Oracle Spatial & Graph SIG

ORACLE SPATIAL SUMMIT 2016

January 26–28, 2016

With BIWA 2016: The Oracle Big Data + Analytics Conference

# Best Practices, Tips and Tricks With Oracle Spatial and Graph


## Daniel Geringer, Spatial Solutions Specialist
## Oracle Corporation


ORACLE

# Spatial Vector Acceleration

# SPATIAL_VECTOR_ACCELERATION

## Oracle 12c Initialization Parameter

- **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

# 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 …

# Multiple Point In Polygon Operations

# Multiple Point In Polygon Requirement

## Insurance Company Requirement

- For every location, perform point in polygon operations against several risk zone layers:
  - Flood zones
  - Earthquake zones
  - Volcanic zones
  - Hurricane zones
  - Wildfire zones

# Multiple Point In Polygon Requirement

- Effective approach when each risk zone query returns one row.
- Run the point in polygon queries for different hazards zones, in parallel (in a single SQL statement)
- Parallelizes extremely well
- Query is on next slide

# Multiple Point In Polygon Requirement – Query Example

```
CREATE TABLE results PARALLEL 16  NOLOGGING AS
SELECT  (SELECT hz.rowid
         FROM hail_zone hz
         WHERE SDO_ANYINTERACT(hz.SHAPE, SDO_GEOMETRY(2001, 4326,
                                                      SDO_POINT_TYPE(ca.longitude,ca.latitude,NULL),
                                                      NULL, NULL)) ='TRUE') as r1,
        (SELECT fz.rowid
         FROM flood_zone fz
         WHERE SDO_ANYINTERACT(fz.SHAPE, SDO_GEOMETRY(2001, 4326,
                                                      SDO_POINT_TYPE(ca.longitude,ca.latitude,NULL),
                                                      NULL, NULL)) ='TRUE') as r2,
        (SELECT ez.rowid
         FROM earthquake_zone ez
         WHERE SDO_ANYINTERACT(ez.SHAPE, SDO_GEOMETRY(2001, 4326,
                                                      SDO_POINT_TYPE(ca.longitude,ca.latitude,NULL),
                                                      NULL, NULL)) ='TRUE') as r3
FROM customer_addresses  ca;
```

# 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 go from
- 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 eachother 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
- 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.

# SDO_NN Example – With SDO_BATCH_SIZE

## Template for SDO_NN with SDO_BATCH_SIZE

- Find the 5 closest banks with name Citibank

```
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

# 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:

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

```
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:

```
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_results RETURN points_result_type
DETERMINISTIC  PIPELINED  PARALLEL_ENABLE

FUNCTION get_circle_results RETURN polygons_result_type
DETERMINISTIC  PIPELINED  PARALLEL_ENABLE

FUNCTION get_polygon_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_results() returns a TABLE of POINTS_RESULT_TYPE

```
CREATE OR REPLACE VIEW get_box_results_view AS
   SELECT id, x, y FROM TABLE (get_box_results());
```

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

```
EXECUTE global_vals.set_box@user1_dblink (‘REMOTE_TABLE', 0,0,1,1);
```

```
SELECT id, x, y FROM get_box_results_view@user1_dblink;
```

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

# Workspace Manager – Optimize Spatial Queries

## When query window is versioned many times

- Imaging neighborhood with ID 170 was versioned 97 times
- Instead of this:

```
SELECT count(*)
FROM zoning_polygons a, neighborhoods qw
WHERE sdo_anyinteract (a.geometry, qw.geometry) = 'TRUE'
    AND qw.id = 170;
```

# Workspace Manager – Optimize Spatial Queries

## When query window is versioned many times

- Try this (query subfactoring) style query:

```
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)

ORACLE

# Some Patch Recommendations

# Oracle 12.1.0.2 – Some Patch Recommendations

- Available on support.oracle.com
- Oracle Database 12.1.0.2 –
  - 21453611 - Performance and other fixes for spatial statistics generation (supersedes patches 19544707, 20653340, 20808043)
  - 21376696 - 12.1.0.2 spatial patch bundle - Also includes faster sdo_intersection with spatial_vector_acceleration=true
- Oracle Database 12.1.0.1 or 11.2.0.4
  - 21453611 - Performance and other fixes for spatial statistic generations (supersedes patches 19544707, 20653340, 20808043)

# Questions and Answers

# The Spatial and Graph SIG


- The SIG promotes interaction and communication that can drive the market for spatial technology and data
- Members connect and exchange knowledge via online communities and at annual conferences and events

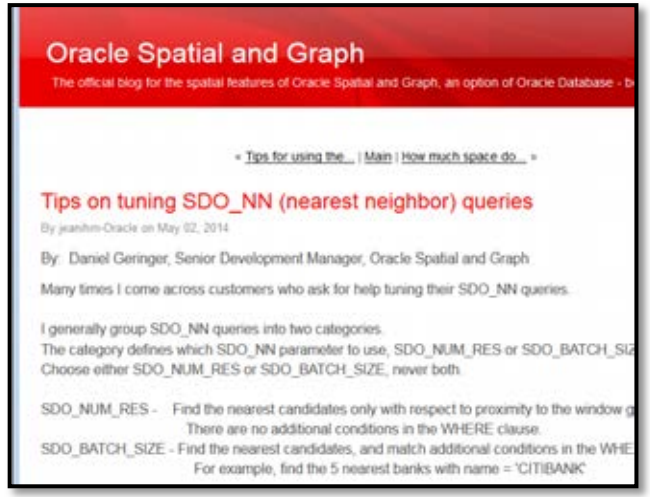
- Meet us at the Summit
  - Morning Receptions
    - Tuesday and Wednesday / 7:45 to 8:30 a.m. / Registration Area
  - Birds of a Feather Session
    - Tuesday / 12 to 1 p.m. / Lunch Room
- Join us online
  - LinkedIn (search for "LinkedIn Oracle Spatial")
  - Google+ (search for "Google+ Oracle Spatial")
  - IOUG SIG (sign up for free membership through www.ioug.org)
  - OTN Spatial – Communities (search for "Oracle Spatial and Graph Community")
- Contact the Board at oraclespatialsig@gmail.com

ORACLE

# Resources

- Oracle Technology Network:
  www.oracle.com/technetwork/database/options/spatialandgraph
  www.oracle.com/database/big-data-spatial-and-graph
- blogs.oracle.com ➔oraclespatial ➔ oracle_maps_blog ➔bigdataspatialgraph





# Resources on Big Data Spatial and Graph

- Oracle Big Data Spatial and Graph on Oracle.com: https://www.oracle.com/database/big-data-spatial-and-graph
- OTN product page (trial software downloads, documentation): http://www.oracle.com/technetwork/database/database-technologies/bigdata-spatialandgraph
- Blog (technical examples and tips): https://blogs.oracle.com/bigdataspatialgraph/
- Big Data Lite Virtual Machine (a free sandbox environment to get started): http://www.oracle.com/technetwork/database/bigdata-appliance/oracle-bigdatalite-2104726.html