Using Oracle Application Express to Build Oracle Text Web Applications

An Oracle White Paper
January 2006

ORACLE

Using Oracle Application Express to Build Oracle Text Web Applications

Using Oracle Application Express to Build Oracle Text Web Applications

EXECUTIVE OVERVIEW

Oracle Application Express enables programmers and non-programmers to rapidly build Web based applications, using the full power of Oracle database 10g. In this paper we describe how Oracle Text, a full text searching capability included in Oracle database 10g, can be incorporated in your web application.

INTRODUCTION

Because the Oracle Application Express engine exists inside the Oracle 10g database, when you build an application in Oracle Application Express, you can utilize all the Oracle 10g database features. By taking advantage of one such feature, Oracle Text, you can use Oracle Application Express to build robust, text-based, Web applications.

Using Oracle Text you can build applications to query text documents. The text documents are stored in a database table, called a text table. You can populate this table with either the actual text documents, path names of documents on your file system, or URLs indicating the location of the documents on the World Wide Web.

BUILDING A TEXT QUERY APPLICATION

By creating a text query application, users can query documents based on one or more search items. The normal flow of a text query application is as follows:

1. User enters search criteria
2. User is presented with sorted list of results based on relevance
3. User selects a document from the list
4. User is presented the document

The following steps are required to create such an application:

a. Create text table

b. Load text table

c. Index text table using Oracle Text

d. Build a front-end application

This document provides instructions on how to build a text query application using Oracle Application Express. The resulting application enables a user to query an HTML version of *Oracle Application Express User's Guide*. These HTML documents are available on-line in any Oracle Application Express instance.

This document assumes the reader is familiar with the following concepts:

1. Database schema objects such as tables and indexes
2. Using SQL Workshop, SQL Command Processor tool
3. Using Application Builder, Creating an Application
4. Using Application Builder, Creating Regions and Items

Create the Text Table

The text table in this example contains *Oracle Application Express User's Guide* documents as URLs. The instance of Oracle Application Express you are using to build your application will contain all of the documentation on-line, as HTML documents. In the text table, you can store the URLs to the various sections of the documentation.

The instructions in this document are specific to Oracle Application Express version 1.6.

To create a text table:

a. Login to Oracle Application Express.

b. Click the **SQL Workshop** icon.

c. Under SQL Workshop, click the **SQL Command Processor** icon.

d. Paste in the following text:

```
create table htmldb_documentation(
   id          number,
   doc_title   varchar2(4000),
   doc_url     varchar2(4000))
/
```

e. Click **Run SQL**.

Load the Text Table

To load the text table, you will create a procedure that accepts the base URL for the Application Express instance. This procedure uses a pre-populated array containing the document names and titles for each section. The full URL is constructed from the base URL parameter that is passed in, the application global that is the virtual directory that points to the images directory on the file system, and the document name.

To load the text table, paste the following code into the SQL Workshop Command Processor and click **Run SQL**.

Note that if you are pasting this code from the PDF version of the file, you may also be pasting the page footers, "Using Application Express to Build…" You will need to remove this text.

```
create or replace procedure load_docs (p_base_url in varchar2) as
    l_docs          wwv_flow_global.vc_arr2;
    l_doc_detail    wwv_flow_global.vc_arr2;
begin
    l_docs(1)   := 'mvl_intro001.htm:About Oracle Application
Express';
    l_docs(2)   := 'mvl_intro002.htm:About Application Builder';
    l_docs(3)   := 'mvl_intro003.htm:About SQL Workshop';
    l_docs(4)   := 'mvl_intro004.htm:About Data Workshop';
    l_docs(5)   := 'mvl_start001.htm:Understanding Oracle
Application Express User Roles';
    l_docs(6)   := 'mvl_start002.htm:Logging into Oracle Application
Express';
    l_docs(7)   := 'mvl_start003.htm:About Oracle Application
Express User Interface';
```

```
    l_docs(8)   := 'mvl_start004.htm:Creating an Application Using
the Create Application Wizard';
    l_docs(9)   := 'mvl_datashop001.htm:About Data Workshop';
    l_docs(10)  := 'mvl_datashop002.htm:Importing Data';
    l_docs(11)  := 'mvl_datashop003.htm:Exporting Data';
    l_docs(12)  := 'mvl_sqlshop001.htm:About SQL Workshop';
    l_docs(13)  := 'mvl_sqlshop002.htm:Viewing Database Objects';
    l_docs(14)  := 'mvl_sqlshop003.htm:Managing Database Objects';
    l_docs(15)  := 'mvl_sqlshop004.htm:Managing User Interface
Defaults';
    l_docs(16)  := 'mvl_sqlshop005.htm:Browsing the Data
Dictionary';
    l_docs(17)  := 'mvl_fund001.htm:About Page Rendering and Page
Processing';
    l_docs(18)  := 'mvl_fund002.htm:How Page Processing and Page
Rendering Work';
    l_docs(19)  := 'mvl_fund003.htm:Understanding Session State
Management';
    l_docs(20)  := 'mvl_fund004.htm:Managing Session State Values';
    l_docs(21)  := 'mvl_fund005.htm:Understanding URL Syntax';
    l_docs(22)  := 'mvl_fund006.htm:Using Substitution Strings';
    l_docs(23)  := 'mvl_buildr001.htm:Understanding the Definition
of a Page';
    l_docs(24)  := 'mvl_buildr002.htm:Creating an Application';
    l_docs(25)  := 'mvl_buildr003.htm:Creating a New Page Using a
Wizard';
    l_docs(26)  := 'mvl_buildr004.htm:Working with Templates';
    l_docs(27)  := 'mvl_buildr005.htm:Viewing Application
Attributes';
    l_docs(28)  := 'mvl_buildr006.htm:Editing Application
Attributes';
    l_docs(29)  := 'mvl_buildr007.htm:Viewing Page Attributes';
    l_docs(30)  := 'mvl_buildr008.htm:Editing a Page Definition';
    l_docs(31)  := 'mvl_buildr009.htm:Running a Page';
    l_docs(32)  := 'mvl_design001.htm:Displaying Components on Every
Page';
    l_docs(33)  := 'mvl_design002.htm:Adding Navigation';
    l_docs(34)  := 'mvl_design003.htm:Creating Regions';
    l_docs(35)  := 'mvl_design004.htm:Creating Buttons';
    l_docs(36)  := 'mvl_design005.htm:Creating Lists of Values';
    l_docs(37)  := 'mvl_design006.htm:Creating Forms';
    l_docs(38)  := 'mvl_design007.htm:Creating Reports';
    l_docs(39)  := 'mvl_design008.htm:Creating Charts';
    l_docs(40)  := 'mvl_design009.htm:Specifying Layout and User
Interface';
    l_docs(41)  := 'mvl_design010.htm:Creating a Help Page';
    l_docs(42)  := 'mvl_design011.htm:Sending E-mail from an
Application';
    l_docs(43)  := 'mvl_debug001.htm:About Tuning Performance';
    l_docs(44)  := 'mvl_debug002.htm:Remembering to Review Session
State';
    l_docs(45)  := 'mvl_debug003.htm:Accessing Debug Mode';
    l_docs(46)  := 'mvl_debug004.htm:Enabling SQL Tracing and Using
TKPROF';
    l_docs(47)  := 'mvl_debug005.htm:Monitoring Application and Page
Resource Use';
    l_docs(48)  := 'mvl_debug006.htm:Viewing Page Reports';
    l_docs(49)  := 'mvl_debug007.htm:Debugging Problematic SQL
Queries';
    l_docs(50)  := 'mvl_debug008.htm:Removing Components to Isolate
a Problem';
    l_docs(51)  := 'mvl_manage001.htm:Accessing Application Builder
Utilities';
    l_docs(52)  := 'mvl_manage002.htm:Exporting and Importing
Applications';
    l_docs(53)  := 'mvl_manage003.htm:Uploading CSS, Images, and
Static Files';
    l_docs(54)  := 'mvl_manage004.htm:Understanding Security';
    l_docs(55)  := 'mvl_manage005.htm:Establishing User Identity
Through Authentication';
    l_docs(56)  := 'mvl_manage006.htm:Providing Security Through
Authorization';
```

```
    l_docs(57)  := 'mvl_wrkspc001.htm:Understanding Administrator
Roles';
    l_docs(58)  := 'mvl_wrkspc002.htm:Managing Users';
    l_docs(59)  := 'mvl_wrkspc003.htm:Monitoring Users';
    l_docs(60)  := 'mvl_wrkspc004.htm:Administering Session State
and User Preferences';
    l_docs(61)  := 'mvl_wrkspc005.htm:Viewing Workspace Reports';
    l_docs(62)  := 'mvl_wrkspc006.htm:Monitoring Developer
Activity';
    l_docs(63)  := 'mvl_wrkspc007.htm:Managing Log Files';
    l_docs(64)  := 'mvl_wrkspc008.htm:Managing Development
Services';
    l_docs(65)  := 'mvl_a_con001.htm:Conditions Available in Oracle
Application Express';

    for i in 1.. l_docs.count loop
        l_doc_detail := htmldb_util.string_to_table(l_docs(i));
        insert into htmldb_documentation
        values (i, l_doc_detail(2),
p_base_url||htmldb_application.g_image_prefix||
                'doc/'||l_doc_detail(1));
    end loop;
end;
/
```

Once you have created the procedure `load_docs`, you execute it to load the `htmldb_documentation` table. Note that you must replace the value passed in for the `p_base_url` parameter, with the base URL of your specific Application Express instance. In other words, replace "http://htmldb.oracle.com" below with the base URL of your Application Express instance. The base URL is everything before "/pls/" in the URL of your Application Express instance.

To execute the procedure, paste the following code into the SQL Workshop Command Processor and click **Run SQL**.

```
begin
    load_docs(p_base_url=>'http://htmldb.oracle.com');
end;
/
```

Verify that the data loaded properly by querying all columns in the `htmldb_documentation` table. There should be 65 rows in the table.

Create a Text Index Using Oracle Text

To utilize Oracle Text, you must index the text column of your text table. Indexing the text column breaks the text into tokens. Tokens are words usually separated by spaces. The text index stores each token and the documents that contain it.

In the simplest case, you can create a text index on just about any table that has a text column. To do so, you use the INDEXTYPE is CTXSYS.CONTEXT clause of the CREATE INDEX statement. There are many ways to customize the text index, and one way is to specify a DATASTORE preference.

The DATASTORE preference specifies how text is stored in the table for which you want to create the index. In this example, the actual document is not stored in the text table, but rather the URL that specifies the location of the document on the World Wide Web. You can specify this by using the PARAMETERS clause and setting the DATASTORE preference to CTXSYS.URL_DATASTORE.

To create a text index using Oracle Text:

1. Navigate back to the **SQL Command Processor**.
2. Paste in the following text:

```
create index htmldb_doc_ctxidx on htmldb_documentation(doc_url)
   indextype is ctxsys.context
   parameters ('datastore CTXSYS.URL_DATASTORE')
/
```

3. Click **Run SQL**.

Build an Application Using Oracle Application Express

Once you have loaded and indexed the text data, you are ready to build a front-end text query application. This section describes how to create a simple one page, query and results application. Using this application, users will be able to enter a text search that returns a list of results on the same page. The results will be scored and sorted by relevance. Additionally, users can display the actual document by clicking document title.

Create the Application

To create a new application:

For any instructions not specifically given, simply keep the default values.

1. Navigate to the Workspace Home page by selecting the **Workspace** breadcrumb.
2. Click **Create Application >**.
3. In the Create Application Wizard:
   a. For Creation Method, select **From Scratch**.
   b. For Name, enter **Text Demo**.
   c. For Pages, select **1**.
   d. For Tabs, select **One Level of Tab**.
   e. For Page Name, enter **Search**.
   f. For Theme, select **Theme 9**.
   g. Click **Create Application**.

Create a Search Region

To complete this task you must create a search region, add a text item to the region, and then add a button.

To create a search region:

1. Navigate to the Page Definition of the Search page you created in your application.
2. Under Regions, click the create icon ().

3. For Region Type, select **HTML**.
4. For Title, enter **Search Terms(s)**.

To add a text item to the search region:

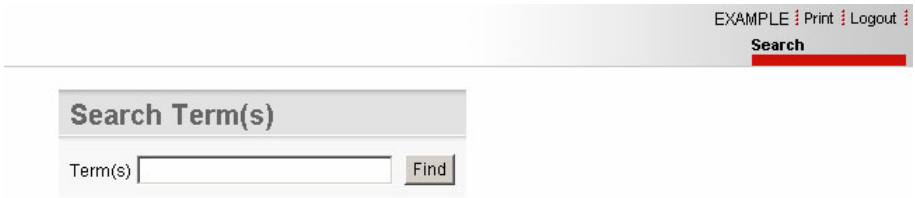
1. Under Items, click the create icon ( ).
2. For Item Name, enter **P1_SEARCH**.
3. For Label, enter **Term(s)**.

To create a button to submit the search:

1. Under Buttons, click the create icon ( ).
2. For Region, select **Search Terms(s)**.
3. For Task, select **Create a button displayed among this region's items**.
4. For Button Name, enter **P1_FIND**.
5. For Label, enter **Find**.
6. For Button Style, select **HTML Button**.

Run your page using the run page icon ( ) and your development username and password. Your page should look similar to the following figure.

Figure 1.0 Initial Search Term Page



Create the Query Region

To utilize the text index, you must include a CONTAINS clause in your SQL query. In the CONTAINS clause, you specify the text column to search for and the search terms. Oracle Text returns all documents that satisfy your criteria, along with a score for the relevance of the document to the criteria.

To create a query region:

1. Navigate back to the Page Definition for the Search page. You can do this by using the edit links at the bottom of the running page, **Edit Page 1**.
2. Under Regions, click the create icon ( ).
3. For Region Type, select **Report.**
4. For Report Implementation, select **SQL Report**.
5. For Title, enter **Results**.

6. Enter the following SQL query:

```
select score(1) relevance, doc_title, doc_url
  from htmldb_documentation
 where CONTAINS (doc_url, :P1_SEARCH, 1) > 0
 order by 1 desc
```

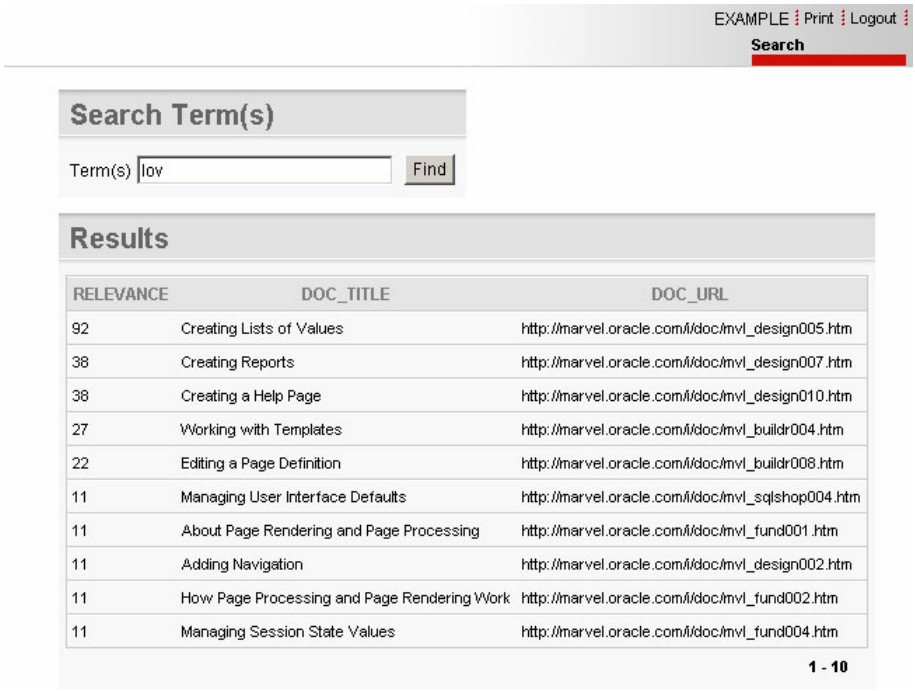
7. For Display Conditions, for Condition Type, select **Value of Item in Expression 1 is NOT NULL**.

8. For Expression 1, enter **P1_SEARCH**.

This condition specifies to only display the Results region if a term is entered as a search criterion.

Run the page and enter a search criterion for **Term(s)**. Your results should look similar to the following figure.

Figure 2.0 Search Term Page with Query Added



Modify Query Region Report Attributes

The next step is to customize the formatting of the column headings and create a hyperlink that will display the actual document.

To make the modifications to the column headings:

1. Navigate back to the Page Definition for the Search page.

2. Under Regions, select the **Q** to the left of the Results region.

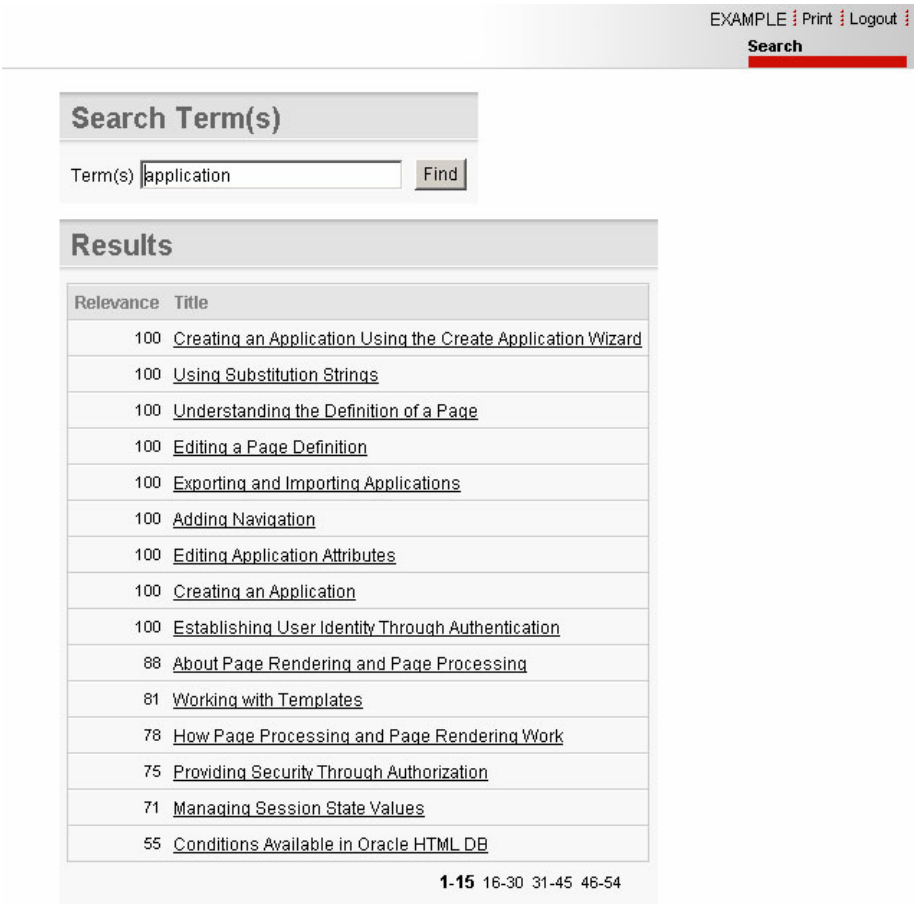
3. Under Report Column Attributes:
    a. For Headings Type, select **Custom**.
    b. For RELEVANCE, for Column Alignment, select **right**.
    c. For DOC_TITLE, for the Heading, enter **Title**.
    d. For DOC_URL, deselect the **Show** check box.
4. Under Layout and Pagination:
    a. For Pagination Scheme, select **Row Ranges 1-15 16-30 (with set pagination)**.
5. Under Report Column Attributes, click the edit icon () to the left of **DOC_TITLE**.
6. Under Column Link, click the popup icon () to the right of **Link Text**.
7. Select **#DOC_TITLE#**.
8. For Target, select **URL**.
9. For URL, enter:

```
javascript:popupURL(&quot;#DOC_URL#&quot;)
```

10. Click **Apply Changes**.
11. Return to the Page Definition by clicking **Cancel**.

Run the page and enter a search criterion for **Term(s)**. Your results should look similar to the following figure.

Figure 3.0 Search Term Page with Formatting Changes and Hyperlink



When you click a document title, a new window displays containing the document. You can scroll through the pages by clicking on the number sets in the bottom right corner.

Understanding the ABOUT Operator

The query defined in the Results query region performs a keyword search based on what a user types in the Term(s) field. In other words, the results returned by the query actually contain the entered term.

When Oracle Text creates a text index, themes are derived from the documents. Themes are concepts that are present in the indexed documents. For example, a search on the theme "sporting events" might return documents about football games and golf matches. The document would not need to contain the exact phrase "sporting events."

You can search for themes by using the ABOUT operator in your CONTAINS clause. The following example uses the previous "sporting events" example:

```
CONTAINS(document, 'about(sporting events)', 1) > 0
```

Adding a Concept Search

You can further utilize themes by creating a concept search that includes the ABOUT operator in your query. That way, you can provide the user with the choice of entering a keyword search or a concept search. The directions below will guide you though the modifications that will allow the user to perform a Keyword or Concept search.

First, you will add an item to the region **Search Terms(s)**:

1. Navigate back to the Page Definition for the Search page.
2. Under Items, click the create icon ().
3. For Item Type, select **Radio**.
4. For Item Name, enter **P1_TYPE**.
5. For Region, select **Search Terms(s)**.
6. For Display Null Option, select **No**.
7. For List of Values Query, enter:
   ```
   STATIC:Keyword;KEYWORD,Concept;CONCEPT
   ```
8. For Identify Item Source, for Default, enter **KEYWORD**.

The new item needs to be edited to make its values display side-by-side, rather than below one another.

1. Under Items, select **P1_TYPE**.
2. Under List of Values, for Columns, enter **2**.
3. Click **Apply Changes**.

Now that the item exists, the query will be modified to reference the new item.

1. Under Regions, select **Results**.
2. For Type, select **SQL Query (PL/SQL Function Body Returning SQL Query).**

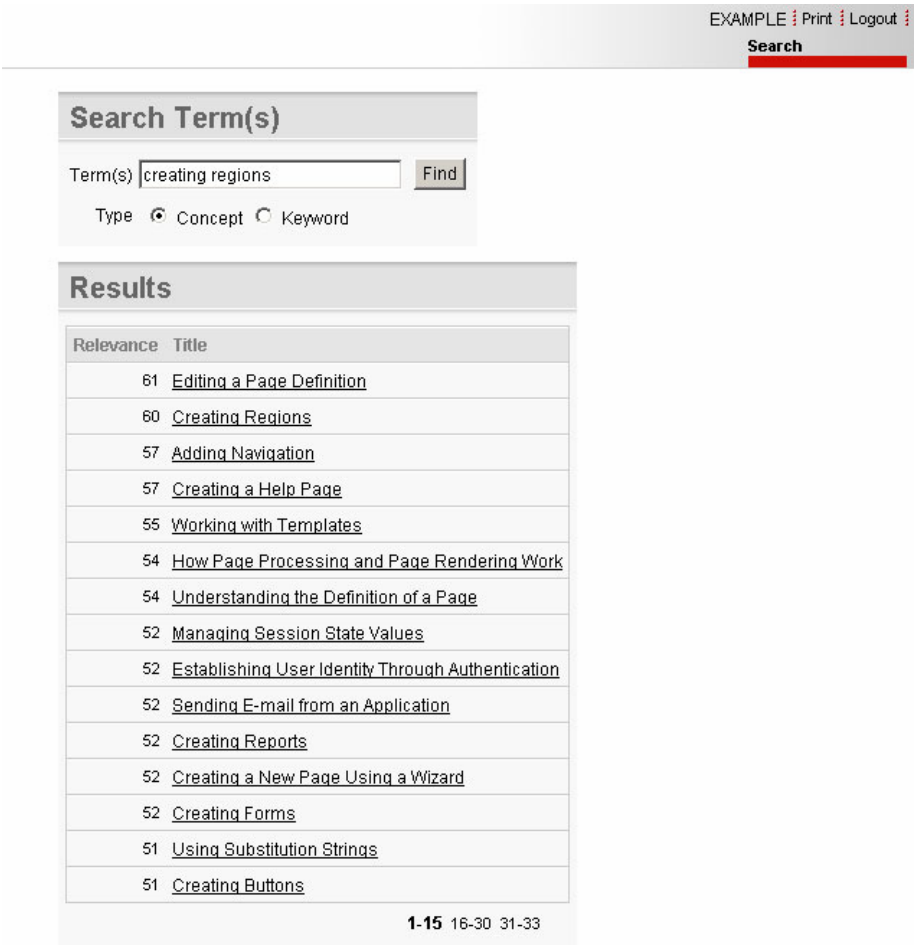
3. For Region Source, enter:

```
declare
  q varchar2(4000);
begin
  q := 'select score(1) relevance, doc_title, doc_url
          from htmldb_documentation
         where CONTAINS ';
  if :P1_TYPE = 'KEYWORD' then
    q:= q|| '(doc_url, :P1_SEARCH, 1) > 0 ';
  else
    q:= q|| '(doc_url, ''about('||:P1_SEARCH||')'', 1) > 0 ';
  end if;
  q := q|| 'order by 1 desc';
  return q;
end;
```

4. Click **Apply Changes**.

Run the page and enter "creating regions" for **Term(s)**. Note that if you change the type to **Concept** and re-run the query there are many more results. The latter approach searches on the theme "creating regions," instead of searching for the exact words "creating regions." Your results should look similar to the following figure.

Figure 4.0 Search Term Page with Concept Search



OTHER ORACLE TEXT INFORMATION

Synchronizing a Text Index

In this example, there is a pre-determined list of text documents in the text table. For some text applications, it may become necessary to add documents to a text table. For example, if a new chapter was added to the documentation, the document references corresponding to the chapter could be inserted into the htmldb_documentation table.

Changing the contents of a text table requires synchronization of the text index. Synchronizing the index processes all updates, inserts, and deletes to the text table. The SYNC_INDEX procedure of the CTX_DDL package is used to synchronize a text index. The procedure accepts parameters for the index name, and the amount of runtime memory to use for synchronization. Using a large amount of memory increases performance of the index operations because less I/O is incurred. A large amount of memory also increases query performance because there would be less fragmentation in the index.

If you added documents to the base table, you could synchronize the `htmldb_doc_ctxidx` by pasting the following in the SQL Command Processor of the SQL Workshop:

```
begin
   ctx_ddl.sync_index('htmldb_doc_ctxidx','2M');
end;
/
```

Utilizing Query Templates

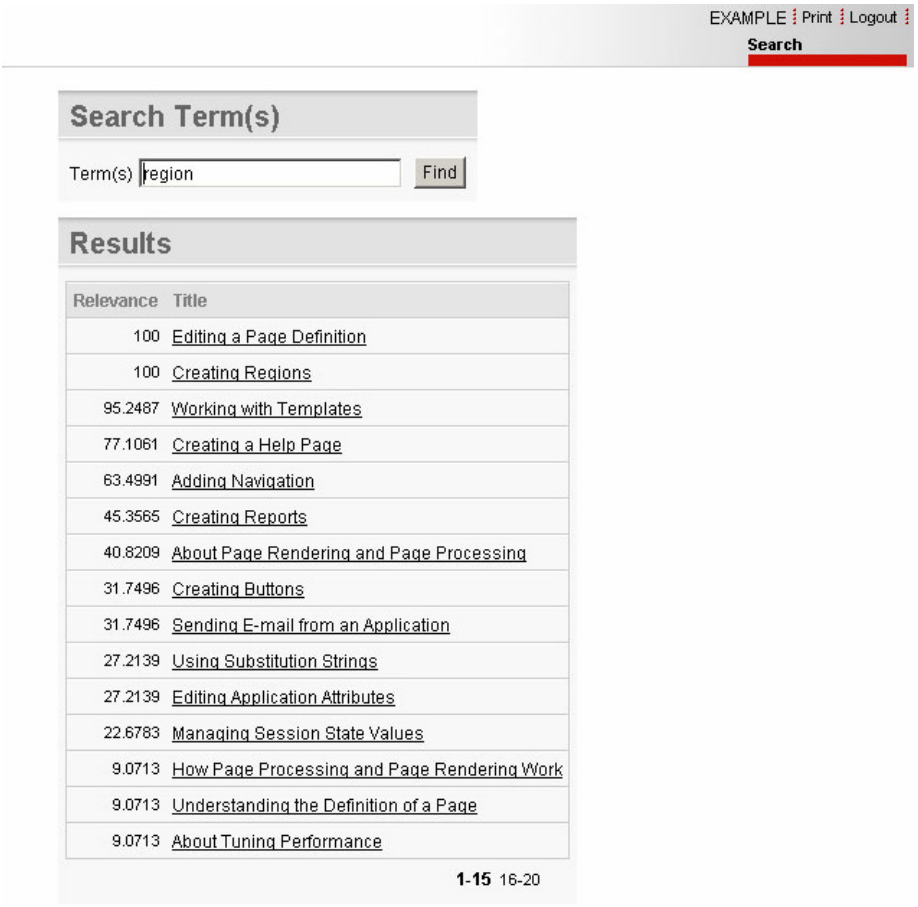
Query templates are an alternative way to represent your query string for a CONTAINS (or CATSEARCH) clause. With query templates, you pass a structured document, which contains the query string in a tagged element. Query templates allow you to also use a <score> element, to specify whether you want the score represented as an INTEGER or FLOAT data type.

To make use of query templates, and to represent the score as a FLOAT, you could alter the original query in this example to:

```
select score(1) relevance, doc_title, doc_url
  from htmldb_documentation
 where CONTAINS (doc_url,
                 '<query>
                    <textquery>'||:P1_SEARCH||'</textquery>
                    <score datatype="float"/>
                  </query>', 1) > 0
 order by 1 desc
```

If you run the page with the query region altered as above, your results should look similar to the following figure.

Figure 5.0 Search Term Page using Query Template



Notice that the relevance column is represented as a FLOAT data type. This is because the "datatype" attribute of the "score" element was specified with the value "float."

More Information

You can find a wealth of information about using Oracle Text, by visiting the Oracle Text page on the Oracle Technology Network at:

```
http://otn.oracle.com/products/text/index.html
```

CONCLUSION

Because Application Express engine exists inside the Oracle 10g database, you can utilize all Oracle 10g database features. This document has demonstrated how you can use Oracle Text to quickly build a text-based Web application in Oracle Application Express.

Using Oracle Application Express to Build Oracle Text Web Applications
January 2006
Author: Sharon Kennedy

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com