Aufgabenstellung

Knowledge Base gestützte Verwaltung und Recherche von Textdokumenten mit Oracle Text

Ontologische Durchdringbarkeit eines Wissensgebietes, Erstellung und Anreicherung einer Knowledge Base, Qualifizierung der Textrecherche durch fortgeschrittene Retrieval- und Präsentationsoptionen.

Die Qualität des Wissensmanagements (Knowledge Management, KM) zeigt sich für den recherchierenden Benutzer hauptsächlich darin, wie gut ohne genaue Kenntnis eines Wissensgebietes eine vage Anfrage mit Treffern beantwortet wird. Kriterien für eine Treffermenge sind u.a. inhaltliche Erwartungskonformität, rasche Entscheidbarkeit der Relevanz, Redundanzfreiheit, gute Erfassbarkeit der inhaltlichen Beziehungen, intuitive Navigation.

Inhaltliche Aspekte der Arbeit:

- Kurzer Abriss über Entwicklung, Stand (KM-Konzepte), Perspektiven der KM-Anwendung.
- Entwicklung einer Demonstrations-Anwendung auf der Basis von Oracle Text für das Content Management und das Knowledge Retrieval unter besonderer Nutzung der jüngsten Oracle9*i* Features.
- Nutzungsmöglichkeiten für weitere, verfügbare Techniken zur Aufbereitung, Recherche und Präsentation (z.B. AIDOS).
- Bewertung mittels eines Kataloges von Qualitätskriterien.
- Strategien für die Integration in bestehende Anwendungen.

Ziele der Arbeit sollen sein:

- Fundierung von Stand und Perspektiven des Gebietes.
- Einordnung, Möglichkeiten und Grenzen von Oracle Text am praktischen Beispiel.
- Kombinierbarkeit mit weiteren Produkten wie der KAI| BOX von AIDOS Software AG.
- Optionen für die Fortentwicklung der Recherche im Informationssystem Gesundheits-Berichterstattung (IS-GBE, http://www.gbe-bund.de/).

Knowledge Base gestützte Verwaltung und Recherche von Textdokumenten mit Oracle Text

Diplomarbeit

zur Erlangung des Grades eines Diplominformatikers (FH)
des Studienganges allgemeine Informatik
der Hochschule Zittau/Görlitz (FH) – University Of Applied Sciences

vorgelegt von

Thiemo Mättig
geboren am 9. November 1976 in Bautzen

Referent: Prof. Dr. phil. Manfred Thiel

Betrieblicher Betreuer: Prof. Dr.-Ing. habil. Dieter Jungmann
(Robotron Datenbank-Software GmbH)

Dresden, 1. Dezember 2003

Kurzreferat

„Text is everywhere.“ – Oracle Text-Dokumentation

Die Robotron Datenbank-Software GmbH Dresden treibt seit mehreren Jahren die Entwicklung eines Internet-Informationssystems für das Statistische Bundesamt voran. Um dem Kunden trotz des fortgeschrittenen Standes des Projektes weitere Perspektiven bieten zu können, sollen fortgeschrittene Techniken des Wissensmanagements zum Einsatz kommen. Ziel dieser Arbeit war, den Entwicklungsstand auf diesem Gebietes im Allgemeinen und im Bezug auf das Datenbanksystem Oracle im Speziellen zu untersuchen, zu dokumentieren und auf die Eignung für das Informationssystem hin zu überprüfen. Anhand einer prototypischen Implementation sollten Techniken und Technologien exemplarisch dargestellt und auf ihren Einsatz hin geprüft werden.

Inhaltsverzeichnis

Abkürzungs- und Begriffsverzeichnis

Begriff	Erklärung
ad hoc	Lateinisch „(eigens) dazu“, „zu dem Zweck“
Adjacency	Relative Wortpositionen/-abstand
ANSI	American National Standard Institute
CHAR	Character, ein einzelnes alphanumerisches Zeichen
CMS	Content Management System
CTX	ConText, Produktname bzw. Indextyp bei Oracle
Data Dictionary	Menge von Pseudo-Tabellen, die Informationen über die tatsächlichen Tabellen bereitstellen
DB	Datenbank
DB2	Datenbank-Management-System von IBM
DBA	Datenbank-Administrator
DBMS	Datenbank-Management-/Verwaltungssystem, auch DMS
DBPL	Data Base Programming Language (Sprachkonstrukte zur Erstellung von Anwendungen)
DDL	Data Definition Language (Sprachkonstrukte zur Erzeugung von Schemata, d.h. `CREATE`, `DROP`, `GRANT`, `REVOKE`)
DIN	Deutsche Industrie-Norm
DML	Data Manipulation Language (Sprachkonstrukte zur Abfrage und Manipulation von Daten, d.h. `DELETE`, `INSERT`, `SELECT`, `UPDATE`)
ERM	Entity Relationship-Modell
iAS	Internet Application Server
IQL	Interactive Query Language (Sprachkonstrukte zur Abfrage und Manipulation von Daten)
IR	Information Retrieval (etwa „Wiedergewinnung von Informationen“)
IRS	Information Retrieval-System

ISO	International Standardization Organisation
ISR	Information Storage and Retrieval
OLAP	Online Analythical Processing
OLTP	Online Transaction Processing
OOP	Objektorientierte Programmierung
OTN	Oracle Technology Network
QBE	Query by Example (von SQL verschiedene Abfragesprache anhand von Beispieleinträgen in Tabellengerüsten)
QUEL	Query Language (Ingres, von SQL verschiedene, unterlegene Abfragesprache)
PL/SQL	Procedural Language Extensions to SQL, prozedurale Spracherweiterung für SQL
RAW	Rohdaten, binäre Ursprungsdaten (wörtlich: roh)
RDB	Relationale Datenbank
RDBMS	Relationales Datenbank-Management-System
SQL	Structured Query Language (auch: Standard Query Language)
SQL*Plus	Interaktives Abfragewerkzeug für Oracle
SSL	Storage Structure Language (Sprachkonstrukte zur Dateiorganisation)
Trigger	Programme, die beim Eintritt bestimmter Ereignisse („Events“) angestoßen werden
URI	Uniform Ressource Identifier
VDL	View Definition Language (Sprachkonstrukte zum Erzeugen von Sichten)
View	Sicht auf eine oder mehrere Tabellen; wie eine Tabelle zu handhaben, ohne dass diese tatsächlich existiert
WDBMS	Web-Datenbank-Management-System
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Charakteristische Fallbeispiele

Such- oder besser gesagt Indizierungs-Maschinen stehen vor einem grundlegenden Problem. Die den Textdokumenten maschinell entnehmbaren Informationen sind begrenzt, da diese in der Regel leider völlig unstrukturierten sind. Der sowohl technische als auch personelle Aufwand für die Steigerung der Qualität einer Textrecherche nimmt ab einer gewissen Ausbaustufe überproportional zu.

Grundlegend lässt sich die Indizierung von Einzeldokumenten, die Suche darin, und darauf aufbauend die Handhabung von Dokumenten-Sammlungen unterscheiden. Wie mit letzteren umzugehen ist, hängt wiederum sehr stark vom Umfang der Sammlung, vom Themenspektrum und der Ziel- bzw. Anwendergruppe ab. Es besteht ein deutlicher Unterschied zwischen einer allgemeinen Internetsuche und einem klar definiertem Dokumentenraum, der beispielsweise ausschließlich Texte umfasst, die innerhalb eines Unternehmens wie der Robotron Datenbank-Software GmbH (nachfolgend Robotron genannt) entstehen. In einem solchen Kontext unterliegen die Erfassung von Dokumenten und die Recherche darin klaren Regeln. Zuständigkeiten und Rechte sind überschaubar oder zumindest streng reguliert.

Ganz ähnlich wie die in der Theorie der Datenübertragung und Kompression bekannte Shannon-Grenze (Claude Shannon, 1916-2001) kann es auch in der Volltextindizierung und Suche niemals einen Algorithmus geben, der für jede beliebige Art zufälliger Daten gleich gut funktioniert. [Held01] Es ist in besonderem Maße notwendig und wichtig, zusätzliches Wissen über die zu behandelnden Texte einzubeziehen.

Als einfachstes Beispiel hierfür ist die natürliche Sprache eines Textes zu nennen. Dass die Suche nach dem Begriff „Wetter“ in einem englischen Text höchstwahrscheinlich sinnlos ist, kann die Maschine nicht ohne weiteres wissen. Sie benötigt zusätzliches Wissen: die Sprache des Suchbegriffes, des Textes, Regeln für die Verarbeitung von Texten dieser Sprache sowie einen Übersetzungskatalog. Erst unter diesen Voraussetzungen kann die korrekte Textstelle „weather“ gefunden werden.

Die selben Mechanismen treffen auf Sammlungen von Fachtexten zu, in denen zum Beispiel nach „Herzinfarkt“ gesucht wird (ein Beispiel aus dem Datenbestand des Statistischen Bundesamtes, der dieser Arbeit zugrunde liegt). Auf diesen Begriff hin ist zunächst nichts zu finden, da die durchsuchten Texte in ihrer Fachsprache in aller Regel von einem „Akuten Myokardinfarkt“ sprechen, selten jedoch vom geläufigen „Herzinfarkt“. Vom Anwender kann weder erwartet werden, dass er das weiß, noch sollte er damit konfrontiert werden, seine Suchanfrage mit allen denkbaren sinnverwandten Begriffen wiederholen zu müssen. Eine ideale Aufgabe für einen Synonymkatalog oder Thesaurus.

Google/Google AdSense

Trotz aller Möglichkeiten kann die Technik im Bezug auf Suchmechanismen natürlich niemals perfekt sein, selbst unter der fernen Annahme, sie würde die Worte „verstehen“. Als bestes Beispiel sei die allgemein bekannte Suchmaschine „Google“ genannt. Jahre des Erfolges und zurzeit über tausend Entwickler und Mitarbeiter konnten nichts daran ändern, dass die Software bei Suchbegriffen wie „`Kohl`“ oder „`Automatik`“ nach wie vor reichlich hilflos dasteht. Ist ein Politiker oder ein Gemüse gemeint? Ist eine großtechnische Anlage, eine Pkw-Ausstattung oder ein Brotbackautomat gemeint?

Narrensicher ist die Technik nie. So berichtete die New York Times über ein peinliches Missgeschick im Zusammenhang mit der kontext-sensitiven Werbetechnologie von Google: „The online edition of The New York Post […] ran an article last month about a murder in which the victim’s body parts were packed in a suitcase [Koffer], and Google served up an ad for a luggage dealer [Kofferhändler].“[1] Googles AdSense konnte nicht wissen, dass diese spezielle Werbeeinblendung in diesem Zusammenhang unpassend war.

Die gezeigten Beispiele wirken simpel, verdeutlichen jedoch das Problem, das mit jeder noch so genau formulierten Suchanfrage bzw. jedem noch so genau

---

[1] Quelle: The New York Times. <www.nytimes.com/2003/08/04/technology/04ECOM.html?ex=1375416000&en=2ac396ed576ca1b3&ei=5007&partner=USERLAND>

formuliertem Text besteht: Eigentlich dürfte nicht das *Wort* betrachtet werden, sondern der *Sinn*. Was der Anwender bei der Eingabe seines Textes tatsächlich *meinte*, weiß letztendlich nur er allein. Andere Menschen verstehen beim Lesen in der Regel, was gemeint ist, weil ihr gesamtes Wissen in das Verständnis eines Textes mit einfließt.

Dieses grundlegende Dilemma ist einer der Gründe dafür, dass es auf dem Gebiet des Information Retrieval nie wirklich bemerkenswerte Entwicklungsschritte gab und sich bisher keine allgemeine, standardisierte Lehrmeinung durchsetzen konnte, wie es sie zum Beispiel für das Fachgebiet der Relationalen Datenbanken gibt.

1.2 Vorstellung des Referenzprojekts „IS-GBE“

Das Statistische Bundesamt (StBA) Deutschlands trieb in Zusammenarbeit mit dem Bundesministerium für Bildung und Forschung sowie dem Bundesministerium für Gesundheit ein Forschungsprojekt zur Gesundheits-Berichterstattung (GBE) voran. Die Zielsetzung des Projekts war, die lückenhafte Datenlage zum Gesundheitswesen in Deutschland zu verbessern und gleichzeitig schrittweise eine Dateninfrastruktur zu schaffen, die neben Wissenschaft und Forschung auch der interessierten Öffentlichkeit als Informationsgrundlage dienen kann. Typische Fragen, die so beantwortet werden sollen, sind: Wie lange verweilt ein Patient durchschnittlich im Krankenhaus? Oder: Wie hoch sind die Gesundheitsausgaben in Deutschland? Anfang 1999 hat das Projekt die Forschungsphase verlassen und wird als gemeinsame Aufgabe des Robert Koch-Instituts und des Statistischen Bundesamts fortgeführt.

Das Projekt sah vor, neben allen gängigen auch die „neuen“ Medien einzubeziehen. Ergebnis letzterer Bemühungen ist das seit etwa 1996 geplante und schrittweise weiterentwickelte Informationssystem Gesundheits-Berichterstattung, kurz IS-GBE, der Robotron Datenbank-Software GmbH. Unter der Adresse http://www.gbe-bund.de/ ist dieses System jedem Interessierten zugänglich.

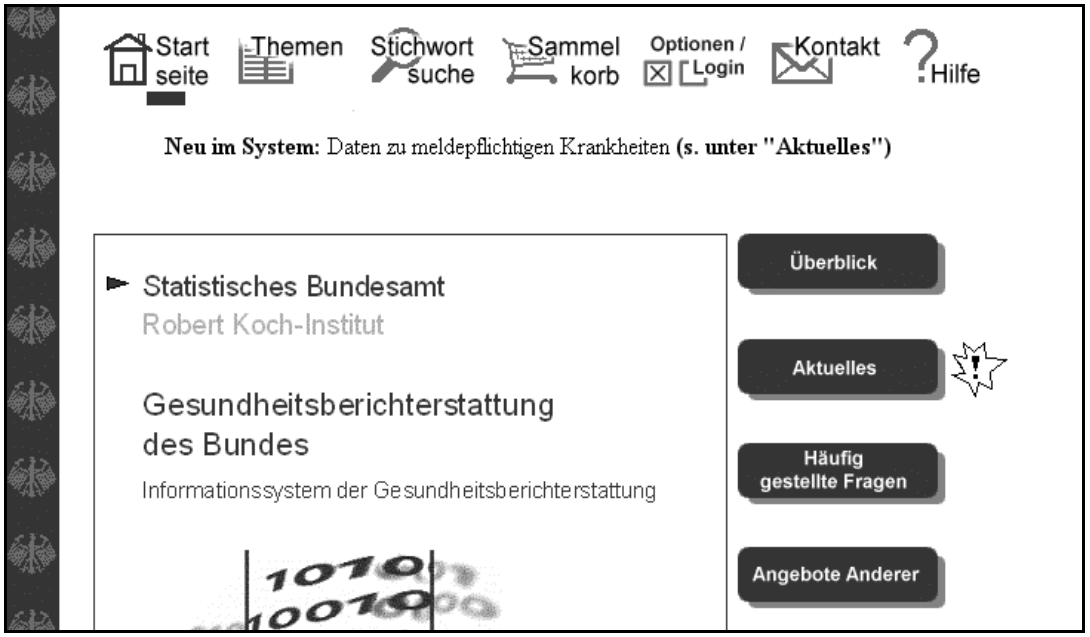

Abbildung 1: Startseite http://www.gbe-bund.de/ des IS-GBE

2 Stand der Entwicklung

2.1 Information Retrieval

„Information Retrieval (IR) im weitesten Sinne des Wortes“ wird laut [Mresse84] kurz und prägnant als „jede Art der Wiedergewinnung gespeicherter Daten“ beschrieben. Diese Definition reißt das Thema der vorliegenden Arbeit kaum an, bietet jedoch einen allgemeinverständlichen Überblick über die praxisnahe Zielsetzung. Die sehr einfache Aussage macht auch deutlich, dass es sich um äußerst „schwammige“ Begriffe in einem nur wenig standardisierten Zweig der Informatik handelt. Sind „Informationen“ und „Daten“ gleichzusetzen? Im Allgemeinen ist das durchaus üblich. Informationen wie beispielsweise „die Veranstaltung beginnt um 14 Uhr“ und Daten wie „14 Uhr“ sind für den Menschen aufgrund seines Erfahrungsschatzes in der Regel gleichwertig – der „Maschine“ jedoch könnte nichts ferner sein als das Verständnis dieser für uns selbstverständlichen Informationsfragmente. Für derartige Entscheidungen müsste dem Rechner ein Modell des Wissens des Benutzers vorliegen, mit dessen Hilfe sich der Informationsgehalt von Daten im Bezug auf den Nutzer und seine Bedürfnisse ermitteln ließe. Dass so etwas höchstwahrscheinlich nie möglich sein wird, beweisen die noch immer eher erfolglosen Bemühungen auf dem Gebiet künstlicher Intelligenzen.

Der Begriff „Wiedergewinnung“ beschreibt zum einen das Suchen und Finden bestimmter Informationen in einem Datenbestand, der im Allgemeinen weitaus größer ist, als dass er von einigen wenigen Personen vollständig überschaubar wäre. Die dabei anzuwendenden Qualitätskriterien spielen eine entscheidende Rolle, da im Idealfall weder zu wenige noch zu viele Fundstellen ausgemacht werden sollten. „Wiedergewinnung“ kann jedoch auch – und tut es in der Regel – das Zusammenstellen und Präsentieren von Daten in einem neuen, erst im Kontext der Recherche gegebenen Zusammenhang sein.

Die Ursprünge dieses vergleichsweise neuen wissenschaftlichen Zweiges reichen bis in die 30er Jahre zurück, als mit dem zunehmenden Einsatz von Mikrofilmen die ersten massenhaften Datensammlungen entstanden. Das Problem der Recherche in diesen Beständen wurde neben den klassischen Methoden (Randlochkarten) zum Teil

schon durch elektrische Maschinen unterstützt, die auf den von Telefonen bekannten Technologien beruhten. Diese numerischen analytischen Maschinen sind das, was wir heute als Computer kennen. Auch wenn die Bezeichnung „Information Retrieval“ wohl um 1950 zum ersten Mal auftauchte, waren es doch Robert Mayo Hayes and Joseph Becker, die den Begriff 1963 in einer ersten großen Publikation prägten: „Information storage and retrieval: tools, elements, theories“.

2.2 Dokumenten- und Wissens-Management

Der Begriff „Content Management“ wird heutzutage leider recht inflationär gehandhabt, da er als klassifizierende Produktbezeichnung für beinahe jede Art von Software verwendet wird, die Informationen oder Dokumente speichert, verwaltet und wieder ausgibt. Dabei ist vieles, was diesen Namen trägt, kaum mehr als ein Dateiverwaltungs- oder im besten Falle Dokumenten-Management-System. Um eine begründete Abgrenzung vornehmen zu können, ist zunächst die Definition einiger Grundbegriffe nötig.

Daten, Informationen und Wissen

„Daten“, „Informationen“ und „Wissen“ werden in der Literatur sehr unterschiedlich beschrieben, so dass sich kaum repräsentative Zitate wiedergeben lassen. Im Allgemeinen wird davon ausgegangen, dass Daten die kleinste zweckdienlich verarbeitbare Einheit der Informatik sind. Ein Datum (Einzahl) ist beispielsweise die Zahl „50,0“ oder das Wort „Aktie“. Daten setzen sich dabei aus Signalen oder auch – für diese Arbeit völlig adäquat – Zeichen zusammen.

Informationen sind zu sinnvollen Strukturen zusammengefasste Daten. Dies geschieht durch die Anreicherung mit „Zweckbezug und Bedeutungsinhalt“ und damit Erhöhung der Relevanz. Daten werden mit anderen Worten dann zu Informationen, wenn die Ungewissheit (Entropie) verringert wird, die Daten also einen Informationszugewinn beim Empfänger hervorrufen. Ein Beispiel wäre, dass der „Wert der Aktie im Januar 2002: 50,0 Euro“ betrug.

Wissen entsteht schließlich durch Regeln und Vernetzung und damit durch Erhöhung der Nachhaltigkeit aus Informationen. Im Beispiel ließen sich aus der Vernetzung mehrerer Informationen Rückschlüsse auf Konjunktur ziehen und Prognosen entwickeln. [Schwickert01] In [Fuhr00] wird treffend geschrieben: „Information ist flüchtig, Wissen permanent“ oder im Umkehrschluss: „Information ist Wissen in Aktion“.

Dokumente setzen sich den eingeführten Grundbegriffen folgend aus einer großen, zusammengehörigen Menge von Daten zusammen, wobei die Reihenfolge, Beziehungen und Strukturierung derselben wiederum Informationen abbilden kann. Content, das heißt Inhalt, sind die Teile eines Dokumentes, die nicht nur rein optischen Charakter haben sondern Informationen oder im besten Fall sogar Wissen in sich tragen. Wissen entsteht wie schon beschrieben durch die gerichtete Verknüpfung von Informationen, wobei neue, vorher nicht in dieser Form existierende Informationen entstehen.

Zurück auf die Ebene der Verwaltungssysteme übertragen heißt das, dass ein Dokumenten-Management-System lediglich über Mechanismen verfügt, um die ihm übergebenen Dokumente als Ganzes zu verwalten. Es handhabt sie als große Datenmenge, ohne die inneren Strukturen der Dokumente zu kennen oder anzutasten. Das System ist nicht in der Lage, sie anders zu präsentieren, als sie ihm vorgelegt wurden. Ein Dokumentenmanagement kann Beziehungen zwischen den Dokumenten als Ganzes, aber nur sehr begrenzt zwischen deren Inhalten darstellen.

Ein Content-Management-System hat den obigen Betrachtungen zufolge grundlegende Vorstellungen vom Aufbau und der äußeren Struktur der verwalteten Inhalte. Es ist in der Lage, die Inhalte in einen anderen Kontext zu stellen, miteinander zu verknüpfen und den verschiedensten Situationen angepasst neu interpretiert wiederzugeben. Ein Beispiel wäre das Ermitteln verwandter Dokumente anhand typischer darin vorkommender Begriffe oder das Zusammenfassen von Absätzen zu Abstracts. Content-Management ist also fähig, Informationen zu verwalten, jedoch kein Wissen.

Ein Wissens-Management-System ist zusätzlich zum bereits Genannten in der Lage, nicht nur die äußere sondern auch die innere Struktur der von ihm verwalteten Inhalte zu interpretieren und auf vielfache Weise auszuwerten. Es erlaubt nicht nur Verknüpfungen und Beziehungen zwischen Dokumenten, sondern auch zwischen deren Inhalten herzustellen. Das so dargestellte Wissen führt zu neuen Informationen, die das System ausgibt und weiterverarbeiten kann. Ein Beispiel wäre die Betrachtung von Kursentwicklungen und daraus ermittelte Prognosen.

2.3 Qualitätskriterien und weitere Begriffe

2.3.1 „Precision" (Präzision) und „Recall" (Ausbeute)

Das Verhältnis der Anzahl der relevanten Objekte im Suchergebnis zur Gesamtzahl der gefundenen Objekte wird als „Präzision" (Relevanzrate) bezeichnet. Es handelt sich um das Maß für das, was zu viel gefunden wird. Suchmethoden mit hoher Präzision, die wiederum gegen 100% tendiert, sind gute Suchmethoden, da sie nur wenige irrtümliche Treffer liefern.

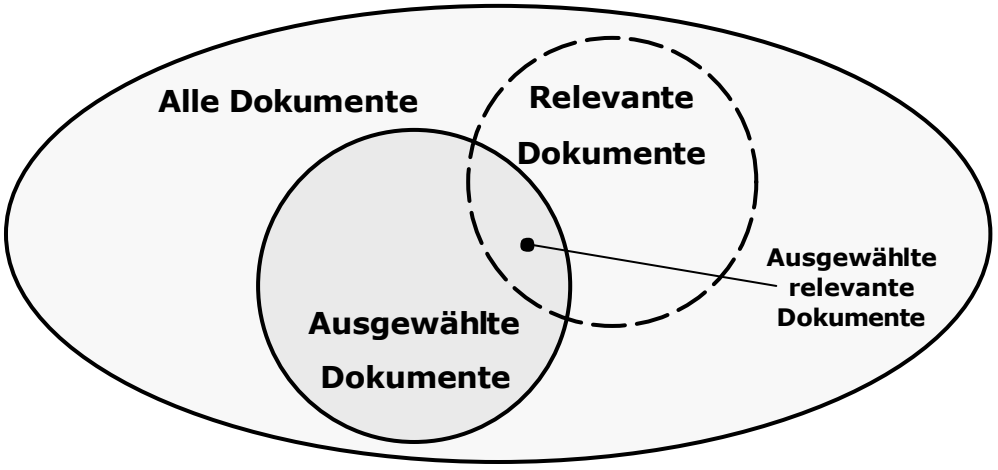

Abbildung 2: Auswahl/Relevanz-Mengendiagramm

Das Verhältnis der Anzahl der gefundenen relevanten Objekte zur effektiven Gesamtzahl der relevanten Objekte wird „Recall" bzw. Ausbeute (Abruf, Vollständigkeitsrate) genannt. Es handelt sich sozusagen um das Maß für das, was zu wenig gefunden wird. Suchmethoden mit hoher Ausbeute, die gegen 100% tendiert,

sind gute Suchmethoden, da sie eine große Anzahl der gemäß Suchkriterium qualifizierten Objekte finden.

Kurz: Die Präzision errechnet sich aus dem Verhältnis zwischen ausgewählten relevanten und allen ausgewählten Dokumenten. Die Ausbeute wird durch die ausgewählten relevanten und alle gespeicherten relevanten Dokumente bestimmt. Im hypothetischen Idealfall wären die Mengen der ausgewählten und der relevanten Dokumente deckungsgleich und die Verhältniszahlen somit 1 bzw. 100%.

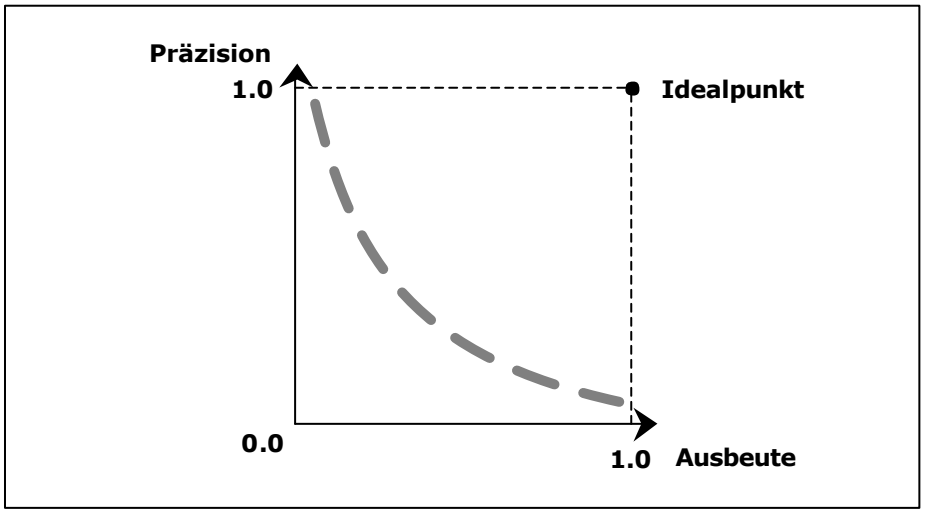

Abbildung 3: Typisches Präzision/Ausbeute-Diagramm

Typischerweise ergibt sich dieses Verhältnis zwischen Präzision und Ausbeute jedoch nie, sondern es entsteht ein Verlauf wie im obigen Diagramm[2] dargestellt. „Bessere" Methoden zeichnen sich durch Verhältniskurven aus, die flacher sind und dem Idealpunkt näher kommen.

2.3.2 „Deskribierung"

Die Transformation eines Dokumentes in eine Dokumenten-Beschreibung wird als Deskribierung bezeichnet. Dabei handelt es sich im Wesentlichen um Metadaten, wie zum Beispiel den Autor, Seitennummern und Heftnummer bei Zeitschriftenartikeln, den Text beschreibende Schlag-/Schlüssel-/Stichworte („keywords") und Zusammenfassungen („abstract").

---

[2] Querschnitt aus mehreren Quellen, u.a. [Mresse84]

2.3.3 „Recherche“

Die eigentliche Aktion des Suchens von Dokumenten unter Angabe von Termen (Suchbegriffen), Operatoren (Und, Oder etc.) und Metadaten wird unter dem Begriff Recherche zusammengefasst.

2.3.4 „Bewertung“

Die Ordnung der gefundenen Dokumente in Abhängigkeit von der gestellten Suchanfrage und/oder nach ihrer Güte wird als Bewertung bezeichnet.

2.4 Klassifikationssysteme, Thesauri, Ontologien

2.4.1 Thesauri

Die DIN 1463 beschreibt „Richtlinien für die Erstellung und Weiterentwicklung von Thesauri“. Als Definition findet sich dort: „Ein Thesaurus ist eine geordnete Zusammenstellung von Begriffen und ihren (vorwiegend natürlichsprachigen) Bezeichnungen, die in einem Dokumentationsgebiet zum Indizieren, Speichern und Wiederauffinden dient.“ Weiter schreibt die DIN: „Eine Assoziationsrelation ist eine zwischen Begriffen bzw. ihren Bezeichnungen als wichtig erscheinende Relation, die weder eindeutig hierarchisch, noch als äquivalent angesehen werden kann.“ [DIN1463] Ein Thesaurus (Nachschlagewerk) enthält also, etwas vereinfacht formuliert, eine Liste von Begriffen und typisierten Verknüpfungen zwischen diesen. Synonyme und bedeutungsgleiche oder sinnverwandte Worte sind nur einige der dabei möglichen Relationstypen. Synonyme sind also eine spezielle Art der Assoziationsrelation.

Die Zuordnungen sind nicht ein-eindeutig, da es für mehrdeutige Begriffe Bedeutungsrelationen geben kann, die sich widersprechen. Ein Beispiel dafür wäre das Wort „tragen“, dessen Synonyme „stützen“ (im Sinne von „helfen“) und „schleppen“ (im Sinne von „wegtragen“, „heben“) ihrerseits keine Relation im Sinne eines Thesaurus mehr haben.

2.4.2 Klassifikationssysteme

Die DIN 32705 schreibt: „Klassifikationssysteme sind Hilfsmittel zur Ordnung von Gegenständen oder Wissen über Gegenstände.“ Sowie: „Eine Klassifikation ist eine strukturierte Darstellung von Klassen und der zwischen den Klassen bestehenden Begriffsbeziehungen, wobei die Klassen durch (von natürlichen Sprachen unabhängige) Notationen repräsentiert werden.“ [DIN32705] Ein ganz typisches Beispiel für eine solche Klassifikation findet sich in jeder Bibliothek.

Ein Thesaurus (nach DIN) umfasst also, im Gegensatz zur Klassifikation, ein in der Regel fachlich eng begrenztes Begriffssystem. Klassifikationen umfassen viele Wortkombinationen, sind in der Regel monohierarchisch und systematisch geordnet, von der natürlichen Sprache unabhängig und weniger ausdrucksfähig. Thesauri umfassen wenige Wortkombinationen, sind in der Regel polyhierarchisch und alphabetisch geordnet, bieten einen natürlichsprachigen Zugang und sind sehr ausdrucksfähig.

2.4.3 Ontologien

Eine Ontologie bestimmt Klassen and Beziehungen zwischen diesen, um ein Wissensgebiet zu beschreiben und zu repräsentieren. [OWLFAQ] Ontologien sind also mit den Begriffen und Relationen von Thesauri identisch, führen diese Gedanken jedoch im Sinne der Objektorientierung fort. Die Klassen der Ontologie können voneinander erben, erlauben Regeln und Relationen über Relationen. Da von diesen komplexeren Möglichkeiten relativ selten Gebrauch gemacht wird, handelt es sich bei als Ontologie bezeichneten Strukturen oft lediglich um kontrollierte Vokabularien wie Klassifikationen oder Thesauri.

Ein praxisnahes Anwendungsbeispiel ist die vom W3C eingeführte Web Ontologie Language (siehe 2.9.5).

2.5 Volltextindex und Volltextsuche

Volltextdokumente setzen sich im Allgemeinen aus den Worten der natürlichen Sprache zusammen. Die Menge aller Worte wird als Vokabular bezeichnet. Die dabei verwendete Sprache spielt hierbei noch keine größere Rolle. Die Worte des Vokabulars sind wiederum aus Zeichen eines Alphabetes aufgebaut, das durch einen spezifischen Zeichensatz wie ISO-Latin1 oder Unicode repräsentiert wird. Weiterhin bestehen Volltextdokumente aus mehr oder weniger ausdrucksstarken Strukturinformationen wie Überschriften und Absätze, die zum Beispiel durch spezifische XML- oder auch HTML-Tags dargestellt werden.

2.5.1 Normalisierung

Um die in der Regel relativ unstrukturierten bzw. schlecht strukturierten Textdokumente einer Indizierung unterziehen zu können, ist ein Normalisierungsprozess angebracht und erforderlich. Dieser umfasst Bearbeitungsschritte wie:

1. Separierung von Text- und Strukturanteilen, um eine Suchanfrage beispielsweise nur auf den Volltext der Überschriften einschränken zu können.

2. Zerlegung in logische Texteinheiten anhand von Interpunktionszeichen, um eine optimale Phrasensuche zu ermöglichen.

3. Eliminierung so genannter „Stoppworte“ wie „der“ oder „was“, die zu häufig vorkommen, um noch einen sinnvollen Effekt auf Suchergebnisse zu haben. (Im Deutschen decken die 100 häufigsten Wörter bereits 50% eines einfachen Textes ab.)

4. Linguistische Normalisierung, das heißt die Vereinheitlichung von Zeitformen und Ähnlichem auf eine Grund- bzw. Stammform. Substantive werden beispielsweise auf Nominativ, Singular reduziert, Verben auf ihre ungebeugte Form.

2.5.2 Stichwortbasierte Suche

Die Stichwortsuche operiert ausschließlich auf den Worten des Volltextes, wobei es nicht möglich ist, Teilzeichenketten in den Worten zu suchen. Der Grund hierfür

liegt üblicherweise in der Basis dieser Suchmethode auf so genannten invertierten Dateien, bei denen zu jedem im Volltext vorkommenden Wort eine Reihe von Referenzen auf die Fundstellen abgelegt sind. Es können also nur Aussagen getroffen werden, die sich auf das schiere Vorhandensein ganzer Worte im Text beschränken, zum Beispiel: „Wort A kommt viermal vor“, „Wort A und B kommen vor“ usw. Eine Mustersuche im Index ist (in Grenzen) zwar denkbar, jedoch keineswegs effizient durchführbar.

Verfeinern und modifizieren lässt sich die stichwortbasierte Suche zum einen durch boolesche Operationen (Und, Oder, Nicht), durch Ranking (z.B. nach den Ähnlichkeitsmaßen des Vektorraummodells oder des probabilistischen Retrievalmodells) oder durch Kontextanfragen (Phrasen bzw. allgemeiner eine so genannte Proximity-Suche, d.h. unter Beachtung von Nähe und Wortabständen).

2.5.3 Mustersuche

Die Mustersuche operiert im Gegensatz zur reinen Stichwortsuche auch mit Wortbestandteilen. Diese Muster können neben ganzen Worten auch Wortanfänge, Wortenden (Präfixe, Suffixe) und beliebige Teilzeichenketten sein. Allgemeiner formuliert handelt es sich bei Mustern um Reguläre Ausdrücke, die aus den üblichen Literalen, Zeichenklassen, Mustern etc. bestehen.

2.5.4 Strukturierte Anfragen

Strukturierte Anfragen beachten die Dokumentenstruktur, das heißt sie können beispielsweise auf Titel oder Kapitel eingeschränkt werden. Besonders interessant wird dies im Zusammenhang mit XML, da diese standardisierte Dokumenten-Beschreibungssprache bereits alle Mittel zur Arbeit mit solchen Strukturen mitbringt. Auch mit Hilfe von Meta-Daten (siehe 2.8) lassen sich strukturiertere Recherchemöglichkeiten finden.

2.6 Retrieval-Strategien

Retrieval-Strategien – Strategien zur Wiedergewinnung und Auffindung von Dokumenten – ordnen einer Suchanfrage und den in Frage kommenden Dokumenten Ähnlichkeitswerte zu. Zumeist basieren diese Strategien auf der einfachen Idee, dass ein Dokument umso relevanter sein muss, je häufiger der gesuchte Begriff darin vorkommt. Dass dieses Konzept nicht immer richtig sein muss, wird anhand von Begriffen wie „Planke“ (mehrfache Bedeutung) oder gar „wir“ (gegen Null tendierende Bedeutung) schnell klar.

Eine Retrieval-Strategie ist ein Algorithmus, der eine Suchanfrage Q sowie eine Menge $\underline{D}$ von Dokumenten D_1, D_2, …, D_n entgegennimmt und ihnen einen Ähnlichkeitskoeffizienten $SC(Q, D_i)$ zuordnet („Similarity Coefficient“). Die Retrievalfunktion selektiert daraufhin anhand von z.B. einem Schwellwert eine Untermenge von $\underline{D}$, die im Idealfall *die und nur die* Dokumente enthält, die der Benutzer für relevant erachtet. [Grossman98]

2.6.1 Das „Vector Space Model“

Das Vektorraum-Modell ist ein einfaches, gut dokumentiertes und formalisierbares Modell, das seit Anfang der 60er Jahre von Gerhard Salton (1927-1995) und seinen Mitarbeitern sowie Nachfolgern an der Harvard-Universität entwickelt wird. Sowohl die Anfrage als auch jedes der Dokumente wird durch einen Vektor in einem n-dimensionalen Vektorraum repräsentiert, der durch die n unterschiedlichen Terme der Indizierungssprache aufgespannt wird. Im einfachsten Fall sind die Terme in den Vektoren ohne weitere Gewichtung binär kodiert (0 oder 1). Im Allgemeinen werden jedoch Häufigkeitsmaße der einzelnen Terme herangezogen, aus denen sich eine Gewichtung derselben ergibt.

Die Formel zur Berechnung des Ähnlichkeits-Koeffizienten beim Vektorraummodell lautet vereinfacht wie folgt.

$$SC(Q, D_i) = \sum_{j=1}^{t} w_{qj} \times d_{ij}$$

Dabei steht SC(Q, D_i) für den zu ermittelnden Koeffizienten zwischen Suchanfrage und Dokument. Aus der gestellten Anfrage Q leiten sich die einzelnen Gewichte w für jeden der vorkommenden Terme ab. Der Gewichtungsfaktor d_j für jeden Term t_j in jedem Dokument D_i errechnet sich letztendlich aus der Häufigkeit des Auftretens des Terms im Dokument und der Anzahl der Dokumente, die diesen Term nicht beinhalten.

Durch die Homomorphie (Strukturähnlichkeit von Mengen) von Anfrage- und Dokumentrepräsentation eignen sich diese Vektoren zum Beispiel auch für allgemeine Ähnlichkeitsanfragen oder „Relevance Feedback“.

2.6.2 Probabilistische Retrieval-Strategien

Beim probabilistischen Retrieval wird anders als beim Vektorraum-Modell die Wahrscheinlichkeit errechnet, mit der jeder Term in einem relevanten Dokument auftreten wird. Das Ähnlichkeitsmaß zwischen Anfrage und Dokument ergibt sich dann aus der Kombination aller Wahrscheinlichkeiten jedes sowohl in der Anfrage als auch im Dokument auftretenden Terms.

Eine vereinfachte Formel für die Berechnung dieser Ähnlichkeitsmaße lautet wie folgt.

$$SC(Q, D_j) = \sum_{i=1}^{t} q_i d_{ij} + \sum_{i=1}^{t} q_i d_{ij} \log \frac{N - n_i}{n_i}$$

Dabei stehen d_{ij} für die Auftritts-Wahrscheinlichkeit eines Terms t_i im Dokument D_j, q_i für die Auftritts-Wahrscheinlichkeit des Terms in der Anfrage, N für die Gesamtzahl aller Dokumente und n_i für die Zahl der Dokumente mit dem bestimmten Term.

2.6.3 Boolesches Retrieval

Konventionelles boolesches Retrieval liefert keine Informationen über eventuelle Rangfolgen, da Terme nur entweder vorkommen oder nicht vorkommen. Demzufolge kann ein einzelnes Dokument auch nur einfach gefunden (1) oder nicht gefunden werden (0). Erweiterte boolesche Methoden versuchen dagegen, jedem gesuchten

und gefundenen Term eine Gewichtung zuzuordnen, aus deren Kombination sich das Ähnlichkeitsmaß für das jeweilige Dokument errechnet. Dabei gilt für die Anfrage „t_1 AND t_2“ beispielsweise:

$$SC\left(Q_{t_1 \wedge t_2}, d_i\right) = 1 - \frac{\sqrt{\left(1-w_1\right)^2 + \left(1-w_2\right)^2}}{\sqrt{2}}$$

Dabei sind w_1 und w_2 die Gewichtungen für die gegebenen Terme. Die Teilung durch 1,414 findet statt, um die Ergebnisse auf Werte zwischen Null und Eins zu normieren.

2.6.4 „Latent Semantic Indexing“

Das Auftreten von Termen in Dokumenten wird durch eine Term-Dokument-Matrix repräsentiert, aus der das „Rauschen“ herausgefiltert wurde. Zwei semantisch ähnliche Dokumente liegen im entstehenden multi-dimensionalen Raum nahe beieinander. Kurz zusammengefasst versucht „Latent Semantic Indexing“, häufig gemeinsam auftretende Wörter miteinander in Beziehung zu setzen. Teile der Semantik eines Textes sind in diesen Wortbeziehungen verborgen (latent).

2.6.5 Strategien der Wissensverarbeitung

Die im Folgenden kurz angeführten Methoden finden nur in speziellen Anwendungsfällen Verwendung, da sie sich praktisch nicht so weit verallgemeinern lassen, dass ein ökonomischer Einsatz möglich wird.

Neuronale Netzwerke

Eine Anordnung von Knoten in einem neuronalen Netzwerk wird mit relevanten und irrelevanten Anfrage- und Ergebnismengen derart trainiert, dass sich auf eine an den Eingängen des Netzwerkes gelegte Anfrage das gesuchte Ähnlichkeitsmaß ergibt. Diese Technik ist mit den bereits erwähnten Methoden von zum Beispiel Probabilistischem Retrieval kombinierbar.

Genetische Algorithmen

Eine dem Fachgebiet der genetischen Algorithmen entstammende Retrievalfunktion könnte ein Suchergebnis durch vom Menschen kontrollierten „Versuch und Irrtum“ derart verbessern, dass die anfänglich im schlechtesten Fall rein zufälligen Fundstellen evolutionär so lange verbessert werden, bis genau das Gesuchte übrig bleibt. Dazu werden die Gewichtungen der einzelnen eingegebenen Suchterme zufällig modifiziert, das heißt „mutiert“. Als besonders überlebenswert werden dann diejenigen Anfragen bewertet, die möglichst viele der vom Anwender bereits als relevant eingestuften Dokumente beinhalten.

Diese Methode ist damit eng mit den Konzepten des Relevanz-Feedbacks verknüpft.

Wissensbasierte Fuzzy-Systeme (“Fuzzy Set Retrieval”)

Fuzzy-Set-Ansätze, das heißt auf unscharfen Mengen basierende Modelle, können als Erweiterung der Booleschen Logik („exact match“) um eine Gewichtungskomponente angesehen werden. Für die Zuordnung von Dokumenten zu inhaltlichen Kategorien gilt beispielsweise, dass keine scharfe relevant/nicht relevant-Zuordnung stattfindet, sondern verschiedene Grade an Mengenzugehörigkeit möglich sind.

2.7 Statistische Textanalyse-Verfahren

2.7.1 Soundex

Der „Soundex“-Algorithmus verfolgt vereinfacht ausgedrückt das Ziel, Worte zu finden, die ähnlich klingen, selbst wenn die genaue Schreibweise unbekannt ist. Die grundsätzlich für die englische Sprache konzipierte und mit einigen Einschränkungen behaftete Methode ist seit etwa 1900 bekannt und wurde 1918 von den Entwicklern Odell und Russell patentiert. Gedacht war sie ursprünglich für die normierte Suche nach Namen im Rahmen der amerikanischen Volkszählungen. Der historische Algorithmus, der aufgrund seiner Einfachheit in sehr vielen Programmiersprachen und Datenbanksystemen Verwendung findet, ist aufgrund dieser hohen Spezialisierung für qualitativ hochwertige Informationsrecherchen kaum geeignet. Von der Verwendung in anderen Sprachen als Englisch wird im Allgemeinen abgeraten, wenngleich

das Verfahren auch im Deutschen zwar fehlerbehaftete, aber durchaus annehmbare Ergebnisse liefern kann. Verfeinerte, unter den Begriffen „Refined“ oder „Extended Soundex“ bekannte Varianten stehen nur sehr selten als vorgefertigte Funktion zur Verfügung.

Der originale Algorithmus basiert auf der Annahme, dass viele Buchstaben wie zum Beispiel „C“ und „Z“ sehr ähnlich ausgesprochen werden, und teilt das Alphabet dahingehend in sechs Gruppen ein. Jeder Buchstabe wird dabei durch eine Ziffer ersetzt.

Tabelle 1: Zuordnungen des klassischen Soundex-Algorithmus

Ziffer	Zugeordnete Buchstaben
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z, ß
3	D, T
4	L
5	M, N
6	R

Alle weiteren Buchstaben (Umlaute, Vokale A, E, I, O, U sowie die im Englischen oft stummen H, W und Y) werden übergangen. Aufeinanderfolgende Buchstaben einer Gruppe werden zu einer einzelnen Ziffer zusammengefasst. Der erste Buchstabe eines Wortes, dem in der Regel eine höhere Bedeutung zukommt, fließt zwar mit in diese Gruppierung ein, wird jedoch von der Übersetzung ausgeschlossen. Schlussendlich wird das entstandene Muster noch auf den Anfangsbuchstaben sowie die ersten drei Ziffern gekürzt bzw. auf diese Länge mit Nullen aufgefüllt.

```
                                Name: MÄTTIG
Zusammenziehen gleicher Wertigkeiten: MÄTIG
              Zuordnung der Ziffern: M32
               Normierung der Länge: M320
```

Abbildung 4: Beispiel für den Soundex-Algorithmus

Namen wie z.B. „Müller, „Müler“, „Mueller“ oder sogar „Müllre“ haben dieser Transformation zufolge alle den selben Code (hier: „M460“).

Das Verfahren hat ganz klare Nachteile, die es für Anwendungen der Volltextsuche ungeeignet machen. In erster Linie ist die Kürzung auf die ersten vier relevanten Buchstaben zu nennen. Auch die Auslegung auf das Englische und die Zuordnung zu nur sechs Buchstabengruppen lassen oft zu ungenaue Ergebnisse entstehen. Erweiterte, an Soundex angelehnte Verfahren wie z.B. Metaphone verbessern die Ergebnisse zum Teil deutlich, ändern aber an der starken Spezialisierung des zugrunde liegenden Prinzips nicht viel.

2.7.2 N-Gramm/Bi-Gramm-Analyse

Bei der so genannten „N-Gramm-Analyse“ handelt es sich um ein einfaches Verfahren zur Bestimmung der statistischen Verteilung von Buchstabenfolgen für bestimmte ausgewählte Textfragmente, Dokumente oder Dokumentenfamilien. Ausgehend von der Textbasis, die beliebig groß oder klein sein kann, wird gezählt, wie oft beispielsweise die Zeichenfolge „qu“ in diesem Zeichenstrom vorkommt. Dies wird für jede mögliche Buchstabenkombination wiederholt.

Üblicherweise werden die ermittelten Häufigkeiten auf Prozentwerte normiert, und zwar gruppenweise. Es entstehen somit prozentuale Häufigkeiten, die zum Beispiel aussagen, wie wahrscheinlich es ist, dass hinter einem „a“ ein „v“ folgt. Die für „a“ insgesamt möglichen 26 Folgebuchstaben ergeben zusammen eine Wahrscheinlichkeit von 100%. Im absoluten Durchschnitt wird also mit Werten von zirka 3,85% hantiert. Eine zweite Möglichkeit ist das Verteilen der Gesamt-Wahrscheinlichkeit über die gesamte 26 mal 26 Felder große Matrix. Hier wird also mit durchschnittlichen Wahrscheinlichkeiten von 0,15% hantiert.

Bei der Normierung der Verteilungswerte je Gruppe werden Kombinationen mit seltenen Zeichen am Anfang deutlich höher gestuft. Beispiel: Kommt in der gesamten Textbasis nur eine einzige Kombination mit einem führenden „q“ vor, so erhält diese eine volle Wertung von 100%, was in einer hypothetischen Umrechnung auf die gesamte Matrix etwa 3,99% ergeben würde. Dagegen würde eine von vornherein

ausgeführte Gleichverteilung über die gesamte 2-dimensionale Matrix nur 0,15% ergeben.

Zusammenfassend lässt sich also festhalten, dass die relative Normierung pro Gruppe seltene Buchstabenkombinationen deutlich anhebt, und zwar um Faktoren bis zu 20. Folgen mit häufig vorkommenden Buchstaben werden dagegen in beiden Varianten identisch behandelt, mit (idealisierten) 0,15%. Das ist zum Beispiel der Fall, wenn von „aa“ bis „az“ alle Kombinationen jeweils einmal vorkommen.

Das Verfahren findet in der Praxis hauptsächlich in der Spracherkennung Anwendung. Dabei wird versucht, die Häufigkeitsverteilung eines Dokumentes einer für jede Sprache ganz typischen Vergleichsmatrix zuzuordnen. Dies liefert trotz der erstaunlichen Einfachheit dieser Technik sehr genaue Ergebnisse, die nur bei gemischtsprachigen oder Texten mit sehr vielen Fremdwörtern eventuell versagen.

Interessant wird die N-Gram-Analyse dadurch, dass sie im Grunde den typischen „Klang“ einer Sprache erfasst und erkennen kann. So lassen sich zum Beispiel aus einer Reihe beliebiger Wörter diejenigen herausfinden, die für das Lateinische typische Buchstaben-Kombinationen aufweisen. Denkbar sind auch Verfahren, welche die typische annähernde Gleichverteilung von Zufallsgeneratoren erkennen und so Wörter und Texte identifizieren, die nicht von Menschen sondern zufallsgesteuert von Computerprogrammen geschrieben wurden. Darauf aufbauend entsteht eine Relevanzbewertung, die in der Lage ist, beispielsweise den Datenmüll des in jüngster Zeit immer aggressiver werdenden E-Mail- und Suchmaschinen-Spams zu filtern.

Eine weitere Einsatzmöglichkeit wäre ein Ähnlichkeitsvergleich zwischen Dokumenten. Obwohl es dafür etabliertere Methoden gibt, die zumeist auf der Häufigkeitsverteilung ganzer Wörter basieren, wurde im Rahmen dieser Arbeit ein Prototyp auf Basis einer Bi-Gram-Analyse implementiert und getestet. Die Methode lieferte in den Tests jedoch unausgewogene Ergebnisse. Einerseits wurden Texte mit typischen und häufigen Fachbegriffen relativ zuverlässig als ähnlich erkannt. Andererseits wirkte sich die Tatsache, dass gleiche Buchstaben-Kombinationen auch in inhaltlich völlig unterschiedlichen Wörtern vorkommen können, in vielen Fällen negativ auf das Ergebnis aus.

Zusammenfassend lässt sich durch die Untersuchung also sagen, dass auf den Inhalt bezogene Ähnlichkeitsanalysen auf Basis von Wörtern deutlich zuverlässiger sind. Letztendlich ist dies auch genau die Methode, die das Oracle-System für seine verschiedenen Klassifizierungsmethoden einsetzt. Für Einsatzzwecke, die nicht den Inhalt, sondern nur die verwendete Sprache eines Textes behandeln sollen (auch „Programmiersprache"), kann die N-Gramm-Analyse dagegen mit dem entscheidenden Vorteil einer sehr geringen Komplexität dienen.

2.7.3 Lesbarkeitsgrad – Der Flesch-Index

Als weiteres Beispiel für ein statistisch fundiertes Klassifikationsverfahren soll kurz eine eigentlich für andere Zwecke gedachte Methode vorgestellt werden. Rudolf Flesch – ein 1938 in die USA ausgewanderter Wiener – entwickelte in seinem Buch „How to Write Plain English" (Barnes & Noble Books, 1979) eine Formel zur Bewertung der Leseleichtigkeit beliebiger englischer Texte. Obwohl die Konstanten seiner Formel für englische Texte ausgelegt sind, lassen sich die Ergebnisse leicht abgewandelt auch aufs Deutsche übertragen.

```
FI = 206,835 – 84,6 × WL – 1,015 × SL
```

Abbildung 5: Formel zur Berechnung des Flesch-Index

Dabei steht `FI` für den resultierenden Flesch-Index, `WL` für die durchschnittliche Silbenzahl je Wort und `SL` für die durchschnittliche Wortanzahl je Satz. Als einzige Besonderheit erwähnt Flesch, dass bei Wörtern mit mehreren möglichen Aussprachen diejenige mit der kürzesten Silbenanzahl anzunehmen ist. Im Deutschen betrifft das vor allem stumme Endsilben auf -e, da zum Beispiel „viele Feinde, viel Ehre" in der Regel „viel' Feind', viel Ehr'" gesprochen wird. Eine sehr einfache näherungsweise Ermittlung der Silbenzahl ist beispielsweise durch Zählung der Vokale und Diphthonge (im Deutschen „au", „ei/ai" sowie „eu/äu" [WikiDiph]) innerhalb eines Wortes realisierbar – unter Beachtung eventueller Sonderfälle und besagtem End-e.

Flesch wählte die Konstanten der Formel so, dass auf der ansonsten offenen Skala typischerweise Ergebnisse im Bereich zwischen 0 und 100 entstehen, wobei größere Zahlen für leichtere Lesbarkeit stehen. Da deutsche Texte durch die Verwendung

zusammengesetzter Substantive oftmals deutlich längere Worte beinhalten, ist eine im Vergleich zum Englischen um etwa 10 Punkte nach unten verschobene Vergleichsskala anzunehmen. Die Luther-Bibel beispielsweise erreicht nach [Bachmann03] einen Index von 76, der einem „sehr leichten" Niveau der 6. bis 8. Klasse entspricht. Werte zwischen 20 und 30 können als schwierig lesbares „Abitur-Niveau" angesehen werden – in diesen Bereich fällt übrigens auch diese Arbeit.

Obwohl das Verfahren lediglich Satz- und Wortlängen statistisch auswertet und somit nichts über den Inhalt und nichts über die Verständlichkeit eines Textes aussagt, lassen sich zum Teil verblüffende Rückschlüsse auf den Satzbau und damit die Lesbarkeit bzw. Lesetauglichkeit finden. Texte mit einem hohen Flesch-Index müssen aufgrund dieser Unzulänglichkeiten nicht unbedingt leicht lesbar sein. Umgekehrt jedoch ist es nahezu unmöglich, Texten trotz eines niedrigen Flesch-Indexes eine gute Lesbarkeit zu bescheinigen.

2.8 Metadaten

„Meta" lautet ein besonders schöner Universalbegriff des Informationszeitalters. Metadaten werden dabei im Allgemeinen als „Daten über Daten" verstanden.

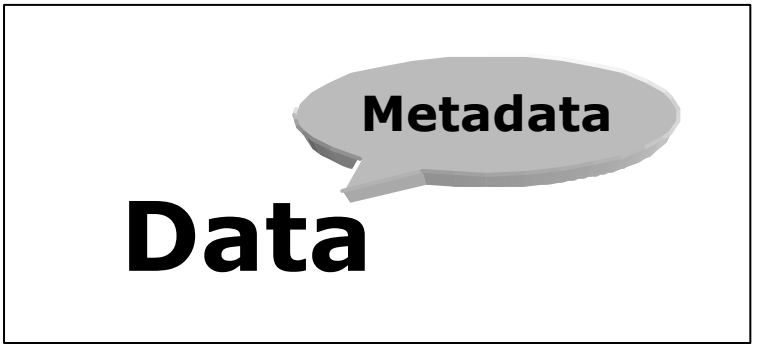

Abbildung 6: Metadaten sind Daten über Daten

Die Metadaten „sprechen" über die eigentlichen Daten und liefern Hintergrundinformationen, die aus den Daten selbst nicht ermittelbar sind, wie zum Beispiel Herkunft, Autor, Alter oder Relationen zu anderen Daten. Dieser Betrachtung zufolge sind sogar „Metadaten über Metadaten" formulierbar, was im jeweiligen Kontext durchaus gängig sein kann (zum Beispiel beim „Resource Description Framework", siehe 2.9.4).

Für die Betrachtungen im Rahmen dieser Arbeit sind Metadaten wichtig, die sich in die Begriffswelten von Thesauri und Ontologien einordnen lassen. Das schließt Übersetzungen und alternative Schreibweisen ebenso ein wie über- und untergeordnete Begriffe, inhaltliche und begriffliche Verwandtschaften sowie typisierte, gewichtete und gerichtete Relationen. Auch Elemente der Textauszeichnungssprache (zum Beispiel HTML), die Informationen ohne unmittelbare darstellende Wirkung bereitstellen, werden in der Regel – so auch hier – mit dem Begriff „Metadaten" erfasst. Auch der Name eines HTML- oder auch XML-Elements wie zum Beispiel „`<strong>`" fällt darunter.

„Metadaten" sind im Kontext dieser Arbeit also in erster Linie als jedwede Art externen Zusatzwissens zu verstehen, das weder den zu durchsuchenden Dokumenten noch den Suchanfragen selbst unmittelbar entnommen werden kann. Die Spanne reicht dabei zum Beispiel vom Namen des Autors bis zu Abkürzungs- und Begriffserklärungen (sofern nicht im Dokument enthalten). Die unter 2.9 vorgestellten Technologien wie das bereits erwähnte RDF bauen sehr stark auf Metadaten und der Arbeit mit ihnen auf bzw. stellen spezialisierte Repräsentationsformen für diese dar.

2.9 Standards/Web-Standards

Um effiziente und vor allem ökonomische Lösungen entwickeln zu können, ist es äußerst sinnvoll, die konkret zu implementierenden Anwendungen an existierenden Standards und Empfehlungen zu orientieren. Einige der für das Gebiet des IR interessanten Standards sollen hier vorgestellt werden.

2.9.1 Geschichtlicher Rückblick

Vor nunmehr 13 Jahren – im November 1990 – wurde das World Wide Web (WWW) am europäischen Labor für Teilchenphysik (CERN) mit einer ersten Internetseite initiiert. Seitdem haben sich das Web und seine Sprachen stetig weiterentwickelt. Im Bezug auf Fragen der Datenbanktechnologien, der Darstellung, Strukturierung und Verknüpfung von Informationen und den Möglichkeiten zur Filterung und Suche dieser Informationen kann das WWW als führender, zumindest aber als populärster Schauplatz des Aufstiegs und Falls von Technologien betrachtet

werden. Allerdings war die Richtung dieser Entwicklungen sehr oft darstellungsbezogen – Schriftgrößen, Farben, Animationen, Effekte. In Verbindung mit der einfachen Erlernbarkeit der Technologien war dies natürlich einer der Gründe für die ungemein schnelle und weitläufige Verbreitung des Webs, das heute in sehr viele auch alltägliche Bereiche des Lebens hineingreift.

Die inhaltliche Strukturierung und automatische Erfassbarkeit von Texten wird erst langsam immer wichtiger und dank der Bemühungen von Institutionen wie zum Beispiel dem World-Wide-Web-Konsortium (W3C) auch zögernd greifbar. In den jungen Jahren von Web- und Datenbanktechnologien stand der Wunsch nach Erfassung möglichst vieler Daten im Vordergrund. Bereits die einfachsten Technologien zur Suche in diesem Datenpool reichten aus, um die Effektivität von in Papierform erfassten Informationen weit zu übertreffen. Heute wächst die Datenflut derart explosionsartig an, dass in aller Regel nicht mehr das Problem ungenügender Daten auftritt, sondern die relevanten und eigentlich gesuchten Informationen in einer nur schwer filterbaren Masse unterzugehen drohen. „Wir ertrinken an Informationen und hungern nach Wissen“ schrieb John Naisbitt, amerikanischer Zukunftsforscher, schon im Jahre 1990 sehr treffend.

Noch können Suchmaschinen weder unterscheiden, ob mit „Kohl“ ein Gemüse oder eine Person dieses Namens gemeint ist, noch können sie sicher sein, ob „01189“ tatsächlich eine Postleitzahl meint. Diese und ähnliche Beispiele mögen dem an suchwortbezogene Suchmaschinen gewohnten Internetnutzer wie seltene Grenzfälle vorkommen. Tatsächlich aber ist es alltäglich, ausufernde Suchergebnisse mehrfach einzuschränken, nach Synonymen zu fahnden und sich minutenlang durch Ergebnislisten zu klicken, um die Relevanz der gefundenen Stellen mit den eigenen Augen abzuschätzen – eben das, was uns eigentlich von der Suchmaschine abgenommen werden sollte.

Wie groß – oder klein – ist der Prozentsatz der Suchanfragen, die in einem überschaubaren und brauchbaren Ergebnis von 10 bis 20 Fundstellen resultiert? Dass dies selten vorkommt und sehr viel „Medienkompetenz“ erfordert, illustriert sehr gut das, was wir heute als „das Netz“ kennen.

2.9.2 Markupsprachen – GML, SGML, XML 1.1, XHTML 2.0

SGML, die Standard Generalized Markup Language, ist eine seit 1986 standardisierte (ISO 8879), rechnerunabhängige Vereinbarung für Taggingsysteme, deren primäres Entwurfsziel die Übertragung beliebiger Informationen ist. Unter Taggingsystemen werden Systeme verstanden, die eine Definition bestimmter Schlüsselbegriffe erlauben, mit deren Hilfe zum Beispiel die Bedeutung von Dokumentenabschnitten festgelegt wird. Dabei handelt es sich nicht um Formatanweisungen, also so genanntes prozedurales Markup (wie typischerweise vom inzwischen überholten HTML 3.2 bekannt), sondern um eine Klassifizierung oder deskriptives Markup, das eine an dieser Stelle noch unbekannte Art der Weiterverarbeitung ermöglicht.

Die Ursprünge von SGML reichen zurück bis ins Jahr 1969 und fußen hauptsächlich auf den Forschungen von Charles F. Goldfarb. Bemerkenswerterweise gründet sich die damalige Entwicklung auf den Anspruch, Textdokumente im Rahmen eines Information Retrieval-Systems besser handhaben zu können. Mit anderen Worten, ohne IR gäbe es kein SGML.

```
.SK 1
Text processing and word processing systems typically require
additional information to be interspersed among the natural text of
the document being processed. This added information, called
"markup", serves two purposes:
.TB 4
.OF 4
.SK 1
1.#Separating the logical elements of the document; and
.OF 4
.SK 1
2.#Specifying the processing functions to be performed on those
elements.
.OF 0
.SK 1
```

Abbildung 7: Markup-Beispiel eines Vorgängers von XML

Die Ansprüche an automatisierbare und wiederverwendende Eigenschaften von Textverarbeitungs-Systemen, naheliegenderweise vor allem im Bereich der Tageszeitungen, stiegen mit Beginn der Computerisierung rasant an. Die bisherige Arbeit bestand prinzipiell darin, Dokumentenauszeichnungen wie Schriftart und Schriftgrö-

ße einer Überschrift in einem formatlosen Rohdruck mit dem Stift einzuklammern und mit Anweisungen für den Schriftsetzer zu versehen. Angelehnt daran entstanden die ersten digitalen Systeme, die einem generellen Markup schon sehr nahe kamen, jedoch noch zu seiteneffektbehaftet waren. [Goldfarb96]

Die Standardisierung der (Meta-) Sprache SGML als ISO-Norm war einer der wichtigsten Schritte auf dem Weg von diesen ersten Anfängen bis heute. Mit Entwicklung des WWW entstand dann wenige Jahre später, 1989, das an SGML orientierte und dank seiner leichten Erlernbarkeit ungeahnt erfolgreiche HTML (1.0, wobei es damals noch keine Versionsnummer hatte). Erst sehr viel später zeichnete sich immer mehr die Notwendigkeit ab, über eine allgemeingültigere Sprache zu verfügen, die nicht so darstellungsorientiert und unflexibel war wie HTML. XML, die von einer Arbeitsgruppe des W3C entwickelte, stark vereinfachte Variante von SGML, wurde im Februar 1998 vorgestellt und freigegeben und erfreut sich seitdem stetig wachsender Beliebtheit.

XML 1.1, XHTML 2.0

Die vor kurzem vorgestellte neue Version 1.1 des XML-Standards ändert gegenüber der etablierten Fassung – abgesehen von einigen Details, die Zeichensätze und Kodierungen betreffen – nichts. Dies zeigt auch, dass es gelungen ist, die (Meta-) Sprache so unkompliziert und allgemeingültig zu konstruieren, dass sie sämtlichen vorstellbaren Anforderungen gerecht werden kann.

Deutlich durchschlagender sind die Änderungen, die der Entwurf für ein zukünftiges, noch nicht vollständig fertiggestelltes XHTML 2.0 aufzeigt. Im Gegensatz zu den Vorgängern HTML 4.01 und XHTML 1.0, die sich auf technischen Gegebenheiten geschuldeten Anpassung und Erweiterung konzentrierten, wird XHTML 2.0 fast völlig auf den bisher fokussierten Aspekt der Abwärtskompatibilität verzichten. Das Konsortium versucht so, die in früheren HTML-Versionen aufgeweichte Trennung zwischen prozeduralem und deskriptivem Markup längerfristig wieder herzustellen (indem darstellungsbezogene Elemente komplett auf die Ebene der Cascading Style Sheets, CSS, verschoben werden). Der momentane Standard HTML 4.01 sowie sein

XML-konformes Äquivalent XHTML 1.0 werden parallel zu dieser begrüßenswerten Entwicklung allerdings auch weiterhin ihre volle Gültigkeit behalten.

Das semantische Web

Standardkonform zu arbeiten und sich beispielsweise an XHTML 2.0 zu halten bedeutet jedoch nicht automatisch, semantisch wertvolle Inhalte zu erstellen. Semantik ist viel mehr als das Einhalten einer W3C-Empfehlung. Dass Dokumente erfolgreich und ohne Fehlermeldungen oder Warnhinweise durch (X)HTML-Validatoren laufen besagt letztendlich nur, dass ausschließlich explizit erlaubte Tags in ihrer explizit erlaubten Reihenfolge und Verschachtelung verwendet wurden. Es sagt nichts darüber aus, wie diese Tags tatsächlich verwendet wurden. Ein typisches Beispiel ist das im Internet nach wie vor stark verbreitete grafische Gestalten ganzer Seiten mit blinden Tabellen.

Die Eigenschaft „semantisch wertvoll" wird in vielen Artikeln, Essays und Diskussionen[3] oftmals in einem Zug mit „Accessibility" (Zugänglichkeit) aber auch „Usability" (Nutzbarkeit) und „Readability" (Lesbarkeit) genannt. Eine konsequente semantische Anreicherung von Dokumenten endet nicht bei der Verwendung oder Nichtverwendung bestimmter empfohlener oder missbilligter Auszeichnungselemente. Ein XHTML 2.0-konformer „`<title />`" wird nicht besser, wenn er „`<title>New Page 1</title>`" lautet (man möge sich den Spaß machen, das Web nach diesem Standard-Seitentitel eines verbreiteten HTML-Editors zu durchsuchen: `http://www.google.de/search?q=allintitle:"New Page 1"`).

Aber: (X)HTML ist keine semantische Sprache.[4] HTML entstand von Anfang an als Mischform einer leidlich strukturierten, stark visuell orientierten Textauszeichnungssprache. Die Bemühungen des W3-Konsortiums bewegen sich zwar in diese Richtung, indem tatsächlich rein visuelle Elemente konsequent auf die Ebene der Cascading Style Sheets verschoben werden, dennoch ist und bleibt der Einsatzzweck von HTML grundsätzlich visuell. [Watson]

---

[3] u.a. <http://mezzoblue.com/>, <http://diveintoaccessibility.org/>
[4] Zitat <http://www.simplebits.com/archives/2003/09/28/about_the_talking_about_semantics.html>

So lange es dem Anwender möglich ist, mit beispielsweise „“ oder gar „<font size="7">“ den selben Effekt zu erzielen wie mit dem in diesem Fall eigentlich wünschenswerten „<h1>“, gibt es keinen wirklich zwingenden Grund, diese letztendlich ja nur „Empfehlungen“ konsequent umzusetzen.

Im Zusammenhang mit den hier behandelten Methoden des Information Retrieval kann das im völligen Gegensatz dazu jedoch ganz anders aussehen. Dem System – hier Oracle Text – ist es nur möglich, nach Überschriften zu fahnden, wenn es auch in der Lage ist, Überschriften zu erkennen. Standards wie das gezeigte „<h1>“ sind dabei extrem hilfreich, wenngleich es mit deutlich höherem Aufwand natürlich auch möglich ist, dem System beizubringen, „<font size="7">“ als Überschrift zu erkennen. Der bereits mehrmals erwähnte ökonomische Nutzen kann, wie an diesen eigentlich einfachen Beispielen zu sehen, eine entscheidende Rolle spielen.

2.9.3 Dublin Core Metadata Initiative – DC

Eine unter dem Namen „Dublin Core“ vereinte internationale Gruppe von Experten stellt eine auch vom W3C begrüßte Empfehlung bereit, die allgemeingültige Metadaten (siehe 2.8) wie „Urheber“, „Titel“ aber auch „Thema“ und „Datum der letzten Änderung“ weitestgehend formalisiert. Die Austauschbarkeit solcher grundlegenden Metainformationen ist auf diese Weise immer gewährleistet, da Metaelemente mit gleicher Bedeutung immer unter gleichem Namen auftreten, selbst wenn das beschriebene Dokument nicht austauschbar ist. Der Standard bleibt dabei sehr unkompliziert, da er im Grunde nur einen Satz von Schlüsselwörtern und die ihnen zukommende Bedeutung spezifiziert. Dublin Core-Metadaten gliedern sich zum Beispiel in HTML ebenso nahtlos ein wie in RDF (siehe 2.9.4). So lassen sie sich zur Beschreibung jeglicher Quellen nutzen, von Text- bis hin zu Grafikdokumenten.

2.9.4 Ressource Description Framework – RDF

Das vom W3C standardisierte Ressource Description Framework, kurz RDF, ist ein eingängiges und vor allem sehr praxisnahes Beispiel für die Arbeit mit Metadaten und Klassifikationssystemen. Jeder Quelle wird hierzu ein Block von Informationen zugeordnet, die Zusatzinformationen oder Verknüpfungen zu anderen Quellen

enthalten. Die Informationsblöcke lassen sich weiter verknüpfen, so dass ein Netzwerk von aufeinander verweisenden und sich gegenseitig näher beschreibenden Quellen entsteht. Der Umfang der in XML formulierten „Quellen-Beschreibungssprache“ ist dabei sehr gering und lässt sich über einzubindende externe Klassen nahezu beliebig erweitern. So kommt oftmals zum Beispiel auch die Klasse der DC-Metadaten für die Quellenbeschreibung zum Einsatz.

RDF lässt sich nutzbringend überall dort einsetzen, wo Internetquellen mit Metadaten näher beschrieben, verknüpft und klassifiziert werden sollen, die Quellen selbst jedoch keine Möglichkeit zur Ablage solcher Daten mitbringen. Überdies sind RDF-Daten von jeder dafür entwickelten Standardsoftware auswertbar, ohne das die tatsächlichen Dateiformate der beschrieben Netzquellen lesbar oder gar bekannt sein müssten.

2.9.5 Web Ontology Language – OWL

Die aus DAML (DARPA Agent Markup Language) und OIL (Ontology Inference Layer) entstandene Sprache entsprang den seit etwa 2002 geführten Bemühungen des W3-Konsortiums, eine einheitliche Sprache für die Abbildung von Ontologien zu bestimmen. Genau wie seine Vorgänger sowie die zwischenzeitlich entstandene Entwicklungsstufe „DAML+OIL“ basiert OWL dabei vollständig auf dem durch RDF vorgegebenen Rahmen. Da RDF wiederum auf XML aufbaut, ist auch OWL sehr leicht maschinenlesbar.

Das Hauptziel der Sprache ist, eine vollständig maschinell verarbeitbare hierarchische Beschreibung von Klassen einer Domäne und ihren Eigenschaften, das heißt Attributen und Assoziationen, bereitzustellen. Eine OWL-Ontologie kann aus den nachfolgenden Elementen bestehen: Taxonomische Beziehungen zwischen Klassen, Datentypen-Eigenschaften (nähere Beschreibung der Attribute von Klassen-Elementen), Objekt-Eigenschaften (Beschreibung der Beziehung zwischen Klassen-Elementen) sowie in begrenztem Umfang Instanzen von Klassen und Instanzen von Eigenschaften. Die drei aufeinander aufbauenden Untermengen der Sprache „OWL Lite“, „OWL DL“ („Description Logics“) sowie „OWL Full“ erlauben dabei die

Verwendung von genau der Anzahl Sprachelemente, die für den jeweiligen Anwendungsfall tatsächlich benötigt werden. [OWL]

Das W3C erläutert die Eigenschaften der Sprache unter anderem anhand eines eingängigen Beispieles, das Wein-Sorten, Eigenschaften wie Farbe, Anbaugebiete und vieles Weitere miteinander in Beziehung stellt. Es entsteht ein wachsendes Netzwerk von Informationseinheiten wie beispielsweise „Riesling – hat Farbe – weiß“. „Riesling“ ist dabei eine Klasse, „hat Farbe“ eine Eigenschaft und „weiß“ der Wert der Eigenschaft, wobei „weiß“ wiederum eine Ausprägung der Klasse „Weinfarbe“ ist und „Weinfarbe“ eine Untermenge der Klasse „Weineigenschaften“, welche sich neben der Farbe noch aus „Weingeschmack“ zusammensetzt.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  …
</owl:Class>
```

Abbildung 8: OWL-Beispiel (Ausschnitt)

Wie das Beispiel zeigt, ließe sich das angedeutete Beziehungsgeflecht nahezu endlos fortsetzen, ohne dabei die maschinelle Lesbarkeit zu verlieren.

LBase – Semantics for Languages of the Semantic Web

Bei LBase handelt es sich um eine „Meta“-Sprache, die dazu dienen soll, die Semantiken von Sprachen wie OWL und RDF formal zu beschreiben. Dazu wird versucht, die Inhalte jeder beliebigen denkbaren semantischen Sprache auf einen kleinsten gemeinsamen Nenner zu bringen. Diese kleinste Basis ist LBase. Da die Empfehlung des W3C bezüglich LBase noch keinen hinreichend finalen Status erreicht hat, sind die Einsatzmöglichkeiten bisher nur schwer abzuschätzen.

3 Ist-Analyse

3.1 Referenzprojekt IS-GBE

Das Internet-gestützte Informationssystem „Gesundheits-Berichterstattung“ (IS-GBE, http://www.gbe-bund.de/) besteht in der von Robotron erstellten Entwicklungsphase IV und basiert zu diesem Zeitpunkt weitestgehend auf den von Oracle8*i* gebotenen Möglichkeiten.

Das System bietet einen Einstieg über eine Volltextsuche sowie über eine vom Menschen vorgenommene Klassifizierung der Textdokumente, Tabellen etc. Alle Dokumente sind durch mehrere Arten von Verwandtschaften miteinander verwoben. Neben manuellen Verknüpfungen, bei denen die Beziehungen zwischen den Dokumenten jeweils explizit bestimmt werden müssen, gibt es ein komplexes System von regelbasierten Links, die sich zum Beispiel durch bestimmte Stichwörter ergeben. Diese Mechanismen greifen in der jetzt bestehenden Ausbaustufe des IS-GBE kaum auf Funktionalitäten der Oracle-Datenbank zurück. Die Untersuchung dieser momentanen Umsetzung ist kein unmittelbarer Bestandteil der vorliegenden Arbeit. Dennoch soll im Laufe der folgenden Kapiteln auch auf Ausbaumöglichkeiten dafür eingegangen werden.

Die recherchierbaren Dokumente gliedern sich in statische Textdokumente, wobei es Sonderformen wie „Definitionen“ und ähnliches gibt, statische Tabellen sowie dynamische, so genannte „Ad-hoc“-Tabellen (zur Anfragezeit generiert). Letztere lassen sich vom Anwender beliebig filtern und umordnen, so dass er aus dem zugrunde liegenden mehrdimensionalem (relationalem) Datenbestand von einigen hunderttausend Werten genau die Informationen erhält, die er sucht. Eine Volltextsuche über diese Ad-hoc-Tabellen ist ebenfalls möglich, beschränkt sich dann naturgemäß aber auf die Beschriftungen der Dimensionen der Tabelle sowie die gesondert vorliegenden Kurzzusammenfassungen.

IS-GBE basiert in seiner seit 2002 bestehenden Ausbaustufe IV auf den Möglichkeiten, die Oracle8*i* bietet, sowie einer Anzahl speziell entwickelter PL/SQL-Prozeduren für zum Beispiel die Arbeit mit Thesauri, Verwandtschaften und

Ähnlichem. Überlegungen zur Integration neuerer Oracle9*i*- und evtl. auch Oracle10*g*-Eigenschaften werden eine zentrale Rolle in den folgenden Kapiteln spielen.

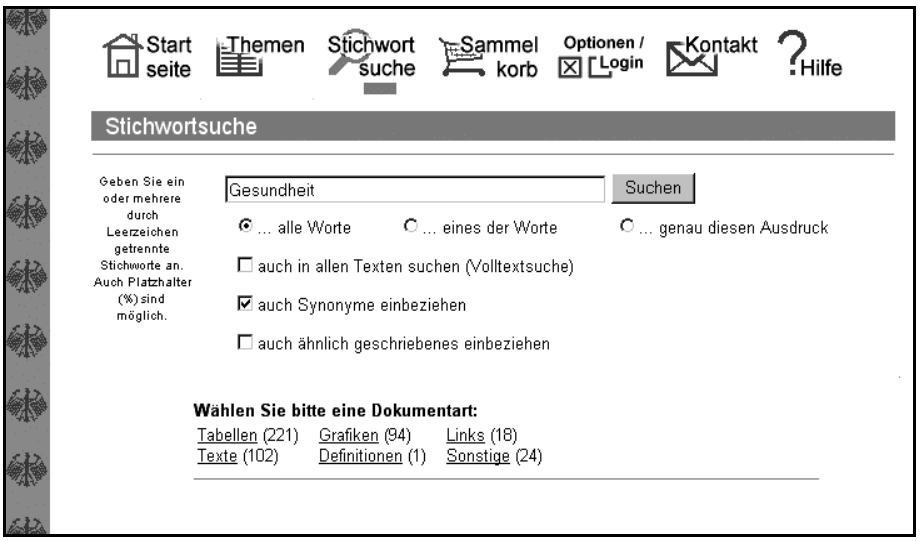

Abbildung 9: Stichwortsuche im IS-GBE

Die recherchierbaren Inhalte liegen in einer relationalen Datenbank in einem speziell strukturierten Dokumentenbereich (HTML-Dokumente, d.h. Texte) sowie einem multidimensionalen Statistikbereich vor (tabellarische bzw. n-dimensionale Daten). Statistiken beschreiben sich über ihre Ausmaße selbst. IS-GBE benutzt mehrere zum Teil ontologische Begriffssysteme: Themenkatalog, Synonym- sowie Merkmalskatalog.

Der Themenkatalog gliedert sich hierarchisch in drei Ebenen: Themenbereiche, Unterthemen und Gliederungspunkte (Begriffe). Dokumente werden diesem Katalog manuell zugeordnet. Daneben gibt es einen flachen Aspektekatalog. Jeder Themenbegriff gehört zu genau einem Aspekt.

Der Merkmalskatalog stellt sich zweistufig dar. Zu einem Merkmal (Oberbegriff) gehören beliebig viele Begriffe (Schlüsselbegriffe).

Im Sinne von Synonymlisten kommen mehrere flache Thesauri zum Einsatz. Neben spracheigenen Synonymkatalogen gibt es Übersetzungskataloge zum Auffinden fremdsprachlicher Dokumente. Diese Synonymlisten werden im Rahmen der Suche – optional – einbezogen.

3.1.1 Die Datenbasis – HTML?

Die Textdokumente des bestehenden IS-GBE wurden in aller Regel mit Microsoft Word 97 erfasst und im Vertrauen auf die Fähigkeiten dieser Standardsoftware über den regulären Dialog „Speichern unter…“ als Dateien mit der Endung „`.html`“ abgelegt. Da die so vorliegenden grafischen Textauszeichnungen hauptsächlich der Bildschirmpräsentation dienen, ergaben sich vorerst keine Probleme. Eine nähere Betrachtung der vorliegenden Dateien ergibt jedoch – gerade im Hinblick auf die in späteren Kapiteln angestrebte Arbeit mit XML und XHTML – ein unschönes Bild.

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type"
      CONTENT="text/html; charset=windows-1252">
    <META NAME="Generator" CONTENT="Microsoft Word 97">
    <TITLE>
      6
    </TITLE>
    <META NAME="Template"
      CONTENT="C:\Programme\office97\Office\html.dot">
  </HEAD>
  <BODY LINK="#0000ff" VLINK="#800080">
    <B>
      <FONT FACE="MyriaMM_215 LT 600 NO" SIZE=5 COLOR="#00734B">
        <P>
          6.12 Station&auml;re und teilstation&auml;re Pflege
        </P>
      </B>
    </FONT>
    …
    <FONT SIZE=2>
      <P>
         
      </P>
    </FONT>
<HTML>
  <HEAD>
    …
```

Abbildung 10: Auszug aus einem typischen IS-GBE HTML-Dokument

Wie im obigen Beispiel zu sehen hat das, was Microsoft Word 97 standardmäßig als „HTML“ generiert, kaum etwas mit den heute etablierten Standards gemein. Abgesehen von dem etwas problematischen Zeichensatz „windows-1252“ („CP1252“), der nicht völlig kompatibel zum standardisierten „ISO-8859-1“ ist, der Großschreibung aller Tags (untersagt nach XHTML), kreuz-verschränkten Verschachtelungen und dem weitestgehenden Fehlen strukturierender Elemente wie „`<h1>`“ fallen auch

offensichtliche Fehlbedienungen der Anwendung auf. So ist beispielsweise die Titel-Angabe so belassen worden, wie vermutlich von Word vorgeschlagen (hier: „6“). Ein automatisiertes Herausgreifen des Dokumententitels wird so – in Kombination mit dem fehlenden `<h1>` – stark erschwert.

Weiterhin gibt es – offensichtlich entstanden durch das Zusammenkopieren mehrerer Dokumente – zusätzliche `<html>`, `<head>` und `<body>`-Elemente im fortlaufenden Text.

Die beschriebenen Probleme treten bei nahezu allen größeren Dokumenten des IS-GBE auf (die im Durchschnitt etwa 25 Kilobytes groß sind). Bei kleineren Dokumenten fehlt dagegen das `<head>`-Element in der Regel völlig, in anderen gibt es lediglich das `<html>`- jedoch kein `<body>`-Element, wieder andere beginnen ansatzlos mit dem Text.

Für das bestehende IS-GBE haben diese Unsauberkeiten keine unmittelbare Auswirkung. Das System betrachtet die Dokumente als Strom von Wörtern und ignoriert die HTML-Tags dabei völlig. Zur Darstellung kommen die üblichen HTML-Browser zum Einsatz, die nahezu lückenlos in der Lage sind, die beschriebenen Unsauberkeiten zu übersehen und trotzdem wie gewünscht darzustellen.

Für die folgenden Untersuchungen dagegen ist dieser Zustand jedoch als unerwünscht bis kritisch zu betrachten. Die von Oracle9*i* und Oracle10*g* zur Verfügung gestellten erweiterten Möglichkeiten wie die Sektionierung von Dokumenten und darauf aufbauend zum Beispiel die Suche nach Überschriften oder Bildunterschriften wird dadurch sehr erschwert bis unmöglich gemacht.

3.2 Oracle9*i*/10*g* – Technologien und Möglichkeiten

3.2.1 Oracle und Oracle Text – Historie

Im Juni 1970 veröffentlichte der Mathematiker und Wissenschaftler Dr. Edgar Frank Codd (1923-2003) im Rahmen seiner Forschungsarbeiten im IBM-Forschungslabor in San Jose in Kalifornien seinen wohl epochemachenden Artikel „A Relational Model of Data for Large Shared Data Banks“. Der darin vorgestellte, standardisierte

Formalismus zur Beschreibung relationaler Datenbanken erregte großes Aufsehen. Darauf und auf weiteren Forschungsarbeiten aufbauend entwickelte IBM 1974 „SEQUEL“, die „Structured English Query Language“, die später aus rechtlichen Gründen in „SQL“ umbenannt wurde.

Angeregt durch die Erfolge des von IBM 1977 fertig gestellten Prototyps *„System R“* begannen mehrere Firmen mit der Entwicklung eigener SQL-basierter Datenbanksysteme. Einer Firma namens „Relational Software Inc.“, die später unter dem Namen ihres Produktes bekannt werden sollte, gelang es, noch vor IBM die erste kommerzielle Implementierung des neuen Modells zur Marktreife zu bringen. Das Datenbank-Managementsystem „Oracle“ kam 1979 auf den Markt.

Bis heute ist Oracle unangefochtener Marktführer, obwohl – und auch weil – es trotz der kontinuierlichen Weiterentwicklung stets auch weitestgehend kompatibel zu älteren Versionen bleibt. „Alte“ Oracle8*i*-Applikationen lassen sich im Allgemeinen ansatzlos auch mit Oracle9*i* betreiben. Dass dabei manchmal auch Altlasten und eigentlich unschöne proprietäre Teillösungen bis in die heutige Zeit hineingetragen werden, ist unvermeidlich.

Produktversionen des Oracle Datenbankmanagement-Systems setzen sich aus bis zu fünf Zahlen zusammen, zum Beispiel „9.2.0.1.0“ (die Versionsnummer des für diese Arbeit eingesetzten Testsystems). Die ersten drei Zahlen sind die in der Regel relevantesten und kennzeichnen Versionen mit signifikant neuen Konzepten, neuen Funktionalitäten oder verbesserten und weiterentwickelten Eigenschaften („version number“, „new feature release number“ und „maintenance release number“). Die vierte und fünfte Zahl bezeichnet Fehlerbehebungen auf systemübergreifender oder systemspezifischer Ebene („generic patch set number“ und „platform specific patch set number“). Die von Oracle Corp. gewählten Produktbezeichnungen wie zum Beispiel „Oracle9*i*“ stellen dabei einen verallgemeinerten Überbegriff für alle mit „9“ beginnenden Versionsnummern dar. Wenn im Folgenden also von „Oracle9*i*“ oder auch „9.x“ die Rede ist, ist damit im Allgemeinen die letzte verfügbare Version 9.2.0 mit den jeweils letzten Fehlerbehebungen gemeint.

Das kleine „*i*“ in „Oracle8*i*“ sowie „9*i*“ steht übrigens für „Internet“, da sich die umfassendsten Neuerungen dieser Versionen auf eben diesen Bereich bezogen. Die Bezeichnung „Oracle10*g*“ dagegen soll den relativ jungen Aspekt des „Grid-Computing“ (verteiltes Rechnen) hervorheben. Ob dieser fraglos wichtige Schritt im technologischen Lebenszyklus das Potential hat, sich auch auf den Alltag durchschnittlicher Geschäftskunden auszuwirken, werden die kommenden drei bis fünf Jahre zeigen. So warnte Carly Fiorina, Chefin von HP, vor einer Überschätzung der noch unzureichend standardisierten Vision. „Bislang ist Grid Computing mehr Hype als Realität gewesen“ sagte sie bei ihrem Vortrag im Rahmen der Messe „Oracle-World“.[5]

3.2.2 Was ist Oracle Text?

Oracle Corp. fasst alle Technologien und Eigenschaften, die sich direkt auf Volltextsuche und Information Retrieval im Allgemeinen beziehen, unter einem einprägsamen Namen zusammen. In Version 8.0.x war das der Name „ConText“. Bis zur Version 8.1.5 stand dann „*inter*Media“ für das eigenständige, gesondert zu bezahlende Zusatzprodukt, das neben weiteren auch das Text-Leistungsmerkmal „*inter*Media Text“ umfasste. Seit Oracle 8.1.6 gehört „Text“ jedoch zum Standardumfang, so dass die Produktbezeichnung *inter*Media ihre Gültigkeit verlor. Stattdessen erhielten die „Text“-bezogenen Leistungsmerkmale den allgemeineren Überbegriff „Oracle Text“ (vergleichbar z.B. mit „PL/SQL“, das ebenso wenig ein eigenständiges Produkt darstellt). [TextFAQ]

Die Oracle Text-FAQ (häufig gestellte Fragen) beschreibt es als Teil der Oracle9*i* Standard und Enterprise Editionen. „Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the Web. Oracle Text can perform linguistic analysis on documents; search text using a variety of strategies including keyword searching, ConText queries, Boolean operations, pattern matching, mixed thematic queries, HTML/XML section searching, etc. Oracle Text can render search results in various formats including

---

[5] „HP-Chefin warnt vor Überschätzung des Grid-Computing“. Heise News-Ticker vom 12. September 2003. <http://heise.de/newsticker/data/tol-12.09.03-001/>

unformatted text, HTML with highlighting, and original document format. Oracle Text supports multiple languages including Japanese, Korean, Traditional and Simplified Chinese.“

Die wichtigsten Eigenschaften von Oracle Text sind “content-based retrieval on free text with both literal (word) predicates and thematic predicates. The main features of Oracle Text include: a comprehensive range of operators and index preferences (e.g. Boolean, exact phrase match, proximity, section searching, fuzzy, stemming, wildcard, thesaurus, stopwords, case sensitivity, and search scoring), "about" search, structured search, broad document format support and multi-language support. Other features are classification, catalog indexing, and XML XPath support.” [TextFAQ]

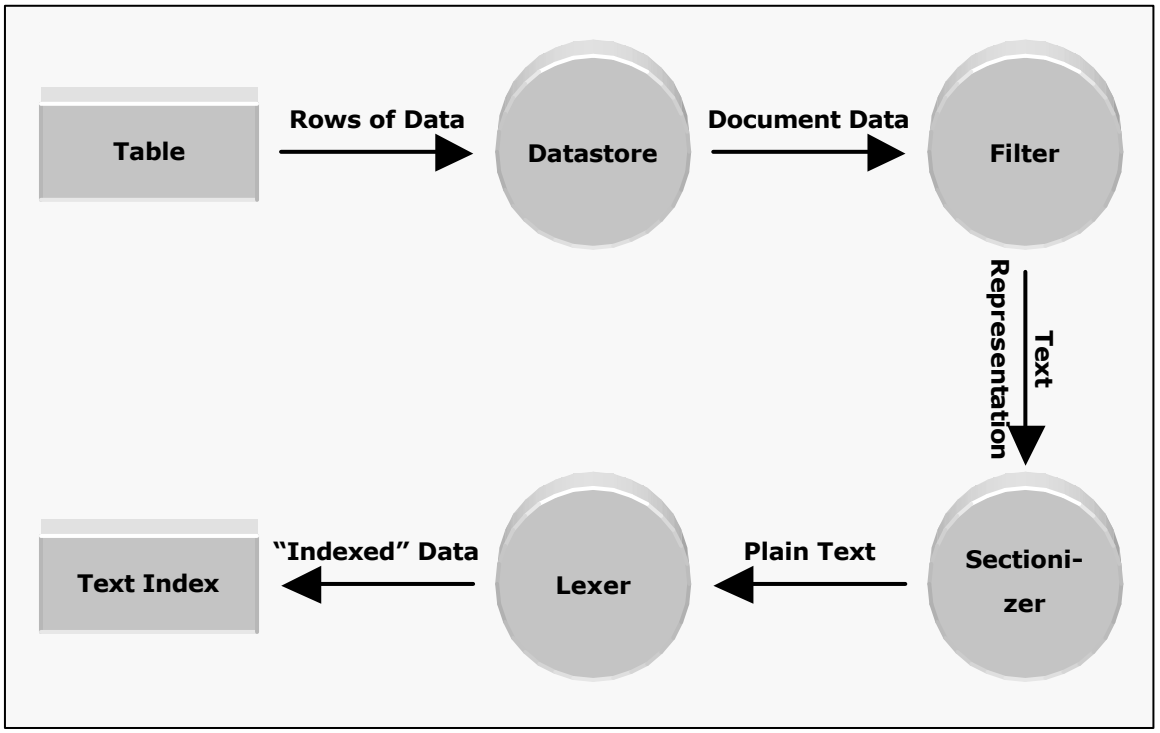

Abbildung 11: Der vierstufige Indizierungs-Vorgang von Oracle Text.

Die Abbildung beschreibt den typischen Indizierungsprozess und damit die Kernaufgaben von Oracle Text. [Scherer01] Aus einer gegebenen Tabelle werden alle Datensätze ausgelesen. Im Allgemeinen wird es sich dabei um eine Textspalte mit gespeicherten HTML- oder ähnlichen Dokumenten handeln. Der „Datastore“ verfügt über das notwendige Wissen, wie genau die Datenquelle einzulesen ist. Das ist insbesondere bei extern abgelegten Daten, mehrspaltigen oder auch vollständig

selbstdefinierten Quellen bedeutsam. Der „Filter“ erhält die unveränderten Daten und wandelt jedes eventuell exotische Dokumentenformat (zum Beispiel PDF) in XML-, HTML- oder Rohtextform um, da die folgenden Schritte nur diese Typen weiterverarbeiten können.

Der „Sectionizer“ zerschneidet diese Daten in Absätze oder Abschnitte und gibt die entstehenden, systematischen Textfragmente weiter. Dabei greift er auf Strukturinformationen zurück, die HTML bzw. XML liefern, so zum Beispiel Überschriften oder Absatzmarken. Erst im allerletzten Schritt extrahiert der „Lexer“ alle im Text vorkommenden Wörter und katalogisiert sie in einer für den Anwender unsichtbaren Indextabelle. Genau wie alle anderen Schritte ist auch dieser von der eingestellten Sprache und anderen Parametern abhängig. So werden im Deutschen Leerzeichen als Abtrennung zwischen Wörtern angesehen (im Gegensatz zu zum Beispiel Japanisch). Als Wortzeichen werden lediglich Buchstaben (inkl. Umlaute), Zahlen, Hochkommas und wahlweise auch Bindestriche anerkannt und alle anderen (Klammern, Satzzeichen etc.) verworfen. Dies ermöglicht zwar später keine Suche nach diesen Zeichen, steigert jedoch die Effizienz.

3.2.3 Was ist neu in Oracle Text 9.0, 9.2 und 10

Im Folgenden soll eine nach den spezifischen Eigenschaften des Systems gegliederte Übersicht über die neuen „Information Retrieval“-relevanten Möglichkeiten von Oracle Text in den Versionen 9.0, 9.2 sowie 10 gegeben werden.

Übersicht

Oracle 9.0 führt eine Dokumenten-Klassifikation ein, die in der Lage ist, eingehende Dokumente anhand ihres Inhaltes zu klassifizieren, das heißt zu filtern bzw. möglichst passend umzuleiten und zu sortieren. Als Beispiel sei eine Nachrichtenagentur genannt, die ihre stetig wachsende Zahl von Artikeln automatisch in Kategorien wie „Politik“, „Kriminalität“ und „Sport“ einordnen möchte. Mit dem ab Version 9.0 eingeführten Indextyp `CTXRULE` sowie dem Operator `MATCHES` erlaubt Oracle Text eine solche Zuordnung. Dabei unterstützt es allerdings lediglich reine Text- sowie XML- und HTML-Dokumente. Version 9.2 erlaubt mittels `CTX_CLS.TRAIN` zusätz-

lich die Erstellung von auf Anfragen basierenden Regeln, anhand deren die Dokumente in spezifische Kategorien gelenkt werden können. Version 10 ergänzt diese Möglichkeit um einen weiteren, „Support Vector Machine“ genannten Klassifikationstyp für ein exakteres, überwachtes Trainieren der zur späteren Klassifikation herangezogenen Dokumentenauswahl (ebenfalls mittels `CTX_CLS.TRAIN`).

Zusätzlich zur Klassifikation auf der Basis von Trainingsdokumenten bietet die neue Methode `CTX_CLS.CLUSTERING` der Version 10 eine Schnittstelle, die das Aufteilen einer vorhandenen Dokumentenmenge in so genannte „Cluster“ erlaubt. Dabei werden die Dokumente vollautomatisch in eine vorher festlegbare Anzahl von Gruppen aufgeteilt, wobei die einander am ähnlichsten Dokumente in je einer zusammengefasst werden. Diese Operation basiert, um sie „echtzeittauglich“ zu machen, auf den Informationen eines vorher angelegten Index, das heißt auf dem Vorkommen bestimmter, den Inhalt des Textes identifizierender Worte.

Die Unterstützung aufgeteilter Indizes ist eine weitere Neuerung der Version 9.0. Ein Index kann somit mittels `CREATE INDEX … LOCAL` auch über lokal aufgeteilte Tabellen, die mit einer `PARTITION BY`-Option erstellt wurden, aufgespannt werden.

Der Indizierungsvorgang selbst lässt seit Version 9.2 weitere performanzsteigernde Neuerungen zu. So können während der Indexerstellung parallel weiter `INSERT`-, `UPDATE`- und `DELETE`-Anfragen ausgeführt werden (Schlüsselwort `ONLINE`). Die für aufgeteilte Indizes eingeführte parallele Indizierung lässt sich mit neuen Parametern der Methoden `CTX_DDL.SYNC_INDEX` und `CTX_DDL.OPTIMIZE_INDEX` auch auf nichtaufgeteilte Tabellen anwenden. Die Wortstammnormalisierung (Stemming) ist seit Version 9.2 nicht nur in den Anfragen (mittels `$`-Operator) sondern auch in der Indexerstellung selbst möglich. Dabei werden sowohl das unveränderte Wort als auch dessen Basisform dem Index hinzugefügt, was die Abfragegeschwindigkeit weiter erhöht.

Eine der wohl signifikantesten Neuerungen erlaubt seit Oracle9*i* die Indizierung von Spalten des neuen Typs `XMLType` (9.0) und des Typs `URIType` (9.2). Die neuen, zu `CONTAINS` gehörenden Operatoren `INPATH` sowie `HASPATH` erlauben die Suche nach spezifischen XML-Pfaden (9.0). Die Prozedur `CTX_DOC.FEATURES` (10) ist mit Hilfe

von vorher mit CTX_DDL.ADD_EXTRACTION_RULE angelegten Regeln in der Lage, eine Reihe von Merkmalen aus einem Dokument zu extrahieren.

Viele kleinere und im Laufe des fortschreitenden Entwicklungsprozesses logische Ergänzungen kamen von Version zu Version hinzu. Die gesteuerte Erkennung des Dokumententyps im Laufe des Indizierungsprozesses erlaubt seit 9.0 auch den Typ „IGNORE“, der die Indizierung einzelner Dokumente völlig unterbinden kann. Die Filter erkennen die „little“ und „big endian“-Variationen von in UTF-16 kodierten Dokumenten vollautomatisch (Version 9.0). Ein zusätzlicher Filter MAIL_FILTER für E-Mails steht bereit (Version 10). Neue Lexer für Koreanisch, Japanisch (9.0) und Chinesisch (9.2) stehen zur Verfügung. Mit USER_LEXER besteht die Möglichkeit zur Definition eigener Lexer für bisher nicht unterstützte Sprachen (9.2). Eine sprachübergreifende Stoppwortliste wurde eingeführt (9.0). Ein CONTEXT-Index erlaubt seit Version 10 wie alle anderen Indextypen auch wahlweise die Synchronisation in Echtzeit (SYNC ON COMMIT), also mit jeder INSERT-, UPDATE- und DELETE-Operation. Dies geschieht selbstverständlich zu Lasten der Performanz und sollte nur mit Bedacht als Alternative zur manuellen Synchronisation mittels CTX_DDL.SYNC_INDEX eingesetzt werden. Mit dem Schlüsselwort METADATA der Anfrage ALTER INDEX kann dieser und alle anderen den Index betreffenden Parameter (FILTER, LEXER, STOPLIST etc.) auch nachträglich verändert werden, ohne den Index unbedingt völlig neu zu erstellen (Version 10).

Im Folgenden soll auf die bereits kurz angerissenen Neuerungen unter Berücksichtigung ihrer Vorzüge, Nachteile und bekannter Probleme im Detail eingegangen werden.

Unterstützte Text-Dokumentenformate (INSO-Filtertechnologie)

Um die INSO-Technologie für Indizierung und DML-Operationen einsetzen zu können, muss das entsprechende INSO_FILTER-Objekt in den Filtereinstellungen gewählt worden sein. Für die einfache Konvertierung mittels des Paketes CTX_DOC ist diese Einstellung nicht notwendig, die Systemumgebung (Pfadangaben etc., d.h. Fragen der Installation) muss jedoch passend konfiguriert sein.

Die wichtigsten unterstützten Formate sind: ASCII/ANSI (7 oder 8 Bit), Unicode, HTML (mit einigen wenigen Einschränkungen), RTF, Microsoft Word, Excel und PowerPoint (bis Version 2002/XP), PDF (bis Version 5.0, inklusive Japanisch) sowie zirka 150 weitere Formate. Eine Sonderstellung nimmt das ebenfalls unterstützte MSG-Format (nur Text) von Microsoft Outlook ein, da es sich um kein tatsächliches Dokumentenformat handelt.

Oracle unterstützt ab Version 9.2.0.2 unter anderem zusätzlich WML (Wireless Markup Language, vgl. „WAP"), PDF in Chinesisch (vereinfacht und traditionell) und Koreanisch sowie nur-lesbares PDF. Verschlüsselte/passwortgeschützte sowie PDF-Dokumente mit eingebetteten Schriftarten ohne Zeichentabelle können dagegen auch von Oracle10*g* nicht indiziert werden.

Weitere unterstützte Dokumentenformate

Mögliche Grafikformate sind unter anderem Bitmap (incl. RLE, OS/2 etc.), Corel Draw (bis Version 9.0), GIF, JPEG (inkl. progressives JPEG), PNG, TIFF und viele andere. Ausführbare Dateien (EXE sowie DLL) sind ebenfalls möglich. „Unterstützung" bedeutet im Fall der Grafikformate wohlgemerkt nicht, dass der INSO-Filter den Dokumenten Textinformationen entnehmen kann, sondern dass das Indizieren dieser Formate keine Fehler auslöst. Der Sinn dahinter ist die Weiterverarbeitung solcher Formate mit anderen Oracle-Technologien (InterMedia). Oracle10*g* bietet hier keine signifikanten Neuerungen.

Filter-Typen

Der `INSO_FILTER` macht nur für binäre Dokumentenformate wie PDF, Word etc. Sinn. Das nachträgliche Filtern von blankem Text, HTML und XML ist wirkungslos und kann deshalb aus Effizienzgründen umgangen werden. Dazu ist für die zu indizierenden Dokumente eine zusätzliche Spalte anzulegen, die entweder den Wert „`BINARY`", „`TEXT`" oder – seit Oracle9*i* – „`IGNORE`" enthält (in allen abweichenden Fällen wird „`BINARY`" als Standard angenommen).

„IGNORE“ sorgt dafür, dass die betreffenden Datensätze vom Indizierungsvorgang gänzlich übersprungen werden. „BINARY“ bezeichnet Datensätze mit binären Dokumentenformaten wie zum Beispiel PDF. „TEXT“ ist für Rohtext, HTML- und XML-Dokumente vorgesehen.

```
CREATE INDEX myIndex ON myTable(myTextColumn)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS('DATASTORE CTXSYS.FILE_DATASTORE
    FILTER CTXSYS.INSO_FILTER
    FORMAT COLUMN myFormatColumn');
```

Abbildung 12: Index-Erstellung mit Format-Informationen

Mit Oracle10*g* hat sich diese Eigenschaft dahingehend verändert, dass Text-, HTML- und XML-Dokumente vollautomatisch unterschieden werden und so erkannt wird, ob ein Filtern Sinn macht oder nicht. Dieser Automatismus kann mit der selben Methode wie bei Oracle9*i* umgangen werden. Zusätzlich bietet Oracle10*g* einen speziellen MAIL_FILTER für gespeicherte E-Mails.

Section Group-Typen

So genannte „Section Groups“ erlauben es, vor allem HTML- und XML-Dokumente nicht nur als Fließtext sondern als gegliederte Dokumente zu betrachten. Texte, die beispielsweise in „<a>…“ eingeschlossen sind, werden gemeinsam mit dieser Eigenschaft indiziert. Im Zusammenhang mit dem Operator WITHIN lassen sich diese Abschnitte später gezielt abfragen (siehe auch 4.6.4).

Lexer-Typen

Der Typ des Lexers hat als einer der letzten Schritte im Indizierungsvorgang primäre Auswirkungen auf vielerlei sprachliche Besonderheiten. Dazu gehört beispielsweise die Handhabung von Groß- und Kleinschreibung, Wortzeichen und Zwischenräumen. Neben dem im Kapitel 5.1 und im Anhang detailliert beschriebenem BASIC_LEXER sind vor allem der für mehrsprachige Dokumentensammlungen bestimmte MULTI_LEXER und der mit Version 9.2 eingeführte USER_LEXER für vollständig selbstdefinierte Sprachen interessant.

WORDLIST-Typen

So genannte `WORDLIST`-Präferenzen erlauben seit Oracle 9.0 das Feintuning von Stemming- und Fuzzy-Operationen sowie die gesonderte Indizierung von Präfixen und Teilzeichenketten. Dies dient in erster Linie der Performanzsteigerung bei rechts- (`'Gesun%'`) und sogar links-beschnittenen (`'%reform'`, `'%ref%'`) Suchanfragen. Besonders letztere sind im Allgemeinen sehr problematisch, da sie den Standardeinstellungen zufolge nicht durch den Index unterstützt werden und stattdessen einen kostenintensiven „full table scan“ nach sich ziehen.

CLASSIFIER-Typen

Die regelbasierte Klassifikation von Dokumentensammlungen mittels der Prozedur `CTX_CLS.TRAIN` erfordert die Festlegung eines Klassifikationstyps, von dem Oracle 9.2 standardmäßig einen (regelbasierten), Oracle 10 zwei („Support Vector Machine“-Klassifikation) zur Verfügung stellt. Oracle 9.0 beherrscht diese Techniken noch nicht.

Alternative Schreibweisen

Deutsch ist (neben z.B. Dänisch und Schwedisch) eine Sprache mit Umlauten, für die es mehr als eine Schreibweise gibt. So können z.B. „ä“ oder „Ü“ sowohl als „a“ und „U“ als auch als „ae“ und „Ue“ geschrieben werden. Die Handhabung solcher alternativen Schreibweisen ist eine Eigenschaft des Lexers und wie folgt zu aktivieren (PL/SQL).

```
CTX_DDL.CREATE_PREFERENCE('myLexer', 'BASIC_LEXER');
CTX_DDL.SET_ATTRIBUTE('myLexer', 'ALTERNATE_SPELLING', 'GERMAN');
CTX_DDL.SET_ATTRIBUTE('myLexer', 'NEW_GERMAN_SPELLING',
  'INDEX_BOTH');
```

Abbildung 13: Nutzerdefinierter Lexer

Oracle wandelt daraufhin während des Indizierungsvorganges sowie in späteren Suchanfragen alle Umlaute in ihre „Basisformen“ um. Als Ergebnis werden Textstellen immer gefunden, egal ob die Schreibweisen in Dokument und Suchanfrage abweichen.

Oracle10*g* (nicht jedoch Beta 2) ist zusätzlich in der Lage, die Regeln der neuen deutschen Rechtschreibung zu behandeln (`NEW_GERMAN_SPELLING`). Worte wie „paßte“ werden entweder nur noch in ihrer neuen Schreibweise „passte“ oder in beiden Formen im Index geführt. Momentan beherrscht diese neue Lexer-Eigenschaft allerdings keine Konventionen, bei denen alte Schreibweisen wie z.B. „soviel“ in mehr als ein Wort (hier: „so viel“) umzuwandeln wären.

Stoppwörter

Oracle wird mit einer vordefinierten Liste von 236 als inhaltslos zu betrachtenden deutschen Stoppworten ausgeliefert, die beispielsweise „ich“, „das“, „mit“, aber auch „dagegen“, „jedesmal“ usw. umfasst. Diese Liste bleibt mit Oracle10*g* unverändert.

Seit Oracle 9.0 besteht die Möglichkeit, eine mehrsprachige Stoppwortliste vom Typ `MULTI_STOPLIST` zu definieren. Dieser können Begriffe beliebig vieler Sprachen hinzugefügt werden, wobei beim Aufruf von `CTX_DDL.ADD_STOPWORD` als neuer dritter Parameter eine gültige Bezeichnung oder Abkürzung der Sprache jedes Stoppwortes übermittelt werden muss. Das Schlüsselwort „`ALL`“ an Stelle der Sprache ermöglicht wahlweise auch das Hinzufügen eines Wortes zu allen Sprachen gleichzeitig.

Dokumenten-Filterung und Highlighting (Paket CTX_DOC)

Die Prozeduren des Paketes `CTX_DOC` bieten zusätzliche, zum Teil sehr weitreichende Funktionalitäten zur Arbeit mit Textdokumenten und einem `CONTEXT`-Index. `CTX_DOC.MARKUP` beispielsweise liefert für ein Dokument und die angewandte `CONTAINS`-Anfrage die exakten Fundstellen inklusive eventuell durch Stemming- oder `ABOUT`-Operatoren entstandener verwandter und normalisierter Wörter. Oracle10*g* bietet mit der Prozedur `CTX_DOC.POLICY_MARKUP` ein Äquivalent, das auch völlig ohne einen vorher angelegten Index funktioniert. Diese neuen Möglichkeiten existieren im Übrigen für nahezu jede Funktion des `CTX_DOC`-Paketes.

`CTX_DOC.HIGHLIGHT` verfolgt das gleiche Ziel, liefert jedoch statt der Positionen einen bereits um wählbare Markierungen erweiterten Text zurück. Dies kann durch

Text- oder HTML-Fragmente geschehen, die alle Fundstellen zum Beispiel farbig hervorheben. Die Prozedur kann dabei nicht für die Wohlgeformtheit[6] des Ergebnisses garantieren, obwohl der Index mit passend gewähltem `FILTER` und `SECTION GROUP` über HTML- oder XML-Tags Bescheid weiß und sie beachtet. Unvorhergesehene Ergebnisse können insbesondere auftreten, wenn der zu markierende Suchbegriff bereits Tags enthält. Oracle10*g* bietet auch hier ein Äquivalent `POLICY_HIGHLIGHT`, das ohne Index arbeitet. Die Prozedur `HIGHLIGHT` selbst blieb unverändert.

`CTX_DOC.GIST` versucht, eine möglichst aussagekräftige Zusammenfassung in Satz- oder Stichpunktform eines längeren Dokumentes zu erstellen. Dabei werden statistische Verfahren angewandt, die unter anderem die Anzahl von Substantiven je Satz in ihre Auswahl einbeziehen. Mit `CTX_DOC.POLICY_GIST` gibt es auch hier ein Oracle10*g*-Äquivalent.

`CTX_DOC.THEMES` greift repräsentative Schlagworte aus dem Text heraus, versucht also, eine Auflistung von Themenbegriffen zu generieren, die den Inhalt des Textes möglichst gut beschreiben. Dabei werden Begriffe, die auch in einer eventuell vorhandenen Wissensbasis auftreten, bevorzugt behandelt.

`CTX_DOC.FILTER` (basierend auf der INSO-Filtertechnologie) erstellt Kopien von Dokumenten in anderen Formaten.

Anfrage-Metadaten (Paket CTX_QUERY)

Die Prozedur `CTX_QUERY.BROWSE_WORDS` liefert eine Auswahl Wörter zurück, die in der während des Indizierungsprozesses angelegten Indextabelle „rund um“ ein gegebenes Wort zu finden sind, in der Regel also Begriffe mit gleichem Wortstamm. `CTX_QUERY.HFEEDBACK` (hierarchische Rückmeldung) ist dem sehr ähnlich, grenzt die Auswahl jedoch auf Phrasen einer vorhandenen Wissensbasis ein (über-, untergeordnete und verwandte Terme). Beides ist nützlich, um dem Anwender zum Beispiel mögliche Tippfehler oder alternative Schreibweisen aufzuzeigen.

---

[6] wohlgeformt: hierarchis che, unzweideutige Verschachtelung

CTX_QUERY.COUNT_HITS zählt lediglich die für eine gegebene Anfrage zurückgelieferten Datensätze. CTX_QUERY.EXPLAIN geht einen Schritt weiter und erläutert detailliert, welche genauen Aktionen (ABOUT-Auflösung, Stoppwörter etc.) eine Anfrage intern auslöst. Alle Methoden dieses Paketes blieben seit Oracle 9.0 unverändert.

Arbeit mit Thesauri (Paket CTX_THES)

Zur Betrachtung und Manipulation von Thesauri bietet das Paket CTX_THES eine ganze Reihe von Funktionen wie zum Beispiel CTX_THES.CREATE_PHRASE zum Hinzufügen neuer Begriffe oder CTX_THES.SYN zum Abruf aller Synonyme einer bestimmten Phrase. Das Kommandozeilen-Werkzeug ctxload („Thesaurus Loader“) bietet die Möglichkeit, ganze Thesauri in Form besonders formatierter Textdateien zu im- und exportieren. Der „Knowledge Base Extension Compiler“ ctxkbtc erlaubt die Generierung oder Erweiterung von Wissensbasen, die Oracle unabhängig vom Thesaurus handhabt, anhand eines oder mehrerer bestehender Thesauri. Diese Prozeduren und Werkzeuge haben sich mit den neusten Oracle-Versionen nicht geändert.

Abfragesprachen

Die standardmäßigen Operatoren = und LIKE (siehe 4.4.1 ff.) haben mit den jüngeren Oracle-Versionen keine Veränderungen erfahren. Auch die deutlich komplexere Sprache des CONTAINS-Operators wurde nur sehr geringfügig erweitert (neuer Operator NEAR_ACCUM in Version 10). So gibt es beispielsweise ABOUT-Querys (siehe 4.6.3) schon seit Oracle 9.0.

3.2.4 Synchronisationverhalten und Performanz

Nach dem Hinzufügen neuer Datensätze ist ein CONTEXT-Index im Gegensatz zu den anderen Indextypen jedes Mal neu zu bilden.

Eine Synchronisation bedeutet – im Gegensatz zu einer vollständigen Neubildung – dass lediglich die Änderungen in den Index übernommen werden. Das optionale Attribut ONLINE legt fest, dass während des Vorganges parallele DML-Operationen

(d.h. SELECTs, INSERTs etc.) erlaubt sind. Ohne diese Angabe würde die Datenbank für den Zeitraum der Indexbildung blockieren, was unter Umständen erwünscht oder nicht erwünscht sein kann. Die CTX_DDL-Routinen arbeiten von sich aus in diesem Online-Modus.

```
-- Empfohlene Möglichkeit:
CTX_DDL.SYNC_INDEX('myIndex');
-- Nicht (!) empfohlene Möglichkeiten (kann hängen bleiben):
ALTER INDEX myIndex
  REBUILD ONLINE PARAMETERS('SYNC');
-- Neuerstellung des gesamten Index:
ALTER INDEX myIndex
  REBUILD;
```

Abbildung 14: Synchronisation/Neubildung eines CONTEXT-Index

Als weitere Möglichkeit zu den bereits genannten kann der Context-Server ctxsrv das transparente Indizieren gelöschter, neuer und geänderter Datensätze in regelmäßigen Abständen übernehmen. Von der Verwendung des noch von Oracle7 stammenden ctxsrv wird allerdings strikt abgeraten. Es soll stattdessen auf die Fähigkeiten von CTX_DDL.SYNC_INDEX und OPTIMIZE_INDEX zurückgegriffen werden. Interessant sind dabei vor allem Kombinationen mit ereignis- oder zeitgesteuerten Verfahren, wie im folgenden Beispiel[7] zu sehen.

```
DECLARE
  v_jobno number;
BEGIN
  dbms_job.submit(v_jobno,
    'ctx_ddl.sync_index(''emp_resume_idx'');',
    interval => 'sysdate + 1/24');
END;
```

Abbildung 15: Zeitgesteuerte Index-Synchronisation

Seit Oracle10*g* ist es möglich, die Synchronisation wahlweise auch vollautomatisch vom Datenbanksystem übernehmen zu lassen. Aus Performanzgründen sollte diese Möglichkeit allerdings nur mit Bedacht Anwendung finden.

---

[7] Naudé, Frank. „Oracle InterMedia FAQ“. 2002. <http://www.orafaq.net/faqctx.htm>

3.3 Untersuchung von Fremdprodukten

Im Laufe der Untersuchungen wurden verschiedene andere Produkte miteinander verglichen und auf ihre Integration in IS-GBE hin geprüft. Für Vergleichszwecke kam neben dem eigenen, vollständig auf Oracle 9.2 aufbauenden Prototyp maßgeblich eine modifizierte Version der von Oracle bereitgestellten Referenz-Implementation „Yapa“ zum Einsatz. Diese PL/SQL- und PSP-Software (PL/SQL Server Pages) bietet ausbaufähige Realisierungen vieler der vorgestellten Oracle-Eigenschaften (Klassifikation, Section Groups etc.).

3.3.1 AIDOS „KAI| BOX“

Das Softwareprodukt KAI| BOX der in Meißen und Berlin ansässigen AIDOS Software AG versucht, eine ganzheitliche, zentralisierte und anwenderfreundliche Lösung für den Zugriff auf immer unüberschaubarere und in der Regel unstrukturierte Bestände von Unternehmens-Informationen zu bieten.

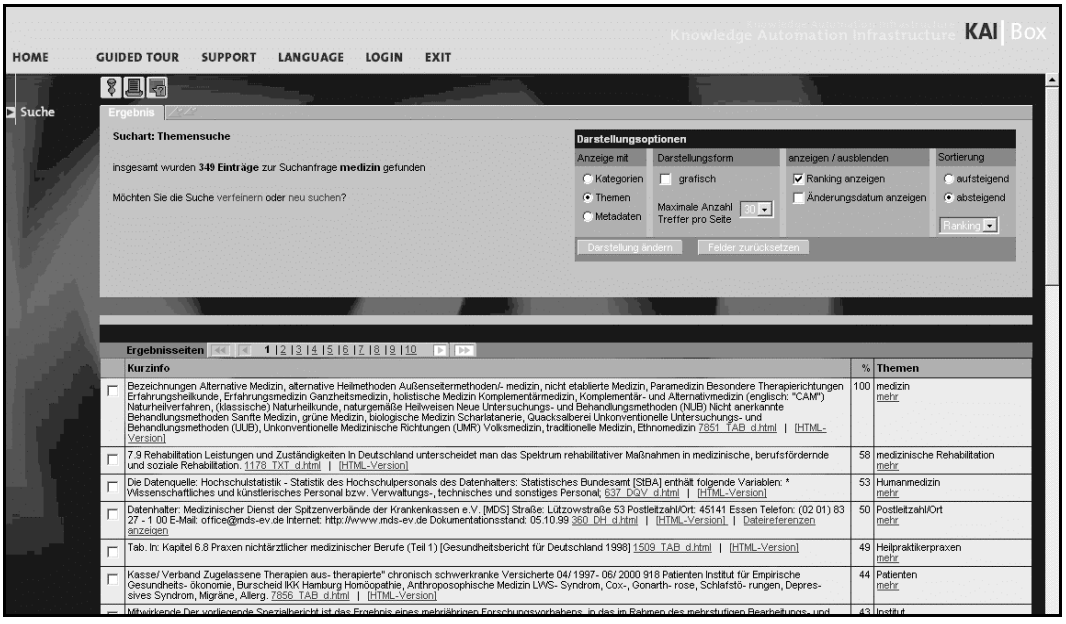

Abbildung 16: AIDOS KAI| BOX

Den Kern dieses Systems bilden die Möglichkeiten zur Erfassung und Archivierung aller Datenquellen. Die Dokumente (über 200 Dokumententypen sind erfassbar) werden nach der Umwandlung in ein Standardformat (XML, Unicode) inhaltlich analysiert. Dabei durchlaufen sie mehrere Stufen der Indizierung: Spracherkennung

(z.B. Deutsch, Englisch); Grundformreduktion/Stemming; Aufbau Wortindex; linguistische Aufbereitung der Informationen (Part-of-Speech Tagging); Statistische Auswertung der Informationen; Themenorientierte automatische Analyse; automatische Metadatenextraktion aus unstrukturierten Informationen (bis zu 30 Felder können extrahiert werden, statische Metadaten sind konfigurierbar); automatische Bildung von Abstracts (einstellbare Länge); Indizierung; Ablage aller Erkennungsmerkmale, Kategorien, Themen und Metadaten und deren Verknüpfung.

Stichprobenvergleiche mit bestehendem IS-GBE

Eine Reihe stichprobenartiger Vergleiche an je einem, mit identischem (Text-) Datenbestand ausgestattetem IS-GBE- sowie KAI| BOX-System zeigte, dass die Qualität (Präzision/Ausbeute) der Suchabfragen bei beiden alles in allem identisch ist. GBE-typische Anfragen nach zum Beispiel „Bevölkerungsentwicklung“ oder „Sterblichkeit“ führen, abgesehen von kleinen, nicht bezifferbaren Abweichungen, zu den selben Ergebnissen. Auch die sogenannte Themensuche kann keine unmittelbar messbare Qualitätssteigerung der Suchergebnisse bewirken. Unterschiede ergeben sich in Bedienung und Ergebnispräsentation durch die zwei verschiedenen Anfragemöglichkeiten der KAI| BOX sowie das beim IS-GBE fehlende Ranking (Rangordnung der Fundstellen nach vermuteter Relevanz).

Besonders verwirrend scheint anfangs der Effekt, dass eine über die Themensuche der KAI| BOX geführte Anfrage mit mehreren Termen deutlich mehr Treffer liefert als die äquivalente, Und-verknüpfte Suchmethode des IS-GBE. Die Ursache liegt in der Nutzung eines akkumulierenden Rankingverfahrens im Gegensatz zu einem ausschließenden Und. Letzteres zeigt Fundstellen, bei denen einer der eingegebenen Begriffe fehlt, nicht mehr an. Dies ist aufgrund des fachlich begrenzten Datenbestandes und der vom Auftraggeber des IS-GBE gewünschten alphabetischen Sortierung der Ergebnisse erwünscht. Das akkumulierende Verfahren der KAI| BOX dagegen wertet die Vorkommen jedes einzelnen Suchbegriffes und summiert diese. Dies kann beispielsweise bewirken, dass ein Dokument, das beide Suchterme enthält, genauso hoch eingestuft wird wie ein anderes, bei dem ein Term auffallend häufig, der zweite jedoch gar nicht vorkommt.

Als entscheidender Unterschied ist die auffällige grafische Ergebnisdarstellung der KAI| BOX zu nennen. Dazu werden die den Dokumenten im Rahmen der Indizierung entnommenen Themenbegriffe (repräsentative, den Inhalt beschreibende Substantive und Phrasen) herangezogen. Diese Themen werden über Ähnlichkeiten im Wortstamm oder häufiges gemeinsames Auftreten in den Texten als hyperbolischer Baum begrenzter Tiefe angeordnet und angezeigt. Die Ergebnisdokumente der ursprünglichen Suchanfrage bilden die Blätter dieses Baumes (wobei ein Dokument unter verschiedenen Themen mehrfach vorkommen kann).

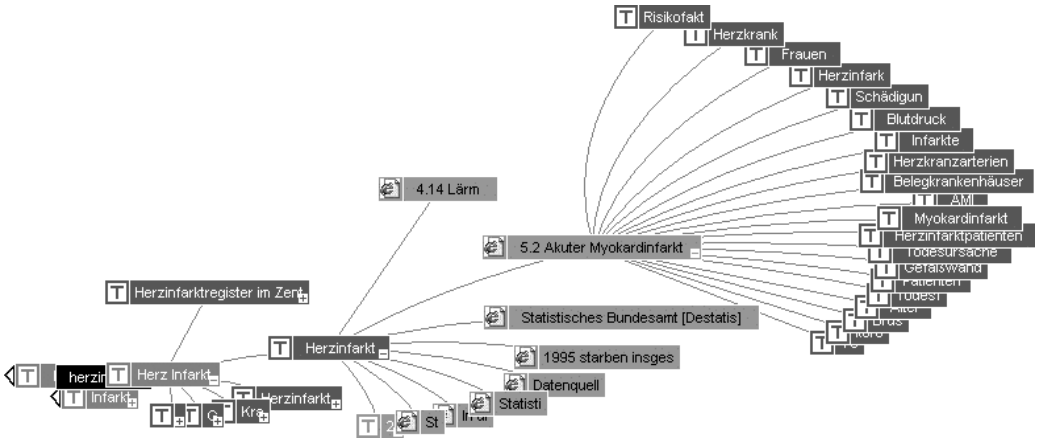

Abbildung 17: KAI| BOX Ergebnisbaum

Die Baumansicht ist trotz ihrer üppigen Darstellung dennoch nichts anderes als eine intuitive, visuelle Einschränkung zu großer Treffermengen über ihre Themen. Die ebenfalls mögliche klassische Listendarstellung liefert nicht nur die selben Ergebnisse, sie stellt diese auch geordnet nach ihrer Relevanz dar. Dieser Ranking-Aspekt fehlt in der Baumdarstellung völlig, weshalb diese für größere Ergebnismengen mit fachlich sehr eng begrenzten und somit wenige Themen beinhaltenden Texten nicht mehr uneingeschränkt sinnvoll erscheint.

Gegenüberstellung KAI| BOX vs. Oracle Text

Der hier gezeigte Vergleich erfolgte in erster Linie anhand der von den Unternehmen zur Verfügung gestellten Prospekte und Dokumentationen (White Papers), aber auch anhand einer KAI| BOX-Testinstallation sowie eigenen Tests mit dem Oracle-System.

Tabelle 2: Oracle Text/KAI| BOX-Gegenüberstellung

	Oracle9*i* Release 2 mit Oracle Text („Oracle's integrated full-text retrieval technology")	KAI| BOX („Web-basierte Infrastruktur für Wissensmanagement")
Daten-/Dokumenten-Quellen	Datenbank (Zeichenketten- und LOB-Felder), Dateisystem, Intranet/Internet-Seiten, nutzerdefiniert (mittels PL/SQL beliebig erweiterbar)	PC (lokale Dokumente, die kontinuierlich bearbeitet werden), Fileserver, Dokumenten- und Content-Management-Systeme, E-Mail, Intranet/Internet-Seiten, externe Datenbanken
Dokumentenformate	Mehr als 150 (Microsoft Office, PDF, HTML, XML, …)	Über 200 (u. a. HTML)
Sprachunterstützung	Englisch (amerikanisches Produkt), gleichwertige (sogar höherwertige) Unterstützung von Deutsch, nahezu jede andere „westliche" Sprache, Multi-Byte-Sprachen (Japanisch, Chinesisch, …)	Deutsch (deutsches Produkt), gleichwertige Unterstützung von Englisch, Französisch usw.
Indizierung	Mehrstufiger Indizierungsprozess mit wortbasiertem, invertiertem Dateiindex	Indizierungsprozess mit invertiertem Dateiindex

Kategorisierung (Taxonomie)	„Classification and Clustering“, Festlegung von Kategorien anhand von Training-Sets, Einordnung neuer Dokumente mittels `MATCHES`-Operator	Manuell zu definierende hierarchische Struktur, Trainingsphase anhand von durch den Knowledge Administrator vorgegebenen Beispieldokumenten (Lernset von Trainingstexten), vollautomatische Zuordnung neuer Dokumente zu je einer Kategorie. Einsatz des „Inxight Categorizers“ der Inxight Software, Inc. (USA)
Thematisierung	Möglich über umfassende, externe PL/SQL-Pakete, Zusammenfassung von Dokumenten in Themenbegriffen, Ähnlichkeitssuche anhand dieser Themen	Analyse des Textes und Zuordnung zu existierenden Themen, automatische Aufnahme neuer Themen in das bestehende Themenmodell. Einsatz von Software der Inxight Software, Inc. (USA)
Abstracts	Möglich über externe PL/SQL-Pakete, wahlweise in Satz- oder Stichpunktform	Vollautomatische Abstractgenerierung, einstellbare Länge, wahlweise in Form einer Liste von Schlüsselphrasen oder in Sätzen, Einsatz des „Inxight Summarizer SDKs“ der Inxight Software, Inc. (USA)
Metadaten	Keine unmittelbare Eigenschaft, unterstütz durch diverse Pakete jedoch unbegrenzt implementierbar	Automatische Metadatenextraktion, Einsatz von Software der Inxight Software, Inc. (USA)

Suchengines	Unbeschränkt über „Ultra Search“ und eigene Implementationen	Klassische Suche (Boolean), thematische Suche, Suche in Kategorien, Suche in Metadaten (parallel anwendbar)
GUI/Benutzungs-Oberfläche	Mit „Ultra Search“ verfügbar, textbasiert (Form der klassischen Suchmaschine)	Zentrales Element der KAI| BOX, klassische Suchmaschinenform oder baumartig („Star Tree“/„Hyperbolic Tree“), zusätzliche Navigationsmöglichkeit innerhalb des Kategorien-Baumes
Benutzerverwaltung	Komplex, nicht unmittelbar vergleichbar	Anwenderprofile, Abonnement bestimmter Dokumente
XML	Interne Ablage in Originalform, Konvertierung zur Indizierzeit sowie zum Export über Filter	Interne Ablage aller Dokumente in normiertem XML, XML für den Datenaustausch
API	Vorhanden	Vorhanden

Anzumerken ist insbesondere, dass einige der fortgeschritteneren Eigenschaften von Oracle Text wie zum Beispiel die Themenfindung auf den selben, lizenzierten Kerntechnologien der Firma Inxight beruht. Dies wird in den Dokumentationen kaum erwähnt. Umfang und Auswirkung dieser Gleichheiten waren damit nicht zuverlässig ermittelbar.

3.3.2 Inxight „SmartDiscovery“

Die Firma Inxight Software, Inc. stellt mit SmartDiscovery ein Vollprodukt bereit, das die hauseigenen Kerntechnologien voll ausschöpft und somit auch als Demonstration für diese angesehen werden kann. Die Parallelen zur KAI| BOX von AIDOS, die ebenfalls einige der Inxight-Technologien einsetzt, sind sehr ausgeprägt. Interes-

sant wird Inxight ganz allgemein auch durch die Tatsache, dass Teile der Inxight-Software von Oracle lizenziert wurden und im Rahmen von Oracle Text zum Einsatz kommen. Weiterführende Lösungen wie der patentierte „Star Tree“ waren zentraler Bestandteil der am Beispiel der KAI| BOX vorgenommenen Untersuchungen, so dass anknüpfend daran ein Blick auf das Referenzprodukt des amerikanischen Herstellers sehr vielversprechend erscheint.

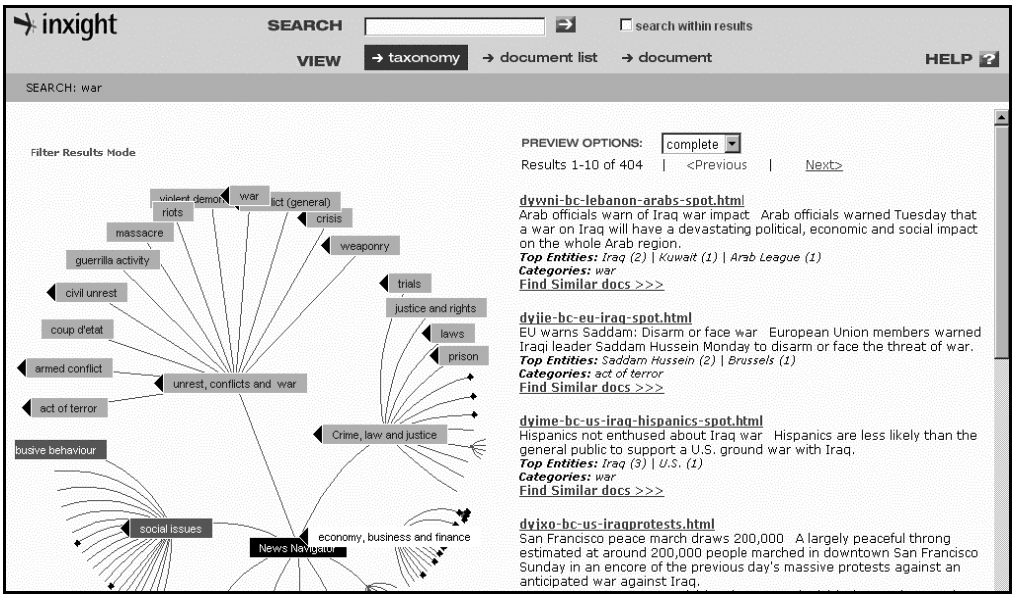

Abbildung 18: Inxight SmartDiscovery

SmartDiscovery ist darauf ausgelegt, unstrukturierten Dokumentensammlungen größtenteils automatisch (Meta-) Informationen zu entnehmen, die Dokumente zu ordnen und sie über eine Web-basierte grafische Oberfläche auf vielerlei Wegen komfortabel recherchierbar zu gestalten. Zur Ermittlung der verwalteten Metadaten werden mehrere Techniken herangezogen. Entitäten wie Firmennamen, Daten, Länder etc. werden regelbasiert aus den Dokumenten ermittelt. Eine taxonomische Kategorisierung erlaubt über eine komfortable Verwaltungsoberfläche die Erstellung von Begriffsbeziehungen unter Berücksichtigung vieler möglicher Regeln (z.B. anhand repräsentativer Phrasen oder Beispieldokumente) und die Zuordnung der Dokumente zu diesen Kategorien. Eine automatische Extraktion von Konzepten (Schlüsselphrasen eines Textes) findet ebenfalls statt. Allerdings gibt es keinen globalen Themenbaum wie bei der KAI| BOX, sondern dieser wird für jede Suchanfrage gesondert generiert. Eine Ähnlichkeitssuche und die automatische Bildung von Zusammenfassungen (Abstracts) vervollständigen das Produkt.

Leider war es im Rahmen dieser Arbeit nicht möglich, über das Maß der von Inxight bereitgestellten White-Papers hinausgehende Untersuchungen durchzuführen, da die Firma auf entsprechende Anfragen bisher nicht reagierte. Es erscheint nichts desto trotz lohnenswert, dies weiterzuverfolgen, um das Potential sowohl des Vollproduktes „SmartDiscovery“ als auch der Kerntechnologien (OEM-Software) wie zum Beispiel „Star Tree“ für zukünftige Kundenanfragen abschätzen zu können. Insbesondere letzteres bietet quasi unbegrenzte und oftmals auch stark verkaufsfördernde Möglichkeiten zur Darstellung von Suchergebnissen und Baumstrukturen jeglicher Art.

4 Konzeption

Aufbauend auf den behandelten theoretischen Grundlagen und den durchgeführten Analysen sollen im Folgenden konkrete, mögliche Implementations-Details im Hinblick auf die Aufgabenstellung betrachtet werden.

4.1 Anforderungen

Der Wunsch nach Verbesserungen des Systems IS-GBE wurde im Vorfeld dieser Arbeit kurz und prägnant als „Qualifizierung der Textrecherche“ charakterisiert. Die Aufgabenstellung „Knowledge Base gestützte Verwaltung und Recherche von Textdokumenten“ spiegelt wieder, dass neben anderen vor allem den Aspekten von Thesauri und Wissensbasen besondere Erwartungen entgegengebracht wurden.

Im Leistungsmerkmal SR-008 des Erstellungsvertrages zum IS-GBE heißt es: „Prüfung der Möglichkeit zur Einbindung von Retrieval-Produkten anderer Hersteller. Stichworte: Wissensnetze, Ontologien, med. Thesauri“.

Der Nutzen des Systems soll möglichst erhöht werden, und zwar durch:

- redundanzfreiere Beantwortung von Anfragen,
- bessere Entscheidbarkeit, ob ein Dokument „passend“ ist,
- mehr Zusatzinformationen zu den Dokumenten,
- bessere Beziehungen untereinander.

Das Ziel ist dabei der exemplarische Test und die Bewertung von Möglichkeiten zur ontologischen Anreicherung der Thesauri sowie die Einschätzung und der konkrete Einsatz der erweiterten Möglichkeiten von Oracle9*i* und eventuell auch 10*g*.

4.2 Ausbaumöglichkeiten, Soll-Zustand

Wie in [Dubrow03] („Perspektiven“, „Oracle9*i* Text Features“) beschrieben, sind auf Basis der erweiterten Eigenschaften von Oracle9*i* sowie 10*g* diverse kleine Verbesserungen und Ausbaustufen bis hin zu einer vollständigen Umstellung des IS-GBE auf Java und XML denkbar.

Unter der Annahme, dass XML-Dokumente bereits so bedacht strukturiert in die (objektrelationale) Datenbank eingepflegt werden, dass sie von Oracle9*i* bzw. 10*g* vollständig ausgewertet, indiziert etc. werden können, wäre Folgendes denkbar: Direkter Zugriff auf einzelne Struktureinheiten des Dokumentes ohne zusätzliche Hilfsmittel, mit anderen Worten ein direktes Hineingreifen in die XML-Struktur mit Hilfe der XPath-Adressierungssprache. Im Resultat wären Anfragen möglich, die beispielsweise alle Dokumente nach der Anzahl ihrer Absätze ordnen, Dokumente mit bestimmten Referenzen im Literaturverzeichnis herausfiltern oder alles finden, was mehr als zehn Bildunterschriften enthält.

Weiterhin ist, wie schon in [Dubrow03] ausgeführt, die Ausweitung der flachen Thesauri zu vollwertigen, hierarchischen Thesauri und Wissensbasen im Sinne von Oracle möglich. Neben Anfragen nach Synonymen wäre dann unter anderem auch die Suche nach sinnverwandten Begriffen realisierbar. Zur Erfassung dieser zusätzlichen Informationen können neben existierenden medizinischen Thesauri gängige Softwarekomponenten zur Arbeit mit Thesauri zum Einsatz kommen. Auf die Einfachheit und Anwenderfreundlichkeit dieser Module ist besonders Wert zu legen, um dem fachlich gebildeten Bearbeiter bei der Formalisierung seines Wissensgebietes so weit wie möglich entgegen zu kommen. Auch Methoden zur Erweiterung dieses Wissens mit Hilfe der Endnutzer wären denkbar, im Rahmen der Konzeption des IS-GBE jedoch nicht unbedingt praktikabel.

Werkzeuge zur Dokumentenerfassung

Um Textdokumente ganz allgemein durchsuchbar zu gestalten, ist zuerst eine digitale Erfassung derselben vonnöten. Im bestehenden IS-GBE geschah und geschieht das mit Hilfe der Standardsoftware Microsoft Word 97. Eine Aufbereitung dieser Quelle in das für den Interneteinsatz bestimmte Zielformat HTML ist mit den von Word mitgelieferten Mitteln zwar auf den ersten Blick ausreichend, jedoch nur sehr eingeschränkt möglich (siehe 3.1.1).

Neuere Versionen der Microsoft-Standardsoftware (Word 2000, 2003) erzeugen deutlich maschinen-freundlichere HTML-Daten, so dass eine Aktualisierung auf diese bereits messbare Verbesserungen mit sehr geringem Umstellungs- und

Lernaufwand mit sich bringen kann. Dennoch erfordert der „richtige“ Umgang mit der für derartige Aufgaben nicht ausgelegten Textverarbeitung nach wie vor sehr viel Disziplin vom Anwender. Der konsequente Umgang mit Dokumentenvorlagen (Stylesheets) wäre für eine optimale maschinelle Verarbeitung unbedingt notwendig, wird jedoch von Word so gut wie nicht unterstützt. Die dabei unweigerlich auftretenden Inkonsistenzen lassen sich längerfristig nur durch speziell dafür ausgelegte Werkzeuge wie beispielsweise Altovas „Authentic“ vermeiden.

Bei Authentic handelt es sich um eine stark abgespeckte Variante des ebenfalls von Altova stammenden Editors XMLSPY. Im Gegensatz zu diesem verzichtet Authentic auf nahezu alle Eigenschaften, die über die reine Erfassung von Dokumenten hinausgehen. Dies erlaubt eine sehr komfortable Bearbeitung, ohne dabei von Eigenschaften und Eigenheiten des Systems abgelenkt zu werden, die bei diesem Arbeitsschritt gar nicht vonnöten sind. Überdies existiert Authentic auch als ActiveX-Control und lässt sich auf diesem Wege und mit Hilfe von ActiveX-fähigen Webbrowser wie dem Internet Explorer oder Mozilla direkt in Internetanwendungen integrieren. Dem Anwender ist es so möglich, beinahe unabhängig vom Standort zwar stark reglementiert, in diesem Rahmen aber dennoch äußerst komfortabel wohlgeformte und geprüfte, gültige[8] XML-Dokumente zu erfassen.

Weitere, Authentic zum Teil unterlegene Lösungen sind beispielsweise der Morphon XML-Editor (http://www.morphon.com/) oder VEX (http://vex.sourceforge.net/). Alle erlauben die grafisch vollwertige (WYSIWYG) Erfassung von XML-Dokumenten beliebiger Art.

Während Textverarbeitungssysteme wie Microsoft Word den in Kapitel 2.2 spezifizierten Begriffen folgend lediglich eine Daten-Erfassung zulassen (und nur in sehr begrenztem Maße mit Informationen hantieren), bewegen sich XML-Editoren wie die genannten fast gänzlich auf der Ebene der Informationen. Die vollständige Trennung zwischen der Spezifizierung einer Dokumenten-Beschreibungssprache und ihrer grafischen Präsentation sowie der eigentlichen Text- bzw. Dokumenten-

[8] gültig: nur den per DTD oder Schema vorgegebenen Sprachumfang beinhaltend

erfassung hebt die genannten XML-Lösungen deutlich von herkömmlichen Textverarbeitungen ab. Die von ihnen erfassten Daten sind – eine adäquate Spezifikation der verwendeten XML-Sprache vorausgesetzt – automatisch mit einem Bedeutungsinhalt behaftet.

Eine Zwischenlösung kann die Konvertierung verschiedener Formate untereinander sein, wobei auch hier anzumerken ist, dass XML-Quellen massive Vorteile für sich verbuchen können. Bestehende HTML-Dokumente können mit dem Werkzeug „Tidy“ (sinngemäß: Säuberer) des W3C vollautomatisch in XHTML- und damit auch XML-Form umgewandelt werden. Die Software ist gerade für mit Word entstandene Dateien geeignet und bietet einen besonderen Konvertierungsmodus dafür an. Die tatsächliche Eignung für den kompletten Bestand des IS-GBE sollte hierfür näher untersucht werden.

4.3 Soziale Ebene

Alle Probleme im Zusammenhang mit Computern müssen in erster Linie auf einer sozialen Ebene gelöst werden und nicht auf technologischem Weg. [Steenis92] Die Behandlung von Symptomen wird gerade heute leider noch all zu oft einer meist deutlich nutzbringenderen Analyse der Ursachen vorgezogen.

Bezogen auf das hier zu behandelnde Gebiet des Informationsmanagements heißt das in erster Linie, dass zum Beispiel Verbesserungen in den Ergebnissen von Suchanfragen ab einem gewissen Punkt nicht mehr realisierbar sind, ohne grundlegende Eingriffe im bestehenden System vorzunehmen. In einer Bibliothek beispielsweise würde ein neues Verwaltungssystem keinen Nutzen bringen, wenn es sich nicht an die gängigen Arbeitsprozesse der Mitarbeiter anpasst und umgekehrt die Mitarbeiter bereit sind, ihre gewohnten Arbeitsweisen zu ändern. Es ist also nötig, Verständnis aufzubringen für den Menschen und die Situation, in der er lebt und in der er den Computer und die Anwendung einsetzen möchte. Zwischen technischer Machbarkeit und Akzeptanz beim Anwender klafft eine Lücke, die nur allzu gern übersehen wird.

In den folgenden Kapiteln und der abschließenden Zusammenfassung wird die grundsätzliche Problematik, technische Eingriffe in bestehenden Systeme immer

auch auf sozialer Ebene betrachten und diskutieren zu müssen, immer wieder eine sehr wichtige Rolle spielen.

Bemerkenswert ist in diesem Zusammenhang insbesondere ein Zitat aus dem Vorwort von [Mresse84]: „[Es existiert] kaum ein Computeranwender, der nicht ein mehr oder weniger leistungsfähiges System zur Bewältigung seiner IR-Aufgaben erarbeitet hat." Offenbar wurde in der Zeit um 1980 davon ausgegangen, dass „Computeranwender" mit Programmierern bzw. Softwareentwicklern gleichzusetzen sind. Dies mag typisch gewesen sein, grenzt aber gleichzeitig diese Arbeit stark von dem ab, womit sich die Publikationen der Zeit beschäftigten. Heute ist und sollte kein Anwender gezwungen sein, entwickelnd tätig zu sein. Die Grundlagen sind längst gelegt. Implementations-Details und Fragen der Performanz sind dank schneller Prozessoren, verfügbarem Speicherplatz und fortgeschrittener Datenbanksysteme weitestgehend nebensächlich geworden. Der Entwickler kann es sich heute leisten, den Fokus auf durchdachte, anwendungsbezogene und vor allem intuitive Bedienkonzepte zu legen. Der Schritt bis hin zur Möglichkeit, verbal mit der Maschine zu kommunizieren und sie einfach nach den gesuchten Informationen zu fragen, scheint nicht mehr weit. Und doch ist die Technik nach wie vor kaum in der Lage, über ein simples „Zeige alle Dokumente, in denen der Suchbegriff vorkommt" hinaus zu gehen. Diese und weitere Details sollen im Folgenden eingehender beleuchtet werden.

4.4 Vergleichsoperatoren ohne Index

4.4.1 Operator „="

Der Gleich-Operator stellt die elementarste Möglichkeit dar, Tabellen unter Zuhilfenahme von Indizes nach bestimmten Werten zu durchsuchen. Anfragen wie „`SELECT * FROM personen WHERE nachname = 'Müller'`" finden dabei ausschließlich Datensätze, bei denen der vorgenommene Vergleich Zeichen für Zeichen (bzw. Byte für Byte) übereinstimmt. Das heißt auch, dass ein solcher Vergleich die Groß-/Kleinschreibung zwingend beachtet („Case-Sensitivität"). Der Typ der abgefragten Spalte ist dabei unerheblich – im Gegensatz zu anderen DBMS, die die Unterscheidung von Zeichen- und Binärtypen auf ein automatisches Ignorieren der

Großschreibung ausweiten. Die Funktion `UPPER()` erlaubt auch bei Oracle die Umgehung dieser Restriktion, zum Beispiel `UPPER(name) = 'JAN'` oder auch `UPPER(name) = UPPER('Jan')`.

Mit Hilfe von Indizes lassen sich Anfragen dieser Art hervorragend beschleunigen. Das Datenbanksystem erstellt dabei standardmäßig so genannte B-Tree-Indizes (daneben gibt es weitere Sonderformen). Ein diesem Index zugrunde liegender Binärbaum teilt die Menge der Datensätze immer weiter in zwei Teile, so dass an jedem Punkt des Baumes nur zwischen zwei Möglichkeiten entschieden werden muss. Beispielsweise benötigt eine Suche in 100 Namen im schlechtesten Fall dann nicht mehr 100 sondern nur noch 7 Schritte bis zum Auffinden des gewünschten Datensatzes (ca. $\log_2$ von 100).

Der Operator `=` kommt in der Regel in Kombination mit den logischen Verknüpfungen `AND`, `OR` und `NOT` sowie eventueller Klammerung zum Einsatz. Die Anfrage „`WHERE (alter > 18 AND alter < 65)`“ liefert beispielsweise alle Datensätze zurück, auf die beide Bedingungen zutreffen (also Altersangaben von 19 bis 64 Jahre). Runde Klammern bestimmen explizit, in welcher Reihenfolge die Bedingungen auszuwerten sind. Eine einfache Regel besagt, dass es nie schaden kann, Klammern zu verwenden, anstatt sich auf die interne Operator-Rangfolge zu verlassen.

Für Fragen der Volltextsuche, die über Beispiele wie eine Liste von Nachnamen hinausgehen, ist der Operator = grundsätzlich nicht gedacht oder geeignet.

4.4.2 Operator „LIKE“

Der auch von anderen Datenbanksystemen bekannte Operator `LIKE` erweitert die Suche nach Textmustern um Platzhalterzeichen, die für unbekannte Suchmuster stehen können. Grundsätzlich ist ein Vergleich der Art `spalte LIKE 'Suchbegriff'` ohne diese Platzhalter mit einem entsprechenden `spalte = 'Suchbegriff'` identisch und liefert auch die selben Ergebnisse.

Im Unterschied dazu erlauben die Platzhalter, die so genannten Wildcards, eine deutlich mächtigere Mustersuche. Die Zeichen _ (Unterstrich) und % stehen für ein einzelnes unbekanntes bzw. beliebig viele unbekannte Zeichen. Ein typischer Anwendungsfall wäre die Suche in einer Namensspalte nach Personen, deren Name mit „M“ beginnt: `WHERE name LIKE 'M%'`. Dabei kommt auch ein bestehender Index zum Einsatz, so dass die `LIKE`-Anfrage in diesem Beispiel nicht langsamer als ein entsprechendes `=` ist. Ein negatives Anwendungsbeispiel wäre `WHERE text LIKE '%Wort%'`, da dabei der gesamte Datenbestand von möglicherweise mehreren Duzend Megabytes durchsucht werden muss. Ein beschleunigender Index kann bei mit Platzhaltern beginnenden Anfragen wie der gezeigten nicht mehr zum Einsatz kommen.

Der Operator `LIKE` ist nur auf Zeichenketten-Datentypen wie `VARCHAR2` und `CLOB` (Zeichenkettenobjekt) sinnvoll anwendbar, nicht jedoch auf Binärobjekte (`BLOB`). Beim Versuch, `LIKE` auf diese anzuwenden, zeigt die Oracle-Datenbank die Meldung „Nicht übereinstimmende Datentypen“.

Kombination mit den logischen Operatoren `AND`, `OR`, `NOT` und der Klammerung sind im Zusammenhang mit LIKE ebenso möglich wie bei =.

4.4.3 Funktionen des Pakets DBMS_LOB

Die Methoden dieses PL/SQL-Pakages sind, wie der Name bereits aussagt, auf die LOB-Datentypen spezialisiert. Für die Suche ist insbesondere die Methode `INSTR` interessant, die das Vorhandensein eines Teilstrings überprüft. Die Funktionalität ist mit dem bereits beschriebenen `LIKE` mit `%`-Platzhalter an Anfang und Ende des Suchmusters prinzipiell völlig identisch (zum Beispiel `LIKE '%Suche%'` und `DBMS_LOB.INSTR('Suche')`). Auch das Verhalten im Bezug auf Groß-/Kleinschreibung unterscheidet sich nicht. Im praktischen Einsatz stellt sich jedoch heraus, dass nur die Anwendung auf `CLOB`-, nicht jedoch auf `BLOB`-Spalten sinnvoll funktioniert. Letztere benötigen ein Hexadezimalformat der Art `'FAFF'`, um wie gewünscht arbeiten zu können.

Im Vergleich zur LIKE-Lösung kann DBMS_LOB.INSTR keinen signifikant messbaren Vorsprung in Sachen Geschwindigkeit für sich verbuchen. In den durchgeführten Tests liegen die Geschwindigkeits-Unterschiede unterhalb der Störgrenze, die durch andere Systemeigenschaften bedingt werden. Sofern die Funktionalität echter Suchmuster mit eingebetteten Platzhaltern nicht benötigt wird, grenzt sich INSTR lediglich durch syntaktische Unterschiede vom äquivalenten LIKE ab, sowie durch seine eigentliche Funktion, die exakte Zeichenposition der Fundstelle zurückzuliefern.

4.5 Indizierungsverfahren

CTXCAT (neu seit Oracle8*i* Release 3, Version 8.1.7) ist ein kombinierter Index über eine Textspalte und eine oder mehrere weitere Spalten. Abfragen lassen sich hier mit dem CATSEARCH-Operator im WHERE-Teil einer Anfrage formulieren. Dieser Indextyp ist für gemischte Anfragen (Kombination mit verschiedenartigen Operatoren) optimiert und aktualisiert sich mit jeder Änderung in der Basistabelle selbst.

Ein CTXRULE-Index überspannt eine Spalte, die einen Satz von Anfragen enthält, welche Klassen von Dokumenten definieren. Die Prozedur CTX_CLS.TRAIN hilft bei der Erstellung dieser Regeln. Mittels des MATCHES-Operators im WHERE-Teil einer SELECT-Anfrage lässt sich dieser Index dann abfragen. Damit ist es möglich, Klassen von Dokumenten zu bestimmen und neu hinzukommende über MATCHES automatisch zu klassifizieren und einzuordnen. Oracle erlaubt diese Operationen jedoch nur mit Text-, HTML sowie XML-Dokumenten (auch Oracle10*g*).

Der Indextyp CTXXPATH ist dann sinnvoll zu verwenden, wenn Zugriffe auf XMLType-Spalten mittels ExistsNode() (und der XML-Adressierungssprache XPath) beschleunigt werden sollen.

Die CTXCAT-, CTXRULE- und CTXPATH-Indizes werden vollautomatisch neu gebildet bzw. synchronisiert, wenn Änderungen auftreten. Das Oracle-Handbuch äußert sich an dieser Stelle wie folgt: „Oracle automatically reflects changes to data, such as adding new rows, updating rows, or deleting rows, in all relevant indexes with no additional action by users.“ Dass die für diese Arbeit so wichtigen CONTEXT-Indizes

eine Ausnahme dieser Regel darstellen, kann für Verwirrung sorgen, hat jedoch seinen Grund. Im folgenden Kapitel wird darauf näher eingegangen.

4.6 Textbasierende „CONTEXT“-Indizierung

Der Indextyp `CONTEXT` umfasst eine Textspalte und lässt sich mit dem `CONTAINS`-Operator im `WHERE`-Teil einer Anfrage nutzen. Um Änderungen der Datenbank auf den Index zu übernehmen, muss dieser manuell mittels `CTX_DDL.SYNC_INDEX` synchronisiert werden (siehe hierzu auch Kapitel 3.2.4).

Eine vollautomatische Synchronisierung, das heißt Aktualisierung des Index ist standardmäßig nicht vorgesehen und auch nicht sinnvoll, da dies viel zu viele Ressourcen fordert. Es würde, um es mit den Worten von Oracle selbst zu formulieren, zu viele Kosten verursachen. Üblich ist, einen Index entweder sehr selten komplett neu zu erstellen (siehe `REBUILD`) oder ihn lediglich den Änderungen entsprechend zu synchronisieren (siehe `SYNC_INDEX`). Beides ist auf verschiedene Weisen und auch kombiniert möglich. Da die Synchronisation vor allem Kosten sparend ausgelegt ist, ist sie mit einem gewissen Grad an Fragmentierung verbunden, die wiederum über eine vollständige Neubildung des Index kompensierbar ist.

4.6.1 Logische und gewichtende Operatoren

Der Operator `AND` (oder `&`) verknüpft innerhalb der `CONTAINS`-Anfrage zwei Suchbegriffe logisch miteinander. Fundstellen, bei denen einer der Begriffe fehlt, werden so gänzlich ausgeblendet. `OR` (oder `|`) ist als logisches Oder das Gegenstück hierzu. `NOT` (oder `~`) kehrt die Bedeutung des nachfolgenden Terms um, schließt also Dokumente mit bestimmte Suchbegriffen explizit aus. `ACCUM` oder `,` (Komma) steht für „Accumulate“ und stellt im Prinzip ein OR dar, bei dem die Berechnung des `SCORE` anders verläuft. `EQUIV` oder `=` bestimmt akzeptierbare Substitutionen für ein Wort. `MINUS` oder `-` reduziert den `SCORE`, wenn der ausgeschlossene Term gefunden wurde. Im Gegensatz zu `NOT` wird diese Fundstelle jedoch nicht völlig ausgeschlossen. `NEAR((a, a))` oder in der älteren Syntax `a NEAR a` oder `a ; a` erhöht den SCORE der Fundstelle, wenn die Suchbegriffe besonders nahe beieinander gefunden wurden.

Gewichtung

Der sogenannte „Threshold-Operator“ `A > 1` („größer als“) erlaubt es, Suchergebnisse auszuschließen, die eine bestimmte Wertung (`SCORE`) unterschreiten. `A * 1` stellt ganz ähnlich dazu eine Gewichtung einzelner Wertungen um beliebige Faktoren bereit. Weitere die Berechnung des `SCORE` betreffende Möglichkeiten stehen mit den weiter oben bereits beschriebenen logischen Operatoren `ACCUM`, `MINUS` etc. zur Verfügung.

Klammerung/Operatorrangfolge

Oracle arbeitet mit einer impliziten Operatorrangfolge, die den Regeln herkömmlicher Programmiersprachen folgt. Mit Hilfe runder Klammern lassen sich Teile der Anfrage gruppieren und so auch in eine andere Reihenfolge bringen.

Geschweifte Klammern erlauben das Maskieren („Escapen“) von Operatoren, um auch Suchbegriffe verwenden zu können, die ansonsten eine Sonderbedeutung hätten, zum Beispiel `CONTAINS(… 'red {and} blue')` oder auch `CONTAINS(… '{red and blue}')`. Die Suche nach Sonderzeichen oder Worten mit Sonderzeichen ist aufgrund des zugrunde liegenden Index-Konzeptes nicht oder abhängig von der Konfiguration des Lexers nur sehr begrenzt möglich, so dass ein Maskieren derselben in der Regel nichts bewirkt.

4.6.2 Operatoren zur Ähnlichkeitssuche

Äquivalent zum bereits beschriebenen `LIKE` erlauben `CONTAINS`-Anfragen die bekannten Platzhalter `%` für Null bis unendlich viele beliebige Zeichen sowie _ (Unterstrich) für genau ein beliebiges Zeichen.

Soundex-Operator

Der auch als eigenständige Funktion `SOUNDEX()` existierende Operator `!` (Ausrufezeichen) direkt vor einem Suchbegriff verfolgt das Ziel, Worte zu finden, die ähnlich klingen, selbst wenn die genaue Schreibweise unbekannt ist. Gedacht war die unter 2.7.1 näher beschriebene Methode ursprünglich für die normierte Suche nach Namen

im Rahmen der amerikanischen Volkszählungen. Der historische Algorithmus, der auch in Oracle Text Verwendung findet (verfeinerte, unter den Begriffen „Refined“ oder „Extended Soundex“ bekannte Varianten werden nicht unterstützt), ist aufgrund dieser hohen Spezialisierung für qualitativ hochwertige Informationsrecherchen kaum geeignet. Oracle selbst rät zudem ganz allgemein von der Verwendung in anderen Sprachen als Englisch ab.

Fuzzy-Operator

Der Operator `FUZZY` (oder `?` in der alten, aus Kompatibilitätsgründen beibehaltenen Syntax) vor einem Suchbegriff ist ähnliche wie Soundex darauf ausgelegt, Wörter anhand ihres „Klanges“ zu normieren und so zum Beispiel trotz typischer Tippfehler im Suchbegriff oder auch in den Dokumenten mehr relevante Fundstellen zu liefern. Neben einigen zusätzlichen, optionalen Parametern zeichnet sich die neue Schreibweise `FUZZY(…)` vor allem durch ihre Beeinflussung der Termgewichtung aus, die bei `?` noch fehlte. Je mehr sich Fundstellen von der eigentlich geforderten Schreibweise unterscheiden, umso geringer wird ihr zurück gelieferter `SCORE`. Die Grenzwerte hierfür sind wahlweise parametrisierbar. Die Eignung des Operators für die deutsche Sprache wird ausdrücklich hervorgehoben – das Oracle-System arbeitet mit eigenen Regelsätzen für jede der unterstützten Sprachen.

Stammerweiterung (mit $)

Der Operator zur Stammerweiterung („Stemming“) sorgt bei der Eingabe von zum Beispiel `'$Gesundheiten'` dafür, dass die Suche Wortformen der Mehrzahl, Beugungen und Ähnliches ignoriert. Das heißt, Wortenden usw. werden ausgeschlossen. Abhängig von der Konfiguration des Lexers arbeitet dieser Operator auch für Deutsch sehr zufriedenstellend. Als Sonderform ist die Suche nach Teilen zusammengesetzter Wörter zu nennen, für die obige Suche beispielsweise auch „Gesundheitsreform“ auffinden kann. Dazu ist das entsprechende `COMPOSITE`-Attribut des Lexers zu aktivieren.

4.6.3 ABOUT

Der ABOUT-Operator bzw. die Funktion kann die Zahl der gefundenen Dokumente erhöhen. Dazu wird die Suche nach einem Begriff mit Hilfe einer bestehenden Wissensbasis auf Synonyme, über-, untergeordnete und verwandte Begriffe ausgeweitet. So liefert die Suche nach 'ABOUT(colors)' auch Dokumente zurück, in denen die Begriffe „red“ oder „yellow“ vorkommen, selbst wenn das Wort „colors“ gar nicht unmittelbar gefunden wurde.

Eine ABOUT-Anfrage lässt sich nur in Verbindung mit einem CONTEXT-Index und dem zugehörigen CONTAINS-Operator einsetzen, sofern für den erstellten Index die Indizierung von Themen aktiviert wurde und eine der Sprache entsprechende Wissensbasis vorhanden ist. Oracle liefert zwei bereits vorgefertigte Wissensbasen („Knowledge Bases“) für Englisch und Französisch, die in sechs Haupt- und einige tausend Unterkategorien gegliedert sind. Diese hierarchische Grundstruktur ermöglicht sozusagen einen „allumfassenden Blick auf die Welt“, von Wissenschaft über Wirtschaft, Politik, Soziales, Geografie bis hin zu abstrakten Ideen und Konzepten. Bei diesen Wissensbasen handelt es sich um einfache Ontologien bzw. Klassifikationssysteme.

Eigene Wissensbasen für weitere Sprachen können eingeführt werden, sofern die Sprache auf einem 1-Byte-Zeichensatz basiert (was im Deutschen in der Regel der Fall ist). Ist keine Knowledge Base für die Sprache vorhanden, werden Themen aus dem entsprechenden Thesaurus abgeleitet (siehe 3.2.3, „Thesaurus Loader“ ctxload und „Knowledge Base Extension Compiler“ ctxkbtc). Mit Hilfe entsprechender PL/SQL-Pakete kann die Wissensbasis schrittweise erweitert werden (siehe 3.2.3, CTX_DOC). Damit lässt sich ABOUT in fast allen Sprachen einsetzen, wenn auch mit verschieden hohem Aufwand.

4.6.4 XML-spezifische Operatoren

Unter Zuhilfenahme des jungen Datentyps XMLType wird es möglich, (XPath-) Anfragen zu stellen, die sich direkt auf die Struktur eines XML-Dokumentes beziehen. Mit Hilfe der von Oracle bereitgestellten Sektionierung lassen sich auch einfache

Datentypen und nicht zwingend XML-konforme HTML-Dokumente derart aufbereiten. Die Operatoren `HASPATH` (Pfad existiert im Dokument), `INPATH` (Suchbegriff kommt im Pfad vor) sowie `WITHIN` (Suchbegriff kommt im Zweig vor) dienen dann der gezielten Abfrage dieser Indizierungsdaten. Letztere unterscheiden sich dahingehend, dass `WITHIN` lediglich einen einfachen Sektionsnamen erhält, der beliebig tief im Dokument vorkommen kann, `INPATH` dagegen einen vollständigen (XPath-) Pfad.

Interessant wird insbesondere der `WITHIN`-Operator, da er auf einfachstem Wege und meist ohne unmittelbare Änderung der HTML-Quelldokumente die Indizierung von zum Beispiel Titeln und Überschriften erlaubt. Wie in Kapitel 3.1.1 gezeigt, erfordert der Datenbestand des IS-GBE jedoch zumindest eine Übersetzung der vorliegenden Dateien, um diese Vorteile nutzen zu können.

4.6.5 Thesaurus-spezifische Operatoren

Neben dem bereits erwähnten Operator `ABOUT` bietet Oracle eine Reihe weiterer Möglichkeiten, um spezifische Beziehungen des zugrunde liegenden Thesaurus in die Suche einzubeziehen.

Tabelle 3: Thesaurus-spezifische Operatoren

Operator	Bezeichnung	Beschreibung
BT (BTG, BTI, BTP)	Broader Terms (Generic, Instance, Partitive)	Ausgedehntere Begriffe (Oberbegriffe, Instanzbegriffe, Teilbegriffe)
NT (NTG, NTI, NTP)	Narrower Terms (Generic, Instance, Partitive)	Engere Begriffe (Oberbegriffe, Instanzbegriffe, Teilbegriffe)
PT	Prefered Term	Bevorzugter Begriff
RT	Related Terms	Verwandte Begriffe
SYN	Synonyms	Bedeutungsgleiche Begriffe
TR	Translation Terms	Übersetzungen
TRSYN	Translation Terms Synonyms	Bedeutungsgleiche Übersetzungen
TT	Top Term	Oberbegriff

4.6.6 Entwicklung eines IS-GBE-spezifischen Sub-Sets

Die aufgezählten Operatoren, die CONTAINS mitbringt, sind kaum geeignet, um sie einem unbedarften Anwender ungefiltert zur Verfügung zu stellen. Selbst gut dokumentiert wären Unklarheiten und Fehleranfälligkeiten unvermeidlich. Aus diesem Grund ist es wünschenswert, dem Anwender einer grafischen Oberfläche nur sehr wenige, möglichst vertraute und dennoch mächtige Operatoren zur Verfügung zu stellen, mit denen er weitestgehende Kontrolle über die durch seine Eingaben generierten SQL-Anfragen hat. Es erscheint sinnvoll, die Erfahrungen der in Kapitel 5.3.2 beschriebenen Implementation zu nutzen.

Internet- und andere Suchmaschinen bieten heute eine schon aus praktikablen Gründen weitestgehend standardisierte Anfragesprache. Zum etablierten Sprachumfang gehören hauptsächlich logische Gruppierungen (Klammern) und Verknüpfungen (`AND`, `+`, `OR`, `NOT`, `-`), Phrasen (in doppelten Anführungszeichen) sowie Sonderfunktionen, die sich im Allgemeinen mit einem Doppelpunkt getrennt in der Form „Operator: Suchbegriff“ präsentieren. Die Auswahl reicht dabei von der auf Titel eingeschränkten Suche („title:“, „intitle:“) über Links, Linkbeschriftungen bis hin zum Alter oder Indizierungszeitpunkt der möglichen Fundstellen. [DeLisle03] Alle weiterführenden und auch die meisten der Doppelpunkt-Operatoren werden dem Anwender zusätzlich als selbsterklärende Checkboxen und alternative Eingabefelder präsentiert. Ein „Erlernen“ der Anfragesprache wird somit weitestgehend vermieden, steht erfahrenen Anwendern jedoch als ständige Option zur Verfügung.

Für `CONTAINS`-Anfragen (im Bezug auf den Datenbestand des IS-GBE) wäre ein daran angelehntes System wünschenswert. Basiselemente der Sprache wie „`AND`“, „`OR`“ sowie Klammerungen können sehr einfach in ihre `CONTAINS`-Äquivalente übersetzt oder sogar so belassen werden. Eingaben wie zum Beispiel das Minuszeichen müssen ihrer externen und internen Bedeutung entsprechend besonders gehandhabt werden. So erwartet der Anwender von einem Minus in der Regel, dass es den dahinter folgenden Begriff völlig ausschließt. `CONTAINS` kennt diese Operation jedoch als „`NOT`“ bzw. „`~`“. Das Weglassen eines Operators kann intern entweder als „`AND`“ oder

„ACCUM“ aufgefasst werden. Letzteres schließt Dokumente, bei denen einer der Suchterme fehlt, nicht völlig vom Ergebnis aus, sondern reduziert lediglich ihren SCORE. Das standardmäßige Annehmen des ACCUM-Operators macht also nur dann Sinn, wenn die Sortierung der Ausgabe auch nach diesem Rang erfolgt.

Andere, typischerweise wünschenswerte Operatoren wie das Stemming mit „$“ (für Beugungsformen, zusammengesetzte Substantive etc.) können allen eingegebenen Suchbegriffen automatisch vorangestellt werden. Dies kann durch (De-) Aktivierung einer Checkbox oder standardmäßig immer geschehen. Die unscharfe Suche mittels FUZZY kann vor allem bei eventuellen Tippfehlern sehr hilfreich sein. Da FUZZY im Gegensatz zum Stemming sehr viele andere Schreibweisen zulässt, sollte die entsprechende Checkbox jedoch nicht automatisch aktiviert sein. Interessant wäre die automatische Zuhilfenahme des Operators jedoch, wenn auf die normale Anfrage hin nichts gefunden werden konnte.

Interessant wäre auch die Zusammenfassung der Operatoren FUZZY und ABOUT zu einer gemeinsamen Option für eine „unscharfe Suche“. Da beide völlig gegensätzliche Strategien verfolgen (ähnliche Schreibweisen und ähnliche Bedeutungen) ließe sich damit eine in jeder Beziehung sehr breite Unschärfe erzielen. Die entstehende CONTAINS-Anfrage müsste mit passend gewählten Und- und Oder-Verknüpfungen so gestaltet werden, dass zu jedem Suchbegriff entweder ähnlich geschriebene (FUZZY) oder verwandte (ABOUT) Begriffe zu einem positiven Suchergebnis führen. Dank der intern kalkulierten Wertigkeiten (SCORE) wäre dafür gesorgt, dass trotz einer erhöhten Trefferanzahl die „schärfsten“ Ergebnisse zuoberst erscheinen.

4.7 Implementationsumfang

Aufgabe dieser Arbeit ist das Erforschen und Bewerten zukünftiger Ausbaumöglichkeiten für das IS-GBE-Gesamtsystem. Der zu implementierende Prototyp folgt somit keinem unmittelbarem Einsatzzweck, sondern soll lediglich exemplarisch bestimmte Technologien und Techniken darstellen, um deren Wirkung und Nutzen belegen zu können.

Die Datenbank der prototypischen Implementierung sieht eine einzige, flache Tabelle für die Speicherung der Dokumente vor. Die Ablage geschieht primär in einem `CLOB`-Datenfeld, zu Testzwecken auch in `BLOB`- und `XMLType`-Feldern.

Der Basisumfang des Prototyps sieht das Hinzufügen von Dokumenten und ganzen Dokumentensammlungen über verschiedene Schnittstellen vor. Daneben wird eine komfortable Oberfläche zur detaillierten Konfiguration des Index zur Verfügung stehen, um Änderungen und Auswirkungen auf Suchanfragen möglichst unmittelbar testen zu können.

Für die Suche kommen mehrere konkurrierende Methoden zum Einsatz, aus denen der Anwender wählen kann. Zu Vergleichs- und Testzwecken sind Index-lose Anfragen mittels `LIKE` und `DBMS_LOB.INSTR` vorgesehen. Dabei kommt die in Kapitel 5.3.2 beschriebene Abfragesprache und der entsprechende Parser zum Einsatz. Für weitere Tests wird die ungefilterte Eingabe des `CONTAINS`-Teils der SQL-Anfrage möglich sein, um die größtmögliche Freiheit zu haben. Als letzte Methode werden die in Kapitel 4.6.6 geführten Überlegungen für eine anwenderfreundliche Repräsentation der `CONTAINS`-Operatoren exemplarisch zum Einsatz kommen.

5 Implementierung/Prototyp (auf 9*i*-Basis)

5.1 Einsatzbeispiel

Ein einfaches aber vollständiges Beispiel soll die Arbeit mit einem `CONTEXT`-Index verdeutlichen. Zuerst wird eine Tabelle mit Hilfe eines PL/SQL-Programms (siehe Anhang „PL/SQL-Programm zur Zufallstext-Erzeugung“) mit einigen zufällig generierten Phrasen gefüllt.

Der nächste Schritt ist äußerst wichtig. Es muss ein spezieller Lexer mit aktivierter Themen-Indizierung vorbereitet werden. Standardmäßig ist dieser Parameter abgeschaltet. `ABOUT` würde in diesem Fall ganz einfach dieselben Ergebnisse präsentieren wie ein einfaches `CONTAINS`, ohne irgendeinen Hinweis oder eine Fehlermeldung auszulösen. Auch die Sprachwahl ist hier wichtig, da die Themen-Indizierung auf eine vorhandene Wissensbasis vertraut, die standardmäßig nur für Englisch und Französisch vorhanden ist. Abhängig von der Oracle-Installation kann die Sprache jedoch von Englisch verschieden sein, so dass die nicht mögliche Themenindizierung in diesem Fall eventuell Fehler auslösen würde.

```
BEGIN
  CTX_DDL.CREATE_PREFERENCE('myLexer', 'BASIC_LEXER');
  CTX_DDL.SET_ATTRIBUTE('myLexer', 'INDEX_THEMES', 'YES');
  CTX_DDL.SET_ATTRIBUTE('myLexer', 'THEME_LANGUAGE', 'ENGLISH');
END;
```

Abbildung 19: Konfiguration eines nutzerdefinierten Lexers

Die hierbei einsetzbaren Attribute sind im Anhang (siehe „Lexer-Attribute“) eingehender beleuchtet. Sodann kann der Index unter Zuhilfenahme des neu definierten Lexers angelegt werden.

```
CREATE INDEX myIndex
  ON myTable(myValue)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS('LEXER myLexer');
```

Abbildung 20: Indexerstellung

Jede nun folgende Abfrage greift auf die Daten dieses Index zurück, abhängig von den eventuell gewählten Operatoren.

```
SELECT SCORE(1), myValue
  FROM myTable
  WHERE CONTAINS(myValue, 'ABOUT(colors)', 1) > 0
  ORDER BY SCORE(1) DESC;
```

Abbildung 21: CONTAINS mit ABOUT-Abfrage

5.2 Konzeptioneller Prototyp mit PHP 5

Ein erster skizzenhafter Prototyp wurde mit der seit Juni 2003 im Betastadium verfügbaren Version 5.0 der Skriptsprache PHP erstellt. Die vorliegenden, persönlichen Erfahrungen legten nahe, dass diese auf äußerst schnelle Ergebnisse konzipierte Sprache gut geeignet ist, um zügig zu ersten Testergebnissen zu gelangen. Mit den Erweiterungen „Oracle Standard“ sowie „Oracle8 Call-Interface“ (OCI8) bietet PHP zwei Schnittstellen zur Kommunikation mit der Datenbank. Die mit Version 5.0 stark ausgebauten objektorientierten Eigenschaften der Sprache erlauben ebenso strukturierte Vorgehensweisen wie sie beispielsweise mit Java-Servlets möglich sind.

Abbildung 22: Suchergebnisse im Prototyp

Konzepte und Vorgehensweisen des Prototyps wurden relativ sprachunabhängig so geplant, dass sie später problemlos in andere Systeme wie Java oder PL/SQL übernommen werden konnten. Dies erwies sich schon nach kurzer Zeit als vorteilhaft, da sich die Implementation nach und nach als deutlich hürdenreicher herausstellte als geplant. Als besonders mangelhaft erwies sich die – aus verschiedenen Gründen – nur wenig ausgebaute Oracle-Schnittstelle, die lediglich den Funktionsumfang von

Oracle8*i* abdecken kann. Insbesondere besteht keine Möglichkeit, den für den Prototyp besonders interessanten Datentyp XMLType zu nutzen.

Die Datentypen `BLOB` und `CLOB` sowie ihre Eigenschaften und PL/SQL-Methoden konnten anhand des ersten PHP-Prototyps dagegen problemlos untersucht werden. Der Datentyp `CLOB` ist von den genannten Möglichkeiten der deutlich einfachst zu handhabende. Die zur Verfügung gestellten Schnittstellenmethoden verhalten sich sehr konsistent zu den Standardmethoden zum Lesen und Schreiben von zum Beispiel `NUMBER`- oder `VARCHAR2`-Datentypen.

Wenngleich sich die Unterschiede zwischen `CLOB` und `BLOB` im Grunde auf die automatische bzw. unterdrückte Umwandlung von Zeichensätzen beschränken, verhalten sie sich im direkten Vergleich doch überraschend anders. So erwartet Oracle beispielsweise, dass mittels „`INSERT … VALUES('Daten')`“ übertragene Daten in einer merkwürdig erscheinenden hexadezimalen String-Darstellung kodiert werden. Das PL/SQL-Paket `UTL_RAW` bietet hierfür eine Methode, die aus `UTL_RAW.CAST_TO_RAW('Daten')` letztendlich `'446174656E'` entstehen lässt. Da es zur Arbeit mit Daten jenseits von 4000 Bytes ohnehin notwendig ist, auf Methoden auszuweichen, die mit Zeigern auf Speicherbereiche agieren, wird man jedoch nicht oft mit diesen Eigenarten konfrontiert.

Die mit dem PHP-Prototyp gewonnenen Erfahrungen und Erkenntnisse und sogar die aufgetretenen Probleme konnten nahezu eins zu eins auf den späteren Java-Prototyp übertragen werden, da sich die Schnittstellen und grundsätzlichen Vorgehensweisen sehr stark ähneln. Die aufgewendete Zeit für die Arbeit mit der „Open Source“-Sprache stellte sich letztendlich also als durchaus sinnvoll heraus, wenngleich sich die Anbindung an Oracle9*i* bzw. 10*g* entgegen den anfänglichen Annahmen leider als alles andere als optimal erwies.

5.3 Prototyp auf Basis von Java/JSP

5.3.1 Struktur/Klassenhierarchie

Der Prototyp versucht, unter Einsatz von JSP (Java Server Pages) und selbstdefinierter Klassen weitestgehend im Rahmen des MVC-Modells (Model-View-Control) zu bleiben. Die Darstellung des Hauptmenüs wird von einer zentralen JSP-Datei übernommen. Die Datenbank, jedes einzelne Dokument sowie jede Suchanfrage wird als Objekt einer entsprechenden Klasse gehandhabt.

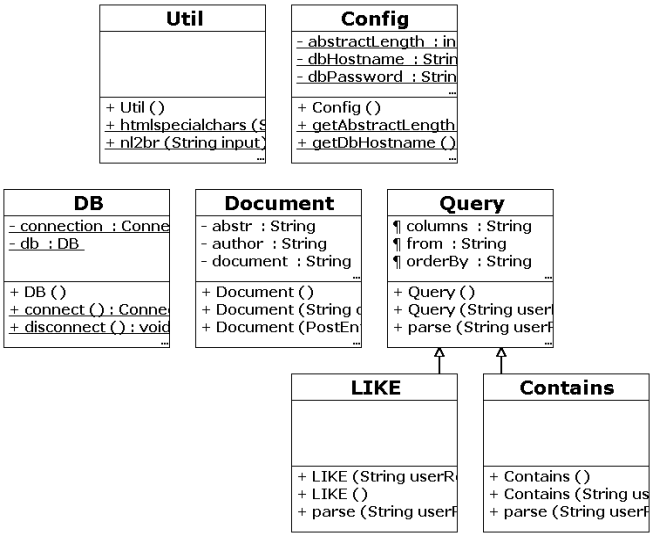

Abbildung 23: Klassendiagramm (gekürzt)

Abhängig von der gewählten Suchmethode erhält das entsprechende Query-Kindobjekt die vom Anwender eingegebene Suchanfrage, erstellt daraus eine SQL-Anfrage, übergibt diese an das Datenbankobjekt und liefert im Ergebnis ein Sammlung aller gefundenen Dokumentenobjekte an das Hauptprogramm (JSP) zurück, das diese dann darstellt.

5.3.2 „User Request/SQL"-Parser

Um die zu formulierenden Anfragen für den Anwender einer Benutzungsoberfläche möglichst leicht zu machen und ihn nicht mit unnötigen SQL-Sprachelementen zu konfrontieren, wurde im Rahmen der in späteren Kapiteln detaillierter erläuterten Entwicklung des Prototyps eine eigene Anfragesprache entwickelt. Diese Abstraktion versucht, sich an den Konzepten etablierter Suchmaschinen zu orientieren. Neben

den Operatoren `AND`, `OR` und `NOT` werden in Anführungszeichen gesetzte Phrasen und viele alternative Schreibweisen für die Operatoren unterstützt. Zwischen Groß- und Kleinschreibung wird generell nicht unterschieden. Der Anwender ist damit nicht mehr gezwungen, sich an eine spezifische Syntax zu halten, sondern kann seine Anfrage so formulieren, wie er es vielleicht von anderen Systemen gewohnt ist.

Der eingegebene Text wird im ersten Schritt anhand von Leerstellen in einzelne Abschnitte zerteilt. Bestimmten Worten oder Zeichen ist dabei eine Sonderbedeutung zugeordnet. Neben `AND` werden unter anderem `UND`, `&`, `&&`, und `+` erkannt. `OR` ist äquivalent zu `ODER`, `|`, `||` und dem Komma. Anstatt `NOT` kann auch `NICHT`, `-` und `~` verwendet werden. Beim Fehlen eines Operators wird `AND` als Standard angenommen. Beim Aufeinanderfolgen mehrerer konkurrierender Operatoren wird der zweite als maskiert betrachtet und wie ein normaler Suchbegriff behandelt. So ist beispielsweise die Eingabe von „`+UND`“ möglich, ohne dass daraus ein fehlerhaftes „`AND AND`“ würde.

Als Sonderform übersetzt der Parser die im Allgemeinen üblicheren Platzhalterzeichen „`?`“ und „`*`“ in die `LIKE`-Äquivalents „`_`“ sowie „`%`“. Letztere sind dennoch bei der direkten Eingabe erlaubt.

Anführungszeichen bestimmen letztendlich ganze Phrasen, in denen auch Leerzeichen, Groß-/Kleinschreibung sowie eventuelle Sprachelemente wie „UND“ als Teil des Suchbegriffs akzeptiert werden. Innerhalb von Phrasen ist ebenso keines der Platzhalterzeichen möglich. Stattdessen nutzt der Parser intern die Syntax „`LIKE '100\%' ESCAPE '\'`“, die diese Zeichen ihrer Sonderbedeutung beraubt.

5.3.3 Typische Probleme mit JSP/Servlets

Das von Java standardmäßig zur Verfügung gestellte Objekt `request` handhabt GET- und POST-Anfragen identisch, solange der eingesetzte Content-Type (`enctype`) dem Standard „application/x-www-form-urlencoded“ entspricht (d.h., wenn kein abweichender Content-Type angegeben wurde). Das Objekt `request` erlaubt es dann zum Beispiel, nach `request.getParameter("parametername")` zu fragen, ohne Rücksicht auf die genaue Art der Quelle zu nehmen.

In Sonderfällen wie dem folgenden Beispiel hat diese Methode den Seiteneffekt, nur einen der vier Werte zurückzuliefern, nämlich den ersten der GET-Parameter (in diesem Fall „1“).

```
<form action="filename.jsp?param=1&param=2" method="post">
<input name="param" value="3" />
<input name="param" value="4" />
</form>
```

Im Gegensatz dazu liefert `request.getParameterValues("param")` ein vollständiges Array aller vier Werte.

Das Verhalten des Objektes `request` ändert sich radikal, sobald der Content-Type des Formulars nicht mehr dem Standard sondern „`multipart/form-data`“ entspricht. Dann werden die – nach wie vor per POST übertragenen, aber anders kodierten – Parameter nicht mehr von der Methode `getParameter` erkannt. Statt dessen muss über externe (d.h. nicht zum Standardumfang von Java und JSP gehörende) Klassen und Methoden ein zum Parsen dieser Kodierung fähiges Objekt generiert und abgefragt werden. [Rossbach00]

Neben weiteren kleineren Schwierigkeiten und allgemeinen Problemen mit der Sprache Java als solches (Stringvergleiche mittels `==` liefern kommentarlos `false`, `toString`-Methoden liefern Speicheradressen etc.) traten konzeptionelle Unklarheiten bezüglich des angewandten MVC-Modells auf. Obgleich JSP einen lobenswerten Ansatz für die View-Schicht des Modells bietet, war besonders die Trennung von Modell und Controller nicht völlig sauber möglich. Durchdachtere Klassenhierarchien und ein konsequenter Einsatz von Beans/EJBs kann hier längerfristig Abhilfe schaffen.

6 Ausblick

6.1 Fazit zum Einsatz Oracle9*i*/10*g*

Die im Moment noch aktuelle Version 9.2 des Oracle Datenbanksystems bietet gerade im Hinblick auf die „Text“-Eigenschaften viele Verbesserungen, die sich hervorragend in das bestehende IS-GBE-System eingliedern lassen. Allerdings muss angemerkt werden, dass die meisten dieser denkbaren Neuerungen im Vergleich zu den jetzt teils nachträglich auf PL/SQL-Ebene implementierten Eigenschaften keinen direkt erkennbaren zusätzlichen Nutzen für den Kunden nach sich ziehen würden. Die Hervorhebung von Suchbegriffen im Text ist ein Beispiel hierfür. Da diese Eigenschaft in früheren Oracle-Versionen fehlerhaft arbeitete, wurde ein eigenes Äquivalent entwickelt. Eine Umstellung auf die eingebaute Prozedur würde das System verschlanken und evtl. leicht beschleunigen, für den Anwender jedoch keinen Unterschied machen.

Lohnenswert erscheint dagegen die Nutzung von Oracle-Eigenschaften, die IS-GBE bisher nicht in dieser Form bietet. In erster Linie ist hier die Suche nach ähnlichen Dokumenten anhand ihres Inhaltes zu nennen, die Generierung von Abstracts und die schon mehrfach genannte Ausweitung der flachen Thesauri zu vollwertigen Wissensbasen. Alle genannten Änderungen würden bei mittlerem bis relativ hohem Aufwand zwar kleine aber attraktive Verbesserungen nach sich ziehen.

Fazit zum besonderen Einsatz Oracle10*g*

Die angekündigten Neuerungen von Oracle10*g* erlauben weitergehende Effizienzsteigerungen, indem sie bisher gesondert zu implementierende Funktionen und „Workarounds“ als direkt einsetzbare PL/SQL-Methoden und SQL-Anfragen anbieten. Summarisch lässt sich festhalten, dass sich ein Großteil der 10*g*-Änderungen auf die logische Fortführung bestehender Konzepte konzentriert. Das heißt, dass bereits mit 9*i* eingeführte, bisher aber eingeschränkte Funktionen ab sofort problemloser und universeller anwendbar und somit auch für IS-GBE endgültig interessant werden. Dies betrifft insbesondere die in Kapitel 3.2.3 erläuterten Cluster und Klassifikationen.

6.2 Fazit zum Einsatz von Fremdprodukten

Ein unmittelbarer Einsatz von Vollprodukten wie der AIDOS KAI| BOX erscheint im Rahmen des bestehenden IS-GBE nur insofern sinnvoll, als dass sie eine zweite, weitestgehend unabhängige Benutzungsoberfläche zur Abfrage der – und nur der – Volltextdokumente bereitstellen können. Eine Integration in den existierenden Kern des Systems zieht streckenweise Änderungen nach sich, die einer Neuimplementation oder auch Ablösung durch das Fremdprodukt gleichkommen. Ob eine weitergehende Untersuchung der von KAI| BOX und SmartDiscovery bereitgestellten APIs (Programmierschnittstellen) sinnvoll ist, muss im konkreten Fall noch entschieden werden. Eine Integration von Teilen der Fremdprodukte wie zum Beispiel der Themen- bzw. Konzepthierarchie erscheint sinnvoll. Es ist jedoch anzumerken, dass die meisten dieser Eigenschaften von Oracle9*i* bzw. 10*g* inzwischen weitestgehend abgedeckt werden.

6.3 Strategien für die Integration in bestehende Produkte

Für die Integration neuer Oracle9*i* und 10*g*-Eigenschaften in das bestehende Produkt IS-GBE lassen sich mehrere Strategien aufzeigen. Zum einen wäre eine völlige Umstellung des Gesamtsystems auf Java denkbar („Servlets“ für Präsentations- und Teile der Geschäftslogik, „Beans“ für möglichst wieder verwertbare Teile des Kernsystems, evtl. auch JSP für die Präsentationsebene). Dies käme letztendlich einer praktisch vollständigen Neuimplementierung des gesamten Systems gleich, da nur einige wenige zentrale Elemente des existierenden, aus mehreren zehntausend Zeilen bestehenden SQL- und PL/SQL-Kerns direkt in Java übernommen werden könnten.

Selbst unter Vernachlässigung finanzieller Faktoren scheint eine solche Vorgehensweise nur wenig sinnvoll. Das gilt auch, wenn es gelingen sollte, große Teile des in jedem Fall zu schaffenden Java-Frameworks so aufzubauen, dass sie in weiteren Produkten Verwendung finden könnten.

Eine kontinuierliche Weiterführung des bewährten PL/SQL-Kerns ist sicherlich zu bevorzugen. Mögliche Ausbaustufen wären hier die Zuhilfenahme bestehender medizinischer Thesauri sowie die Möglichkeit zur Erfassung eigener Ontologien bzw.

Wissensbasen. Die schrittweise Ausweitung auf neuere, insbesondere 10g-Eigenschaften sollte beständig weiterverfolgt werden. Die Integration der externen (OEM-) Software „Star Tree“ zur grafischen Aufbereitung von Suchergebnissen und Klassifizierungsbäumen sollte ebenfalls einer detaillierteren Kosten/Aufwands-Analyse unterzogen werden.

Eine Umstellung der Datenbasis auf XML, wie schon mehrfach u.a. in Kapitel 4.2 angeführt, ist eine weitere Ausbaumöglichkeit. Wenngleich dieser Schritt mit hohen Kosten auch für den Kunden verbunden ist (Vorgang der Dokumentenerfassung wird stark verändert, existierende Dokumente müssen konvertiert werden), ist dies aus Sicht des Autors vor allem längerfristig eine sehr lohnenswerte Investition. Nicht nur die Durchsuchbarkeit innerhalb des Web-Informationssystems wäre auf diesem Wege langfristig verbesserbar, auch die in absehbarer Zeit unzweifelhaft notwendig werdende Umstellung auf wohlgeformtes, gültiges XHTML und die Wiederverwendbarkeit der Dokumente in anderen Medien (CD-ROM, Druck) ließe sich deutlich vorantreiben.

Quellenverzeichnis

[Bachmann03] Bachmann, Christian. „Leicht lesbar?“. 2003. <http://www.leichtlesbar.ch/>

[DeLisle03] DeLisle, Judi. „Web Search Engines Comparison Chart“. Valencia Community College, 2003. <http://valencia.cc.fl.us/lrcwest/searchchart.html>

[DIN1463] DIN 1463-1 und 2. „Erstellung und Weiterentwicklung von Thesauri“. <http://www.din.de/>

[DIN32705] DIN 32705. „Erstellung und Weiterentwicklung von Klassifikationssystemen“. <http://www.din.de/>

[Dubrow03] Dubrow, Manfred. „Qualifizierung der Textrecherche – Stand und Perspektiven“. rds GmbH. Dresden, 2003.

[Fuhr00] Fuhr, Norbert. „IR-Konzepte“. Universität Duisburg, 2000. <http://www.is.informatik.uni-duisburg.de/teaching/dortmund/lectures/ir_ws00-01/folien/irfk2.ps>

[Goldfarb96] Goldfarb, Charles F. „The Roots of SGML – A Personal Recollection“. 1996. <http://www.sgmlsource.com/history/roots.htm>

[Grossman98] Grossman, David A.; Frieder, Ophir. „Information Retrieval: Algorithms and Heuristics“. Kluwer Academic Publishers, London, 1998.

[Held01] Held, Werner. „Quantentheorie der Information“. 2001. <http://www.datadiwan.de/experten/he_002d_.htm>

[Klettke03] Klettke, Meike, Meyer, Holger. „XML & Datenbanken. Konzepte, Sprachen und Systeme“. dpunkt.verlag, Heidelberg, 2003.

[Loney01] Loney, Kevin; Koch, George. „Oracle8*i* – Die umfassende Referenz“. Carl Hanser Verlag, München, 2001.

[Losee98] Losee, Robert M. „Text Retrieval and Filtering. Analytic Models of Performance“. Kluwer Academic Publishers, 1998.

[Mresse84] Mresse, Moscheh. „Information Retrieval: Eine Einführung“. B.G. Teubner, Stuttgart, 1984.

[OWL] „OWL Web Ontology Language Overview“. W3C, 2003. <http://www.w3.org/TR/owl-features/>

[OWLFAQ] „Frequently Asked Questions on W3C's Web Ontology Language (OWL)“. W3C, 2003. <http://www.w3.org/2003/08/owlfaq>

[Rahm03] Rahm, Erhard; Vossen, Gottfried. „Web & Datenbanken. Konzepte, Architektur, Anwendungen“. dpunkt.verlag, Heidelberg, 2003.

[Rossbach00] Rossbach, Peter; Schreiber, Hendrick. „Java Server und Servlets – Portierbare Web-Applikationen effizient entwickeln“. Addison-Wesley, München, 2000.

[Scherer01] Scherer, Douglas and Brennan, Carol. „Exploring Oracle Text Basics“. Oracle Magazine March/April 2001. <http://www.oracle.com/oramag/oracle/01-mar/o21int.html>

[Schwickert01] Schwickert, Dr. „Organisation und Management der IV im Unternehmen“. Universität Trier. 2001. <http://www.wi.uni-trier.de/lehre/skripte/AWI/Schwickert/AWI_SS_2001_BWL_056_112.pdf>

[Steenis92] van Steenis, Hein. „Informationssysteme – Wie man sie plant, entwickelt und nutzt.“ Carl Hanser Verlag, München; Wien, 1992.

[TextFAQ] „Oracle Text FAQ“. <http://otn.oracle.com/products/text/x/FAQs/imt_Faq.html>

[Watson] Watson, Dennis G. „Brief History of Document Markup”. University of Florida. <http://edis.ifas.ufl.edu/BODY_AE038>

[WikiDiph] Wikipedia. „Diphthong“. 2003-05-13. <http://de.wikipedia.org/wiki/Diphthong>

Anhänge

Einstieg/Erfahrungsbericht PL/SQL

Was ist SQL*Plus

SQL*Plus ist die Produktbezeichnung für das – oder vielmehr für die – interaktiven oder im Stapelbetrieb arbeitenden Abfragewerkzeuge, die zusammen mit jeder Oracle Server- und Client-Installation zur Verfügung stehen. Das kommandozeilenorientierte Werkzeug steht neben einer DOS- und Windows-Version als Webbasiertes „*i*SQL*Plus“ sowie mit „SQL*Plus Worksheet“ auch als Editor-orientierter Teil des Oracle Enterprise Managers zur Verfügung. Neben normalen SQL-Kommandos kann das Werkzeug auch PL/SQL-Blöcke ausführen, die sich in der Regel dann auch editieren, speichern und laden lassen. Protokollfunktionen zum Wiederaufrufen und Bearbeiten der zuletzt ausgeführten Kommandos sowie Syntax-Highlighting vervollständigen die nichts desto trotz einfachen Hilfsmittel.

Einführung

PL/SQL steht für „Procedural Language extensions to SQL“, ist also eine prozedurale Spracherweiterung für SQL. Es verbindet die Vorteile einer Programmiersprache der 4. Generation (d.h. SQL) mit den Vorteilen einer Sprache der 3. Generation (angelehnt an Ada, etwa vergleichbar mit Pascal). Die Verwandtschaft zu Ada lässt sich beispielsweise im Zuweisungsoperator `:=` oder den mit `BEGIN END;` eingefassten Blockstrukturen ausmachen.

PL/SQL ist voll in Oracle-SQL integriert und kann in beiden Richtungen eingeflochten werden: Übliche SQL-Anweisungen lassen sich direkt in den PL/SQL-Quellcode einbetten. Umgekehrt erfordert beispielsweise die SQL-Anweisung `CREATE TRIGGER` einen in PL/SQL verfassten Trigger-Rumpf. Zur Kompilierungszeit wird der PL/SQL-Quellcode einer Prozedur oder eines Packages in maschinenlesbaren, so genannten m-Code übersetzt. Dieser wird in der Datenbank abgelegt und zur Laufzeit wieder in den Speicher geladen und ausgeführt.

```
-- Alle Programmausgaben anzeigen (einzeiliger Kommentar)
SET SERVEROUTPUT ON
/* Begin des Hauptprogramms (mehrzeiliger Kommentar) */
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hallo Welt!');
END;
/
```

Der abschließende Schrägstrich des obigen „Hallo Welt“-Programms sorgt für die unmittelbare Beendigung der SQL*Plus-Konsole. Ein weiteres Beispiel soll die bei PL/SQL übliche Verbindung von SQL-Anweisungen wie `UPDATE` und `INSERT` mit Variablendeklarationen und bedingten `IF`-Anweisungen verdeutlichen.

```
DECLARE
  newName VARCHAR2(50) := 'Scott Urman';
  newMajor VARCHAR2(10) := 'History';
BEGIN
  UPDATE students SET major = newMajor WHERE name = newName;
  IF SQL%NOTFOUND THEN
    INSERT INTO students (ID, name, major)
      VALUES (student_sequence.NEXTVAL, newName, newMajor);
  END IF;
END;
```

Operator-Rangfolge

Tabelle 4: PL/SQL-Operatorrangfolge

Operatoren	Beschreibung
`**`	Exponential-Operator
`+, -`	Vorzeichen-Operator
`*, /`	Multiplikations- und Divisionsoperator
`+, -, ||`	Additions-, Subtraktions- und Verkettungsoperator. Letzterer ist mit der Funktion `CONCAT` völlig identisch, auch hinsichtlich seines Verhaltens verschiedenen Datentypen gegenüber: Obgleich `CONCAT` nur auf Strings, d.h. `CHAR`-, `VARCHAR2`-, `CLOB`-sowie die entsprechenden `N`...-Datentypen anwendbar ist, sorgt die automatische Typumwandlung von Oracle dafür, dass sämtliche Operatoren nahezu unabhängig vom Datentyp sind.

`=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN`	Vergleichsoperatoren. Die Operatoren `<>`, `!=`, `~=` sowie `^=` sind gleichbedeutende Synonyme für eine Überprüfung auf Ungleichheit. Zuweisungsoperatoren wie das von C++ bekannte `+=` gibt es bei PL/SQL nicht (`^=` steht nicht für das bitweise XOR). Des weiteren ist interessant zu wissen, dass die Vergleiche auch mit Datumswerten wie erwartet arbeiten.
`NOT`	Logische Negation
`AND`	Logisches Und
`OR`	Logisches Oder

Kontrollstrukturen

```
Einfache Bedingung:
  IF bedingung THEN
    …;
  ELSIF … THEN
    …;
  ELSE
    …;
  END IF;

Kopfgesteuerte Schleife:
  WHILE bedingung < 5 LOOP
    …;
  END LOOP;

Einfache Schleife (Endlosschleife, fußgesteuerte Schleife) mit
  Abbruchbedingung:
  LOOP
    …;
    EXIT WHEN myCursor%NOTFOUND;
    …;
  END LOOP;

Numerische FOR-Schleife:
  FOR I IN 0..100
  LOOP
    …;
  END LOOP

Cursor-FOR-Schleife:
  FOR myRecord IN myCursor LOOP
    …;
  END LOOP;
```

```
Mehrfachbedingung auf eine Variable:
  rueckgabewert :=
    CASE ausdruck
      WHEN 'A' THEN 'ein A wurde gewählt'
      WHEN 'B' THEN 'das B stand zur Debatte'
      ELSE 'weder A noch B'
    END;

Mehrfachbedingung auf mehrere Variablen:
  rueckgabewert :=
    CASE
      WHEN ausdruck = 'A' THEN 'ein A wurde gewählt'
      WHEN ausdruck = 'B' THEN 'das B stand zur Debatte'
      ELSE 'weder A noch B'
    END;
```

Blockstruktur

Unterprogramme werden in der Syntax von PL/SQL in Prozeduren und Funktionen separiert. Prinzipiell sind beide völlig identisch, wie es auch in der Syntax anderer Sprachen wie beispielsweise C++ gängig ist. PL/SQL konkretisiert die Unterscheidung dahingehend, dass Prozeduren keinesfalls einen Rückgabewert haben dürfen, Funktionen dagegen zwingend einen liefern müssen.

```
CREATE OR REPLACE PROCEDURE myProcedure IS
  /* lokaler Deklarationsteil */
BEGIN
  /* Anweisungen */
  RETURN;
END;
CALL myProcedure();

CREATE FUNCTION mySqrt (myNumber OUT NUMBER) RETURN NUMBER IS
  /* lokaler Deklarationsteil */
BEGIN
  /* Anweisungen */
  RETURN SQRT(myNumber);
END;
CALL mySqrt(12);
```

Neben `CREATE PROCEDURE` und `FUNCTION` stehen mit `CREATE OPERATOR`, `PACKAGE` und `CREATE JAVA` weitere Sprachelemente zur Konstruktion von Operatoren, PL/SQL-Paketen und Java-Methoden zur Verfügung.

Ausnahmebehandlung

Oracle definiert viele vorgefertigte Ausnahmen für die gebräuchlichsten Laufzeitfehler wie beispielsweise Division durch Null (Exception `ZERO_DIVIDE`).

```
DECLARE
  myException EXCEPTION;
  …;
BEGIN
  …;
  RAISE myException;
  …;
EXCEPTION
  WHEN myException THEN
    …;
  WHEN … THEN
    …;
  WHEN OTHERS THEN
    …;
END;
```

Innerhalb des `EXCEPTION`-Blockes stehen verschiedene vordefinierte Konstanten/Pseudospalten zur Verfügung: `SQLCODE`, `SQLERRM` oder `USER`. Diese können beispielsweise in geeigneter Weise mittels `INSERT` in einer Error-Log-Tabelle abgelegt werden, um die aufgetretenen Fehler zu einem späteren Zeitpunkt auswerten und beheben zu können.

Die Quelltexte jedes beliebigen prozeduralen Objektes – egal ob Prozedur, Funktion oder Package, egal ob vor- oder benutzerdefiniert – lässt sich mit Hilfe der Data Dictionary-Views `USER_SOURCE` oder `ALL_SOURCE` abfragen. Die folgende SQL-Anfrage zeigt beispielsweise die vollständige Quelle des Packages `DBMS_OUTPUT` an. (Vorsicht bei der Ausführung, da es sich um einige hundert Ergebniszeilen handeln kann.)

```
SELECT text FROM ALL_SOURCE
  WHERE name = 'DBMS_OUTPUT' ORDER BY line;
```

Mit Oracle ausgelieferte PL/SQL-Packages

Tabelle 5: PL/SQL-Pakete

Package	Beschreibung
DBMS_ALERT	Bietet Unterstützung für die asynchrone Verarbeitung von Datenbank-Ereignissen. Dieses Package spielt seine Stärken vor allem im Zusammenhang mit Triggern aus, indem beispielsweise der (synchrone) Trigger ein Signal auslöst (Prozedur DBMS_ALERT.SIGNAL), das dann eine wartende, asynchrone Prozedur anstößt.
DBMS_BACKUP_-RESTORE	Normalisiert Dateinamen auf Windows NT-Platformen.
DBMS_DEBUG	Implementiert einen serverseitigen Debugger und bietet Möglichkeiten für eine serverseitige Fehlersuche in PL/SQL-Programmen. Besser gesagt handelt es sich bei DBMS_DEBUG um eine PL/SQL-API zum PL/SQL-Debug-Layer – „Probe“ – des Oracle-Servers. Um den Debugger nutzen zu können, muss eine zweite Datenbank-Session gestartet sein, um den zu debuggenden Code überwachen zu können. DBMS_DEBUG stellt dann Funktionen für die Kontrolle von Breakpoints, Timeouts, Quellcodeanzeige und einiges mehr zur Verfügung.
DBMS_LOB	Generelle Routinen zur Arbeit mit LOB-Datentypen sowie dem BFILE-Typ. DBMS_LOB kann Dateien einlesen und schreiben und überträgt zahlreiche der üblichen String-Funktionen wie SUBSTR, TRIM etc. auf die LOB-Datentypen.
DBMS_LOCK	Lässt die Arbeit mit dem Oracle Lock Management-Service zur Sperre von Tabellen etc. zu. Außerdem implementiert es eine Prozedur DBMS_LOCK.SLEEP, die das Programm eine wählbare Anzahl von hundertstel Sekunden schlafen lässt.
DBMS_-OBFUSCATING_-TOOLKIT	Bietet Funktionen nach dem symmetrischen Verschlüsselungsverfahren DES („Data Encryption Standard“). Zusätzliche 3DES-Prozeduren arbeiten anstatt der üblichen 56 mit bis zu 168 Bit langen Schlüsseln.
DBMS_OUTPUT	Sammelt Informationen zur späteren Ausgabe in einem Puffer. Die PUT, PUT_LINE, NEW_LINE, GET_LINE und GET_LINES-Prozeduren erlauben die Interaktion mit dem Anwender und lassen sich beispielsweise für zusätzliche, abschaltbare Debug-Ausgaben verwenden.

DBMS_PROFILER	Stellt eine „Probe“ Profiler-API zur Verfügung, mit der existierende PL/SQL-Applikationen profiliert werden können, um Performanceprobleme und Engpässe aufzuspüren.
DBMS_RANDOM	Stellt mit der Funktion DBMS_RANDOM.RANDOM sowie INITIALIZE und SEED einen Zufallsgenerator zur Verfügung.
DBMS_REPAIR	Bietet Prozeduren zur Reparatur zerstörter Daten.
DBMS_ROWID	Stellt Prozeduren für das Generieren und Interpretieren von ROWIDs zur Verfügung.
DBMS_SQL	Lässt die Verwendung von dynamischem SQL für den Zugriff auf die Datenbank zu. Die von DBMS_SQL gebotenen Möglichkeiten erlauben es, SQL-Anfragen als Strings zu handhaben und so dynamisch zusammenzusetzen und zu ergänzen. Das mit Oracle8*i* eingeführte native dynamische SQL ist in der Regel aber gleichwertig und einfacher zu handhaben als die DBMS_SQL-Prozeduren.
DBMS_TRACE	Bietet Routinen, mit denen sich das Tracen von PL/SQL-Programmen starten und stoppen lässt.
DBMS_-TRANSACTION	Bietet Zugang zu den SQL-Transaktionen (COMMIT, ROLLBACK).
DBMS_TYPES	Besteht aus Konstanten, die alle eingebauten und nutzerdefinierten Datentypen repräsentieren (TYPECODE_NUMBER, TYPECODE_REF etc.).
DBMS_UTILITY	Bietet verschiedene Werkzeug-Routinen, z.B. DB_VERSION, Analyse-, CSV-Funktionen und einiges mehr.
DBMS_XMLGEN	Konvertiert das Ergebnis einer SQL-Anfrage in strukturiertes XML.
SDO_CS	Bietet Funktionen zur Transformation von Koordinatensystemen.
UTL_ENCODE	Kodiert Rohdaten in und aus den kodierten Standardformaten Base64, UUEncode sowie Quoted-Printable.
UTL_FILE	Erlaubt PL/SQL-Programmen, Dateien des Betriebssystems zu lesen und zu beschreiben (FOPEN, GET_LINE, FCLOSE etc.).
UTL_HTTP	Bietet HTTP-Funktionalitäten inklusive Cookies, HTTPS, FTP, Authentisierung usw.
UTL_URL	Stellt ESCAPE- und UNESCAPE-Mechanismen für Internetadressen zur Verfügung.

SQL*Plus-Umgebungsvariablen

Im SQL*Plus-Umfeld existieren verschiedene Systemvariablen, mit denen sich die Umgebung den jeweiligen Bedürfnissen anpassen lässt und die auch für die Arbeit mit PL/SQL wichtig sind. Im Folgenden wird eine kleine Auswahl der subjektiv wichtigsten kurz vorgestellt und erläutert. Mit dem Befehl `SHOW` lassen sich die momentanen Einstellungen dieser SQL*Plus-Merkmale anzeigen, mit dem `SET`-Befehl können sie eingeschaltet, ausgeschaltet oder auf einen bestimmten Wert gesetzt werden.

```
SET AUTOCOMMIT { ON | OFF | IMMEDIATE | n }
```

`AUTOCOMMIT ON` (bzw. das identische `IMMEDIATE`) veranlasst die Datenbank, nach der Ausführung jedes einzelnen SQL-Befehles die durchgeführten Änderungen sofort in der Datenbank festzuschreiben. Standardmäßig ist `AUTOCOMMIT` abgeschaltet und Änderungen müssen mit dem `COMMIT`-Befehl manuell festgeschrieben werden bzw. lassen sich mit `ROLLBACK` rückgängig machen. Die Angabe einer Zahl bewirkt, dass Änderungen immer nach jeweils n SQL-Kommandos bzw. PL/SQL-Blöcken gesichert werden.

```
SET AUTOTRACE { ON | OFF | TRACEONLY } [ EXPLAIN ] [ STATISTICS ]
```

Die Trace- und Explain-Eigenschaft zeigt Analysen und Aufwandsstatistiken zu jeder Datenbankanfrage an. Außerdem werden die Ergebnisse in der Tabelle `EXPLAIN_PLAN` mitprotokolliert, sofern diese existiert. Die mit der Installation mitgelieferte Datei `$ORACLE_HOME/rdbms/admin/utlxplan.sql` erstellt diese Tabelle auf Wunsch.

```
SET ECHO { ON | OFF }
```

Mit eingeschaltetem `ECHO` werden alle SQL*Plus-Befehle bei der Ausführung einer Startdatei zur Kontrolle am Bildschirm angezeigt.

```
SET FEEDBACK { 6 | n | ON | OFF }
```

FEEDBACK meint die Anzeige der Anzahl von Zeilen, die durch eine SQL-Abfrage ausgewählt wurden. Diese informelle Zahl wird normalerweise nur dann angezeigt, wenn eine Aktion wenigstens 6 Datensätze betrifft. Mit Hilfe der FEEDBACK-Umgebungsvariable lässt sich dieser Grenzwert verändern oder die Anzeige der Anzahl generell an- oder abschalten.

```
SET HEADING { ON | OFF }
```

HEADING OFF verhindert die Anzeige der in Reports normalerweise über den Spalten erscheinenden Überschriftentexte und Unterstreichungen.

```
SET LINESIZE { 80 | n }
```

LINESIZE legt die maximale Zeilenlänge fest, die auf den Bildschirm oder den Ausdruck eines Reports ohne Umbruch nebeneinander passen.

```
SET LONG { 80 | n }
```

Legt die maximale Anzahl von Zeichen fest, mit denen der Inhalt einer LONG- oder CLOB-Spalte innerhalb eines Reports gekürzt dargestellt wird. Dies ist sinnvoll, um den Report nicht unnötig aufzublähen und somit unleserlich zu machen.

```
SET MARKUP HTML [ ON | OFF ] [SPOOL { ON | OFF }] […]
```

Wird in der Regel von *i*SQL*Plus genutzt, um die von SQL*Plus generierten Reports anstatt in Textform als angenehm formatiertes HTML darzustellen bzw. abzuspeichern. In Zusammenhang mit dem SQL*Plus-Kommando SPOOL lassen sich diese Reports auch in Dateien umleiten.

```
SET NEWPAGE { 1 | n | NONE }
```

Legt die Anzahl von Leerzeilen fest, die zu Beginn jeder Report-Seite noch vor dem Seitentitel eingefügt werden.

```
SET NUMFORMAT format
```

NUMFORMAT legt das Standardformat fest, mit dem Zahlen innerhalb von Reports dargestellt werden. So bestimmt beispielsweise SET NUMFORMAT 9999D99C länderspezifische Dezimal- und Währungszeichen und führt in deutschsprachigen Installationen zu Ausgaben wie „1234,56EUR“. SET NUMFORMAT RN bringt die Datenbank dazu, sämtliche Zahlen in römischer Schreibweise anzuzeigen.

```
SET NUMWIDTH { 10 | n }
```

Die Standardlänge für die Anzeige von Zahlen lässt sich mit NUMWIDTH festlegen. Dies hat nur Auswirkung, wenn kein NUMFORMAT definiert bzw. dieses auf "" gesetzt wurde.

```
SET PAGESIZE { 24 | n }
```

Stellt die Anzahl von Zeilen pro Seite ein. Mittels PAGESIZE 0 lassen sich die dabei entstehenden Zwischenüberschriften und Seitenumbrüche vollständig unterdrücken.

```
SET PAUSE { ON | OFF | text }
```

Bei aktivierter PAUSE wartet SQL*Plus nach der Anzeige jeder Report-Seite, bis der Anwender die Eingabetaste drückt. Mit *text* lässt sich eine Nachricht festlegen, die in diesem Fall erscheinen soll.

```
SET SERVEROUTPUT { ON | OFF } [SIZE n] [FORMAT { WRAPPED |
WORD_WRAPPED | TRUNCATED }]
```

Legt fest, ob und wie die z.B. mittels DBMS_OUTPUT.PUT_LINE gemachten Ausgaben von PL/SQL-Prozeduren angezeigt werden. Mittels SIZE kann die Größe des Ausgabepuffers in Bytes festgelegt werden, mit FORMAT wird das Verhalten bei eventuell notwendigen Zeilenumbrüchen bestimmt.

```
SET TIMING { ON | OFF }
```

TIMING ON aktiviert eine Statistikfunktion, die mit Beendigung jedes ausgeführten PL/SQL-Blockes automatisch die jeweils benötigte Laufzeit anzeigt.

PL/SQL-Programm zur Zufallstext-Erzeugung

Das folgende Beispielprogramm, das für Tests und Messungen so auch zum Einsatz kam, erzeugt Satzfragmente aus einer Auswahl zufälliger Wörter (hier gekürzt dargestellt).

```
SET SERVEROUTPUT ON
-- Generiert 100 Datensätze mit Zufallstexten zu je 3 Worten
DECLARE
  n NUMBER;
  word VARCHAR(100);
  sentence VARCHAR2(1000);
BEGIN
  FOR sentenceCount IN 1..100
  LOOP
    sentence := '';
    FOR i IN 1..3
    LOOP
      n := MOD(ABS(DBMS_RANDOM.RANDOM), 5);
      word :=
        CASE n
          WHEN 0 THEN 'color'
          WHEN 1 THEN 'blue'
          WHEN 2 THEN 'red'
          WHEN 3 THEN 'travel'
          WHEN 4 THEN 'the'
        END;
      sentence := sentence || word || ' ';
    END LOOP;
    -- Testweise Ausgabe auf dem Bildschirm
    -- DBMS_OUTPUT.PUT_LINE(sentence);
    INSERT INTO tabelle (spalte) VALUES (sentence);
  END LOOP;
END;
```

Versionsgeschichte des Oracle-RDBMS

Tabelle 6: Oracle-RDBMS Versionsgeschichte

Releasenummer	verfügbar seit	vollständige Versionsnummer
Oracle1	1978	-
…	…	…
Oracle7	1993	7.3.4.x.x
Oracle8	1997	8.0.4.x.x bis 8.0.6.x.x
Oracle8*i* Release 1	1999	8.1.5.x.x
Oracle8*i* Release 2	Januar 2000	8.1.6.x.x
Oracle8*i* Release 3	November 2000	8.1.7.x.x (aktuell: 8.1.7.4.x vom Mai 2002)
Oracle9*i* Release 1	Juni 2001	9.0.1.x.x (aktuell: 9.0.1.4.x vom August 2002)
Oracle9*i* Release 2	Juni 2002	9.2.x.x.x (aktuell: 9.2.0.4.x vom August 2003)
Oracle10*g*	2003/2004	10.x.x.x.x (aktuell: 10.1.x.x.x Beta 2 vom Juni 2003)

Lexer-Attribute

Die für die Konfiguration eines nutzerdefinierten Lexers einsetzbaren Attribute werden in der folgenden Tabelle detailliert beleuchtet. Neben den dabei in der Regel üblichen Werten „`YES`“ und „`NO`“ sind im Übrigen auch die Zeichenketten „`Y`“, „`N`“, „`TRUE`“, „`FALSE`“, „`T`“, „`F`“ sowie die echten boolesche Werte `TRUE` und `FALSE` (PL/SQL-spezifisch) erlaubt.

Tabelle 7: Lexer-Attribute

Attribut	Wert	Erläuterung
`WHITESPACE`	*Zeichen*	Zeichen, die Leerraum zwischen Token darstellen. Voreinstellung: Leerzeichen und Tabulator. Diese beiden Zeichen können nicht verändert werden. Die Angabe weiterer `STARTJOINS`-Zeichen fügt diese lediglich hinzu.

Attribut	Wert	Erläuterung
CONTINUATION	*Zeichen*	Eines oder mehrere Zeichen, die ein auf der nächsten Zeile fortgesetztes Wort markieren. Im Deutschen der Bindestrich '-', in anderen Sprachen manchmal auch der Backslash '\'.
PUNCTUATIONS	*Zeichen*	Zeichen, die das Ende eines Satzes markieren. Voreinstellung: Punkt, Frage- und Ausrufezeichen '.?!'.
NUMJOIN	*Zeichen*	Das Dezimaltrennzeichen. Im Deutschen das Komma ',', in anderen Sprachen oftmals der Punkt '.'.
NUMGROUP	*Zeichen*	Das Tausendertrennzeichen. Im Deutschen der Punkt '.', in anderen Sprachen oftmals das Komma ','.
PRINTJOINS	*Zeichen*	Nicht-alphanumerische Zeichen, die als Verbindung mehrerer Worte zu einem Token betrachtet werden sollen. Beispiel: Der Bindestrich '-' sorgt dafür, dass „XML-basiert“ als eigenständiges Wort in den Index aufgenommen wird.
SKIPJOINS	*Zeichen*	Nicht-alphanumerische Zeichen, die genau wie die PRINTJOINS-Zeichen als Verbindung mehrerer Worte zu einem Token betrachtet werden sollen. Unterschied: „XML-basiert“ wird als „XMLbasiert“ in den Index aufgenommen wird. Achtung: PRINT- und SKIPJOINS-Zeichen schließen sich gegenseitig aus.
STARTJOINS/ ENDJOINS	*Zeichen*	Zeichen, die explizit den Beginn eines Tokens festlegen. Im Deutschen/Englischen nicht verwendet.
NEWLINE	{ **NEWLINE (\n)** | CARRIAGE_ RETURN (\r) }	Zeichen, die das Ende einer Zeile festlegen. Voreingestellt auf das sowohl unter Linux/Unix (\n) als auch Windows (\n\r) übliche CARRIAGE_RETURN.
BASE_LETTER	{ **NO** | YES }	Gibt an, ob Umlaute, Akzentzeichen etc. in ihre Basisformen umzuwandeln sind, bevor sie im Index abgelegt werden. Voreinstellung: NO (Umlaute etc. bleiben so erhalten).

Attribut	Wert	Erläuterung
MIXED_CASE	{ **YES** | NO }	Gibt an, ob alle Token in ihrer Groß-/Kleinschreibung belassen werden sollen, wenn sie in den Index aufgenommen werden. Voreinstellung für Deutsch: YES (Groß-/Kleinschreibung wird beachtet), Voreinstellung für alle andere Sprachen laut Oracle-Handbuch: NO (alles wird intern in Großschreibung umgewandelt)
COMPOSITE	{ **GERMAN** | DEFAULT | DUTCH }	Aktiviert die gesonderte Indizierung bekannter Fragmente zusammengesetzter Worte. Abzufragen mit dem Stemming-Operator $. Beispiel: '$versicherung' findet „Rentenversicherung“ ('$rung' findet nichts) und ist dabei um Größenordnungen schneller als '%versicherung'. Voreinstellung für Deutsch: GERMAN, Voreinstellung für andere Sprachen laut Oracle-Handbuch: DEFAULT (deaktiviert)
INDEX_TEXT	{ **YES** | NO }	Lässt die Deaktivierung der normalen Volltextindizierung zu, falls ausschließlich nach Themen gesucht werden soll.
INDEX_THEMES	{ **NO** | YES }	Aktiviert die Themenindizierung, abhängig von der eingestellten THEME_LANGUAGE und der Existenz einer entsprechenden Wissensbasis.
THEME_ LANGUAGE	{ **AUTO** | *Sprache* }	Sprache der zu verwendenden Wissensbasis. Vordefiniert nur für ENGLISH und FRENCH. Bei AUTO wird die systemweit eingestellte Sprache angenommen.
PROVE_THEMES	{ **YES** | NO }	Die „Themenprüfung“ versucht, verwandte Themen in einem Dokument zu finden. Bei Misserfolg werden alle übergeordneten Themen aus dem Dokument entfernt. Für kleine Dokumente sorgt das Deaktivieren der „Themenprüfung“ für ungenauere Ergebnisse bei ABOUT-Anfragen, erhöht jedoch den Recall erheblich. Voreinstellung ist YES.
INDEX_STEMS	{ **NONE** | GERMAN | ENGLISH | *Sprache* }	Indiziert Worte zusätzlich in ihrer normalisierten, ungebeugten Form. Dies vergrößert den Index, erhöht jedoch die Performance für Anfragen mit dem Stemming-Operator $.

Attribut	Wert	Erläuterung
ALTERNATE_SPELLING	{ **GERMAN** | NONE | DANISH | SWEDISH }	Voreinstellung für Deutsch: GERMAN, Voreinstellung für andere Sprachen laut Oracle-Handbuch: NONE
NEW_GERMAN_SPELLING	{ **OFF** | INDEX_NEW | INDEX_BOTH }	Ab Oracle10*g* (jedoch nicht Beta 2!) sind dem System die Regeln der neuen deutschen Rechtschreibung bekannt. INDEX_NEW besagt, dass Token nur noch in ihrer neuen Schreibweise im Index zu führen sind, selbst wenn sie im Text in der alten Schreibung auftreten. INDEX_BOTH führt beide Schreibungen im Index. Voreinstellung: OFF. Anmerkung: Da insbesondere die neue deutsche Rechtschreibung viele Worte mit Bindestrichen einführt, wird empfohlen, das Attribut PRINTJOINS um diesen Strich zu ergänzen.

Gegenüberstellung MySQL – Oracle

Um den bestehenden Wissens- und Erfahrungsstand aus der Arbeit mit dem äußerst weit verbreiteten Open Source-Datenbanksystem MySQL möglichst effektiv weiterverwenden zu können, bot sich eine direkte Gegenüberstellung bekannter Eigenheiten beider Systeme an.

Tabelle 8: MySQL/Oracle-Gegenüberstellung

MySQL	Oracle
ALTER TABLE	Grundsätzlich identisch
AS	Identisch
AUTO_INCREMENT	Deutlich mächtiger mittels CREATE SEQUENCE und CREATE TRIGGER konstruierbar
Backup manuell möglich mittels SELECT INTO OUTFILE, BACKUP TABLE oder mysqldump (Kommandozeilenwerkzeug)	Backup-Möglichkeiten in duzenden Variationen (online, offline, zeitgesteuert etc.)

MySQL	Oracle
Keine Entsprechung, mittels `VARCHAR` sowie eigener Programmlogik konstruierbar	`BFILE` (Zeiger auf externe `LOB`-Daten)
`BLOB` (< 64 KB), `LONGBLOB` (< 4 GB), `MEDIUMBLOB` (< 16 MB), `TINYBLOB` (bis 255 Bytes)	`BLOB` (bis 4 GB), `RAW` (bis 255 Bytes), `LONG RAW` (bis 2 GB)
`BOOLEAN` existiert nicht als Datentyp	`BOOLEAN` existiert nicht als Datentyp, PL/SQL stellt jedoch einen solchen Pseudotyp zur Verfügung.
Keine Entsprechung	`CALL`, `EXECUTE`
Case-insensitiver Stringvergleich	Nicht unterstützt, mittels `UPPER()` konstruierbar, bei der Erstellung von z.B. `CONTEXT`-Indizes global einstellbar
`CHAR` (Zeichenkette fester Länge, bis 255 Bytes), `VARCHAR` (Zeichenkette variabler Länge, bis 255 Bytes)	`CHAR` (Zeichenkette fester Länge, bis 2000 Bytes), `VARCHAR2` (Zeichenkette variabler Länge, bis 4000 Bytes), `NCHAR` (Multibyte-Zeichenkette fester Länge), `NVARCHAR2` (Multibyte-Zeichenkette variabler Länge)
Bedingt konstruierbar mittels `LIKE`, `FULLTEXT`, `MATCH … AGAINST`	`CONTAINS()`
`CREATE DATABASE`	Grundsätzlich identisch
`CREATE TABLE`	Grundsätzlich identisch
`DATE_ADD()`, `DATE_SUB()`, `INTERVAL` mit `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND` (z.B. `SELECT NOW() + INTERVAL 1 YEAR`)	`ADD_MONTHS()`, `NEXT_DAY()` etc., mit +/- lassen sich ganze Tage addieren/subtrahieren, identische Syntax für `INTERVAL`
`DATE_FORMAT(…, '%Y-%m-%d %H-%i-%s')`, außerdem `EXTRACT()`	`TO_CHAR(…, 'YYYY-DD-MM HH24:MI:SS')` in Kombination mit der Umkehrfunktion `TO_DATE()`
`DATE`, `DATETIME`, `TIMESTAMP`, `YEAR` (1000 bis 9999)	`DATE`, `TIMESTAMP` (4712 v. Chr. bis 4712 n. Chr.)
`CASE … WHEN … THEN`, `IF()`	`DECODE()`
`DELETE`	Identisch
`DISTINCT`	Identisch

MySQL	Oracle
`DROP TABLE`	Identisch
`ENUM` (Aufzählung, Zeichenkettenobjekt, kann nur einen Wert der Auflistung haben oder `NULL` oder `""` im Fehlerfall), `SET` (Reihe, Zeichenkettenobjekt, kann beliebig viele Werte der Auflistung haben)	Eingeschränkt konstruierbar mittels `DECODE()`
`FLOAT` (einfache Genauigkeit), `DOUBLE` (doppelte Genauigkeit), `DECIMAL` (unkomprimierte Fließkommazahl)	`NUMBER`
Keine Entsprechung	Foreign keys (`TRIGGER`, `ON DELETE`, `REF` etc.)
`GROUP BY`	Identisch
Gruppenfunktionen `AVG()`, `COUNT()`, `MAX()`, `MIN()`, `STDDEV()`, `SUM()`	Grundsätzlich identisch, zusätzliche Funktion `VARIANCE()`
`HAVING` (nur im Zusammenhang mit `GROUP BY`)	Prinzipiell identisch
`IFNULL()`	`NVL()`
`IN()`, `NOT IN()`, `BETWEEN`	Identisch
`INDEX`	Grundsätzlich identisch
`INSERT INTO tabelle VALUES ('value'); INSERT INTO tabelle (spalte) VALUES ('value'); INSERT INTO tabelle SET spalte = 'value';`	Identisch, lediglich die `SET`-Syntax wird nicht unterstützt
`INT` (4 Bytes), `BIGINT` (8 Bytes), `MEDIUMINT` (3 Bytes), `SMALLINT` (2 Bytes), `TINYINT` (1 Byte), wahlweise vorzeichenbehaftet oder vorzeichenlos	`NUMBER`, `DECIMAL`, `FLOAT`, `INTEGER`, `SMALLINT` (alle gleichbedeutend mit `NUMBER`)
`IS NULL`, `IS NOT NULL`	Identisch
`JOIN`, `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, ...	Identisch seit Version 9.0, mittels `WHERE` konstruierte `JOIN`s kennen auch Sonderformen wie `SELECT * FROM a, b WHERE a.x = b.x(+)` für `OUTER JOIN`s

MySQL	Oracle
`LAST_INSERT_ID()`	Siehe `AUTO_INCREMENT`, der aktuelle Wert der Sequenz ist via Pseudospalte `sequenzname.CURRVAL` ermittelbar
`LIKE`	Grundsätzlich identisch
`LIMIT`	Eingeschränkt realisierbar mittels Pseudospalte `ROWNUM`, z.B. `WHERE ROWNUM <= 7`, oder mittels `ROW_NUMBER() OVER(…)`
Logische Operatoren `AND`, `NOT`, `OR`, `=`, `>`, `<` etc.	Weitestgehend identisch
Keine Entsprechung, mittels `INSERT … SELECT` bedingt konstruierbar	`MATERIALIZED VIEW` bzw. `SNAPSHOT` (austauschbare Begriffe)
`NOW()`	`CURRENT_DATE`, `CURRENT_TIMESTAMP`, `SYSDATE`, `SYSTIMESTAMP` (alles Pseudospalten)
`NULL`	Identisch
Numerische Funktionen `ABS()`, `FLOOR()`, `MOD()`, `POWER()`, `SIN()`, `SQRT()` etc.	Weitestgehend identisch
Keine Entsprechung	Objektrelationales Datenbank-Management-System (optional zur relationalen Datenhaltung)
`ORDER BY`, `ASC`, `DESC`	Identisch
Keine Entsprechung	`PROCEDURE` (stored procedures), `FUNCTION`, `OPERATOR`
`RAND()`	Eingeschränkt mittels `SAMPLE()` möglich, siehe auch `RANDOM`-Funktion des `DBMS_RANDOM`-Packages
`REPLACE()`	Identisch
`ROLLBACK`, `COMMIT`, `AUTOCOMMIT`	Identisch
Keine Entsprechung	`ROWID` (Pseudospalte, eindeutiger Zeilenidentifikator)
`SELECT`	Grundsätzlich identisch

MySQL	Oracle
`SELECT` ohne `FROM`, die Datenbank als „Taschenrechner“ missbrauchen, z.B. `SELECT SQRT(9)`	`SELECT SQRT(9) FROM DUAL;` (zur Verfügung gestellte Dummy-Tabelle, da `FROM` nicht weggelassen werden darf)
`SHOW COLUMNS`	`SELECT * FROM ALL_TAB_COLUMNS` / `USER_TAB_COLUMNS` / `COLS;`
`SHOW DATABASES`	Keine Entsprechung
`SHOW TABLES`	`SELECT TABLE_NAME FROM ALL_TABLES / USER_TABLES / TABS;`
String-Funktionen `CONCAT()`, `LENGTH()`, `SOUNDEX()`, `SUBSTRING()` etc.	Weitestgehend identisch, verkürzte Schreibweise `SUBSTR()`, alternative Schreibweise `||` bei `CONCAT()`
Sub-Selects / Unterabfragen sind nur sehr eingeschränkt anwendbar, unterstützt werden `INSERT … SELECT`, `REPLACE … SELECT` sowie `IN()`	Sub-Selects werden voll unterstützt, z.B. `SELECT city, country FROM location WHERE city IN (SELECT city FROM weather WHERE condition = 'cloudy')`
`TEXT` (< 64 KB), `LONGTEXT` (< 4 GB), `MEDIUMTEXT` (< 16 MB), `TINYTEXT` (bis 255 Bytes)	`CLOB` (bis 4 GB), `NCLOB` (Multibyte-Objekt), `LONG` (bis 2 GB)
Keine Entsprechung, mittels `REPLACE()` konstruierbar	`TRANSLATE()`
Keine Entsprechung	`TRIGGER`
Keine Entsprechung	`TYPE`
`UNION`	Identisch, zusätzlich erlaubt Oracle `INTERSECT` und `MINUS`
`UPDATE tabelle SET spalte = 'value' WHERE spalte = 'value';`	Grundsätzlich identisch
Keine Entsprechung, mittels `INSERT … SELECT` bedingt konstruierbar	`VIEW`
`WHERE`	Grundsätzlich identisch
Keine Entsprechung	`XMLType`

Überlegungen zur Texterfassung mit LaTeX

Zur konsistenten Erfassung von Dokumenten jedweder Art ist es erforderlich, ein Textverarbeitungs-System auszuwählen, das den jeweiligen Ansprüchen so weit wie möglichst gerecht wird. Das im technischen Bereich etablierte LaTeX war einer der Kandidaten, die unter diesem Aspekt und am konkreten Beispiel dieser Arbeit näher untersucht wurden.

Die Untersuchung zeigte, dass LaTeX gegenüber dem etablierten Microsoft Word (hier: 2002) viele Vorteile, aber auch viele signifikanten Nachteile aufweist. Hauptsächlich ist die Tatsache zu nennen, dass es mit Word ebenso möglich ist, sauber strukturierte Dokumente zu erstellen, wie es mit LaTeX möglich ist, „semantischen Unsinn“ zu verfassen. Beides erfordert etwa ebensoviel Einarbeitungszeit und Disziplin vom Anwender. Gekonnt erfasste Word-Dokumente sind in keiner Beziehung schlechter als LaTeX-Dokumente, zumal früher eventuell auftretende Probleme wie der Export ins Standardformat PDF keine Herausforderung mehr darstellt.

LaTeX selbst ist ein Makropaket, das in der für den Druckereiprozess konzipierten Sprache TeX geschrieben wurde und dem Benutzer eine deutlich vereinfachte Kommandostruktur zur Verfügung stellt, ohne dabei die Flexibilität zu verlieren. Der Benutzer hat die Möglichkeit, auf die Voreinstellungen von LaTeX und weiterer Standardpakete zurückzugreifen und sich auf einige wenige grundlegende Befehle zu beschränken. Die Dokumente werden dabei mit einem beliebigen Texteditor erfasst.

Die von LaTeX oder besser gesagt TeX verwendeten Steuerzeichen wie \, { usw. müssen für die Ausgabe innerhalb des normalen Textes kodiert werden. Deutsche Anführungszeichen sind beispielsweise mit "` und "' zu erzeugen. Leerzeilen ergeben Absatzschaltungen, wogegen einfache Umbrüche ignoriert werden. Überschriften, Tabellen etc. werden mit einer Auswahl von mit \ beginnenden Kommandos erzeugt. Die unterschiedlichen Pakete (zum Beispiel `booktabs` für einfacher lesbare Tabellen) stellen jeweils weitere dieser Kommandos bereit.

Die Handhabung ist alles in allem etwa mit XML vergleichbar, wobei zugunsten der Kürze jedoch grundsätzliche Nachteile im Aufbau der TeX/LaTeX-Sprache in Kauf

genommen werden. Zahlreiche Sprachelemente, Regeln und Ausnahmen erschweren das „Erlernen“ der Sprache sehr und lassen den Einsatz für „normale“ Anwender nicht sinnvoll erscheinen.

```
% Dokumentenklasse "scrbook" des KOMA-Pakets,
% Voreinstellungen: 11pt, a4paper, twoside
\documentclass[12pt, oneside, BCOR1cm]{scrbook}
% Allgemeines Sprachpaket für dt. Anführungszeichen etc.
\usepackage{ngerman}
% Paket, das die direkten Eingabe von Umlauten ermöglicht
\usepackage[latin1]{inputenc}
% Paket für korrekte Trennung von Wörtern mit Umlauten
\usepackage[T1]{fontenc}
\begin{document}
  \title{Diplomarbeit \thanks{Ich danke ...}}
  \author{Thiemo Mättig}
  \date{Heute, \today}
  % Vordefinierte Titelseite erzeugen
  \maketitle
  \tableofcontents
  \listoffigures
  \listoftables
  \include{Theorie}
  \include{Praxis}
  \begin{appendix}
    \include{Anhang}
  \end{appendix}
\end{document}
```

Abbildung 24: LaTeX-Beispieldokument

Vor allem im Linux-Bereich existieren zahlreiche Lösungen, die diese ansonsten eher komplizierte Dokumentenerfassung unterstützen oder sogar WYSIWYG- bzw. Teil-WYSIWYG-Implementationen anbieten. Auch die am Markt immer stärker werdenden, funktional äquivalenten XML-Lösungen setzen sich immer weiter durch.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dresden, 1. Dezember 2003