Oracle Spatial Best Practices and Tuning Tips for DBAs and Developers

Daniel Geringer
Senior Software Development Manager
Oracle’s Spatial Technologies
Spatial Data Validation
Spatial data comes from many sources.
Very often, spatial data contains invalid geometries.
Spatial analysis results are only guaranteed when geometries are valid.
Very important to run geometry validation, either:
- `VALIDATE_GEOMETRY_WITH_CONTEXT`
- `VALIDATE_LAYER_WITH_CONTEXT`
The table must be created before calling the procedure

```sql
DROP TABLE validation_results;
CREATE TABLE validation_results (
    sdo_rowid ROWID,
    status VARCHAR2(2000));
```

- The user defines the table name and column names
- The columns must be in data type order
Validate Layer with Context Example (continued)

BEGIN
  SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT
    ('RIVERS', 'GEOM', 'VALIDATION_RESULTS');
END;
/

SELECT * FROM validation_results;

<table>
<thead>
<tr>
<th>SDO_ROWID</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rows Processed &lt;752&gt;</td>
</tr>
<tr>
<td>AAADCsAABAAAPUpAAA</td>
<td>13341 [Element &lt;1&gt;]</td>
</tr>
<tr>
<td>AAADCsAABAAAPUpAAB</td>
<td>13356 [Element &lt;1&gt;] [Coordinate &lt;2&gt;]</td>
</tr>
<tr>
<td>AAADCsAABAAAPUpAAC</td>
<td>13356 [Element &lt;1&gt;] [Coordinate &lt;4&gt;]</td>
</tr>
<tr>
<td>AAADCsAABAAAPUpAAD</td>
<td>13356 [Element &lt;1&gt;] [Coordinate &lt;3&gt;]</td>
</tr>
<tr>
<td>AAADCsAABAAAPUpAAE</td>
<td>13356 [Element &lt;1&gt;] [Coordinate &lt;4&gt;]</td>
</tr>
</tbody>
</table>

ORA-13341 = a line geometry has less than two coordinates
ORA-13356 = adjacent points in a geometry are redundant
Fixing Invalid Geometries

Repeat the following for each type of error.

Isolate invalid geometries

```sql
CREATE TABLE roads_13356 AS
SELECT *
FROM roads
WHERE rowid in (SELECT sdo_rowid
                     FROM validation_results
                     WHERE substr(STATUS, 1, 5) = 13356);
```

Remove invalid geometries from your spatial layer

```sql
DELETE FROM roads
WHERE rowid in (SELECT sdo_rowid
                 FROM validation_results
                 WHERE substr(STATUS, 1, 5) = 13356);
```
Fixing Invalid Geometries (Continued)

SDO_UTIL.RECTIFY_GEOMETRY fixes geometries

-- Make a copy of the geometries before they are updated.
CREATE TABLE roads_13356_save AS SELECT * FROM roads_13356;

UPDATE roads_13356
SET geom = sdo_util.rectify_geometry (geom, tolerance) ;

• After update, run validation on ROADS_13356
• Insert all fixed geometries back into the ROADS table
• Repeat this process for each type of validation error
Create Spatial Index
Best Practices
Create Spatial Index
Some Important Parameters

• LAYER_GTYPE
  • Can be set to:
    • POINT
    • LINE or CURVE
    • POLYGON or SURFACE
    • COLLECTION
    • MULTIPOINT
    • MULTILINE or MULTICURVE
    • MULTIPOLYGON or MULTISURFACE
    • SOLID
    • MULTISOLID
  • Ensures all data inserted into spatial column is a certain type.
Create Spatial Index
Some Important Parameters (continued)

• LAYER_GTYPE (continued)
  • Very important when indexing point data
  • Faster queries – internal optimizations

• TABLESPACE
  • If you want to specify the tablespace for the MDRT$ table
  • If not specified, MDRT$ created in users default tablespace

• WORK_TABLESPACE
  • Very important
  • Reduces fragmentation in the index tablespace
  • Requires 2 times the final index size
  • Work tablespace can be dropped or reused after index is created
Create Spatial Index
Some Important Parameters (continued)

Estimate the size in MB, of an R-Tree index on a table with:
- 250,000 geometries
- 8K DB_BLOCK_SIZE
- SDO_RTR_PCTFREE=15 (percent)
- Two dimensional
- Is geodetic

```sql
SELECT sdo_tune.estimate_rtree_index_size
(250000, 8192, 15, 2, 1)
FROM dual;
```

Create work tablespace at least two times this size.
Create Spatial Index - Examples

Create index example for line spatial layer

```
CREATE INDEX roads_sidx ON roads (geom)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('tablespace=index_tablespace
  work_tablespace=work_ts
  layer_gtype=line');
```

Create index example for point spatial layer

```
CREATE INDEX cities_sidx ON cities (geom)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('tablespace=index_tablespace
  work_tablespace=work_ts
  layer_gtype=point');
```
Local Partitioned Spatial Indexes
Best Practices
Local Partitioned Spatial Indexes

- Major benefits are performance and manageability of very large data sets
- **EXCHANGE PARTITION INCLUDING INDEXES** supported for spatial indexes too
- Customer example:
  - Requirement:
    - Ingest and maintain 2 days of weather data online
    - 270,000 samples every 30 seconds
  - Implemented with:
    - 30 second partitions (5760 partitions over 2 days)
    - New partitions rolled on, older partitions rolled off
ALTER TABLE weather_data_part EXCHANGE PARTITION p1
WITH TABLE new_weather_data
INCLUDING INDEXES
WITHOUT VALIDATION;

• Parallel create index (spatial index too) on new_weather_data
• Partition P1 is an empty leading partition
• Update partitioned table with new weather data in a fraction of a second.
• No need to maintain an index on INSERT.

ALTER TABLE weather_data_part EXCHANGE PARTITION p1
WITH TABLE new_weather_data
INCLUDING INDEXES
WITHOUT VALIDATION;
Create LOCAL Partitioned Spatial Indexes

Create LOCAL spatial index with UNUSABLE keyword

```
CREATE INDEX yp_part_sidx ON yellow_pages_part (location)
INDEXTYPE IS MDSYS.Spatial_Index
PARAMETERS ('LAYER_GTYPE=POINT')
LOCAL
(PARTITION p1 PARAMETERS ('TABLESPACE=TBS_1'),
PARTITION p2 PARAMETERS ('TABLESPACE=TBS_2'),
PARTITION p3 PARAMETERS ('TABLESPACE=TBS_3'),
PARTITION p4 PARAMETERS ('TABLESPACE=TBS_4'),
PARTITION p5 PARAMETERS ('TABLESPACE=TBS_5'),
PARTITION p6 PARAMETERS ('TABLESPACE=TBS_6') )
UNUSABLE;
```
Create Local Partitioned Spatial Indexes (continued)

• Rebuild each index with ALTER INDEX REBUILD
• Can invoke multiple ALTER INDEX REBUILD’s at a time
• Advantage: If one partition’s index fails, it doesn’t invalidate others that have completed.

```
ALTER INDEX yp_part_sidx REBUILD PARTITION p1;
ALTER INDEX yp_part_sidx REBUILD PARTITION p2;
ALTER INDEX yp_part_sidx REBUILD PARTITION p3;
ALTER INDEX yp_part_sidx REBUILD PARTITION p4;
ALTER INDEX yp_part_sidx REBUILD PARTITION p5;
ALTER INDEX yp_part_sidx REBUILD PARTITION p6;
```
Spatial Query
Best Practices
Spatial Operators

• SDO_FILTER
• SDO_RELATE
• SDO_ANYINTERACT
• SDO_WITHIN_DISTANCE
• SDO_NN
• SDO_INSIDE
• SDO_OVERLAPS
• SDO_TOUCH
• SDO_EQUAL
• Etc…
Spatial Operators – Template

- All spatial operators follow a similar template
- First argument is always the search column
- Second argument is always the query window

For Example:

```sql
SELECT a.customer_name, a.phone_number
FROM policy_holders a
WHERE sdo_within_distance(a.geom, hurricane_path_geom,
                          'distance = 10 unit = mile') = 'TRUE';
```
Spatial Operators – Ordered Hint

• When query window comes from a table, use the ORDERED hint.
• ORDERED hint is not a spatial specific
• Hints the optimizer to use indexes on tables as they are listed in the FROM clause
• Execution plan tracing is important

For Example:

```
SELECT /*+ ordered */
    i.highway
FROM geod_states s,
    geod_interstates i
WHERE s.state = 'Arizona'
    AND sdo_relate (i.geom, s.geom,'mask=ANYINTERACT')='TRUE';
```
Look At Your Execution Plan

From SQL*Plus:

```
SQL> SET AUTOTRACE TRACE EXPLAIN
SQL> SET AUTOTRACE OFF
```
Which of the millions of roads in the U.S. have some interaction with this county?
Spatial Optimizations – Behind the Scenes

Which of the millions of roads in the U.S. have some interaction with this county?
Spatial Optimizations – Behind the Scenes

Which of the millions of roads in the U.S. have some interaction with this county?

- Primary filter compares geometry approximations, so result is not exact.
Which of the millions of roads in the U.S. have some interaction with this county?

- Primary filter compares geometry approximations, so result is not exact.
Spatial Optimizations – Behind the Scenes

Which of the millions of roads in the U.S. have some interaction with this county?

- Primary filter compares geometry approximations, so result is not exact.
- Interior optimizations are applied to candidate set.
Spatial Optimizations – Behind the Scenes

Which of the millions of roads in the U.S. have some interaction with this county?

- Primary filter compares geometry approximations, so result is not exact.
- Interior optimizations are applied to candidate set.
Spatial Optimizations – Behind the Scenes

Which of the millions of roads in the U.S. have some interaction with this county?

- Primary filter compares geometry approximations, so result is not exact.
- Interior optimizations are applied to candidate set.
- Geometry comparisons are done only where required.
Spatial Operators

When possible make the query window a polygon.
Nearest Neighbor – SDO_NN

• Very effective way to find geometries closest to a window geometry.
• For example, find the five closest banks to my location.
• SDO_NUM_RES or SDO_BATCH_SIZE parameter, which should you use?
  • SDO_NUM_RES
    • When only proximity is considered (closest bank example above)
  • SDO_BATCH_SIZE
    • When other columns from the same table as the nearest neighbor search column are considered in the WHERE clause.
    • For example, find the five closest banks named Citibank.
    • The Bank table’s geometry and bank_name columns are being searched.
• Find the 5 closest banks
• Only proximity is considered, so use SDO_NUM_RES.

```
SELECT b.bank_name,
       sdo_nn_distance (1) distance
FROM banks_table b
WHERE sdo_nn (b.geom, :my_hotel,
                 'SDO_NUM_RES=5', 1) = 'TRUE';
```
SDO_NN Example – With SDO_BATCH_SIZE

- Find the 5 closest banks named Citibank
- A column besides the SDO_GEOMETRY column in the BANK_TABLE is considered, use SDO_BATCH_SIZE.
- This is the new recommended way to write SDO_NN queries with SDO_BATCH_SIZE.

```
SELECT bank_address
FROM (SELECT /*+ FIRST_ROWS */ b.bank_address
     FROM bank_table b
     WHERE SDO_NN(b.geometry, :my_hotel,
                   'sdo_batch_size=10', 1) = 'TRUE'
           AND b.bank_name = 'Citibank'
     ORDER BY SDO_NN_DISTANCE(1))
WHERE ROWNUM <= 5;
```
SDO_NN Example – Distance Parameter

• Stop searching for nearest neighbor once the cutoff distance is reached (specified by the distance parameter).
• Distance can be specified with both SDO_NUM_RES or SDO_BATCH_SIZE.
• Find the 5 closest banks, but none more than 2 miles away.

```sql
SELECT b.bank_name, 
  sdo_nn_distance (1) distance 
FROM banks_table b 
WHERE sdo_nn (b.geom, :my_hotel, 
  'SDO_NUM_RES=5 DISTANCE=2 UNIT=mile' , 1) = 'TRUE';
```
Anyone using SDO_NN and Oracle 10.2.0.4, 11.1.0.6 or 11.1.0.7, make sure the following patches are applied:

- Patch 8758818
- Patch 8773211

A patch merge label already applied to your database may contain patches 8758818 and 8773211. Through a service request, Oracle Support can verify this for you.

Oracle 11.2 contains the fixes for patches 8758818 and 8773211.
Spatial Operators Can Parallelize with Create Table As Select (CTAS)
Parallel and CTAS With Spatial Operators

- Spatial operators can parallelize with CTAS when multiple candidates feed the second argument.
  For example:

A GPS records thousands of train positions. For each:
- Find the closest track to the train (with SDO_NN)
- Then calculate the position on the track closest to the train

```sql
CREATE TABLE results NOLOGGING PARALLEL 4
AS SELECT /*+ ordered */
    a.locomotive_id,
    sdo_lrs.find_measure (b.track_geom, a.locomotive_pos) measure
FROM locomotives a,
     tracks b
WHERE sdo_nn (b.track_geom, a.locomotive_pos, 'sdo_num_res=1') = 'TRUE';
```
Parallel and CTAS With Spatial Operators

- Works with all spatial operators:
  - SDO_ANYINTERACT
  - SDO_INSIDE
  - SDO_TOUCH
  - SDO_WITHIN_DISTANCE
  - SDO_NN
  - Etc…

- SDO_NN and CTAS with parallel minimally requires Oracle Database versions 10.2.0.4 or 11.1.0.7.
Transportable Tablespaces and Spatial
Transportable Tablespaces and Spatial

• Very fast way to move large spatial dataset from one instance to another
• Can also archive data in TTS format, for example, rolling partitions that age out

**Spatial data** - can transport across hardware with different Endian formats

**Spatial index** -
  • Prior to Oracle 11g Release 2, transports require hardware with same Endian format
  • Oracle 11g Release 2 and beyond, transports between hardware with different Endian format supported. Requires running RMAN.
-- Each user with spatial indexes associated with tables
-- in transportable tablespace TTBS_DAT that will also be
-- moved as part of the transport set will need to do the
-- following. This example is for user SCOTT. SCOTT has
-- tables in TTBS_DAT and indexes in TTBS_IDX.
CONNECT system/oracle

-- First, make sure SCOTT’s default tablespace is in the
-- transport set, because SDO_UTIL.PREPARE_FOR_TTS will
-- create a table with metadata to be moved in the tablespace.
ALTER USER scott DEFAULT TABLESPACE TTBS_DAT;
CONNECT scott/tiger

-- Only need to perform this step if the source database
-- release is older than 11.2.0.1 (11g Release 2),
-- In 11g Release 2 and beyond, no need to call PREPARE_FOR_TTS.
EXECUTE SDO_UTIL.PREPARE_FOR_TTS('TTBS_DAT');
Transportable Tablespace: Example

-- Connect as sysdba to check if tablespace is OK to move
CONNECT system/oracle AS SYSDBA
-- Execute procedure which checks transportability
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('TTBS_DAT,TTBS_IDX',TRUE);
-- Valid transport set if the following returns no rows
SELECT * FROM TRANSPORT_SET_VIOLATIONS;
-- Set tablespace to read only
ALTER TABLESPACE ttbs_dat READ ONLY;
ALTER TABLESPACE ttbs_idx READ ONLY;

-- Using data pump export, the directory to write the dump
-- file (which will contain metadata only) has be be created
-- on the file system, and also in the database.
-- For example: mkdir d:\labs\data
CREATE DIRECTORY student_dumpdir AS 'd:\labs\data';
EXIT;
Transportable Tablespace: Example

-- NOTE, for expdp, all parameters must be on the same line.
expdp system/oracle DIRECTORY=student_dumpdir
DUMPFILE=ttbs_md.dmp TRANSPORT_TABLESPACES=TTBS_DAT,TTBS_IDX
-- Copy dump (ttbs_md.dmp) and tablespace files
-- (ttbs_dat.dbf and ttbs_idx.dbf) to the other Oracle instance.
-- The tablespace dbf files can go to their destination
-- directory.
-- Create a directory on the file system and in the database.
-- For example:
--   mkdir d:\labs\datadmp
--   Copy the dump file (ttbs_md.dmp) to that directory.
-- Cross-endian transportable tablespaces are supported with
-- spatial data and not supported with spatial indexes
CONNECT system/oracle
CREATE DIRECTORY other_dumpdir AS 'd:\labs\datadmp';
EXIT;
## Transportable Tablespace: Example

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Import using data pump. All parameters below should be</td>
</tr>
<tr>
<td>-- on the same line, or in a parameter file</td>
</tr>
<tr>
<td><code>impdp system/oracle DIRECTORY=other_dumpdir</code></td>
</tr>
<tr>
<td><code>DUMPFILE=ttbs_md.dmp</code></td>
</tr>
<tr>
<td><code>TRANSPORT_DATAFILES='d:\course_datafiles\course_dat.dbf',</code></td>
</tr>
<tr>
<td><code> 'd:\course_datafiles\course_idx.dbf'</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Allow reads and writes on new tablespaces (just imported)</td>
</tr>
<tr>
<td><code>sqlplus system/oracle</code></td>
</tr>
<tr>
<td><code>ALTER TABLESPACE ttbs_dat READ WRITE;</code></td>
</tr>
<tr>
<td><code>ALTER TABLESPACE ttbs_idx READ WRITE;</code></td>
</tr>
</tbody>
</table>
Transportable Tablespace: Example

-- Call initialize_indexes_for_tts if one of the following is true:

-- -- The source transport was created in a pre 11.2 database.
-- -- The source and target databases have a different Endian format.
-- For each user who has spatial indexes in transportable tablespace:
CONNECT scott/tiger;
EXEC sdo_util.initialize_indexes_for_tts;

--

-- If the transport was generated in a pre 11.2 database, and transported to an 11.2 or newer database, then each spatial index must be altered with 'CLEAR_TTS=TRUE'
ALTER INDEX xyz1_spatial_idx PARAMETERS ('CLEAR_TTS=TRUE');
ALTER INDEX xyz2_spatial_idx PARAMETERS ('CLEAR_TTS=TRUE');
Oracle Application Server
MapViewer
MapViewer (Leverage Oracle Maps Feature)

- Easy to use Javascript API
- No plug-ins
Oracle Spatial
Network Data Model
Network Data Model Analysis: Shortest Path Between Two Nodes

Oracle Spatial Network Data Model Demo

- **ShortestPath**
  - NearestNeighbor
  - AllPaths
  - WithinCost
  - TSP
  - Show/Hide Partitions
  - Show Route
  - Show Link

**Network Analysis on Network [HDML_CA]**
- **Start Node ID**: 199598844
- **End Node ID**: 199755839

**Analysis Result**
- **Path Information**: (199598844, 199755839)
- **Cost**: 1.440.46, 44 links

**Execution Statistics**
- **Analysis and partition loading took**: 0.116 sec
- **Path P01 generation took**: 0.229 sec

**Network Analyst Function**
```
NetworkAnalyst.shortestPathDijkstra(...)
```
Network Data Model Analysis: All Paths Between Two Nodes

Oracle Spatial Network Data Model Demo

All Paths Analysis on [Network] [ roadmap ]
- Start Node ID: 199560667
- End Node ID: 1995606722
- Number of Paths: 5

Analysis Result:
Result contains 5 paths:
1: (199560667->1995606722)
[cost:5903.39,54 links]
2: (199560667->1995606722)
[cost:5903.39,54 links]
3: (199560667->1995606722)
[cost:5903.39,54 links]
4: (199560667->1995606722)
[cost:5903.39,54 links]
5: (199560667->1995606722)
[cost:5903.39,54 links]

Analysis and partition loading took: 1.005 sec...
Path F01 generation took: 0.215 sec...

NetworkAnalyst.kShortestPaths(...)

ORACLE
Network Data Model Analysis: All Nodes Within Cost

Oracle Spatial Network Data Model Demo

Within Cost Analysis on Network: [NDM_CAI]
- Start Node ID: 199540121
- Cost: 750

Show Path: All Paths
Analysis Result:
- Result contains 72 paths:
- Analysis and partition loading took 0.040 sec...
- Path FDI generation took 0.064 sec...

Execute

NetworkAnalyst.withinCost(…)

Oracle Spatial Network Data Model Demo
Network Data Model Analysis:
Traveling Salesman Problem (Setup)

Oracle Spatial Network Data Model Demo

Closed TSP Analysis on Network[MOM_CA]
- Tsp Node[1] 19360.124
- Tsp Node[2] 19360.748
- Tsp Node[3] 19375.983
- Tsp Node[5] 19374.224
- Tsp Node[8] 1931.957
- Tsp Node[9] 19633.111
- 1 More Tsp Node...

Analysis Result:
Path information:
10580.124-15980.124
[cost:268.59, 235 links]

Analysis and partition loading took 0.972 sec...
Path FCM generation took 1.306 sec...

NetworkAnalyst.tspPath(…)

ORACLE
Network Partitioning Best Practices

- Increase the size of the network partition cache. After the cache is full, a Least Recently Used (LRU) algorithm is used to purge partitions from the cache and load others.
- Increase Java heap size, for example, Java –Xmx –xms.
- Generate partition BLOBs, a binary representation of the network partitions that make it much faster to load them into memory.
- Choose an optimal partition size. Very small partitions may result in excessive partition loads. If partitions are too large, more data may get loaded than is necessary to perform network analysis.
- Try to minimize the number of links between partitions.
- Leverage hierarchical networks when possible.
- If running on 11.1.0.7, apply patch 7700528 available on metalink.
Some Patch Recommendations
Spatial Patch Recommendations

- If possible, upgrade to 11g Release 2.
- The following patches, and patch are recommended for bug fixes and better performance and are available on MetaLink.

- For Oracle 10.2.0.3:
  - Apply 10.2.0.4 and patches mentioned below or upgrade to Oracle 11.1.0.7 or 11.2.0.1.

- For Oracle 10.2.0.4:
  - 8930049 – Spatial merge label with several fixes. If not available on your hardware platform, please open a service request to request a backport.
  - 6989483 – Redefinition of all_sdo_geom_metadata view, improves performance
  - 6501889 – RDBMS patch if using local partitioned spatial indexes and not running in a RAC environment.
Spatial Patches - Continued

• For Oracle 11.1.0.6:
  • Apply Oracle 11.1.0.7 patchset, or upgrade to 11g Release 2.

• For Oracle 11.1.0.7:
  • Patch 6501889 – RDBMS patch if using local partitioned spatial indexes and not running in a RAC environment.
  • Patch 6504890 – Spatial patch if using local partitioned spatial indexes and not running in a RAC environment.
  • Patch 7307918 – Improved SDO_ANYINTERACT performance for MBR queries.
  • Patch 7700528 - Network Data Model Load On Demand patch.
  • Patch 8758818 – SDO_NN performance optimization when using DISTANCE parameter
  • Patch 8773211 – SDO_NN optimization patch

• For MapViewer:
  • Use latest production version available on the MapViewer page on OTN
  • http://www.oracle.com/technology/software/products/mapviewer/index.html