

Oracle Database 11g Release 2 Semantic Technologies

Semantic Indexing for Documents

Purpose

This tutorial shows how to use the Semantic Technologies feature of the Oracle Spatial Option to semantically index documents stored in relational tables and search for these documents using SPARQL-based document search criteria.

Time to Complete

Approximately 30 minutes

Topics

This tutorial covers the following steps

- [Setup the Environment](#)
- [Create an ontology of related concepts to use in document searches](#)
- [Configure the Calais information extractor](#)
- [Create the table with textual data to be indexed](#)
- [Create extractor policies](#)
- [Index documents using extractor policies](#)
- [Search for documents using extracted information](#)
- [Search documents using an ontology.](#)

Overview

Oracle Database 11g Release 2 Semantic Technologies supports the ability to search for documents of interest based on the semantics or meaning of the words in a document, a significant enhancement over keyword-based searches supported by full-text search engines. Oracle Database Semantic Technologies interoperates with semantic metadata information extractors, such as the open source General Architecture for Text Engineering (GATE) engine and the OpenCalais semantic metadata extraction service from Thomson Reuters to locate and extract meaningful information from unstructured documents.

The tutorial demonstrates this functionality using the OpenCalais information extractor to index a set of documents stored in a relational table.

Scenario

A relational table storing some brief notes about U.S states and their capitals is indexed using OpenCalais information extractor. The OpenCalais extractor identifies the references to cities as geographical locations and this information is eventually recorded in the semantic index created for the documents. The indexed documents are searched using SPARQL based query patterns.

Prerequisites

Before starting this tutorial you should.

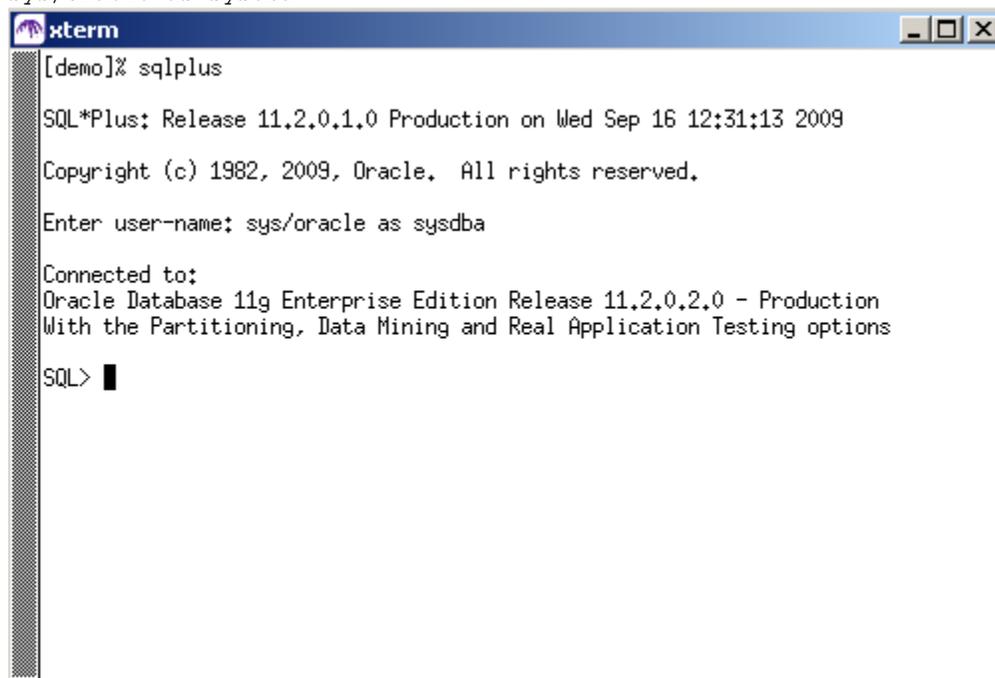
- Install Oracle Database 11g Release 2 with the Oracle Spatial and Partitioning Options.
- Install the Semantic Technologies support.
- Register at <http://www.opencalais.com> and obtain an API or license key to be used with OpenCalais web service calls for information extraction.

Setup the Environment

As a prerequisite for semantically indexing documents, you must have already created the semantic network.

1. Start SQL*Plus. At the Enter username prompt, enter the following to log-in as a privileged user:

```
sys/oracle as sysdba
```



```
[demo]% sqlplus
SQL*Plus: Release 11.2.0.1.0 Production on Wed Sep 16 12:31:13 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.
Enter user-name: sys/oracle as sysdba
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing options
SQL> █
```

2. Create the `rdf_users` tablespace and create the semantic network by running the following commands.

```
create tablespace rdf_users datafile 'rdf_users01.dbf'
    size 128M reuse autoextend on next 64M
    maxsize unlimited segment space management auto;

begin
    sem_apis.create_rdf_network('RDF_USERS');
exception
    when others then null;
end;
/
```

```
xterm
With the Partitioning, Data Mining and Real Application Testing options

SQL> @1_setup_rdf_network
SQL> -- create tablespace for the user data --
SQL> create tablespace rdf_users datafile 'rdf_users01.dbf'
      2      size 128M reuse autoextend on next 64M
      3      maxsize unlimited segment space management auto;

Tablespace created.

SQL>
SQL> -- create the RDF network if it does not exist --
SQL> begin
      2  sem_apis.create_rdf_network('RDF_USERS');
      3  exception
      4  when others then null;
      5  end;
      6  /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;
```

3. Create a user rdfusr and grant necessary privileges to the user.

```
create user rdfusr identified by rdfusr
           default tablespace rdf_users;
grant connect, resource to rdfusr;
```

```
xterm

      2  sem_apis.create_rdf_network('RDF_USERS');
      3  exception
      4  when others then null;
      5  end;
      6  /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;

SQL>
SQL> create user rdfusr identified by rdfusr
      2      default tablespace rdf_users;

User created.

SQL> grant connect, resource to rdfusr;

Grant succeeded.

SQL>
SQL> pause;
```

[Back to topics](#)

Create an RDF model to host ontology data

1. Create a table and a model to store an ontology. Use the ontology to include related concepts in document searches for more complete results. .

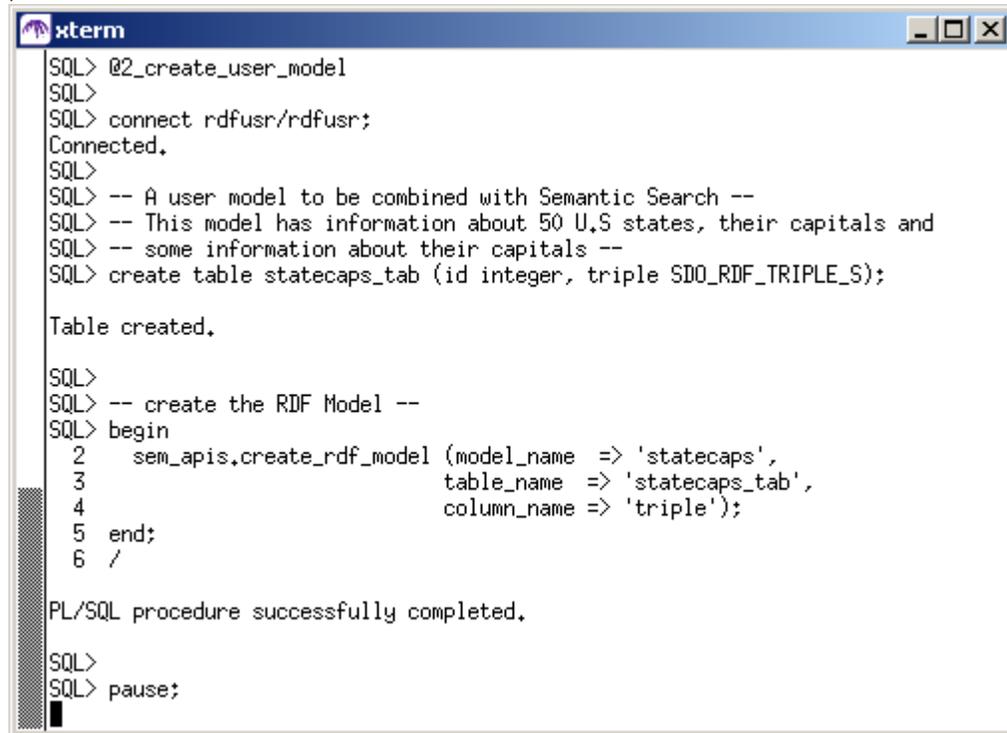
```

connect rdfusr/rdfusr;

create table statecaps_tab (id integer, triple SDO_RDF_TRIPLE_S);

-- create the RDF Model --
begin
  sem_apis.create_rdf_model (model_name => 'statecaps',
                             table_name => 'statecaps_tab',
                             column_name => 'triple');
end;
/

```



The screenshot shows an xterm window with the following text:

```

SQL> @2_create_user_model
SQL>
SQL> connect rdfusr/rdfusr;
Connected.
SQL>
SQL> -- A user model to be combined with Semantic Search --
SQL> -- This model has information about 50 U.S states, their capitals and
SQL> -- some information about their capitals --
SQL> create table statecaps_tab (id integer, triple SDO_RDF_TRIPLE_S);

Table created.

SQL>
SQL> -- create the RDF Model --
SQL> begin
2   sem_apis.create_rdf_model (model_name => 'statecaps',
3                               table_name => 'statecaps_tab',
4                               column_name => 'triple');
5 end;
6 /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;

```

2. Populate the data in the user model.

```

insert into statecaps_tab values (1, SDO_RDF_TRIPLE_S('statecaps',
'<http://geoont.org/state/Alabama>', '<http://geoont.org/prop/name>',
'"Alabama"^^xsd:string'));

insert into statecaps_tab values (51, SDO_RDF_TRIPLE_S('statecaps',
'<http://geoont.org/state/Alabama>', '<http://geoont.org/prop/statehoodIn>',
'"1819"^^xsd:integer'));
...

```

```
xterm
SQL>
SQL> pause;

SQL>
SQL> -- populate some data --
SQL> insert into statecaps_tab values (1, SDO_RDF_TRIPLE_S('statecaps',
  2 '<http://geoont.org/state/Alabama>', '<http://geoont.org/prop/name>', '"Alab
ama"^^xsd:string'));

1 row created.

SQL>
SQL> insert into statecaps_tab values (51, SDO_RDF_TRIPLE_S('statecaps',
  2 '<http://geoont.org/state/Alabama>', '<http://geoont.org/prop/statehoodIn>',
'"1819"^^xsd:integer'));

1 row created.

SQL>
SQL> prompt more data ....
more data ....
SQL> pause;
```

[Back to topics](#)

Configure the Calais Information Extractor

We will use the OpenCalais Information extractor to semantically index documents stored in a user table. To enable the database instance to access the OpenCalais web service for information extraction, the web service configuration information must be registered in the database.

1. In the SQL*Plus session connect back as a privileged user

```
connect sys/oracle as sysdba
```

2. Register the OpenCalais web service end-point and the SOAP action in the database.

```
begin
  sem_rdfctx.set_extractor_param(param_key => 'CALAIS_WS_ENDPOINT',
    param_value => 'http://apil.opencalais.com/enlighten/calais.asmx',
    param_desc => 'RDFCTX Calais web service end-point');
end;
/

begin
  sem_rdfctx.set_extractor_param(param_key => 'CALAIS_WS_SOAPACTION',
    param_value => 'http://clearforest.com/Enlighten',
    param_desc => 'RDFCTX Calais web service SOAP Action');
end;
/
```

```
xterm
SQL> @3_setup_extractor_calais
SQL> --- Set up the Calais web service based information extractor for the
SQL> -- database instance
SQL> connect / as sysdba;
Connected.
SQL>
SQL> begin
2   sem_rdfctx.set_extractor_param(param_key => 'CALAIS_WS_ENDPOINT',
3                                   param_value => 'http://api1.opencalais.com/enlighten/calais.asmx',
4                                   param_desc => 'RDFCTX Calais web service end-point');
5 end;
6 /

PL/SQL procedure successfully completed.

SQL>
SQL> begin
2   sem_rdfctx.set_extractor_param(param_key => 'CALAIS_WS_SOAPACTION',
3                                   param_value => 'http://clearforest.com/Enlighten',
4                                   param_desc => 'RDFCTX Calais web service SOAP Action');
5 end;
6 /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;
```

3. Also record the API or the license key obtained from <http://www.opencalais.com> in the database.

```
begin
  sem_rdfctx.set_extractor_param(param_key => 'CALAIS_KEY',
    param_value => '<license key>',
    param_desc => 'RDFCTX Calais extractor license key');
end;
/
```

```
xterm
SQL>
SQL> pause;

SQL>
SQL> begin
2   sem_rdfctx.set_extractor_param(param_key => 'CALAIS_KEY',
3                                   param_value => 'XXXXXXXXXXXXXXXXXXXX',
4                                   param_desc => 'RDFCTX Calais extractor license key');
5 end;
6 /

PL/SQL procedure successfully completed.

SQL>
SQL> -- if the database instance is running behind a firewall, a proxy server
SQL> -- should also be configured.
SQL> begin
2   sem_rdfctx.set_extractor_param(param_key => 'HTTP_PROXY',
3                                   param_value => 'www-proxy.us.oracle.com',
4                                   param_desc => 'RDFCTX proxy for web service requests');
5
6 end;
7 /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;
```

[Back to topics](#)

Create a table with textual data to be indexed

Connected back as RDFUSR and create a user table with VARCHAR2 column to store unstructured data to be indexed semantically.

1. In the SQL*Plus session connect as RDFUSR

```
connect rdfusr/rdfusr;
```

2. Create the table to store textual data.

```
create table statenotes (docid NUMBER, notes VARCHAR2(4000));
```

3. Populate the statenotes table with some data. The statenotes table has some text about the U.S states and their capitals in the notes column.

```
insert into statenotes values(1,
    'Montgomery is the capital of Alabama. Birmingham is the state''s
    largest city.');
```

```
insert into statenotes values(2,
    'Anchorage is the capital of Alaska. Anchorage is the state''s
    largest city.');
```

```
...
```

```
xterm
SQL> @4_create_table_with_documents
SQL>
SQL> connect rdfusr/rdfusr;
Connected.
SQL>
SQL> create table statenotes (docid NUMBER, notes VARCHAR2(1000));

Table created.

SQL>
SQL> insert into statenotes values(1,'Montgomery is the capital of Alabama. Birmingham is the
state's largest city.');
```

```
1 row created.

SQL> insert into statenotes values(2,'Anchorage is the capital of Alaska. Anchorage is the sta
te's largest city.');
```

```
1 row created.

SQL> set echo off;
more data ..
SQL> commit;

Commit complete.

SQL> █
```

[Back to topics](#)

Create Extractor policies

Extractor policies identify the information extractor to be used to index the documents and specify any preferences for the extraction. They may also include one or more user-defined RDF models to allow searching for documents based on related concepts. This example creates two extractor policies, one stipulating the OpenCalais extractor and the other combining the extracted information with the user model created earlier.

1. While connected as RDFUSR, create the base extractor policy using the OpenCalais information extractor.

```
begin
  sem_rdfctx.create_policy (policy_name => 'CAPITAL_CITIES',
                           extractor   => mdsys.calais_extractor(null));
end;
/
```

```
xterm
Commit complete.
SQL> @5_create_extractor_policy_calais.sql
SQL>
SQL> -- Create extractor policies --
SQL> begin
2   sem_rdfctx.create_policy (policy_name => 'CAPITAL_CITIES',
3                           extractor => mdsys.calais_extractor(null));
4   end;
5   /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;
```

2. Also create a dependent extractor policy that combines the extracted information with the STATECAPS user model created earlier.

```
begin
  sem_rdfctx.create_policy (policy_name => 'CAPITAL_CITIES_PLUS',
                           base_policy => 'CAPITAL_CITIES',
                           user_models => SEM_MODELS('statecaps'));
end;
/
```

```
xterm
SQL>
SQL> pause;

SQL>
SQL> begin
2   sem_rdfctx.create_policy (policy_name => 'CAPITAL_CITIES_PLUS',
3                           base_policy => 'CAPITAL_CITIES',
4                           user_models => SEM_MODELS('statecaps'));
5   end;
6   /

PL/SQL procedure successfully completed.

SQL>
SQL> pause;
```

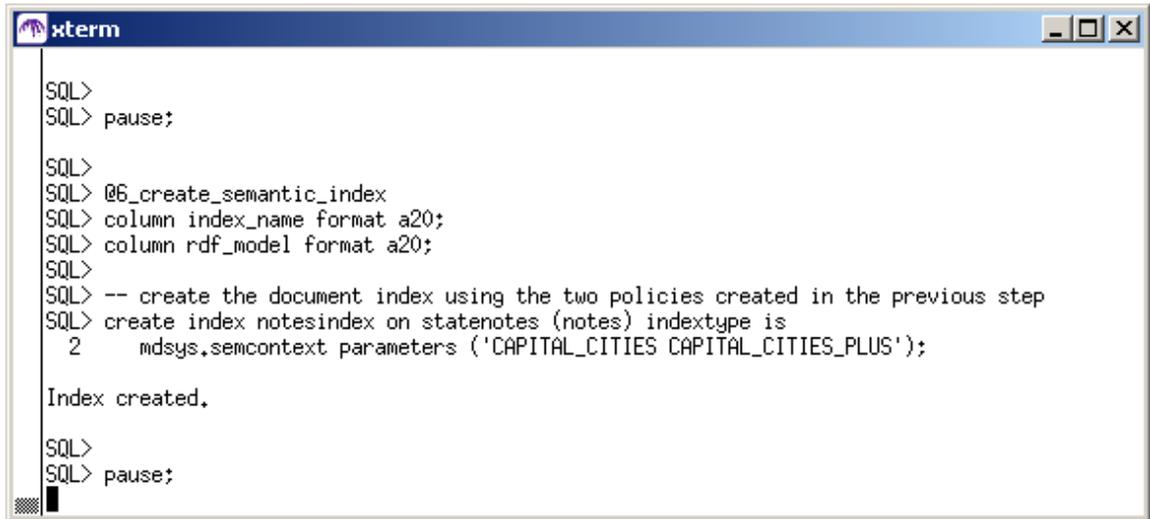
[Back to topics](#)

Index documents using Extractor policies

Create a semantic document index using the extractor policies created in the previous step.

1. Issue the CREATE INDEX command to create the index on the `Notes` column in the `Statenotes` table. Use both extractor policies created in the previous step to maintain the index metadata.

```
create index notesindex on statenotes (notes) indextype is
  mdsys.semcontext parameters ('CAPITAL_CITIES CAPITAL_CITIES_PLUS');
```



```
xterm
SQL>
SQL> pause;

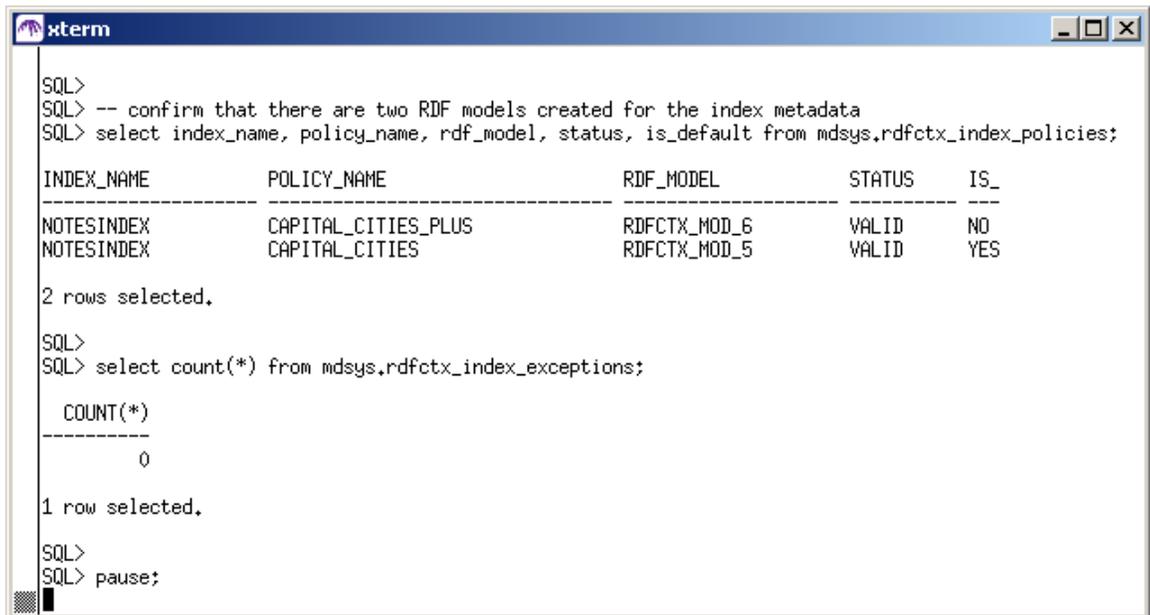
SQL>
SQL> @6_create_semantic_index
SQL> column index_name format a20;
SQL> column rdf_model format a20;
SQL>
SQL> -- create the document index using the two policies created in the previous step
SQL> create index notesindex on statenotes (notes) indextype is
  2      mdsys.semcontext parameters ('CAPITAL_CITIES CAPITAL_CITIES_PLUS');

Index created.

SQL>
SQL> pause;
```

2. Confirm the created index instances are in a valid state.

```
select index_name, policy_name, rdf_model, status, is_default
from mdsys.rdfctx_index_policies;
```



```
xterm
SQL>
SQL> -- confirm that there are two RDF models created for the index metadata
SQL> select index_name, policy_name, rdf_model, status, is_default from mdsys.rdfctx_index_policies;

INDEX_NAME          POLICY_NAME          RDF_MODEL          STATUS  IS_
-----
NOTESINDEX          CAPITAL_CITIES_PLUS  RDFCTX_MOD_6      VALID   NO
NOTESINDEX          CAPITAL_CITIES       RDFCTX_MOD_5      VALID   YES

2 rows selected.

SQL>
SQL> select count(*) from mdsys.rdfctx_index_exceptions;

COUNT(*)
-----
0

1 row selected.

SQL>
SQL> pause;
```

Note that there will be one instance of the index for each extractor policy specified in the `CREATE INDEX parameters` clause. The first policy specified is the default policy used at the time of query.

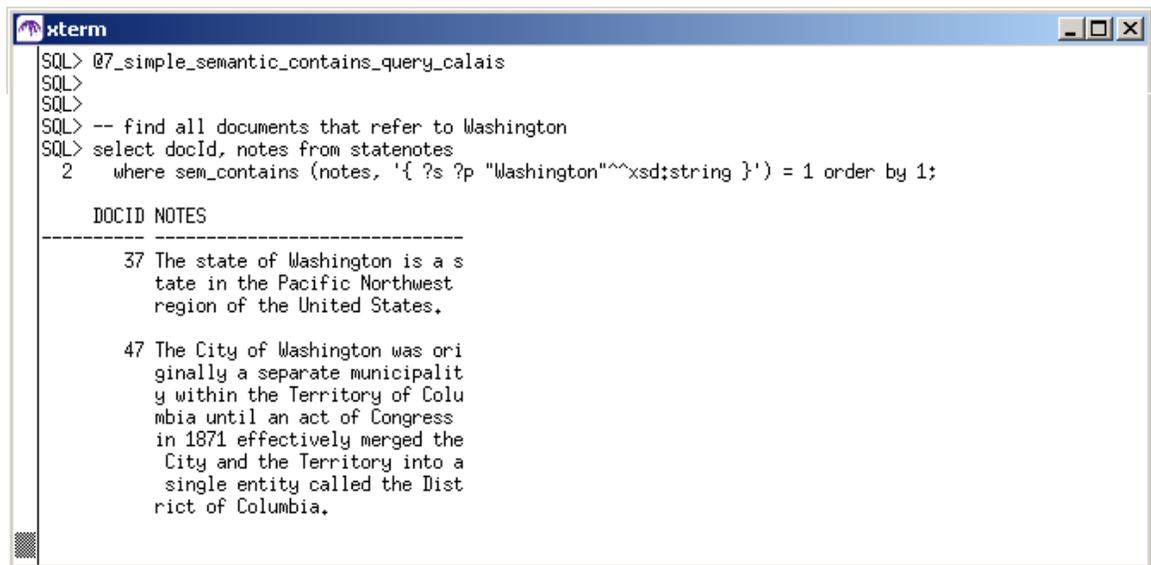
[Back to topics](#)

Search for documents using extracted information

Semantically indexed documents can be searched using a SQL query with SEM_CONTAINS operator that specifies the document search criteria. Try some sample queries.

1. Using the semantic index created on the `statenotes` table's `notes` column, find all documents that refer to "Washington". The document search criteria in this query searches for triples that have "Washington" in the object position and thus identifies the corresponding documents as the query result. Note that this is similar to a keyword search, which simply tests the existence of the given term in a document.

```
select docId, notes from statenotes
where sem_contains (notes,
                    '{ ?s ?p "Washington"^^xsd:string }') = 1
order by 1;
```



The screenshot shows an xterm window with the following content:

```
SQL> @?7_simple_semantic_contains_query_calais
SQL>
SQL>
SQL> -- find all documents that refer to Washington
SQL> select docId, notes from statenotes
  2  where sem_contains (notes, '{ ?s ?p "Washington"^^xsd:string }') = 1 order by 1;

DOCID NOTES
-----
37 The state of Washington is a s
tate in the Pacific Northwest
region of the United States.

47 The City of Washington was ori
ginally a separate municipalit
y within the Territory of Colu
mbia until an act of Congress
in 1871 effectively merged the
City and the Territory into a
single entity called the Dist
rict of Columbia.
```

2. Find the type information for the resource that matches the document search criteria. For each document matching the search criteria, the following query returns information about the matching entities as extracted by the OpenCalais information extractor. This query demonstrates the use of SEM_CONTAINS_SELECT operator to return bindings for the variables in the document search criteria.

```
select docId, notes, sem_contains_select(1) sparqlxml from statenotes
where sem_contains(notes, 'select ?s ?p1 ?o1 where
                        { ?s ?p "Washington"^^xsd:string .
                          optional { ?s ?p1 ?o1 } }', 1) = 1 order by 1;
```

```
xterm
SQL> -- find the type information about the resources that matched the document search criteria.
SQL> select docId, notes, sem_contains_select(1) sparqlxml from statenotes
2   where sem_contains(notes, 'select ?o1 where
3     { ?s ?p "Washington"^^xsd:string .
4       optional { ?s rdf:type ?o1 } }', 1) = 1 order by 1;

DOCID NOTES                                     SPARQLXML
-----
37 The state of Washington is a state in the Pacific Northwest region of the United States.
<results>
  <result>
    <binding name="01">
      <uri>http://s.opencalais.com/1/type/em/e/ProvinceOrState</uri>
    </binding>
  </result>
  <result>
    <binding name="01">
      <uri>http://s.opencalais.com/1/type/er/Geo/ProvinceOrState</uri>
    </binding>
  </result>
  <result>
    <binding name="01">
      <uri>http://s.opencalais.com/1/type/sys/InstanceInfo</uri>
    </binding>
  </result>
</results>

47 The City of Washington was originally a separate municipality within the Territory of Columbia until an act of Congress
```

3. Make use of the properties generated by the extractor to query using entity value and its type. In this case we further qualify the search criteria by the type of the resource as identified by the extractor. By querying for documents that refer to "Washington" as a city (and not a state), only a subset of documents that would have otherwise matched the keyword-based search are returned.

```
select docId, notes, sem_contains_select(1) sparqlxml from statenotes
where sem_contains (notes,
' { ?s <http://s.opencalais.com/1/pred/name> "Washington"^^xsd:string .
  ?s rdf:type <http://s.opencalais.com/1/type/em/e/City> } ', 1) = 1
order by 1;
```

```
xterm
2 rows selected.

SQL>
SQL> pause;

SQL>
SQL> -- make use of the properties generated by the extractor to query based on the literal
SQL> -- value as well as its resource type.
SQL> select docId, notes, sem_contains_select(1) sparqlxml from statenotes where sem_contains (notes,
2   ' { ?s <http://s.opencalais.com/1/pred/name> "Washington"^^xsd:string .
3     ?s rdf:type <http://s.opencalais.com/1/type/em/e/City> } ', 1) = 1 order by 1;

DOCID NOTES                                     SPARQLXML
-----
47 The City of Washington was originally a separate municipality within the Territory of Columbia until an act of Congress in 1871 effectively merged the City and the Territory into a single entity called the District of Columbia.
<results>
  <result>
    <binding name="S">
      <uri>http://d.opencalais.com/genericHasher-1/1d1529b7-da5f-3884-8de0-c765b3b7d3a3</uri>
    </binding>
  </result>
</results>

1 row selected.

SQL>
SQL> pause;
```

Search documents using an Ontology

The RDF user models associated with the extractor policy may define some standard ontologies that can be used in document searches. In the following example, `statecaps` model is used to identify documents that refer to cities that are capitals of some state for at least 200 years. This search criterion is expressed as SPARQL query pattern with `FILTER` clause.

```
select docId, notes, sem_contains_select(1) sparqlxml from statenotes where
sem_contains (notes,
'SELECT ?state ?capsince
WHERE { ?s      rdf:type    <http://s.opencalais.com/1/type/em/e/City> .
      ?s      :name ?capcity .
      ?state gp:hasCapital  ?capcity .
      ?state gp:captailSince ?capsince .
      FILTER (?capsince < 1809) }', 'CAPITAL_CITIES_PLUS',
mdsys.rdf_aliases(mdsys.rdf_alias('gs', 'http://geoont.org/state/'),
mdsys.rdf_alias('gp', 'http://geoont.org/prop/'),
mdsys.rdf_alias('', 'http://s.opencalais.com/1/pred/')),
1) = 1 order by 1;
```

```
xterm
SQL>
SQL> select docId, notes, sem_contains_select(1) sparqlxml from statenotes where
2 sem_contains (notes,
3 'SELECT ?state ?capsince
4 WHERE { ?s      rdf:type    <http://s.opencalais.com/1/type/em/e/City> .
5       ?s      :name ?capcity .
6       ?state gp:hasCapital  ?capcity .
7       ?state gp:captailSince ?capsince .
8       FILTER (?capsince < 1809) }', 'CAPITAL_CITIES_PLUS',
9 mdsys.rdf_aliases(mdsys.rdf_alias('gs', 'http://geoont.org/state/'),
10 mdsys.rdf_alias('gp', 'http://geoont.org/prop/'),
mdsys.rdf_alias('', 'http://s.opencalais.com/1/pred/')),
1) = 1 order by 1;
-----
DOCID NOTES                                SPARQLXML
-----
14 Annapolis is the capital of Ma <results>
ryland. Annapolis is the third <result>
-longest serving capital in th <binding name="STATE">
e United States after Santa Fe <uri>http://geoont.org/state/Massachusetts</uri>
and Boston. Its capitol build </binding>
ing is the oldest still in use <binding name="CAPSINCE">
. Baltimore is the state's lar <literal>1630</literal>
gest city. </binding>
</result>
<result>
<binding name="STATE">
<uri>http://geoont.org/state/New_Mexico</uri>
</binding>
<binding name="CAPSINCE">
<literal>1610</literal>
</binding>
</result>
</results>
15 Boston is the capital of Massa <results>
```

By explicitly specifying the extractor policy, `CAPITAL_CITIES_PLUS`, in the query, the document search criteria is matched against the triples extracted by OpenCalais for the Semantic index as well as the triples defined in the user model containing the ontology of states.

Due to differences in the information extractors, their default ontologies and the quality of information extraction, the result for queries involving different extractors may not match.

Summary

This tutorial is an introduction to the document indexing feature using the semantic data management capability of Oracle Spatial Option for Oracle Database 11g Release 2.

In this lesson you learned how to:

- Configure the OpenCalais information extractor
- Create extractor policies
- Semantically Index documents using extractor policies
- Search for indexed document using SPARQL-based search criteria.