

Bathymetry SPOT - Technical

Simon Greener,
The SpatialDB Advisor



Database Synchronisation

- ➔ Need to synchronise production and distribution databases.
- ➔ Approach:
 - Use “out of the box” Oracle capabilities
 - Single point of support (DBAs);
 - Integrated with data management activities (backup etc);
 - Server-side processing using single technology platform;
 - Metalink support enabled;
 - Declarative vs Programmatic;
 - Scalable in terms of processing (parallel option) and personnel (generic skills easier to contract out);



Approach

- ➔ Read Only Materialized View Replication
 - Fast Refresh
 - Web Access to distribution database may occur outside normal GeoScience Australia work hours;
 - Complete object rebuilding takes a significant enough amount of time to narrow available processing windows;
 - May cause objects to “invalidate”;
 - Fast refresh only pushes changed rows.
 - However, Fast Refreshable MVs using Oracle Objects have restrictions that introduce complexity....



Fast Refresh MVs & Oracle Spatial

- ➔ SDO_Geometry is an Oracle Object.
- ➔ There are a number of restrictions with the use of Oracle Objects in MV creation especially where FAST REFRESH is desirable.
- ➔ While 11 constraints were identified and solved, only 4 specific ones relate to Oracle Objects.



Summary of 4 Main (SDO_Geometry related) Findings

1. You can FAST REFRESH a materialized view that has an Oracle object such as SDO_Geometry in its select list only if it is based on a single table (ie one entry in a FROM clause), BUT you cannot reference it in a where clause:

```
CREATE MATERIALIZED VIEW mv_a
AS
SELECT a.ID,
       a.attribute1,
       a.GEOM
FROM table_a a
WHERE geom IS NOT NULL;
```

2. ENABLE QUERY REWRITE doesn't work when the select list contains an Oracle object such as SDO_Geometry.



Findings (2)

3. Sdo_Geometry constructors are not allowed for FAST REFRESH:

```
CREATE MATERIALIZED VIEW mv_a
BUILD IMMEDIATE REFRESH FAST ON DEMAND
AS
SELECT id,
       MDSYS.SDO_GEOMETRY(2003,8311,NULL,
                          MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
                          MDSYS.SDO_ORDINATE_ARRAY(a.W_LONG,a.S_LAT,a.E_LONG,a.N_LAT))
FROM table_a a;
```

You will get this reported in the MV_CAPABILITIES_TABLE:

"the reason why the capability is disabled has escaped analysis"

4. Union All MVs that include SDO_Geometry will not work because (cf 2.):

"Each query block in the UNION ALL query must satisfy the requirements of a fast refreshable materialized view with aggregates or a fast refreshable materialized view with joins".

The Geom attributes need to be moved to separate materialized views

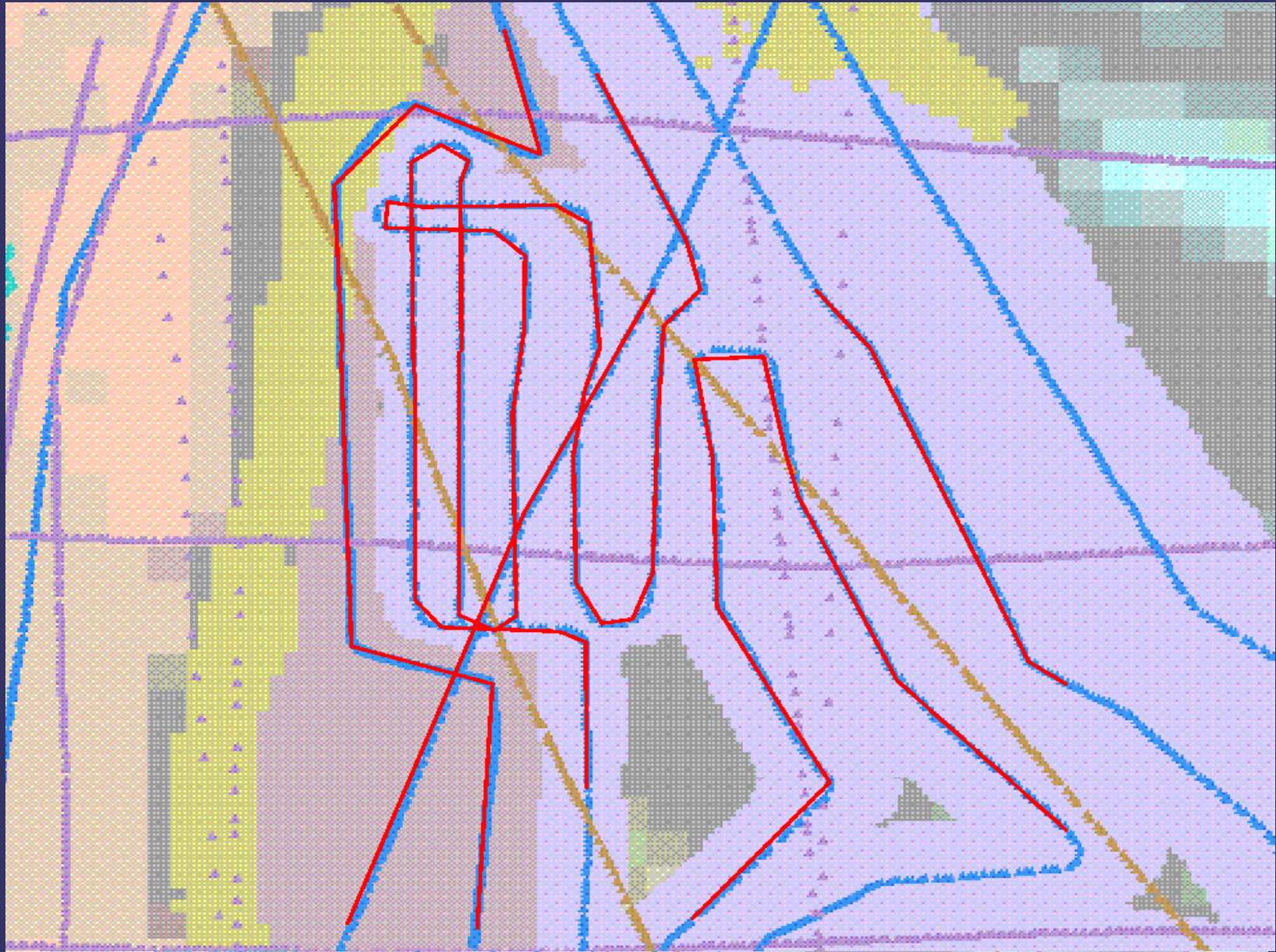


MultiPoints

- ➔ Bathymetry SPOT contains significant holding of 3D point data.
- ➔ One table alone (MarineObs) has nearly 70 million observations, yet these only describe around 13,000 surveys.
- ➔ Tests made to see if use of multi-point descriptions of surveys might help in distribution.
- ➔ Surrogate linestrings are used to describe these objects as query and draw based on MarineObs would be slow (though GA has Parallel Option) – see next slide.
- ➔ Multi-points objects considered more “natural” to domain experts and addressed representation issues with surrogate linestrings.



MultiPoint vs Surrogate Linestrings



Multi-Point Issues

➔ Construction of Multi-Points

- Need efficient approach to building from individual observations.
 - Synchronisation with Fast Refresh required custom solution.
 - Small number of singlebeam marine surveys break SDO_ORDINATE_ARRAY ordinate limit.
 - The ordinate array can only hold 1,048,576 ordinates which is 349,525 3D points.
 - Had to modify distribution data model to support required 1:M model change.



Oracle 10g Spatial Appendix D

➔ Example:

SDO_AGGR_UNION query (shown with Parallel option) using recommended method from Appendix D of Oracle Spatial 10gR2 documentation.

(HAVING clause to filter out large surveys not included).

```
SELECT /*+ PARALLEL(3) */
      SDO_AGGR_UNION(MDSYS.SDOAGGRTYPE(aggr_geom,0.05))
FROM ( SELECT /*+ PARALLEL(3) */
      sb_eno,
      SDO_AGGR_UNION(MDSYS.SDOAGGRTYPE(aggr_geom,0.05))
      as aggr_geom
FROM (SELECT /*+ PARALLEL(3) */
      sb_eno,
      SDO_AGGR_UNION(
        MDSYS.SDOAGGRTYPE(aggr_geom,0.05))
      as aggr_geom
FROM (SELECT /*+ PARALLEL(3)*/
      sb_eno,
      SDO_AGGR_UNION(
        MDSYS.SDOAGGRTYPE(aggr_geom,
          0.05))
      as aggr_geom
FROM (SELECT /*+ PARALLEL(3)*/
      a.SB_ENO,
      SDO_AGGR_UNION(
        MDSYS.SDOAGGRTYPE(a.geom,
          0.05))
      as aggr_geom
FROM mv_singlebeam_bathymetry a
WHERE a.sb_eno = rec.sb_eno
GROUP BY a.SB_ENO,
         mod(rownum,16)
      )
      )
      GROUP BY SB_ENO, mod (rownum, 8)
      )
      GROUP BY SB_ENO, mod (rownum, 4)
      )
      GROUP BY SB_ENO, mod (rownum, 2)
      )
GROUP BY SB_ENO;
```



MultiPoint Creation Issues

- ⇒ Initially used SDO_AGGR_UNION to test construction feasibility;
 - Used Appendix D (nested SQL);
 - Parallel processing improved performance (still slow).
 - SDO_AGGR_UNION still had to be wrapped by PL/SQL procedure to handle SDO_ORDINATE_ARRAY limits.
- ⇒ A faster algorithm/method for constructing multi-point objects through a pure PL/SQL implementation was devised.
 - Initial implementation provided sufficient performance to warrant further investigation.
 - The PL/SQL package DBMS_PROFILER was used to identify bottlenecks in the implementation and various changes were made and compared. The following chart shows the final set of implementations.



Essentials

➔ Adopted approach has these SQL statements

```
SELECT ords.*
  BULK COLLECT INTO v_3D_ordinates
  FROM mv_singlebeam_bathymetry a,
       TABLE(mdsys.sdo_ordinate_array(a.geom.sdo_point.x,
                                       a.geom.sdo_point.y,
                                       a.geom.sdo_point.z) ) ords

  WHERE a.sb_eno = rec.sb_eno
  ORDER BY a.pointno;
```

➔ Where point count < ordinate limit

➔ Where point count > ordinate limit

```
SELECT ords.*
  BULK COLLECT INTO v_3D_ordinates
  FROM (SELECT rownum as pointorder,geom
        FROM mv_singlebeam_bathymetry a
        WHERE a.sb_eno = rec.sb_eno
        ORDER BY a.pointno ) b,
       TABLE(mdsys.sdo_ordinate_array(b.geom.sdo_point.x,
                                       b.geom.sdo_point.y,
                                       b.geom.sdo_point.z) ) ords

  WHERE b.pointorder BETWEEN v_min_point AND v_max_point;
v_mpoint_rec.pointcount := ( v_max_point - v_min_point + 1 );
v_mpoint_rec.geom_mpoint := MDSYS.SDO_GEOMETRY(3005,8311,NULL,
                                                MDSYS.SDO_ELEM_INFO_ARRAY(1,1,v_mpoint_rec.POINTCOUNT),
                                                v_3D_ordinates);
```



Comparison Table

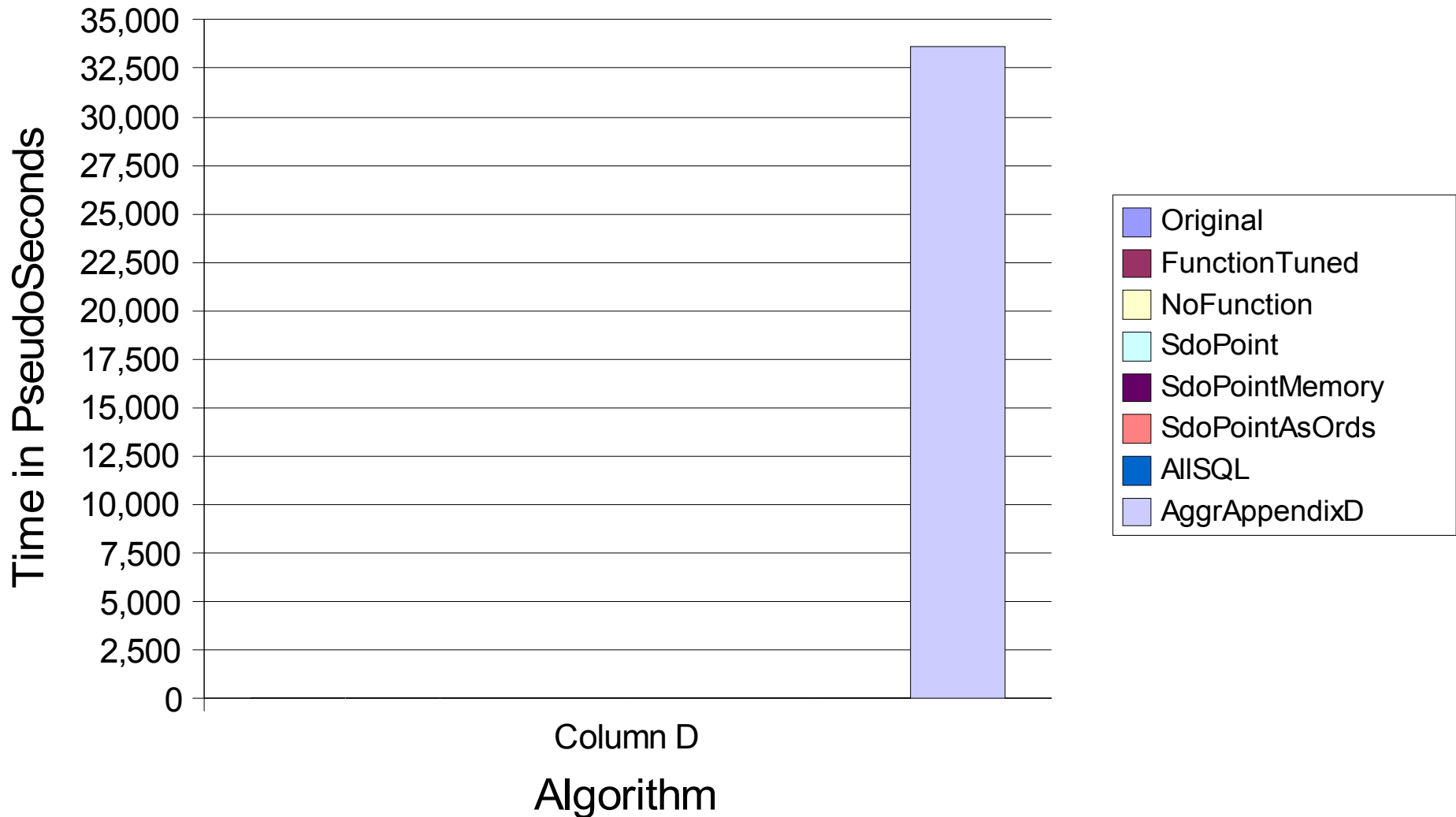
Algorithm	Time to Complete (nanoseconds)	Milliseconds	PseudoSeconds	Minutes
Original	3.39E+012	3.39E+006	34	0.57
FunctionTuned	3.34E+012	3.34E+006	33	0.56
NoFunction	5.44E+011	5.44E+005	5	0.09
SdoPoint	8.95E+011	8.95E+005	9	0.15
SdoPointMemory	8.32E+011	8.32E+005	8	0.14
SdoPointAsOrds	7.84E+011	7.84E+005	8	0.13
AllSQL	1.99E+012	1.99E+006	20	0.33
AggrAppendixD	3.36E+015	3.36E+009	33,613	560.22

- ➔ Algorithm that used SDO_AGGR_UNION
1000 times slower than worst custom
PL/SQL



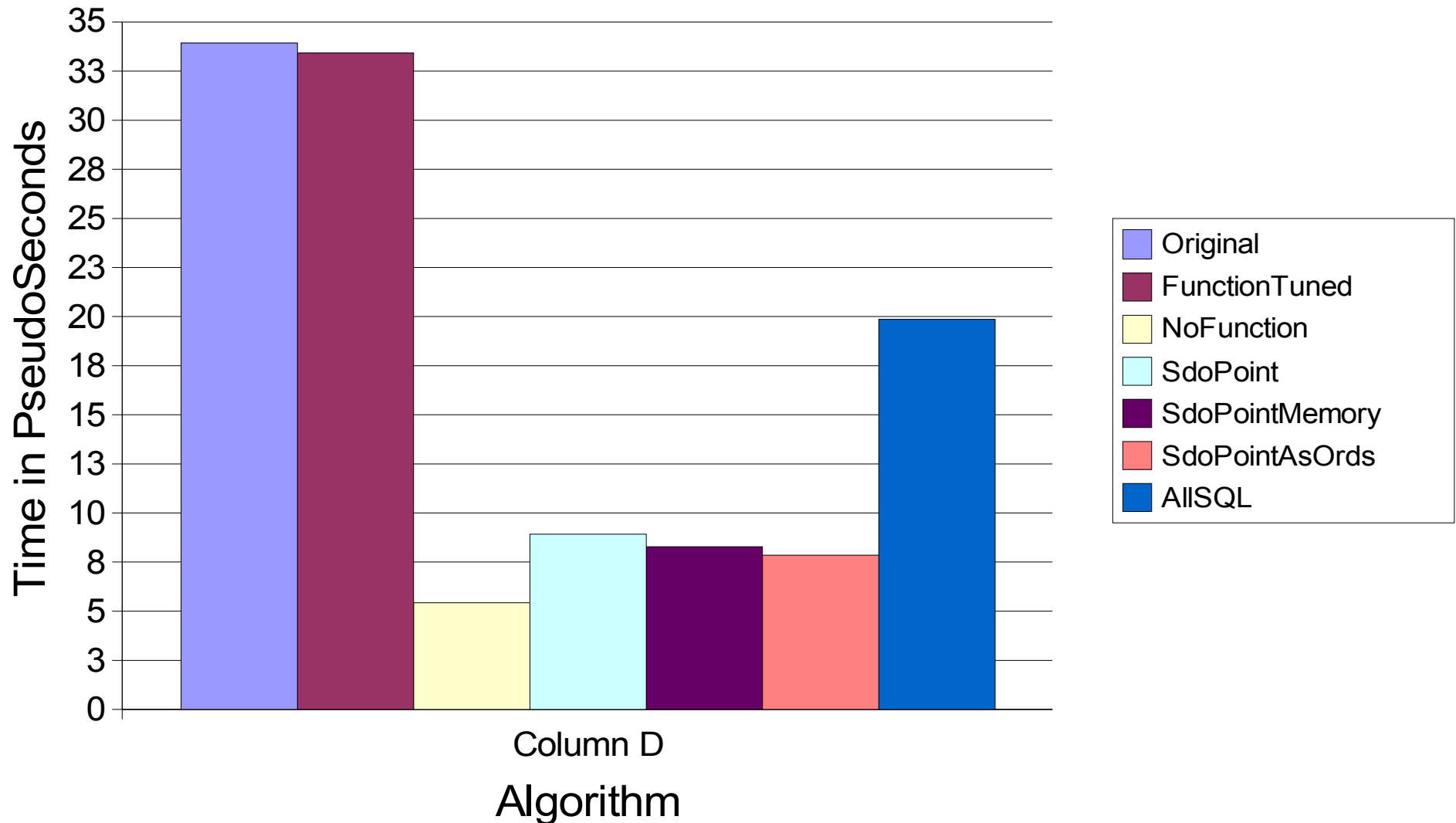
Comparison Chart (1)

Custom PL/SQL vs SDO_AGGR_UNION



Comparison Chart 2

Comparison of Non-SDO_AGGR_UNION Algorithms



Solution...

- ⇒ Aggregation approach inside final PL/SQL function used custom code and not SDO_AGGR_UNION.
 - Procedure has 100 lines of code (all work done in 55 lines).
 - Generation of 13,000 individual surveys from 70million individual 3D points took 50minutes!
- ⇒ DBMS_PROFILER was a great tool!
- ⇒ Oracle, please improve:
 - Performance of SDO_AGGR_UNION
 - Remove limit on SDO_ORDINATE_ARRAY

