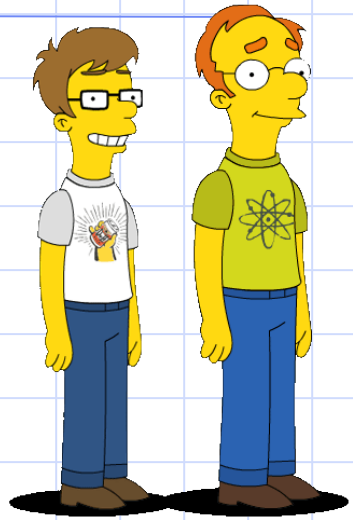
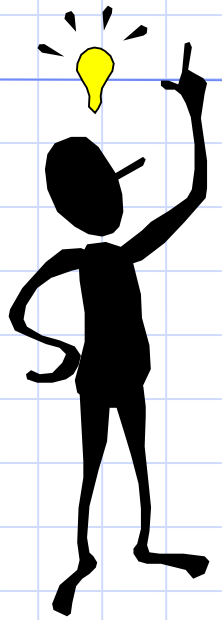
Automated Layer Management - Experiences At Mid Coast Water



Simon Greener,
The SpatialDB Advisor
&
Brendan Soustal,
GIS Manager,
Mid Coast Water

The Business Problem

- *Policy*



Title: **Building Near Water And Sewer Services**

MCW Ref: 64752

Date of Issue: 16 July 2003

Prepared by: Design and Construction Branch

Policy

- Mid Coast Water:
  - "... will not give approach for structures ... built over a sewer rising main or water main ... within distances specified in this policy [where easement does not exist]" (similarly for gravity sewers);
  - "...may approve structures ... built adjacent to a sewer, a sewer rising main or a water main providing precautions are taken with the design of the footings".
- While policy discriminates between pipes with an easement and those without, this presentation assumes all pipes do not have easements.

Policy Application and Scope

- Application
  - “.. applies to construction of all buildings, dwellings, carports, garages, sheds, swimming pools, pergolas, retaining walls and permanent structures ... built over or adjacent to ...”
- Coverage
  - “ ... [Mid Coast Water] owned pipes throughout the Mid Coast Water area [of operations]:”
    - Sewers
    - Sewer rising mains
    - Water mains

Policy Aim

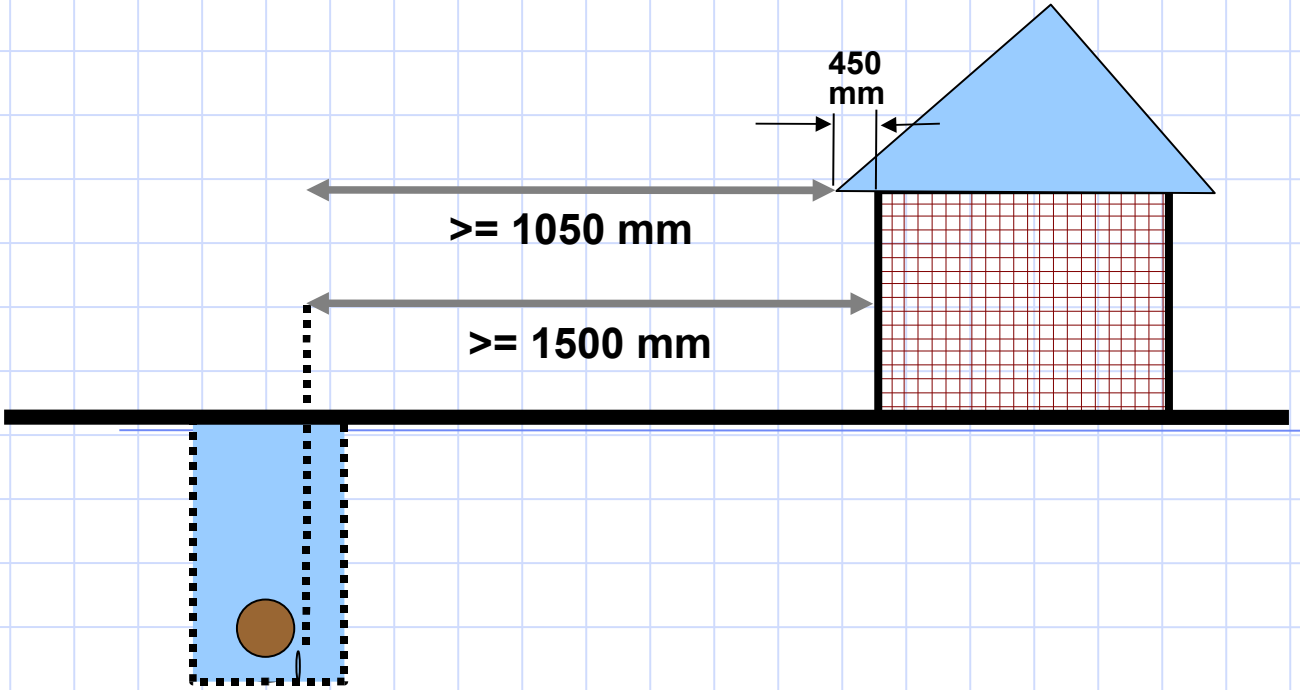
- Policy aimed at:
  - Preventing structure damage to pipes.
  - Preventing damage (eg subsidence) to buildings.
  - Maintaining access to manholes, junctions and inspection shafts;
  - Enable efficient and economical access to pipework for major repairs or replacement;
  - Reduce future maintenance costs to Mid Coast Water;
  - Provide a consistent approach to building over or near underground pipework throughout the area of operations.

Building near Water Main and Sewer Rising Main

- These are pressurised structures.
- Normally located in footpaths, roadways or well away from structures.
- Sometimes located in private property.
- If not in easement then a corridor *at least 3m wide* and centred on pipe is used to determine the area in which a structure cannot be located.

Building Near Gravity Sewer – No Easement

- No external wall closer than 1500mm to side of pipe;
- Standard overhang of 450mm assumed.
- If overhang > 450mm then external wall distance will have to be > 1500mm.

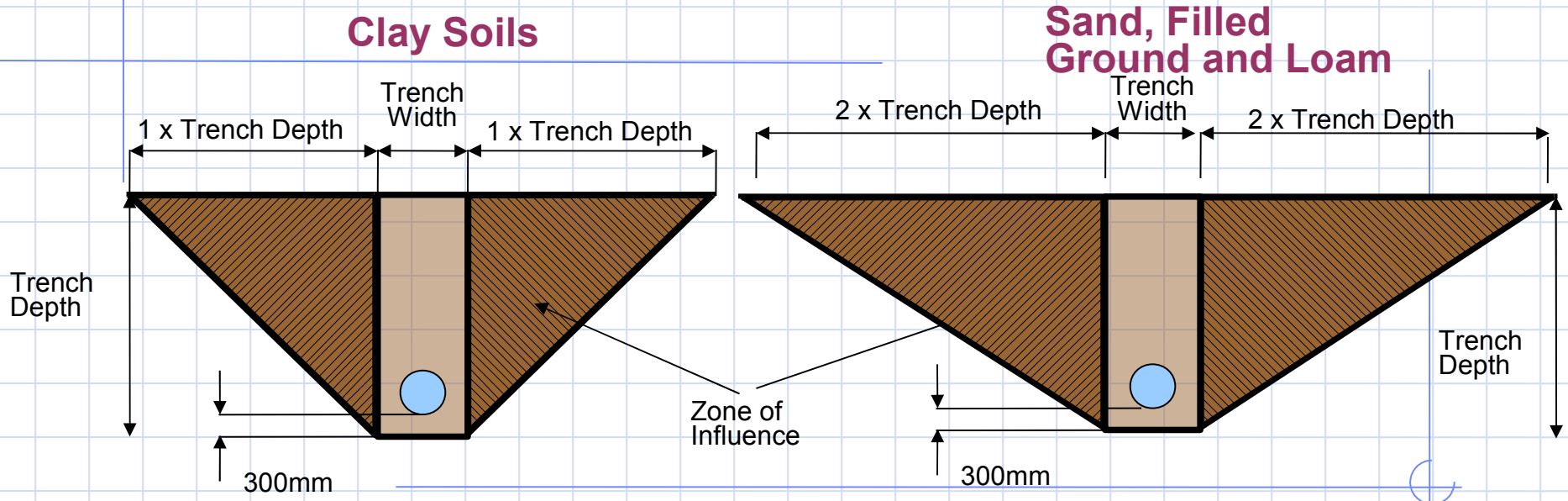


Zone of Influence

- "... is that part of the soils that will be affected by any damage occurring to the pipe or during excavation of a trench".
- Is estimation of the amount of subsidence that may occur.
- Calculation:
  - Pipe Depth is calculated from invert level at both ends of pipe.
  - Depth of Trench containing the pipe is calculated by adding 300mm to Pipe Depth.
  - Width of Trench depends on Pipe Diameter.
    - Pipes up to 225mm diameter will have trench width of 600mm; pipes over 225mm, 1000mm trench width; pipes larger than 1000mm individually assessed.

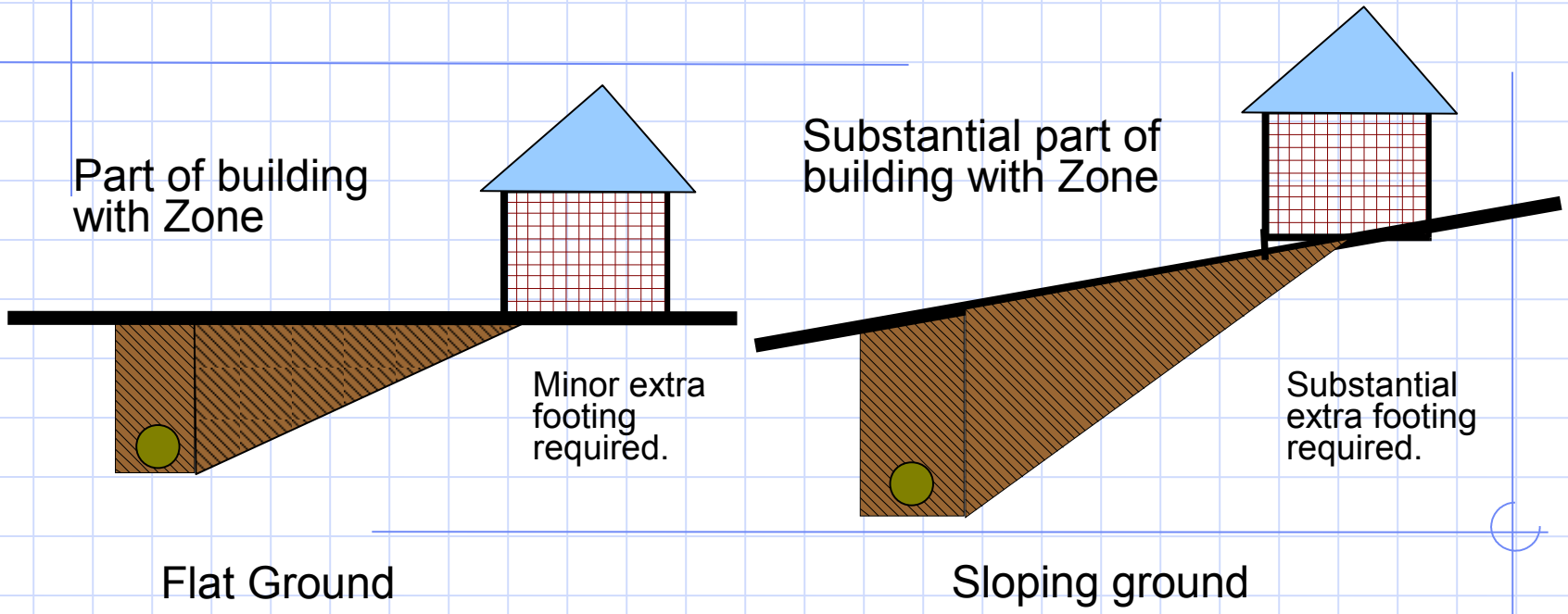
Zone of Influence – Clay Soils

- Zone is calculated as the depth of trench plus half trench width.
- But varies due to soil type.



Zone of Influence – Effect of Topography

- Zone also varies due to site topography (not yet implemented)



Problem

- Small (4) GIS staff busy editing water and sewer networks (core business), easements, sewer basins, land parcels, road areas, buildings, soils, elevation and drainage data etc.
- Also managing technological infrastructure: software, databases, mapping applications etc.
- Much of the existing work is, as expected, manual.
- Limited resources to implement and maintain Yet Another GIS Layer (YAGL).

However

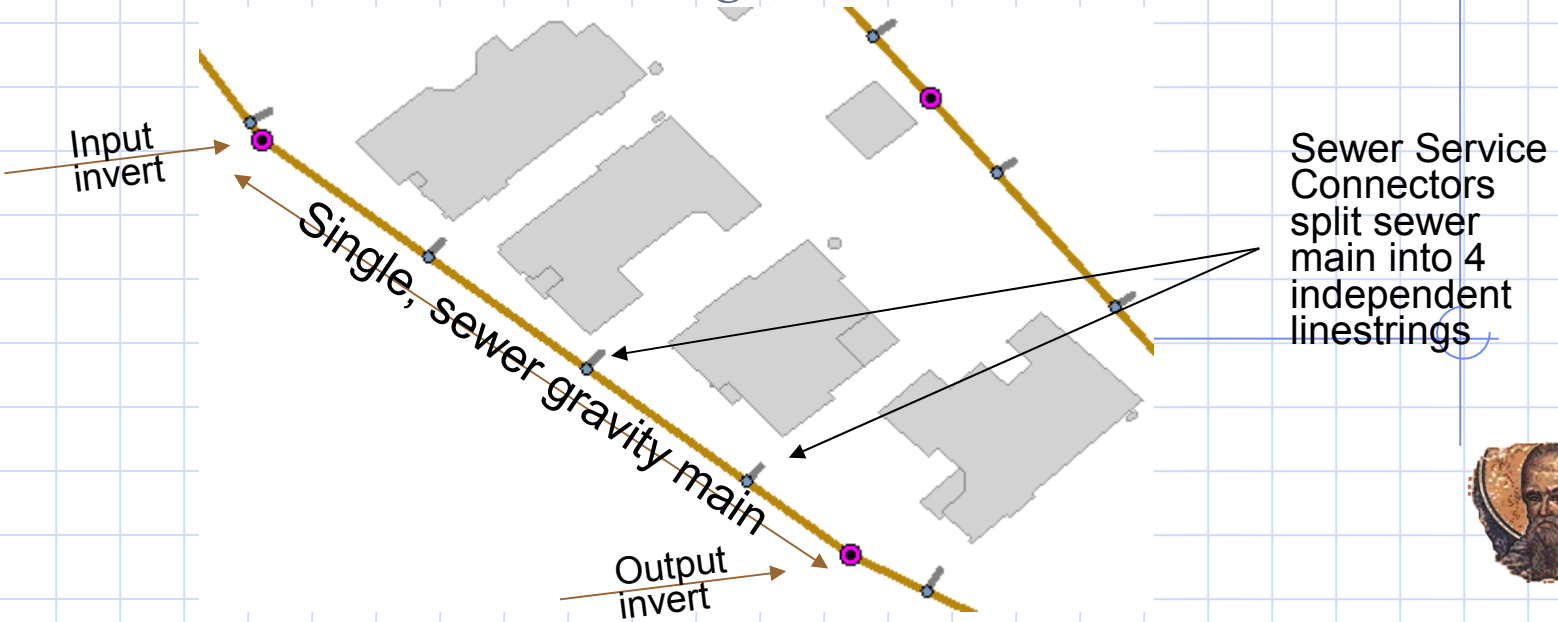
- Sewer and water networks are well attributed (invert levels and diameters)
- Soils and Easement data (polygons) are available.
- Elevation data to be brought online Q4 2008.
- *Zone of Influence* is mathematically well defined.
- So why not make this a fully automated layer and let Oracle Spatial do its stuff (improving Return on Investment)?

Design Goals

- Minimal PL/SQL programming;
- Extract as much value from Oracle Spatial as possible;
- Implement efficient trigger based recalculation system (as computations could impose unacceptable delay on transaction commits).

Implementation: Issue 1

- TopoBase data model
  - In AutoDesk's TopoBase model a business feature, such as a sewer gravity pipe is physically stored as a set of independent linestrings (pipes are split at point where sewer service connector joins)
  - These must be aggregated to a single linestring because invert depths are only recorded at manholes (ie at end of first and last linestrings)



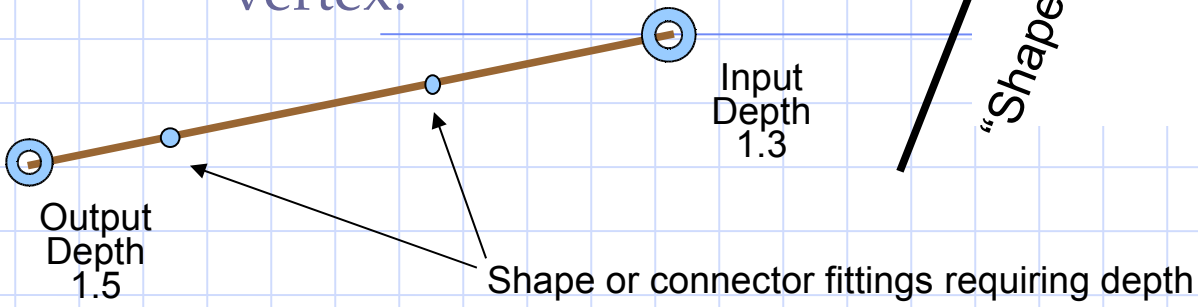
Implementation: Solution 1

- Use SDO_AGGR_CONCAT_LINES aggregate operator.

```
(SELECT /*+ INDEX(l WW_LINE_IX2) */
        SDO_AGGR_CONCAT_LINES(l.geom)
   FROM mcw_sewer.ww_line l
  WHERE l.FID_ATTR = p.FID ),
```

Implementation: Issue 2

- Interpolated Depths
  - Aggregated pipes from Issue 1 need depths assigned to vertices created by service connectors.
  - Also, while pipes normally are straight having two attributed ends, one can get pipes that have multiple "shape" vertices.
  - In both cases, Invert Depth are held only at the ends: need depth at each "shape" vertex.



Implementation: Solution 2

- Use:
  - TABLE function to "explode" line into 2 vertex vectors
  - LRS function, SDO_LRS.CONVERT_TO_LRS_GEOM, to interpolate depths at "shape" vertices (effectively generates a 3D linestring)

SDO_LRS.CONVERT_TO_LRS_GEOM

Format

```
SDO_LRS.CONVERT_TO_LRS_GEOM(
    standard_geom IN SDO_GEOMETRY
    [, start_measure IN NUMBER,
    end_measure IN NUMBER]
    ) RETURN SDO_GEOMETRY;
```

```
FROM mcw_sewer.ww_section p,
     TABLE(mcw_sewer.zoi.GetVector(
             SDO_LRS.CONVERT_TO_LRS_GEOM(
                 (SELECT /*+ INDEX(l WW_LINE_IX2) */
                         SDO_AGGR_CONCAT_LINES(l.geom)
                    FROM mcw_sewer.ww_line l
                   WHERE l.FID_ATTR = p.FID ),
                 p.input_depth,
                 p.output_depth))) v
WHERE p.id_function in (10004 /*gm*/,10007/*rmain*/,10008/*vmain*/)
  AND p.input_depth is not null
  AND p.output_depth is not null
```

Custom Vectorisation function

Interpolation function

Start Measure

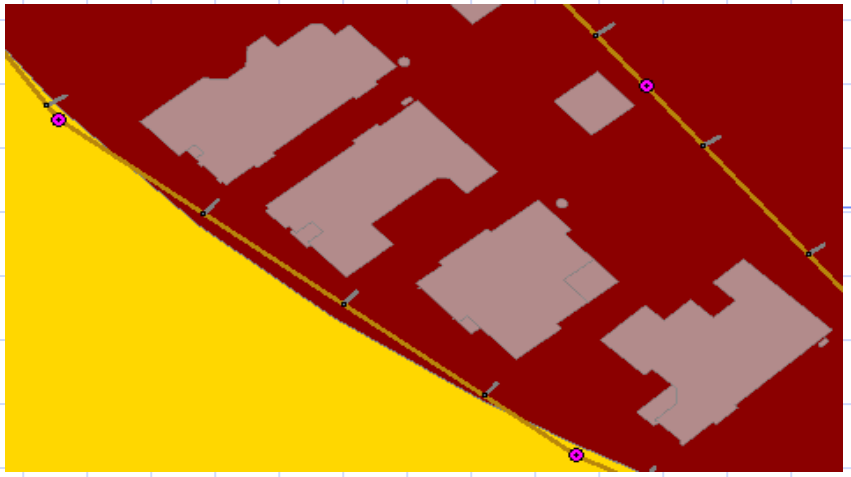
End Measure

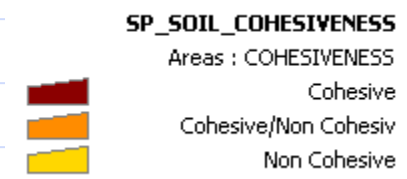
WW_SECTION to WW_LINE (1:M) aggregation

Predicates to select valid sewer lines

Implementation: Issue - Soils

- Some pipe ends fall just outside any soils polygon;
- Soil type could change anywhere along a single pipe;
- Full solution would be to split pipes and calculate depths as per "shape" and connector points;
- However, soil data is inaccurate and pipe lengths are short, so initial solution is to calculate soil type at each manhole (invert), "shape" or connector point.





Algorithm

- After calculating depths for intermediate points the first part was to:
  - Split the 3D linestring representing a single manhole-to-manhole pipe into 2-vertex vectors;
  - Construct two point sdo_geometries from ends of vector;
  - Discover soil type of both ends;
  - Compute pipe diameter;
  - Compute trench depth;
- Then from these components:
  - The final trench width at each end was computed.
  - Each end point was buffered using trench width / 2
  - A final output polygon was constructed by:
    - Unioning the two buffered polygons;
    - Computing the convex hull of the result.

Algorithm in SQL

```
SELECT ....
       sdo_geom.sdo_convexhull(
         sdo_geom.sdo_arc_densify(
           sdo_geom.sdo_union(
             sdo_geom.sdo_buffer(c.input_geom, (c.input_trench_width /2.0),0.01),
             sdo_geom.sdo_buffer(c.output_geom,(c.output_trench_width/2.0),0.01),
             0.01),
         0.01,'arc_tolerance=0.1'),
       0.01) as geom
  FROM (SELECT ...
               ROUND(case when (b.pipe_diameter + (2.0 * b.input_trench_depth  * b.input_soil_factor)) < 3.0
                          then 3.0
                          else (b.pipe_diameter + (2.0 * b.input_trench_depth  * b.input_soil_factor))
                     end,3) as input_trench_width,
                     .....) as output_trench_width,
               ...
          FROM ( SELECT ...
                        (SELECT /*+INDEX(s SP_SOIL_COHESIVENESS_GEOM)*/
                                case when s.cohesiveness = 'Cohesive' then 1 else 2 end
                                /* We use SDO_NN in case point is not within a soil polygon */
                           FROM mcw_gis.sp_soil_cohesiveness s
                          WHERE SDO_NN(s.geometry,sdo_geometry(2001,82469,sdo_point_type (v.startCoord.x,v.startCoord.y,NULL),NULL,NULL),
                                       'sdo_num_res=1') = 'TRUE'
                        ) as input_soil_factor,
                        ...
                        ) as output_soil_factor,
                        (SELECT /*+INDEX(leb LM_EASEMENT_BOUNDARY_S)*/ 1
                           FROM mcw_land.lm_easement_boundary leb
                          WHERE ( SDO_ANYINTERACT(leb.geom,sdo_geometry(2001,82469,sdo_point_type(v.startCoord.x,v.startCoord.y,NULL),NULL,NULL)) = 'TRUE'
                                  OR
                                  SDO_ANYINTERACT(leb.geom,sdo_geometry(2001,82469,sdo_point_type(v.startCoord.x,v.startCoord.y,NULL),NULL,NULL)) = 'TRUE' )
                            AND leb.id_easement_type = 3
                        ) as inEasement,
                        (SELECT case when m.diameter_outside is null or m.diameter_outside <= 225 then 0.600 else 1.000 end
                           FROM MCW_SEWER.WW_SECTION_MODEL m
                          WHERE m.FID = p.FID_MODEL)
                           as pipe_diameter,
                        ROUND((abs(v.startCoord.z) + 0.3),3) as input_trench_depth,
                        ROUND((abs(v.endCoord.z  ) + 0.3),3) as output_trench_depth,
                        sdo_geometry(2001,82469,sdo_point_type(v.startCoord.x,v.startCoord.y,NULL),NULL,NULL) as input_geom,
                        sdo_geometry(2001,82469,sdo_point_type(  v.endCoord.x,  v.endCoord.y,NULL),NULL,NULL) as output_geom
                   FROM mcw_sewer.ww_section p,
                        TABLE(mcw_sewer.zoi.GetVector(
                              SDO_LRS.CONVERT_TO_LRS_GEOM(
                                 (SELECT /*+ INDEX(l WW_LINE_IX2) */
                                         SDO_AGGR_CONCAT_LINES(l.geom)
                                    FROM mcw_sewer.ww_line l
                                   WHERE l.FID_ATTR = p.FID ),
                                 p.input_depth,
                                 p.output_depth))) v
                  WHERE p.id_function in (10004 /*gm*/,10007/*rmain*/,10008/*vmain*/)
                    AND p.input_depth is not null
                    AND p.output_depth is not null
          ) b
    WHERE b.inEasement is null
  ) c;
```

Create final Zone of Influence polygon

Trench Width Calculations

Compute soil type of input end

Output soil type computation not shown

Compute pipe diameter

Compute trench depth

Create point geometries

Custom Vectorisation function

Depth Interpolation function

Start_Measure

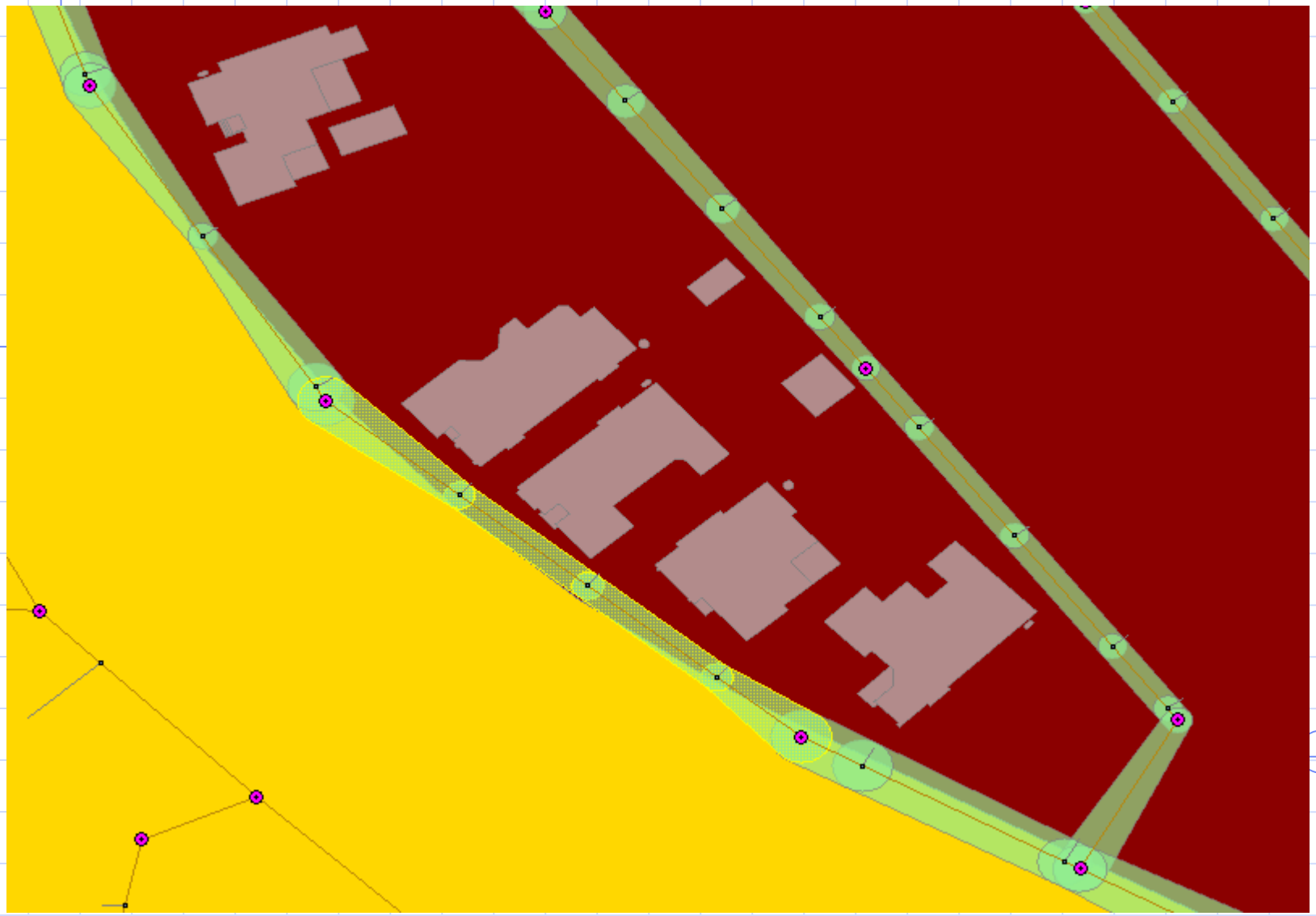
End_Measure

WW_SECTION to WW_LINE (1:M) aggregation

Not processing those already in easements

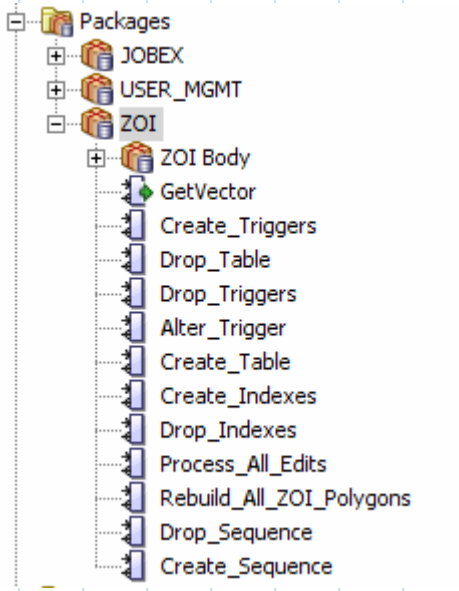
Predicates to select valid sewer lines

Output of SQL



Framework

- Now have single SQL statement that will generate all zone of influence polygons.
  (Speed is around 10 minutes to create 20,000 polygons.)
- Deployment framework created:
  - PL/SQL package developed



  - Triggers handle attribute and spatial edits

Trigger Design

- Before row trigger writes affected rows id to global temporary table.
- After statement Trigger then calls Process_All_Edits which uses modified version of SQL to create Zone of Influence Polygons.
- However...
  - After statement call is “synchronous” and so can delay write time for GIS Operator;
  - After statement Trigger is called once for each DML statement regardless as to rows affected.
    - Rows for most GIS clients = 1 as don't do array processing;
    - Updates in TopoBase split between two tables;
    - Therefore:
      - Trigger firing is inefficient.
      - And is possible to rebuild same polygon more than once as change to geometry and attributes occurs against different tables.

Trigger Solution

```
create or replace TRIGGER zoi_ww_line_aft_stmt
after insert
   or delete
   or update
on MCW_SEWER.ww_line
declare
  Pragma AUTONOMOUS_TRANSACTION; -- needed because of dbms_scheduler
  vJob Pls_Integer;
begin
  -- Check if a rebuild job is scheduled in the next 5 minutes.
  select count(*)
    into vJob
    from user_scheduler_jobs
   where job_name = 'ZOI_SWEEP'
     and run_count is null
      or run_count < 1;
  If ( vJob = 0 ) Then
    -- Spawn a job
    dbms_scheduler.create_job(
      job_name => 'ZOI_SWEEP',
      job_type => 'STORED_PROCEDURE',
      job_action => 'mcw_sewer.zoi.process_all_edits',
      number_of_arguments => 0,
      start_date => CURRENT_TIMESTAMP + 5 / (24 * 60),
      enabled => TRUE);
      commit;
  End If;
  Exception
    When No_Data_Found Then
      Null;
end;
```

- Solution is to delay execution of package's Process_All_Edits procedure.
- This is done by the after statement trigger creating a DBMS_SCHEDULER job to run a user-definable number of minutes later.
- After statement trigger checks to make sure a DBMS_SCHEDULER job does not already exist.

To Be Done...

- Sewer is done, water still to be done.
- Splitting of pipes at soils polygon boundary;
- Inclusion of elevation data to model topographic effects;
- Construction of Application Express front end to allow GIS operators to drop and create triggers, change delay time, rebuild whole Zone of Influence layer.

Conclusion

- Implementation Goals Achieved:
  - Minimal PL/SQL programming
    - GetVector() was reuse of existing code.
    - Rest of PL/SQL package was about “framework” and not programming a solution/algorithm.
  - Extract as much value from Oracle and Oracle Spatial as possible;
    - Pure SQL approach to solution with clever use of TABLE function.
    - Use of LRS functions in non-LRS application;
    - SDO_GEOM Sdo_ConvexHull, Sdo_Arc_Densify, Sdo_Buffer and Sdo_Union functions used.
    - SDO_AGGR_CONCAT_LINES() used.
    - SDO_NN for finding nearest soils polygon.
  - Implement efficient trigger based recalculation system (as computations could impose unacceptable delay on transaction commits).
- Full rebuild time is less than 10 minutes for around 23,000 polygons.

Questions

- Any questions?