"Googeln" in der Datenbank: Oracle TEXT

Dr. Frank Haney
Consultant
Jena

Schlüsselworte:

Datenbankentwicklung, Text Retrieval, Volltextindizierung

Zum Gegenstand

Wenn man an Unternehmensdaten dankt, dann zunächst meist in Form strukturierter Daten. Ein Mitarbeiter hat einen bestimmten Namen, bekommt ein Gehalt etc. Die Daten sind in Zeilen und Spalten relationaler Tabellen abgelegt. Diese Daten kann man mit SQL effizient durchsuchen, weil die WHERE-Klausel den Zugriff auf einzelne Datensätze ermöglicht. Dabei wird häufig vergessen, daß der Großteil der Daten in Unternehmen unstrukturiert ist, also in Form von Dokumenten (Texten) unterschiedlichster Formate anfällt. Diese unstrukturierten Daten im System nicht nur einfach mit ihren Metadaten und vielleicht noch Schlagworten abzulegen, sondern mit Mitteln der Datenbank recherchierbar zu machen, ist eine wichtige Aufgabe für moderne Informationssysteme.

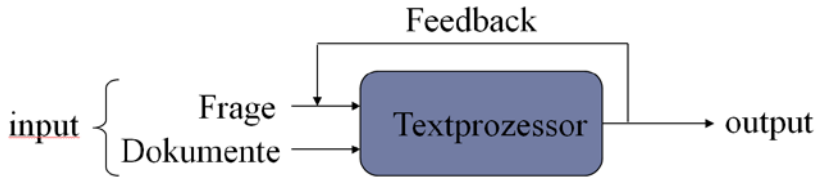
Dafür gibt es bei Oracle seit der Version 8 eine datenbankbasierte Volltextsuche. Diese ist integraler Bestandteil aller Editionen und braucht nicht zusätzlich lizenziert zu werden. Von Release zu Release wird sie weiterentwickelt und um neue, attraktive Features angereichert. Trotzdem überrascht es, daß Oracle Text derzeit eher im Verborgenen blüht, was sich in der überschaubaren Anteilnahme durch die Community zeigt. Eine dem Thema gewidmete SIG der DOAG ist vor Jahren sanft entschlafen. Man kann sogar den Eindruck gewinnen, daß selbst die Firma Oracle nur wenig zur Propagierung tut. Es gibt außer der Dokumentation kein Buch dazu und auch keinen Kurs bei Oracle University

Grundlagen des Text Retrieval

Stichworte für den Umgang mit unstrukturierten Daten im Unternehmen 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. Schematisch kann man ein Text Retrieval System folgendermaßen darstellen:



Wie wichtig das Thema ist, sieht man daran, daß in den USA das National Institute of Standards and Technology (NIST) seit 1992 jährlich TREC (die **T**ext **RE**trieval **C**onference) veranstaltet. Diese ist nicht nur ein Erfahrungsaustausch, sondern die Hersteller von Text Retrieval Systemen treten dort mit ihren Produkten bezüglich bestimmter Fragestellungen an standardisierte Dokumentensammlungen gegeneinander an. Solche Fragestellungen sind z.B.:

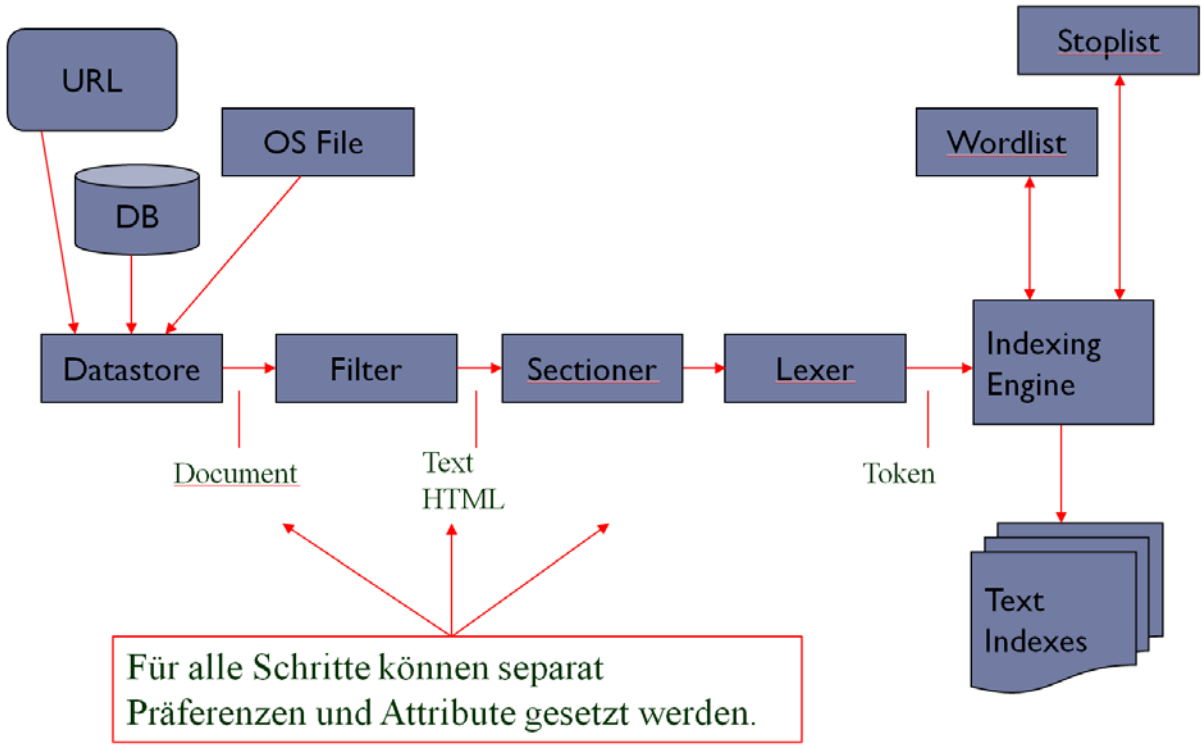
Track von TREC	**Wonach wird gesucht?**
Ad hoc queries	Hitliste von Dokumenten
Known item search	Hitliste von Dokumenten
Filtering	akkumulierte Menge von relevanten Dokumenten
Novelty	Suche nach neuen, nichtredundanten Informationen
Question answering	Antwort auf Frage, keine Hitliste von Dokumenten
Relevance Feedback	Automatische Relevanzbeurteilung

Worauf kommt es nun beim Text Retrieval im Unterschied zu „normalen“ Datenbankabfragen an? Zunächst einmal geht es bei Textabfragen weniger um die Korrektheit des Ergebnisses, sondern um die Relevanz des Dokuments. Die Text Engine des RDBMS berechnet für die Relevanz einen Score auf der Basis eingebauter Algorithmen, z.B. Salton’s Formula. Das Stichwort ist hier *Inverse Frequency Scoring*. Ein hoher Score wird dann erzielt, wenn der Suchbegriff häufig im Dokument, aber selten in der Dokumentensammlung vorkommt. Dabei ist es in letzter Konsequenz Sache des Nutzers, die Relevanz zu beurteilen.
Ein gravierendes Problem ist das Verhältnis von *Präzision* (Precision) und *Wiederfindung* (Recall). Das kann man als die Unschärferelation des Text Retrieval ansehen. Wenn man eine möglichst hohe Präzision haben möchte, d.h. nur relevante Dokumente werden zurückgegeben, bleiben viele andere relevante Dokumente außen vor, weil sie im Resultat nicht erscheinen. Die Wiederfindung ist gering. Wenn dagegen die Anwendung dahingehend optimiert wird, daß möglichst alle relevanten Dokumente angezeigt werden, wird das damit erkauft, daß auch viele irrelevante Dokumente ausgegeben werden, die Präzision also niedrig ist. Das ist letztlich ein in letzter Konsequenz nicht auflösbares Dilemma, vor dem sicher schon jeder Anwender einer Suchmaschine gestanden hat.

Nach diesen allgemeinen Bemerkungen sollen jetzt die Grundlagen von Oracle Text erläutert werden.

Der Volltextindex als Grundlage des Text Retrieval

Um Volltextsuche zu ermöglichen, muß ein Volltextindex auf der Dokumentensammlung angelegt werden. Dieses ist ein Domain Index, der aus einer Vielzahl einzelner Datenbankobjekte besteht, die zusammen die gewünschte Funktionalität gewährleisten. Der Prozeß der Indizierung läßt sich schematisch folgendermaßen darstellen:



Die einzelnen Schritte bedeuten:
Datastore: Die zu indizierende Dokumentensammlung wird zugänglich gemacht. Die Dokumente werden in die Datenbank geladen oder bleiben an ihrem ursprünglichen Ort (Betriebssystem, Internet etc.).
Filter: Binärdokumente der verschiedenen Formate werden in Text oder HTML umgewandelt.
Sectioner: Die Abschnittsgruppen des Dokuments werden identifiziert. Das ermöglicht später dann die Suche auf der Basis der inneren Hierarchie des Dokuments.
Lexer: Als entscheidende Voraussetzung für die Indizierung wird der Text in Tokens (Wörter oder Wortbestandteile) zerlegt.
Indexerstellung: Die Fundstellen für die einzelnen Wörter werden ermittelt. Dabei werden in der Stoplist enthaltene Wörter ausgeschlossen.
Für den Prozeß der Indizierung sind sehr differenzierte Einstellungen möglich, die hier nicht im Detail erläutert werden können.

Die Syntax für die Indexerstellung ist einfach. Die normale Syntax wird um die Angabe des Index-Typ (`INDEXTYPE IS` …) und die vielen angebbaren Präferenzen und Attribute erweitert. Es gibt drei Index-Typen im Umfeld von Oracle Text:
CONTEXT: Das ist der klassische Volltextindex für die entsprechenden Abfragen. Er ist gedacht für große kohärente Dokumentensammlungen verschiedener Formate (z. B. Word, HTML, XML, PDF; intern: BLOB, CLOB, VARCHAR2, BFILE). Der zugehörige Abfrageoperator ist CONTAINS.
CTXCAT: Hierbei handelt es sich um einen kombinierten Index auf Text und Metadaten für sogenannte Katalogabfragen. Er ist geeignet für kurze Texte und gemischte Abfragen (struk-

turierte und unstrukturierte Daten) und verfügt gegenüber dem CONTEXT-Index über eine eingeschränkte Abfragesyntax. Der Abfrageoperator ist CTXCAT.
CTXRULE: Dieser Index-Typ ist für Dokumentenklassifizierung und -routing bestimmt. Er wird nicht auf eine Dokumentensammlung, sondern auf eine Menge von Such-Strings angewendet. Der zugehörige Abfrageoperator ist MATCHES.

Der wichtigste Bestandteil des Domain-Index vom Typ CONTEXT ist die Tabelle DR$<index-name>$I. Diese enthält in einer BLOB-Spalte die Tokens. Bei Änderungen auf der Dokumentensammlung (neue, geänderte oder gelöschte Dokumente) muß der Index erneuert werden. Dieses Refresh ist auf verschiedene Art möglich. Ein REBUILD des Index geht natürlich immer, ist aber bei CONTEXT-Indizes nicht die bevorzugte Methode. Dort unterscheiden wir Synchronisation und Optimierung. Synchronisation fügt die Token-Einträge für die neuen Dokumente einfach am Ende des Index an. Das ist manuell mit der Prozedur `ctx_ddl.sync_index`, zeitbasiert oder transaktional möglich. Bei viel DML auf der Dokumentenspalte führt dies natürlich zu einer Aufblähung des Index und möglicher Performancedegradation. Deswegen ist es wichtig, den Index von Zeit zu Zeit zu optimieren. Damit werden die neuen und die alten Einträge integriert, und der Index wieder kompakt. Man macht das manuell mit `ctx_ddl.sync_optimize`. Dafür sollte man aber ein Wartungsfenster wählen oder eine Zeit relativer Ruhe auf der Datenbank.

Volltextabfragen

Konzentrieren wollen wir uns hier auf Abfragen mittels CONTEXT-Indizes, die mit dem Operator CONTAINS realisiert werden. Die allgemeine Syntax ist:

```
CONTAINS(
     [schema.]text_column,
     text_query [VARCHAR2|CLOB]
     [,label NUMBER])
     RETURN NUMBER;
```

Dazu kommt der SCORE-Operator, mit dessen Hilfe sich die berechnete relative Relevanz des Dokuments ausgeben läßt: `SCORE(label NUMBER)`
Der Score wird als Maß von 0 bis 100 ausgegeben. Das Label referenziert den entsprechenden CONTAINS-Block. Eine Beispielabfrage sieht so aus:

```
SELECT SCORE(1), autor, titel
     FROM text_table
     WHERE CONTAINS(text_column, 'oracle', 1) > 0
     ORDER BY SCORE(1) DESC;
```

Diese Abfrage ist sehr einfach, weil sie den Score auf der Basis des Vorkommens eines einzelnen Begriffs errechnet. Oracle Text bietet aber sehr vielfältige Möglichkeiten, den Volltextteil der Abfrage zu gestalten. Das kann nicht im Detail wiedergegeben werden. Ein paar markante Möglichkeiten sollen aber Erwähnung finden:

- Es kann eine Wichtung der Bestandteile des Test-Strings für die Score-Berechnung vorgenommen werden.
- Dokumente, die bestimmte Begriffe nicht enthalten, können höher bewertet werden.
- Der Abstand von Suchbegriffen im Text kann die Berechnung der Relevanz beeinflussen.
- Unscharfe Suche ist möglich. Begriffe mit ähnlicher Schreibweise gehen in die Berechnung ein.
- Die Bestandteile von Composita werden getrennt bewertet, was vor allem für die deutsche Sprache interessant ist.
- Die Suche kann spezifisch auf Wortstämme ausgerichtet werden.
- Es sind Abfragen bezogen auf die innere Struktur der Dokumente (Überschriften, Absätze etc.) möglich.

Es ist wichtig festzustellen, daß bei diesen Abfragen zur Bestimmung des Score *allein* der Index ausgewertet wird. Das heißt, die Dokumente bleiben außen vor. Deswegen ist es auch wichtig, den Index so kompakt wie möglich zu halten (Optimierung) und nur die wirklich benötigten Features im Index anzulegen (Anzahl der Indexobjekte).

Erweiterte Funktionalität

Man kann hier unterscheiden zwischen Features, die die Möglichkeiten zur Abfrage erweitern und solchen, die es erlauben, das Resultat nutzerfreundlich darzustellen.

1. Der Such-String kann in Richtung von Unter- und Oberbegriffen sowie Synonymen expandiert werden. Das verlangt aber einen Thesaurus. Ein einfacher, allgemeiner Thesaurus für die englische Sprache wird von Oracle Text mitgeliefert. Den Thesaurus für eine andere Sprache bzw. für ein spezielles Wissensgebiet muß man selber erstellen und implementieren.
2. Wenn man den Thesaurus zur Knowledge Base kompiliert, dann sind auch thematische Abfragen mit dem Operator ABOUT möglich. Für das Dokument wird dann ein Score errechnet, auch wenn es den Suchbegriff gar nicht enthält, sondern nur verwandte Begriffe, die aber zum gleichen Thema gehören.
3. Oracle Text beherrscht nicht nur mehre Sprachen, sondern kann diese auch automatisch in einer Dokumentensammlung erkennen und bei der Indizierung berücksichtigen. Verantwortlich dafür ist ein mehrsprachiger Lexer.
4. Viele zusätzliche Funktionen stellt das Package CTX_DOC bereit:
   - HIGHLIGHT produziert eine Liste mit den Offsets der Fundstellen im Dokument.
   - MARKUP: Man kann sich das Dokument selber mit markierten Fundstellen ausgeben lassen. Optionale Navigation Tags ermöglichen es, im Dokument per Mausklick von Fundstelle zu Fundstelle zu springen.
   - SNIPPET: Diese Funktion erzeugt eine Konkordanz für das Dokument und stellt die Fundstellen in ihrem Kontext dar, so wie man das auch von anderen Suchmaschinen kennt.
   - GIST: Eine thematische Zusammenfassung des Dokuments auf Satz- oder Absatz-Basis wird ausgegeben. Voraussetzung ist allerdings eine kompilierte Knowledge Base.

Hier als Beispiel die Verwendung von CTX_DOC.SNIPPET:
Syntax:

```
select ctx_doc.snippet('text_index', 'doc_id', 'query_string') from dual;
```

z.B. (Gesucht wird nach *Microsoft* in der Nähe von *Patch*.)

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

Resultat:

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

Oracle Text und XML-Daten

XML-Daten stehen zwischen den relationalen Daten und unstrukturierten Dokumenten. Es handelt sich um Daten, denen gewissermaßen das Datenmodell inhärent ist. Man kann XML-Daten auf verschiedene Weise in der Datenbank speichern und auch indizieren. Das wäre ein Thema für sich und soll hier nicht vertieft werden.
Es gibt zwei Varianten von Volltextabfragen auf XML-Daten:

1. Mit einem XPATH-Ausdruck ohne Notwendigkeit eines Volltextindex:

   ```
   SELECT id, m.xml_column.GetCLOBVal() AS XML_Daten
   FROM xml_table m
   WHERE existsNode(m.xml_column,
   '/book/chapter/text[ora:contains(text(),
   "Oracle")>0]', 'xmlns:ora="http://xmlns.oracle.com/xdb"' )=1
   ```

   In dieser Abfrage wird ein Score daraufhin errechnet, ob das Dokument im Pfad `/book/chapter/text` den Text `"oracle"` enthält.
2. Mit einem Volltextindex vom Typ CONTEXT ergeben sich weiter Möglichkeiten für Abfragen auf XML-Spalten.
   Gefragt wird nach dem Vorkommen eines Begriffs in einem bestimmten XML-Element, in einem Pfad oder nach der Existenz eines Pfades:

   ```
   SELECT id, score(1)
   FROM xml_table
   WHERE CONTAINS(xml_column, 'Oracle WITHIN text', 1)>0;
   SELECT id, score(1)
   FROM xml_table
   WHERE CONTAINS(xml_column, 'Oracle INPATH
   (/book/chapter/text)', 1)>0;
   SELECT id, score(1)
   FROM xml_table
   WHERE CONTAINS(xml_column, 'HASPATH
   (/book/chapter/ueberschrift)', 1)>0;
   ```

Neuerungen in Oracle 12c

Auch wenn Oracle Text nicht im Focus der Aufmerksamkeit steht, was die New Features des letzten Release anbetrifft, so gibt es doch einige wichtige Änderungen. Vor allem betreffen sie Möglichkeiten zur Erhöhung der Performance. Einige sollen hier kurz aufgezählt werden:

- Man kann den Context-Index mit dem Attribut **BIG_IO** der Präferenz BASIC_STORAGE anlegen. Dann wird bei gleichem Token-Text für jeden Token-Typ ein separater LOB angelegt.
- Man kann den Context-Index mit dem Attribut **SEPARATE_OFFSETS** der Präferenz BASIC_STORAGE anlegen. DOCIDs und die Offsets werden separat gespeichert. Das ist schneller bei Abfrage einzelner Wörter bzw. mit Booleschen Operatoren und sollte zusammen mit BIG_IO verwendet werden.
- **Forward Indexing:** Highlighting und Snippets funktionieren üblicherweise indem die Fundstellen in der Index-Tabelle DR$<index-name>$I abgelegt werden. Zur Laufzeit wird mit diesen Informationen ein Dokument mit hervorgehobenen Fundstellen generiert. Das kann bei großen Dokumenten schlecht für die Performance sein. Forward Indexing bedeutet, daß es zusätzlich eine DR$<index-name>$O-Tabelle für die Offsets gibt, die ein Mapping auf DR$<index-name>$I darstellt. Auf deren Basis wird dann Higlighting, Snippet und Markup gemacht.
- **Save Copy:** Zusätzlich zu Forward Index werden die Dokumente in einer $D-Tabelle gespeichert. Ob das Dokument in DR$<index-name>D$ gespeichert werden soll, kann mit einer zusätzlichen Tabellenspalte bestimmt werden, die folgende Einträge enthalten kann: PLAINTEXT für SNIPPET, FILTERED für MARKUP oder HIGHLIGHTING, NONE für Spalten mit VARCHAR2 oder CLOB.
- **Near Real-time Indexing:** DML auf den Textspalten erfordert Pflege des Index. (Synchronisierung und Optimierung). Das Problem bei intensiver DML, daß der Index tendenziell veraltet oder Beeinträchtigung der Performance ist. Eine Alternative ist die neue Option STAGE_ITAB. Neue Einträge werden in die Staging Table DR$<index-name>$G geschrieben. Mit der MERGE Methode zur Optimierung werden die Einträge zusammengefaßt (optlevel => CTX_DDL.OPTLEVEL_MERGE). Die Tabelle DR$<index-name>$G wird im KEEP-Pool gehalten, was schnellere Abfragen ermöglicht.

Ausblick

Oracle Text ist ein ziemlich mächtiges, oft unterschätztes Werkzeug. Es ist fest in die Oracle-Datenbank integriert und erfordert keine zusätzlichen Kosten oder Installationen. Ein zusätzlicher Vorteil ist, daß alle Text-Operationen sehr datenbanknah erfolgen und deswegen vom Datenbankmanagementsystem besser optimiert werden können, als wenn die Text-Logik in einer clientseitigen Applikation angesiedelt wäre.
Anliegen des Vortrags war, Oracle Text wieder etwas mehr Aufmerksamkeit zu verschaffen. Deswegen habe ich auch in Kooperation mit der Ordix AG einen dreitägigen Kurs konzipiert. Die Inhalte und nächsten Termine finden sich hier:
http://training.ordix.de/siteengine/action/load/kategorie/Oracle/nr/1824/index.html

Kontaktadresse:
Dr. Frank Haney
Anna-Siemsen-Str. 5
D-07745 Jena

Telefon: +49(0)3641-210224
E-Mail: info@haney.it
Internet: http://www.haney.it