Best Practices, Tips and Tricks With Oracle Spatial and Graph

Daniel Geringer, Spatial Solutions Specialist
Oracle Corporation
Oracle 12.2 – Some New Features
Oracle 12.2 – Some New Features

System Managed Spatial Indexes

• In 12.2, System Managed Spatial Indexes
• During CREATE INDEX, specify INDEXTYPE=MDSYS.SPATIAL_INDEX_V2
• _V2 recommended for partitioned and non-partitioned indexes
• Local partitioned spatial index no longer limited to Range
  – Range, Hash, List
  – Composite partitioning
  – Interval partitioning
  – Reference partitioning (partition a table like another partitioned table)
  – Virtual column partition key
Oracle 12.2 – Some New Features

Composite B-Tree Spatial Index – For Point Data

• Only for point data
• Use B-Tree spatial index:
  – To eliminate spatial index contention associated with concurrent DML from multiple sessions (ie connection pool)
  – Faster spatial index creation
• No spatial functionality compromised
• B-Tree query performance comparable to R-Tree (nearest neighbor a little better with R-Tree)
• Specify cbtree_index=true

  CREATE INDEX point_sidx ON cities (geometry)
  INDEXTYPE IS mdsys.spatial_index_v2
  PARAMETERS ('layer_gtype=point cbtree_index=true');
LAYER_GTYPE=POINT
**LAYER_GTYPE=POINT – For Point Data**

- Provides faster Nearest Neighbor performance

```sql
CREATE INDEX point_sidx ON cities (geometry) indextype is mdsys.spatial_index_V2
PARAMETERS ('layer_gtype=point');
```
Oracle 12c
Optimizer is Spatial Aware
Spatial Statistics - Used When Query Window Is Dynamic

- DBMS_STATS also gathers statistics on spatial column and index
- MDXT and MDXT_MBR tables created with histogram information
- Spatial queries need less hints going forward.
Spatial Vector Acceleration
New faster algorithms for spatial operators and functions (100’s of times faster)

Metadata caching increases performance:
  – For all spatial operators and functions
  – For all DML operations (INSERT, UPDATE, DELETE)

Recommended for any application with mission critical spatial query performance requirements.

Requires Oracle Spatial and Graph License
  – ALTER SYSTEM SET SPATIAL_VECTOR_ACCELERATION = TRUE
  – ALTER SESSION SET SPATIAL_VECTOR_ACCELERATION = TRUE
Spatial operators

- Performance optimizations for "high vertex count" query window (2\textsuperscript{nd} argument of spatial operator).
- Relation masks hundreds of times faster (ie. COVEREDBY, COVERS, TOUCH, etc..)
  - Time Zone Polygon Example
  - Very detailed coast line
  - 343,395 vertices
  - Hundreds of times faster
  - 300x faster for this test
Oracle Spatial and Graph – Included in the following cloud offerings

- Oracle Database Cloud Service – High Performance and Extreme Performance
- Oracle Database Exadata Cloud Service
- Oracle Database Exadata Cloud Machine
- Oracle Database Exadata Express Cloud Service
Geometry Validation
Fastest Way To Validate Geometries – With Parallel Query

• Similar output to SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT
• You control the parallel degree

```
CREATE TABLE validation_results PARALLEL 16 NOLOGGING AS
SELECT sdo_rowid, status
FROM (SELECT rowid sdo_rowid,
      validate_geometry_with_context (geom, tolerance) status
FROM roads)
WHERE status <> ' TRUE';
```
Alter Parallel Degree Of Tables To 1
Alter Table Parallel 1 - Recommendation

• After a table is created with parallel DDL, set table’s parallel degree to 1.
  – CREATE TABLE results NOLOGGING PARALLEL 16 AS SELECT ...
  – ALTER TABLE results PARALLEL 1;
  – SELECT table_name, degree FROM user_tables;

• Ensures you are not using parallel query when you don’t intend to

• Enables you to control query parallel degree with a parallel hint for example:
  – SELECT /*+ parallel (16) */ FROM results WHERE...
  – INSERT /*+ APPEND PARALLEL (16) */ INTO table_name NOLOGGING SELECT column FROM ...
Point In Polygon – Best Practices
Point In Polygon – Best Practices

• Common requirement for many applications, i.e. insurance, permitting, etc.:
  – Does a customer address (point) fall in a high risk zone (polygon)
  – Does a permit application address (point) fall in an environmentally protected region (polygon)

• Two major considerations of polygon “spatial layers” (not query window):
  – “Spatial Layer” Polygons with very large surface area
  – “Spatial Layer” Polygons with very large number of vertices
  – More details on next two slides
Point In Polygon – Best Practices
This Polygon Has Both – Large Surface Area …. And Many Vertices

• Single “red” polygon
• MBR is yellow
• Many unnecessary comparisons when performing point in polygon
• Any point in yellow region compared to red polygon
• Affects performance

• Best Practice – Dice red polygon
Point In Polygon – Best Practices

- Dice red polygon
- Yellow boxes are MBR’s of diced regions
- Much faster point in polygon
- Use any dicing strategy
- Dicing can be done with Oracle Spatial and Graph (ie. with sdo_util.quad_tiles and sdo_intersection), or with other tools.
Point In Polygon – Insurance Company Example

• Requirement – **Compare 10,000 points to millions of polygons**
  – 2 polygons have > 100,000 vertices (anomalies)
  – 30 polygons have > 10000 vertices (anomalies)
• 10,000 point in polygon operations - results on next slide
Point In Polygon – Insurance Company Example

10,000 Point in Polygon Operations - Results

• Each line is a run of 10,000 point in polygons:
  – Exclude none - 1:31.06
  – Exclude > 100000 vertices (only two) - 25.09 seconds
  – Exclude > 10000 vertices (only thirty) - 14.31 seconds
  – Exclude > 5000 vertices – 10.13 seconds
  – Exclude > 2000 vertices – 5.45 seconds
  – Exclude > 1000 vertices – 3.62 Performance levels off here
  – Exclude > 500 vertices – 3.14
• Less than .05% of polygons had > 1000 vertices... those were diced
Large Linear Features – Best Practices
Large Linear Features – Rail and Contour Examples

• Red track 387 miles long
• 5094 vertices

• Red contour 2104 miles long
• 120,688 vertices
Large Linear Features – Rail and Contour Examples

Nearest Neighbor Requirement

- For each train GPS position, find the nearest track
- For each location, find the nearest contour line
Large Linear Features – Rail and Contour Examples

Increase Nearest Neighbor Performance

• Decompose each track into a set of two point lines

• Decompose each contour into a set of two point lines
Large Linear Features – Rail and Contour Examples

Increase Nearest Neighbor Performance

• Decompose each (track or contour) into a set of two point lines

  **TWO_POINT_LINE_TABLE**

<table>
<thead>
<tr>
<th>ID</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Y1</td>
<td>NUMBER</td>
</tr>
<tr>
<td>X2</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Y2</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

• Create a spatial function based index on \((x1,y1,x2,y2)\)

• Run spatial query (ie. nearest neighbor)
Contour Examples – Performance Results

Increase Nearest Neighbor Performance

- Nearest Neighbor query
  - Before splitting – 5.53 seconds each
  - After splitting – 0.05 seconds each
  - Over 110x performance gain

- Major performance gain, especially when issuing 1000’s of nearest neighbors

- Similar gains for rail queries (ie. Find nearest track to GPS position)
Functions That Return Geometries

Recommendations
Functions That Return SDO_GEOMETRY

Always declare as DETERMINISTIC

- Documented in the Database Object-Relational Developer’s Guide
- http://docs.oracle.com/database/121/ADOBJ/adobjplsql.htm#ADOBJ7181
- If a PL/SQL function is used in place of an ADT constructor during a DML operation, the function may execute multiple times as part of the DML execution. For the function to execute only once per occurrence, it must be a deterministic function.

```sql
CREATE FUNCTION get_geometry (in_x NUMBER, in_y NUMBER) RETURN SDO_GEOMETRY DETERMINISTIC AS
BEGIN
    RETURN sdo_geometry(2001, 4326, sdo_point_type(in_x, in_y, NULL), NULL, NULL);
END;
/
```
Functions That Return SDO_GEOMETRY

Never call a function that returns SDO_GEOMETRY in CREATE TABLE AS SELECT

• **This is fine:** CREATE TABLE tname AS SELECT sdo_geometry_column
• **This is not:** CREATE TABLE tname AS SELECT sdo_geom.sdo_intersection (...)

• Recommendation: Instead of CTAS with a function that returns SDO_GEOMETRY
  – First create the table structure with no rows
  – Then call **INSERT INTO table_name SELECT function_that_returns_sdo_geometry**
Parallel DML (PDML) and Parallel DDL (PDDL)
For Non Point SDO_GEOMETRY
PDML and PDDL – For Point Geometries

Supported Prior to 12c

• Parallel CREATE TABLE AS SELECT works prior to Oracle 12c when selecting point geometries.
• Parallel INSERT SELECT works prior to 12c when selecting point geometries.
• Both are very effective strategies.
Some Operations Work Prior to Oracle 12c

• The following will parallelize prior to 12c, even for non point geometries

```
INSERT /*+ APPEND PARALLEL(16) */
INTO geom_areas
NOLOGGING
SELECT geom_id, sdo_geom.sdo_area(geom, 0.05)
FROM table_with_many_rows;
```

• Works because sdo_geom.sdo_area returns a NUMBER
PDML and PIAS – With Non Point Geometries

New in 12.1.0.2

- Parallel DML for non point geometries

```
INSERT /*+ APPEND PARALLEL(16) */
INTO insertsection_results
NOLOGGING
SELECT b.id b_id, c.id c_id, sdo_geom.sdo_intersection(b.geom, c.geom, 0.05)
FROM rowid_pairs a,
    geom_table1 b,
    geom_table2 c
WHERE a.rowid1 = b.rowid
AND a.rowid2 = c.rowid;
```

- Oracle 12.1.0.2 patch required – Patch 23613512 (supercedes 21925655)
PDML and PDDL Support For SDO_GEOMETRY

• SDO_ELEM_INFO and SDO_ORDINATE varrays must be stored as SECUREFILE LOBS.

• If init.ora parameter db_securefile=PREFERRED or ALWAYS, SDO_GEOMETRY varrays should get created as securefile LOBS.

• To check:
  – SELECT table_name, column_name, securefile FROM user_lobs;
Parallel INSERT SELECT – A Few More Recommendations

• When SELECT contains a function that returns a geometry, set the following event:
  – ALTER SESSION set events '22821 trace name context forever, level 52';
  – May (or may not) be necessary in a future release.

• Also, don’t forget these:
  – ALTER SESSION enable parallel QUERY;
  – ALTER SESSION enable parallel DML;
  – ALTER SESSION enable parallel DDL; -- For CTAS
SDO_GEOMETRY and NOLOGGING
CTAS - NOLOGGING

• Sufficient to specify NOLOGGING on CTAS
• But for SDO_GEOMETRY DML, must explicitly set sdo ordinates and sdo elem info arrays to NOLOGGING (see next slide)
SDO_GEOMETRY DML – You Can Disable LOGGING

Reduce REDO considerably

-- Copy a table nologging
INSERT /*+ parallel (16) append */
INTO test_table
NOLOGGING
SELECT * FROM original_table;

• BUT…. for INSERT NOLOGGING, varray LOBS must be explicitly set to
  NOCACHE, NOLOGGING.
 ALTER TABLE test_table MODIFY LOB(geom.sdo_elem_info)(NOCACHE NOLOGGING);
 ALTER TABLE test_table MODIFY LOB(geom.sdo_ordinates)(NOCACHE NOLOGGING);
Datapump Import - NOLOGGING

• Alter nocahce and nologging parameters
  – alter table table_name modify lob(geom.sdo_elem_info) (nocache nologging);
  – alter table table_name modify lob(geom.sdo_ordinates) (nocache nologging);

• impdp parameters:
  – TABLE_EXISTS_ACTION=APPEND
  – ACCESS_METHOD=EXTERNAL_TABLE
  – TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y
Order By Linear Key
Order By Linear Key

• Most effective on tables with many rows
• Especially if table has dense clusters of spatial data, for example, customer locations in big cities
• Can dramatically increase performance for spatial operators:
  – SDO_NN with SDO_BATCH_SIZE
  – SDO_WITHIN_DISTANCE
  – SDO_ANYINTERACT, or any of the other operator masks
Order By Linear Key

• Recently ran SDO_ANYINTERACT against 170 million row table
• Returned 442,442 rows
• Not ordered by linear key
  – 35 minutes 12 seconds
• Ordered by linear key
  – 1.49 seconds (over 1400x faster)
• Slower query was also run on Oracle 11.2.0.3
  – Linear key was not the only factor, but it help considerably.
  – Faster query was run on Oracle 12.1.0.2 with SPATIAL_VECTOR_ACCELERATION=TRUE
Order By Linear Key

Why is it effective?

• The R-Tree index stores geometry approximations (mbr’s), and clusters them by proximity.

• Geometries whose MBR’s are close to each other are clustered in the database blocks populated by the R-Tree index.

• But this is not likely the case for the geometries stored in the base table, along with other attribute columns.

• When searching the R-Tree index:
  – groups of geometry MBR’s are selected to satisfy a spatial query
  – Then ROWID pointers access geometries (and their attributes) in the base table.
  – Too many database blocks are accessed, because the geometries are scattered (not clustered) in the base table.
Order Spatial Data By Linear Key – How To

• Blog post with details here:
  – [https://blogs.oracle.com/oraclespatial/entry/tips_on_tuning_sdo_nn](https://blogs.oracle.com/oraclespatial/entry/tips_on_tuning_sdo_nn)

• For those familiar with blog post, the appendix of this presentation contains a source code example that expands on the blog post example.

• The order by linear key source code example supplement of this presentation demonstrates how to handle null geometries, and also geometries that fall outside of the defined coordinate system bounds.
SDO_NN – Nearest Neighbor Best Practices
Nearest Neighbor – SDO_NN

SDO_NUM_RES or SDO_BATCH_SIZE

• 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?
Nearest Neighbor – SDO_NN

• **SDO_NUM_RES**
  • When only proximity is considered (closest bank example, next slide)

• **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.
SDO_NN Example – With SDO_NUM_RES

• 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

Additional Predicates From Same Table as SDO_NN search column

- Find the 5 closest banks named Citibank
- A column besides the SDO_GEOMERY column (ie. bank_name) in the BANK_TABLE is considered, use SDO_BATCH_SIZE.
- Next slide demonstrates the recommended way to write SDO_NN queries with SDO_BATCH_SIZE.
- Use next slide as a template for SDO_NN with SDO_BATCH_SIZE.
• Find the 5 closest banks with name Citibank

```sql
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 5 closest banks, but none more than 2 miles away

```
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';
```
SDO_NN with SDO_BATCH_SIZE

Order By Linear Key

• Batches with respect to distance are retrieved, and compared to predicates in WHERE clause.
• More database reads if batches are spread across may database blocks.
• When table is organized by distance in database blocks, retrieving blocks for next batch also contains attributes to compare in WHERE clause.
• Much fewer database blocks reads.
SDO_NN with SDO_BATCH_SIZE

Order By Linear Key

• Blog post with details here:
  – https://blogs.oracle.com/oraclespatial/entry/tips_on_tuning_sdo_nn
• Also see “Order By Linear Key” section in this presentation
• Give it a try
Server Side Parallel Enabled Clustering
Server Side Parallel Enabled Clustering

• Cluster millions of rows in seconds (server side)
• Returns cluster center point and count
• Effective for Auto Clustering in mapping applications
• Especially when too many rows to cluster client side
• Works in 12.1.0.2
• SQL example provided with this presentation
Spatial Query Over DBLINK
DBLINK Limitation With Spatial Operators

• You can select SDO_GEOMETRY over a DBLINK

• For Example:
  – In USER1, create tables POINT_TABLE and POLYGON_TABLE, and spatial indexes
  – In USER2, create a DBLINK that connects to USER1
  – These all work:
    • SELECT count(*) FROM point_table@user1_dblink;
    • SELECT count(*) FROM polygon_table@user1_dblink;
    • SELECT * FROM point_table@user1_dblink;
    • SELECT * FROM polygon_table@user1_dblink;
DBLINK Limitation With Spatial Operators

You can’t execute a spatial operator on a remote table

• ORA-13226: interface not supported without a spatial index

• For Example:
  – In USER1, create tables POINT_TABLE and POLYGON_TABLE, and spatial indexes
  – In USER2, create a DBLINK that connects to USER1
  – These all fail
    • SELECT count(*) FROM point_table@user1_dblink WHERE sdo_anyinteract (…) = ‘TRUE’
    • SELECT count(*) FROM polygon_table@user1_dblink WHERE sdo_anyinteract (…) = ‘TRUE’;
    • SELECT * FROM point_table@user1_dblink WHERE sdo_anyinteract (…) = ‘TRUE’;
    • SELECT * FROM polygon_table@user1_dblink WHERE sdo_anyinteract (…) = ‘TRUE’;
DBLINK With Spatial Operator – Workaround – Step 1

• In the remote database, create a type that contains the columns to return over the dblink.

• For example:

```sql
CREATE OR REPLACE TYPE one_point_type AS OBJECT
(  id NUMBER,
  x NUMBER,
  y NUMBER
);
/
```

```sql
CREATE OR REPLACE TYPE one_polygon_type AS OBJECT
(  id NUMBER,
  polygon_wkt VARCHAR2(32767)
);
/
```

NOTE: If wkt is larger than 32767, split it up into multiple fields
DBLINK With Spatial Operator – Workaround – Step 2

• On remote database, create remote procedures to set a query window.

• Remote procedure examples:

```sql
PROCEDURE set_box (in_tname VARCHAR2,
                   in_lower_x NUMBER, in_lower_y NUMBER,
                   in_upper_x NUMBER, in_upper_y NUMBER)

PROCEDURE set_circle (in_tname VARCHAR2,
                      in_center_x NUMBER, in_center_y NUMBER, in_radius NUMBER)

PROCEDURE set_polygon (in_tname VARCHAR2,
                       in_polygon_wkt VARCHAR2)
```

• On remote database, query windows are stored as global variables in a PL/SQL package.
DBLINK With Spatial Operator – Workaround – Step 3

• Create remote pipelined table function that executes a spatial query
• For example:

  FUNCTION get_box_window_results RETURN points_result_type
  DETERMINISTIC PIPELINED PARALLEL_ENABLE

  FUNCTION get_circle_window_results RETURN polygons_result_type
  DETERMINISTIC PIPELINED PARALLEL_ENABLE

  FUNCTION get_polygon_window_results RETURN points_result_type
  DETERMINISTIC PIPELINED PARALLEL_ENABLE

• Remote functions can contain SDO_ANYINTERACT, SDO_NN, etc…
DBLINK With Spatial Operator – Workaround – Step 4

Here Is The Magic

• On remote database, create view that converts the “object type” returned by the pipelined function to simple data types

• For example:

  Remember, function get_box_window_results() returns a TABLE of POINTS_RESULT_TYPE

```sql
CREATE OR REPLACE VIEW get_box_window_results_view AS
SELECT id, x, y FROM TABLE (get_box_window_results());
```

• From user2, access the remote view that runs the remote spatial query

```sql
EXECUTE global_vals.set_box@user1_dblink ('REMOTE_TABLE', 0,0,1,1);
SELECT id, x, y FROM get_box_window_results_view@user1_dblink;
```
Workspace Manager Queries
Workspace Manager Basics

- `dbms_wm.enable_versioning`
  - Renames `ORIGINAL_TABLE_NAME` to `TABLE_LT`
  - Adds columns to `TABLE_LT` that are managed by Workspace Manager versioning
  - Creates a view with the same name as the `ORIGINAL_TABLE_NAME` by only selecting the original table’s columns from `TABLE_LT`. All SQL that worked before the table was version enabled works after it is version enabled.
  - Instead of triggers are built against the `ORIGINAL_TABLE_NAME` view (for DML)

- Potentially, many versions of a geometry exist in `TABLE_LT`, but you only see the version relevant to your current workspace.
Workspace Manager - Optimization

• Imagine neighborhood with ID 170 was versioned 97 times
• Instead of this:

```sql
SELECT count(*)
FROM zoning_polygons a, neighborhoods qw
WHERE sdo_anyinteract (a.geometry, qw.geometry) = 'TRUE'
  AND qw.id = 170;
```
Workspace Manager - Optimization
When query window is versioned many times

• Try this (query subfactoring) style query:

```sql
WITH
  qw AS (SELECT /*+ materialize */ geometry
          FROM neighborhoods
          WHERE id = 170)
SELECT count(*)
FROM zoning_polygons a, qw
WHERE sdo_anyinteract (a.geometry, qw.geometry) = 'TRUE';
```

• Forces just one query window. **0.12 instead of 8.5 seconds (70x gain)**
Parallel Spatial Operators And Functions In Database Parallel Processing (Key Differentiator)
Parallel Query and Spatial

US Rail Application
Parallel Query And Spatial Operators

US Rail Application

• Requirement
  – GPS locations for each train collected throughout the day
  – Each location has other attributes (time, speed, and more)
  – GPS locations have a degree of error, so they don’t always fall on a track.
  – Bulk nearest neighbor queries to find closest track, and project reported train positions onto tracks

• This information is used for:
  – Tracking trains
  – Analysis for maintenance, ensure engineers are within parameters, etc
Parallel Query And Spatial Operators

What we tested

- 45,158,800 GPS train positions.
- For each train position:
  - Find the closest track to the train (with SDO_NN)
  - Then calculate the position on the track closest to the train
CREATE TABLE results PARALLEL 72 NOLOGGING AS
SELECT /*+ ordered index (b tracks_spatial_idx) */
    a.locomotive_id,
    sdo_lrs.find_measure (b.track_geom, a.locomotive_pos)
FROM locomotives a,
    tracks b
WHERE sdo_nn (b.track_geom, a.locomotive_pos,
    'sdo_num_res=1') = 'TRUE';
Parallel Query And Spatial Operators

Exadata Results

• On older Exadata (X2-2 Half RAC):
  • 34.75 hours serially vs. 44.1 minutes in parallel
  • 48 database cores - 47x faster
• X6-2 even faster – Easily exceed 100x faster
Parallel Enabled Geocoding
Oracle Spatial & Graph - Geocoder

- Geocoder is included in your Oracle Spatial and Graph license.
- Open data model for Geocoder reference data
- If you have reference data, you can populate the data model yourself
- If you don’t have the reference data, Oracle Partners sell it in Transportable Tablespace format (plug and play data).
  - HERE
  - Tom Tom
  - ADCI
  - others
Oracle Spatial and Graph Geocoder

• Forward / Reverse / Street Centerline / Rooftop (point based) support

• In database geocoding –
  – PL/SQL APIs
  – Optimal for parallel enabled batch geocoding
  – For batch processing, leverage parallel enabled pipeline table functions

• Web service based geocoding
  – Java servlet based with XML geocoding APIs
  – Deployed in J2EE container
  – Optimal for non-batch request in web based applications.
  – Can perform batch processing too
Geocode Times On Exadata X4-2 1/2 RAC

• X4-2 with 96 cores
• Geocoded 77216 addresses in 3.32 seconds
• 23,257 geocodes per second
Oracle 12.1.0.2 – Patch Recommendations
Oracle 12.1.0.2 – Patch Recommendations

• Patch recommendations in an ideal world

• I say ideal world, because even though the patches in the next few slides are backported to Oracle 12.1.0.2, they may not be available on your specific platform or your Oracle 12.1.0.2 dot release

• If the patch is not on support.oracle.com, please file an SR for a backport

• For some of the patches listed, I mention the patch(es) it supersedes. If only the superseded patch (the one with some of the fixes) is on support.oracle.com, you can use it while the comprehensive patch is being generated by your service request.
Oracle 12.1.0.2 – Patch Recommendations - Continued

• Patche recommendation are split into two groups:
  – RDBMS specific patches
  – Spatial specific patchs
Oracle 12.1.0.2 – Patch Recommendations - Continued

• **Spatial Specific Patches**
  
  – 21453611 – Spatial index statistics patch bundle (many improvements)
  
  – 24616772 - 12.1.0.2 spatial patch bundle - Also includes faster sdo_intersection with spatial_vector_acceleration=true (supersedes 22079444 or 21376696)
  
  – 22246206 - Improve mechanism for spatial feature check in 12c
  
  – 22300040 - ALL_SDO_GEOM_METADATA- Use ALL_CATALOG instead of ALL_OBJECTS in view definition
  
  – 25107080 - Spatial index creation is doing full table scan in parallel when no parallel is defined on 12c
Oracle 12.1.0.2 – Patch Recommendations - Continued

• **RDBMS Specific Patches**
  - 23613512 - Parallel CTAS and INSERT SELECT for non-point SDO_GEOMETRY (supersedes 21925655)
  - 8617254 - UNION ALL support, also enables spatial queries against Esri Multiversion Views
  - 22652097 and 21171382 - Per Doc 2187449.1... New 12.1 recommendations for optimizer_adaptive_features (see next slide for more details)
Some excerpts from Doc 2187449.1 (doc mentioned on previous slide)

- In 12.2, the parameter `optimizer_adaptive_features` has been obsoleted.
- The adaptive features are controlled by two new parameters, `optimizer_adaptive_plans` and `optimizer_adaptive_statistics`.
- The `optimizer_adaptive_plans` parameter controls whether the optimizer creates adaptive plans and defaults to TRUE. When `optimizer_features_enable` is set to 12.1.0.1 or higher, all features controlled by `optimizer_adaptive_plans` are enabled.
- The `optimizer_adaptive_statistics` parameter controls whether the optimizer uses adaptive statistics and defaults to FALSE. These defaults have been chosen to place emphasis on achieving stable SQL execution plans.
- We recommend that upgrades to 12.1 adopt the 12.2 defaults. This may be done by applying the following patches 22652097 and 21171382.
Questions and Answers