How To Use the PL/SQL API of the Oracle Spatial Network Data Model

- Introduction
- What are the Prerequisites?
  - What Should You Know?
  - What Are the Software Requirements?
  - Where Can You Get Additional Information?
- How to Compute Shortest Path and Its Geometry
- How to Compute Shortest Path with a Preloaded Network Constraint
- How to Share a Preloaded Network Constraint with Another User
- How to Perform Within Cost Analysis and Access Constituent Links of a Path
- How to Perform Traveling Salesman Problem Analysis and Access Constituent Nodes of a Path
- Summary

Introduction

This demo illustrates some of the PL/SQL APIs supported by the Oracle Spatial Network Data Model:

- Shortest Path Analysis
- Within Cost Analysis
- Traveling Salesman Problem Analysis

What Are the Prerequisites?

What Should You Know?

You should be familiar with these technologies: Oracle Spatial, Oracle Spatial Network Data Model, Oracle PL/SQL.

What Are the Software Requirements?

This How-To requires that the following software components be installed and configured correctly:

- Oracle Database 10g Release 2

Where Can You Get Additional Information?

Please post your questions regarding this How-To or Oracle Spatial Network Data Model to the Oracle Spatial Discussion Forum on OTN. The URL for the forum is: http://forums.oracle.com/forums/forum.jsp?forum=76
See the Oracle Spatial Topology and Network Data Models manual for 10g Release 2 for detailed information on Network Data Model.


**How to Compute Shortest Path and Its Geometry**

To keep the path generated from shortest path analysis, you should have created path tables and registered them in the network metadata table. In addition, you need to call sdo_net_mem.path.get_geometry explicitly as shown below to populate a path’s geometry.

```sql
-- Ensure that path-related tables have been registered.

SQL> set linesize 500
SQL> select network, network_category, path_table_name,
       path_geom_column, path_link_table_name
       from user_sdo_network_metadata
          where network='HILLSBOROUGH_NETWORK';

NETWORK                  NETWORK_CATE PATH_TABLE_NAME
------------------------ ------------ --------------------------------
HILLSBOROUGH_NETWORK     SPATIAL      HILLSBOROUGH_NETWORK_PATH$
PATH_GEOMETRY                    HILLSBOROUGH_NETWORK_PLINK$
----------------------------------------------------------------------
-- In case path_table_name, path_geom_column and path_link_table_name
-- are empty, execute the following SQL statement to populate them.
----------------------------------------------------------------------
SQL> exec sdo_net.create_path_table('HILLSBOROUGH_NETWORK_PATH$','PATH_GEOMETRY');
SQL> exec sdo_net.create_path_link_table('HILLSBOROUGH_NETWORK_PLINK$');
SQL> update  user_sdo_network_metadata
     set   path_table_name='HILLSBOROUGH_NETWORK_PATH$',
           path_geom_column='PATH_GEOMETRY',
           path_link_table_name='HILLSBOROUGH_NETWORK_PLINK$
     where network = 'HILLSBOROUGH_NETWORK';

SQL> commit;

-- Perform shortest path analysis.
```

Set echo on
set serveroutput on
exec dbms_java.set_output(10000);

DECLARE
    net_mem       VARCHAR2(100);
    res_val       VARCHAR2(1000);
    path_id       NUMBER;
    res_num       NUMBER;
    cost          NUMBER;
    res_geom      SDO_GEOMETRY;
    res_varray    SDO_NUMBER_ARRAY;
    res_varray1   SDO_NUMBER_ARRAY;
    input_varray  SDO_NUMBER_ARRAY;
    indx          NUMBER;
BEGIN

    net_mem := 'HILLSBOROUGH_NETWORK';

    -- Read in the network.
    sdo_net_mem.network_manager.read_network(net_mem, 'TRUE');

    -- Validate the network.
    res_val :=
    sdo_net_mem.network_manager.validate_network_schema(net_mem);
    DBMS_OUTPUT.PUT_LINE('Is network ' || net_mem || ' valid? ' || res_val);

    DBMS_OUTPUT.PUT_LINE('---------------shortest_path---------------');
    res_num := sdo_net_mem.network_manager.shortest_path(net_mem, 1167,
              2887, NULL);
    DBMS_OUTPUT.PUT_LINE('The ID of the shortest path from node 1167' ||
              ' to node 2887 is ' || res_num);

    -- Get some characteristics of the resultant path.
    DBMS_OUTPUT.PUT_LINE('The characteristics of path ' || res_num);
    cost := sdo_net_mem.path.get_cost(net_mem, res_num);
    DBMS_OUTPUT.PUT_LINE('The path ' || res_num || ' has cost: ' || cost);
    res_val := sdo_net_mem.path.is_closed(net_mem, res_num);
    DBMS_OUTPUT.PUT_LINE('Is path ' || res_num || ' closed ? ' || res_val);

    -- Compute the geometry of the resultant path.
    sdo_net_mem.path.compute_geometry(net_mem, res_num, 0.0000005);
    res_geom := sdo_net_mem.path.get_geometry(net_mem, res_num);
    if (res_geom is NULL) then
        DBMS_OUTPUT.PUT_LINE('NULL geometry');
    end if;

    -- Add the resultant path to the network.
    sdo_net_mem.network.add_path(net_mem, res_num);

    ---
    -- Write out the network.
---
  sdo_net_mem.network_manager.write_network(net_mem);
  sdo_net_mem.network_manager.drop_network(net_mem);
END;
/

-------------------
-- Verify the result.
-------------------

SQL> SELECT path_id, path_geometry
       FROM hillsborough_network_path$
       WHERE start_node_id = 1167 AND end_node_id = 2887
/

How to Compute Shortest Path with a Preloaded Network Constraint

You can register an NDM Java network constraint using the sdo_net_mem.network_manager.register_constraint function. The subsequent network analysis functions can perform analysis with this network constraint. The following example demonstrates this option.

-------------------------------------------------------
-- Ensure that path-related tables have been registered.
-------------------------------------------------------

SQL> set linesize 500
SQL> select network, network_category, path_table_name,
          path_geom_column, path_link_table_name
       from user_sdo_network_metadata
       where network='HILLSBOROUGH_NETWORK';

NETWORK                  NETWORK_CATE PATH_TABLE_NAME             PATH_LINK_TABLE_NAME
------------------------ ------------ ------------------------- -------------------------
HILLSBOROUGH_NETWORK     SPATIAL     HILLSBOROUGH_NETWORK_PATH$           HILLSBOROUGH_NETWORK_PLINK$
----------------------------------------------------------------------
-- In case path_table_name, path_geom_column and path_link_table_name
-- are empty, execute the following SQL statement to populate them.
----------------------------------------------------------------------

SQL> exec sdo_net.create_path_table('HILLSBOROUGH_NETWORK_PATH$','PATH_GEOMETRY');
SQL> exec sdo_net.create_path_link_table('HILLSBOROUGH_NETWORK_PLINK$');
SQL> update user_sdo_network_metadata
set path_table_name='HILLSBOROUGH_NETWORK_PATH$'
path_geom_column='PATH_GEOMETRY'
path_link_table_name='HILLSBOROUGH_NETWORK_PLINK$'
where network = 'HILLSBOROUGH_NETWORK';

SQL> commit;

-- Perform shortest path analysis.
---------------------------------
Set echo on
set serveroutput on
exec dbms_java.set_output(10000);

DECLARE
    net_mem       VARCHAR2(100);
    res_val       VARCHAR2(1000);
    path_id       NUMBER;
    res_num       NUMBER;
    cost          NUMBER;
    res_geom      SDO_GEOMETRY;
    res_varray    SDO_NUMBER_ARRAY;
    res_varray1   SDO_NUMBER_ARRAY;
    input_varray  SDO_NUMBER_ARRAY;
    indx          NUMBER;
BEGIN
    net_mem := 'HILLSBOROUGH_NETWORK';
    -- Read in the network.
    sdo_net_mem.network_manager.read_network(net_mem, 'TRUE');
    -- Validate the network.
    res_val :=
        sdo_net_mem.network_manager.validate_network_schema(net_mem);
    DBMS_OUTPUT.PUT_LINE('Is network ' || net_mem || ' valid? ' || res_val);
    -- Without constraint:
    DBMS_OUTPUT.PUT_LINE('---------------shortest_path---------------');
    res_num := sdo_net_mem.network_manager.shortest_path(net_mem, 1533,
        10043, NULL);
    DBMS_OUTPUT.PUT_LINE('The ID of the shortest path from node 1533' ||
        ' to node 10043 w/o network constraint is ' || res_num);
    -- Get some characteristics of the resultant path.
    cost := sdo_net_mem.path.get_cost(net_mem, res_num);
    DBMS_OUTPUT.PUT_LINE('The path ' || res_num || ' has cost: ' || cost);
    -- Access constituent links of this path.
    res_varray1 := sdo_net_mem.path.get_link_ids(net_mem, res_num);
DBMS_OUTPUT.PUT('  Path ' || res_num || ' has links: '); FOR rec IN (SELECT column_value val FROM TABLE(res_varray1) ORDER BY column_value) LOOP  DBMS_OUTPUT.PUT(rec.val || ' '); END LOOP; DBMS_OUTPUT.PUT_LINE(chr(9)); DBMS_OUTPUT.PUT_LINE(chr(9));

-- With constraint:

-- Load the network constraint.
sdo_net_mem.network_manager.deregister_constraint('GivenProhibitedTurn');
sdo_net_mem.network_manager.register_constraint('GivenProhibitedTurn', 'ProhibitedTurnDemo', 'WORK_DIR', 'This is a network constraint' || 'that specifies certain prohibited turn network constraints');

-- Perform shortest path analysis with network constraint.
res_num := sdo_net_mem.network_manager.shortest_path(net_mem, 1533, 10043, 'GivenProhibitedTurn'); DBMS_OUTPUT.PUT_LINE('The ID of the shortest path from node 1533 to node 10043 with network constraint is ' || res_num);

-- Get some characteristics of the resultant path.
DBMS_OUTPUT.PUT_LINE('The characteristics of path ' || res_num); cost := sdo_net_mem.path.get_cost(net_mem, res_num); DBMS_OUTPUT.PUT_LINE('The path ' || res_num || ' has cost: ' || cost);

-- Access constituent links of this path.
res_varray1 := sdo_net_mem.path.get_link_ids(net_mem, res_num);
DBMS_OUTPUT.PUT('  Path ' || res_num || ' has links: '); FOR rec IN (SELECT column_value val FROM TABLE(res_varray1) ORDER BY column_value) LOOP  DBMS_OUTPUT.PUT(rec.val || ' '); END LOOP; DBMS_OUTPUT.PUT_LINE(chr(9)); DBMS_OUTPUT.PUT_LINE(chr(9));

---

-- Write out the network.
---
sdo_net_mem.network_manager.drop_network(net_mem);
END;
/

How to Share a Preloaded Network Constraint with Another User

SET serveroutput on
EXEC dbms_java.set_output(10000);

-- Assume you already have a user called mdnetwork that has a network
-- HILLSBOROUGH_NETWORK.
-- Create another user.
connect system/manager as sysdb;
drop user s_user cascade;
create user s_user identified by s_user;
grant resource, connect, create procedure, create any directory to
s_user;
connect s_user/s_user;

-- Note:
-- 1. You must first compile ProhibitedTurnDemo.java to
--    ProhibitedTurnDemo.class
-- 2. Note the 'path name' should be the full path name of the
--    operating system directory that holds ProhibitedTurnDemo.class
CREATE OR REPLACE DIRECTORY work_dir AS 'path name';

DECLARE
BEGIN
    sdo_net_mem.network_manager.deregister_constraint('ProhibitedTurn');
sdo_net_mem.network_manager.register_constraint('ProhibitedTurn',
                                                'ProhibitedTurnDemo', 'WORK_DIR', 'This is a test network
constraint ');
END;
/

connect mdnetwork/mdnetwork;
select owner, class_name from all_sdo_network_constraints where
constraint = 'ProhibitedTurn';
-- At this point, you won't be able to see the constraint
-- 'ProhibitedTurn'.

connect s_user/s_user;
grant execute on "ProhibitedTurnDemo" to mdnetwork;
connect mdnetwork/mdnetwork;
select owner, class_name from all_sdo_network_constraints where
constraint = 'ProhibitedTurn';
-- Now, you should be able to see this constraint.

-- Use the network constraint from a different schema.
DECLARE
    cost          NUMBER;
    net_mem       VARCHAR2(100);
    res_num       NUMBER := 0;
    res_val       VARCHAR2(1000);
    res_varray    SDO_NUMBER_ARRAY;
BEGIN

net_mem := 'HILLSBOROUGH_NETWORK';

-- Read in the network.
sdo_net_mem.network_manager.read_network(net_mem, 'TRUE');

-- Shortest_path without the constraint.
res_num := sdo_net_mem.network_manager.shortest_path(net_mem, 1533, 10043, null);
DBMS_OUTPUT.PUT_LINE(chr(9));
DBMS_OUTPUT.PUT_LINE('The shorest path from node 1533 to node 10043 without ' ||
                        'constraint is ' || res_num);

DBMS_OUTPUT.PUT_LINE('The characteristics of path ' || res_num);
cost := sdo_net_mem.path.get_cost(net_mem, res_num);
DBMS_OUTPUT.PUT_LINE('The path ' || res_num || ' has cost: ' || cost);
res_val := sdo_net_mem.path.is_closed(net_mem, res_num);
DBMS_OUTPUT.PUT_LINE('Is path ' || res_num || ' closed ? ' || res_val);

res_varray := sdo_net_mem.path.get_link_ids(net_mem, res_num);
DBMS_OUTPUT.PUT('Path ' || res_num || ' has links: ');
FOR rec IN (SELECT column_value val
                   FROM TABLE(res_varray)
                   ORDER BY column_value)
  LOOP
    DBMS_OUTPUT.PUT(rec.val || ' ');
  END LOOP;
DBMS_OUTPUT.PUT_LINE(chr(9));

-- Shortest_path with the constraint.
-- Note: you need specify the constraint as
-- 'schema_name:constraint_name' if you access another user's
-- constraint.
res_num := sdo_net_mem.network_manager.shortest_path(net_mem, 1533, 10043, 's_user:ProhibitedTurn');
DBMS_OUTPUT.PUT_LINE(chr(9));
DBMS_OUTPUT.PUT_LINE('The shorest path from node 1533 to node 10043 with ' ||
                        'constraint is ' || res_num);

DBMS_OUTPUT.PUT_LINE('The characteristics of path ' || res_num);
cost := sdo_net_mem.path.get_cost(net_mem, res_num);
DBMS_OUTPUT.PUT_LINE('The path ' || res_num || ' has cost: ' || cost);
res_val := sdo_net_mem.path.is_closed(net_mem, res_num);
DBMS_OUTPUT.PUT_LINE('Is path ' || res_num || ' closed ? ' || res_val);

res_varray := sdo_net_mem.path.get_link_ids(net_mem, res_num);
DBMS_OUTPUT.PUT('Path ' || res_num || ' has links: ');
FOR rec IN (SELECT column_value val
                   FROM TABLE(res_varray)
                   ORDER BY column_value)
LOOP
    DBMS_OUTPUT.PUT(rec.val || ' ');
END LOOP;
DBMS_OUTPUT.PUT_LINE(chr(9));
DBMS_OUTPUT.PUT_LINE(chr(9));

-- Wrap up
sdo_net_mem.network_manager.drop_network('HILLSBOROUGH_NETWORK');

END;
/

-- Connect to your system account.
connect system/manager as sysdba;
drop directory temp_dir;
drop user s_user cascade;

How to Perform Within Cost Analysis and Access Constituent Links of a Path

-----------------------------
-- Perform within cost analysis.
-----------------------------
set echo on
set serveroutput on
exec dbms_java.set_output(10000);

DECLARE
    net_mem       VARCHAR2(100);
    res_val       VARCHAR2(1000);
    path_id       NUMBER;
    res_num       NUMBER;
    cost          NUMBER;
    res_geom      SDO_GEOMETRY;
    res_varray    SDO_NUMBER_ARRAY;
    res_varray1   SDO_NUMBER_ARRAY;
    input_varray  SDO_NUMBER_ARRAY;
    indx          NUMBER;
BEGIN

    net_mem := 'HILLSBOROUGH_NETWORK';

    -- Read in the network.
    sdo_net_mem.network_manager.read_network(net_mem, 'TRUE');

    -- Validate the network.
    res_val :=
        sdo_net_mem.network_manager.validate_network_schema(net_mem);
    DBMS_OUTPUT.PUT_LINE('Is network ' || net_mem || ' valid? ' ||
        res_val);

    ---
-- Perform Within Cost Analysis.
---
DBMS_OUTPUT.PUT_LINE('---------------within_cost---------------');
res_varray := sdo_net_mem.network_manager.within_cost(net_mem, 1167, 2000, NULL);
DBMS_OUTPUT.PUT_LINE('Node 1167 has these paths within cost of 2000:');
FOR indx IN res_varray.FIRST..res_varray.LAST LOOP
  res_num := res_varray(indx);
  DBMS_OUTPUT.PUT_LINE('  The path ' || res_num || ' has following properties: ');
  cost := sdo_net_mem.path.get_cost(net_mem, res_num);
  DBMS_OUTPUT.PUT_LINE('  The path ' || res_num || ' cost: ' || cost);
  -- Access constituent links of this path.
  res_varray1 := sdo_net_mem.path.get_link_ids(net_mem, res_num);
  DBMS_OUTPUT.PUT('  Path ' || res_num || ' has links: ');
  FOR rec IN (SELECT column_value val FROM TABLE(res_varray1) ORDER BY column_value) LOOP
    DBMS_OUTPUT.PUT(rec.val || ' ');
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(chr(9));
  DBMS_OUTPUT.PUT_LINE(chr(9));
END LOOP;
--
-- Drop the network after analysis.
--
sdo_net_mem.network_manager.drop_network(net_mem);
END;
/

How to Perform Traveling Salesman Problem Analysis and Access Constituent Nodes of a Path

---------------------------------------------
-- Perform TSP analysis.
---------------------------------------------
set echo on
set serveroutput on
exec dbms_java.set_output(10000);

DECLARE
  net_mem       VARCHAR2(100);
  res_val       VARCHAR2(1000);
  path_id       NUMBER;
  res_num       NUMBER;
  cost          NUMBER;
  res_geom      SDO_GEOMETRY;


res_varray    SDO_NUMBER_ARRAY;
res_varray1   SDO_NUMBER_ARRAY;
input_varray  SDO_NUMBER_ARRAY;
indx          NUMBER;
BEGIN

net_mem := 'HILLSBOROUGH_NETWORK';

-- Read in the network.
sdo_net_mem.network_manager.read_network(net_mem, 'TRUE');

-- Validate the network.
res_val :=
   sdo_net_mem.network_manager.validate_network_schema(net_mem);
DBMS_OUTPUT.PUT_LINE('Is network ' || net_mem || ' valid? ' || res_val);

---
---
-- Perform Traveling Salesman Analysis.
---
DBMS_OUTPUT.PUT_LINE('---------------TSP---------------');
input_varray := SDO_NUMBER_ARRAY(1167, 2887, 2487, 2038);
res_num :=
   sdo_net_mem.network_manager.tsp_path(net_mem,input_varray,
      'FALSE','TRUE',NULL);
DBMS_OUTPUT.PUT_LINE(' The TSP path for node 1167, 2887, 2487, 2038
w/o constraint is ' || res_num);
res_varray := sdo_net_mem.path.get_node_ids(net_mem, res_num);
DBMS_OUTPUT.PUT('TSP path ' || res_num || ' has links: ');
FOR rec IN (SELECT column_value val
         FROM TABLE(res_varray)
         ORDER BY column_value)
   LOOP
      DBMS_OUTPUT.PUT(rec.val || ' ');
   END LOOP;
DBMS_OUTPUT.PUT_LINE(chr(9));

--
---
-- Drop the network after analysis.
--
   sdo_net_mem.network_manager.drop_network(net_mem);
END;
/

Summary

In this document, you should have:
• become aware that the Oracle Spatial Network Data Model (NDM) supports a fully functional PL/SQL Interface
• learned to write NDM PL/SQL queries to perform some common network analysis functions and to access network information
• learned to use a preloaded network constraint with NDM PL/SQL queries and share a preloaded network constraint with other users