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.

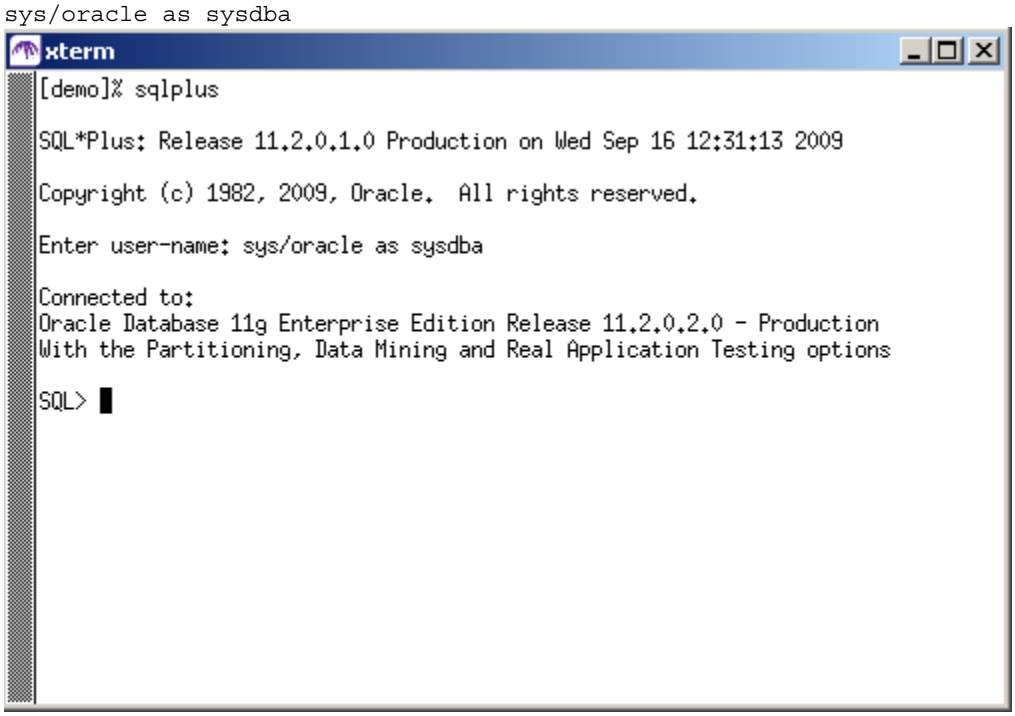
- Install Oracle Database 11g Release 2 with the Oracle Spatial and Partitioning Options.
- Install the Semantic Technologies support.
- Register at http://www.opencalais.com and obtain an API or license key to be used with OpenCalais web service calls for information extraction.

Setup the Environment

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

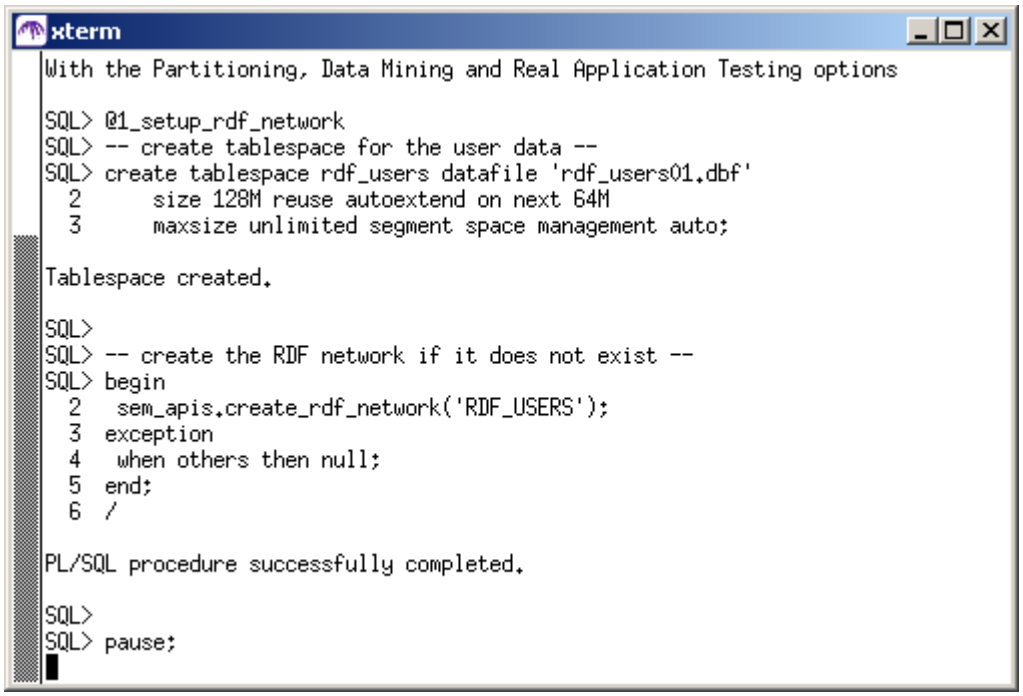
1. Start SQL*Plus. At the Enter username prompt, enter the following to log-in as a privileged user:

```
sys/oracle as sysdba
```



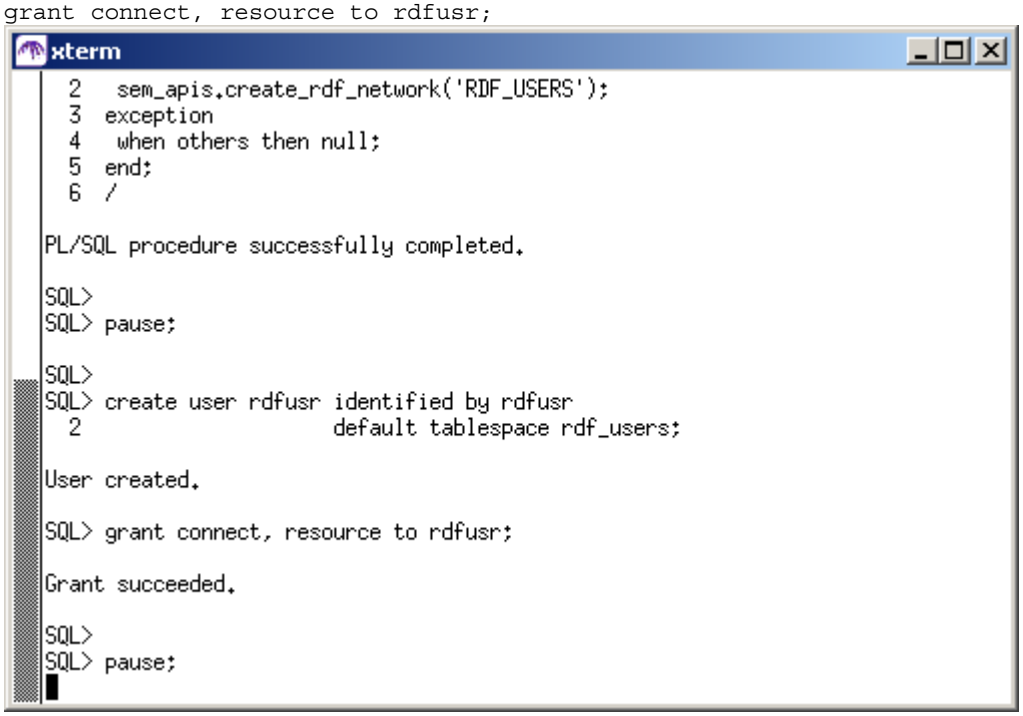
2. Create the `rdf_users` tablespace and create the semantic network by running the following commands.

```
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;
/
```



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;
```

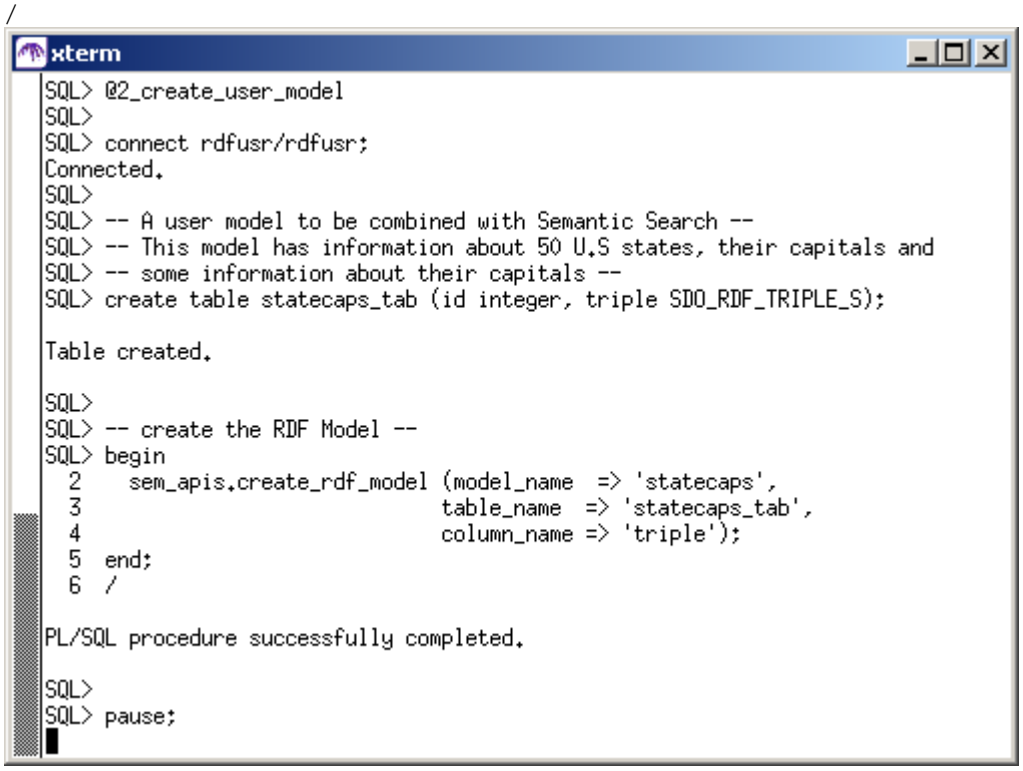


[Back to topics]

Create an RDF model to host ontology data

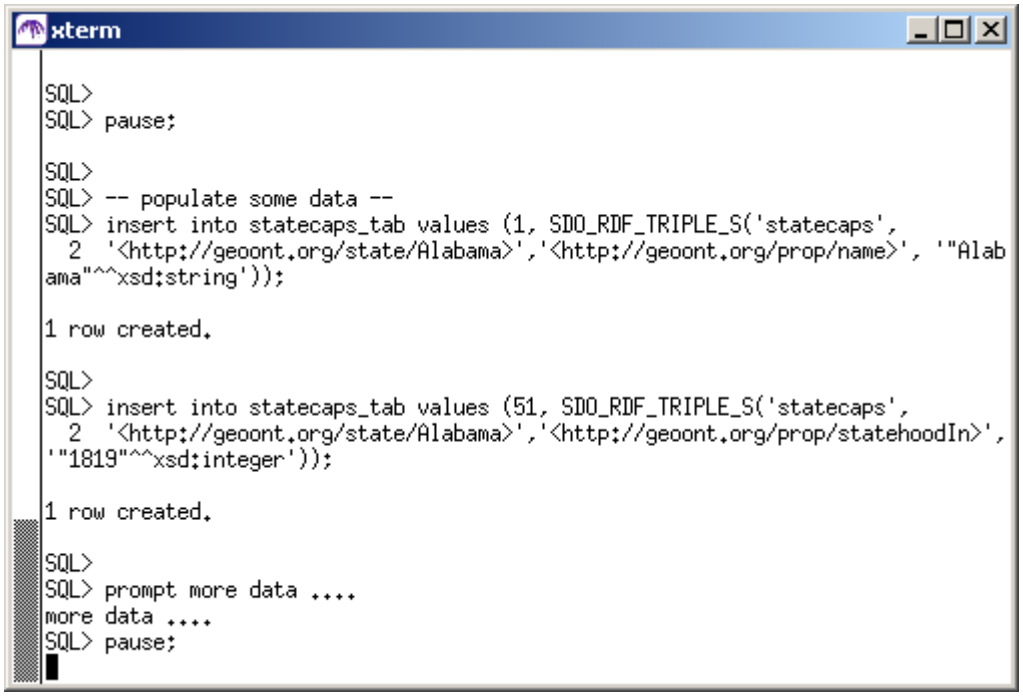
1. Create a table and a model to store an ontology. Use the ontology to include related concepts in document searches for more complete results. .

```
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;
/
```



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'));
...
```



Back to topics

Configure the Calais Information Extractor

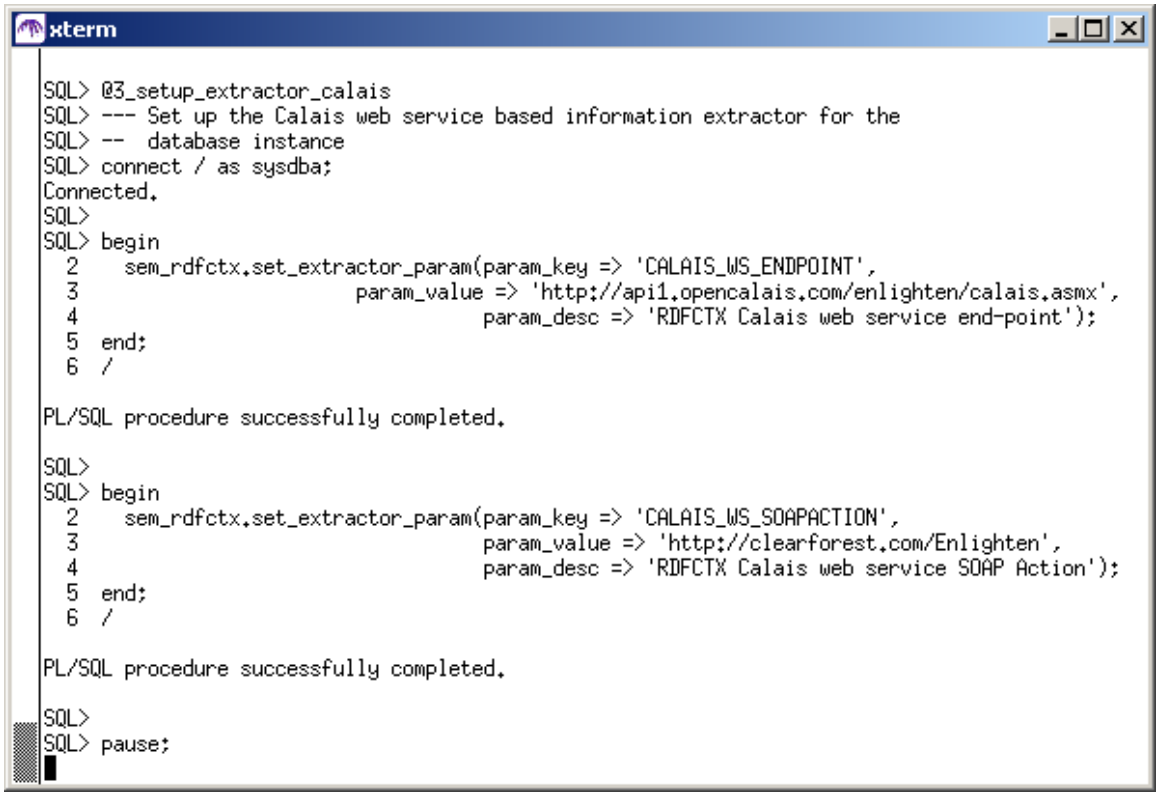
We will use the OpenCalais Information extractor to semantically index documents stored in a user table. To enable the database instance to access the OpenCalais web service for information extraction, the web service configuration information must be registered in the database.

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

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

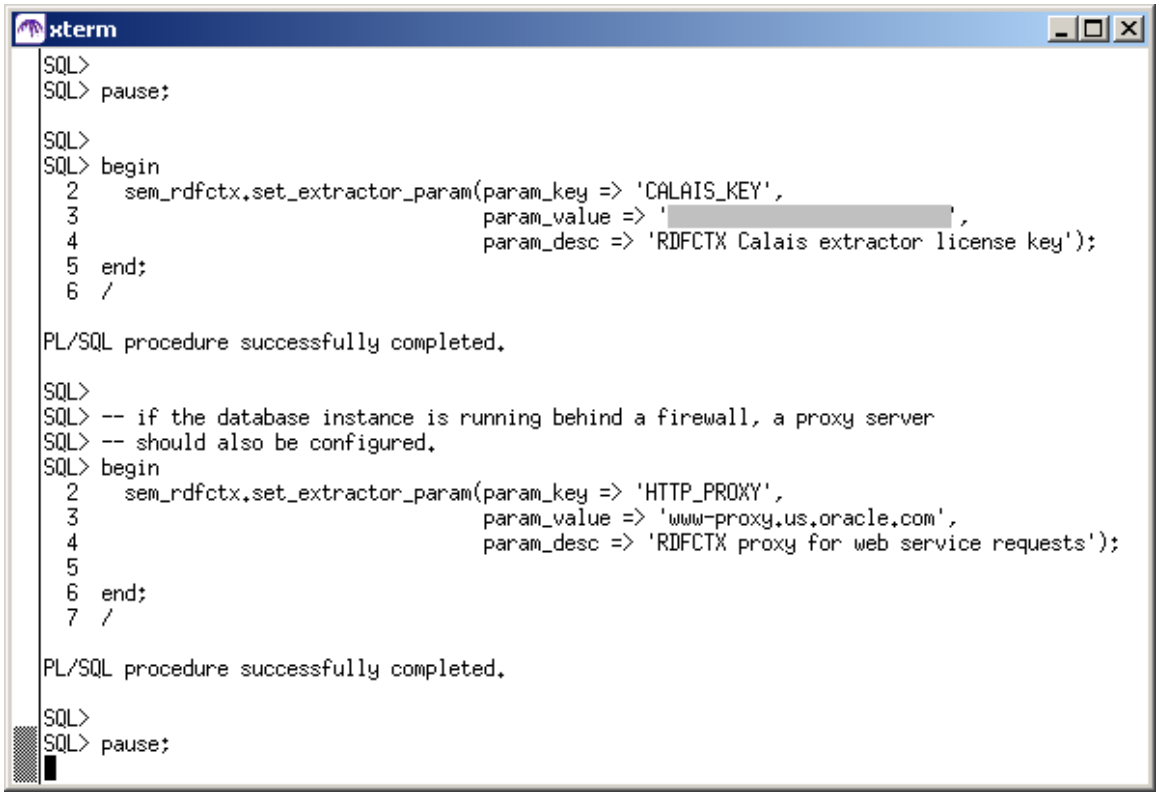
2. Register the OpenCalais web service end-point and the SOAP action in the database.

```
begin
  sem_rdfctx.set_extractor_param(param_key => 'CALAIS_WS_ENDPOINT',
       param_value => 'http://api1.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;
/
```



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;
/
```



[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

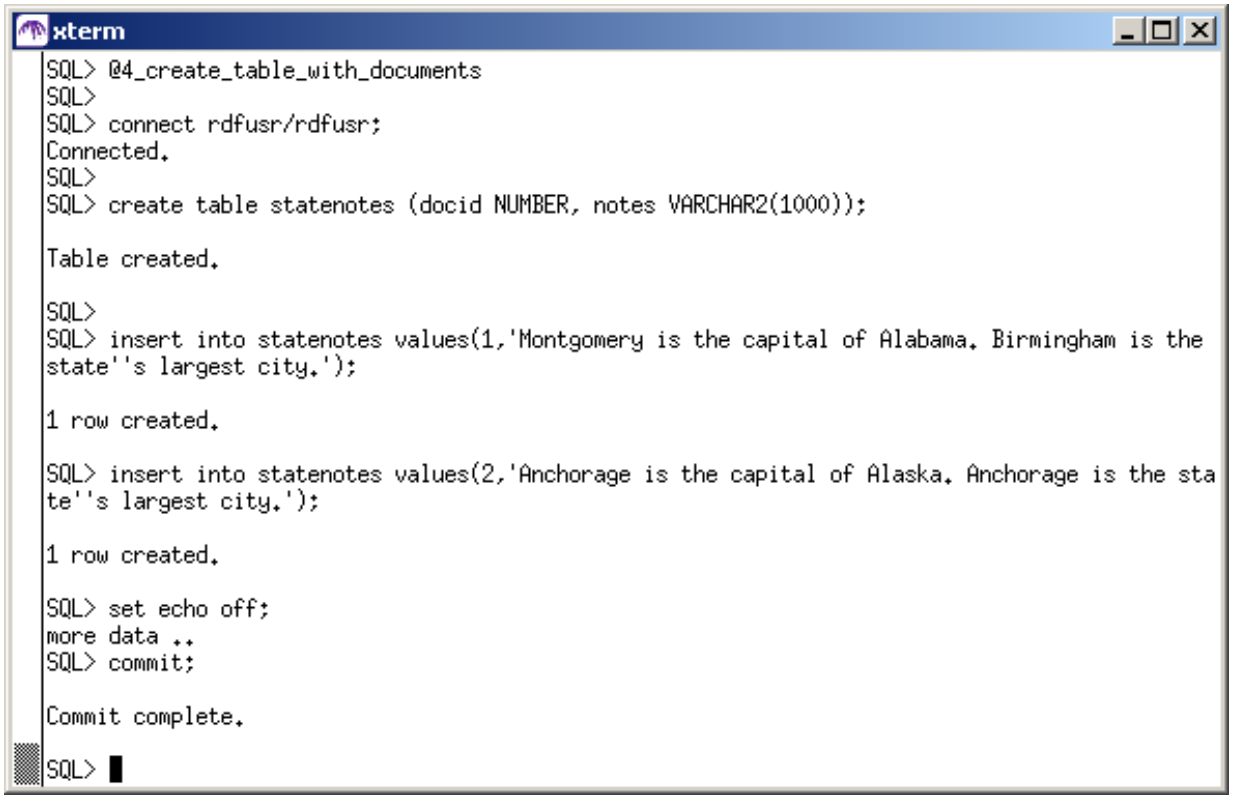
```
connect rdfusr/rdfusr;
```

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.');
...
```



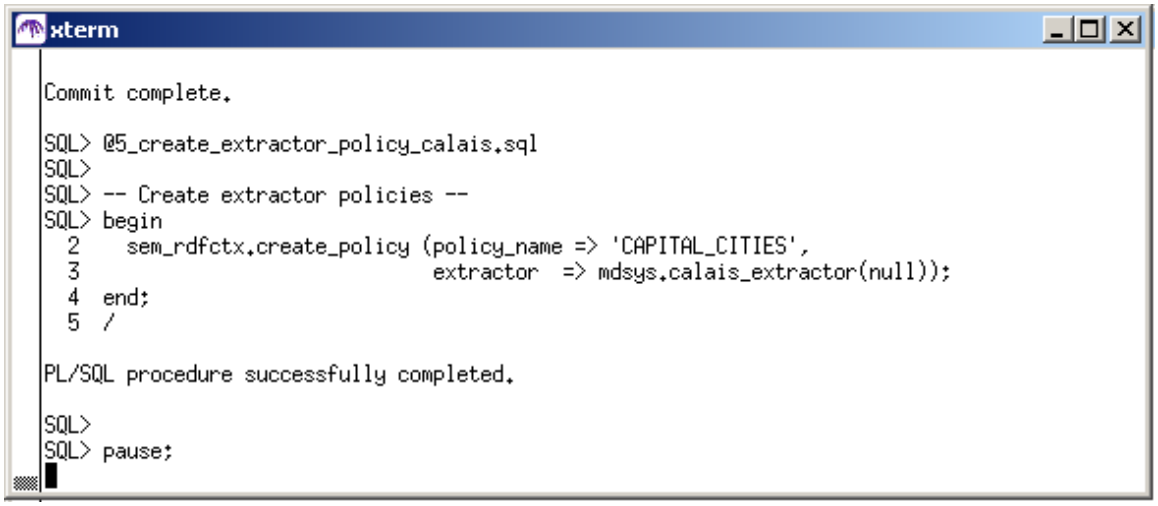
Back to topics

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.

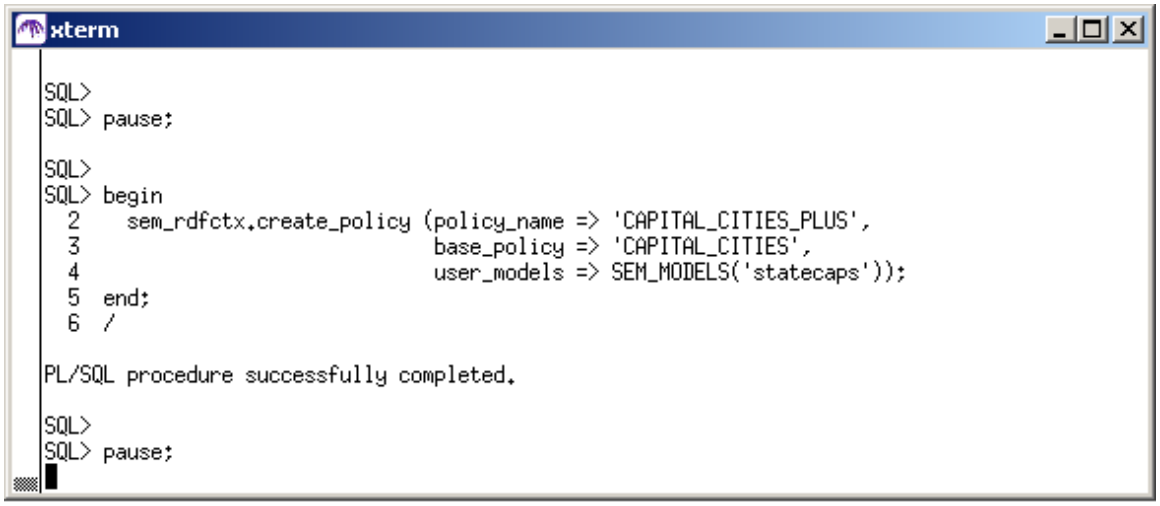
1. While connected as RDFUSR, create the base extractor policy using the OpenCalais information extractor.

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



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;
/
```



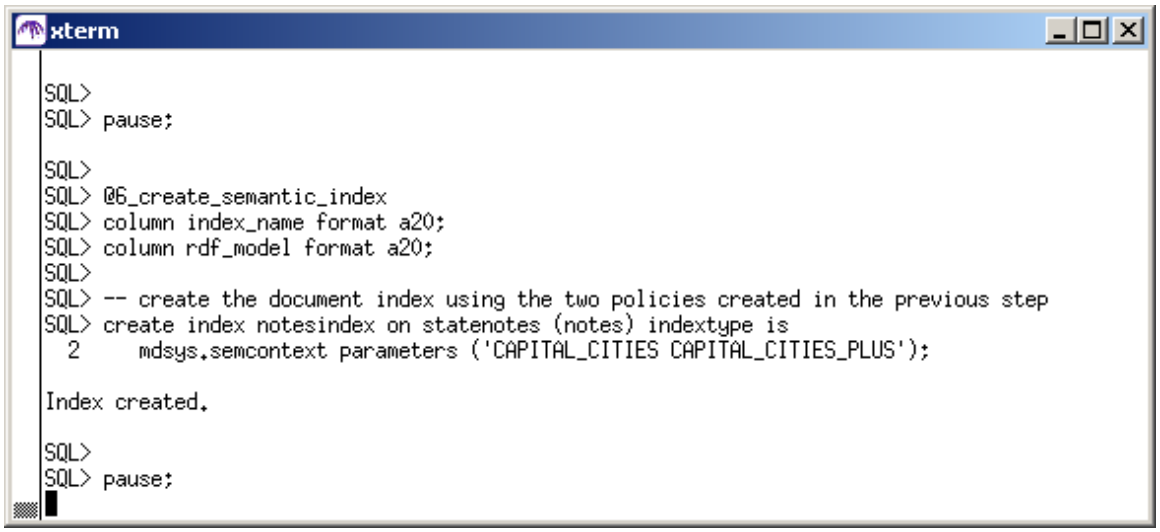
Back to topics

Index documents using Extractor policies

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

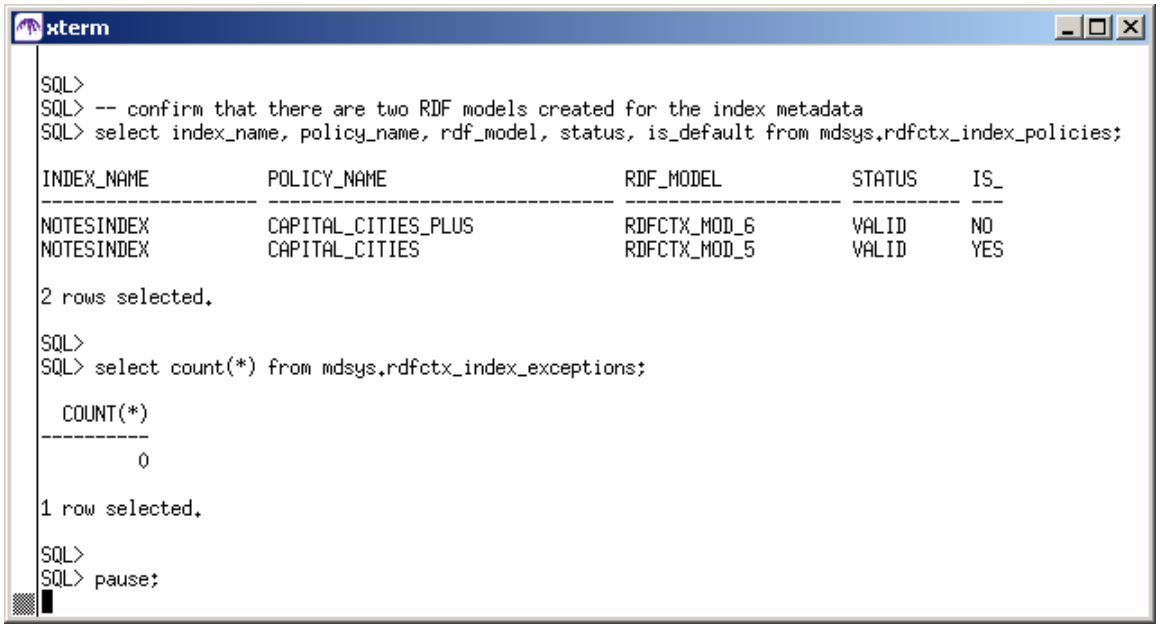
1. Issue the CREATE INDEX command to create the index on the `Notes` column in the `Statenotes` table. Use both extractor policies created in the previous step to maintain the index metadata.

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



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;
```



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.

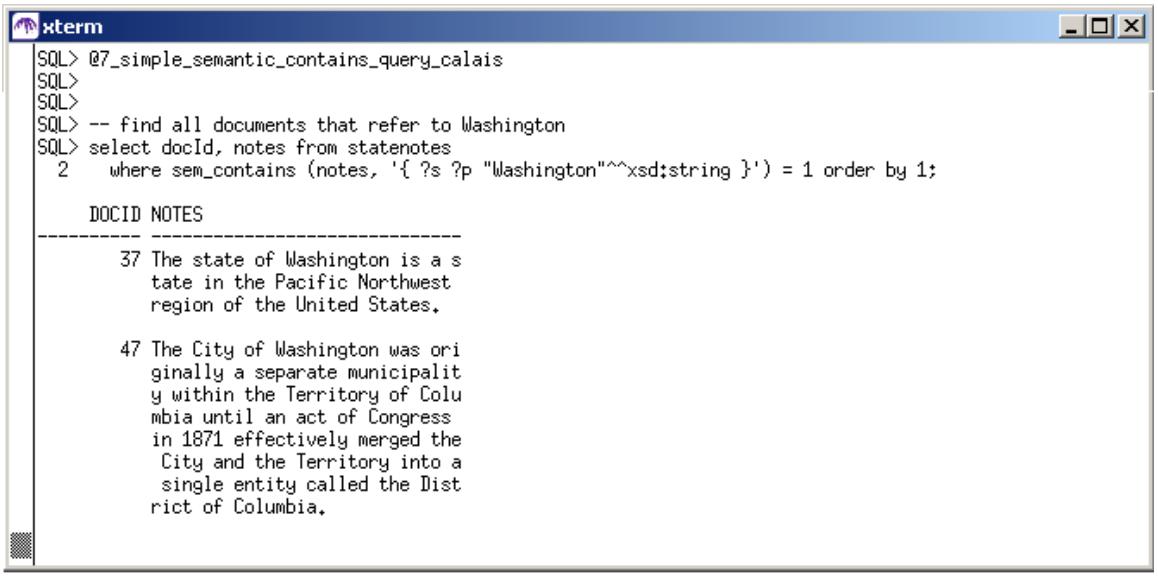
Back to topics

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.

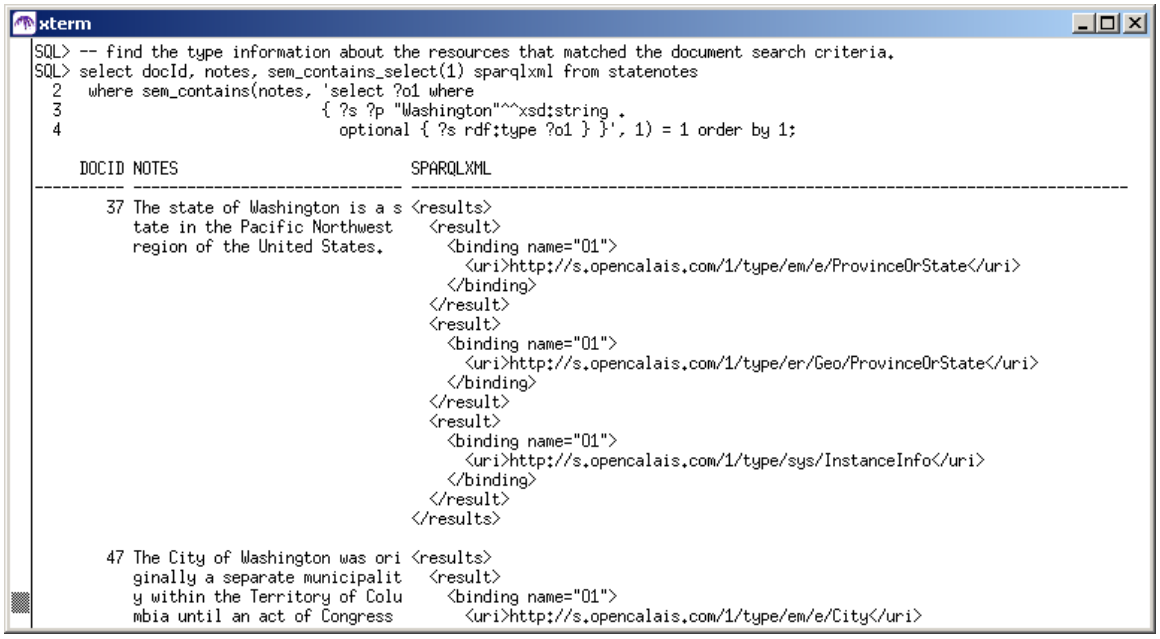
1. Using the semantic index created on the `statenotes` table's `notes` column, find all documents that refer to "Washington". The document search criteria in this query searches for triples that have "Washington" in the object position and thus identifies the corresponding documents as the query result. 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;
```



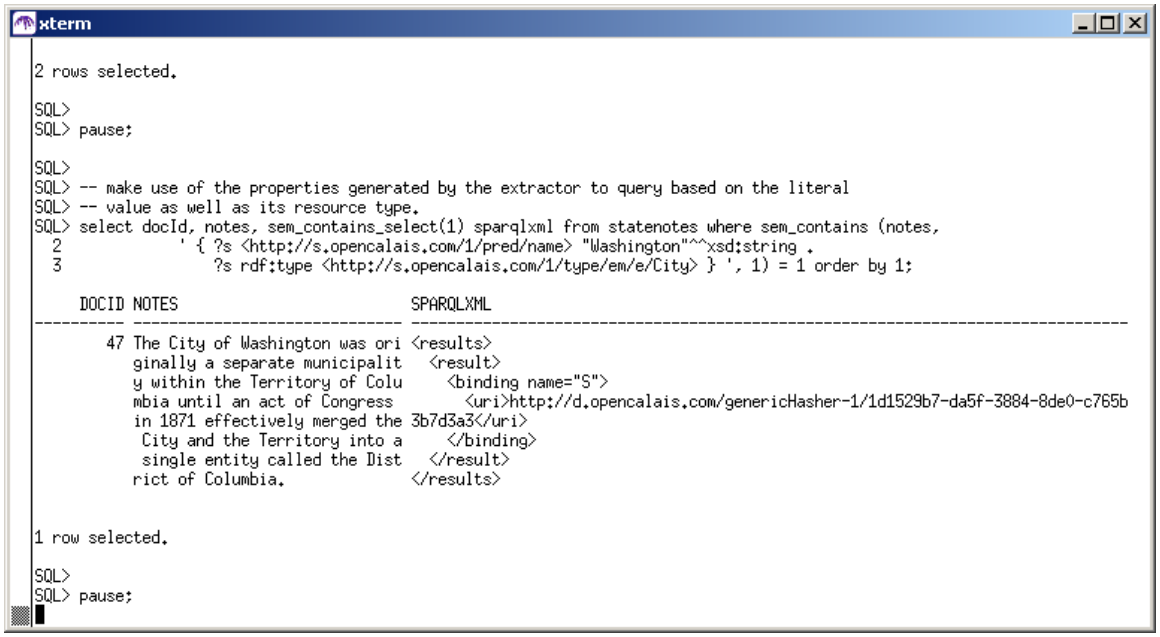
2. Find the type information for the resource that matches the document search criteria. For each document matching the search criteria, the following query returns information about the matching entities as extracted by the OpenCalais information extractor. This query demonstrates the use of SEM_CONTAINS_SELECT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



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;
```



Back to topics

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;
```

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.