

Oracle Spatial - inside
the Square

Simon Greener

www.SpatialDBAdvisor.com



Presentation

- ◆ This presentation will provide a “taster” for some non-traditional ways that Oracle Spatial/Locator data can be used in consort with other Oracle database technologies.
- ◆ I will provide practical examples of the use of:
 - Generation of test data;
 - Function based indexes
 - XMLDB
 - SQL99 OLAP functions
- ◆ **ALL DONE WITH ORACLE XE!**

Generation of test data

- ◆ I often have to generate test data for use in activities such as tuning or deciding what data model choices would be best for my customer's applications.
- ◆ I will now show how to generate test data but, for the sake of brevity 😊 I will only generate some point data.

Random Point Data

◆ Here is a table definition:

```
CREATE TABLE ProjXYZ3D (  
  Id      Integer,  
  X       Number,  
  Y       Number,  
  Z       Number )  
PCTFREE 99 TABLESPACE USERS  
NOLOGGING;
```

Some Constants..

Prompt Define some constants and variables ...

```
DEFINE mysrid=82468
VARIABLE CX number
BEGIN :CX := 358880; END;
/
VARIABLE CY number
BEGIN :CY := 5407473; END;
/
VARIABLE MINZ number
BEGIN :MINZ := -1000; END;
/
VARIABLE MAXZ number
BEGIN :MAXZ := 1000; END;
/
VARIABLE CZ number
BEGIN :CZ := :MAXZ - :MINZ; END;
/
VARIABLE EXTZ number
BEGIN :EXTZ := 10000; END;
/
VARIABLE EXTY number
BEGIN :EXTY := 5000; END;
/
VARIABLE NUMPTS number
BEGIN :NUMPTS := 1000; END;
/
```

Populate Table: ProjXYZ3D

```
INSERT /*+APPEND*/ INTO ProjXYD3D (ID,X,Y,Z)
SELECT ROWNUM,
        ROUND(dbms_random.value(:CX - ( :EXTX / 2 ),
                                :CX + ( :EXTX / 2 )),2),
        ROUND(dbms_random.value(:CY - ( :EXTY / 2 ),
                                :CY + ( :EXTY / 2 )),2),
        ROUND(dbms_random.value(:MINZ, :MAXZ),2)
FROM DUAL
CONNECT BY LEVEL <= :NUMPTS;
COMMIT;
ALTER TABLE ProjPoint3D LOGGING;
```

Function based Indexes

- ◆ Now, I would like to build an RTree index on the spatial data ProjXYZ3D contains and use the data in run-of-the-mill GIS software.
 - Note:
 - ◆ It is common to find tables with X, Y (and, occasionally Z) columns in them.
 - ◆ Often it is hard to change a production table to include an SDO_Geometry column.
- ◆ How?
 - Via a *Function-Based* Index!

Function-Based Index

◆ I will now:

- Explain what is a function based index;
- Show how to construct one;
- Then use one (and a View) to create a spatial object that can be queried and rendered by a GIS package.

Function-Based Indexes...

- ◆ Oracle provides the ability to index functions and use these indexes in queries.
 - For example, we have mixed case data in a “Customer_Name” column but want to query the data using upper-case queries..
 - If the Customer_Name column is indexed a query such as:

```
SELECT Customer_Id, Billed_Amount  
FROM Customer  
WHERE UPPER(Customer_Name) = 'JOHN SMITH'
```

will NOT use the index to locate the customer records.

Function-Based Indexes (2)

◆ But we can do the following...

```
CREATE INDEX Customer_Upper_idx  
      ON Customer(UPPER(Customer_Name));
```

Which we can then use as follows...

```
SELECT Customer_Id, Billed_Amount  
      FROM Customer  
      WHERE UPPER(Customer_Name) = 'JOHN SMITH';
```

but this time, the query optimizer will use the function-based index.

Why?

- ◆ Why use this feature (*Tom Kyte*)?
 - It's easy and provides immediate value
 - It can be used to speed up existing applications without changing any of their logic or queries
 - It can be used to supply additional functionality to applications with very little cost
- ◆ So, armed with this knowledge, we can create function-based RTree index!

First, Create Metadata: ProjXYZ3D

Prompt Let's not do this manually...

DECLARE

v_geom mdsys.sdo_geometry;

BEGIN

SELECT SDO_AGGR_MBR(SDO_GEOMETRY(
3001, &&MYSRID., SDO_POINT_TYPE(X, Y, Z), NULL, NULL))

INTO v_geom

FROM ProjPoint3D;

INSERT INTO USER_SDO_GEOM_METADATA(

SRID, TABLE_NAME, COLUMN_NAME, DIMINFO

) **VALUES** (

v_geom.sdo_srid, 'PROJPOINT3D',

'MDSYS.SDO_GEOMETRY(3001, '||v_geom.sdo_srid||

', SDO_POINT_TYPE(X, Y, Z), NULL, NULL)',

MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', v_geom.SDO_ORDINATES(1), v_geom.SDO_ORDINATES(4), 0.05),

MDSYS.SDO_DIM_ELEMENT('Y', v_geom.SDO_ORDINATES(2), v_geom.SDO_ORDINATES(5), 0.05),

MDSYS.SDO_DIM_ELEMENT('Z', v_geom.SDO_ORDINATES(3), v_geom.SDO_ORDINATES(6), 0.05))

);

END;

/

Notice
Constructor
Function

Create RTree Index: PointXYZ3D

◆ We create a function-based RTree index as follows...

```
CREATE INDEX ProjXYZ3D_Geom
ON ProjXYD3D(
    MDSYS.SDO_GEOMETRY(3001,&&MYSRID.,
    SDO_POINT_TYPE(X,Y,Z),NULL,NULL))
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('sdo_indx_dims=2, layer_gtype=point');
```

◆ Notes:

- The Function being used is the SDO_Geometry constructor.
- The function “*signature*” should be **exactly** the same as in the column_name column of the relevant USER_SDO_GEOM_METADATA entry.
- The MDSYS prefix IS REQUIRED both in the **INDEX** statement and in the column_name in USER_SDO_GEOM_METADATA entry.

Example Queries...

```
SELECT COUNT (*)
FROM ProjXYD3D pp
WHERE SDO_INSIDE (
SDO_GEOMETRY(3001, &&MYSRID., SDO_POINT_TYPE(pp.X, pp.Y, pp.Z), NULL, NULL),
SDO_GEOMETRY(2003, &&MYSRID., NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 3),
SDO_ORDINATE_ARRAY(:CX - 1000, :CY - 500, :CZ,
:CX + 1000, :CY + 500, :CZ))
) = 'TRUE';
```

Prompt Find nearest 10 points to centre of the table extent.

```
SELECT ID
FROM ProjXYD3D pp
WHERE SDO_NN (
SDO_GEOMETRY(3001, &&MYSRID., SDO_POINT_TYPE(pp.X, pp.Y, pp.Z), NULL, NULL),
SDO_GEOMETRY(3001, &&MYSRID., SDO_POINT_TYPE(:CX, :CY, :CZ), NULL, NULL),
'sdo_num_res=10'
) = 'TRUE';
```

Note: There is a problem with using MDSYS.SDO_Geometry for a function-based index. (See the sample code at www.spatialdbadvisor.com for an example and an approach to correct this).

XML DB

- ◆ In this section I will show how to:
 - Load an Victorian Incremental Update Format (IUF) XML file from disk;
 - Query elements of it using Xpath expressions and SQL;
 - Construct a view through which its coordinate data can be represented via Sdo_Geometry, indexed and visualised with MapViewer;

Spatial XML...

- ◆ Overseas, quite a bit of data is being interchanged based on GML (eg Ordnance Survey).
- ◆ Here in Australia, we don't see much GML being exchanged, but in Victoria, both updates to, and full-exports of, the Victorian Property Map data is exchanged frequently.
- ◆ So, I thought we could look at how to use it, natively, inside Oracle.

Oracle's XML DB

◆ From the Oracle web-site:

(<http://www.oracle.com/technology/tech/xml/xmlldb/index.html>)

- *Oracle XML DB is a feature of the Oracle Database [Including XE].*
- *It provides a high-performance, native XML storage and retrieval technology.*
- *It fully absorbs the W3C XML data model into the Oracle Database, and provides new standard access methods for navigating and querying XML.*
- *With Oracle XML DB, you get all the advantages of relational database technology plus the advantages of XML.*

IUF and XML DB

- ◆ The Victorian IUF data (thanks to Department Sustainability and Environment, DSE) comes in two forms:
 - Actual data (we will only look at this one)
 - Statistical information.
- ◆ The IUF XML data are defined by Data Type Definitions (DTDs) and not via XML Schemas.
 - With XML Schema can create an mapping between the XML data and normal Oracle tables as XML Schema's contain attribute data type information.
 - DTDs do not contain attribute data types so can only be used for validating XML data.

IUF_Full DTD Sample ...

```
<!ELEMENT IUF (DESCRIPTION+, INSERT*, REPLACE*, DELETE*)>
...
<!ELEMENT INSERT (POLYGON*, LINE*, POINT*, ASPATIAL*)>
<!ELEMENT POLYGON (POLYGON_NAME*)>
...
<!ELEMENT HEADER (#PCDATA)>
...
<!ELEMENT XY (#PCDATA)>
...
<!ELEMENT ATT (#PCDATA)>
...
<!ELEMENT POINT (POINT_NAME*)>
<!ELEMENT POINT_NAME (POINT_ITEM*)>
<!ELEMENT POINT_ITEM (HEADER+, XY*, CONNECT*, ATT*)>
<!ELEMENT ASPATIAL (ASPATIAL_NAME*)>
...
<!ELEMENT REPLACE (POLYGON*, LINE*, POINT*, ASPATIAL*)>
<!ELEMENT DELETE (POLYGON*, LINE*, POINT*, ASPATIAL*)>
```

IUF_Full – Header ...

Note Change to
DTD name

```
<?xml version="1.0"?>
  <!DOCTYPE IUF SYSTEM "/public/dtd/iuf_full.dtd">
  <IUF>
    <DESCRIPTION>
      <AUTHORITY>Land Victoria</AUTHORITY>
      <DATASET>Property Maintenance</DATASET>
      <EXPORT SCHEMA="METRO" LGA="YARRA" FROM DATE="18-MAR-
2007" TO DATE="01-APR-2007" REFERENCE_SYSTEM_IDENTIFIER="LL-
GDA94" DATE="01-APR-2007" BY="LogicaCMG"/>
      <SUPPLY_VERSION>3.76.0</SUPPLY_VERSION>

      <PREVIOUS_SUPPLY_VERSION>3.75.0</PREVIOUS_SUPPLY_VERSION>
      <SUPPLY_TYPE>INCREMENTAL</SUPPLY_TYPE>
      <RESTRICTED_SUPPLY_DETAIL></RESTRICTED_SUPPLY_DETAIL>
      <PUBLIC_SUPPLY_DETAIL></PUBLIC_SUPPLY_DETAIL>
      <MODEL_VERSION>1</MODEL_VERSION>
    </DESCRIPTION>
```

IUF_Full – Sample ...

```
-- Lots of elements deleted.....
<INSERT>
  <POINT>
    <POINT_NAME>
      <!-- INSERT for POINT table CENTROID commencing -->
      <POINT_ITEM>
        <HEADER TABLE="CENTROID" PFI="209095553"
CREATED="20070330134038" UFI="283445465"></HEADER>
        <XY>144.98459988,-37.80611174</XY>
        <ATT>FEATURE_TYPE="5701"</ATT>
        <ATT>JOB_CREATED="2671187"</ATT>
        </POINT_ITEM>
        .... More CENTROID points deleted...
      <!-- INSERT for POINT table ADDRESS_LOCATION commencing -->
      <POINT_ITEM>
        <HEADER TABLE="ADDRESS_LOCATION" PFI="209056655"
CREATED="20070321145655" UFI="283346821"></HEADER>
        <XY>144.99570254,-37.8208677</XY>
        <ATT>FEATURE_TYPE="27731"</ATT>
        <ATT>PFI_CREATED="20070321133902"</ATT>
        <ATT>JOB_CREATED="2670459"</ATT>
        </POINT_ITEM>
        ....
```

Loading data...

1) Create MYXML User.

```
CREATE USER MYXML  
IDENTIFIED BY MYXML;
```

2) Create a DIRECTORY within Oracle to access XML and DTD files from disk.

```
CREATE OR REPLACE DIRECTORY  
XMLDIR AS 'e:\oraxe\xml\';  
GRANT READ, WRITE ON DIRECTORY  
XMLDIR TO PUBLIC;
```

Loading data ...

3) Load DTD from XMLDIR ...

Declare

```
res Boolean;
```

Begin

```
res := dbms_xdb.createResource(  
    '/public/dtd/iuf_full.dtd',  
    bfilename('XMLDIR','iuf_full.dtd'));
```

If res **Then**

```
    dbms_output.put_line('DTD Loaded');
```

Else

```
    dbms_output.put_line('DTD failed to load.');
```

End If;

COMMIT;

End;

/



Note Change to
DTD name

Check Loading of DTD

4) Query the Oracle catalog to find out about our DTD ...

```
SELECT PATH,  
        extract (LINK,  
                 '/LINK/Name/text()').getstringval(),  
        extract (LINK,  
                 '/LINK/ParentName/text()').getstringval(),  
        extract (LINK,  
                 '/LINK/ChildName/text()').getstringval(),  
        extract (RES,  
                 '/Resource/DisplayName/text()').getstringval()  
FROM PATH_VIEW  
WHERE PATH = '/public/dtd/iuf_full.dtd';
```

Load actual IUF XML ...

5) Now we can load the IUF XML data into a table...

Prompt Create table to hold the Incremental Yarra data
YARRA_METRO_incremental_data_01-APR-2007.xml

```
DROP    TABLE IncrementalUpdate;
CREATE TABLE IncrementalUpdate
  OF XMLTYPE;
INSERT INTO IncrementalUpdate
VALUES (
  XMLType (
    bfilename (
      'XMLDIR',
      'YARRA_METRO_incremental_data_01-APR-2007.xml'),
    nls_charset_id('AL32UTF8'))
  );
COMMIT;
```

IUF Metadata Queries ...

- ◆ Simple query returning all sub-elements of the DESCRIPTION element ...

```
SELECT extract(object_value, '/IUF/DESCRIPTION')  
FROM IncrementalUpdate;
```

Query full DESCRIPTION

- ◆ Use TABLE function with XMLSequence/extract to extract the DESCRIPTION element and then “unpack” it to extract contents (**extractValue**) of sub-elements and attributes (@).

Prompt Get each and every value of the tags in the description ...

```
SELECT extractValue(value(d), '/DESCRIPTION/AUTHORITY')           As Authority,
extractValue(value(d), '/DESCRIPTION/DATASET')                   As Dataset,
extractValue(value(d), '/DESCRIPTION/EXPORT/@SCHEMA')           As Schema,
extractValue(value(d), '/DESCRIPTION/EXPORT/@LGA')              As LGA,
extractValue(value(d), '/DESCRIPTION/EXPORT/@FROM_DATE')       As From_Date,
extractValue(value(d), '/DESCRIPTION/EXPORT/@TO_DATE')         As To_Date,
extractValue(value(d), '/DESCRIPTION/EXPORT/@REFERENCE_SYSTEM_IDENTIFIER')
    As Reference_System_Identifier,
extractValue(value(d), '/DESCRIPTION/EXPORT/@DATE')             As IUF_Date,
extractValue(value(d), '/DESCRIPTION/EXPORT/@BY')               As By_Supplier,
extractValue(value(d), '/DESCRIPTION/SUPPLY_VERSION')           As Supply_Version,
extractValue(value(d), '/DESCRIPTION/PREVIOUS_SUPPLY_VERSION')
    As Previous_Supply_Version,
extractValue(value(d), '/DESCRIPTION/SUPPLY_TYPE')              As Supply_Type,
extractValue(value(d), '/DESCRIPTION/MODEL_VERSION')            As Model_Version
FROM IncrementalUpdate iu,
TABLE (
    XMLSequence (
        extract(iu.OBJECT_VALUE,
            '/IUF/DESCRIPTION')
        )
    ) d;
```

Query Inserted Polygon Centroids

Prompt Get coordinates of inserted Centroid points ..

```
SELECT      substr(extractValue(value(c), '/POINT_ITEM/HEADER/@TABLE'), 1, 32)
              As Header_Type,
to_number(extractValue(value(c), '/POINT_ITEM/HEADER/@PFI')) As PFI,
to_date(extractValue(value(c), '/POINT_ITEM/HEADER/@CREATED'),
        'YYYYMMDDHH24MISS') As Created,
to_number(extractValue(value(c), '/POINT_ITEM/HEADER/@UFI')) As UFI,
substr(extractValue(value(c), '/POINT_ITEM/XY'), 1, 25) As XY,
to_number(regex_substr(extractValue(value(c),
        '/POINT_ITEM/ATT[1]'), '[0-9]+')) As FeatureType,
to_number(regex_substr(extractValue(value(c),
        '/POINT_ITEM/ATT[2]'), '[0-9]+')) As Job_Created

FROM IncrementalUpdate iu,
      TABLE (
        XMLSequence (
          Extract (iu.OBJECT_VALUE,
            '/IU/INSERT/POINT/POINT_NAME/POINT_ITEM/HEADER[@TABLE="CENTROID"]/..'
          )
        )
      ) c
WHERE ExistsNode ( iu.object_value,
        '/IU/INSERT/POINT/POINT_NAME/POINT_ITEM/HEADER[@TABLE="CENTROID"]'
      ) = 1;
```

ExistsNode
+ Xpath
finds nodes
with
inserted
Centroids

Result ...

HEADER_TYPE	PFI	CREATED	UFI	XY	FEATURETYPE	JOB_CREATED
CENTROID	209039709	19/MAR/07	283294788	145.00039493,-37.83101651	27701	2.0070E+13
CENTROID	209039711	19/MAR/07	283294793	145.00019272,-37.83219362	27701	2.0070E+13
CENTROID	209039713	19/MAR/07	283294798	144.9976457,-37.82469758	27701	2.0070E+13
CENTROID	209056648	21/MAR/07	283346800	144.99569883,-37.82090245	5701	2670459
CENTROID	209056651	21/MAR/07	283346809	144.99570254,-37.8208677	5701	2670459
...						

Create Table to hold Inserted Centroids ...

```
CREATE TABLE IUF_CENTROID (  
  Table_Name      varchar2(32),  
  PFI             integer,  
  Created         Date,  
  UFI            Integer,  
  FeatureType     Number(38),  
  Job_Created     Number(38),  
  geom            SDO_Geometry  
);
```

◆ Note: An XML Schema would help automate this.

Insert Polygon Centroids

```
INSERT INTO IUF_Centroid (
  Table_Name,PFI, Created, UFI, FeatureType, Job_Created, Geom )
SELECT      substr(extractValue(value(c), '/POINT_ITEM/HEADER/@TABLE'),1,32)
              As Header_Type,
  to_number(extractValue(value(c), '/POINT_ITEM/HEADER/@PFI')) As PFI,
  to_date(extractValue(value(c), '/POINT_ITEM/HEADER/@CREATED'),
              'YYYYMMDDHH24MISS') As Created,
  to_number(extractValue(value(c), '/POINT_ITEM/HEADER/@UFI')) As UFI,
  to_number(regex_substr(extractValue(value(c),
              '/POINT_ITEM/ATT[1]'),'[0-9]+')) As FeatureType,
  to_number(regex_substr(extractValue(value(c),
              '/POINT_ITEM/ATT[2]'),'[0-9]+')) As Job_Created
  SDO_Geometry(2001,8311,SDO_POINT_TYPE(
    regex_substr(
      extractValue(value(c), '/POINT_ITEM/XY'),'[-]?[0-9][0-9.]*',1,1),
    regex_substr(
      extractValue(value(c), '/POINT_ITEM/XY'),'[-]?[0-9][0-9.]*',1,2),
    NULL),NULL,NULL) As Geom
FROM IncrementalUpdate iu,
TABLE( XMLSequence(
  Extract(iu.OBJECT_VALUE,
    '/IUF/INSERT/POINT/POINT_NAME/POINT_ITEM/HEADER[@TABLE="CENTROID"]/..'
  ))) c
WHERE ExistsNode ( iu.object_value,
  '/IUF/INSERT/POINT/POINT_NAME/POINT_ITEM/HEADER[@TABLE="CENTROID"] '
) = 1;
```

Create Metadata: IUF_CENTROID

Prompt Let's not do this manually...

```
DECLARE
```

```
  v_geom mdsys.sdo_geometry;
```

```
BEGIN
```

```
  SELECT SDO_AGGR_MBR (geom)
```

```
    INTO v_geom
```

```
  FROM IUF_CENTROID;
```

```
INSERT INTO USER_SDO_GEOM_METADATA (
```

```
  SRID, TABLE_NAME, COLUMN_NAME, DIMINFO
```

```
) VALUES (
```

```
  v_geom.sdo_srid, 'IUF_CENTROID', 'GEOM',
```

```
  MDSYS.SDO_DIM_ARRAY (
```

```
  MDSYS.SDO_DIM_ELEMENT('X', v_geom.SDO_ORDINATES(1), v_geom.SDO_ORDINATES(4), 0.05),
```

```
  MDSYS.SDO_DIM_ELEMENT('Y', v_geom.SDO_ORDINATES(2), v_geom.SDO_ORDINATES(5), 0.05))
```

```
);
```

```
END;
```

```
/
```

Create Index and Query...

```
◆ DROP INDEX IUF_CENTROID_GEOM;
CREATE INDEX IUF_CENTROID_GEOM
ON IUF_CENTROID(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('sdo_indx_dims=2, layer_gtype=point');
```

Prompt Find nearest 2 points to a 5% sample ..

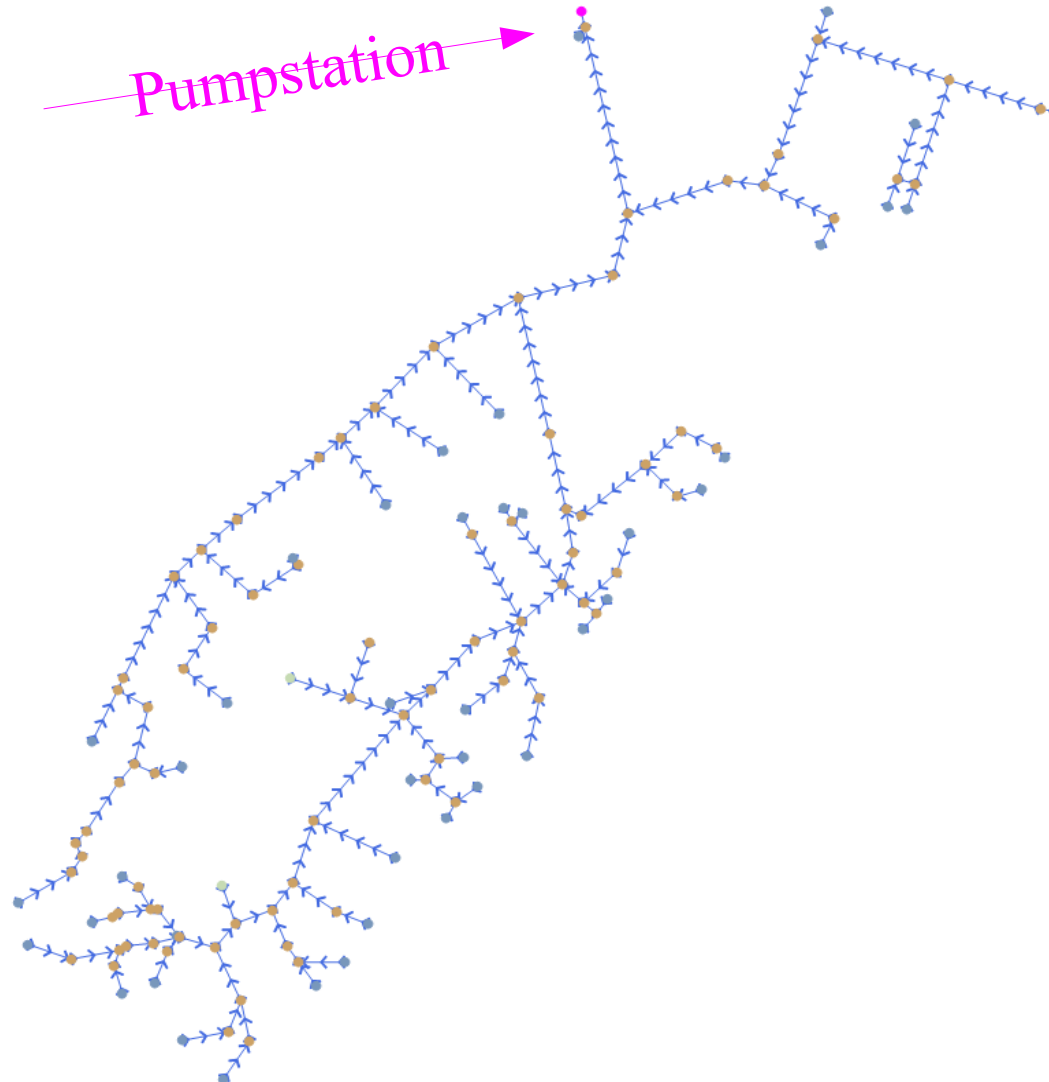
```
SELECT /*+ ORDERED*/
iufc.pfi,
SDO_NN_DISTANCE(1) As distance
FROM IUF_CENTROID SAMPLE (5) iufc ,
IUF_CENTROID iufc2
WHERE SDO_NN(iufc2.GEOM,
iufc.geom,
'sdo_num_res=2, unit=Meter',
1) = 'TRUE';
```

Hierarchical Queries

- ◆ Before discussing the use of SQL99 OLAP function I will firstly outline Hierarchical Queries (we saw a simple “**CONNECT BY**” query when generating test points)
- ◆ I will use part of a customer's pipe network (suitable changed to hide details) to show how basic, Oracle + Locator functionality can be used to conduct “network traces”.

(Thanks to Mid Coast Water, NSW for making the data available.)

A Pipe Network with Nodes



Customer's data...

- ◆ This customer's data has coded the pipes (My_Pipe) and nodes (My_Node) connections via attributes:
 - Node_Id (My_Node)
 - ◆ Of nodes in the network representing valves, manholes etc.
 - Start_Node/End_Node (My_Pipe)
 - ◆ For each pipe showing which nodes are at start and end.
 - Following query uses these attributes to conduct a hierarchical query that traverses every pipe in the network.

A simple network trace ...

Prompt Conduct a simple trace back up the network from our pumpstation...

```
SELECT wp.pipe_id,  
        wp.start_node,  
        wp.end_node,  
        wp.pipe_diameter,  
        wp.pipe_material,  
        SDO_GEOM.SDO_LENGTH(wp.geometry,0.01) as pipe_length,  
        LEVEL  
FROM my_pipe wp  
START WITH OID =  
        (SELECT sgj.oid  
         FROM my_node snj  
         INNER JOIN  
         my_pipe sgj  
         ON ( sgj.end_node = snj.node_id)  
        WHERE snj.node_type = 'PUMPSTATION'  
        )  
CONNECT BY NOCYCLE PRIOR start_node = end_node;
```

Result ...



PIPE_ID	START_NODE	END_NODE	PIPE_DIAMETER	PIPE_MATERIAL	PIPE_LENGTH	LEVEL
18918	18024	12210	150	PVC-U	11.6317825	1
13715	18028	18024	150	PVC-U	6.89768229	2
18917	11204	18024	150	PVC-U	133.368761	2
13445	11202	11204	150	PVC-U	71.2474506	3
13437	11201	11202	150	PVC-U	24.7904271	4
13438	11200	11201	150	PVC-U	23.6856216	5
.....						
22642	22844	22843	150	PVC-U	27.0856449	10
22641	21548	22844	150	PVC-U	8.41058057	11

127 rows selected.

But ...

◆ But what if we didn't have the node attributes?

- Note: Our network's pipe SDO_Geometries all flow “downhill” in the direction of the pumpstation.
- We can still do this but it does require a trivial piece of code (Programming? Me? Heaven forbid!)

Network PL/SQL Package

```
CREATE OR REPLACE PACKAGE Network
AS
    FUNCTION get_point (
        p_geom          SDO_GEOMETRY,
        p_point_number NUMBER DEFAULT 1
    ) RETURN SDO_GEOMETRY DETERMINISTIC;
    FUNCTION get_Start_Point (
        p_geom          SDO_GEOMETRY
    ) RETURN SDO_GEOMETRY DETERMINISTIC;
    FUNCTION get_End_Point (
        p_geom          SDO_GEOMETRY
    ) RETURN SDO_GEOMETRY DETERMINISTIC;
    FUNCTION get_point_text (
        p_geom          SDO_GEOMETRY,
        p_point_number NUMBER DEFAULT 1
    ) RETURN VARCHAR2 DETERMINISTIC;
    FUNCTION get_Start_point_text (
        p_geom          SDO_GEOMETRY
    ) RETURN VARCHAR2 DETERMINISTIC;
    FUNCTION get_End_point_text (
        p_geom          SDO_GEOMETRY
    ) RETURN VARCHAR2 DETERMINISTIC;
END Network;
```

*Original Function
(modified) from Pro Oracle
Spatial*

**_text functions
return packed string
of form "X@Y@Z"*

A simple network trace - revisited

Prompt Conduct a simple trace back up the network ...

```
SELECT wp.pipe_id,  
        wp.start_node,  
        wp.end_node,  
        wp.pipe_diameter,  
        wp.pipe_material,  
        SDO_GEOM.SDO_LENGTH(wp.geometry,0.01) as pipe_length,  
        LEVEL  
  
        FROM my_pipe wp  
START WITH OID =  
        (SELECT sgj.oid  
         FROM my_node snj  
         INNER JOIN  
         my_pipe sgj  
         ON (sgj.end_node = snj.node_id)  
         WHERE snj.node_type = 'PUMPSTATION'  
        )  
CONNECT BY NOCYCLE PRIOR  
        Network.Get_Start_Point_Text(wp.geometry) =  
        Network.Get_End_Point_Text(wp.geometry);
```

Too slow, big network?

- ◆ Let's apply what we learned earlier about function-based indexes ...

Prompt If the network gets big...

```
CREATE INDEX My_Pipe_Start_Point_Index  
            ON My_Pipe  
            (Network.Get_Start_Point_Text(geometry));
```

- ◆ Same result!

SQL99 OLAP Functions

- ◆ So, to conclude, Oracle has implemented some really cool SQL99 based “analytic” functions.
 - Eg GROUP BY CUBE(), ROLLUP() etc
- ◆ But there is included in the SQL99 standard a set of “*windowing*” functions that allow us to calculate “*moving*” averages etc and not just an average at the end of each grouping set!
- ◆ The following query shows what can be done with our network...

Computing missing values, etc.

Prompt First let's compute any missing values ...

```
SELECT rownum as row_num,
        LEVEL as row_level,
        wp.end_invelev,
        wp.start_invelev,
        wp.pipe_gradient,
        SDO_GEOM.SDO_LENGTH(wp.geometry,0.01) as pipe_length,
        CASE WHEN wp.end_invelev is not null
            AND wp.start_invelev is not null
            AND wp.pipe_length is not null
            THEN CASE WHEN ( wp.start_invelev - wp.end_invelev ) <> 0
                THEN ROUND(wp.pipe_length / ( wp.start_invelev -
wp.end_invelev ),3)
                ELSE 0
            END
        ELSE
            NULL
        END As comp_gradient
FROM my_pipe wp
START WITH OID = (SELECT wp.oid
                    FROM my_node snj
                    INNER JOIN
                    my_pipe wp
                    ON ( wp.end_node = snj.node_id)
                    WHERE snj.node_type = 'PUMPSTATION')
CONNECT BY NOCYCLE PRIOR
        Network.Get_Start_Point_Text(wp.geometry) =
        Network.Get_End_Point_Text(wp.geometry);
```

Now use “windowing” functions to compute rolling total length (SUM) and average gradient (AVG)...

Prompt Use row_level to do some “pretty printing”...

```
SELECT row_level,  
LPAD('_', row_level - 1, '_') ||  
  ' Invrt(End,Start)->(' ||  
    DECODE(h.end_invelev, NULL, 'NULL',h.end_invelev) ||  
  ', '  
    DECODE(h.start_invelev, NULL, 'NULL',h.start_invelev) ||  
  ') Grdnt(DB,Query)->(' ||  
    DECODE(h.pipe_gradient, NULL, 'NULL',h.pipe_gradient) ||  
  ', '  
    DECODE(h.comp_gradient, NULL, 'NULL',h.pipe_gradient) ||  
  ') MvgTotalLen(' ||  
    ROUND( SUM(h.pipe_length)  
           OVER (ORDER BY row_num ROWS UNBOUNDED PRECEDING)  
           ,3) ||  
  ') MvAvgGrdnt(' ||  
    ROUND( AVG(h.comp_gradient)  
           OVER (ORDER BY row_num ROWS UNBOUNDED PRECEDING),3) ||  
  ')'  
As Information  
FROM (  
  SELECT ... statement from previous slide  
) h;
```

Result...

ROW	LEVEL	INFORMATION
1		Invrt (End, Start) -> (0, -.685) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (401.296) MvAvgGrdnt (-16.981)
2		Invrt (End, Start) -> (-.685, 0) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (639.267) MvAvgGrdnt (-3.456)
2		Invrt (End, Start) -> (-.685, -3.1) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (5240.489) MvAvgGrdnt (-20.712)
3		Invrt (End, Start) -> (-3.1, -2.07) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (7698.526) MvAvgGrdnt (1.759)
4		Invrt (End, Start) -> (-1.88, -2.07) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (8553.796) MvAvgGrdnt (-24.688)
5		Invrt (End, Start) -> (-1.88, -1.64) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (9370.945) MvAvgGrdnt (-4.125)
6		Invrt (End, Start) -> (-1.64, -1.04) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (12291.457) MvAvgGrdnt (16.62)
7		Invrt (End, Start) -> (-1.04, 0) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (13020.131) MvAvgGrdnt (17.081)
7		Invrt (End, Start) -> (-1.04, -.26) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (16243.283) MvAvgGrdnt (28.491)
.....		
10		Invrt (End, Start) -> (0, 0) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (141705.275) MvAvgGrdnt (58.46)
11		Invrt (End, Start) -> (0, 0) Grdnt (DB, Query) -> (0, 0) MvgTotalLen (141995.437) MvAvgGrdnt (57.999)

127 rows selected.

Summary

- ◆ Spatial data is not something held in corporate databases in isolation from the rest of an enterprise's data.
- ◆ This means that when we work with spatial data within an information system's framework we should examine the complete technology stack in which it resides for functionality that could be of use in “extending” the reach of our spatial data beyond that of simple mapping and editing.
- ◆ This talk has shown how some of the wider technology within Oracle database can be used with spatial data.
- ◆ I hope it was a useful “taster”, enough to whet your appetite and to dig deeper into the Oracle technology stack.



Thank you.

Simon Greener.

www.spatialdbadvisor.com