
- ODBC-Fetch
- KnowledgeUpdate

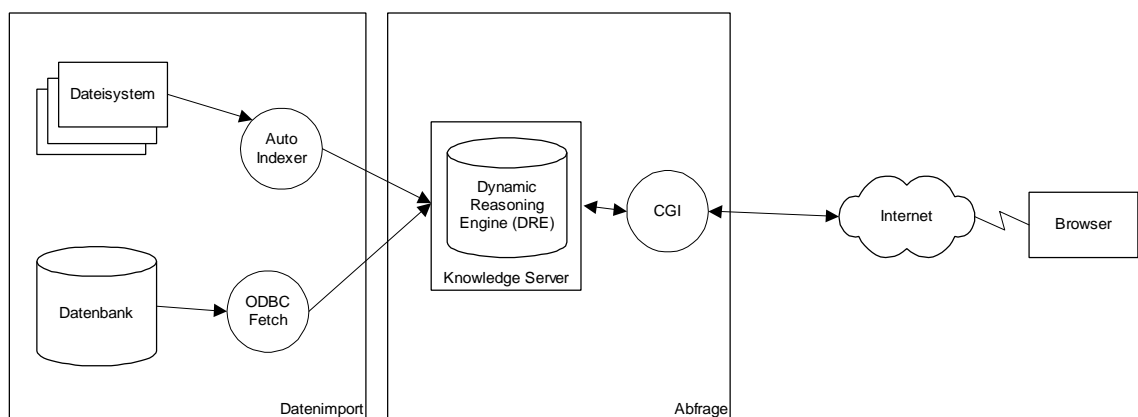


Abbildung 5-1 Architektur^{*} des Autonomy Knowledge Servers / Interaktion der einzelnen Komponenten

Der Knowledge Server beherbergt die **Dynamic Reasoning Engine (DRE)**, das Herzstück Autonomy Technologie. In der DRE werden die Dokumente indiziert und Informationen darüber in einem proprietären^{*} Datenformat vorgehalten. Die DRE

enthält auch die Vergleichstechnologie für die Suchfunktionalität. Der Knowledge Server verfügt über ein WWW Frontend auf CGI-Basis*. Abfragen an den Knowledge Server werden über das HTTP-Protokoll* vorgenommen.

Der **Autoindexer** ist ein Dienst, der in bestimmten Intervallen definierte Unterverzeichnisse nach neuen Dokumenten untersucht und diese in die DRE aufnimmt. Der Autoindexer ist eine Art Aktualisierungsservice für den Index / die DRE.

ODBC-Fetch ist ein Programm, mit dessen Hilfe Datenbankinhalte in die DRE aufgenommen werden können. Dabei geht das Programm so vor, dass Datenbankinhalte in Templates* eingefügt und als Dateien gespeichert werden. Diese können anschließend mit dem Autoindexer in die DRE eingefügt werden. Für die Datenbankinhalte werden in der DRE Referenzen auf ein CGI-Programm* gespeichert, das im Abfragefall die Daten aus der Datenbank herausholt und somit immer datenbankaktuell darstellt.

KnowlegdeUpdate ist eine Gruppe an Programmen, mit denen benutzerspezifische Agenten erzeugt und konfiguriert werden können. Im Rahmen dieser Programme können Abfragen gespeichert, verfeinert, rekonfiguriert, gelöscht usw. werden.

KnowledgeUpdate verfügt über ein WWW Frontend auf CGI-Basis*.

Nach der Vorstellung der einzelnen Produkte wird nun der Evaluierungsvorgang beschrieben und letztlich ein kurzes Fazit gezogen.

Datenbeschaffung aus dem Multimedia-Management-System

Die Daten, die für die Evaluierung verwendet wurden, kommen direkt aus dem Multimedia-Management-System (MMS) von SAP. Um die Daten leichter handhaben zu können, werden die Daten aus der ASCII-Textdatei in eine MS-Access Datenbankdatei importiert. Dies stellt im Hause Conrad.com AG eine Standardprozedur dar. Diese Datenbank wurde als Ausgangsmaterial übernommen.

Die zugrundeliegende Datenbank stellt einen Teilauszug aus der kompletten Datenbasis des Conrad.com AG – Webauftrittes dar. Berücksichtigt wurden lediglich Daten aus dem deutschen Internetshop "www.conrad.de". Die Datenbankstruktur stellt einen Ausschnitt aus der real verwendeten Datenbank dar. Für die Evaluation des Autonomy KnowledgeServers hingegen sind diese Attribute der Datenbank vollkommen ausreichend. Weitergehende Informationen, die aus dem MMS der Conrad Electronic und aus dem SAP-Warenwirtschaftssystem abrufbar sind, sind für die Suche irrelevant.

Im Anhang D ist der Diplomarbeit ein Auszug aus der Artikeldatenbank beigelegt. Der Auszug soll die folgende Tabellenbeschreibung mit Daten beispielhaft ausfüllen.

Feldname	Felddatentyp	Beschreibung
pos_name	Zahl	Artikelnummer
lay_header_06	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_header_05	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_header_04	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_header_03	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_header_02	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_header_01	Text	Allgemein gültiger Kategorienname, z.B. für Aufbau der Bereichssuche
lay_type_01	Text	Kennzeichen für Tabellenartikel ("TB")
lay_text_01	Memo	Tabellenartikel-Beschreibung
lay_ftext_01	Memo	Tabellenartikel-Beschreibung
pos_unit	Text	Mengeneinheit
pos_header	Text	Artikelbezeichnung
pos_text	Memo	Artikelbeschreibung
pos_acc01	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc02	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc03	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc04	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc05	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc06	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
pos_acc07	Memo	Zubehörartikelnummer und Zuberhörartikeltext durch <TAB> getrennt
id	Autowert	indexierbarer eindeutiger Wert

Tabelle 5-1 Beschreibung der einzelnen Artikelattribute, die für die Suche von Bedeutung sind

Installation und Konfiguration des Apache WebServers unter Windows NT

Nach der Konfiguration der Datei "httpd.conf" startet der Apache WebServer, der das Frontend von Autonomy KnowledgeServer darstellt. Nach der Installation der Autonomy-Komponenten wird der Apache WebServer als Service unter NT zur Verfügung gestellt.

Konfiguration und Installation des KnowledgeServers unter NT

Die Installation ist ein geführtes Setup-Programm. Die zu Autonomy gehörende Dokumentation geht eingehend auf die Installation ein. Wichtig bei der Installation ist die Festlegung der einzelnen Ports, die später die Dienste des KnowledgeServers nach außen hin veröffentlichen.

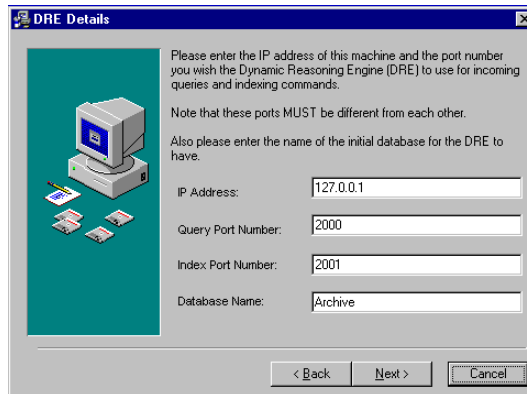


Abbildung 5-2 Installation des KnowledgeServers von Autonomy - Auswahl der Ports

Die IP-Adresse ist die IP-Adresse des KnowledgeServers. Der Query Port ist derjenige Port, an den die HTTP-Requests gesendet werden, die als Inhalt eine Anfrage an den KnowledgeServer haben. Der Index Port wird zur Neuaufnahme von Dokumenten verwendet. Der Datenbankname ist der Name der zu erstellenden Instanz.

Anschließend werden noch einige webserverrelevanten Daten abgefragt und die Installation geht automatisiert zu Ende. Anschließend steht der Autonomy KnowledgeServer als Dienst unter NT zur Verfügung.

Nach der Installation des Autonomy KnowledgeServers kann der Dienst gestartet werden. Mit einem mitgelieferten Administrationstool kann der KnowledgeServer zu Administrationszwecken abgefragt werden. Nach dem Start ist die zugrundeliegende Datenbank natürlich leer.

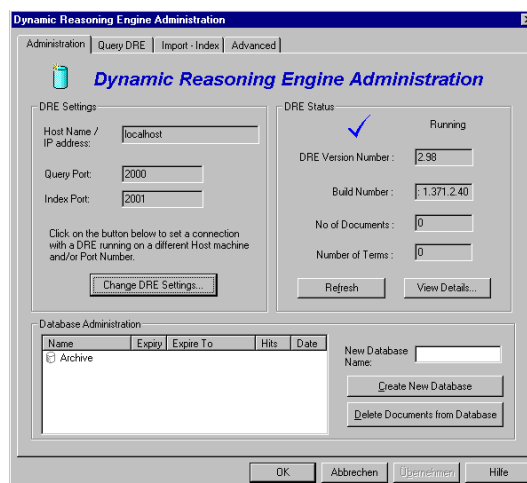


Abbildung 5-3 Administrationstool zum Zugriff auf die DRE - leere Datenbank

Verknüpfung der Artikeldatenbank via ODBC (DSN)

In der Systemsteuerung wird via ODBC-Kontrolle ein System DSN* (Data Source Name) eingerichtet. Hier wird die oben erwähnte MS-Access-Datenbank verwendet, die die Artikeldaten enthält.

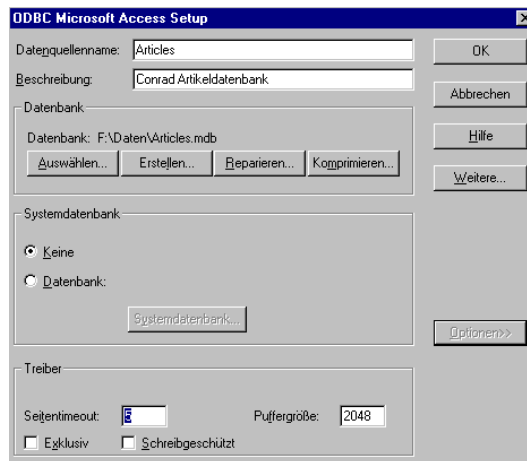


Abbildung 5-4 Einrichten eines systemweiten ODBC-Synonyms für die Artikeldatenbank

Installation und Konfiguration des ODBCFetch-Services unter NT

Die Installation des ODBC-Fetch-Services geht genauso einfach, wie die Installation des KnowledgeServers. Bei der Installation werden hier die obig konfigurierten Ports als Eingabe erwartet. Mit der Eingabe der korrekten Ports wird die Verbindung zum KnowledgeServer hergestellt.

Prinzipiell wäre es zwar möglich zyklische Fetches über die Datenbank laufen zu lassen, aber zu Testzwecken ist hier lediglich der Import der Datenbank in die DRE wichtig.

Die nachfolgenden Schritte konfigurieren den ODBCFetch. Zunächst wird die Datei "ODBCFetch.cfg" angepasst.

```
[License]
Holder=autonomy
Key=816908461V90231964036Q19076069

[Default]

// Dre information
DREIndexPort=2001
DreHost=127.0.0.1
```

```

// The DRE database that the information is to be indexed into
Database=ODBC
FetchStartTime=12:00
FetchRepeatSecs=86400
FetchCycles=1

// Fields which go in as meta data WITH THE SAME VALUES FOR EACH DOCUMENT
// FixedFieldName0=<Name>
// FixedFieldValue0=<Value>
// ..
// FixedFieldNameN=<Name>
// FixedFieldValueN=<Value>

// The name and delimiters of fields the values of which are to be extracted from the
content
// FieldName0=<Name>
// FieldStart0=<Start String>
// FieldStop0=<End String>
// ..
// FieldNameN=<Name>
// FieldStartN=<Start String>
// FieldStopN=<End String>

// Whether summaries are to be stored at indexing time
ImportSummary=FALSE

// Parameters controlling whether individual documents are indexed -- min and max sizes
// in both words and bytes
ImportMinLength=0
ImportMinLengthWords=0
ImportMaxLength=100000
ImportMaxLengthWords=100000

// Whether large documents are to be split into sections in the idx file
// (requires COMBINE=1 in the DRE.INI file)
ImportBreaking=FALSE
ImportRecursiveDirectoryImporting=TRUE

// Parameters controlling document splitting
// ImportBreakingMinParagraphWords=<MinWordsInPara>
// ImportBreakingMaxParagraphWords=<MaxWordsInPara>
// ImportBreakingMinDocWords=<MinWordsInDocSection>

ImportPath=f:\autonomy\ODBCFetch
IndexPath=f:\autonomy\ODBCFetch
ODBCIndexPath=f:\autonomy\ODBCFetch

// Fetches

[Fetch]
FetchName0=MichaelsFetch

[MichaelsFetch]
ODBC=DSN=Articles;UID=;PWD=
Parser=michaelsfetch.cfg

```

Ausschnitt 5-1 Konfigurationsdatei für den ODBCFetch – "ODBCFetch.cfg"

Die geänderten Stellen im Konfigurationsfile sind fett hervorgehoben. Die erste Änderung legt das Startverhalten des Dienstes fest. Der Dienst startet spätestens um 12:00 Uhr (FetchStartTime) und läuft dann exakt einmal (FetchCycles). Die folgenden Änderungen legen den Import- und Indexpfad fest. Diese Einstellung ist für den Datenaustausch mit dem KnowledgeServer wichtig (ImportPath & IndexPath). Anschließend wird ein Fetchlauf namens "MichaelsFetch" (FetchName0) definiert. Hier könnten mehrere Datenquellen via ODBC eingebunden werden. Abschließend wird die

Datenquelle "Articles" definiert (ODBC) und das zu parsende Konfigurationsfile für diesen Fetch festgelegt (Parser).

```
[Configuration]

// Tables
NumTables=1
TableName0=complete

// Prefix of file to create "MsFxxx.xxx"
DataType=MsF

// Directory to store temporary created files
BaseDirectory=MichaelsFetch

PostProcess=true

// Generate one file instead of a file for each record.
SingleFile=false
//OutputFile=Example.idx

// Generate an index file
Extension=.htm

MainTable=complete
PrimaryKeyFields=1
PrimaryKeyName0=id
PrimaryKeyType0=Unquoted

// Tables

[complete]
// The name of the table in the database.
Target=f:\daten\articles.mdb..complete

// Uniquely identify a row.
PrimaryKeyFields=1
PrimaryKeyName0=id
PrimaryKeyType0=Unquoted
Template=complete.tmpl
```

Ausschnitt 5-2 Konfigurationsdatei "michaelsfetch.cfg"

Die Konfiguration des Fetch-Prozesses geht ähnlich vonstatten, wie die Basiskonfiguration des ODBCFetch. Zunächst wird die Anzahl der zu verarbeitenden Tabellen auf 1 festgelegt (NumTables) und der Name der Tabelle festgelegt (TableName0). Anschließend wird ein Präfix für die zu generierenden temporären Dateien und das temporäre Verzeichnis für die zu generierenden Daten festgelegt (DataType & BaseDirectory). Weiter unten in der Datei wird die Haupttabelle (MainTable) festgelegt und der Name des Primärschlüsselattributes (PrimaryKeyName0) definiert. In der folgenden Sektion ([complete]) wird noch einmal der genaue Pfad zur Datenbank definiert (Target) und das Primärschlüsselattribut für die Tabelle definiert (PrimaryKeyName0). Zum Abschluss wird das Template definiert, das mit den Daten aus der Datenbank ausgefüllt wird.

Das Template "complete.tmpl" dient zur Positionierung der einzelnen Datenbankattribute in den zu erzeugenden HTML-Seiten.

```
<META NAME="DREREFERENCE" CONTENT="http://127.0.0.1/cgi-  
bin/ODBCFetch.exe?Method=FetchData&keyID=
```

```

F:\Autonomy\ODBCFetch\odbcfetch
Creating subdirectories
Starting data generation
Executing primary key select statement
Starting data generation
Created file: MichaelFetch\49\MsP1.htm
Created file: MichaelFetch\50\MsP2.htm
Created file: MichaelFetch\51\MsP3.htm
Created file: MichaelFetch\52\MsP4.htm
Created file: MichaelFetch\53\MsP5.htm
Created file: MichaelFetch\54\MsP6.htm
Created file: MichaelFetch\55\MsP7.htm
Created file: MichaelFetch\56\MsP8.htm
Created file: MichaelFetch\57\MsP9.htm
Created file: MichaelFetch\58\MsP10.htm
Created file: MichaelFetch\59\MsP11.htm
Created file: MichaelFetch\60\MsP12.htm
Created file: MichaelFetch\61\MsP13.htm
Created file: MichaelFetch\62\MsP14.htm
Created file: MichaelFetch\63\MsP15.htm
Created file: MichaelFetch\64\MsP16.htm
Created file: MichaelFetch\65\MsP17.htm
Created file: MichaelFetch\66\MsP18.htm
Created file: MichaelFetch\67\MsP19.htm
Created file: MichaelFetch\68\MsP20.htm
Created file: MichaelFetch\69\MsP21.htm
Created file: MichaelFetch\70\MsP22.htm
Created file: MichaelFetch\71\MsP23.htm
Created file: MichaelFetch\72\MsP24.htm

```

Abbildung 5-5 ODBCFetch läuft in einem DOS-Fenster

Aus der Datenbank heraus werden HTML-Seiten nach dem Templatemuster erzeugt. Exemplarisch nachfolgend eine HTML-Seite, gefüllt mit Daten.

```

<META NAME="DREREFERENCE" CONTENT="http://127.0.0.1/cgi-
bin/ODBCFetch.exe?Method=FetchData&keyID=24205"><HTML>
<HEAD>
<TITLE>618330 Clip Timer</TITLE>
</HEAD>
<BODY>
<H1>618330 Clip Timer</H1><BR>
Max. Timerzeit 99 Min. und 59 Sek., wobei die ablaufende Zeit durch Blinken des
Doppelpunktes im Display symbolisiert wird. Bei diesem Timer kann man auf die Sekunde
genau die Count-Down-Zeit eingeben. Mit einem an der R&#252;ckseite angebrachten Clip
kann der Timer ohne Probleme an einem Schrank aufgestellt oder mit dem Magnet am
K&#252;hlschrank etc.<P>befestigt werden. Lieferung erfolgt mit Bedienungsanleitung und
Knopfzelle 1,5 V, Typ LR 44.<BR>
617474<TAB>Pass. Ersatzbatterie Typ LR 44 (Pkg. &#224; 2 St.)<BR>
<BR>
<BR>
<BR>
<BR>
<BR>
<BR>
<BR>
ST<BR>
<BR>
<BR>
conrad.de<BR>
Schaltuhren/Timer<BR>
Timer<BR>
<BR>

</BODY>
</HTML>

```

Ausschnitt 5-4 eine generierte HTML-Seite aus der ODBC-Datenquelle

Import und Indexierung in den KnowledgeServer unter zur Hilfenahme des Autonomy Autoindexers

Der Autoindexer von Autonomy ist ein Dienst, der zyklisch zu definierende Verzeichnisse auf Vorhandensein neuer Dokumente überprüft und bei Auffinden eines

neuen noch nicht indexierten Dokumentes in die DRE importiert. Bevor die gerade erzeugten Dateien importiert werden können, muss der Autoindexer erst korrekt konfiguriert werden.

```
[Configuration]
// 1: file
// 2: directory
// If PollingAction=4 then PollingMethod is ignored
PollingMethod=2
// In milliseconds (0 to do only once)
PollingPeriod=5000
RemoveLogFileOnStart=on
Number=1
0=Job0

// *****
// ***** Default Configuration settings *****
// *****
[Default]
// 1: index (from idx file) into DRE
// 2: import+index (any format) into DRE
// 7: import+index+delete
// 8: delete from DRE
PollingAction=2

// 0: do nothing
// 1: delete the file after processing
// 2: move the file to another directory
PollingPostAction=0
MoveToDirectory=f:\Autonomy\KnowServer\autoindexer\processed
PollingMaxNumber=1000

// *** File/Directory Default Polling settings ***
filePollFilename=queue
fileBaseDirectory=F:\wwwroot\htdocs

directoryPathCSVs=F:\autonomy\odbcfetch\michaelsfetch
directoryFileMatch=*.htm,*.html
directoryRecurse=on
//directoryMustHaveCSVs=
//directoryCantHaveCSVs=
//directoryBeforeDate=
//directoryAfterDate=

// *** DRE Settings ***
DreHost=127.0.0.1
QueryPort=2000
IndexPort=2001

// *** Import Settings ***
//0: delete IDX file
//1: move the IDX file to another directory
//2: leave in directory
importIDXFilesAction=0
importIDXFilesMoveTo=f:\Autonomy\KnowServer\autoindexer\importidx
Database=Archive
ImportStoreContent=on
ImportTempDir=f:\Autonomy\KnowServer\autoindexer\importTemp
//ImportStripTagCSVs=B,H1,ZZ
//ImportPathReplaceUpToSlash=4
//ImportPathReplaceString=
ImportRefReplaceCSVs=F:\wwwroot\htdocs,\
ImportRefReplaceWithCSVs=http://127.0.0.1/,/
//ImportMinLength=0
//ImportMinLengthWords=0
//ImportMaxLength=10000000
//ImportMaxLengthWords=500000
```

```

ImportSummary=on
ImportBreaking=on
ImportBreakingMinParagraphWords=100
ImportBreakingMaxParagraphWords=1000
ImportBreakingMinDocWords=1500
//ImportStripLinks=off
//ImportMetaToFields=off
//TextImportExtns=
//HtmlImportExtns=
//FixedFieldName0=SiteImage
//FixedFieldValue0=/sourceimages/image.gif
//FieldName0=URL_NAME
//FieldStart0=<!--URL_NAME=
//FieldStop0=-->

// *** Delete Settings ***
deleteRefReplaceCSVs=
deleteRefReplaceWithCSVs=

// *****
// ***** User Defined Configuration settings *****
// ***** Anything in here will overwrite defaults above *****
// *****

[Job0]

```

Ausschnitt 5-5 Konfigurationsdatei des Autonomy AutoIndexers

Die Konfigurationsarbeit beschränkt sich hier darauf, den korrekten Pfad (directoryPathCSV) einzutragen. Anschließend kann der Autoindexer gestartet werden.

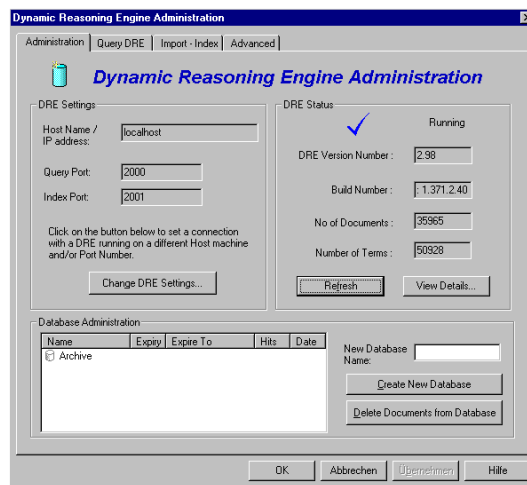


Abbildung 5-6 Administrationstool für die DRE - 35965 indexierte Dokumente und 50928 indexierte Begriffe

Nach dem Dokumentendatenimport kann mit dem Administrationstool eingesehen werden, dass insgesamt 35965 Dokumente indexiert wurden.

Konfiguration des CGI-Frontend des KnowledgeServers

Beim KnowledgeServer werden bereits CGI-Programme* mitgeliefert, die den Zugang zum KnowledgeServer vom WWW her steuern. Diese sind nach der korrekten Installation sofort erreichbar.

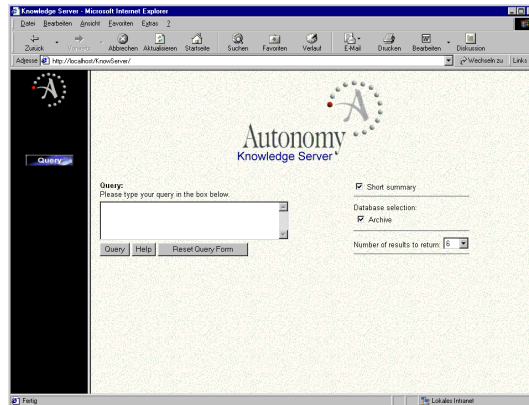


Abbildung 5-7 WWW-Front End des KnowledgeServers

Konfiguration des CGI-Frontend* des ODBCFetch-Services

Um Daten direkt aus der Datenbank abgreifen zu können, muss der CGI-Teil* von ODBCFetch konfiguriert werden. Im Skriptverzeichnis des WebServers (meist "cgi-bin") wird die Datei "ODBCFetch.cfg" angepasst.

```
[Configuration]
Parser=F:\wwwroot\cgi-bin\MichaelsFetch.cfg
ODBC=DSN=Articles;UID=;PWD=
```

Ausschnitt 5-6 Konfiguration des ODBC-CGI-Interfaces

Hierbei wird zunächst die zu parsende Konfigurationsdatei (Parser) festgelegt und anschließend die Datenquelle für das CGI-Frontend* definiert.

Die Datei "MichaelsFetch.cfg" ist zu der obig definierten äquivalent.

Lediglich das Template für die WWW-Ausgabe wird noch etwas umstrukturiert und angepasst.

```
<META NAME="DRREFERENCE" CONTENT="http://127.0.0.1/cgi-
bin/ODBCFetch.exe?Method=FetchData&keyID=<!--id-->">
<HTML>
```

```

<HEAD>
<TITLE><!--pos_name--> <!--pos_header--></TITLE>
</HEAD>
<BODY>
<font face="verdana">
<H1><!--pos_name--> <!--pos_header--></H1><BR>
<table>
<tr>
<td bgcolor="CCCCCC">pos_text
<td><!--pos_text-->
<tr>
<td bgcolor="CCCCCC">pos_acc01
<td><!--pos_acc01-->
<tr>
<td bgcolor="CCCCCC">pos_acc02
<td><!--pos_acc02-->
<tr>
<td bgcolor="CCCCCC">pos_acc03
<td><!--pos_acc03-->
<tr>
<td bgcolor="CCCCCC">pos_acc04
<td><!--pos_acc04-->
<tr>
<td bgcolor="CCCCCC">pos_acc05
<td><!--pos_acc05-->
<tr>
<td bgcolor="CCCCCC">pos_acc06
<td><!--pos_acc06-->
<tr>
<td bgcolor="CCCCCC">pos_acc07
<td><!--pos_acc07-->
<tr>
<td bgcolor="CCCCCC">lay_text_01
<td><!--lay_text_01-->
<tr>
<td bgcolor="CCCCCC">lay_ftext
<td><!--lay_ftext_01-->
<tr>
<td bgcolor="CCCCCC">pos_unit
<td><!--pos_unit-->
<tr>
<td bgcolor="CCCCCC">lay_header_06
<td><!--lay_header_06-->
<tr>
<td bgcolor="CCCCCC">lay_header_05
<td><!--lay_header_05-->
<tr>
<td bgcolor="CCCCCC">lay_header_04
<td><!--lay_header_04-->
<tr>
<td bgcolor="CCCCCC">lay_header_03
<td><!--lay_header_03-->
<tr>
<td bgcolor="CCCCCC">lay_header_02
<td><!--lay_header_02-->
<tr>
<td bgcolor="CCCCCC">lay_header_01
<td><!--lay_header_01-->
</table>
</font>
</BODY>
</HTML>

```

Ausschnitt 5-7 Template für die Ausgabe der ODBC-Datenquelle über das CGI-Interface^{*}

Erste Tests

Die Suchanfrage "Nokia Telefon Handy" liefert der Autonomy Ideologie entsprechend einige Ergebnisse.



Abbildung 5-8 Produkttreffer nach Eingabe des Suchwortes "Nokia Telefon Handy"

Hier hat der Benutzer nun die Möglichkeit seine Suche anhand bereits indexierter Dokumente zu verfeinern. Durch Anwahl verschiedener "Select"-Boxen und Anwahl des auf der Seite vorhandenen "Submit"-Buttons kann Autonomy durch die selektierten Dokumente das Suchergebnis verfeinern und die Relevanz in der Treffermenge erhöhen.

Beim Klick auf das Ergebnis "Nokia 6110" folgt dieser Bildschirm:



Abbildung 5-9 Produktbeschreibung für Nokia 6110 und ähnliche Produkte im unteren Fensterteil

Im rechten oberen HTML-Frame wird hier das in Ausschnitt 5-7 konfigurierte HTML-Template verwendet. Autonomy schlägt im unteren rechten Frame gleich ähnliche Dokumente vor.

*Lasttest**

Während der Evaluation wird der Knowledge Server auch einem Stresstest* unterzogen, um sein Verhalten zu beobachten. Zu diesem Zweck wird ein Tool der Firma Mercury Interactive verwendet, das gleichzeitig operierende Benutzer, die auf einen Webserver zugreifen, simuliert. Aus organisatorischen Gründen konnte der Test nur auf ein und demselben Computer durchgeführt werden. Der Rechner musste also für das Testing Tool und die zu testende Applikation Ressourcen zur Verfügung stellen. So ein Test ist eigentlich nicht aussagekräftig, aber das prinzipielle Verhalten von Autonomy kann so erkannt werden. Anhand dieser Aussagen kann auf gar keinen Fall eine Vergleichsaussage "Produkt A ist besser als Produkt B" getroffen werden.

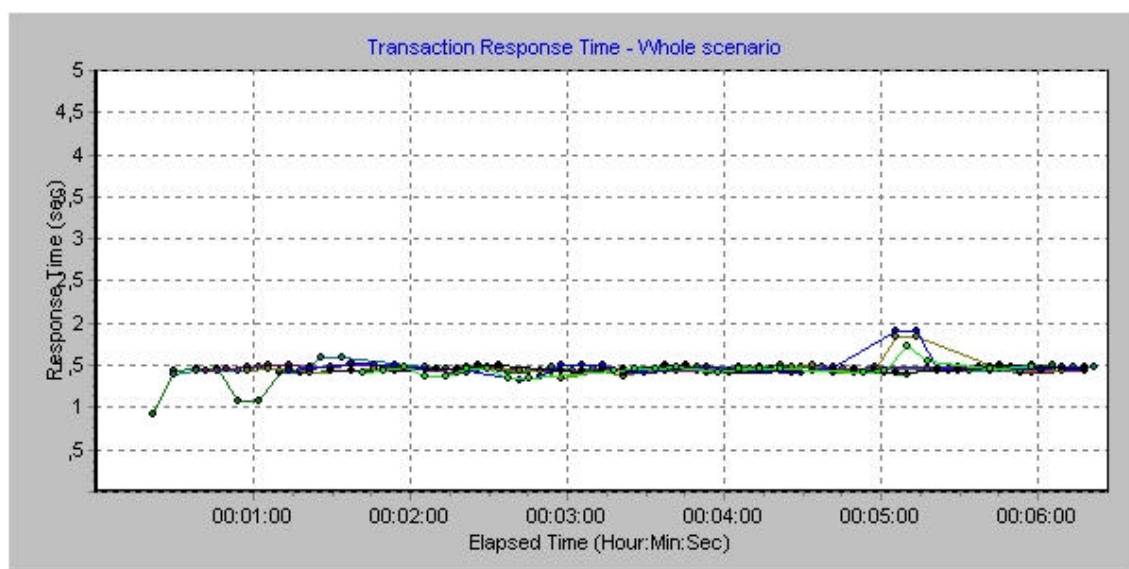


Abbildung 5-10 Antwortzeiten während des Lasttests

Farbe	Aktion	Max	Min	Avg
	[A] Eingabe Suchbegriff	1,5222	0,8112	1,3517
	[B] Verfeinern auf Nokia 6110	1,7125	1,3419	1,4540
	[C] Produktdaten zum Nokia 6110	1,5222	1,4120	1,4672
	[D] Klick auf Nokia 5110 verfeinern	2,3334	1,3319	1,4606
	[E] Klick auf Artikelnummer 765422	2,3834	1,3319	1,4912
	[F] Klick auf Artikelnummer 765430	1,5022	1,3820	1,4446
	[G] Anzeige Produktdaten 765430	1,7325	1,3319	1,4457

Tabelle 5-2 Aufstellung der Antwortzeiten beim Lasttest

Aktion	Beschreibung
[A]	Aufruf der ersten HTML-Seite; Eingabe des Suchbegriffs "handy nokia 6110"; setzen der zurückzugebenden Ergebnisse auf 50; Klick auf den "Query" Knopf
[B]	in der angezeigten Treffermenge das erste gefundene Produkt zur Detailansicht anklicken; Klick auf "767557 Nokia"
[C]	Anzeige der Produktdaten zu "767557 Nokia 6110"; Klick auf "Original Document" im unteren Vorschlagsframe;
[D]	Klick auf "765465 Nokia" im unteren Vorschlagsframe;
[E]	Anzeige der Produktdaten zu "765465 Nokia 5110"; Klick auf "765422" im Vorschlagsframe
[F]	Anzeige der Produktdaten zu "765422"; Klick auf "765430" im Vorschlagsframe
[G]	Anzeige der Produktdaten zu "765430"

Tabelle 5-3 Erläuterung der Aktionen beim Lasttest

Die gemessenen Werte in der Tabelle 5-2 stellen Sekundenwerte dar. Trotz der Tatsache, dass nur ein Rechner für den Test zur Verfügung stand, liefert der Knowledge Server von Autonomy doch beachtliche Resultate für diesen Test. Der Test wurde über sechs Minuten mit zehn gleichzeitigen Benutzern gemacht. Geht man davon aus, dass über den 24 Stundentag verteilt insgesamt acht Stunden unter 100% Last gefahren wird, so kann man für den Monat mit 30 Tagen die Gesamtbenutzer von 24.000 pro Monat errechnen.

Die erreichten Werte signalisieren doch, dass die Werte die geforderten sieben Sekunden deutlich unterschreiten.

Ergebnisse

Bei der Evaluation von Autonomy ist aufgefallen, dass Autonomy für die Verwaltung von unstrukturierten Daten hervorragend geeignet ist. Autonomy kann gut mit unstrukturierten Informationen umgehen. Im Falle der Conrad.com AG muss Autonomy sehr gut strukturierte Informationen bearbeiten. Bei den Tests hatte der Knowledge Server Probleme bei der reinen Stichwortsuche. Die Artikelnummer "767557" als Sucheingabe ergab leider nicht das Produkt "Nokia 6110", sondern lieferte eine leere Treffermenge zurück. Während der Evaluierung wurden auch Gespräche mit Technologieberatern der Firma Autonomy geführt. Laut deren Aussagen wäre es überhaupt kein Problem die Stichwortsuche oder die Artikelnummersuche zum Erfolg zu bringen. Mit den Firmenvertretern wurde eine prototypische Evaluierung auf Basis der Artikeldaten vereinbart. Die speziellen Anforderungen seitens Conrad.com AG an diesen Prototypen ist die unscharfe Suche und die Fähigkeit überhaupt nach Stichwörtern zu suchen. Bei der Auswahl von Autonomy als eine mögliche kommerzielle Suchmaschine wurde dem extern erstellten Konzept für die

Verschlagwortung zu wenig Bedeutung zugesprochen. Wie sich im Nachfeld der Evaluierung herausstellt ist dieses Konzept eines der grundlegendsten Elemente der Stichwortsuche. Autonomy als Standardprodukt ist leider nicht in der Lage dieses komplexe Konzept 100%-ig umzusetzen. Inwiefern jedoch, bei Aufweichung der entsprechenden Anforderungen, ein Einsatz von Autonomy's Knowledge Server doch noch möglich ist, bleibt abzuwarten.

5.2.1.2. Informationsserver Tamino der Software AG

[SAG99, SDC99] In diesem Abschnitt soll zunächst der Installations- und Konfigurationsvorgang des Informationsserver Tamino kurz beschrieben werden. Danach wird die Problematik der Dateneingabe bzw. Datenkonvertierung vom relationalen Datenmodell in das hierarchische Datenmodell beleuchtet, bevor ein kurzer Einblick in die Abfragesprache XQL* folgt. Eine Beschreibung der entworfenen DTD* zur Speicherung der Conrad.com AG Artikeldaten runden die Erfahrungen mit XML ab und abschließend werden die Ergebnisse eines Lasttests erläutert und ein kurzes Fazit gezogen.

Informationsserver Tamino

Der Informationsserver Tamino ist ein XML-basiertes DBMS, das speziell für Internet Applikationen ausgelegt ist. Der Server hebt sich im Wesentlichen durch die Art und Weise der Speicherung der Objekte von der Konkurrenz ab. XML Objekte werden in ihrer hierarchischen Struktur nativ in der Objektdatenbank abgespeichert. Dabei ist das DBMS speziell für die Speicherung, Verwaltung und Verarbeitung strukturierter und unstrukturierter Daten – Texte und Bilder – ausgelegt. Der Name "Tamino" ist einerseits ein Akronym für "**T**ransaktions-**A**rchitektur zum **M**anagement von **I**Nternet-**O**bjekten" und andererseits der Name des Helden in Mozarts Oper "Die Zauberflöte". Das DBMS Tamino basiert auf offenen Kommunikationsstandards HTTP und TCP/IP*. Die Datenbanken können über URLs angesprochen werden. Zum Betrieb des Tamino ist ein Webserver erforderlich. Der Tamino unterstützt die Abfragesprache XQL, um Inhalt und Struktur der gespeicherten Informationen abzufragen. Neben der nativen XML-Speicherung verfügt der Tamino noch über eine SQL-Engine für die nahtlose Integration von externen Datenquellen. Tamino verfügt über eine integrierte Volltextsuche mit eigener Indexierung. Auch bei der Indexierung wird die native

Speicherung der XML-Objekte ohne Konvertierung in relationale Modelle unterstützt.
Die eigene Hierarchie der XML-Objekte bleibt erhalten.

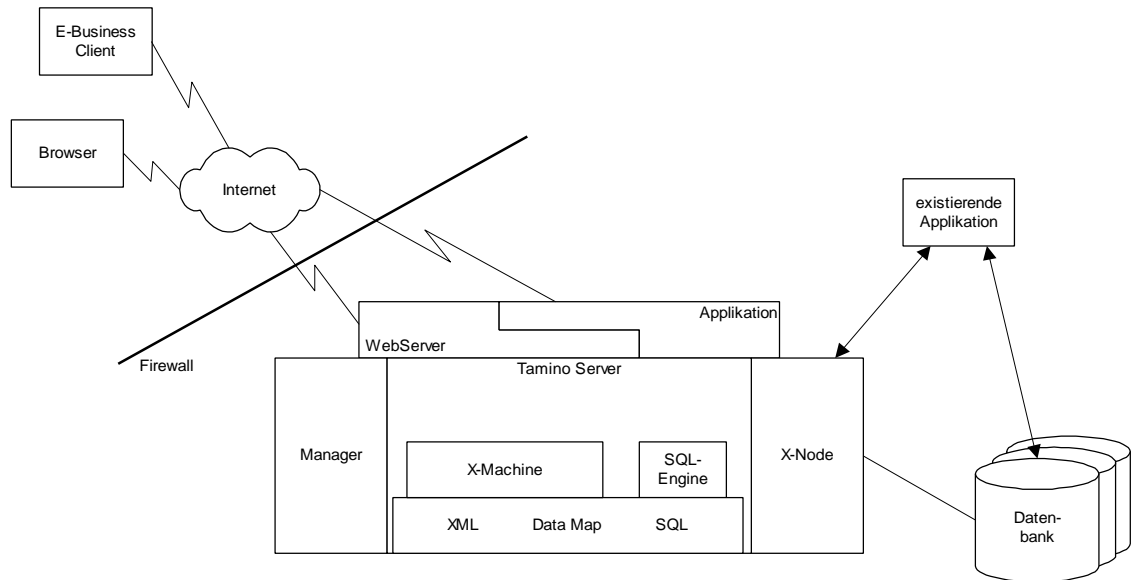


Abbildung 5-11 Architektur* des Informationsservers Tamino

Zentraler Bestandteil der DBMS Tamino ist die **X-Machine**. Die X-Machine* ist die XML-Engine, die für die Speicherung und die Abfrage von XML-Objekten zuständig ist. Die Objekte werden direkt im XML-Datenspeicher abgelegt. Parallel zur X-Machine arbeitet die **SQL-Engine**. Sie ist für die Migration von RDBMS nach XML zuständig und verwaltet Taminos internen SQL-Speicher. Die SQL-Engine ist auch für die Bearbeitung von SQL-Abfragen zuständig. Die **DataMap**-Komponente verhält sich für den Tamino wie ein Repository. Dort sind Systemdaten wie DTDs oder StyleSheets und relationale Schemata gespeichert. In der DataMap ist auch die Beschreibung gespeichert, wie XML-Elemente auf den XML-Speicher, den SQL-Speicher oder über den X-Node auf externe Datenquellen abgebildet werden. Die DataMap-Komponente ist also für das Mapping von XML auf das jeweilige Speichermedium zuständig und sorgt damit für eine Transparenz beim Zugriff auf die Daten. Die Komponente **X-Node*** ist für die Integration externer Datenspeicher zuständig. Der X-Node bildet die externen Datenquellen auf XML-Objekte ab und propagiert diese transparent nach außen. Der X-Node ist die integrierende Schnittstelle zu existierenden externen Datenquellen. Als letzte Komponente bleibt der **Manager** zu nennen, der für die Verwaltung des DBMS

zuständig ist. Der Manager ist eine internetbasierte Schnittstelle zur Remoteverwaltung. Er stellt zur Administration ein HTTP-basiertes Frontend zur Verfügung. Durch die Architektur* des Tamino ist es möglich, gewachsene, heterogene Datenbanklandschaften zu verknüpfen und auf diese Datenbasen transparent mit XML zuzugreifen.

Installation Tamino

Die Installation des Tamino unter Windows NT erfolgt mit Hilfe eines mitgelieferten Setup-Programmes. Die Installation eines Webservers – hier: Apache – wird für die erfolgreiche Installation des Informationsservers vorausgesetzt. Bei der Installation kann eine Trennung zwischen den Serverprogrammen und den Client-Tools erfolgen. Die Serverinstallation enthält zentral die Datenbankengine, während die Client-Tools die Zugriffswerkzeuge beinhalten. Für die Evaluierung wurden beide Seiten auf einem Rechner installiert. Für die Datenbank wird auf dem Server ein TCP/IP-Port (hier: 3200) definiert. Der Port wird für die Administration der Datenbank verwendet. Für die Kommunikation mit der Datenbank wird der Webserver benötigt. Über das HTTP-Protokoll erfolgen Abfragen an die Datenbank und werden Antworten zurückgesendet. Die Datenbank kann allgemein über eine einfache URL angesprochen werden. Hier ist es die URL "http://localhost/tamino/MyDB". Bei der Installation wird der Server und einige Administrationstools installiert. Die Tools für den Betrieb des Tamino sind der "Schema Manager", der "Tamino Manager" und der "Tamino Interactive Manager". Sie arbeiten als Java Applikationen auf der Java Virtual Machine* des installierten Browsers*.

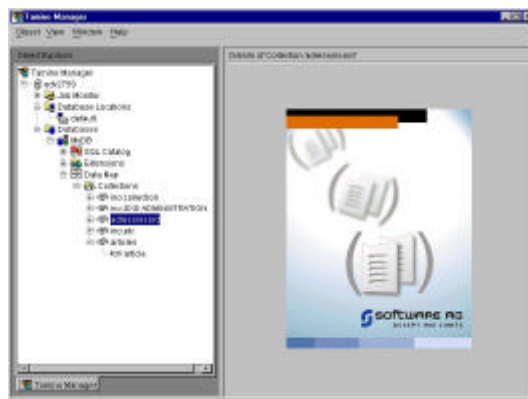


Abbildung 5-12 Der Tamino Manager zur Verwaltung der Datenbank

Der **Tamino Manager** ist für die eigentliche Verwaltung der Datenbank zuständig. Mit dem Tamino Manager kann die Datenbank beobachtet werden, Jobs abgearbeitet werden, die Datenbank gestartet und gestoppt werden. In der DataMap der administrierten Datenbank können die Schema Mappings angesehen werden. In der Abbildung 5-12 sieht man bereits die XML-Definitionen für den "adressrecord" und "articles", die nachher noch näher erläutert werden.

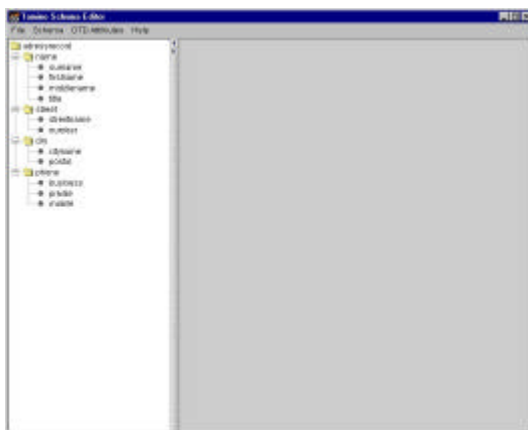


Abbildung 5-13 Der Schema Manager

Der installierte **Schema Manager** ist für den Entwurf und die Pflege der XML-Schemata in der Datenbank zuständig. Es ist möglich in der Baumsicht hierarchische Strukturen einfach durch Zusammenklicken zu erstellen. Ebenso können DTDs in die Datenbank eingespielt und herausgeholt werden. Der Schema Manager ist für die Verwaltung der logischen Sicht auf die Datenbank zuständig.



Abbildung 5-14 Das Interactive Interface

Das **Tamino Interactive Interface** ist für die Abfrage und die Einpflege von Daten zuständig. Das Interactive Interface ist eine HTML-Seite, die über den HTTP-Kanal mit der Datenbank kommuniziert. Anfragen können in der oberen Hälfte der HTML-Seite abgesetzt werden, während Antworten des Servers in der unteren Hälfte erscheinen.

Dateneingabe

Um ein Handling für den Informationsserver Tamino zu entwickeln wurde zunächst ein einfaches Beispiel erstellt. Anschließend soll für die komplexeren Artikeldaten der Conrad.com AG eine DTD* entwickelt werden. Als Beispiel wird eine kleine Adressverwaltung herangezogen. Vor der Dateneingabe muss das Schema definiert werden. Für das Schema wurde ein einfacher, gut durchschaubarer Ansatz gewählt.

```
<!-- Elements -->
<!ELEMENT adressrecord
      ( name, street, city, phone* )>
<!ELEMENT name
      ( surname, firstname, middlename?, title? )>
<!ELEMENT surname      (#PCDATA)>
<!ELEMENT firstname    (#PCDATA)>
<!ELEMENT middlename   (#PCDATA)>
<!ELEMENT title        (#PCDATA)>

<!ELEMENT city
      ( cityname, postal? )>
<!ELEMENT cityname     (#PCDATA)>
<!ELEMENT postal       (#PCDATA)>

<!ELEMENT phone
      ( business*, private?, mobile? )>
<!ELEMENT business     (#PCDATA)>
<!ELEMENT private      (#PCDATA)>
<!ELEMENT mobile       (#PCDATA)>

<!ELEMENT street
      ( streetname, number? )>
<!ELEMENT streetname   (#PCDATA)>
<!ELEMENT number       (#PCDATA)>
```

Ausschnitt 5-8 DTD zum Adressrecord

Die DTD in Ausschnitt 5-8 ist bereits in Abbildung 5-13 in der Baumansicht zu sehen. Am Baum sieht man, dass die Struktur nicht besonders hierarchisch ist, sich aber gut für erste Schritte eignet. Die DTD wird in einer Datei gespeichert und über den Tamino Schema Manager in der Datenbank definiert. Anschließend können über das Interactive Interface Daten eingegeben werden.

```

<?xml version="1.0"?>
<adressrecord>
  <name>
    <surname>Testlooser</surname>
    <firstname>Tea</firstname>
  </name>

  <street>
    <streetname>Johannesburger Straße</streetname>
    <number>101</number>
  </street>

  <city>
    <cityname>Hirschau</cityname>
    <postal>09666</postal>
  </city>

  <phone>
    <private>0899/87887</private>
    <mobile>0171/3123499</mobile>
    <business>08111/1446336</business>
  </phone>
</adressrecord>

```

Ausschnitt 5-9 Beispieldatensatz zum Adressrecord in XML - konform zur obigen DTD

Die XML-Daten wurden in einem Editor erfasst und anschließend in einer Textdatei gespeichert. Diese Textdatei kann über das Interactive Interface in die Datenbank eingepflegt werden. Wichtig bei der Erstellung der XML-Dateien ist, dass die erste Zeile "<?xml version='1.0'?>" vorhanden ist. Ansonsten werden die XML-Dateien nicht als XML-Dateien erkannt und auch nicht im XML-Speicher des Tamino abgelegt. Neben der Eingabe der Daten ist auch die gezielte Selektion von Daten wichtig. Hierfür bietet der Informationsserver Tamino die Abfragesprache XQL. Nachfolgend eine knappe Einführung in die Abfragesprache.

Ein kurzer Überblick über XQL – XML Query Language

[SDC99] Mit XQL-Anweisungen können Informationen aus XML-Dokumenten bzw. XML-Objekten abgefragt werden. Um eine saubere XQL-Anfrage formulieren zu können, muss der Anfragende Informationen über die Struktur der XML-Dokumente besitzen. Die Struktur eines XML-Dokumentes ist hierarchisch aufgebaut. Ein Startknoten kann also andere Elemente, darunter auch weitere Knoten und Attribute, enthalten. Die Struktur des XML-Dokumentes wird formal durch die DTD beschrieben. DTDs legen fest, welche Elemente in einem XML-Dokument erlaubt sind, und wo und wie oft diese auftreten dürfen.

Um mit XQL Informationen aus einer XML-Datenbank abfragen zu können, muss der Name des Elements, das die Daten enthält, bekannt sein. Zusätzlich sollten noch weitere Filterkriterien bekannt sein, um die Treffermenge einschränken zu können. Soll nicht das oberste Element, also der Wurzelknoten, abgefragt werden, so muss die XQL-Abfrage den Pfad durch die Dokumentenstruktur festlegen, um die richtige Hierarchiestufe des abzufragenden Elements zu adressieren.

Eine allgemein gehaltene XQL-Abfrage ohne Filterkriterien sieht beispielsweise so aus:

```
Wurzel/Hierarchie1/Hierarchie2/.../HierarchieN/Element
```

Eine XQL-Abfrage mit Filterkriterien könnte so aussehen:

```
Wurzel/Hierarchie1/Hierarchie2/.../HierarchieN[Element = Filterkriterium]
```

Im Beispiel der Adressdatensätze wählt die XQL-Abfrage

```
adressrecord/street/streetname
```

aus den eingegebenen Adressdatensätzen nur die Straßennamen der gespeicherten XML-Dokumente aus. Die XQL-Abfrage mit Filterkriterium

```
adressrecord/street[streetname="Street*"]
```

wählt aus dem Datenbestand nur solche aus, die als Straßename mindestens die Zeichenfolge "Street" beinhalten. Um Abfragen zu vereinfachen kann in den Pfad zu den Elementen auch ein Jokerzeichen eingefügt werden. Die XQL-Anfragen

```
adressrecord/*/streetname
```

```
adressrecord/street/streetname
```

erzeugen beispielsweise die gleiche Treffermengen.

XQL-Abfragen können durch eine Reihe von Operatoren verfeinert werden.

Operator	Funktion	Beispiel
\$AND\$	boole'scher UND-Operator	adressrecord/street[streetname \$EQ\$ "Street" \$AND\$ streetname \$EQ\$ "*name*"]
\$OR\$	boole'scher ODER-Operator	adressrecord/street[streetname \$EQ\$ "Street" \$OR\$ streetname \$EQ\$ "*name*"]
\$EQ\$ oder "="	relationaler Gleichheitsoperator	adressrecord/street[streetname \$EQ\$ "Street"]
\$LE\$ oder "<="	relationaler Kleiner-Gleich-Operator	adressrecord/street[number \$LE\$ "15"]
\$GE\$ oder ">="	relationaler Größer-Gleich-Operator	adressrecord/street[number \$GE\$ "15"]
\$LT\$ oder "<"	relationaler Kleineroperator	adressrecord/street[number \$LT\$ "15"]
\$GT\$ oder ">"	relationaler Größeroperator	adressrecord/street[number \$GT\$ "15"]

Tabelle 5-4 Operatoren in XQL-Anfragen

Neben der Suche nach Elementinhalten kann in XQL auch nach Attributen einzelner Elemente gesucht werden. Die folgende XQL-Abfrage enthält alle "widget"-Elemente, deren "quality_checked"-Attribut den Wert "yes" haben

```
components//widget[@quality_checked="yes"]
```

Die Abfrage

```
components//widget[@quality_checked]
```

liefert alle "widget"-Elemente zurück, für die das Attribut "quality_checked" spezifiziert ist. Mit XQL ist es möglich Inhalte und auch Struktur von Dokumenten abzufragen.



Abbildung 5-15 Das Interactive Interface mit einer durchgeführten XQL-Anfrage

Für die Abfragen an den Informationsserver Tamino ist das Tamino Interactive Interface vorgesehen. In Abbildung 5-15 ist das Resultat der Abfrage

adressrecord[name/surname="test*"]

sichtbar.

Entwurf einer DTD* für Artikeldaten der Conrad.com AG

Der Entwurf der DTD für die Artikeldaten gestaltete sich als wesentlich komplexer, als der zur Speicherung von Adressen. Beim Entwurf der DTD für die Artikeldaten wurde zunächst von einigen Beispielartikeln ausgegangen und die Zusammenhänge zwischen den einzelnen Attributen erfasst. Anschließend wurde die Strukturierung der Artikel Zusammenhang für Zusammenhang in die DTD übertragen.

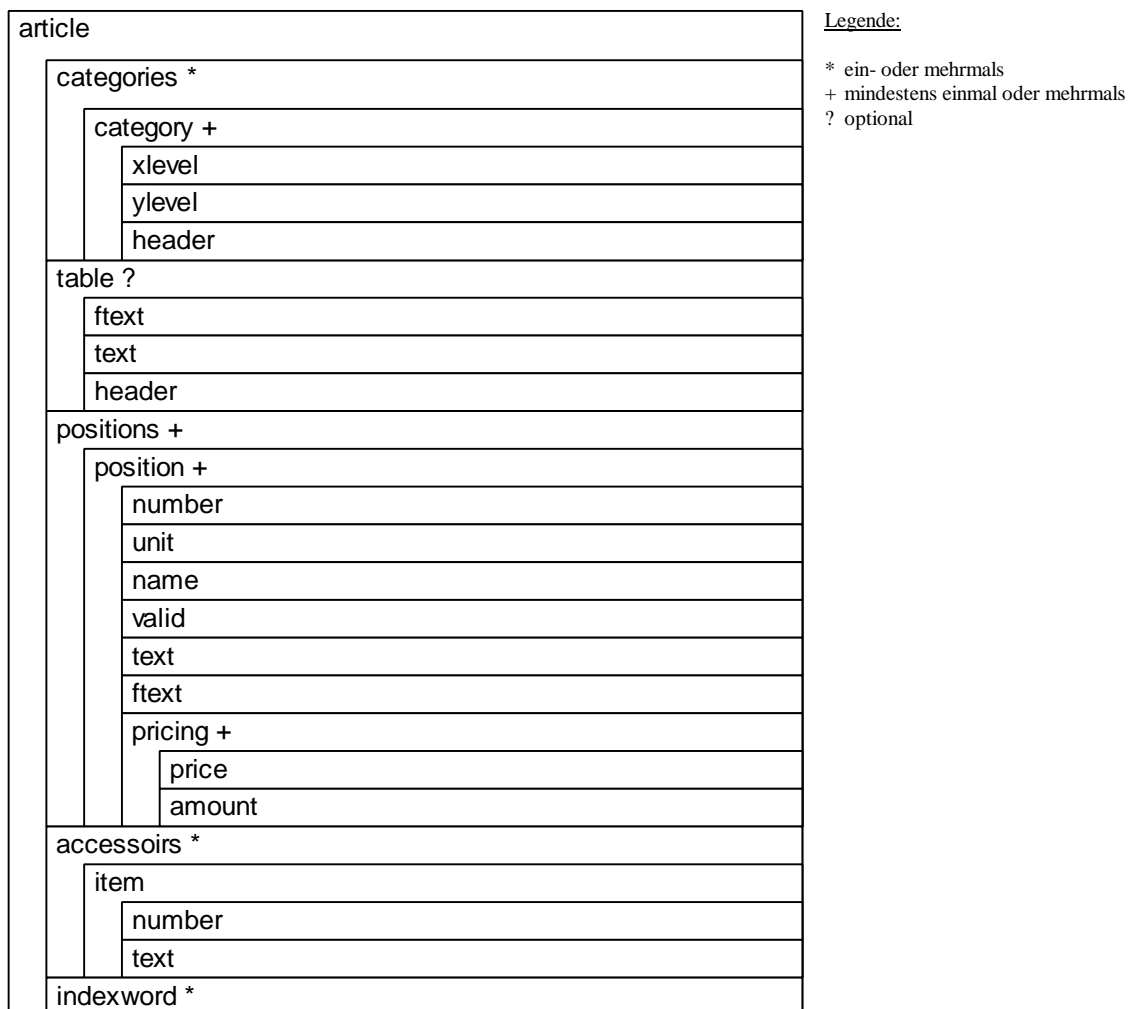


Abbildung 5-16 grafische Abbildung der DTD für die Artikeldaten der Conrad.com AG

Die Zusammenhänge, die in der Abbildung 5-16 angedeutet sind, werden nachfolgend noch verbal erfasst.

Ein Artikel kann zu keiner oder mehreren Kategorien [categories] gehören. Eine Kategorie besteht zumindest aus einer Kategoriebeschreibung [category]. Die Kategorie besitzt die Attribute Kategorielevel in X-Richtung [xlevel] und den Kategorielevel in Y-Richtung [ylevel]. Die beiden Level sind abhängig von der externen Kategorisierung der Produkte. Zusätzlich ist natürlich der Kategorienname [header] definiert.

[CIF00] Ein Artikel kann ein Tabellenartikel [table] sein. Ein Tabellenartikel ist ein Artikel, der Teil einer gleichartigen Untermenge ist. Solche Artikel, beispielsweise Widerstände von 1 Ohm bis zu 25 kOhm, werden auf der Produktdetailseite tabellenartig dargestellt. Der ausgewählte Artikel wird mit seinem Artikellangtext dargestellt und alle anderen Artikel der gleichartigen Untermenge werden selektierbar in einer Tabelle dargestellt. Daher der Name Tabellenartikel. Ist der Artikel ein Tabellenartikel, so wird zusätzlich zu den anderen Attributen noch ein Tabellenartikeltext [text], ein weiterführender Tabellenartikeltext [ftext] und die Tabellenüberschriften [header] gespeichert. Der Artikel kann aus mehreren Positionen [positions] bestehen. Der Artikel hat in jedem Fall eine Position [position]. Die Positionen sind wieder untergliedert in Positionseinträge. Eine Position ist gekennzeichnet durch eine Artikelnummer [number], die Stückerinheit [unit], den Artikelnamen [name], das Gültigkeitsdatum [valid], den Artikelbeschreibungstext [text], einen weiterführenden Artikelbeschreibungstext [ftext] und mindestens eine Preisübersicht [pricing]. Die Preisübersicht ist unterteilt in den Preis [price] in Verbindung mit der abzunehmenden Menge, ab der der Preis gilt [amount]. Zu jedem Artikel können mehrere Zubehörartikel [accessoirs] festgelegt werden. Ein Zubehörartikel [item] wird durch die Artikelnummer [number] und den beschreibenden Text [text] beschrieben. Letztendlich können für einen Artikel noch Schlagwörter [indexword] gespeichert werden.

```
<!-- DTD for doctype Article. -->

<!-- Elements -->

<!ELEMENT article
      ( categories*, table?, positions+, accessoires*, indexword* )>
<!ELEMENT indexword  (#PCDATA)>

<!ELEMENT categories
      ( category+ )>

<!ELEMENT category
```

```

( xlevel, ylevel, header )>
<!ELEMENT xlevel      (#PCDATA)>
<!ELEMENT ylevel      (#PCDATA)>
<!ELEMENT header      (#PCDATA)>

<!ELEMENT table
( ftext, text, header )>

<!ELEMENT ftext      (#PCDATA)>
<!ELEMENT text       (#PCDATA)>

<!ELEMENT positions
( position+ )>

<!ELEMENT position
( number, unit, name, pricing+, valid, text, ftext )>

<!ELEMENT name        (#PCDATA)>
<!ELEMENT number      (#PCDATA)>
<!ELEMENT valid        (#PCDATA)>
<!ELEMENT unit        (#PCDATA)>

<!ELEMENT pricing
( price, amount )>

<!ELEMENT price       (#PCDATA)>
<!ELEMENT amount      (#PCDATA)>

<!ELEMENT accessoires
( item )>

<!ELEMENT item
( number, text )>

<!-- Attributes -->
<!ATTLIST article
  istable    ( true | false )      #REQUIRED
>

<!-- End of DTD -->

```

Ausschnitt 5-10 DTD für die Artikeldaten der Conrad.com AG

Die DTD in Ausschnitt 5-10 wurde sukzessive verfeinert, bis auch tatsächlich Artikel korrekt abgespeichert werden konnten. Nachdem die DTD erstellt war, mussten die Artikeldaten konvertiert werden. Diese Artikeldaten sind in relationalen Tabellen im MS-Access Datenbankformat gespeichert. Die Konvertierung der relationalen Zusammenhänge in hierarchische Strukturen erweist sich als nicht trivial. Das MS-Access-Format entsteht aus einer Datei, die aus dem SAP^{*} bzw. MMS^{*} erstellt wird. In der Textdatei ist die Reihenfolge der Datensätze entscheidend für die Zusammengehörigkeit von Tabellenartikeln. Tabellenartikel haben im Datenfeld "lay_type_01" die Kennzeichnung "TB". Alle nachfolgenden Artikel mit dem gleichen Inhalt im Datenfeld "lay_name_01" gehören zum Tabellenartikel. Durch diese Zusammenhänge ist es nicht ohne weiteres möglich die gesamten Daten der MS-Access

Datenbank nach XML zu übersetzen. Zu Testzwecken wurde daher der Weg gewählt, einige wenige Artikel von Hand nach XML zu übersetzen. Nachfolgend ein Beispiel eines Artikels.

```
<?xml version="1.0"?>
<article istable="false">
  <categories>
    <category>
      <xlevel>1</xlevel>
      <ylevel>15</ylevel>
      <header>Audio/HiFi</header>
    </category>
    <category>
      <xlevel>2</xlevel>
      <ylevel>44</ylevel>
      <header>Boxenbauzubeh&#246;r</header>
    </category>
  </categories>
  <positions>
    <position>
      <number>342718</number>
      <name>Lautsprecher-Pegelregler</name>
      <unit>ST</unit>
      <pricing>
        <price>24.95</price>
        <amount>ST</amount>
      </pricing>
      <pricing>
        <price>22.95</price>
        <amount>5 ST</amount>
      </pricing>
      <valid>01.01.1980</valid>
      <text>Preiswerter, jedoch extrem leistungsf&#228;higer
Pegelregler. Verwendbar f&#252;r Boxen von 4 - 16 <FONT FACE="Symbol">&#87;</FONT>.
Spitzenbelastbarkeit 100 W, Drehbereich 300&#176; &#177;5&#176;.</text>
      <ftext></ftext>
    </position>
  </positions>
  <indexword>100 W</indexword>
  <indexword>Audio</indexword>
  <indexword>bauen</indexword>
  <indexword>Box</indexword>
  <indexword>Boxenbau</indexword>
  <indexword>Boxenbauzubeh&#246;r</indexword>
  <indexword>Fitting</indexword>
  <indexword>HiFi</indexword>
  <indexword>Lautsprecher</indexword>
  <indexword>Lautsprecherbau</indexword>
  <indexword>Lautsprecherbox</indexword>
  <indexword>Lautsprechergeh&#228;use</indexword>
  <indexword>Lautsprecherpegelregler</indexword>
  <indexword>SAT-Audio-Video</indexword>
  <indexword>Selbstbau</indexword>
  <indexword>Unterhaltungselektronik</indexword>
</article>
```

Ausschnitt 5-11 ein Beispielartikel in XML - konform zur obigen DTD

Für die Erstellung der DTD* bzw. vor der Erstellung wurden zur Orientierung Anhaltsmustern für die DTD eines Artikels gesucht. Für die Speicherung bietet sich der Quasi-Standard BMECat [BME00] an. Die BMECat DTD beschreibt Artikel von einem allgemeinen Standpunkt und geht noch über die reine Abspeicherung der Artikel hinaus.

Ziel des BMECat ist die Ablösung des EDI-Standards. Hinter dem BMECat stehen bereits einige große Firmen, die diesen Quasi-Standard unterstützen wollen. Während der Evaluation wurde bewusst gegen die Verwendung des Standards entschieden, da die Erstellung einer eigenen DTD einfacher war, als die Einarbeitung in den Quasi-Standard. Zudem genügt die erstellte DTD den Ansprüchen für Evaluierungszwecke. Eine weitere Vorgehensweise wäre eine genauere Prüfung inwieweit die DTD BMECat für die hausinternen Zwecke eingesetzt werden kann.

Lasttest

Für die Durchführung des Lasttests wurde ein Prototyp, den die Software AG erstellt hat, eingesetzt. In der Datenbank des Prototypen befanden sich in etwa 11.000 Artikel. Ziel beim Lasttest war es den Informationsserver unter Last zu beobachten. Wichtig hierbei sind die Parameter Antwortzeit und auch Stabilität.

Zunächst einige Rahmenparameter für den Test. Die Hardwarekonfiguration der beteiligten Rechner ist die folgende:

Server:

- Dual Pentium III, SCSI-SubSystem, 512 MB RAM
- Windows NT 4.0 Server
- Tamino Version 1.2.1.4
- Apache WebServer

Client:

- Pentium III, EIDE-SubSystem, 128 MB RAM
- Windows NT 4.0 Workstation SP 6
- Tamino Interactive Interface
- Apache WebServer

Die beiden Rechner sind über eine 10 Mbit Twisted Pair Verkabelung verbunden. Als Verbindungselement dient ein Hub. Beide Rechner befinden sich im gleichen Netzsegment. Die IP-Adresse des Servers war zum Zeitpunkt des Tests 194.174.253.251. Die IP-Adresse des Clients zum Zeitpunkt des Tests war die 194.174.253.249. Die Verbindung war nicht durch Gateways oder Firewalls behindert.

Nachfolgend eine Beschreibung der Definition der Szenarien für das Tool Astra LoadTest. Teil des Prototypen der Software AG ist eine HTML-Seite, auf der auf komfortable Art die zugrundeliegende Datenbank abgefragt werden kann. Für den Lasttest wird das clientseitig installierte Tamino Interactive Interface verwendet. Über

das Lasttest Tool werden XQL-Abfragen via Interactive Interface an den Informationsserver gesendet.

Das Szenario wurde wie folgt definiert:

- Eintragen des XQL-Statements in das entsprechende Texteingabefeld des Interactive Interfaces
- Klick auf "Process"
- Klick auf "Set" neben der Database URL
- Klick auf "Process"

Dieses Szenario wird mit verschiedenen XQL-Abfragen durchgeführt. Während der Durchführung des Szenarios werden 50 gleichzeitige Benutzer simuliert. Die Anzahl der Benutzer wird rampenartig gesteigert, und letztlich ab der 6. bis zur 12. Minute 50 gleichzeitige Benutzer simuliert.

XQL-Abfragen

Nachfolgend die XQL-Abfragen, die im Rahmen des Szenarios an den Tamino gesendet wurden:

- Article/Positions/Position[Name=""070224""]
- Article/Positions/Position[Text="Kampftechniken"]
- Article/Positions/Position[Name=""*2*"]
- Article/Positions/Position[Name=""*1*"]
- Article/Positions/Position[Text="Nico Toscani*"]
- Article/Positions/Position[Text="vom"]
- Article/Positions/Position[Header="Twister"]
- Article/Positions/Position[Text="Bauteil"]
- Article[Positions/Position/Text="Bauteil"]
- Article/Positions/Position[Text="vom"]
- Article/Positions/Position[Text="Widerstand"]
- Article/Categories/Category[Name="DVD"]

Die Tests wurden mittels des definierten Szenarios durchgeführt. Ursprünglich war geplant vier bis fünf Testdurchläufe durchzuführen, um kurzzeitige Schwankungen zu eliminieren. Leider musste nach dem zweiten Testdurchlauf der Lasttest abgebrochen

werden. Die Datenbank des Informationsservers befand sich ab diesem Zeitpunkt in einem undefinierten Zustand. Versuche, die Datenbank zu stoppen (im normalen oder auch in den anderen angebotenen Modis) und wieder hochzufahren, sind leider fehlgeschlagen.

Durch dieses unerwartet negative Verhalten ist das Vertrauen in die Software Tamino sehr stark erschüttert und der Informationsserver der Software AG hat sich damit de facto aus der engeren Auswahl bewegt.

Fazit

Bis zur Durchführung des Lasttests festigte sich die Überzeugung, dass der Tamino ein wirklich gelungenes Produkt zur Verwaltung von Daten sei. Der zukunftsweisende Ansatz ein XML-basierendes DBMS auf den Markt zu bringen ist sicherlich sehr innovativ. Zudem ist mit dem Tamino auch die Realisierung der externen Verschlagwortung möglich. Eigentlich ist der Tamino das Produkt der Wahl. Nur diese Instabilität während des Lasttests deplaziert das Produkt. Vielleicht ist nur eine Kleinigkeit falsch gelaufen, aber man muss doch von der Herstellerfirma eines Produktes erwarten können, dass die Technik-Spezialisten die Installation des Produktes soweit optimieren, dass der Lasttest (hier ja eigentlich nur die Absendung vieler XQL-Statements) reibungslos abläuft. Selbst wenn das Produkt wirklich gut ist, so wirft doch die Vorgehensweise der Firma einen Schatten auf das Produkt.

5.2.2 Eigenentwicklung

Bei der Eigenentwicklung bieten sich grundsätzlich zwei Optionen:

- komplette Neuentwicklung der Suchapplikation
- Optimierung der alten Suchapplikation

Bei beiden Varianten wird die "neue" Suchapplikation auf einen dedizierten Datenbankserver installiert. Grundlegendstes Kriterium der Eigenentwicklung ist die Möglichkeit einer 100%-igen Überdeckung des theoretischen Konzeptes für die Verschlagwortung mit der letztlich implementierten Suche zu erreichen. Das Konzept kann also – soweit verstanden – in "Code gegossen" werden. Einzigster Problempunkt hierbei ist der Programmierer, der das Konzept natürlich verstehen muss, bevor es performant umgesetzt werden kann. Für und gegen die Eigenentwicklung sprechen eine Reihe von Vor- und Nachteilen.

Da bei der Eigenentwicklung ein Teil der Applikation ausgetauscht bzw. erneuert wird, ist es möglich das zugrundeliegende Datenmodell zu optimieren. Die Suche stellt eine dedizierte Aufgabe in der Applikation dar und verwendet auch weitgehend ein eigenständiges Datenmodell*. Dieses Datenmodell kann optimiert oder komplett neu entworfen werden, um die Suche zu optimieren und die Performanz zu erhöhen. Mögliche Vorgehensweise für das neue Datenmodell wäre Datenredundanz im Sinne einer Denormalisierung* in das relationale Datenmodell einzubauen. Bei der Neuentwicklung und auch bei der Reimplementierung der jetzt verwendeten Suche ist es möglich zusätzliche Features wie eine unscharfe Suche o.ä. einzubauen. Neben diesen eher technischen Vorteilen gibt es auch organisatorische Vorteile. Für die abgekapselte Suchapplikation ist es möglich einen Workflow "Konzeption – Implementierung – Test – Wartung" zu installieren. Dadurch ist es möglich eine strukturierte Vorgehensweise durchzusetzen. Als "Abfallprodukt" dieser Vorgehensweise fällt dann ein Konzept und eine Dokumentation ab. Neben den Vorteilen existieren auch Nachteile. Im Vergleich zur Implementierung eines Standardproduktes dauert die Eigenentwicklung tendenziell eher länger und ist ein aufwändigerer Vorgang. Neben dem zeitlichen Aspekt ist auch die Gefahr eines Misserfolges oder auch nur einer Nichtverbesserung gegeben. Versteht der Programmierer oder der Konzepter das Konzept nicht, oder wird das Konzept nicht optimal umgesetzt, so kann die Neuentwicklung schlechter werden, als die derzeit implementierte Umsetzung. Durch den Programmierer oder besser gesagt den Faktor "Mensch" in der Eigenentwicklung sind auch neue und mehrere unbekannte Fehlerquellen an der Entwicklung mitbeteiligt. Auch kann die zu erwartende Leistung nicht anhand von Referenzinstallationen nachvollzogen werden. Es existieren einige nicht bekannte und schwer fassbare Faktoren, die die Eigenentwicklung zum eindeutig riskanteren Unterfangen machen. Ganz speziell bei der Implementierung in diesem Fall sind mögliche Seiteneffekte nicht auszuklammern. Dieser Faktor ist einer der gefährlichsten. Ganz ausschließen kann man diesen Faktor aber auch nicht bei der Verwendung einer Standardanwendung. Neben den ganzen technischen und organisatorischen Nachteilen birgt auch die Eigenentwicklung einen finanziellen Nachteil. Da die Suche bereits einmal bezahlt und nicht optimal umgesetzt worden ist,

kann die zweite Entwicklung als Reimplementierung gewertet werden. Die gleiche Funktionalität wird zweimal bezahlt.

Nach den Vor- und Nachteilen der Eigenentwicklung generell, sollen nachfolgend die Vor- und Nachteile der kompletten Neuentwicklung und der Optimierung der bestehenden Suchapplikation aufgezeigt werden.

5.2.2.1. Neuentwicklung einer Suchapplikation

Hier in diesem Zusammenhang wird unter Neuentwicklung die komplette Reimplementierung des Konzeptes verstanden. Zentralster Vorteil an der Reimplementierung ist die freie Sprachwahl. Es besteht die Chance durch Wahl einer geeigneten Sprache die Nachteile von Interpretersprachen wie PERL zu umgehen. Performanzprobleme, die in der Struktur der Sprache stecken, können auf diesem Wege leicht umgangen werden. Neben der Sprache ist auch der Programmierer frei wählbar. Durch die Neuentwicklung der Suche kann ein unbelasteter, kompetenter Programmierer eingesetzt werden. Diese personelle Neubesetzung bringen auch Synergieeffekte mit in die Applikation. "Neue" Personen sind von der Thematik unbelastet und haben meist eine andere Sichtweise.

Durch die Loslösung von der Applikation wird auch eine Herauslösung aus dem gesamten Komplex Applikation erreicht. Dies hat nicht nur Vorteile. Ist die Suche komplett aus der Applikation herausgelöst, so müssen Datenbankverbindungen separat von der Suchapplikation verwaltet, aufgebaut und geschlossen werden. Die Suche extrahiert sich aus dem Gesamtkonzept der bestehenden PERL Applikation. Die neue Suchapplikation lässt sich zudem schwer in das existierende Applikationsmodell integrieren. Die Integration ist wegen fehlender Schnittstellenbeschreibungen nicht einfach. Nebeneffekte lassen sich nur schwer abschätzen. Soweit zu den Vor- und Nachteilen einer kompletten Neuimplementierung.

5.2.2.2. Optimierung der bestehenden Suchapplikation

Die Optimierungsmaßnahmen an der existierenden PERL Suche implizieren in jedem Falle einen Umzug auf einen dedizierten Suchserver mit darunter liegender Datenbankinstanz. Vorteil der Optimierung hier ist, dass kaum Änderungen an der restlichen Shoppingapplikation nötig sind. Die optimierten Suchroutinen integrieren sich also nahtlos in die Shop-Applikation. Die Analyse der PERL Sourcen brachten ein

erhebliches Optimierungspotential hervor. Problematisch an dieser Erkenntnis ist nur, dass es sehr schwer wird das Konzept der aktuell implementierten Suche vollständig funktional zu reproduzieren. Diese Bedenken ergeben sich aus der Undurchschaubarkeit des Sources. Eigentlich sollte den Source der originäre Programmierer überarbeiten, da dieser die Konzepte als einziger versteht. Daraus entstehen aber kaum Synergieeffekte. Der Programmierer, der seinen eigenen Code im nachhinein wirklich optimiert, existiert nicht, da die Optimierung schon während der eigentlichen Programmierung erfolgen hätte können.

6. Integration des bestmöglichen Systems

Hier sollen die weiteren Schritte für die Integration der Suche in die Shoppingapplikation beschrieben werden. Zuvor wird noch ein abschließendes Fazit gezogen bzw. die einzelnen Lösungsmöglichkeiten abschließend bewertet. Für die Bewertung wird eine Leistungsmatrix (siehe dazu Tabelle 6-1) verwendet.

Neben den rein technisch orientierten Aspekten der Lösung ist auch die politische Ebene einer geschäftlich orientierten Entscheidung aufgefallen. Die technischen Aspekte sprechen zum Teil eindeutig für ein Produkt, werden aber durch politische Ausrichtungen überdeckt. Durch eine anders geartete politische Ausrichtung für die Zukunft werden technisch priore Produkte eventuell schon im Vorfeld inakzeptabel. Die rein technisch orientierte positive Entscheidung kann durch Entscheidungen des Managements gekippt werden.

6.1. Abschließende Bewertung der evaluierten Systeme

Nachfolgend sollen nacheinander die Möglichkeiten "Tamino", "Autonomy" und "Eigenentwicklung" abschließend beleuchtet werden. Neben den spezifischen Gründen für oder gegen die einzelnen Lösungen sind noch die gemeinsamen Randparameter zu beachten. Zunächst einmal sind die Lizenzierungskosten für die Produkte Tamino und KnowledgeServer relativ hoch. An sich wäre diese Tatsache kein großes Problem, da Leistung in der heutigen Zeit Geld kostet. Problematisch jedoch ist, dass wahrscheinlich keines der beiden Produkte in der neuentwickelten Plattform Verwendung findet. Die einfachste und günstigste Lösung ist in jedem Fall die Optimierung der momentan implementierten PERL Suche. Damit kann der höchste Deckungsgrad zwischen Verschlagwortungskonzept und Implementierung erreicht werden.

6.1.1 Software AG – Tamino

Der wesentlichste Vorteil des Informationsserver Tamino ist, dass die Abbildung der externen Verschlagwortung möglich ist. Der verschlagwortende Dienstleister liefert zwar ein relationales Datenmodell, das aber mit einem gewissen Aufwand in die hierarchische Strukturierung des XML-Datenmodelles umgesetzt und damit vom Tamino bedient werden kann. Die Abbildung der Verschlagwortung ist eines der oben erwähnten politischen Ziele und stellt somit für das jeweilige Produkt ein K.O.-Kriterium dar. Der Informationsserver ist eine offene Plattform, die sich an Standards orientiert und komplett in Java programmiert ist. Java bietet bereits einige Ansätze

(RMI, CORBA, Beans, ...) für die Integration von Fremdfunktionalitäten. Durch die Offenheit und die gewählte Programmiersprache ist es möglich Produkte von Fremdherstellern zu integrieren. Der Informationsserver basiert auf XML als zukunftsweisender Technologie und ist für die Verwaltung strukturierter hierarchischer Daten gebaut.

Nach diesen positiven Gesichtspunkten und dem positiven Gesamteindruck bleibt nur der Absturz des Informationsservers beim Lasttest mit einem Kompletterlust der Daten als negativer Eindruck. Dieser Eindruck überwiegt die positiven Punkte, denn ein Datenbanksystem, das seine Daten verliert, verfehlt seine Basisziele.

6.1.2 Autonomy – Knowledge Server

Eigentlich hat sich der KnowledgeServer von Autonomy bereits durch die Tatsache disqualifiziert, dass die Verschlagwortung des externen Dienstleisters nicht abgebildet werden kann. Dieses Kriterium ist ja, wie bereits erwähnt, ein politisches K.O.-Kriterium. Die Vertriebsmannschaft von Autonomy ist aber guter Dinge und möchte beweisen, dass die maschinelle Verschlagwortung bzw. Kategorisierung einfacher, besser und schneller funktioniert, als die manuelle Variante vom Dienstleister. Es bleibt fraglich, ob die maschinelle Kategorisierung von Autonomy genauso leistungsfähig ist, wie die manuelle Verschlagwortung. Aber das zu beweisen wird die Technik- und Vertriebsmannschaft von Autonomy mit einem Prototypen versuchen.

Autonomy ist ein schnelles System zur Verwaltung von unstrukturierten Massendaten. Die Katalogdaten liegen als strukturierte Daten vor. Allein diese Tatsache und die fehlende Unterstützung der Verschlagwortung lassen den Knowledge Server als Lösungsmöglichkeit wegfallen. Weitere Produkte von Fremdherstellern lassen sich auf den Knowledge Server aufsetzen und integrieren bzw. auch der Knowledge Server kann auf andere Produkte aufgesetzt werden. Autonomy sieht die Suche im Knowledge Server eher als Abfallprodukt an. Die Kernaufgabe des Knowledge Servers ist die automatische Kategorisierung von unstrukturierten Massendaten und die Unterstützung von Personalisierung des Information Retrieval bzw. der Shoppingplattform.

Autonomy erfüllt die geforderten Kriterien nicht annähernd, ist aber als technologische Herausforderung zu werten, für die bei positivem Abschneiden des Prototypen, es durchaus anstrebenswert wäre, die geforderten Kriterien nochmals zu überdenken und aufzuweichen.

6.1.3 Eigenentwicklung

Das schlagkräftigste Argument für eine Eigenentwicklung bzw. eine Optimierung der bestehenden PERL Skripten ist die Möglichkeit eine 100%-ige Überdeckung des Verschlagwortungskonzeptes und der Implementierung zu erreichen. Die Qualität der Implementierung bzw. der Deckungsgrad der Implementierung mit dem Konzept ist durch die Auswahl eines kompetenten Entwicklers gut steuerbar. Bei der Eigenentwicklung sind die Aufwände und die Probleme bei der Integration vermutlich am geringsten, da vollkommene Transparenz über die Funktionalität der Lösung besteht. Das Entwicklungs-Know-How bleibt in der Firma und die Suche kann bei entsprechend offener Konzeptionierung zum Alleinstellungsmerkmal ausgebaut werden. Stimmt das Feinkonzept für die Implementierung und auch das Datenmodell, so kann die Eigenentwicklung schon als Prototyp für die neue Plattform angesehen werden. Die Implementierung muss nur noch in die Zielsprache des Application Servers portiert werden und kann dort ebenfalls als Alleinstellungsmerkmal dienen.

Die Eigenentwicklung bzw. die Verbesserung der implementierten Suche erscheint an dieser Stelle als die Lösung der Wahl. Allein die Übereinstimmung mit den politischen Zielen lassen die Eigenentwicklung gut aussehen. Wenn dann auch noch die Möglichkeit der flexiblen Ergänzung bis hin zum Alleinstellungsmerkmal bedacht wird, so ist die Eigenentwicklung das Mittel der Wahl.

6.1.4 Entscheidungsmatrix

Für eine objektive Entscheidungsgrundlagen, werden die Anforderungen aus Kapitel 3 in eine Entscheidungsmatrix eingebaut. Die Entscheidungsmatrix ist ein Hilfsmittel, mit dem regelbasierte Entscheidung getroffen werden können und somit subjektive Einflüsse weitgehend eliminiert werden können. Ziel für die Matrix war es die wesentlichen Kriterien des Soll-Modells auf genau ein DIN-A4-Blatt abzubilden.

Die *Eigenschaft* gibt das geforderte Kriterium wieder. Die *Gewichtung* entspricht der Skala von 1-20. Die Gewichtung 20 wird nur für absolute K.O.-Kriterien verwendet. In der *Kategorie* gibt es die Unterscheidung in Musskriterien (**M**) und nicht notwendige Kriterien (**N**). Die Lösungen (Eigenentwicklung / Tamino / Autonomy) können in den jeweiligen Zellen ein "X" (erfüllt), ein "-" (nicht erfüllt) oder ein "n.b." (nicht bewertet) haben. Die *Wertung* errechnet sich jeweils aus dem Produkt der Gewichtung mit der

Kategorie, wenn die Lösung in der jeweiligen Zeile ein **X** für vorweisen kann. Für die Kategorie **M** wird zudem noch der Faktor 2 in das Produkt mit einbezogen.

	Gewichtung	Kategorie	Eigen-entwicklung	Wertung	Timino	Wertung	Autonomy	Wertung
Rahmenparameter				344		284		328
Zielplattform – Sun Solaris	10	M	X	20	X	20	X	20
Funktionalität der existierenden Suche mindestens abdecken	10	M	X	20	X	20	X	20
Stabilität unter Lastbedingungen	20	M	X	40	-	-	X	40
Schnelle Lösung								
Antwortzeit < 7 Sekunden	10	M	X	20	X	20	X	20
DB-Anbindung nur über allgemeine DB-Treiber	3	M	X	6	X	6	-	-
native DB-Treiber für Sybase	5	M	X	10	-	-	-	-
native DB-Treiber für Oracle	5	M	X	10	-	-	-	-
Skalierbare Lösung								
verteilbare Applikation	7	N	-	-	X	7	X	7
annähernd lineare Skalierung	4	N	-	-	-	-	-	-
mehrere parallele Prozesse	10	M	X	20	X	20	X	20
aufsetzbar auf Oracle – Suchdatenbank	4	M	X	8	-	-	-	-
aufsetzbar auf Sybase – Suchdatenbank	4	M	X	8	-	-	-	-
eigenes Indexformat	3	N	-	-	X	3	X	3
leicht wartbare Lösung								
Verwendung von Standards	6	N	X	6	X	6	X	6
geringer Aufwand für Dateneinspielung	8	M	-	-	-	-	X	16
geringer Aufwand für Indexaktualisierung	6	M	-	-	-	-	X	12
geringer Aufwand personeller Natur für Administration	6	M	-	-	X	12	X	12
geringer Aufwand maschineller Natur für Administration	4	M	X	8	X	8	X	8
Pflege ohne manuelle Zuarbeit	3	N	-	-	-	-	X	3
Zukunftsorientierte Lösung								
offene Plattform – leicht und flexibel erweiterbar	10	M	X	20	X	20	-	-
Transparenz in der Applikation	8	M	X	16	-	-	-	-
Leistungsfähigkeit des Herstellers	10	M	n.b.	-	-	-	X	20
Glaubwürdigkeit des Herstellers	10	M	n.b.	-	-	-	X	20
Minimalfunktionalität								
Schlagwortsuche nach manuellem Indexkonzept	20	M	X	40	X	40	-	-
Suche nach Artikelnummer	10	M	X	20	X	20	X	20
verknüpfte Suche – Suchbegriffe mit boolschen Operatoren	10	M	X	20	X	20	X	20
Volltextsuche	10	M	X	20	X	20	X	20
Ermittlung einer optimalen Treffermenge	10	M	X	20	X	20	X	20
gewünschte Funktionalität								
Ähnlichkeit zu bestehenden Suchmaschinen – optisch und	2	N	X	2	X	2	X	2

funktionell								
Hilfestellungen für die Benutzer	5	N	X	5	X	5	X	5
gefundene Treffermenge verfeinern / vergrößern	5	N	X	5	X	5	X	5
Verfeinerung über logische Verknüpfung mehrerer Begriffe	3	N	-	-	-	-	-	-
Suche über Preisspannen	5	N	-	-	X	5	-	-
Suchhistorie	2	N	-	-	-	-	X	4
kundenspezifischer Cookie	4	N	-	-	-	-	-	-
unscharfe / phonetische Suche	5	N	-	-	X	5	X	5
Eigenentwicklung								
Interaktion mit der DB minimieren	10	M	X	-	n.b.	-	n.b.	-
SQL – Statements optimieren	10	M	X	-	n.b.	-	n.b.	-
Optimierung des Datenmodells	10	M	X	-	n.b.	-	n.b.	-
schlanker Code in der Applikation	10	M	X	-	n.b.	-	n.b.	-
Verwendung von Standards	10	M	X	-	n.b.	-	n.b.	-
Quellcode strukturieren	10	M	X	-	n.b.	-	n.b.	-
Quellcode intern dokumentieren	10	M	X	-	n.b.	-	n.b.	-
Quellcode extern dokumentieren	10	M	X	-	n.b.	-	n.b.	-

Tabelle 6-1 Entscheidungsmatrix

Der letzte Abschnitt für die Eigenentwicklung stellt die wesentlichen Bestandteile der zu erstellenden Eigenentwicklung dar. Das sind sozusagen die Knackpunkte für die Optimierungen. Diese sind nicht in die Bewertung mit eingeflossen.

Aus der Matrix geht letztlich folgende Bewertung hervor:

1. Eigenentwicklung 344 / 462 Punkte [75%]
2. Autonomy Knowledge Server 328 / 462 Punkte [70%]
3. Software AG Tamino 284 / 462 Punkte [61%]

6.1.5 Entscheidung für ein System

Die Entscheidung für ein System darf nicht nur anhand der Bewertung in der Matrix getroffen werden. Diese stellt zwar eine Entscheidungshilfe dar, aber die einzelnen Kriterien passen zum Teil nicht direkt auf die einzelnen Lösungskandidaten. Trotzdem deckt sich die Matrix mit der zuvor getroffenen Entscheidung. Das Mittel der Wahl ist hier und jetzt die Eigenentwicklung der Suche bzw. die Optimierung der implementierten Suche. Größte Bremse von Autonomy's Knowledge Server ist die Unmöglichkeit die Verschlagwortung umzusetzen. Beim Informationsserver Tamino ist die Stabilität nicht ausreichend.

6.2. Integrationsschritte ins existierende System

Für die Integration einer neuen Lösung in die existierende Shoppingapplikation sind folgende Komponenten wichtig: Kommunikation, Integration und Migration.

Die Kommunikation der Suche mit der Shoppingapplikation läuft über die CGI-Schnittstelle*. Übergabeparameter werden hierbei HTTP-konform in die URL encodiert. Die Kommunikationsstruktur kann von der alten Suche übernommen werden. Die Integration des neuen Suchmoduls in die Shoppingapplikation erfolgt dadurch, dass das komplette alte Suchmodul durch das neue Suchmodul ersetzt wird. Diese Vorgehensweise impliziert natürlich, dass die Funktionalität der alten Suche zu 100% durch die neue Suche abgedeckt werden muss. Das neue Suchmodul kann das alte Suchmodul ebenfalls nur dann ohne Probleme ersetzen, wenn alle Schnittstellen vorher genauestens analysiert wurden und von der Reimplementierung zu 100% abgedeckt werden. Beide Module müssen funktionell identisch sein.

Parallel zur Integration muss eine Migration erfolgen. Diese Migration bezieht sich im Wesentlichen auf die für die Suche relevanten Daten und das Datenmodell*. Für die Migration sind Schritte zu definieren, die Daten als auch Code umfassen.

6.3. Jetziger Status – prototypische Implementierung

Parallel zur Evaluierung der kommerziellen Produkte wurde das Ist-System weiter analysiert und gleichzeitig versucht die unklare Datenmodelloptimierung des Dienstleisters zu prüfen. Dabei wurde das Konzept der Verschlagwortung des einen Dienstleisters und die Implementierung des Konzepts durch den anderen Dienstleister überprüft. Dabei lag der Fokus darauf, zu prüfen, ob das Datenmodell des konzepterstellenden Dienstleisters, das bewusst normalisiert an den implementierenden Dienstleister geliefert wird, optimiert und damit geändert wurde. Parallel dazu wird durch die Erstellung eines Prototypen versucht die Datenbankzugriffe, die für die Verschlagwortung nach dem alten Modell nötig sind, zu reduzieren. Die Sourcen des Prototyps können im Anhang E nachgeschlagen werden.

6.3.1 Datenmodelloptimierung

Für die Datenmodelloptimierung wurde das normalisierte Datenmodell (basierend auf einer Access-Datenbank), das vom konzepterzeugenden Dienstleister an den implementierenden Dienstleister geliefert wird, herangezogen. Ziel des Prototypen ist

die funktionelle Nachbildung der Verschlagwortung, wie sie im momentanen Webauftritt zu sehen ist.

Bei der Analyse der PERL Skripten für die Suche, die auch die Verschlagwortung beinhaltet, wurden zunächst die verwendeten SQL-Statements und auch das Datenmodell untersucht. Das folgende SQL-Statement wird vom Dienstleister, der das Konzept erstellt, mitgeliefert, und operiert auf dem normalisierten Datenmodell. Das SQL-Statement soll lediglich als Beispiel dafür dienen, wie die Suche auf den Schlagwörtern ablaufen soll.

```
SELECT schlagwort.wort, produkt_schlagwort.wort_id, produkt_schlagwort.level1,
gruppe.gruppe_name, gruppe.gruppe_id, produkt_gruppe.bestellnummer
FROM schlagwort LEFT JOIN (gruppe RIGHT JOIN (produkt_gruppe RIGHT JOIN
produkt_schlagwort ON produkt_gruppe.bestellnummer = produkt_schlagwort.bestellnummer)
ON gruppe.gruppe_id = produkt_gruppe.gruppe_id) ON schlagwort.wort_id =
produkt_schlagwort.wort_id
WHERE (((produkt_schlagwort.wort_id)="wort_id"));
```

Ausschnitt 6-1 SQL-Statement für das normalisierte Datenmodell - Suchabfrage

Ein komplexes SQL-Statement, das die Datenbank stark belastet. Durch Vereinfachung des Datenmodells wurde die Abfrage stark vereinfacht. Vorrangiges Ziel dabei war die Eliminierung der Table-Joins.

```
SELECT gruppe_name, level1, gruppe_id FROM suchtabelle WHERE wort_id = "wort_id" ORDER
BY level1;
```

Ausschnitt 6-2 SQL-Statement für das denormalisierte Datenmodell - Suchabfrage*

Die Tabelle "Suchtabelle" ist aus der Abänderung des komplexen SQL-Statements entstanden. Für die Erstellung der "Suchtabelle" wird der komplexe Table-Join nur noch einmalig durchgeführt. Anschließend kommt das vereinfachte SQL-Statement zur Verwendung.

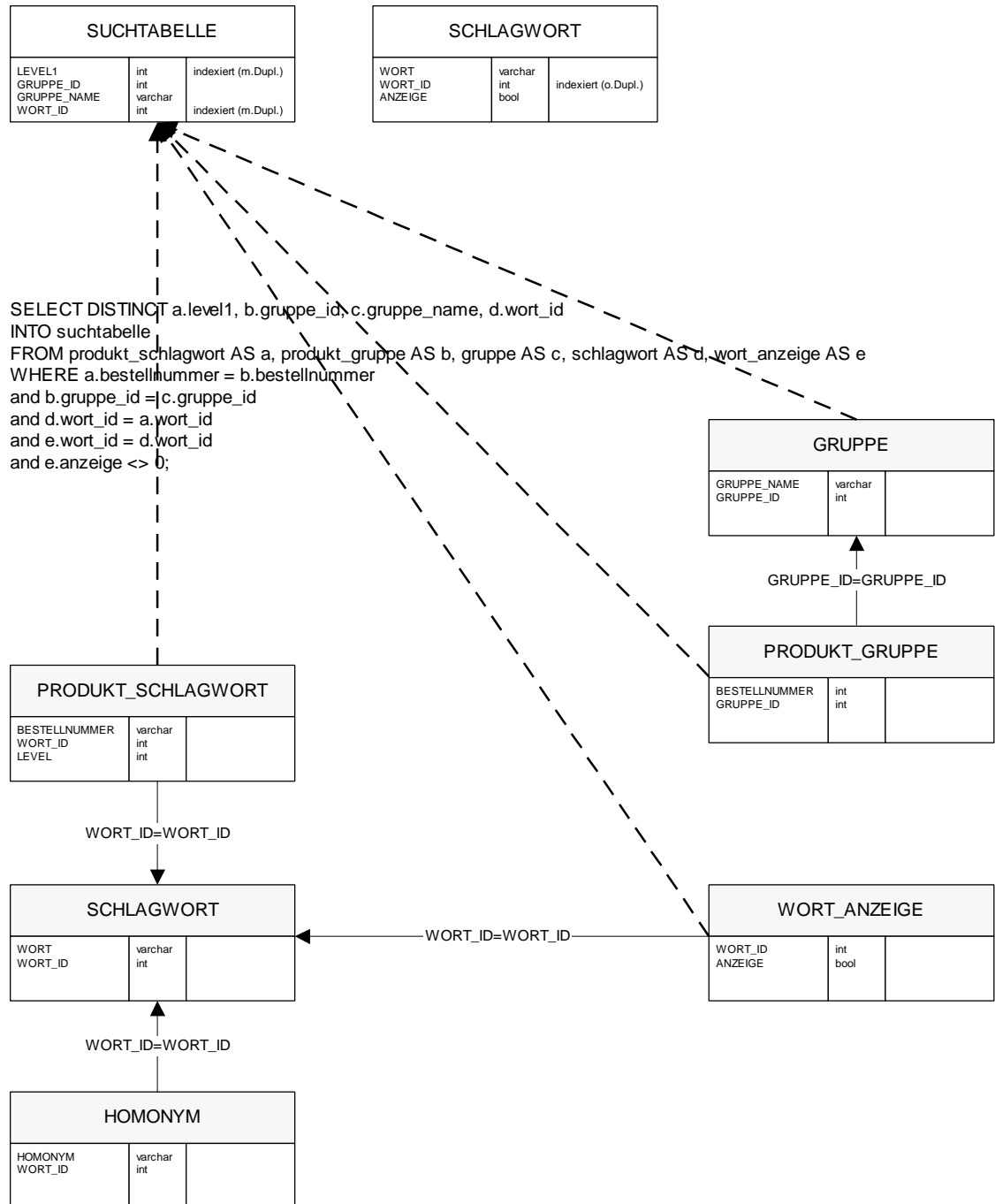


Abbildung 6-1 Übergang vom normalisierten (grauen) Datenmodell hin zum optimierten Datenmodell (farblos)

Das originäre Datenmodell (in Abbildung 6-1 grau dargestellt) wird durch den abgebildeten SQL-Statement in das "optimierte" Datenmodell (in Abbildung 6-1 farblos dargestellt) transformiert. Diese Transformation ist bei jeder Neulieferung der Daten nötig. Da ein Import in das Produktivdatenmodell sowieso von Nöten ist, ist das kein nennenswerter Mehraufwand.

6.3.2 prototypische PERL Applikation

Neben der Datenmodelländerung wurde die Suchapplikation prototypisch auf PERL basierend reimplementiert. Dabei war das Ziel die wesentliche Funktionalität der Suche – also die Verschlagwortung – auf dem geänderten Datenmodell durchzuführen. Die Funktionalität entspricht bei weitem noch nicht genau dem Original. Jedoch kann man klar erkennen, dass mit wesentlich weniger Aufwand zur Laufzeit ein grundsätzlich gleichwertiges Ergebnis erzeugt werden kann. Für den Prototypen wurde PERL in Verbindung mit Access und einer ODBC-Anbindung verwendet. Der PERL-Code ist funktionell in Ordnung, könnte aber hinsichtlich Performanz nochmals überarbeitet werden.

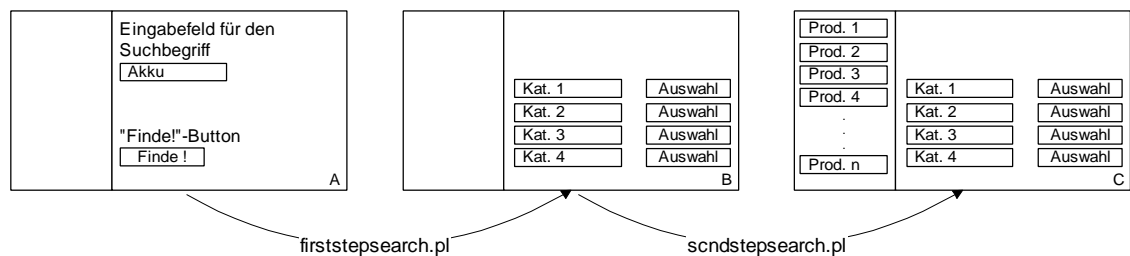


Abbildung 6-2 Ablauf im Prototypen für die Verschlagwortung

Der Ablauf der Suche ist der Folgende: In Seite A wird der Suchbegriff eingetippt und mit dem "Finde!"-Button die Suche gestartet. Dabei wird das PERL Skript *firststepsearch.pl* aufgerufen.

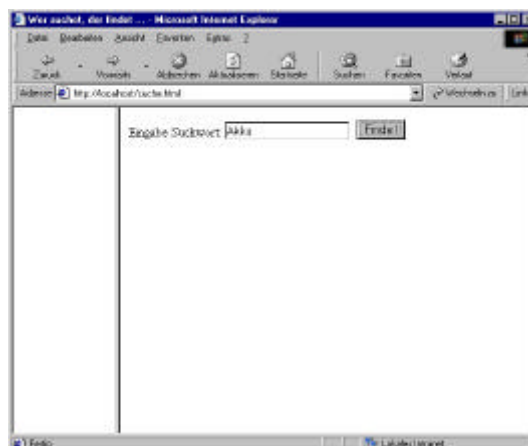


Abbildung 6-3 Erste Seite im Suchprototypen für die Verschlagwortung - Eingabe des Suchwortes

```

1: print "Content-type: text/html\n\n";
2: print "<html><head><title>CGI-Feedback</title></head>\n";

3: $daten = $ENV{'QUERY_STRING'};
...
4: print "<body>\n";

5: @form = split(/&/, $daten);
...
6:     if (! $db->Sql("SELECT wort, wort_id FROM schlagwort where wort=\"$suchwort\""))
        {
            ...
7:             while($db->FetchRow())
8:             {
9:                 undef %data;
10:                %data = $db->DataHash();

11:                $wort = $data{"wort"};
12:                $wort_id = $data{"wort_id"};
            }

...
13:    $sql = "SELECT gruppe_name, level1, gruppe_id FROM suchtabelle WHERE
        wort_id=$wort_id ORDER BY level1";
    ...
14:    if (! $db->Sql($sql))
        {
            ...
15:            print "<table width=\"50%\"><form target=\"left\" action=\"/cgi-
                bin/scndstepsearch.pl\"><tr><td>\n";

16:            while($db->FetchRow())
            {
                ...
17:                %data = $db->DataHash();

18:                print "</select><td><input type=submit name=\"submit$soldlevel\"
                    value=\"Auswahl\"></form><form target=\"left\" action=\"/cgi-
                    bin/scndstepsearch.pl\"><tr><td>\n";

...
19:                print "<select name=\"$soldlevel\" size=1>\n";

20:                $dummy = $data{"gruppe_name"};
21:                $temp = $data{"gruppe_id"};
22:                print "<option value=\"$temp\"> $dummy\n";
            }

23:            print "</select><td><input type=submit name=\"submit$soldlevel\"
                value=\"Auswahl\"></td></form></table>\n";
            ...
24:            print "</body></html>\n";

```

Ausschnitt 6-3 Codefragmente aus firststepsearch.pl

Zeile	Funktion
1-5	HTML-Header generieren und CGI-Parameter extrahieren
6-12	"wort_id" für das übergebene Suchwort (\$suchwort) aus der Tabelle "schlagwort" heraussuchen
13-15	anhand der vorher gefundenen "wort_id" die Ergebnismenge ermitteln; die Ergebnismenge besteht aus Produktgruppen, die je nach Relevanz zum gesuchten Artikel sortiert zurückgegeben werden (die Sortierung könnte eventuell durch einen sortiert vorgehaltenen Index optimiert werden);
16-23	Aufbau der Combo-Boxen, die die Produktgruppen enthalten
24	HTML-Seitenende generieren

Tabelle 6-2 Erläuterung des Sourcecodes zu firststepsearch.pl

Auf der Seite B hat der Benutzer nun die Möglichkeit anhand seiner Eingabe das Produkt seines Interesses näher zu spezifizieren. Er wählt aus den Combo-Klappboxen

eine Produktgruppe aus und wählt den jeweiligen "Auswahl"-Button. Jetzt tritt das PERL Skript *scndstepsearch.pl* in Aktion.

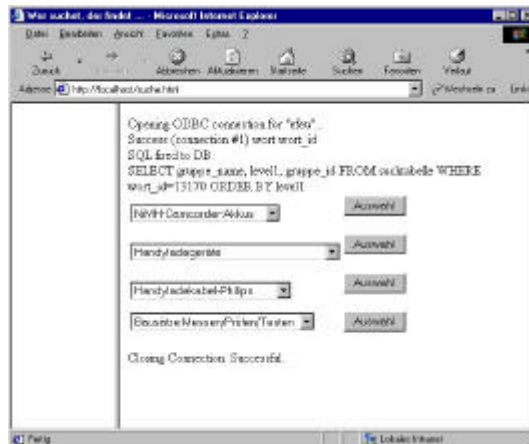


Abbildung 6-4 Zweite Seite im Suchprototypen für die Verschlagwortung – Anzeige der Produktgruppen und Selektion einer dieser Gruppen

```

1: print "Content-type: text/html\n\n";
2: print "<html><head><title>Search Results</title></head>\n";

3: $daten = $ENV{'QUERY_STRING'};

4: print "<body>\n";

5: @form = split(/&/, $daten);
...
6: if (! $db->Sql("select bestellnummer from produkt_gruppe where gruppe_id =
   $group"))
   {
...
7:     while($db->FetchRow())
   {
...
8:         %data = $db->DataHash();

9:         $bestellnummer = $data{"bestellnummer"};
10:        print "$bestellnummer<br>\n";
...
11:    }
    print "</body></html>\n";

```

Ausschnitt 6-4 Codefragmente aus *scndstepsearch.pl*

Zeile	Funktion
1-5	HTML-Header generieren und CGI-Parameter extrahieren
6	Bestellnummer der Artikel herausselektieren, die der gewählten Produktgruppe zugeordnet sind
7-10	Ergebnismenge darstellen
11	HTML-Seitenende generieren

Tabelle 6-3 Erläuterung des Codes zu *scndstepsearch.pl*

Nach dem Ablauf des PERL Skriptes *scndstepsearch.pl* werden die Produkte auf der linken Seite angezeigt.

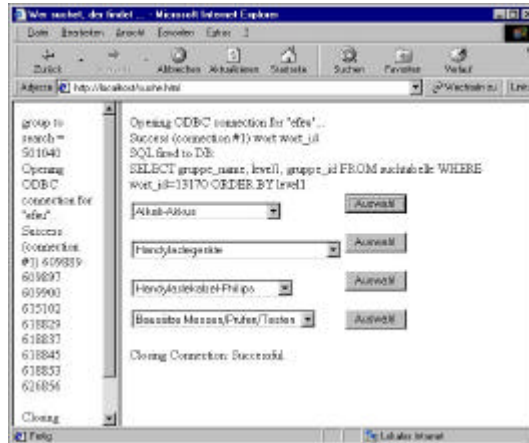


Abbildung 6-5 6-6 Dritte Seite im Suchprototypen für die Verschlagwortung – Anzeige der Produkte

Mit diesen Skripten ist grundsätzlich die Funktionalität der Verschlagwortung abgebildet. Die nächsten Schritte werden sein, den Dienstleister von der Dringlichkeit der Überarbeitung der Sourcen zu überzeugen. Die optimierte und überarbeitete Suchfunktionalität soll anschließend auf den Backup*-Datenbankserver ausgelagert werden.

7. Ausblicke auf neuere / alternative Techniken

Nachdem im obigen Teil der Diplomarbeit detailliert auf einige wenige Techniken zur Lösung der Problematik in der Suche der Conrad.com AG eingegangen wurde, sollen nachfolgend noch einige alternative Techniken für die Lösung der Suchproblematik vorgestellt werden. Inwiefern diese Technologien tatsächlich zur Lösung beitragen können, kann nur anhand des Potentials der einzelnen Lösung abgeschätzt werden. Für eine genauere Beurteilung sollte eine Installation und ein Test bzw. eine Evaluierung des jeweiligen Systems in Betracht gezogen werden. Es gibt auch noch viele weitere potentielle Ansätze und Technologien, die ebenfalls als Kandidaten für die Lösung der Problematik in Betracht kommen. Diese wurden jedoch nicht weiter betrachtet und nachfolgend drei potentielle Verbesserungen fokussiert. Zunächst wird der Einsatz von Oracle als RDBMS in Verbindung mit den Parallel Query Optionen erläutert. Anschließend wird der Oracle Parallel Server in Verbindung mit einem SUN-Cluster^{*} betrachtet. Abschließend wird der Prototypen um eine unscharfe Suche erweitert. Die einzelnen Technologien unterscheiden sich z.T. sehr stark in den einzelnen Randparametern. Die Lösung mit SUN-Clustern^{*} beispielsweise ist extrem teuer. Trotzdem sollen diese Möglichkeiten zumindest als theoretisch interessant in Betracht gezogen werden.

7.1. Einsatz von Oracle Datenbank in Verbindung mit Parallel Query Execution

[ODC99a] Grundsätzlich ist bei der Conrad.com AG eine Abkehr vom Datenbanksystem Sybase zu beobachten. Die Tendenz läuft stark in Richtung Marktführer Oracle. Beim RDBMS Oracle ist ein Feature besonders interessant in diesem Zusammenhang: die Parallel Query Execution – also die parallele Ausführung einer SQL-Anweisung. Grundsätzlich werden SQL-Anweisungen nicht parallelisiert, sondern nacheinander durch einen einzelnen Prozess abgearbeitet. Im Oracle DBMS steht ab der Enterprise Edition die Parallel Query Execution zur Verfügung. Dabei können mehrere Prozesse gleichzeitig an der Ausführung einer einzelnen SQL-Anweisung arbeiten. Die Ausführung der Abfrage wird schneller, wenn die Gesamtarbeit, die für die Ausführung der SQL-Anweisung nötig ist, unter mehreren parallelen Prozessen aufgeteilt werden kann. Durch diese Art der Abarbeitung von

SQL-Anweisungen wird vorhandene Hardware (idealerweise mehrere CPU's) optimal ausgenutzt.

Um die parallele Abarbeitung von SQL-Anweisungen effektiv zu unterstützen ist es sinnvoll die verwendeten Tablespaces zu "stripen". Dadurch werden potentielle I/O-Flaschenhälse während der Bearbeitung vermieden. Die Tablespaces sollten dabei auf mindestens so vielen Geräten verteilt sein, wie CPUs vorhanden sind. Das folgende Bild zeigt eine Stripingmöglichkeit für 4 CPUs und 4 Tablespaces.

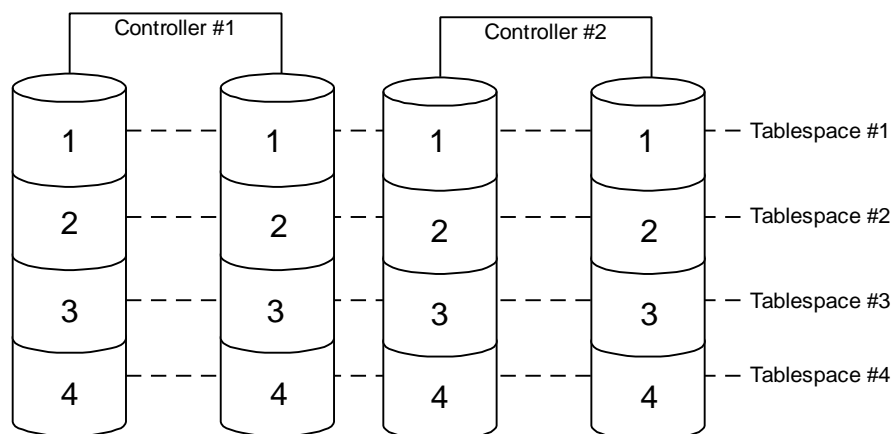


Abbildung 7-1 Vier Tablespaces gestript auf vier unterschiedliche Geräte

Neben der physischen Ablage auf den einzelnen Speichermedien ist auch die Partitionierung der Tabellen und Indexe von Bedeutung für die performante Ausführung der parallelen SQL-Anweisungen. Partitionierung bedeutet hier die Einteilung von Daten anhand Kriterien in kleinere logische Portionen. Die Einteilung kann anhand eines Bereichskriterium geschehen (range), mit einer Hashfunktion (hash) oder durch die gemischte Anwendung beider Methoden (composite). Durch die Partitionierung wird der Daten- oder Indexbestand bereits in trennbare Portionen, die auch für die parallele Bearbeitung geeignet sind, aufgeteilt.

Oracle parallelisiert nach Aktivierung der korrekten Optionen die SQL-Anweisungen automatisch. Der Grad der Parallelisierung, also die Anzahl der aktiven Prozesse pro SQL-Anweisung kann justiert werden. Dies kann durch die Einstellung der globalen Optionen geschehen, aber auch direkt für eine Tabelle fixiert werden.

Oracle parallelisiert Anfragen nach diesem Schema:

- Parallelisierung in Bereichen bei Scan-Operationen (SELECTs und Subabfragen in DML & DDL (DML = Data Manipulation Language; DDL = Data Definition Language))

Hierbei wird die Anfrage dynamisch zur Laufzeit parallelisiert. Dabei wird die Tabelle oder der Index in Bereiche von Datenbankblöcken unterteilt und auf jedem Bereich wird die Abfrage ausgeführt.

- Parallelisierung in Partitionen

Partitionen sind wie schon erwähnt eine logische, statische Unterteilung von Tabellen und Indexen, die dazu verwendet werden kann, langlaufende Operationen in kleinere Operationen zu unterteilen, die auf einzelnen Partitionen ausgeführt werden. Die kleinste Granularität der Parallelisierung ist hier die Partition. Innerhalb der Partition gibt es keine Parallelisierung. Operationen auf partitionierten Tabellen und Indexen werden parallel ausgeführt, indem verschieden parallele Serverprozesse zu verschiedenen Partitionen der Tabelle oder des Index zugeordnet werden. Operationen auf partitionierten Tabellen und Indexen werden nur dann parallel ausgeführt, wenn auf mehr als eine Partition zugegriffen wird.

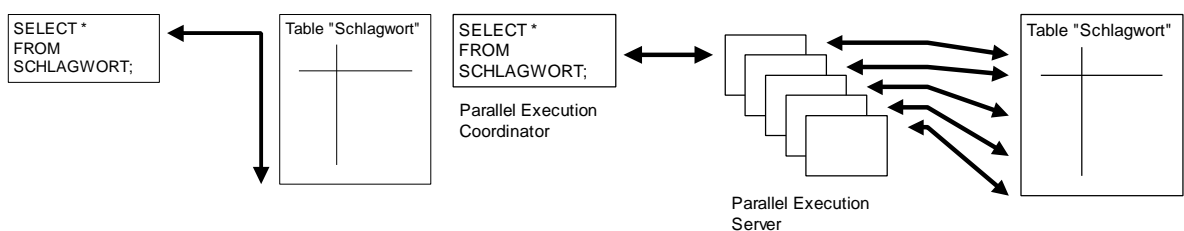


Abbildung 7-2 links die serielle Abarbeitung einer SQL-Anfrage, rechts die parallele Abarbeitung

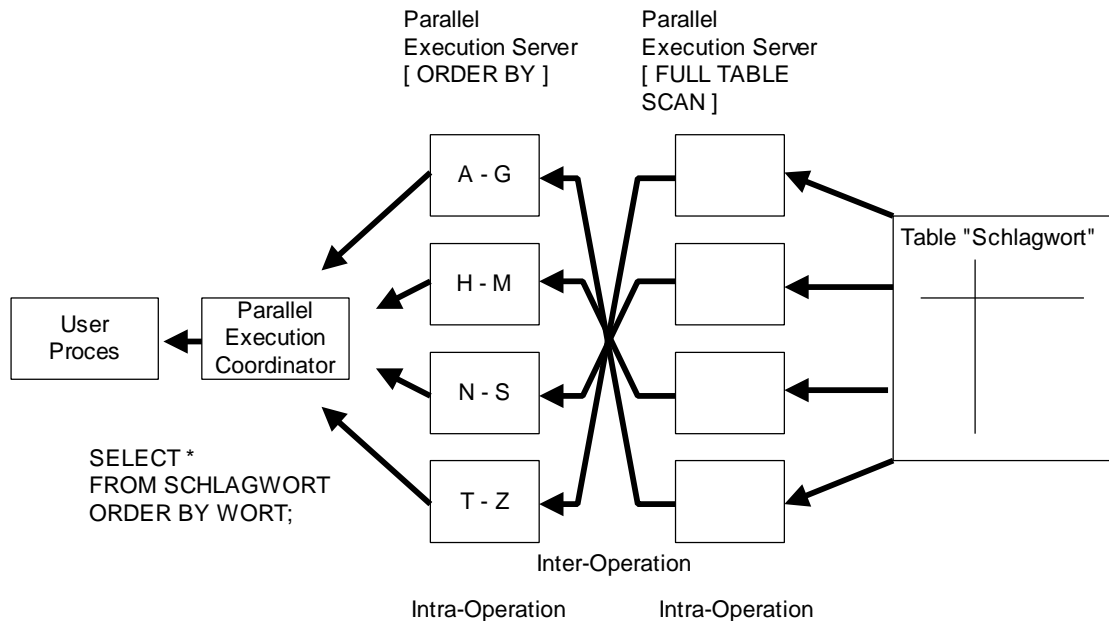


Abbildung 7-3 Abarbeitung einer beispielhaften SQL-Anweisung durch 8 parallele Prozesse

Nachdem kurz die prinzipielle Funktionsweise der Parallel Query Option der Oracle Enterprise Edition erläutert wurde, soll der Nutzen für die Shoplandschaft der Conrad.com AG dargestellt werden. Der wesentlichste Nutzen der parallelen Abarbeitung der SQL-Anfragen ist die schnellere Bearbeitung durch Zerlegung der großen Gesamtaufgabe in kleinere Teilaufgaben. Gerade bei der Conrad.com AG, wo Systeme mit mehreren CPUs zur Verfügung stehen, ist eine Beschleunigung der SQL-Anfragen zu erwarten. Die Lösung von Oracle beschleunigt die SQL-Anweisungen und hat noch den Vorteil, dass grundsätzlich das bereits verwendete Datenmodell weiterhin verwendet werden kann. Das aktuell verwendete Datenmodell unter der Sybase-Datenbank würde zwar sowieso bei einer Migration auf eine Oracle-Datenbank überarbeitet werden, könnte aber unverändert mit den Parallel Query Options betrieben werden. Bei der Migration sollte lediglich die Partitionierung der Daten und Indexe überdacht werden. Gerade im Bereich der Suche – speziell die z.T. recht komplexen SQL-Anweisungen in der aktuellen Applikation – kann eine Performanzsteigerung erreicht werden. Einziger Wermutstropfen bei der Verwendung paralleler Abarbeitung von SQL-Anweisungen ist der erhöhte Kommunikations- und Synchronisationsaufwand zwischen den einzelnen Prozessen.

Zudem könnte man noch prüfen inwieweit es möglich ist, die Sybase Datenbank in Verbindung mit Parallel Query Execution zu verwenden.

7.2. Einsatz von SUN-Clustern* / Oracle Parallel Server

Neben der Parallelisierung auf Abfrageebene bietet sich auch die Hardwareebene zur Parallelisierung an. Um eine Parallelisierung auf Hardwareebene zu erreichen bietet sich zunächst einmal der Einsatz von SUN-Clustern* und darauf basierend der Oracle Parallel Server an. Nachfolgend wird auf SUN-Cluster* und anschließend auf den Oracle Parallel Server eingegangen. Dabei sollen grundlegende Merkmale erwähnt werden und Erfahrungswerte für die Verwendbarkeit bei der Conrad.com AG berücksichtigt werden.

7.2.1 SUN-Cluster*

[SUN97] Ein Hardware-Cluster* ist eine systematische Anordnung von Serversystemen mit dem Ziel Performanz zu gewinnen und über Redundanz der Funktionalität einen hohen Grad an Ausfallsicherheit zu erreichen.

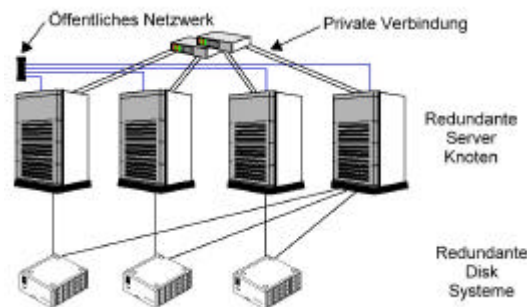


Abbildung 7-4 Ein Cluster*

Die einzelnen Serverknoten im Cluster* greifen auf unterschiedliche oder auch die gleichen Speichermedien zurück. Für die Anordnung der Serverknoten sind unterschiedlichste Topologien (Ring, Stern, Clustered Pairs, ...) denkbar. Jede Topologie erfüllt unterschiedliche Anforderungen mehr oder weniger gut. Wie schon an der obigen Abbildung zu ersehen ist, ist die Bildung von Clustern eine sehr teure Angelegenheit. Um redundante Applikationsumgebungen im Sinne der Ausfallsicherheit zu schaffen, müssen äquivalente Computersysteme ausgewählt werden. Je höher der Grad an Sicherheit, desto mehr Computersysteme müssen für Redundanz sorgen. Problematisch an Clustern ist, dass der Leistungsgewinn auf gar keinen Fall linear ist. Bildet man einen Cluster aus zwei Rechnern hat man nicht die doppelte Leistung. Der Overhead zur Kommunikation und Synchronisation wird mit

Anzahl der Rechner immer größer. Dies geht soweit, dass ab einer bestimmten Anzahl an Rechnern der Leistungsgewinn durch Zufügen eines weiteren Rechners komplett durch den erhöhten Kommunikationsoverhead aufgebraucht wird. Das Gesamtsystem wird nicht performanter. In kleineren Clustern bietet aber die Möglichkeit, einfach ein neues Computersystem in den Cluster zu integrieren, eine einfache und elegante Möglichkeit der Leistungsskalierung. Beim Cluster stehen immer Anforderungen mit starkem Sicherheitsbezug im Vordergrund. Dazu zählen vor allem: Hochverfügbarkeit, skalierbare Performanz und Redundanz um "Single Points of Failure" zu eliminieren. Neben den Sicherheitsaspekten löst der Cluster auch Wartungsprobleme. Im Cluster ist die Funktionalität der Applikation oder der Datenbank auf mehrere Rechner verteilt. Dadurch kann für kurze Zeit durchaus ein Rechner aus dem Cluster für Wartungstätigkeiten entfernt werden. Für diese Zeit werden die Aufgaben des Rechners vom Cluster übernommen. Ein Cluster funktioniert nach dem Prinzip "Alle für einen – Einer für alle", das ja schon im Mittelalter funktionierte, wenn man Alexander Dumas' [DUM44] glauben schenken darf.

Neben der reinen Hardware benötigt man für den Betrieb einer Clusterlösung natürlich auch Software. Da ist zum einen das Betriebssystem – hier: SUN Solaris – und die zu betreibende Applikation – hier: Oracle Parallel Server – zu betrachten. SUN bietet für Solaris eine clusterfähige Zusatzsoftware an, die für die Verwaltung und Steuerung des Clusters auf Betriebssystemebene zuständig ist. Der reine Cluster auf Betriebssystemebene ist performant und wird bereits häufig eingesetzt.

7.2.2 Oracle Parallel Server

[ODC99b] Beim Oracle Parallel Server wurden durch einen unserer Dienstleister praxisnahe Erfahrungen mit dem OPS (Oracle Parallel Server) zur Verfügung gestellt. Prinzipiell ist der OPS eine verteilte Datenbank, bei der verschiedene Instanzen der Datenbank auf unterschiedlichen Clusterknoten* arbeiten und auf ein gemeinsames Stagesubsystem* zugreifen. Die Erfahrungen mit dem OPS sollen hier als praxisnahe Aussage wiedergegeben werden. Die Anwendung, die auf den OPS aufsetzt, sollte verteilt auf verschiedenen Servern laufen. Der Zugriff auf die Datenbank erfolgt in diesem Zuge über verschiedene Server. Für die Datenbank wird eine gut gesicherte Storage-Einheit benötigt, die die Datenbankfiles speichert. Als Clusterknoten* sind zwei identische Systeme nötig, die mit der Speichereinheit umgehen können müssen. Neben

der Hardware muss auch noch bei der Datenbank einiges beachtet werden. Das Design der Datenbank muss bereits im Vorfeld auf den Einsatz des OPS angepasst werden. Nicht nur das Design muss angepasst werden, sondern auch Datenbankjobs müssen explizit auf eine einzelne Instanz verteilt werden, da diese Jobs sonst eventuell auf beiden Instanzen der Datenbank ablaufen. Für den Betrieb des OPS müssen spezielle Einstellungen und Parameter mit Bedacht gesetzt werden. Der Abgleich der Instanzen des OPS erfolgt über den Distributed Lock Manager, der ebenfalls über Parameter konfiguriert werden muss. Alle getroffenen Einstellungen müssen während des Betriebs angepasst und überwacht werden. Im Parallelbetrieb kann es, abhängig von der Art des Zugriffs auf die Datenbank und dem Replikationsaufwand, zu einem Performanceeinbruch im Vergleich zur Single-Instanz-Installation kommen. Das bedeutet effektiv, dass trotz der doppelten Hardware und der aufgebauten Redundanz die Konfiguration im Cluster weniger schnell arbeitet, als eine Installation auf einem einzelnen Rechner. Dieser Effekt hängt stark von der Inter-OPS-Kommunikation* ab. Grundsätzlich ist der Betrieb einer Oracle Parallel Server Installation mit sehr viel Administrationsaufwand verbunden. Für den Betrieb müssen Experten vor Ort sein, die sich mit dem OPS sehr gut auskennen und Erfahrung auf diesem Gebiet haben. Die Probleme, die beim OPS auftreten sind grundsätzlich anderer Natur, als die, die bei der Single-Instanz-Installation auftreten.

Nach dieser geschilderten Erfahrung und den gegenteiligen Schilderungen von Oracle ist diese Variante ob der hohen Kosten – software- und hardwareseitig – und dem zu erwartenden Aufwand als eher ungeeignet zu betrachten.

7.3. Verwendung phonetischer Algorithmen für eine bessere Treffermenge

Die derzeit verwendete Suche ist Fehlern gegenüber intolerant und funktioniert nur bei Eingabe korrekter Suchwörter. Vertippt sich der Benutzer oder ist sich über die Schreibweise des Produktes oder der Kategorie unsicher, so kommt er momentan nicht zum Ziel. Gibt er beispielsweise "Feschdblade" ein und meint damit "Festplatte", so ist in der derzeitigen Implementierung keine Fehlertoleranz implementiert. Recherchen zum Thema "Unscharfe Suche" haben zunächst den Algorithmus "soundex" hervor gebracht. Nachdem dieser für den englischen Sprachgebrauch gebaute Algorithmus im deutschen Sprachgebrauch nicht zur 100%-igen Zufriedenheit arbeitet, aber schon

beachtliche Resultate liefert, wurde nach weiteren Recherchen im Internet der Algorithmus "phonet" vom c't-Autor Jörg Michael [CT99a] entdeckt. Dieser Algorithmus arbeitet zu unserer Zufriedenheit und wird wohl in den nächsten Wochen in die Shoppingapplikation integriert werden. Nachfolgend werden die einzelnen Algorithmen kurz beschrieben und auf die Änderungen eingegangen, die am Prototypen zu machen waren.

7.3.1 Algorithmus "Soundex"

[KNU97] Margaret K. Odell und Robert C. Russel patentierten das "Soundex phonetic comparison system" im Jahre 1918 und 1922. Die Soundex Codierung erstellt aus einem englischen Wort eine vierstellige Repräsentation, die die Lautsprache des Wortes ausdrücken soll. Soundex wird normalerweise für unscharfe Suche verwendet, wo Schreibfehler mit einer möglichst guten Treffermenge beantwortet werden sollen. Rechtschreibprogramme beispielsweise zeigen alternative Möglichkeiten für ein falsch geschriebenes Wort an, dessen Soundex-Code berechnet wurde und mit den gespeicherten Codes der Alternativen übereinstimmt. Zusätzlich werden Soundex-Codes oft für Namen verwendet, da dort ein hohes Potential an Schreibfehler verborgen ist.

Die phonetische Repräsentation eines Wortes bildet der Algorithmus auf einfache Weise. Zunächst werden alle nichtenglischen Buchstaben und Symbole aus dem Ursprungswort entfernt. Buchstaben mit Accents werden ohne Accent verwendet. Gedankenstriche, Leerzeichen und ähnliche Zeichen werden einfach entfernt. Zusätzlich werden noch alle Buchstaben "H" und "W" entfernt, solange diese nicht die Anfangsbuchstaben des Wortes sind. Anschließend wird der erste Buchstabe des Wortes zum ersten Buchstaben des Soundex Codes. Jeder übrigbleibende Buchstabe wird nach der folgenden Tabelle in eine Zahl übersetzt und zum Soundex Code zusammengefügt.

Buchstabe	Code
A, E, I, O, U, Y	0
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

Tabelle 7-1 Konstruktionsregeln des Soundex Codes

Anschließend werden gleiche aufeinanderfolgende Zahlen durch nur einen Repräsentanten dieser Zahl ersetzt. Weiterhin wird die erste Zahl des Soundex Codes gelöscht, wenn der Repräsentant des ersten Buchstaben gleich der ersten Zahl ist. Jetzt werden alle Nullen aus dem Code gelöscht. Letztlich werden die ersten vier Zeichen des Soundex Codes als Ergebnis des Algorithmus zurückgegeben. Ist der zurückzugebende Code kürzer als vier Zeichen, so wird er mit Nullen aufgefüllt.

Dieser Algorithmus wurde für den Prototypen etwas abgewandelt. Das erste Zeichen des Soundex Codes ist nicht der erste Buchstabe des Eingangswortes, sondern die zugeordnete Zahl. Zudem wurde der besseren Streuung wegen der Soundex von vier Zeichen auf acht Zeichen erweitert.

7.3.2 Algorithmus "phonet"

[CT99a] Bei der Recherche nach geeigneten Transformationsalgorithmen wurde auf eine Anfrage in diversen Newsgroups auf einen Beitrag des Computermagazins "c't" hingewiesen. In seinem Beitrag stellt der Autor Jörg Michael seine Entwicklung eines fehlertoleranten Stringverarbeitungsalgorithmus für die deutsche Sprache vor. Sein erklärtes Ziel – die fehlertolerante Stringverarbeitung – ist mit dem des Soundex-Algorithmus identisch. Für den deutschen Sprachraum jedoch stellt Soundex durch die Löschung aller Vokale keine richtige Alternative dar. Der Soundex bietet durch seine Vorgehensweise eine recht grobe Abbildung des Wortes auf die jeweilige phonetische Repräsentation. Gerade in der deutschen Sprache ist es aber sehr wichtig, dass die Abbildung kontextsensitiv* und nicht einfach stur nach einem einfachen Regelsatz vorgeht. Beim Phonet Algorithmus wird das "CHS" in "Dachs" korrekt in ein "X" übersetzt → "DAX", wohingegen "Blechschmidt" nicht nach "Blexschmidt" umgesetzt wird. Dies wird über ein komplexes Regelwerk erreicht, das vom Umfang her der Regelsprache der Compilertools LEX* und YACC* nahe kommt. Die einzelnen Regeln zu programmieren bedeutet natürlich einen höheren Aufwand. Die Regeln sind untereinander priorisiert, so dass "GS" zuerst nach "X" umgesetzt wird, bevor "G" nach "K" transformiert wird. Das komplette Regelwerk enthält ca. 650 Regeln, die im Artikel näher beschrieben werden. Um einen näheren Eindruck zu erhalten ist auch ein Blick in den Code im Anhang zu empfehlen. Der Sourcecode untersteht der "GNU Public Licence" und darf frei kopiert und geändert werden. Das komplette Regelwerk ist in einem Characterarray namens "phonet_rules_german" definiert. Der Phonet

Algorithmus kann zwei Varianten der phonetischen Repräsentation errechnen. Variante eins hat einen geringeren Informationsverlust bei der Transformation zur Folge und bildet das Wort genauer ab. Diese Variante ist besser geeignet bei sehr großen Datenbeständen, da die ähnlichen Worte minimiert werden. Die zweite Variante löst sich stärker vom Wort und bildet das Wort abstrakter ab. Diese Variante ist eher geeignet für kleinere Datenbestände, da eher ähnliche Worte gefunden werden. Im Prototypen wird aufgrund des Datenbestandes die Variante zwei gewählt. Der Quelltext des Phonet Algorithmus steht unter [CT99b] zum Download zur Verfügung.

7.3.3 Änderungen am Suchprototypen

Der in Kapitel 6.3 vorgestellten Prototyp für die Suche wurde auch um die phonetischen Elemente erweitert. Ein laufender Prototyp findet leichter Zustimmung, als nur die reine Versprechung "Ja, es geht".

7.3.3.1. Datenmodell

Das Datenmodell muss kaum verändert werden. In der Tabelle "schlagwort" wird das numerische Attribut "soundex" und das alphanumerische Attribut "phonet" hinzugefügt und über ein PERL Skript mit den passenden Werten versorgt. Das PERL Skript generiert aus einer Textdatei, die separiert durch Semikolon die Attribute "wort", "wort_id" und "anzeige" enthält eine Textdatei, die zusätzlich noch die adäquaten Attribute "soundex" und "phonet" enthält. Die Attribute werden mit den gleichen Algorithmen berechnet, wie später für die Transformation des Suchwortes verwendet werden.

wort	wort_id	anzeige	soundex	phonet
Festinduktivitäten	29894	Nein	12353231	FEZTINTUKTIFITETN
Festmacherboot	29895	Nein	12352613	FEZTNAKABUT
Festmacherboot Vento	29896	Nein	0	
Festnetz	29897	Nein	123532	FEZTNEZ
Festo	29898	Nein	123	FEZTU
Festplatte	29899	Nein	123143	FEZTBLATE

Tabelle 7-2 Ausschnitt aus der Artikeldatenbank mit phonetischen Attributen

7.3.3.2. Applikation

In der Applikation gibt es noch einige Änderungen. Um die phonetische Repräsentation des Suchwortes auf Basis des "phonet" Algorithmus zu berechnen wird das CGI-Programm* *phonet.exe* zwischen die Seiten "B" und "A" geschaltet. Das CGI-

Programm*, das in C geschrieben ist, errechnet die phonetische Repräsentation des Suchwortes und hängt das Schlüssel-Wert-Paar "&phonet=XXX" an die URL zum Aufruf des CGI Skriptes *firststepsearch.pl* an. Im Skript wird nun der Schlüssel "phonet" und dessen Wert für die Suche in der Datenbank nach phonetischen Äquivalenten verwendet. Im Prototyp kann alternativ die Soundex Implementierung oder aber die Phonet Implementierung verwendet werden. Die Auswahl erfolgt auf der Seite A mit einem Radio-Button.

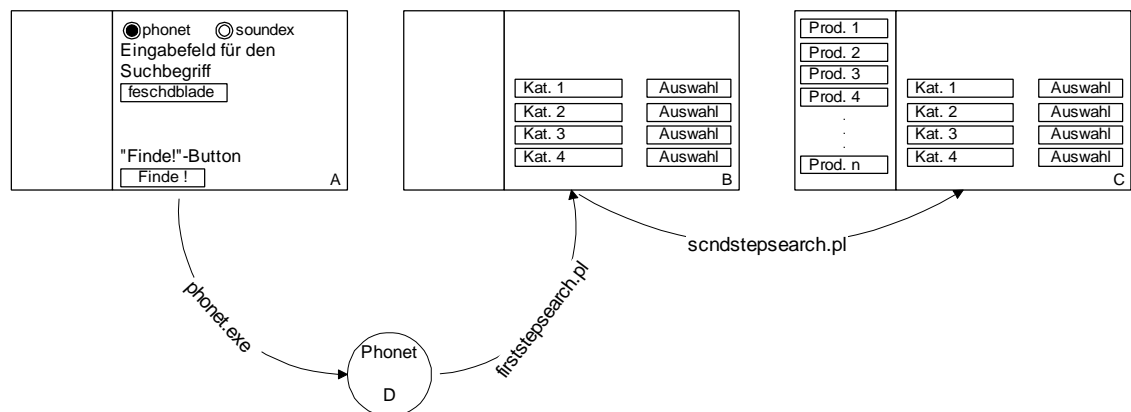


Abbildung 7-5 Ablaufplan des Prototypen für die Verschlagwortung - mit phonetischen Erweiterungen

Auf Seite B wird ausgegeben, ob der "Non-Phonetic" oder der "Phonetic" Suchweg zum Einsatz gekommen ist. Wird das Suchwort gleich in der Datenbank gefunden, so ist das eine nichtphonetische Suche. Ergibt die Anfrage an die Datenbank mit dem originalen Suchwort allerdings keine Treffermenge, so wird entweder die Soundex- oder die Phonet-Repräsentation des Suchwortes für die Bildung der Treffermenge verwendet. Hierbei werden alle Schlagwörter aus der Tabelle "schlagwort" herausselektiert, die entweder mit der Soundex- oder der Phonet-Repräsentation des Suchwortes übereinstimmen. Sie bilden die Treffermenge.

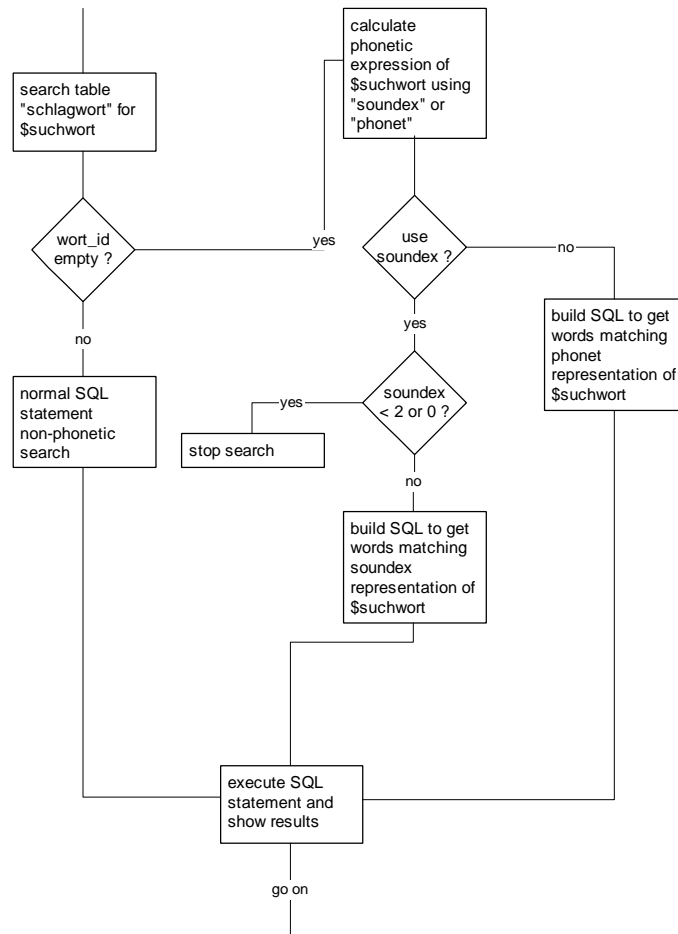


Abbildung 7-6 Ablaufplan des Prototypen für die Verschlagwortung mit phonetischen Ergänzungen

Die Ergebnisse des Soundex-Algorithmus sind weniger brauchbar, als die des Phonet-Algorithmus. Der Algorithmus Phonet bildet für die deutsche Sprache brauchbarere Repräsentationen der einzelnen Schlagworte. Der Algorithmus erzeugt eine bessere Streuung für die Schlagworte. Während Soundex für "Feschedplatte" einen Wert (123143) errechnet, der auf drei weitere Schlagworte (Festplatte, Epoxyd-Platte, Epoxydplatte) zutrifft, berechnet der Phonet eine Repräsentation (FEZTBLATE), die genau nur auf das Schlagwort "Festplatte" abbildet. Der Phonet Algorithmus berechnet beispielsweise auch für die Eingaben "Festplade", "Veschdblade", "Fesdblade" und weitere ähnliche die Repräsentation "FEZTBLATE". Damit ist der Nutzen des Phonet-Algorithmus natürlich wesentlich höher, da der Benutzer trotz der Eingabe haarsträubender Begriffe die Produkte in der Treffermenge erhält, die er (hoffentlich) auch möchte.

8. Schlusswort und Erstellungserklärung

8.1. Schlusswort

Die Diplomarbeit. Nun ist sie also geschafft. Innerhalb von 14 Wochen wurde die Diplomarbeit erstellt. Warum so eine Hetze ? Der Job wartet. Die Firma Conrad.com AG hat mir bereits bei Beginn der Erstellung der Diplomarbeit einen Job angeboten. Voraussetzung für diesen Job war allerdings die schnelle Durchführung der Diplomarbeit zu einem firmennahen Thema.

Die Diplomarbeit ist entstanden in einem Spannungsfeld zwischen "Daily Work" und dem eigenen Anspruch an die wissenschaftliche Gründlichkeit. Dank meiner Arbeitskollegen Stefan Armbruster, Urs Gulba und meinem "Chief Technology Office" Michael Thomas habe ich jedoch immer genügend Zeit für meine Diplomarbeit gehabt. Trotzdem ist ein inneres Spannungsfeld immer vorhanden. Eigentlich möchte man ja schon gerne an den kniffligen Problemen des täglichen Geschäftes teilhaben und an Lösungen arbeiten, doch vorerst sollte ja die Diplomarbeit erledigt werden.

Ebenfalls an dieser Stelle möchte ich meiner Betreuerin Prof. Dr. Lore Kern-Bausch für die freie Hand bei der Erstellung der Diplomarbeit bedanken. Nachdem ich ein fundiertes Gerüst der Diplomarbeit mit ihr abgestimmt habe, konnte ich bei fachlichen Fragen immer auf die Unterstützung ihrerseits bauen und dennoch nach meinem Arbeitsstil vorgehen.

Neben meinen IT-Arbeitskollegen möchte ich allen Arbeitskollegen der Conrad.com AG für das Verständnis danken, dass sie mir entgegen gebracht haben. Selten wurde ich mit diplomarbeitsfremden Themen betraut.

Ganz zum Schluss möchte ich natürlich auch meiner Frau danken, für das Verständnis und die freie Hand, die ich jederzeit bei der Erstellung meiner Diplomarbeit hatte. Sie war mir jederzeit eine wirkliche Hilfe.

8.2. Erstellungserklärung

Erklärung

Diplomarbeit gemäß §31 der Rahmenprüfungsordnung für die Fachhochschulen in Bayern (RaPO) vom 07.11.80 mit Ergänzung durch die Prüfungsordnung (PO) der Fachhochschule Augsburg vom 01.10.1981.

Ich versichere, dass ich die Diplomarbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben, sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

Memmingen, den _____

9. Glossar

Abnahme	Prüfung der Übereinstimmung zwischen Spezifikation eines Systems und der letztlichen Umsetzung
Affiliate Shops	Affiliate Shops sind Shops, die andere WWW-Site-Betreiber in Ihre WWW-Präsenz einbinden können und je nach Umsatz eine Provision erhalten
Anchor-Text	ein Anchor-Text beschreibt einen → Hypertextlink kurz und bündig; der Anchor-Text ist optional und muss nicht beim Hypertextlink stehen
Architektur	Art und Weise der Zusammenarbeit / des Zusammenwirken der einzelnen Systemkomponenten
API	Application Programming Interface; Schnittstellensammlung zur Programmierung
Ass	Advanced Small Shop – nach bestimmten Themenvorgaben aufgebauter Online-Shop
Black Box	Bezeichnung für ein System / eine Komponente, die aufgrund fehlendem Wissen über die Internas als "schwarze Schachtel" angesehen wird; Abläufe in dieser Box sind nicht bekannt
Backend, Backend-System	System, dass Arbeiten erledigt, die im Hintergrund anfallen (z.B. Warenwirtschaftsystem)
Backup	Sicherung des Datenbestandes des jeweiligen Computersystems
boole'sche Operatoren	Operatoren zur logischen Verknüpfung (z.B. AND oder OR)
Bottlenecks	Flaschenhälse; Engpässe
breadth-first	eine Strategie, bei der eine zu untersuchende Menge zuerst oberflächlich untersucht wird und anschließend das Wissen über die Menge vertieft wird
Browser	Programm zur Anzeige von HTML-Dateien; meist sendet der Browser Anfragen an den Webserver; dieser sendet anschließend die über die URL referenzierten HTML-Daten
btree	spezielle baumartige Anordnung der Daten; Vorteil ist die automatische Indexierung und die platzsparende Speicherung

	der Daten
Bulkload	Import von Massendaten in eine Datenbank
Businessmodell	Modell der Geschäftsbeziehungen zwischen verschiedenen Subjekten
CGI-Skripte	CGI steht für Common Gateway Interface; CGI-Skripte werden zur Verbindung von Frontend-Systemen mit Backend-Systemen im Zusammenhang mit Internetservern verwendet
Cluster, Computercluster	systematische Zusammenschaltung von Computern im Sinne der Erhöhung der Performanz und Verfügbarkeit
Clusterfähigkeit	Fähigkeit des System in einem → Computercluster zu arbeiten
Connectivity	Kommunikationsverbindung zwischen zwei Systemen
Datenbankserver	Computersystem, auf dem die Datenbank abläuft
Datenmodell	datentechnische Abbildung in Modellform eines realen Systems
Denormalisierung, denormalisiertes Datenmodell	Änderung des → Datenmodells, um von einem logisch richtigen, optimalen Datenmodell zu einem "schnellen" Datenmodell zu gelangen
DRE, Dynamic Reasoning Engine	Herzstück des Autonomy Knowledge Servers; beurteilt Dokumente nach dem von Autonomy definierten Verfahren
DSN, Data Source Name	der DSN wird in der Windows-Umgebung dazu verwendet eine bestimmte Datenquelle mit einem Systemweit eindeutigen Namen anzusprechen; mit dem DSN können auch → ODBC-Treiber verwendet werden
DTD, Data Type Definition	Definition, in der festgelegt wird, welche Elemente der Sprache → XML wo und wie oft vorkommen dürfen; in der DTD werden nur Regeln definiert
E-Business	Allgemeine Bezeichnung für über Datennetze abgewickelten Geschäftsverkehr. Darunter fallen Warenbestellungen über das Internet ebenso wie der Kontakt zwischen einzelnen Firmen.
EDI	Electronic Data Interchange; Standard zum elektronischen

	Austausch von artikelbezogenen Informationen
Evaluation	Untersuchung und Bewertung
Feinkonzept	weitere Konkretisierung des abstrakteren Grobkonzeptes; Schnittstellen werden definiert und Funktionen ausgearbeitet
Fetch	engl. "holen"; hier: der Vorgang, neue Daten in den Index oder die → DRE aufzunehmen
Frame, Frames	HTML-Gestaltungselement; das darstellbare Browserfenster kann in unterschiedliche Abschnitte unterteilt werden – den Frames
Frontend, Frontend- System	System, dass Arbeiten erledigt, die im Vordergrund anfallen (z.B. Webserver)
Grobkonzept	das allgemeine Konzept für das System wird verfeinert und konkretisiert; Funktionalitäten werden festgelegt
Hypertextlinks	Hypertextlinks sind in der HTML-Sprache Markierungen "<A>", die auf ein anderes Dokument oder eine Webressource verweisen
HTTP-Protokoll	die Sprache des World Wide Web; einfaches auf Zeichenketten basierendes Kommunikationsprotokoll; zustandslos (nach jeder Anfrage / Antwort – Sequenz endet die Kommunikation)
HTTP-Server	→ Webserver
Imagemaps	HTML-Gestaltungselement; Imagemaps definieren auf Bildern bestimmte Regionen, die beim Anklicken wie ein HTML-Link reagieren
Implementierung	Umsetzung eines abstrakten Konzeptes in ein konkretes Vorgehensmodell (z.B. die Programmierung eines Feinkonzeptes in einer speziellen Programmiersprache)
Inter-OPS- Kommunikation	Kommunikation zwischen Oracle Parallel Servers
Ist-System	ein bereits eingesetztes System
Java Virtual Machine, JVM	ein definiertes System, das zur Ausführung von Java Bytecode (compilierter Java-Source) verwendet wird; oft in

	→ Browsern integriert
kontextsensitiv	vom näheren Umfeld abhängig; das nähere Umfeld beachten
Lasttest	gesteuerter Vorgang, bei dem mehrere Benutzer auf dem zu testenden System simuliert werden; Ziel ist es zu erfahren, wie sich das getestete System in der Realität verhält
LEX	Programm zur einfachen Erstellung von im Compilerbau benötigten Tools
lexikalischer Analyzer	Programm, das eine Eingabemenge hinsichtlich eines Lexikons untersucht und in eine Ausgabemenge umwandelt
libWWW	Sammlung von nützlichen Programmiersourcen, um die Internetprogrammierung zu vereinfachen und zu vereinheitlichen; wird vom W3C-Konsortium, das für Standardisierungen im Internetumfeld zuständig ist gepflegt
linear skalieren	wird zu einem Computersystem ein zweites System dazugestellt, erwartet man von einem linear skalierenden System, dass doppelt so viel Rechenleistung zur Verfügung steht
Links	→ Hypertextlinks
MMS	Multimedia-Management-System; Komponente des →SAP-Systems, die zur Verwaltung von Multimediadaten dient
ModPERL	PERL Interpreter, der als Modul in den Webserver eingebunden wird
monolithische Applikation	Programm, das nur aus einem Block besteht; ist nicht verteilbar und relativ unflexibel; nicht skalierbar
neuronales Netzwerk	System bestehend aus einzelnen Komponenten, die ähnlich dem menschlichen Gehirn zusammengeschaltet werden können; neuronale Netzwerke sind in der Lage ihr Verhalten selbst zu ändern
normativ	auf Regeln basierendes Vorgehen; durch die starke Strukturierung ist es ein transparentes, nachvollziehbares Verfahren
Normalisierung,	Änderung des Datenmodells, um ein logisch richtiges,

normalisiertes Datenmodell	optimalen Datenmodell zu erhalten; normalisierte Datenmodelle sind meist auf der logischen Ebene korrekt, aber selten performant in der praktischen Anwendung
native Datenbanktreiber	Treiber- bzw. Schnittstellenprogramme, die speziell für die Datenbank eines Herstellers zugeschnitten sind; meist performant und für mehrere Betriebssysteme zu bekommen
Notfallszenario	gesteuerter Ablauf von Aktivitäten beim Eintreten eines Fehlers
ODBC	Open DataBase Connectivity; proprietäres Treiberformat, damit über diese Schnittstellen Datenbanken unterschiedlicher Hersteller angesprochen werden können; meist nicht sonderlich performant und in der Windows-Welt zu finden
Offline-Batchbetrieb	Ablauf eines Programms, der nicht beeinflusst werden kann; es werden nacheinander mehrere Aufträge abgearbeitet
Online-Shop	virtueller Verkaufskatalog im Internet mit Bestellmöglichkeit
Pageimpressions	Anzahl der Seiten, die von der WWW-Adresse abgerufen wurden
Parser	System, das eine Eingabe auf Regelkonformität hin untersucht und anhand der definierten Regeln Umwandlungen der Eingabe vollzieht
PERL	interpretierte Programmiersprache
Pflichtenheft	Beschreibung des Systems auf relativ abstrakter Ebene
Primärschlüssel, Primärschlüsselattribut	ein oder mehrere Datenbankfeld/er, das/die für die gesamte Tabelle eindeutig und charakteristisch ist/sind; anhand des Primärschlüssels kann eine Zeile in der Datenbanktabelle genau identifiziert werden
proprietär	ein selbstentwickeltes System, das sich nicht an Standards hält; meist eine Insellösung; oft sehr schnell und optimiert
Prozess	Programm auf einem Computersystem mit definierten Eigenschaften und Funktionen
Query-Optimizer	Programm, das Anfragen an die Datenbank auf die Möglichkeit einer Optimierung untersucht

redundant, Redundanz	mehrfache Vorhaltung von Systemen oder Daten; Platz oder auch Geld wird bewusst mehrfach geopfert, um die Funktionalität v.a. im Fehlerfall aufrecht zu erhalten
Reengineering, Reengineeringprozeß	von der bereits vorhandenen Implementierung wird versucht die Konzepte zu erschliessen; meist werden dazu Struktogramme gezeichnet
Routingebene	niedere Ebene eines Kommunikationsprotokolls
SAP	bekanntes Warenwirtschaftssystem
Schlagwortsuche	verschiedene Schlagworte werden eindeutig Produkten zugeordnet; ressourcenschonend aber aufwändig in der Vorbereitung
Seitenabrufe	→ Pageimpressions
Server	→ Webserver
Serverausfall	ein Computer, auf dem ein Programm arbeitet bricht auf unkontrolliertem Wege die Bearbeitung des Programmes ab
Session, Sessionhandling	zusammengehörige Seitenabrufe eines bestimmten identifizierbaren Benutzers; das im Internet verwendete Protokoll HTTP ist ein zustandsloses Protokoll und kann sich daher keinen Zustand merken; Sessions
Single Point of Failure	Schwachpunkt in einem System; ein System, bestehend aus mehreren Komponenten, ist nur so gut, wie die schwächste aller Komponenten
Suchmaschine	spezielles Programm, das nur für die Suche auf einem Datenbestand erstellt wurde
Shoppingapplikation	→ Online-Shop
Soll-Modell	Modell eines Systems, das alle definierten Bedingungen erfüllt
Sourcen	Programmdateien, die den Ablauf eines Programmes in einer bestimmten Programmiersprache beschreiben
Spezifikation	genaue Beschreibung der einzelnen Funktionalitäten eines Systems
SQL, SQL-Statement,	Anfrage an eine Datenbank in der Structured Query

SQL-Abfrage	Language; SQL ist eine spezielle Datenbankabfragesprache
Stop-List	hier: beinhaltet gewöhnliche und gebräuchliche Wörter, die keinen Informationsgehalt haben
Storagesubsystem	System, das nur aus Festplatten besteht und dessen Zweck es ist, Daten zu speichern
Stored Procedures	Programmteile, die zur Geschwindigkeitsoptimierung direkt von der Datenbank abgearbeitet werden und auch in der Datenbank gespeichert sind; meist in einer datenbankeigenen Programmiersprache geschrieben
straight ahead	ohne spezielle Überlegungen im Vorfeld dem intuitiv vorgegebenen Weg folgen
Stresstest	→ Lasttest
Systemplattform	→ Online-Shop
TCP/IP	Transfer Control Protocol / Internet Protocol; spezielle Kommunikationsprotokollfamilie, die vorwiegend durch die Verbreitung des Internets bekannt wurde
tempdb	Speicherbereich der Sybasedatenbank, der für temporäre Aufgaben verwendet wird
Templates	Vorlagen, Schablonen; werden mit Daten befüllt; Platzhalter geben an, wo welche Daten eingefüllt werden sollen
Token	Signalmuster; hier ein Wort oder eine Zeichenfolge
Transition	Übergang
URL	Uniform Ressource Locator; eine Zeichenkette, die eindeutig den Speicherort von Daten angibt
Vector Space Modell	bestimmte Art und Weise Informationen zu speichern und zu suchen → 4.3.2
Visits	Anzahl der unterscheidbaren Internetsurfer (z.B. anhand der IP-Adresse)
Vollreplikat	Erstellung einer Kopie der gesamten Datenbank
Volltextsuche	Suche über alle Texte, die in der Datenbank gespeichert sind; ressourcenintensiv und ohne Vorbereitung durchführbar

Warenwirtschaftssystem	System zur Verwaltung von Materialien und Waren; Abwicklung von Rechnungen; Steuerung von Auslieferlogistiken
Webbrowser	→ Browser
Webserver	System, das Anfragen von Webbrowsern bearbeitet; meist steckt hinter einer URL ein Webserver (z.B. www.conrad.de)
White Box	Bezeichnung für ein System / eine Komponente, deren Funktionalität aufgrund der Dokumentation über die Internas; bekannt sind
X-Machine	XML-Engine des Tamino Informationsservers
XML, eXtensible Meta Language	Metasprache, mit deren Hilfe neue Sprachen definiert werden können
X-Node	für die Integration externer Datenbankquellen in den Tamino zuständig
XQL, XML Query Language	Abfragesprache, um Datenbestände, gespeichert in → XML abzufragen; ähnlich → SQL
YACC	Programm zur einfachen Erstellung von im Compilerbau benötigten Tools

10. Literaturverzeichnis

- [AUT00] Autonomy, Unternehmensgeschichte,
<http://www.autonomy.com/company/background.htm>, 2000
- [ADC99a] Autonomy, Produktdokumentation, Knowledge Server User Manual zur
Version 1.9.1, 1999
- [ADC99b] Autonomy, Produktdokumentation, ODBCFetch User Manual zur
Version 1.9.1, 1999
- [ATW00a] Autonomy, Technology White Paper,
<http://www.autonomy.com/tech/wp.html>, 2000
- [ATW00b] Autonomy, Technology White Paper,
<http://www.autonomy.com/tech/wp.html>, Seite 6-9, 2000
- [ATW00c] Autonomy, Informationen über die Autonomy Produkte auf der Website,
<http://www.autonomy.com>, 2000
- [BME00] Bundesverband Materialwirtschaft, Einkauf und Logistik – BME,
Spezifikationen und Informationen über BMECat,
<http://www.bmecat.de/bmecat/>, 2000
- [CAW99a] Cawsey, Alison: Natural Language Processing Course,
<http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html>, dort:
Boolean Retrieval, Heriot-Watt-Universität Edinburgh, 1999
- [CAW99b] Cawsey, Alison: Natural Language Processing Course,
<http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html>, dort:
Vector Space Retrieval, Heriot-Watt-Universität Edinburgh, 1999
- [CAW99c] Cawsey, Alison: Natural Language Processing Course,
<http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html>, dort:
Probabalistic Retrieval, Heriot-Watt-Universität Edinburgh, 1999
- [CAW99d] Cawsey, Alison: Natural Language Processing Course,
<http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html>, dort:
Relevance Feedback, Heriot-Watt-Universität Edinburgh, 1999
- [CAW99e] Cawsey, Alison: Natural Language Processing Course,
<http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html>, dort:
Ranked Retrieval, Heriot-Watt-Universität Edinburgh, 1999
- [CCM00] Conrad.com AG, Unternehmensprofil, Hirschau, 2000

- [CIF00] CoMedia – Hennig, M.: Internet Schnittstellen – Schnittstellend es Conrad Internetauftritts zu den Core-Systemen der Conrad-Holding, Hirschau, 2000
- [CIN97] Conrad Electronic GmbH, Internetauftritt, <http://www.conrad.de>, dort: "Kontakte", "Conrad Historie", 1997
- [CIS] Centre for Interactive Systems Research: The Probabilistic Retrieval Model, <http://web.cs.city.ac.uk/research/cisr/okapi/prm.html>
- [CNM99] Conrad Neue Medien GmbH, Unternehmensprofil, Hirschau, 1999
- [CON98] Conrad Electronic GmbH, Broschüre: 75 Jahre Conrad Electronic, Hirschau, 1998
- [CT99a] Jörg, M.: Doppelgänger gesucht – Ein Programm für kontextsensitive phonetische Textumwandlung, in Magazin c't Ausgabe 25/99, S.252, 1999
- [CT99b] Jörg, M.: Sourcen zum Artikel Doppelgänger gesucht unter <ftp://ftp.heise.de/pub/ct/listings/phonet.zip>
- [DRA96a] Drabenstott, K.M.: Enhancing a new design for subject access to online catalogs, Library Hi Tech 14, Seite 87-109, 1996
- [DRA96b] Drabenstott, K.M.; Weller, M.S.: Failure analysis of subject searches in a test of a new design for subject access to online catalogs, J. Amer. Soc. Inform. Sci. 47, Seite 519-537, 1996
- [DRA96c] Drabenstott, K.M.; Weller, M.S.: The exact-display approach for online catalog subject searching, Inform. Proc. Manag. 32, Seite 719-745, 1996
- [DUM44] Dumas, Alexandre: Les Trois Mousquetaires, 1844
- [EFE99] EFEU-Elektronik GmbH, Joachim Katz, Dokument: Artikel finden im Internet - Leitfaden für ein kundenfreundliches Suchsystem im Internet-Auftritt der Firmen Conrad Electronic und Völkner, Berlin, 1999
- [GUA99] Guach, S.: Search improvement via automatic query reformulation, ACM Trans. Inform. Syst. 9, 1991
- [HIL95] Hildreth, Charles R.: Online Catalog Design Models: Are We Moving in the Right Direction ?, <http://www.ou.edu/faculty/H/Charles.R.Hildreth/clrthree.html>, Oklahoma, 1995

- [KNO94a] Knorz, Gerhard: Automatische Indexierung, <http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm>, dort: 3.1. Boole'sches Retrieval versus Rankingverfahren, Universität Potsdam, 1994
- [KNO94b] Knorz, Gerhard: Automatische Indexierung, <http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm>, dort: 3.3.1. Das Vektormodell, Universität Potsdam, 1994
- [KNO94c] Knorz, Gerhard: Automatische Indexierung, <http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm>, dort: 3.4. Ein einfaches Verfahren der Termgewichtung, Universität Potsdam, 1994
- [KNO94d] Knorz, Gerhard: Automatische Indexierung, <http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm>, dort: 3.3.2. Probabilistische Verfahren, Universität Potsdam, 1994
- [KNU97] Knuth, D.E.: The art of Computer Programming – Sorting and Searching 2nd ed., ISBN 0-201-89685-0, Seite 394-395, 1997
- [LEE95a] Lee, J.H.: Analyzing the Effectiveness of Extended Boolean Models in Information Retrieval, Seite 3f, <http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR95-1501>, Cornell Universität, März 1995
- [LEE95b] Lee, J.H.: Analyzing the Effectiveness of Extended Boolean Models in Information Retrieval, Seite 4ff, <http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR95-1501>, Cornell Universität, März 1995
- [OAR99] Oard, Douglas W.: The Boolean Retrieval Model, <http://raven.umd.edu/courses/708a/fall99/notes/708f992/ppframe.htm>, Maryland, 1999
- [ODC99a] Oracle Corporation, Oracle8 Documentation – Oracle 8i Concepts Release 8.1.5, dort: 26 Parallel Execution, Redwood City, 1999
- [ODC99b] Oracle Corporation, Oracle8 Documentation – Oracle 8i Parallel Server Concepts and Administration Release 8.1.5, Redwood City, 1999

- [PFP00] PixelFactory GmbH, Rene Herzer, Dokument: CONRAD ONLINE AG Pflichtenheft Version 1.0 vom 28.02.2000, Offenbach, 2000
- [PIN94] Pinkerton, B.: Finding what people want – Experiences with the WebCrawler, <http://www.thinkpink.com/bp/WWW94.html>, 1994
- [PIP99] PixelPark AG, Susan Plaumann, Dokument: Projekt-Architektur, Berlin, 2000
- [POO99a] Poo, D.C.C.; Toh, T.K.; Khoo, C.S.G.: Design and implementation of the E-referencer, Data & Knowledge Engineering 32, Seite 199-201, 2000
- [POO99b] Poo, D.C.C.; Toh, T.K.; Khoo, C.S.G.: Design and implementation of the E-referencer, Data & Knowledge Engineering 32, Seite 202, 2000
- [PSW99] PixelPark AG, Bernd Schmeil, Dokument: CONRAD Schlagwortsuche, Berlin, 1999
- [PVT99] PixelPark AG, Bernd Schmeil, Dokument: CONRAD Volltextsuche, Berlin, 1999
- [RAN96a] Rankins, R.; Garbus, J.R.; Solomon, D.; McEwan, B.W.: Sybase SQLServer unleashed, ISBN 0-672-30909-2, SAMS Publishing Indianapolis, 1996
- [RAN96b] Rankins, R.; Garbus, J.R.; Solomon, D.; McEwan, B.W.: Sybase SQLServer unleashed, ISBN 0-672-30909-2, Seite 546ff, SAMS Publishing Indianapolis, 1996
- [ROB76] Robertson, S.E.; Sparck, J.K.: Pervance weighting of search terms, Journal of the American Society for Information Science, 27, Seite 129-146, 1976
- [ROU00] Rougier, J.: Introduction to Dynamic Linear Modelling, Seite 3ff, 2000
- [SAG99a] Software AG – Oppel, K.: Tamino White Paper, Oktober 1999
- [SAL82a] Salton, G.; Fox, E.A.; Wu, H.: Extended Boolean Information Retrieval, Seite 2ff,
<http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR82-511>, Cornell Universität, August 1982
- [SAL82b] Salton, G.; Fox, E.A.; Wu, H.: Extended Boolean Information Retrieval, <http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR82-511>, Cornell Universität, August 1982

- [SDC99] Software AG: Dokumentation zum Tamino Informationsserver, 1999
- [SUF99a] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, <http://www.suchfibel.de/5technik/systeme.htm>, Leipzig, 1999
- [SUF99b] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, <http://www.suchfibel.de/5technik/sammeln.htm>, Leipzig, 1999
- [SUF99c] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, <http://www.suchfibel.de/5technik/struktur.htm>, Leipzig, 1999
- [SUN97] SUN Microsystems: The Sun Enterprise Cluster Architecture – Technical White Paper, <http://www.sun.com/clusters/wp.html>, 1997
- [TON87] Tong, R.M.; Applebaum, L.A.; Askmann, V.N.; Cunningham, J.F.: Conceptual information retrieval using RUBIC in Yu, C.T.; Van Rijsbergen, C.J.: Proceedings of the Tenth Annual International ACM SIGIR Conference on Research and Development in Formation Retrieval, Seite 247-253, 1987
- [VIC87] Vickery, A.; Brooks, H.M.: PLEXU – The expert system for referral, Inform. Proc. Manag. 23, Seite 99-117, 1987
- [WIR00a] Silberman, Steve: The Quest for Meaning, Magazin Wired, <http://www.wired.com/wired/archive/8.02/autonomy.html>, Februar 2000

11. Verzeichnisse

11.1. Abbildungsverzeichnis

Abbildung 2-1 Architektur der Conrad Online Plattform	8
Abbildung 2-2 implementierte Suche in der Conrad Online Plattform	16
Abbildung 2-3 Prinzip des Verschlagwortungskonzeptes	19
Abbildung 2-4 temporäre Tabellen, die zur Resultatübergabe der Produktgruppensuche verwendet werden	20
Abbildung 2-5 temporäre Tabellen, die zur Resultatübergabe der Produktsuche verwendet werden	21
Abbildung 2-6 Datenmodell für die Volltextsuche	22
Abbildung 4-1 Architektur* des WebCrawlers	57
Abbildung 5-1 Architektur* des Autonomy Knowledge Servers / Interaktion der einzelnen Komponenten	65
Abbildung 5-2 Installation des KnowledgeServers von Autonomy - Auswahl der Ports	68
Abbildung 5-3 Administrationstool zum Zugriff auf die DRE - leere Datenbank	68
Abbildung 5-4 Einrichten eines systemweiten ODBC-Synonyms für die Artikeldatenbank	69
Abbildung 5-5 ODBCFetch läuft in einem DOS-Fenster	73
Abbildung 5-6 Administrationstool für die DRE - 35965 indexierte Dokumente und 50928 indexierte Begriffe	75
Abbildung 5-7 WWW-Front End des KnowledgeServers	76
Abbildung 5-8 Produkttreffer nach Eingabe des Suchwortes "Nokia Telefon Handy"	78
Abbildung 5-9 Produktbeschreibung für Nokia 6110 und ähnliche Produkte im unteren Fensterteil	78
Abbildung 5-10 Antwortzeiten während des Lasttests	79
Abbildung 5-11 Architektur* des Informationsservers Tamino	82
Abbildung 5-12 Der Tamino Manager zur Verwaltung der Datenbank	83
Abbildung 5-13 Der Schema Manager	84
Abbildung 5-14 Das Interactive Interface	84
Abbildung 5-15 Das Interactive Interface mit einer durchgeführten XQL-Anfrage	88
Abbildung 5-16 grafische Abbildung der DTD für die Artikeldaten der Conrad.com AG	89
Abbildung 6-1 Übergang vom normalisierten (grauen) Datenmodell hin zum optimierten Datenmodell (farblos)	106
Abbildung 6-2 Ablauf im Prototypen für die Verschlagwortung	107
Abbildung 6-3 Erste Seite im Suchprototypen für die Verschlagwortung - Eingabe des Suchwortes	107
Abbildung 6-4 Zweite Seite im Suchprototypen für die Verschlagwortung – Anzeige der Produktgruppen und Selektion einer dieser Gruppen	109
Abbildung 6-5 6-6 Dritte Seite im Suchprototypen für die Verschlagwortung – Anzeige der Produkte	110
Abbildung 7-1 Vier Tablespace gestript auf vier unterschiedliche Geräte	112
Abbildung 7-2 links die serielle Abarbeitung einer SQL-Anfrage, rechts die parallele Abarbeitung	113
Abbildung 7-3 Abarbeitung einer beispielhaften SQL-Anweisung durch 8 parallele Prozesse	114
Abbildung 7-4 Ein Cluster*	115
Abbildung 7-5 Ablaufplan des Prototypen für die Verschlagwortung - mit phonetischen Erweiterungen	121
Abbildung 7-6 Ablaufplan des Prototypen für die Verschlagwortung mit phonetischen Ergänzungen	122

11.2. Tabellenverzeichnis

Tabelle 2-1 Übersicht über die Shops der Conrad.com AG	6
Tabelle 2-2 Hardwareaufstellung der Conrad Online Plattform	9
Tabelle 2-3 Datenbankinstallation auf der Maschine demdwu18 - Größenangaben in GigaBytes	10
Tabelle 2-4 performanzsteigernde Maßnahmen für die Conrad Online Plattform	14
Tabelle 2-5 Datenmodelle für Schlagwortsuche links vom verschlagwortenden Dienstleister, rechts vom shopbetreibenden Dienstleister	20
Tabelle 2-6 Rückgabewerte der Produktgruppensuche	21
Tabelle 2-7 Rückgabewerte der Produktsuche	21

Tabelle 4-1 Vor- und Nachteile der Suche im Dateisystem.....	38
Tabelle 4-2 Vor- und Nachteile der Suche in einer unstrukturierten Indexdatei	38
Tabelle 4-3 Vor- und Nachteile der Suche in einer strukturierten Indexdatenbank	38
Tabelle 4-4 Arten von Suchmaschinen und deren Aufgaben.....	40
Tabelle 4-5 Beispiel einer Indextabelle	42
Tabelle 4-6 Beispielabfragen auf der obigen Indextabelle.....	42
Tabelle 4-7 beispielhafte Indexierung von Dokumenten	45
Tabelle 4-8 beispielhafte Indexierung für drei Beispieldokumente.....	46
Tabelle 4-9 beispielhafte Gewichtung der drei Beispieldokumente	46
Tabelle 4-10 beispielhafte Anwendung des Verfahrens "Grad der Wortübereinstimmung"	48
Tabelle 4-11 verschiedene Operatoren des Extended Boole'schen Modells.....	50
Tabelle 5-1 Beschreibung der einzelnen Artikelattribute, die für die Suche von Bedeutung sind	67
Tabelle 5-2 Aufstellung der Antwortzeiten beim Lasttest	79
Tabelle 5-3 Erläuterung der Aktionen beim Lasttest.....	80
Tabelle 5-4 Operatoren in XQL-Anfragen.....	88
Tabelle 6-1 Entscheidungsmatrix.....	103
Tabelle 6-2 Erläuterung des Sourcecodes zu firststepsearch.pl.....	108
Tabelle 6-3 Erläuterung des Codes zu scndstepsearch.pl.....	109
Tabelle 7-1 Konstruktionsregeln des Soundex Codes	118
Tabelle 7-2 Ausschnitt aus der Artikeldatenbank mit phonetischen Attributen	120

11.3. Ausschnittverzeichnis

Ausschnitt 2-1 SQL-Anweisungen aus dem PERL Skript AssPowerSuche.pl.....	26
Ausschnitt 2-2 SQL-Anweisungen aus der Stored Procedure "sp_schlagworte"	26
Ausschnitt 5-1 Konfigurationsdatei für den ODBCFetch – "ODBCFetch.cfg"	70
Ausschnitt 5-2 Konfigurationsdatei "michaelsfetch.cfg"	71
Ausschnitt 5-3 Das Template "complete.tmpl" für den Datenexport	72
Ausschnitt 5-4 eine generierte HTML-Seite aus der ODBC-Datenquelle	73
Ausschnitt 5-5 Konfigurationsdatei des Autonomy AutoIndexers	75
Ausschnitt 5-6 Konfiguration des ODBC-CGI-Interfaces.....	76
Ausschnitt 5-7 Template für die Ausgabe der ODBC-Datenquelle über das CGI-Interface*	77
Ausschnitt 5-8 DTD zum Adressrecord	85
Ausschnitt 5-9 Beispieldatensatz zum Adressrecord in XML - konform zur obigen DTD	86
Ausschnitt 5-10 DTD für die Artikeldaten der Conrad.com AG	91
Ausschnitt 5-11 ein Beispielartikel in XML - konform zur obigen DTD	92
Ausschnitt 6-1 SQL-Statement für das normalisierte Datenmodell - Suchabfrage	105
Ausschnitt 6-2 SQL-Statement für das denormalisierte* Datenmodell - Suchabfrage	105
Ausschnitt 6-3 Codefragmente aus firststepsearch.pl.....	108
Ausschnitt 6-4 Codefragmente aus scndstepsearch.pl.....	109