Oracle Network Model

Oracle Network Model

- Oracle has built-in network model for
  - nodes (node table)
  - link (link table)
  - paths (path and path link table)
- can be created by hand (following conventions) *or* using built-in functions
- supports network analysis functions
- for more details, see Chapter 10 of *Pro Oracle Spatial* by Kothuri, Godfrink, Beinat available online at books 24x7 (http://www.lib.depaul.edu/Find/resourceList.aspx?s=89)

Conventions

- node table:
  node_id number
- link table:
  link_id, start_node_id, end_node_id number
- path table:
  path_id, start_node_id, end_node_id number
- path link table:
  path_id, link_id, seq_no number

Also: for the network to validate and some of the functiosn to work

- include geometry attributes in node, link and path, all of type sdo_geometry
- names should be: node_name, link_name, path_name
- create metadata

Example: creating tables

```
create table ctanode (
  node_id number primary key,
  node_name   varchar2(30),
  location   sdo_geometry, -- the real geometry,
  -- we'll use our homemade version for now
  our_loc     gpoint); -- homemade geometry

create table ctalink (
  link_id  number primary key,
  link_name     varchar2(40),
  line      varchar2(32),
  direction varchar2(5),
  start_node_id  number,
  end_node_id    number,
  distance      number,
  link_geom sdo_geometry
);
```

```
create table ctapath (
  path_id  number primary key,
  path_name     varchar2(30),
  start_node_id  number,
  end_node_id    number,
  cost          number,
  path_geom     sdo_geometry
);

create table ctaplink (
  path_id  number,
  link_id  number,
  seq_no   number,
  constraint ctaplink_pk primary key (path_id, link_id)
);
```

Example: creating metadata

```
insert into user_sdo_network_metadata (network,
    network_category, geometry_type,
    no_of_hierarchy_levels, no_of_partitions,
    link_direction, node_table_name,
    node_geom_column, node_cost_column,
    link_table_name, link_geom_column,
    link_cost_column, path_table_name,
    path_geom_column, path_link_table_name)
values ('CTA_ROUTES', 'spatial', 'sdo_geometry', 1, 1,
    'directed', 'ctanode', 'location', null, 'ctalink',
    'link_geom', 'distance', 'ctapath', 'path_geom',
    'ctaplink');
```

Nearly done, but

- test that everything went fine:

```
select sdo_net.validate_network('CTA_ROUTES') from dual;
```

- to read network into memory (for analysis):

```
execute sdo_net_mem.network_manager.read_network('CTA_ROUTES',
    'FALSE');  -- false means read-only, true allows write
```

- to drop network from memory (not database):

```
execute sdo_net_mem.network_manager.drop_network('CTA_ROUTES');
```

- to write network into memory (after changes):

```
execute sdo_net_mem.network_manager.read_network('CTA_ROUTES');
```

Basic functions in sdo_net

- get_no_of_nodes()
  number of nodes in the network
- get_no_of_links()
  number of links in the network
- get_isolated_nodes()
  isolated nodes (no links) in network
- get_invalid_links()
- get_invalid_paths()

Basic function example

```
set serveroutput on size 10000;
declare
  x integer;
begin
  x := sdo_net.get_no_of_nodes('CTA_ROUTES');
  dbms_output.put_line('No of nodes: ' || x);
end;
```

More basic functions

- get_node_degree()
  number of incident links
- get_node_in_degree()
  number of links going out
- get_node_out_degree()
  number of links coming in
- get_in_links()
- get_out_links()

How to deal with a list: show all links out of Belmont

```
set serveroutput on size 10000;
declare
  x number;
link_ids sdo_number_array;
begin
  select node_id into x
  from ctanode
  where node_name = 'Belmont';
  link_ids := sdo_net.get_in_links('CTA_ROUTES', x);
  for i in link_ids.first..link_ids.last loop
       dbms_output.put_line(link_ids(i));
  end loop;
end;
```

modify so we see the names of stations we can get to

Analysis functions

- sdo_net_mem.network_manager.shortest_path(<network>, <start_node>, <end_node>)
  - shortest path in terms of cost
  - returns an integer identifying a path, information can be accessed using:
- sdo_net_mem.path.get_cost(<network>, <path>)
- sdo_net_mem.path.get_link_ids(<network>, <path>)
- sdo_net_mem.path.is_simple(<network>, <path>)
- sdo_net_mem.link.get_name(<link>) and node.get_name(<node>)
- sdo_net_mem.link.get_cost(<link>) and node.get_cost(<node>)
- sdo_net_mem.link.get_start_node_id(<link)
- sdo_net_mem.link.get_end_node_id(<link)

Exercise: Write PL/SQL function that finds shortest path between two stations and lists the path taken.

Shortest in what sense?

- if cost not specified:
  - cost(link) = 1, cost(node) = 0
- up to you, can define arbitrary cost
  - time, length
  - can define multiple networks over same tables
  - if tables don’t have the right structure, create views and set up network over views

Exercises:

- what could the cost of a node mean?
- how would you model smallest number of changing trains?

Find close neighbors:

- sdo_net_mem.network_manager.nearest_neighbors(<network>, <start_node>, k)
  - returns the k closest neighbors
- sdo_net_mem.network_manager.within_cost(<network>, <start_node>, c)
  - returns neighbors within cost c

- both functions use the cost attribute
- both functions do not return just the nodes, but arrays of paths to the nodes

  can use sdo_net_mem.path.get_end_node_id

Exercise: find stations within half a mile of Belmont

Finally, TSP

sdo_net_mem.network_manager.tsp_path(<network>, <nodes>, <is_closed>, <use_exact_cost>);

- Very restricted: list of nodes needs to be supplied
- is_closed: ‘TRUE’ forces return to first node

Automatic Network Set-up

- contains functions to automatically generate network tables and metadata, e.g.

```
begin
  sdo_net.create_sdo_network (
    network => 'CTA',
    no_of_hierarchy_levels => 1,
    is_directed => TRUE,
    node_with_cost => FALSE
  );
end;
```

- will set up tables CTA_NODE$, CTA_PATH$, CTA_LINK$, CTA_PLINK$
- there are more detailed versions (allowing specification of attribute names)