Tips and Tricks – PL/SQL, Locator & Application Express (Apex)

Simon Greener,
The SpatialDB Advisor
&
Jamie Keene,
Senior Spatial Analyst
Open Spatial Pty Ltd

Genesis ...

- The background to this joint presentation was work done when both of us worked at Forestry Tasmania Pty Ltd.
  - The work being presented includes two elements
    - A set of PL/SQL packages that were written (8i) to augment Spatial/Locator functionality;
      - Tip on using pipelined functions
    - A series of applications built using WebDB, HtmlDB and APEX.
      - Demonstration of Apex, PL/SQL and MapViewer

Genesis – PL/SQL packages

- Oracle's excellent PL/SQL allowed us to create additional functionality for deployment within an Oracle database.
  - Used for spatial referential integrity checks;
  - Automation of sdo_geometry construction (eg trigger to construct point from bearing/distance from known point stored in northing/easting columns);
  - Use of spatial functionality in Oracle's run queue;
  - Augmenting the standard Locator/Spatial PL/SQL packages.
- These packages have been completely re-written since Simon Greener left Forestry.
  - He will present a few slides on these packages highlighting one particular performance tip.

Genesis - WebDB

- We first started integrating Spatial and Oracle's “out of the database” with WebDB.
    - Built database reports on spatial database activities (eg production of PDF based maps), data access etc
        - Used for billing and budgetry reasons
    - Also built point “editing” applications that allowed foresters to create, move and delete simple point based data via 2 attribute columns (northing and easting)
        - Spatial prototype that extended point editing to include graphic drawing in an SVG plugin (with full synchronisation) was demonstrated but not deployed (MapViewer did not exist at the time).

Genesis - HtmlDB/APEX

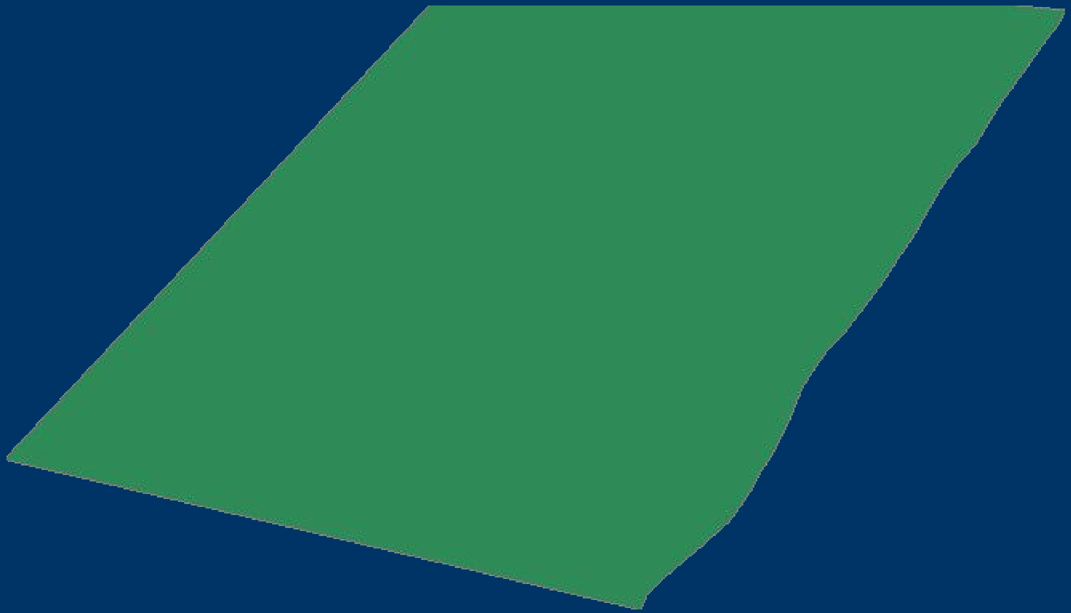
- With release of 10gR1 all our WebDB applications were ported to HtmlDB (now Application Express - Apex)
  - Experience gained with WebDB, coupled with greater capability and flexibility of HtmlDB/Apex, has allowed for an explosion in spatial integration cf Google Maps via “mashups” etc.
- Jamie Keene will present some slides on integrating Apex with MapViewer and Simon's PL/SQL packages.

PL/SQL COGO Package

- The packages that will be used today are the COGO and GEOM packages.
- The GEOM package contains general functions such as a “point in polygon” function (guarantees the point falls inside its polygon), and a “vectorisation” function that is used in this presentation.
- The COGO package was first constructed for use inside a critical application “inventory” database at Forestry.

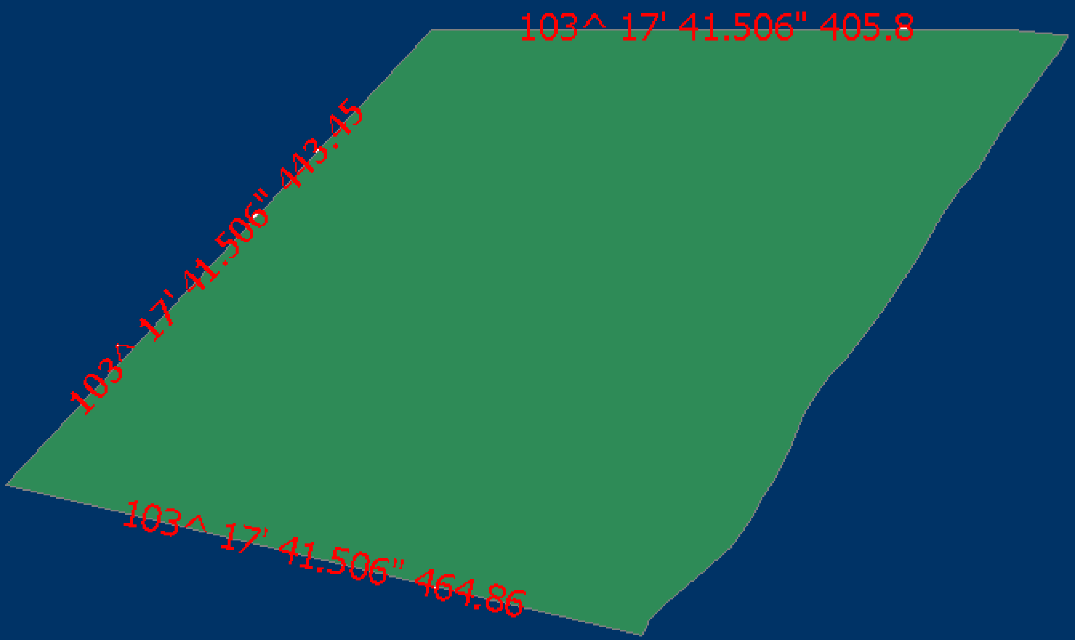
Extracting bearings and distances

- Let's start with something simple
  - Here is a polygon



Metes and Bounds..

- But we have a business requirement to annotate it by bearings and distances dynamically generated from the actual vectors that make up the boundary of the polygon.



How?

- A little bit of PL/SQL in some types & packages...

```
CREATE OR REPLACE PACKAGE COGO
AS

    FUNCTION PI        RETURN NUMBER;


    FUNCTION Bearing( dE1 in number,
                      dN1 in number,
                      dE2 in number,
                      dN2 in number)
    RETURN NUMBER DETERMINISTIC;

    FUNCTION Distance( dE1 in number,
                       dN1 in number,
                       dE2 in number,
                       dN2 in number)
    RETURN NUMBER DETERMINISTIC;

...

END COGO;
```

```
CREATE OR REPLACE TYPE Coord2DType AS
OBJECT (
   x NUMBER,
   y NUMBER );

CREATE OR REPLACE TYPE Vector2DType AS
OBJECT (
   startCoord  Coord2DType,
     endCoord  Coord2DType );

CREATE OR REPLACE TYPE Vector2DSetType
   AS TABLE OF Vector2DType;

CREATE OR REPLACE PACKAGE GEOM
AS

    FUNCTION GetVector2D (
    p_geometry in mdsys.sdo_geometry)
    RETURN CODESYS.Vector2DSetType
      DETERMINISTIC;

END GEOM;
```

A view ...

```
CREATE OR REPLACE VIEW apex_demo
AS
SELECT rownum AS gid,
       codesys.Cogo.DD2DMS(
           codesys.Cogo.Bearing(startx,starty,endx,endy)
                                *
                                (180/codesys.Cogo.PI) )
              AS bearing,
       ROUND(codesys.Cogo.Distance(startx,starty,endx,endy),2)
              AS distance,
       MDSYS.sdo_geometry(2002,NULL,NULL,
                    MDSYS.SDO_ELEM_INFO_ARRAY(1,2,1),
                    MDSYS.SDO_ORDINATE_ARRAY(startx,startY,endX,endY))
           AS geometry
  FROM ( SELECT DISTINCT c.StartCoord.X AS startX,
                         c.StartCoord.Y AS startY,
                         c.EndCoord.X   AS endX,
                         c.EndCoord.Y   AS endY
           FROM ( SELECT geom
                    FROM ProjPoly2D
                   WHERE gid = 5 ) a,
                 TABLE(CAST(codesys.Geom.GetVector2D(a.geom)
                         AS codesys.Vector2DSetType)) c
       );
```

Performance Tip - Pipelining

- Use of Pipelined functions substantially improves performance, reduces memory use and is more scalable.
  - 2 Definitions of Vector2D

```
  Function GetVector2D (
    p_geometry in mdsys.sdo_geometry )
    Return Vector2DSetType Deterministic
...

  Function GetVector2DAsPipelined (
    p_geometry in mdsys.sdo_geometry )
    Return Vector2DSetType Pipelined
```

  - Difference?

Difference: Ordinary

- Non-Pipelined functions require memory...

```
FUNCTION GetVector2D ( p_geometry IN mdsys.sdo_geometry)
  RETURN CODESYS.Vector2DSetType DETERMINISTIC;
    vectors Vector2DSetType := Vector2DSetType();
BEGIN
...
  IF v_vertex = 1 THEN
    vectors.EXTEND;
    v_vector := vectors.LAST;
    vectors(v_vector) := Vector2DType(Coord2DType(-1,1),Coord2DType(-1,1));
    vectors(v_vector).startCoord.x := v_coord.x;
    vectors(v_vector).startCoord.y := v_coord.y;
  ELSE
    vectors(v_vector).endCoord.x := v_coord.x;
    vectors(v_vector).endCoord.y := v_coord.y;
    vectors.EXTEND;
    v_vector := vectors.LAST;
    vectors(v_vector) := Vector2DType(Coord2DType(-1,1),Coord2DType(-1,1));
    vectors(v_vector).startCoord.x := v_coord.x;
    vectors(v_vector).startCoord.y := v_coord.y;
  END IF;
  ...
  RETURN vectors;
END;
```

Define array of vectors

Allocate memory and add object to the set

Must return the filled set

Difference: Pipelined

- Pipelined use internal Oracle inter-process kernel communications which are not dependent on user memory...

```
FUNCTION GetVector2D ( p_geometry IN mdsys.sdo_geometry)
  RETURN CODESYS.Vector2DSetType PIPELINED;
  ...
BEGIN
...
   IF v_vertex = 1 THEN
     v_vector.startCoord.x := v_coord.x;
     v_vector.startCoord.y := v_coord.y;
   ELSE
     v_vector.endCoord.x := v_coord.x;
     v_vector.endCoord.y := v_coord.y;
     PIPE ROW ( v_vector );
     v_vector.startCoord.x := v_coord.x;
     v_vector.startCoord.y := v_coord.y;
     v_vector.endCoord.x   := -1;
     v_vector.endCoord.y   := -1;
   END IF;
   ...
   RETURN;
END;
```

Note keyword

Pushes object into FIFO

Note: no return value

Performance metrics...

- Pipelining is FAST!
  - But don't take my word for it.
  - Let's vectorise some polygon data using the GetVector2D() function and then compare it to GetVector2DAsPipelined()

```
SELECT count(*)
  FROM sp_parcel;

  COUNT(*)
----------
     57453
```

Performance metrics...

```
CREATE TABLE {non_}pipelined_version
AS
SELECT rownum AS gid,
       MDSYS.SDO_GEOMETRY(2002,NULL,NULL,
             MDSYS.SDO_ELEM_INFO_ARRAY(1,2,1),
             MDSYS.SDO_ORDINATE_ARRAY(startx,startY,endX,endY))
           AS geometry
  FROM ( SELECT DISTINCT c.StartCoord.X AS startX,
                         c.StartCoord.Y AS startY,
                         c.EndCoord.X   AS endX,
                         c.EndCoord.Y   AS endY
           FROM ( SELECT geometry
                    FROM SP_PARCEL
                ) a,
                TABLE(CAST(
                      codesys.Geom.GetVector2D{AsPipelined}(a.geometry)
                           AS codesys.Vector2DSetType)) c
       );
```

Numbers...

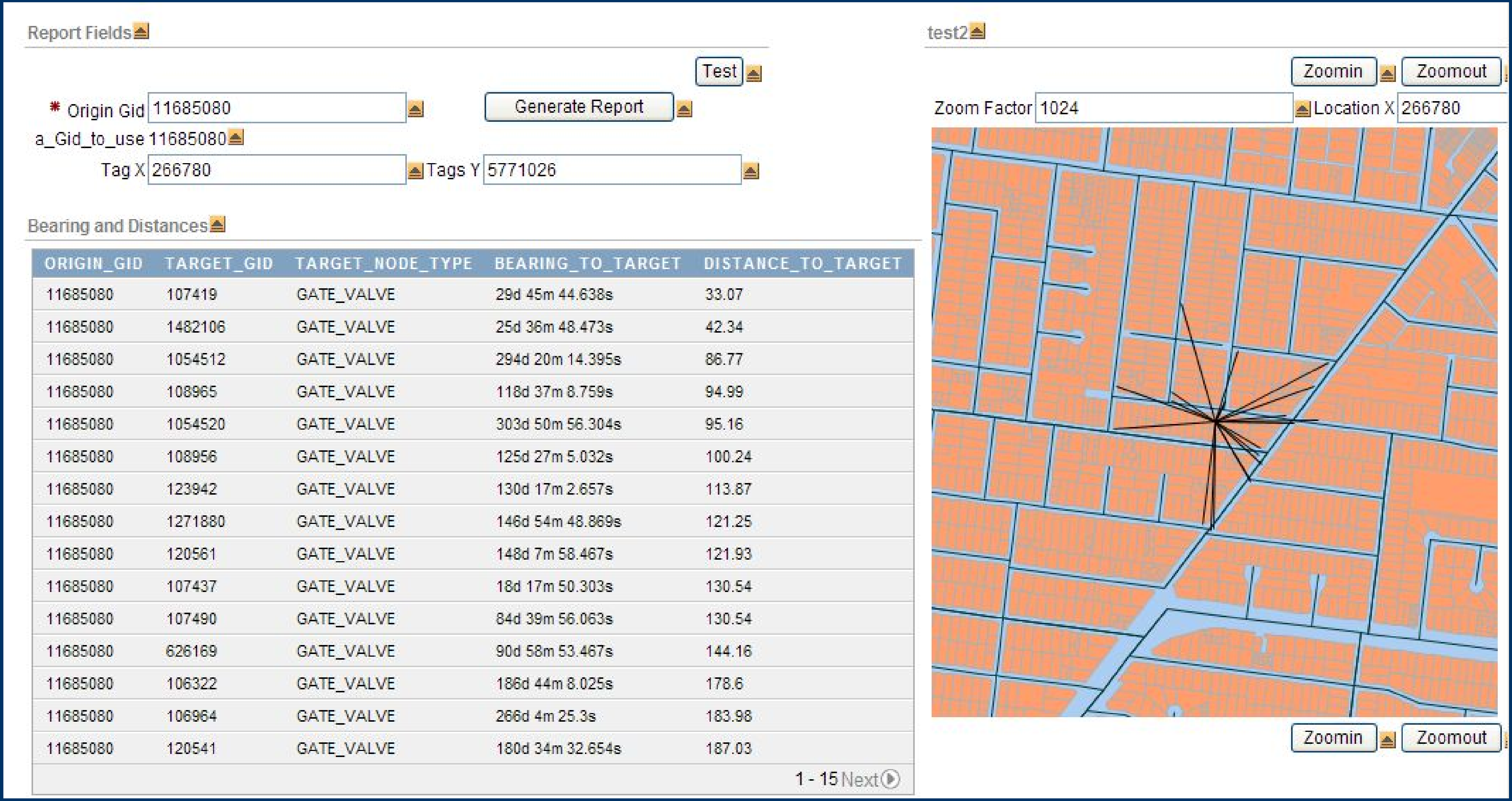
```
SELECT COUNT(*)
  FROM PIPELINED_VERSION;

  COUNT(*)
----------
    763916
```

Function	TimeInSeconds
Vector2DSetType	Elapsed: 00:02:18.13
Vector2DSetTypeAsPipelined	Elapsed: 00:00:47.90

Pipelining improved performance by:

(1 / (48 / 138) * 100 = 287%

Use of functions in Apex



Report Fields

Test

* Origin Gid	11685080
a_Gid_to_use	11685080
Tag X	266780
Tags Y	5771026

Generate Report

Bearing and Distances

ORIGIN_GID	TARGET_GID	TARGET_NODE_TYPE	BEARING_TO_TARGET	DISTANCE_TO_TARGET
11685080	107419	GATE_VALVE	29d 45m 44.638s	33.07
11685080	1482106	GATE_VALVE	25d 36m 48.473s	42.34
11685080	1054512	GATE_VALVE	294d 20m 14.395s	86.77
11685080	108965	GATE_VALVE	118d 37m 8.759s	94.99
11685080	1054520	GATE_VALVE	303d 50m 56.304s	95.16
11685080	108956	GATE_VALVE	125d 27m 5.032s	100.24
11685080	123942	GATE_VALVE	130d 17m 2.657s	113.87
11685080	1271880	GATE_VALVE	146d 54m 48.869s	121.25
11685080	120561	GATE_VALVE	148d 7m 58.467s	121.93
11685080	107437	GATE_VALVE	18d 17m 50.303s	130.54
11685080	107490	GATE_VALVE	84d 39m 56.063s	130.54
11685080	626169	GATE_VALVE	90d 58m 53.467s	144.16
11685080	106322	GATE_VALVE	186d 44m 8.025s	178.6
11685080	106964	GATE_VALVE	266d 4m 25.3s	183.98
11685080	120541	GATE_VALVE	180d 34m 32.654s	187.03

1 - 15 Next

test2

Zoomin Zoomout

Zoom Factor 1024 Location X 266780

Zoomin Zoomout

Summary

- PL/SQL is part of your Oracle Spatial “Swiss Army Knife”
  - Pipelining is fast, scalable and memory friendly.
- Apex is free, fully integrated, fast and powerful
  - Apex + Spatial + MapViewer is a powerful combination.

- Thanks: Mid Coast Water and Barwon Water for use of their data.