Volltextsuche mit Oracle

Findet Memo

Ein Archiv für Texte und andere Dokumente in Office-Formaten ist mit Oracle relativ einfach zu realisieren: Die Datenbank stellt dafür einen ganzen Werkzeugkasten mächtiger Funktionen bereit. Badran Farwati

Während Geschäftszahlen vielerorts seit Jahrzehnten mit Hilfe von Business-Software analysiert und verwaltet werden, blieben Texte lange außerhalb der Reichweite solcher Applikationen. Erst seit ein paar Jahren geraten zentrale, rechnergestützte Archive für Dokumente aller Art – Korrespondenz, Rechnungen, Berichte, E-Mails oder Notizen – stärker in den Blickpunkt. Schlagworte dafür sind etwa DMS (Document Management System) oder ECM (Enterprise Content Management). Derartige Anwendungen präsentieren ihre Dienste oft über Webportale und integrieren Workflow-Komponenten.

Wer auf solche Erweiterungen verzichten kann, außerdem bereits eine Oracle-Datenbank verwendet und ein wenig Kapazität für Skripting und SQL-Entwicklung frei hat, kommt aber schnell und kostengünstig auch mit einem Eigenbau zum Ziel. Denn Oracle bietet, was nicht jedem geläufig ist, ein Paket leistungsstarker Funktionen für die Suche nach Texten in allen üblichen Office-Formaten – damit hat man das halbe Textarchiv schon in der Tasche.

Dieses Feature steht seit den frühen 8er Versionen auf der Liste, wurde allerdings von Version zu Version umbenannt: Erst hieß es Context (bis 8.1.5), dann Intermedia Text (bis 8.1.7), und schließlich Oracle Text for Oracle 8i, was sich inzwischen zur aktuellen Schreibweise Oracle Text verkürzt hat. Gemeint ist in jedem Fall ein Mechanismus, der das Indizieren von Texten und die Suche über diesen Index erlaubt. Die Metadaten der Texte – etwa Angaben zu Autor, Quelle, Titel, Entstehungszeit und so weiter – speichert die Datenbank in normalen Tabellenfeldern.

Quellenvielfalt

Je nach Größe finden die Texte selbst entweder in Textfeldern Platz oder werden in so genannten LOBs (Large Objects) abgelegt. Diese wiederum können Bestandteil einer Tabelle sein (Datentyp BLOB, Binary Large Object; Maximalgröße 4 GByte) oder im Filesystem abgebildet werden. In diesem Fall enthält die Datenbank einen Verweis auf den Speicherort (Datentyp BFILE). Eine Datenquelle dieser Klasse nennt sich im Oracle-Text-Jargon Direct Datastore.

Alternativ können die Texte aber auch gewöhnliche Files (File Datastore) oder über das Inter- oder Intranet erreichbar sein (URL Datastore). Sie dürfen sogar auf Wunsch von einer Funktion geliefert werden, die der Anwender selber beisteuert (User Datastore).

Der Datenstrom, den Oracle aus einer dieser Quellen liest, wird dabei zunächst gefiltert. Passende Filter gibt es für mehr als 150 Datenformate, angefangen von Ascii-Text über PDF bis Word oder Excel. Zusätzliche Filter bieten mehrere Drittthersteller an. Den nächsten Arbeitsschritt übernimmt ein so genannter Sectioner, der in HTML- und XML-Dokumenten durch Tags strukturierte Abschnitte erkennt, nach denen zu suchen ebenfalls möglich ist.

Wörter, Themen, Klassen

Unstrukturierte Texte zerlegt der nachfolgende Lexer in kleine Einheiten (Tokens). Dabei berücksichtigt er die Eigenarten verschiedener Sprachen (bis hin zu Japanisch oder Koreanisch). Bestimmte Tokens lassen sich über konfigurierbare Stopp-Listen ausschließen. Am Ende stehen einzelne Wörter, aus denen ein inverser Index aufgebaut wird, wobei jedem Wort auch eine Liste seiner Fundstellen zugeordnet ist (siehe Abbildung 1).

Zusätzlich zur Volltextsuche bietet Oracle die Suche nach Themen an (per De-

fault allerdings nur in den Sprachen Englisch und Französisch). Dafür wird eine hierarchische Liste von Kategorien herangezogen, die es in Grenzen gestattet abzufragen, wovon der Text handelt. Ebenso möglich ist schließlich die direkte Klassifizierung von Texten. So könnten beispielsweise eingehende Nachrichten sortiert werden – eine Reihe spezieller Abfragen würde sie dann automatisch Ressorts wie Sport, Politik, Lokales oder Kultur zuordnen.

Volltextsuche

Das hier vorgestellte Beispiel behandelt nur einen Aspekt des komplexen Instrumentariums, die Volltextsuche. Da sich das Beispiel auf eine Prinziplösung mit exemplarischen Funktionen beschränkt, fehlt eine Reihe von Funktionen, die eine vollständige Applikation haben müsste. Beispielsweise ein Benutzer- und Berechtigungskonzept, eine Sessionverwaltung oder eine ausgefeilte grafische Oberfläche. Das alles und mehr kann aber entsprechend den eigenen Bedürfnisse ergänzt werden.

Das Beispiel stützt sich auf PHP für das Browser-GUI. Diese Skriptsprache verbinden die meisten sicherlich zuerst mit MySQL, sie bietet jedoch auch Schnittstellen zu einer ganzen Palette von Datenbanken. Der Oracle-Support wird beim Übersetzen von PHP mittels »./configure .. --with-oci8 = $ORACLE_HOME« eingebunden [1].
Im vorgestellten Beispiel erfolgt die Ablage der Texte im Filesystem des Datenbank-Servers. Das Webinterface bietet zu diesem Zweck die Möglichkeit, Dateien dorthin zu importieren.

Tabellenbau

Zuerst müssen die zu archivierenden Dokumente auf den Server und in die Datenbank gelangen. Listing 1 stellt ein SQL-Skript vor, das zunächst die nötige Tabellenstruktur erzeugt. Die Tabelle »mein_doc« nimmt sowohl die Metadaten als auch die Verweise auf den Ablageort im Filesystem auf. Der wiederum ist in der Oracle-Prozedur »BFILENAME« als Parameter in Form eines Aliasnamens anzugeben, der sich mittels »CREATE DIRECTORY« erzeugen lässt. Das Abspeichern übernimmt »put_file()«. Gelöschte Datensätze werden vorsichtshalber in die Tabelle »mein_doc_h« kopiert, wo sie nach Ablauf einer Sicherheitsfrist entsorgt werden können. Unter [2] finden sich auch alternative Skripte, die demonstrieren, wie Tabellen mit

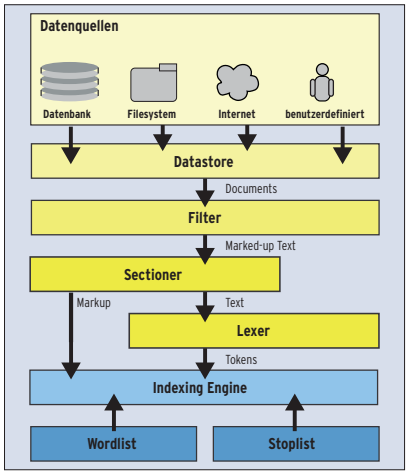

Abbildung 1: So entsteht ein Oracle-Index für Text. Die Daten wandern aus einer der möglichen Quellen durch eine konfigurierbare Verarbeitungskette und werden am Ende wortweise im Index erfasst.

Listing 1: »bfile_loesung.sql«

```
01 DROP TABLE mein_doc;
02 DROP TABLE mein_doc_h;
03 DROP SEQUENCE mein_doc_seq;
04 DROP INDEX mein_doc_idx;
05 CREATE TABLE mein_doc (
06   id NUMBER,
07   file_name VARCHAR2(21),
08   user_file_name VARCHAR2(255),
09   upload_datum VARCHAR2(10),
10   titel VARCHAR2(255),
11   autor VARCHAR2(255),
12   beschreibung VARCHAR2(255),
13   groesse VARCHAR2(20),
14   mime VARCHAR2(50),
15   inhalt BFILE,
16 CONSTRAINT doc_pk PRIMARY KEY (id)
17 );
18
19 create table mein_doc_h (
20  id NUMBER,
21  file_name VARCHAR2(21),
22  user_file_name VARCHAR2(255),
23  upload_datum VARCHAR2(10),
24  titel VARCHAR2(255),
25  autor VARCHAR2(255),
26  beschreibung VARCHAR2(255),
27  groesse VARCHAR2(20),
28  mime VARCHAR2(50)
29 );
30
31 CREATE SEQUENCE mein_doc_seq;
32
33 CREATE OR REPLACE DIRECTORY dokumente AS
   '&1';
34
35 GRANT READ ON DIRECTORY dokumente TO ctxsys;
36
37 CREATE INDEX mein_doc_idx ON mein_doc(inhalt)
   INDEXTYPE IS CTXSYS.CONTEXT
38 PARAMETERS ('SYNC ( ON COMMIT)');
39
40 CREATE OR REPLACE PROCEDURE put_file
41 (
42  p_file_name  IN  mein_doc.file_name%TYPE,
43  p_user_file_name  IN
   mein_doc.user_file_name%TYPE,
44  p_upload_datum IN
   mein_doc.upload_datum%TYPE,
45  p_titel IN mein_doc.titel%TYPE,
46  p_autor IN mein_doc.autor%TYPE,
47  p_beschreibung IN
   mein_doc.beschreibung%TYPE,
48  p_groesse IN mein_doc.groesse%TYPE,
49  p_mime IN mein_doc.mime%TYPE
50 ) AS
51   index_name varchar2(20) := 'mein_doc_idx';
52 BEGIN
53   INSERT INTO mein_doc (id, file_name,
   user_file_name, upload_datum,titel,
54   autor, beschreibung,groesse, mime, inhalt)
55   VALUES (mein_doc_seq.NEXTVAL, p_file_name,
   p_user_file_name, p_upload_datum,
56   p_titel , p_autor, p_beschreibung,
   p_groesse, p_mime,
57   BFILENAME('DOKUMENTE',p_file_name));
58   COMMIT;
59
60   -- Für Oracle < 10 aktivieren:
61   --
   CTX_DDL.SYNC_INDEX(idx_name=>index_name);
62
63 END;
64 /
65
66 CREATE OR REPLACE PROCEDURE history
67 (
68   p_id IN mein_doc.id%TYPE
69 )   as
70 begin
71  insert into mein_doc_h select id, file_name,
   user_file_name, upload_datum,
72  titel, autor, beschreibung, groesse, mime
   from mein_doc where id = p_id;
73 delete from mein_doc where id = p_id;
74 commit;
75 end;
76 /
```

BLOBs oder externen Files anzulegen und zu füllen sind. Einige Skripte, darunter auch das aus Listing 1, sind dort in jeweils einer Version für Oracle 9 und 10 zu finden.

Beim Generieren des Index gibt es einen entscheidenden Unterschied zwischen Oracle 10 und den Vorgängerversionen: Während im jüngsten Oracle der im Skript aufgeführte Parameter »SYNC (ON COMMIT)« dafür sorgt, dass bei jeder Änderung der Tabelle (genauer: nach jedem Commit) auch der Index aktualisiert wird, bedarf es unter Oracle 9 und früher dafür eines expliziten Aufrufs von »CTX_DDL.SYNC_INDEX«.

Listing 2: Index-Statistik

```
01 CREATE TABLE</tbody>
02  index_statistik (ergebnis CLOB);
03 DECLARE
04       ausgabe clob := null;
05  BEGIN
06       CTX_REPORT.INDEX_STATS('mein_doc_idx',ausgabe);
07       INSERT INTO index_statistik
08           VALUES (ausgabe);
09       COMMIT;
10       DBMS_LOB.FREETEMPORARY(ausgabe);END;
11 /
12 SET LONG 32000
13 SELECT *  FROM index_statistik;
```

Im Gleichschritt

Dies kann zum Beispiel zeitgesteuert geschehen. Oracle stellt dafür in »$ORACLE_HOME/ctx/sample/script/« ein eigenes Skript mit dem Namen »drjobdml.sql« zur Verfügung, das die Synchronisation in festen Zeitintervallen startet. Mit »@drjobdml.sql *Indexname* 120« etwa würde der Synchronisierungsjob alle zwei Stunden aufgerufen. Eine andere Möglichkeit besteht darin, einen Trigger zu definieren. Oder man startet wie im Beispiel die Synchronisation direkt beim Speichern. Eine entsprechende Taktik müsste der Programmierer allerdings auch für alle anderen DML-Operationen (Data Manipulation Language) wie Update oder Delete wählen.

Die jüngste Oracle-Version beseitigt mit dieser automatischen Kopplung von DML-Operationen und Index-Synchronisation (transaktionaler Index) einen großen Nachteil des Context-Index seiner Vorgänger.

Text-Tools

Die Funktion zum Synchronisieren des Index stammt aus dem CTX-DDL-Paket, das noch rund 30 weitere nützliche Prozeduren enthält [4]. Darunter zum Beispiel »ADD_STOPWORD« oder »CREATE_STOPLIST«, die einzelne Wörter – etwa die häufig vorkommenden Artikel

Listing 3: »upload.php«

```
01 <?
02 session_start();
03 include_once('./class_putfile.inc');
04 ?>
05 <html>
06 <head>
07  <title>Archiv | upload</title>
08  <meta http-equiv="Content-Type" content=
   "text/html; charset=utf-8">
09  <link href="archiv.css" rel="stylesheet"
   type="text/css">
10 </head>
11 <body bgcolor="#FFFFFF">
12
13 <br><br>
14 <table width="200" bgcolor="#333333" border=
   "0" cellspacing="1"
15 align="left">
16             <tr>
17                  <td bgcolor="#ffffff" align=
   "center"><a href="./get.php"
18 class="tableLikeLink2">Archiv durchsuchen
   </a></td>
19             </tr>
20       </table>
21 <br><br>
22 <?
23
24 $datei_speichern = new putfile();
25 if (isset($_POST[upload]) &&
   $_FILES[file1][name] !=""){
26 $datei_speichern->speichern($_FILES
   [file1],$_POST);
27 }else echo '<p class="tableHeading">Bitte eine
   Datei w&auml;hlen</p>';
28 ?>
29
30  <form name="opus" enctype="multipart/form-
   data" method="post"
31 action="upload.php" accept-charset="utf-8">
32  <table>
33
34   <tr>
35    <td align="left" class="tableLike" valign=
   "top">Lokaler Pfad:
36    </td>
37    <td align="left">
38    <input type="file" name="file1" value="<?
   echo $datei_speichern->file1;
39 ?>" size="56">
40    </td>
41   </tr>
42       <tr>
43   <td align="left" colspan="2"> 
44    </td>
45   </tr>
46   <tr>
47    <td align="left" class="tableLike" valign=
   "top">Titel des Dokuments:
48    </td>
49    <td align="left" valign="top">
50    <textarea name="titel" rows="2" cols="56"
   wrap="virtual"><? echo
51 $datei_speichern->titel; ?></textarea>
52    </td>
53   </tr>
54   <tr>
55    <td align="left" class="tableLike" valign=
   "top">Autor des Dokuments:
56    </td>
57    <td align="left" valign="top">
58    <input type="text" name="autor" value="<?
   echo $datei_speichern->autor;
59 ?>" size="56">
60    </td>
61   </tr>
62   <tr>
63    <td align="left" class="tableLike" valign=
   "top">Kurze Beschreibung des
64 Dokuments:
65    </td>
66    <td align="left" valign="top">
67    <textarea name="beschreibung" rows="3"
   cols="56" wrap="virtual"><? echo
68 $datei_speichern->beschreibung; ?></textarea>
69    </td>
70   </tr>
71     <tr>
72   <td align="left" colspan="2"> 
73    </td>
74   </tr>
75   <tr>
76   <td align="left" class="tableLike"
   valign="top"> 
77    </td>
78    <td align="left">
79    <input type="submit" name="upload"
80
value="Schicken">     
81    <input type="reset" name="reset" value=
   "Eingabe l&ouml;schen">
82    </td>
83   </tr>
84  </table>
85  </form>
86 </body>
87 </html>
88 ?
```

- Anzeige -

und Präpositionen, die allein keinen inhaltlichen Sinn haben – von der Indizierung ausschließen.
Wird der Index häufig synchronisiert, kommt es zu einer Fragmentierung. Deren Grad kann man mit Hilfe der folgenden SQL-Funktion ermitteln (Listing 2). Als Gegenmaßnahme bei zu starker Fragmentierung liefert Oracle die Prozedur »CTX_DDL.OPTIMIZE_INDEX« mit. Mögliche Parameter sind »FAST« oder »FULL«, zum Beispiel »EXEC CTX_DDL.OPTIMIZE_INDEX(idx_name = > 'anno_doc_idx', optlevel = > 'FULL');«.
Schließlich ist es empfehlenswert, von Zeit zu Zeit zu kontrollieren, ob es beim Indizieren zu Fehlern gekommen ist. Dies verrät ein Blick in die Tabelle »CTX_USER_INDEX_ERRORS«:

```
SELECT err_index_name, err_timestamp,
err_text
FROM ctx_user_index_errors
ORDER BY err_timestamp;
```

Ein Index für jeden Zweck

Der vorgestellte Index vom Typ Context beherrscht außerdem die Suche nach Wörtern des gleichen Wortstamms oder nach ähnlich klingenden Wörtern (so genannte Fuzzy-Suche). Er kann auch größere Datentypen bis zu 4 GByte indizieren und eignet sich daher für »CLOB«, »BLOB«, »BFILE«, »VARCHAR 2«, »CHAR« und für »XMLTYPE«.
Daneben kennt Oracle eine Reihe weiterer, spezialisierter Indexarten. Ctxcat eignet sich für die Indizierung von kürzeren Texten bei Abfragen, in denen Texte und Metadaten gemischt werden. In diesen Fällen verspricht er eine bessere Performance als ein Context-Index. Dafür verbraucht er aber 10- bis 15-mal mehr Platz als die Tabelle selbst und kennt weniger Suchoptionen als Context. Context benötigt dagegen maximal so viel Platz wie die Tabelle.

Fragewörter

Der dritte Indextyp, Ctxrule, kann nur auf »CLOB«- und »VARCHAR2«-Datentypen aufgebaut werden. Dieser Indextyp ist für die Klassifikation von Dokumenten bestimmt. Außer der Tabelle mit den Dokumenten legt Oracle eine zweite Tabelle an, in der die verschiedenen Dokumentklassen und vordefinierten Queries gespeichert sind, deren Ergebnisse der Zuordnung eines Dokuments zu einer der Klassen dienen.
Für die Volltextsuche kommen die üblichen Operatoren wie »LIKE« oder » = « nicht in Frage. Stattdessen verlangt jeder Indextyp einen eigenen Suchoperator: Ctxcat erwartet in der »WHERE«-Klausel den Operator »CATSEARCH«, Context

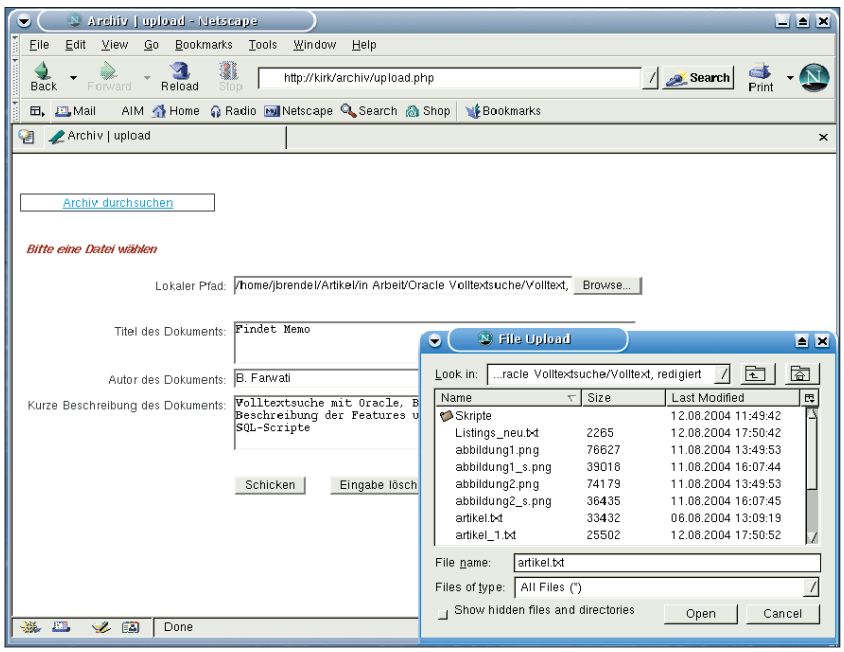

Abbildung 2: Über dieses Upload-Interface kann der Benutzer eigene Dokumente in das Archiv aufnehmen, in dem er sie später mit der Oracle-Volltextsuche durchforsten kann.

wird mit Hilfe von »CONTAINS« befragt und Ctxrule braucht »MATCHES«. Daneben gibt es »ABOUT« für die Themensuche und »WITHIN« für die Suche nach Abschnitten von HTML- oder XML-Dokumenten.

Datentypen wie »CLOB« oder »BLOB« lassen sich nicht in einer Ergebnistabelle darstellen, sie dürfen daher auch nicht als gewünschte Ergebnis-Spalten hinter dem »SELECT« vorkommen. Oracle wertet die Abfrage zuerst nach der Rangfolge der Operatoren aus. Gleichwertige Operatoren werden von links nach rechts gelesen.

Die wichtigsten Operatoren sind das logische »OR« (|), »AND« (&), »ACCUMulate« (,), »$« als Symbol für die Stammwortsuche, »NOT« mit dem Symbol »~« und »!« für die Suche nach ähnlichen Wörtern (Soundex).

Rangordnung

Oracle unterteilt die Operatoren in zwei Gruppen: Jene, die nur einen Operanden verlangen, und jene, die mindestens zwei brauchen. Innerhalb der Operatorengruppen gibt es jeweils eindeutige Prioritätsregeln: Für unäre Operatoren gilt Stem vor Soundex; für die andere Gruppe gilt NOT vor AND vor OR vor ACCU [4].

Einige Beispiele demonstrieren die Verwendung dieser Operatoren. Eine bestimmte Zeichenkette findet man so:

```
SELECT id, file_name
FROM tabellen_name
WHERE CONTAINS(spalten_name,➥
'Universität Wien' ) > 0
ORDER BY id;
```

Wer nach mehreren Wörtern zugleich fahnden will, verknüpft sie mit dem Operator »&«:

```
SELECT id, file_name
FROM tabellen_name
WHERE CONTAINS(spalten_name,➥
'Universität & Wien' ) > 0
ORDER BY id;
```

Beim Testen dieser Abfrage mit SQLPlus ist eine wichtige Kleinigkeit zu beachten: Für SQLPlus ist »&« der Anfang eines Variablennamens, weshalb hier der vorangestellte Befehl »scan off« notwendig ist. Reservierte Wörter gehören in geschwungene Klammern.

Oracle kann die Fundstellen nach ihrer Relevanz bewerten, indem es einen Punktwert (Score) pro Suchresultat berechnet. Der Programmierer bindet die Trefferbewertung an ein so genanntes Label – im folgenden Beispiel »1«. Eine entsprechende Abfrage sieht so aus:

```
SELECT SCORE(1), title
FROM tabellen_name
WHERE CONTAINS(spalten_name,➥
'oracle', 1) > 0
ORDER BY SCORE(1) DESC;
```

In Listing »get.php« (zu finden unter [2]) ist ein bescheidenes, allgemeines Webfrontend skizziert, das die Funktionen Stem, Soundex und AND verwendet. Diese Suchmöglichkeiten kann jeder nach seinen Erfordernissen weiter ausbauen. Ein Beispiel dafür wäre:

```
SELECT id, file_name
FROM mein_doc
WHERE CONTAINS(inhalt,'(urlaub & reise) ➥
| ( hotel & appartement ) ~schnee') > 0 ;
```

Select findet Dokumente, die entweder die zwei Wörter Urlaub und Reisen beziehungsweise Hotel und Appartement (oder alle vier) enthalten, nicht aber das Wort Schnee.

Wer das hier beschriebene Beispiel auf seinem Rechner installieren möchte, ko-

Listing 4: »class_db.inc«

```
<?
include_once('./config.inc');
// Database Class for oracle
class db {
      var $db;
      var $user;
      var $pass;
      var $error_msg;

      function db(){
            global $DB, $USER, $PASS;
            $this->db = $DB;
            $this->user = $USER;
            $this->pass = $PASS;
            $this->open();
      }
      function open(){
            $conn = OCILogon($this->user,$this->pass,$this->db);
            $this->conn = $conn;
            if ($conn == false) {
                 echo OCIError($conn);
                 return false;
            }else return true;
      }
      function query($query) {
            $this->query = $query;
            $handler = OCIParse($this->conn,$query);
            OCIExecute($handler,OCI_DEFAULT);
            $i =0;
            while(OCIFetchInto($handler, $result, OCI_RETURN_NULLS +
OCI_ASSOC)) {
                 foreach($result as $key=>$value){
                      $k = strtolower($key);
                      $ergebnis[$i][$k] = $value;
                 }
                 $i++;
            }
            return ($ergebnis);
            OCIFreeStatement($handler);
            OCILogoff($this->conn);
      }
      function insert($insert_stmt){
            $handler = OCIParse($this->conn,$insert_stmt);
            //OCIExecute($handler,OCI_DEFAULT);
            if(!is_resource($handler) || !OCIExecute($handler,
OCI_DEFAULT))
{
                 $this->error_msg = "SQL-Statement-Handler ist nicht
korrekt";
                 echo $this->error_msg;
                 return false;
            }else return true;
            OCIFreeStatement($handler);
            OCILogoff($this->conn);
      }
}
?>
```

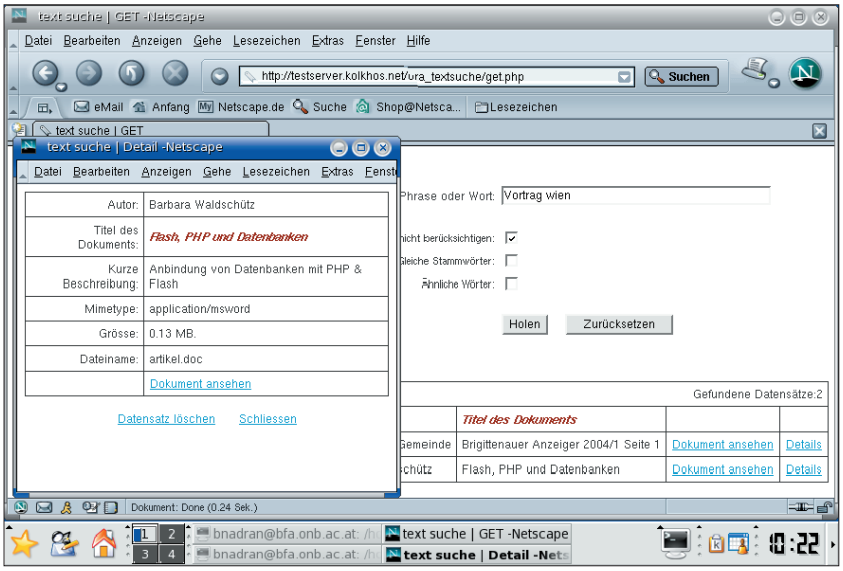

Abbildung 3: Die Suchmaske der Beispiel-Applikation: Rechts die Eingabefelder für die Suchbegriffe, links eine Trefferanzeige, die auch Metadaten des gefundenen Dokuments zeigt.

piert sich zuerst alle ».php«- und ».inc«-Dateien aus [2] in ein Unterverzeichnis des »htdocs«-Directory des Webservers und loggt sich dann als Oracle-User »system« in die Datenbank ein.

Anpassen, loslegen

Mit Hilfe des SQL-Skripts »create_user.sql« [2] legt man einen neuen User für die Applikation an. Dieses Skript erfordert vier Parameter: »@*pfad*/create_user.sql *Username Passwort* default_tablespace temp_tablespace«. Anschließend führt der neu angelegte User das Skript »bfile_loesung.sql« aus, das als Parameter den Pfad zu jenem Verzeichnis verlangt, in dem die Dokumente abgelegt werden sollen: »@bfile_loesung.sql *Pfad zum Upload-Verzeichnis*«.

Nun sind noch das Upload-Verzeichnis selbst und darunter ein Unterverzeichnis zu erzeugen, in dem später die aus der Datenbank gelöschten Dokumente landen sollen (im Beispiel das Verzeichnis »deleted«). Das Upload-Verzeichnis muss für den User beschreibbar sein, der den Webserver gestartet hat:

```
mkdir /upload
mkdir /upload/deleted
chown -R nobody:nobody
```

Darauf folgt die Anpassung der »conf.inc«-Datei. Hier gehören vor allem die jeweilige SID der Datenbank sowie Username und Passwort des Beispielbenutzers hinein. Zum Schluss legt man in der Datei »httpd.conf« des Apache-Servers noch einen Alias für das Upload-Verzeichnis an:

```
Alias /upload/ "pfad/upload/verzeichnis"
    <Directory "pfad/upload/verzeichnis">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

Die Änderungen werden nach einem Neustart des Webservers wirksam.

Buchstabensalat

Bei der Oracle-Installation – sie ist hier nicht näher beschrieben – sollte der Anwender darauf achten, dass er die Datenbank am besten mit dem Character-Set UTF-8 (Option »AL32UTF8«) anlegt. Dann kann die Applikationen Dokumente in unterschiedlichen Sprachen archivieren und durchsuchen. Nachträglich lässt sich das Character-Set der Datenbank nur dann ändern, wenn das alte Character-Set eine Untermenge des neuen ist, wobei sich alle gemeinsamen Zeichen an identischen Positionen befinden müssen.

Ein Wechsel etwa von ISO88591 zu UTF-8 ist deshalb nach der Installation der Datenbank nicht mehr möglich. Doch nicht nur bei der Datenbank ist auf den Zeichensatz zu achten: Auch Webserver und PHP müssen sowohl beim Doku-

menten-Upload als auch während der Suche UTF-8 verwenden.

Zwar werden im Beispiel alle Dokumente beim Upload mit »charset = utf-8« im HTML-Metatag gekennzeichnet, doch würde das überhaupt nichts bewirken, wenn in der »php.ini« gleichzeitig »default_charset = "iso-8859-1"« vorgegeben wäre. Denn seit der Version 4.0b4 ersetzt PHP automatisch die Zeichensatz-Einstellung im HTML-Header durch seine eigene Vorgabe.

Ein Ausweg besteht darin, in der PHP-Konfigurationsdatei »php.ini« die Definition des Standard-Zeichensatz frei zu lassen (genau ein Leer- zwischen zwei Anführungszeichen setzen). Das bewirkt, dass PHP auf eigene Eintragungen verzichtet und so die in den HTML-Seiten bereits vorhandenen Kennzeichnungen erhalten bleiben.

Im Unterschied zur Beispielapplikation wären eigentlich Datenbankverbindungen wünschenswert, die nicht für jede Abfrage neu geöffnet werden müssen, sondern so lange offen blieben, bis der Benutzer sich ausloggt oder bei Inaktivität ein sinnvoller Timeout-Wert überschritten wird.

Sitzungsstress

Diese Technik ist allerdings nicht besonders einfach zu realisieren. Zum Beispiel müsste die maximale Anzahl der Datenbank-Verbindungen abhängig vom verfügbaren Hauptspeicher des Servers berechnet und eine Reihe von Oracle-Parametern daran angepasst werden. Der Webserver wäre ebenfalls gezwungen, sich an dieses vorgegebene Limit zu halten. Möglich ist auch ein Connection-Pool, aus dem sich die Client-Prozesse bedienen können.

Oracle empfiehlt, den Datenbankserver in einem derartigen Szenario generell als so genannten Multi-Threaded-Server (MTS) zu konfigurieren. Weitergehende Hinweise dazu finden sich beispielsweise auf **[6]**. *(jcb)*

Infos

[1] PHP-Manual: [http://www.php.net/manual/de/ref.oracle.php]

[2] Listings zum Artikel: [ftp://ftp.linux-magazin.de/pub/listings/magazin/2004/10/Volltextsuche]

[3] Oracle Text Application Developer's Guide: [http://download-west.oracle.com/docs/cd/B13789_01/text.101/b10729/toc.htm]

[4] Funktionen des CTX_DDL-Package: [http://www.cise.ufl.edu/help/database/oracle-docs/text.920/a96518/cddlpkg.htm#11769]

[5] Oracle Text Reference 10g: [https://cwisdb.cc.kuleuven.ac.be/ora10doc/text.101/b10730/title.htm]

[6] Oracle Multi-Threaded-Server: [http://www.oracleadvice.com/Tips/MTS.htm]

Der Autor

Mag. Badran Farwati arbeitet als Datenbankadministrator und -Programmierer in der Österreichischen Nationalbibliothek und beteiligt sich in seiner Freizeit an mehreren Open-Source-Projekten.

User Friendly – der monatliche Comic im Linux-Magazin