Ein Hidden Star am Oracle-Firmament: Oracle Text

Dr. Frank Haney

Vorstellung

Selbständiger Oracle-Berater seit 2002

- OCP DBA
- Oracle University zertifizierter Trainer
- Auditleiter für geprüfte IT-Sicherheit

Administration

- Implementierung und Test
- Hochverfügbarkeit
- Backup und Recovery
- Migration und Upgrade
- Performance Diagnostik und Tuning
- Sicherheit

Applikationsentwicklung

- Design
- SQL und PL/SQL
- **Oracle Text**
- Oracle und XML
- Apex
- Objektrelationale Erweiterungen

Agenda

- Grundfragen des Information Retrieval
- Oracle Text – Überblick
- Speicherung, Filterung und Indizierung der Dokumente
- Volltextabfragen
- Visualisierung der Dokumente
- Mehrsprachigkeit
- Gemischte Abfragen (Text und Metadaten)

Nicht explizit:

- Administration und Performance
- Oracle Text und XML
- Oracle Text und JSON

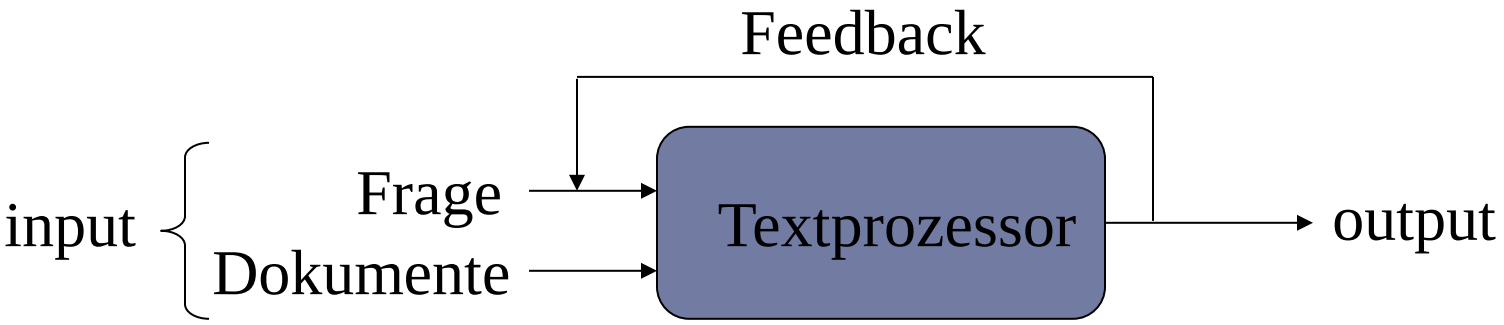
Zum Gegenstand

Moderne Informationsverarbeitung im Unternehmen muß neben der Verarbeitung strukturierter auch den Umgang mit unstrukturierten Daten (ca. 80%) ermöglichen. Stichworte dafür sind:

- Document Workflow
- Content Management
- Knowledge Management

Das bedeutet in erster Linie intelligentes Text Retrieval, denn dieses vermittelt einen Zugang zu den Inhalten von Texten in natürlicher Sprache, wobei es sich in der Regel um eine große Anzahl von Dokumenten handelt, die thematisch sehr vielfältig sein können.

Aufgaben von Text Retrieval Systemen



Typische Aufgaben (siehe auch die Tracks nach TREC):

- Ad hoc queries
- Known item search

  } Hitliste von Dokumenten

- Filtering — akkumulierte Menge von relevanten Dokumenten
- Novelty — Suche nach neuen, nichtredundanten Informationen
- Question answering — Antwort auf Frage, keine Menge von Dokumenten
- Plagiatsermittlung, Feststellung der Autorschaft, Trollsuche

TREC = **T**ext **RE**trieval **C**onference, seit 1992 jährlich veranstaltet vom NIST

Korrektheit vs. Relevanz

Bei einer Abfrage auf relationale Daten erwarte ich ein korrektes Ergebnis. Die Abfrage

```
SELECT gehalt FROM mitarbeiter WHERE name = 'KRAUSE';
```

liefert das Gehalt aller Mitarbeiter namens Krause. Wenn ich einen bestimmtem Krause will, dann muß ich anders fragen.

>>> TRUE/FALSE

Die Volltextabfrage

```
SELECT gehalt FROM mitarbeiter WHERE CONTAINS(lebenslauf,
'KRAUSE')>0;
```

kann irrelevante Zeilen zurückliefern, z.B. auch Mitarbeiter, deren Doktorvater Krause heißt, falls das für mich irrelevant ist.

>>> Relevanz

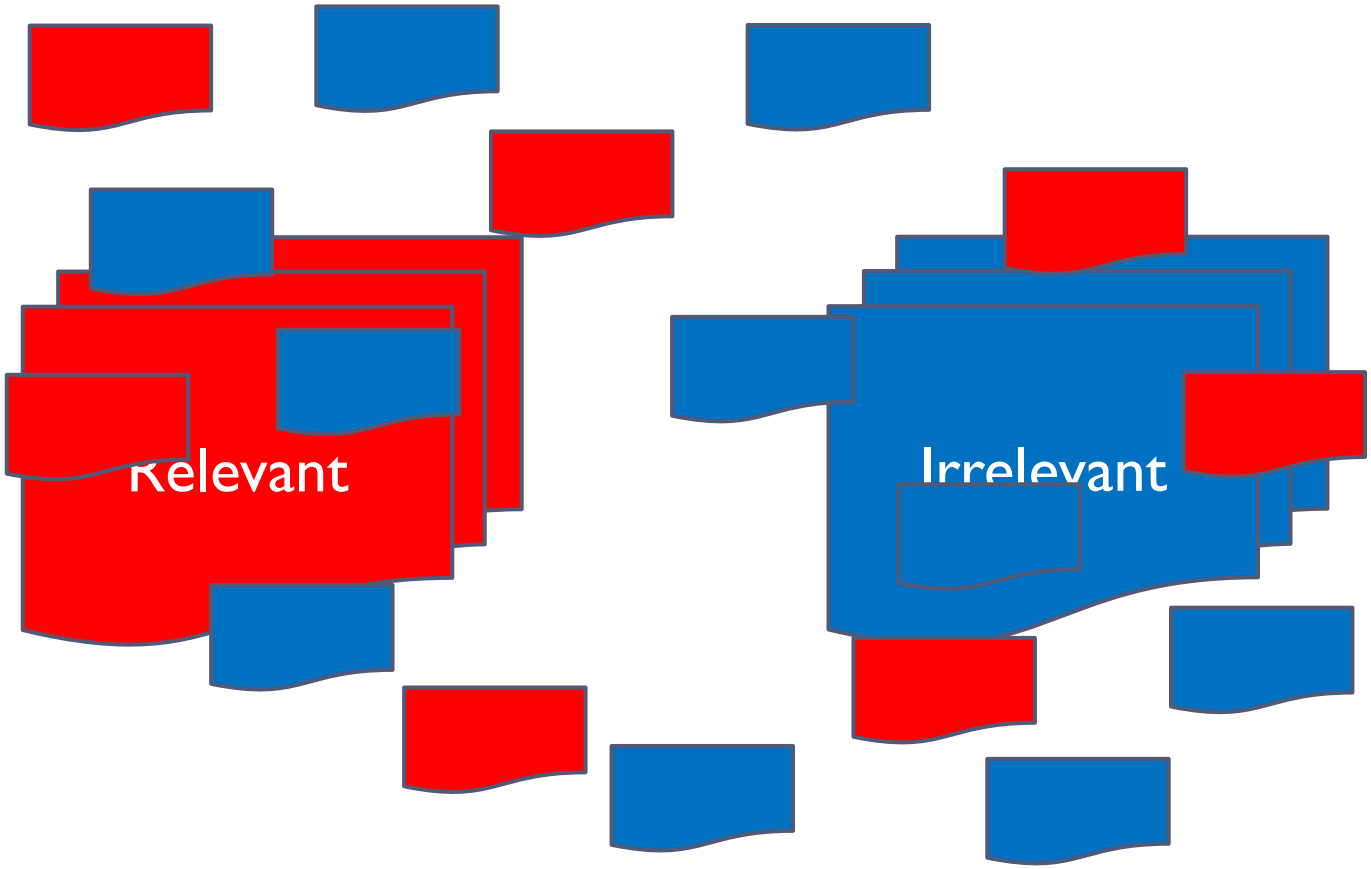
Präzision vs. Wiederfindung

Präzision (precision):	Verhältnis der zurückgegebenen relevanten Dokumente zur Gesamtzahl der zurückgegebenen Dokumente
Wiederfindung (recall):	Verhältnis der zurückgegeben relevanten Dokumente zur Gesamtzahl der relevanten Dokumente

Probleme:

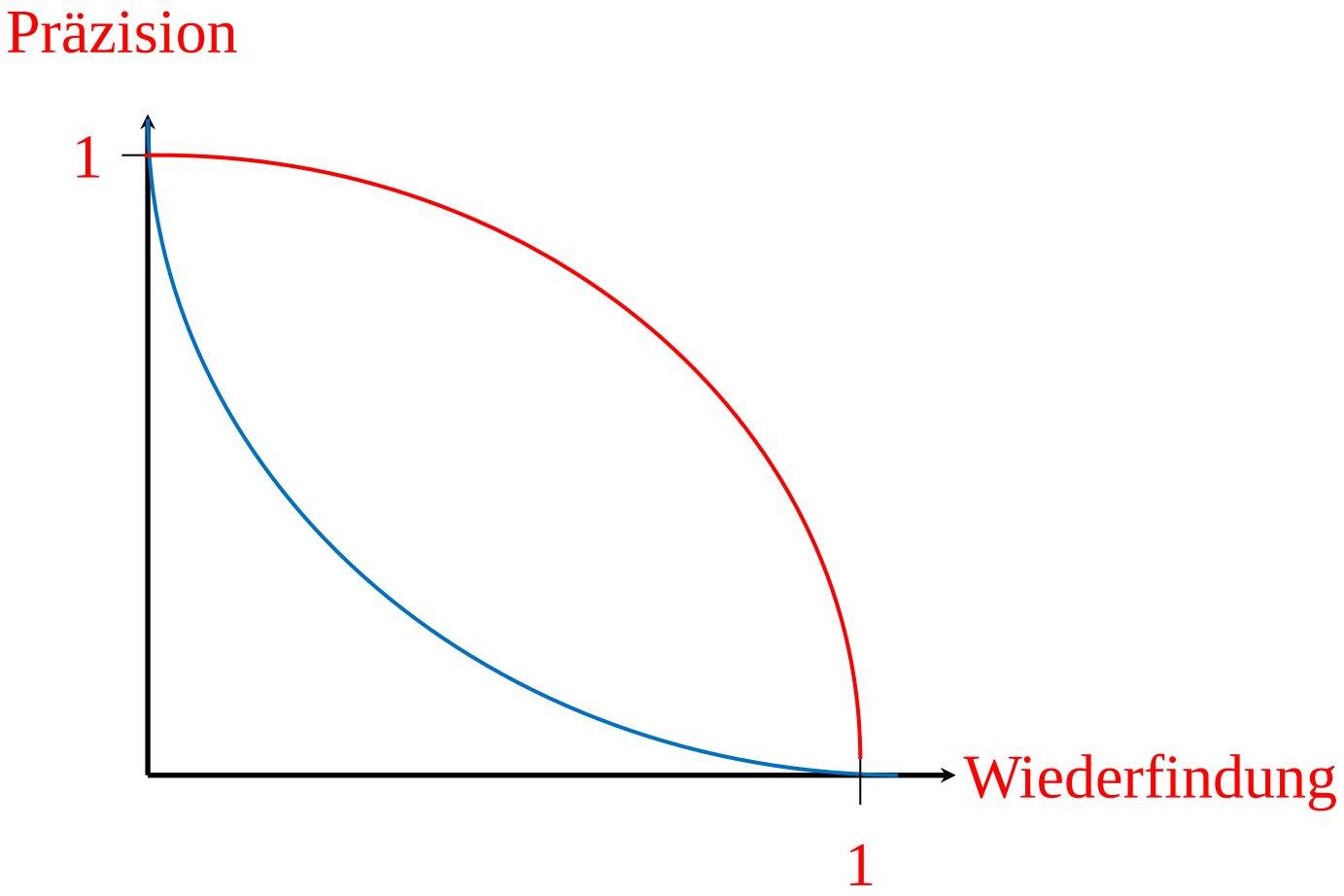
- Beurteilung der Relevanz (subjektive Seite)
- Wichtung der Fragen
- Position in der Hitliste (Qualität des Ranking – „Precision at n“)

Die ideale Suchmaschine



Die Sammlung

Die reale Suchmaschine



Texte in Datenbanken

	Datenhaltung	Zugriff/Textanalyse
Stufe 1	Dateisystem	Proprietäre externe Applikationen
Stufe 2	Datenbank ohne Strukturinformation (VARCHAR2, LOB)	SQL Oracle Text CONTAINS
Stufe 3	Datenbank mit Strukturinformation (XMLType, JSON)	SQL/XML Xquery JSON_TEXTCONTAINS

Datenbankgestütztes Text Retrieval

- Nur eine Plattform (Oracle DB), keine Schnittstellen zu externen Applikationen
- Transaktionskonzept relationaler RDBMS wird für die Manipulation von Texten nutzbar (konkurrierender Zugriff)
- Backup und Recovery der Dokumente gemeinsam mit der Datenbank
- Synchrone Speicherung und Auswertung von Text und Metadaten in Spalten einer Tabellenzeile oder als XML-Dokument
- Texte werden mit „Standard-SQL“ analysiert und mit PL/SQL manipuliert
- Texte werden für Aggregation und Data „Text“ Mining nutzbar
- Schnelle Applikationsentwicklung mit APEX und anderen DB Features

Alternativen

- Enterprise Information Retrieval mit Apache LUCENE
  - Open Source
  - JAVA-basierte Programmbibliothek
  - Grundlage: TOKEN-Extraktion
  - BigData-Anbindung (JSON)
- Projekte und Produkte, z.B.
  - SOLR
  - ElasticSearch
- Nachteile (aus heutiger Sicht)
  - Separates Backup & Recovery
  - Schwierige Integration mit relationalen Tabellen - also Unternehmensdaten (gemischte Abfragen)
  - Keine Transaktionen
  - Aufwändige Implementierung per Coding und Java API

Relevante Oracle Software

- ***Oracle Database*** – Server und Repository für strukturierte und unstrukturierte Daten (Text, Multimedia)

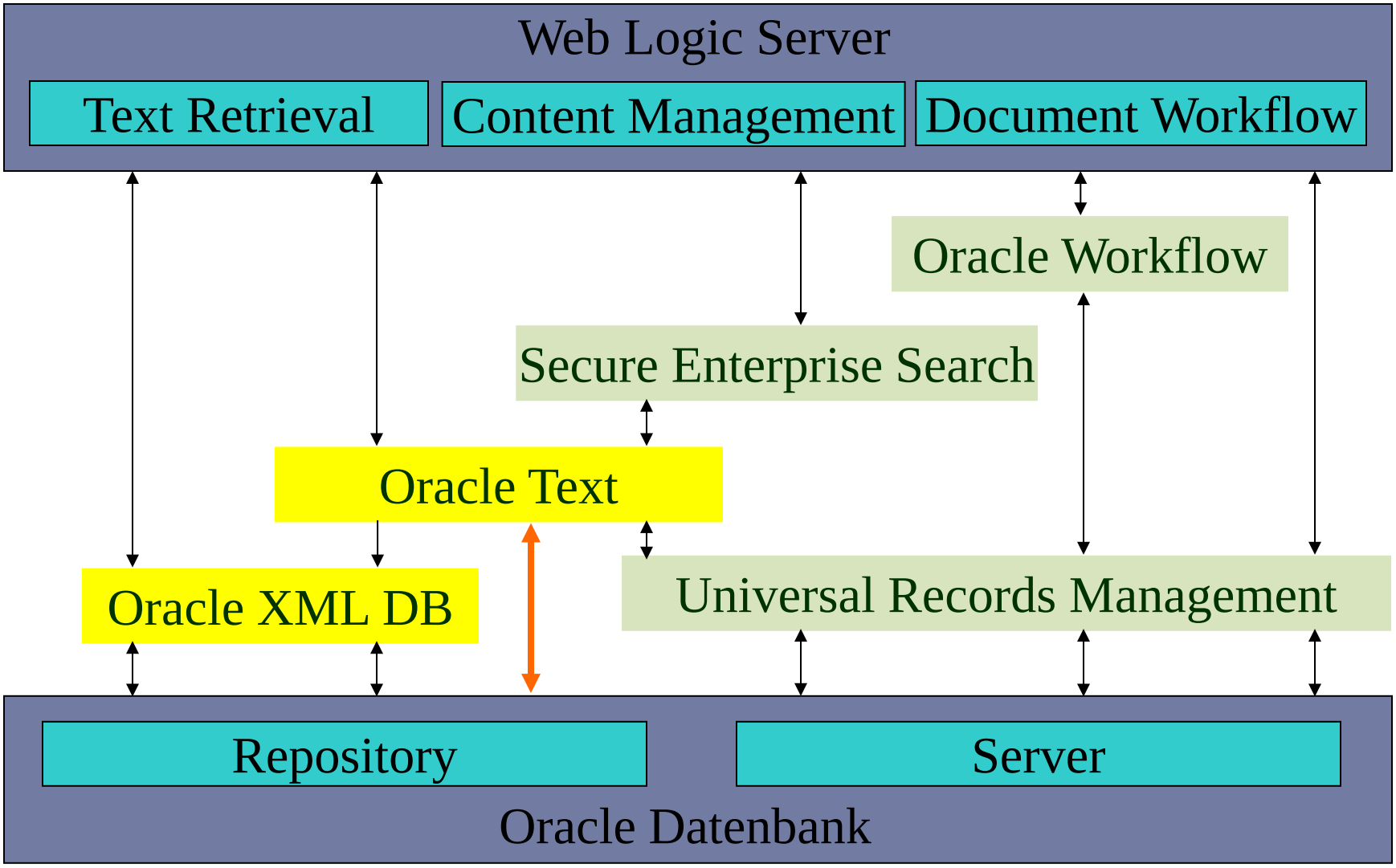
Als Bestandteile der Oracle-Datenbank-Software:

- ***Oracle Text*** – leistet die intelligente Textanalyse
- ***Oracle XML DB*** – semistrukturierte Ablage von Dokumenten in und außerhalb der Datenbank

Als Bestandteile des Oracle-Middleware-Portfolios:

- ***Oracle Secure Enterprise Search*** – portalgestützte Suchfunktion auf der Basis von Oracle Text
- ***Oracle WebCenter Content*** – integriert die verschiedenen Datenhaltungen und Formate in ein einheitliches Repository
- ***Oracle Workflow*** – Dokumenten- und Task-Management
- ***Oracle WebCenter Portal*** – Bereitstellung einer einheitlichen Oberfläche zur Bedienung und Administration.

Architektur (schematisch)



Oracle Text – Entwicklung

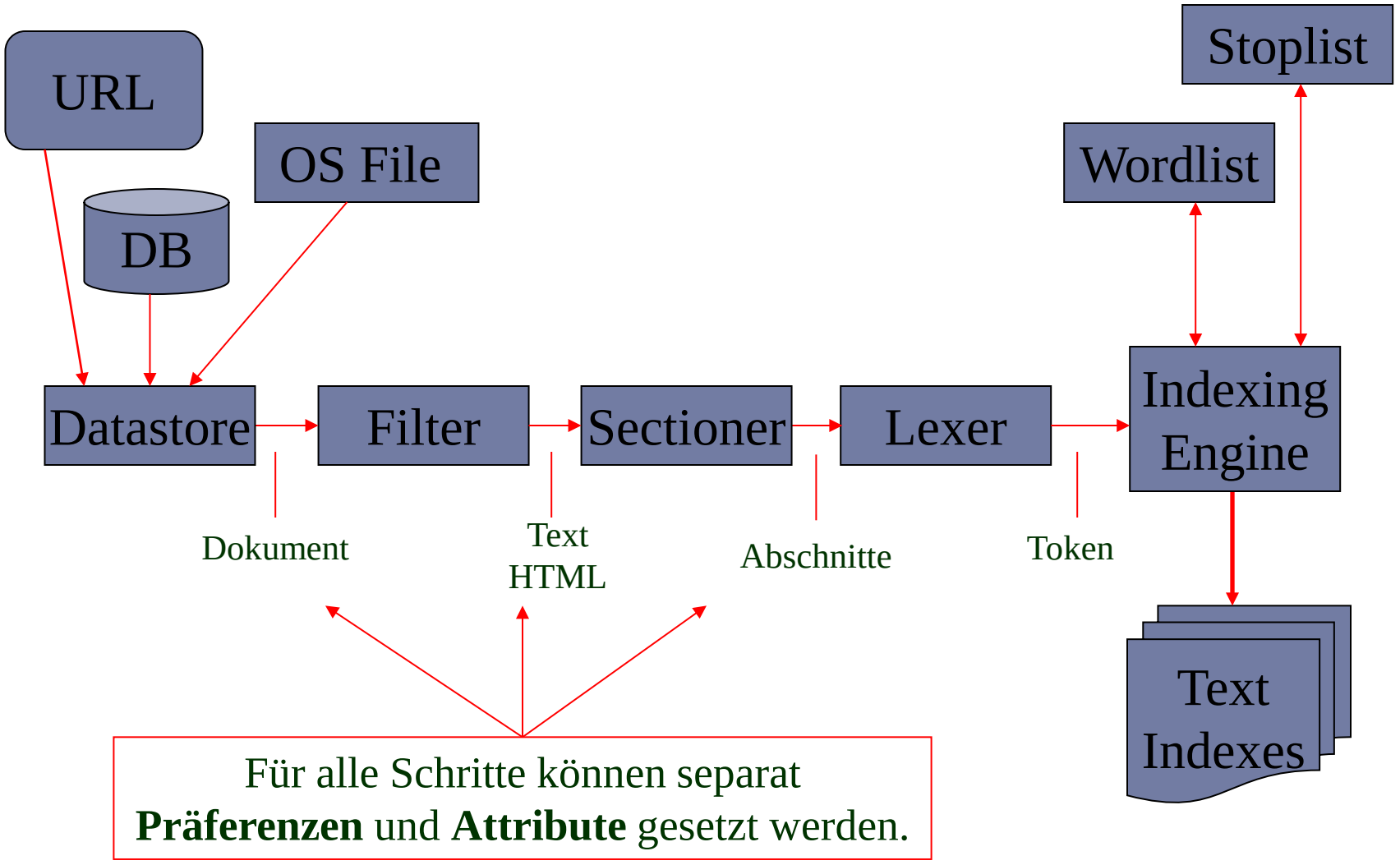
- Eingeführt mit ***Oracle 7***, dort als zusätzlich zu lizenzierende Software.
- Mit Oracle 8i in ***Oracle Intermedia*** als kostenpflichtige Option integriert.
- Seit Oracle 9i **Kernbestandteil** der Datenbank-Software und in allen Editionen frei verfügbar, auch in der Express Edition.
- Weiterentwicklungen von Release zu Release, aber kaum Änderungen am Grundbestand

Oracle Text basiert auf der Erweiterungsschnittstelle von Oracle (*Data Cartridge* genannt)

Oracle Text wird serverseitig ausgeführt. Die Grundlagen bilden spezifische SQL-Features und PL/SQL-Routinen.

Oracle Text blüht etwas im Verborgenen, es gibt kein Buch und auch keinen Kurs bei Oracle University.

Der Prozeß der Indizierung



Indextypen

1. CONTEXT-Index: Für große kohärente Dokumentensammlungen verschiedener Formate (z. B. extern: Word, HTML, XML, PDF; intern: BLOB, CLOB, VARCHAR2, BFILE) ➡ CONTAINS
2. CTXCAT-Index: Für kurze Texte und gemischte Abfragen (strukturierte und unstrukturierte Daten), eingeschränkte Abfragesyntax ➡ CATSEARCH
3. CTXRULE-Index: Klassifikation und Routing von Dokumenten, der Index wird über eine Tabellenspalte mit Abfragestrings erzeugt ➡ MATCHES
4. CTXXPATH-Index: Für XMLType-Spalten, beschleunigt XML-Abfragen (desupported in 12c) ➡ ExistsNode

CONTEXT-Index – Syntax

```
CREATE INDEX text_idx ON schema.table(column)
INDEXTYPE IS CTXSYS.CONTEXT [ONLINE]
[PARAMETERS('paramstring')];
```

Parameter (Präferenz)	Bedeutung
DATASTORE	Wie und wo werden die Dokumente gespeichert?
FILTER	Wie werden die Dokumente nach Text oder HTML gefiltert?
LEXER	Welche Sprache wird bei der Token-Zerlegung verwendet?
WORDLIST	Wie werden Wordstamm- und Fuzzy-Abfragen gehandhabt?
STORAGE	Wie erfolgt die Speicherung der Index-Objekte?
STOPLIST	Welche Worte werden nicht indiziert?
SECTION GROUP	Wie werden Abschnitte der Dokumente definiert und dafür spezifische Abfragen ermöglicht?

Voreingestellte Präferenzen

Präferenz	Voreinstellungen (Auswahl)
DATASTORE	DIRECT_DATASTORE, MULTI_COLUMN_DATASTORE, DETAIL_DATSTORE, FILE_DATASTORE, URL_DATASTORE, USER_DATASTORE
FILTER	AUTO_FILTER, INSO_FILTER, NULL_FILTER
LEXER	AUTO_LEXER, BASIC_LEXER, MULTI_LEXER, WORLD_LEXER, USER_LEXER
WORDLIST	BASIC_WORDLIST
STORAGE	BASIC_STORAGE
STOPLIST	DEFAULT_STOPLIST, EMPTY_STOPLIST
SECTION GROUP	NULL_SECTION_GROUP, HTML_SECTION_GROUP, JSON_SECTION_GROUP, AUTO_SECTION_GROUP, PATH_SECTION_GROUP, XML_SECTION_GROUP

Attribute der Präferenzen (Beispiele)

FILE_DATASTORE	path	string
BASIC_LEXER	continuation	character (z.B. - \)
	punctuations	character (. ,)
	whitespace	character
	newline	NEWLINE, CARRIAGE_RETURN
	base_letter	YES/NO
	mixed_case	YES/NO
	composite	DEFAULT, GERMAN, DUTCH
	alternate_spelling	NONE, GERMAN, DANISH, SWEDISH
	new_german_spelling	YES/NO
BASIC_WORDLIST	stemmer	AUTO, NULL, ENGLISH, GERMAN
	fuzzy_match	GENERIC, AUTO, ENGLISH, GERMAN
	substring_index	FALSE/TRUE
	prefix_index	FALSE/TRUE
	wildcard_maxterms	20000

CONTEXT-Index – Abfragesyntax

```
SELECT id, SCORE(1) FROM table WHERE
CONTAINS(column, 'query_string',1)>0;
```

Beispiele für Abfragestrings:

'Thüringen ~ Erfurt'	NOT-Operator
'DB = Datenbank'	EQUIValence-Operator
'Oracle, Text, Präferenz*3'	ACCUMulate-Operator

SCORE = 3f(1+log(N/n))

f = Häufigkeit des Terms im Text
N = Gesamtzahl der Texte
n = Gesamtzahl der Texte, die den Term enthalten

Ergebnisse:

ID	**SCORE(1)**
2	70
3	35
1	4

Durchsucht wird nur der Index, nicht die Textspalte!

Achtung: Der Wert von SCORE ist keine Anzahl der Fundstellen, sondern ein von 1 bis 100 laufendes, nach *Saltons Formel* berechnetes Verhältnismaß: Damit ein Dokument hoch bewertet wird, muß es oft im Dokument und selten in der ganzen Sammlung vorkommen. (Inverse frequency scoring) Häufiges Vorkommen in allen Dokumenten wird von der Text Engine als Rauschen gewertet.

Ausgewählte Abfrageoperatoren

Operator	Funktion
FUZZY	Ähnliche Worte (Meier, Mayer, Meuer)
NEAR (;)	Zwei Begriffe in bestimmtem Abstand
stem ($)	Begriffe mit gleichem Wortstamm
WITHIN	Suche in einer Abschnittsgruppe
INPATH	Suche in XML-Dokumenten
SQE	Stored Query Expression
Narrower Term (NT)	Suche nach Begriff und Unterbegriffen *
ABOUT	Thematische Suche **

* Setzt einen bei der Text Engine registrierten Thesaurus voraus

** Erfordert die Kompilierung des Thesaurus zur Knowledge Base

Erweiterte Funktionalität

Filterung	Filterung von über 200 Formaten in HTML oder Text [1]
Highlighting	Ausgabe von OFFSET und LENGTH der Fundorte
Markup	Markieren der gefundenen Stellen
Navigation	Einfügen von Navigation-Tags an den Fundstellen
Snippets	Terme in ihrem Kontext (Konkordanz des Texts)
Gist	Thematische Zusammenfassung von Texten [2]
Themes	Ausgabe aller Themen eines Dokuments [2]
Word Browsing	Ausgabe benachbarter Einträge im Index

1) Dazu gehören alle gängigen Textverarbeitungs- und DTP-, Tabellenkalkulations-, Präsentations-, Datenbank-, Archivierungs- und Mail-Formate. Siehe Dokumentation

2) Setzt die Existenz einer kompilierten Knowledge Base voraus

Beispiel Markup und Navigation

```
begin
   ctx_doc.markup(index_name => 'TEXT_IND_1',
   textkey => '1',
   text_query => 'history',
   restab => 'MARKUP_TEST',
   query_id => '1',
   tagset => 'HTML_NAVIGATE');
end;
```

The < **history** > of text retrieval is almost as long as the < **history** > of the written word. We know that Pliny the Elder (died 79 A.D.) used a Table of Contents structure to help readers navigate his mammoth "The Natural < **History** > in 37 Books". In this work, he refers to Valerius Soranus who, in the second century BC, was the first in Latin literature to use a Table of Contents. And in the third century B.C., Kallimachos of Kyrene created the first subject catalog to keep track of the scrolls at the Great Library of Alexandria in Egypt.

Snippets

Syntax

```
CTX_DOC.SNIPPET(
    index_name IN VARCHAR2,
    textkey IN VARCHAR2,
    text_query IN VARCHAR2,
    starttag IN VARCHAR2 DEFAULT '<b>',
    endtag IN VARCHAR2 DEFAULT '</b>',
    entity_translation IN BOOLEAN DEFAULT TRUE,
    separator IN VARCHAR2 DEFAULT '<b>...</b>'
)
return varchar2;
```

Beispiel

```
select ctx_doc.snippet('CLOB_TAB_IDX', '1',
'microsoft near patch') from dual;
```

Ergebnis

```
SNIPPET
--------------------------------------------------------------------
refer to [NOTE:77627.1].
<b>Microsoft</b> Service Pack and Oracle <b>Patch</b> Set Support
================
```

Thesaurus* (Ausschnitte)

Christianity
  BT world religions
  NT Mormonism
  NT Roman Catholicism
  NT evangelism
  RT Bible
  RT Israel
  RT death and burial
DBMS - database management system
  BT databases
Euclidean geometry
  BT geometry
IT&T
  SYN information technology

* Verfügbar nur für Englisch

Beispiel ABOUT-Abfrage

```
SELECT id, score(1) FROM test2 WHERE CONTAINS
(dokumente, 'ABOUT(history of text retrieval)',1)>0;
```

ID	SCORE(1)
4	4
2	53
1	54

Vergleiche dagegen:

```
SELECT id, score(1) FROM test2 WHERE CONTAINS
(dokumente, 'history of text retrieval',1)>0;
```

ID	SCORE(1)
2	20

Gist (Zusammenfassung) / thematisch

```
begin
   ctx_doc.gist(index_name => 'TEXT_IND_1',
                textkey    => '2',
                restab     => 'GIST_TEST',
                pov        => 'history',
                query_id   => 1,
                glevel     => 'S');
end;
```

We start by reviewing the history of text retrieval over the last 2,000+ years, focusing on the trends of the last decade "A Short History of Text Retrieval" describes the major milestones in text retrieval, from the first subject catalog in the third century B.C. A Short History of Text Retrieval The history of text retrieval is almost as long as the history of the written word.) used a Table of Contents structure to help readers navigate his mammoth "The Natural History in 37 Books". All sorts of information - news articles, Christmas presents, historical references, investment advice, business and personal e-mails, technical support bulletins, competitive data - was "out there somewhere", reachable, at least in theory, from a search box. We started by reviewing the history of text retrieval over the last 2,000+ years, focusing on the trends of the last decade How Information Retrieval Started, American Society of Indexers (http://www.asindexing.org/site/history.shtml)

Sprachauswahl – Grundeinstellungen

- Oracle nimmt an, daß die Dokumente in der Sprache vorliegen, die während des Setup der Datenbank angegeben wurde.
- Die Dokumente werden dann standardmäßig mit dem BASIC_LEXER indiziert. Dieser ist für Sprachen mit Leerzeichen als *Delimiter* gedacht: Deutsch, Englisch etc.
  Typische Einstellungen sind (siehe Lexer-Präferenzen):
  - Groß- und Kleinschreibung
  - Base Letter Conversion
  - Alternate Spelling
  - Composite Word Indexing
- Der MULTI_LEXER ist für mehrsprachige Dokumentensammlungen
- Der WORLD_LEXER kann automatisch die Sprache im Dokument erkennen.
- Mit dem USER_LEXER läßt sich ein Lexer für eine bestimmte Sprache implementieren.

Ändern der Basissprache

- Ausführen des Skripts
  `$ORACLE_HOME/ctx/admin/defaults/drdefxxx.sql`
  für deutsch z.B. `drdefd.sql`
- Bedeutet die Änderung des Lexer-Standards für deutsch auf
  - ALTERNATE_SPELLING: GERMAN
  - COMPOSITE: GERMAN
  - MIXED_CASE: YES
- Änderung des Wordlist-Standards für deutsch auf
  - STEMMER: GERMAN
  - FUZZY_MATCH: GERMAN
- Generierung der entsprechenden (deutschen) Stoplist

Einsatz des MULTI_LEXER

- Anlegen der Dokumententabelle mit einer Spalte zur Sprachidentifizierung vom Datentyp VARCHAR2
  Die Sprache wird mit einem gültigen Code entsprechend ISO 639-2 angegeben.
- Wert AUTO ermöglicht die Spracherkennung für jede unterstütze Sprache
- Erzeugen von Sub-Lexern für jede vorkommende Sprache
- Einstellung der Attribute für diese Sub-Lexer
- Erzeugen einer MULTI_LEXER-Präferenz
- Hinzufügen der Sub-Lexer zum MULTI_LEXER mit ctx_ddl.add_sub_lexer (Ein Default kann festgelegt werden.)
- Anlegen eines Index unter Verwendung der MULTI_LEXER-Präferenz
- Sinnvoll ist das Anlegen einer Stoplist vom Typ MULTI_STOPLIST

Gemischte Abfragen – Überblick

Häufig soll nicht nur der Volltext durchsucht, sondern auch nach den Metadaten gefiltert werden.

Verschiedene Möglichkeiten:

- Zwei getrennte Prädikate - Text und relational
  - inperformant
- CTXCAT-Index
  - CATSEARCH: eingeschränkte Abfragesyntax gegenüber CONTAINS
  - keine LOB-Speicherung
- Auszeichnung der Metadaten im Dokument selber (Sections)
- MDATA Section (Metadaten)
- SDATA Section (Strukturierte Daten)
- Composite Domain Index (Mixed Query)

Auszeichnung von Section Groups

- Zone Section
  - Wird zusammen mit dem Text indiziert
  - Gut bei häufigem Auftreten im Dokument
  - Können einander einschließen oder überlappen
- Field Section
  - Wird getrennt indiziert
  - Gut bei einmaligem Auftreten der Zone im Dokument
  - Kann für den Rest des Dokuments sichtbar gemacht werden
  - Dürfen einander nicht einschließen oder überlappen
- Stop Section – wird nicht indiziert
- MDATA Section – nichtindizierte Metadaten
- SDATA Section – Auszeichnungen werden wie relationale Spalten behandelt
- NDATA Section – für unscharfe Namenssuche
- Voreingestellte HTML, XML und JSON Section Groups
- Satz oder Abschnitt in Textdokumenten

Auszeichnung von Metadaten

- Grundlage ist die Möglichkeit Sektionen im Dokument auszuzeichnen (HTML, XML)
- Die Tags der Sektionen werden für die Indizierung benutzt.
- Definition

```
ctx_ddl.create_section_group('htmgroup',
  'HTML_SECTION_GROUP');
ctx_ddl.add_zone_section('htmgroup', 'heading',
  'H1');
```

- Indizieren unter Angabe der Section Group
- Verwendung in Abfragen

```
SELECT id FROM documents WHERE CONTAINS(text,
  'database AND oracle WITHIN heading')>0;
```

Nachteile dieser Methode

- Jedes Ändern eines Attributs oder Tag-Inhalts erfordert das Re-Indizieren des Dokuments.
- Sonderzeichen in den Metadaten (z.B. Namen) bzw. zusammengesetzte Inhalte beeinflussen die Token-Zerlegung und damit die Effizienz der Abfragen
- Besser
  - Field Section
  - MDATA
  - SDATA

SDATA Section

- Im Dokument werden Sektionen definiert, die als relationale Daten indiziert werden.
- Verschiedene Kombinationen von Text und relationalen Prädikaten können abgefragt werden
- Updates auf den SDATA sind möglich `CTX_DDL.UPDATE_SDATA`
- Tabelle anlegen und befüllen

```
CREATE TABLE items (id NUMBER PRIMARY KEY, doc VARCHAR2(4000));

INSERT INTO items VALUES (1, '<description> Honda Pilot
  </description> <category> Cars & Trucks </category> <price>
  27000 </price>');
INSERT INTO items VALUES (2, '<description> Toyota Sequoia
  </description> <category> Cars & Trucks </category> <price>
  35000 </price>');
…
COMMIT;
```

SDATA anlegen und indizieren

- Präferenz

```
CTX_DDL.CREATE_SECTION_GROUP('my_sec_group',
  'BASIC_SECTION_GROUP');
  CTX_DDL.ADD_SDATA_SECTION('my_sec_group',
  'category', 'category', 'VARCHAR2');
  CTX_DDL.ADD_SDATA_SECTION('my_sec_group', 'price',
  'price', 'NUMBER');
```

- Index anlegen

```
CREATE INDEX items$doc ON items(doc) INDEXTYPE IS
  CTXSYS.CONTEXT PARAMETERS('SECTION GROUP
  my_sec_group');
```

- Abfragen

```
SELECT id, doc FROM items WHERE contains(doc, 'Toyota
  AND SDATA(category = ''Cars & Trucks'') AND
  SDATA(price <= 40000 )') > 0;
```

Mixed Query (Composite Domain Index)

Index anlegen

```
CREATE INDEX comp_ind ON sh.customers(cust_first_name)
INDEXTYPE IS ctxsys.context
FILTER BY cust_id
ORDER BY cust_year_of_birth;
```

Abfrage

```
SELECT /*+ first_rows(10) */ cust_id
FROM (select cust_id, cust_first_name,cust_year_of_birth from sh.customers
WHERE contains (cust_first_name, 'A% or D% or N% or B%',1)>0  AND cust_id>100000
ORDER BY cust_year_of_birth, score(1))
WHERE rownum<10;
```

Ausführungsplan

```
| Id  | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |     1 |    13 |     5  (20)
|*  1 |  COUNT STOPKEY                |           |       |       |
|   2 |   VIEW                        |           |     1 |    13 |     5  (20)
|*  3 |    SORT ORDER BY STOPKEY      |           |     1 |    25 |     5  (20)
|   4 |     TABLE ACCESS BY INDEX ROWID| CUSTOMERS |     1 |    25 |     4   (0)
|*  5 |      DOMAIN INDEX             | COMP_IND  |       |       |     4   (0)
-------------------------------------------------------------------------------
5 - access("CTXSYS"."CONTAINS"("CUST_FIRST_NAME",'A% or D% or N% or B%',1)>0)
    filter("CUST_ID">100000)
```

Unscharfe Namenssuche mit NDATA

- Analog zu MDATA, aber für unscharfe Suche optimiert
- Besser geeignet als FUZZY (Namen sind orthogonalisiert im Dokument, es gibt aber alternative Schreibweisen)
- Ein Thesaurus alternativer Namen (Spitznamen) kann verwendet werden.
- Funktioniert mit Auszeichnungen der Metadaten im Text und mit relationalen Spalten

NDATA mit Auszeichnung

Vorbereitung

```
create table names (
  id number(10),
  name varchar2(200));

insert into names values (1, '<name>Max Mustermann</name>');
insert into names values (2, '<name>Larry Ellison</name>');
insert into names values (3, '<name>Ulrike Schwinn</name>');
insert into names values (4, '<name>Carsten Czarski</name>');
insert into names values (5, '<name>Günther Stürner</name>');

begin
  ctx_ddl.create_section_group('name_sg', 'BASIC_SECTION_GROUP');
  ctx_ddl.add_ndata_section('name_sg', 'name', 'name');
end;
/

create index ft_names on names (name)
indextype is ctxsys.context
parameters ('section group name_sg');
```

Resultat

```
select * from names where contains(name,'NDATA(name, Saarski Karsden)')>0;
 ID   NAME
----  ------------------------------
   4  <name>Carsten Czarski</name>
```

NDATA mit Multi Column Datastore

- Tabelle für die Namenssuche

```
create table mitarbeiter(vorname varchar2(80),
  name varchar2(80));
insert into mitarbeiter values('John', 'Smith');
commit;
```

- Präferenz Multi Column Datastore

```
ctx_ddl.create_preference('nameds',
  'MULTI_COLUMN_DATASTORE');
ctx_ddl.set_attribute('nameds', 'columns',
  'vorname, name');
```

- Erzeugt für die Indizierung ein virtuelles Dokument

```
<VORNAME> John </VORNAME>
<NAME> Smith </NAME>
```

NDATA Sections und relationale Spalten

- Präferenz für die Name Sections

```
ctx_ddl.create_section_group('namegroup',
  'BASIC_SECTION_GROUP');
ctx_ddl.add_ndata_section('namegroup', 'VORNAME',
  'VORNAME');
ctx_ddl.add_ndata_section('namegroup', 'NAME',
  'NAME');
```

- Index erstellen

```
create index mitarbeiteridx on
  mitarbeiter(vorname) indextype is ctxsys.context
  parameters('section group namegroup datastore
  nameds');
```

Mehr Informationen

- Zum Nach- und Weiterlesen:
  - Oracle Text Reference
  - Oracle Text Application‘s Developer Guide
  - http://www.oracle.com/technetwork/database/enterprise-edition/index-098492.html
  - http://oracle-text-de.blogspot.com/
  - http://trec.nist.gov/
  - C.J. van Rijsbergen: Information Retrieval, Glasgow 1979
  - G. Salton, M. McGill: Introduction to Modern Information Retrieval, McGraw-Hill 1983
- Training
  - http://training.ordix.de/siteengine/action/load/kategorie/Oracle/nr/1824/index.html
  - Nächste Termine: 21.03. – 23.03.2016, 09.05. – 11.05.2016 …

Vielen Dank!

Q & A

Dr. Frank Haney
info@haney.it
Tel.: 03641-210224

ORACLE | CERTIFIED PROFESSIONAL