Managing Point Clouds in Oracle Databases

NCG/OGh Point Cloud Seminar
December 8 2015

Albert Godfrind
Oracle Corporation

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Overview

- Some Background …
- Oracle **Data Models** for Point Clouds
  - Blocked (R-tree, Hilbert), flat, hybrid
- **Loading** Point Clouds
  - Create, load, block, pyramid
- **Processing** Point Clouds
  - Clip and filter, nearest neighbor, contours
- Conclusions

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Some Background …

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Requirements for Managing 3D Point Data at National Level

Screenshot courtesy of Rico Richter, HPI

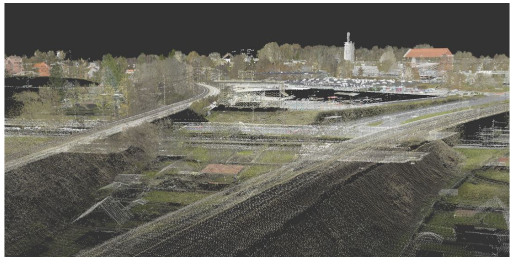


Image courtesy of: IQsoft, Austria

Large user community

Large data sets

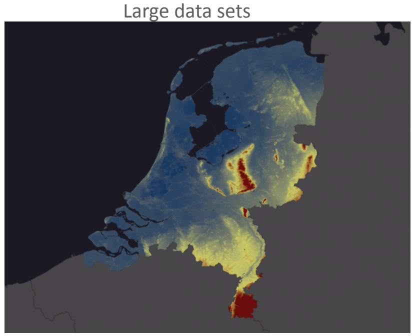


Image courtesy of: PDOK, NL

LIDAR

City Models

Integration of large data sets from diverse sources & formats

Open Standards

Spatial DB

Publish to Web Services

Data intensive processing

Screenshot courtesy of Rico Richter, HPI

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Example Dataset:
Actual Height Model of the Netherlands (AHN2)

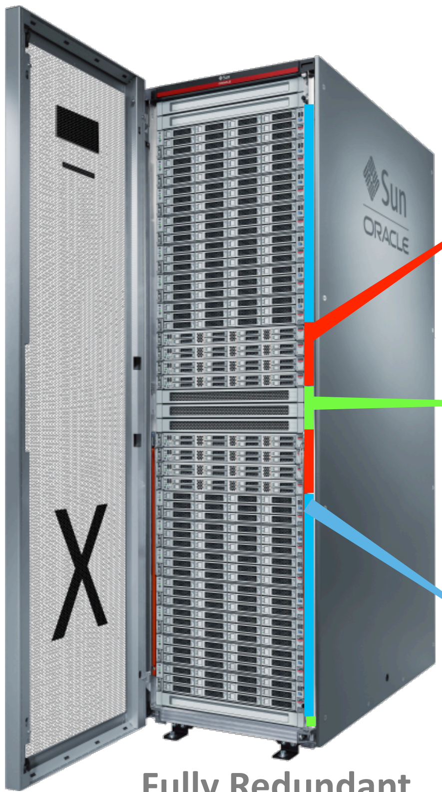
- Covering surface of the entire country
- 6 -10 pts/m^2 → 640 billion pts
- 60,185 LAZ files, 987 GB in total, 11.64 TB uncompressed
- (X, Y, Z) only
- Future plans
  - AHN3 at even higher resolution
  - Cyclorama-based photogrammetric datasets (50x AHN2, and with RGB)

Tested with Peter v. Oosterom, TU Delft, NLeSC (Netherlands eScience Center), Fugro

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Introducing Oracle **Exadata** Database Machine



Fully Redundant

- **Standard Database Servers**
  - 8x 2-socket servers → 192 cores, 2TB DRAM
    or
  - 2x 8-socket servers → 160 cores, 4TB DRAM

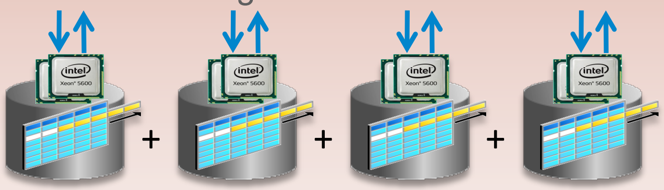
- **Unified Ultra-Fast Network**
  - 40 Gb InfiniBand internal connectivity
  - 10 Gb or 1 Gb Ethernet data center connectivity
- **Scale-out Intelligent Storage Servers**
  - 14x 2-socket servers → 168 cores in storage
  - 168 SAS disk drives → 672 TB HC or 200 TB HP
  - 56 Flash PCI cards → **44** TB Flash + compression

ORACLE

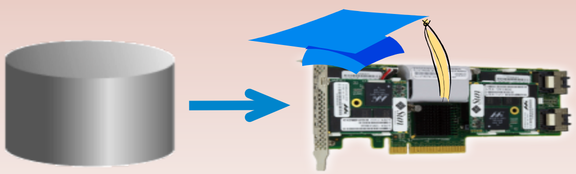
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Key Exadata Innovations

- **Intelligent storage**
  - Smart Scan query offload
  - Scale-out storage

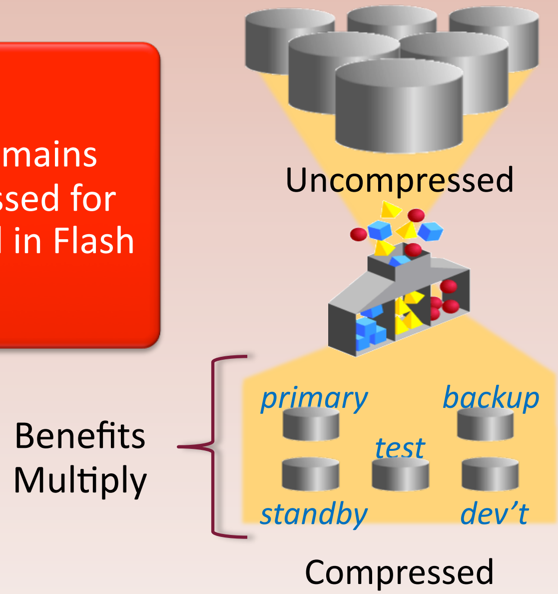


- **Smart Flash Cache**
  - Accelerates random I/O up to 30x
  - Doubles data scan rate



- **Hybrid Columnar Compression**
  - 10x compression for warehouses
  - 15x compression for archives

Data remains compressed for scans and in Flash



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Start Small and Grow

Eighth Rack
24 cores

Quarter Rack
48 cores

Half Rack
96 cores

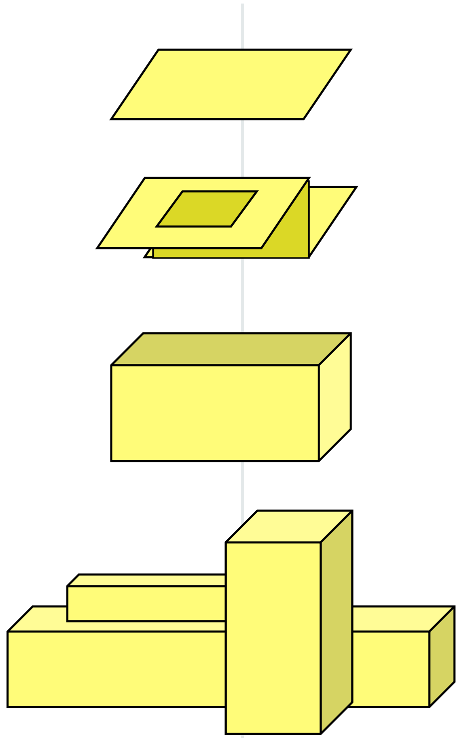
Full Rack
192 cores

ORACLE

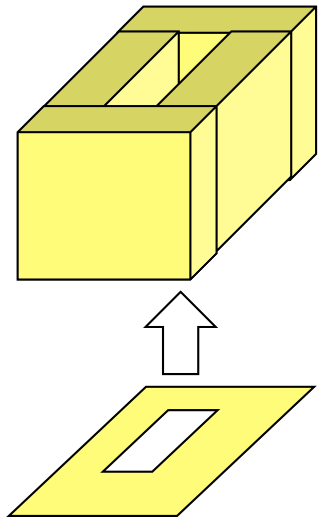
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Managing 3D objects in databases

- Simple Surfaces
  - Face = 3D Polygon
- Composite Surface
  - Multiple connected faces
- Simple Solid
  - Closed composite surface
- Composite Solid
  - Multiple connected simple solids



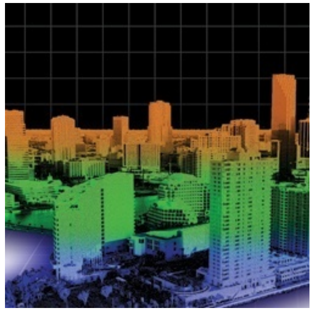
- Extrusion
  - Generating solids from 2D polygons

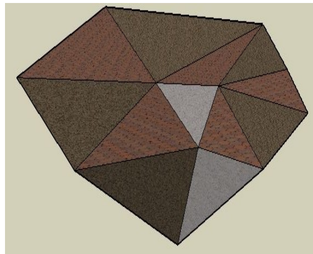


ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Managing 3D objects in databases



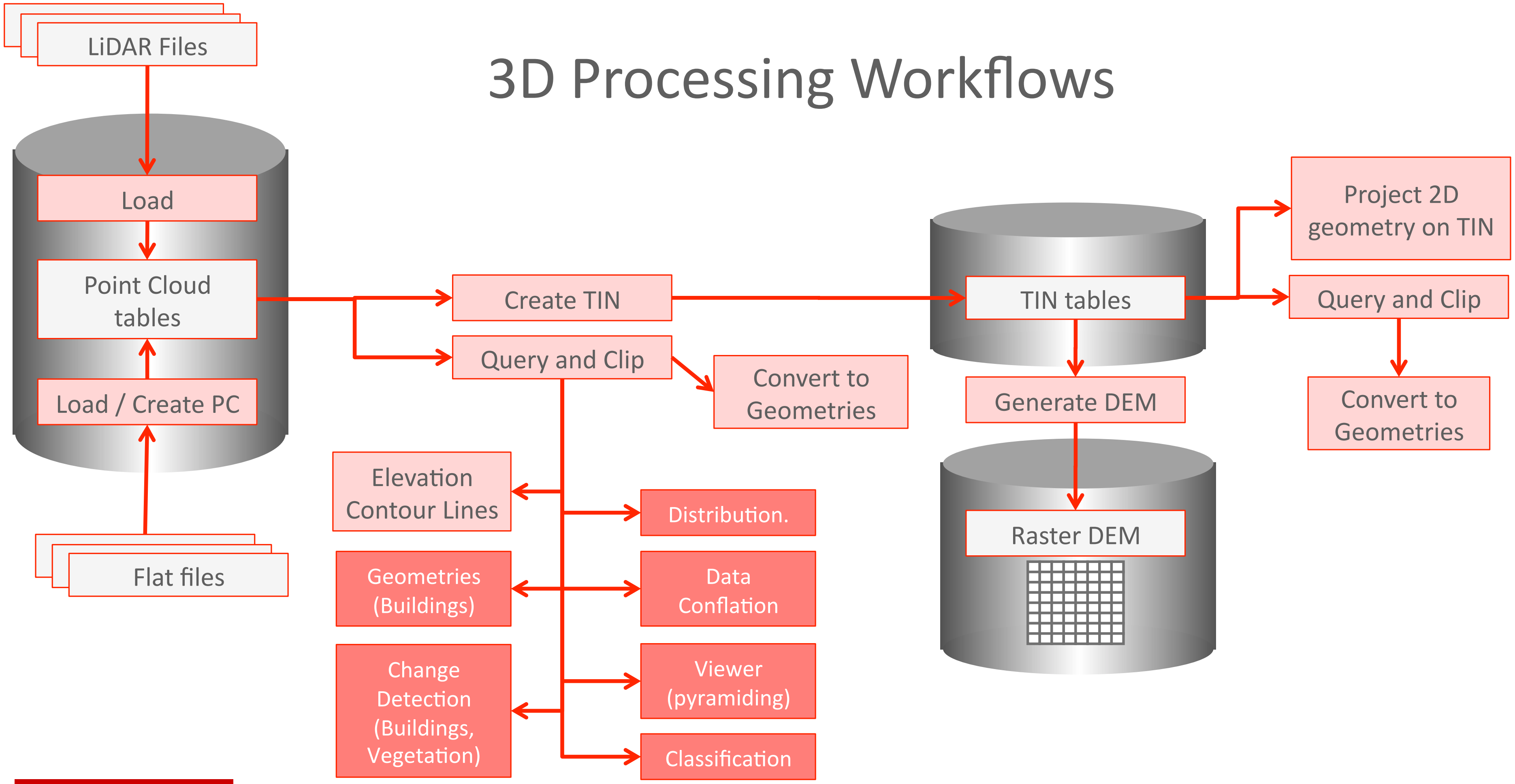


- Other 3D Data types:
  - Point Clouds
  - Triangulated Irregular Networks (TINs)
  - Geo-referenced gridded data (DEMs)
- Storage Model
  - One logical object, containing all attributes and metadata
  - Many physical storage blocks which can be managed individually
- Examples of in-database processing
  - Pyramiding for efficient visualization
  - Point cloud to contour line conversion

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

3D Processing Workflows



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

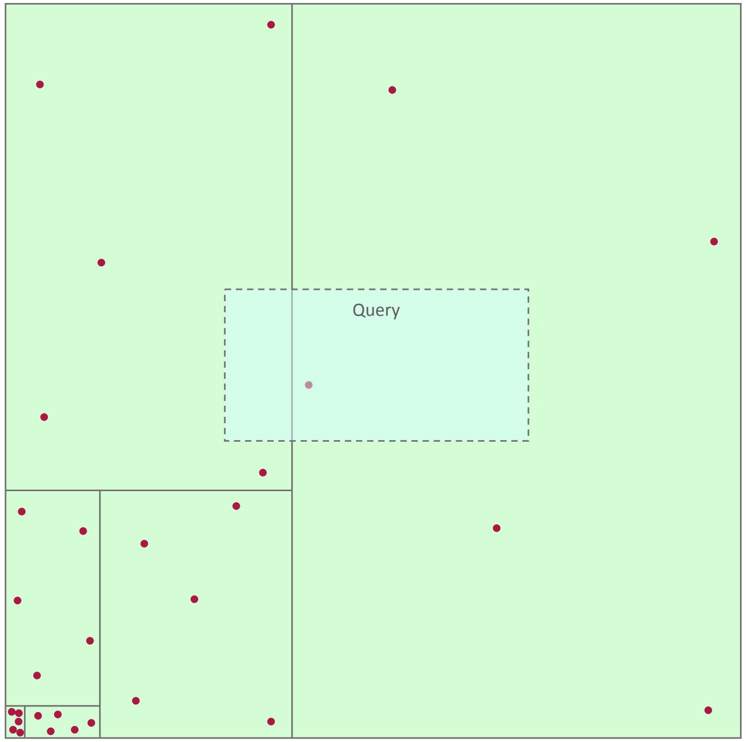
Oracle Point Cloud Models

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

How To Manage Large Point Sets in Databases ?

Principle: Spatial Partitioning

- Create:
  - Spatially partition the points into “blocks”
- Query:
  - Find candidate blocks
  - Test points within candidates
  - Very scalable



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Oracle Storage Models

- **Blocked** (Spatial Partitioning & BLOB encoding)
  - R-tree
  - Hilbert R-tree

  ... However: partitioning expensive; is it even required w/ query offload?
- **Flat** (Non-Spatial Partitioning on flat tables)
  - B-tree
  - Exadata: no index / query offload / compression
  - Range Partitioning

  ... However: blocked model scaled better, on non-Exadata
- **Hybrid** (Spatial Partitioning via Index Organized Table)
  - Hilbert R-tree

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Pro's and Con's of each Model

Storage model	Pro	Con
Blocked	• Storage (compression) • Scaling • Indexing • DB functionalities • Complex queries	• Loading (create blocks) • Block overhead in queries (noticeable in simple queries)
Flat	• Faster loading • DB functionalities • Dynamic schema (→ blocked) • Simple queries	• Storage (except Exadata) • Limits to scaling (except Exadata) • Indexing (except Exadata)
Hybrid	•Faster queries (→ blocked) •More scalable queries (→ flat) •Dynamic schema (→ blocked)	• No compression (no HCC with IOT)

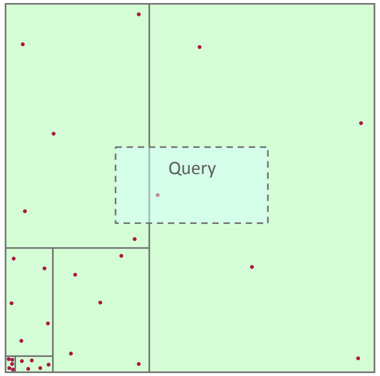
ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

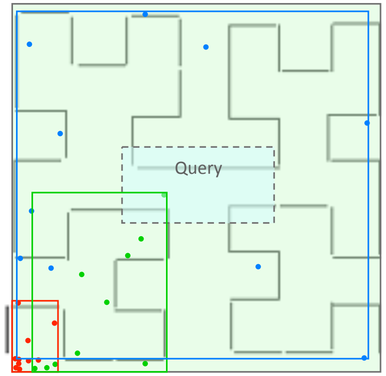
The “**Blocked**” Model

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Use **Spatial** Partitioning with binary **blocks**



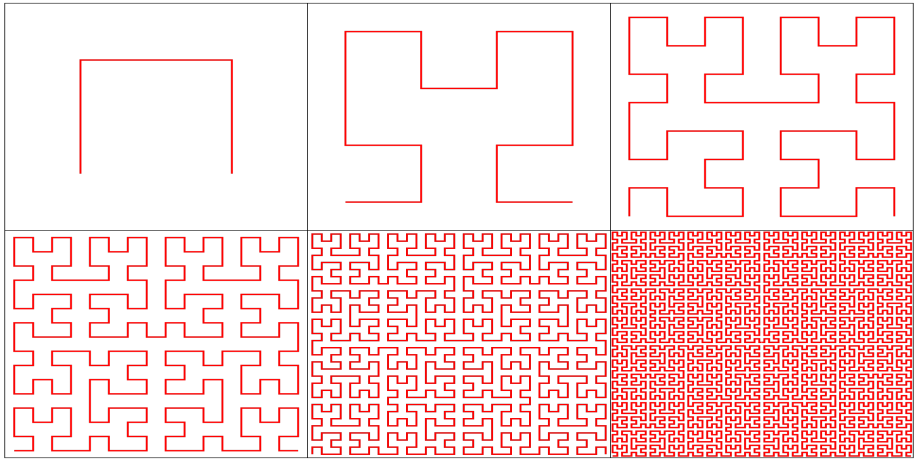
RTREE



HILBERT

BLOB Encoding

- 64bit IEEE double (for each dimension)
- 4-byte big-endian int (Block ID)
- 4-byte big-endian int (Pt ID)



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Object-based Storage Model: the **SDO_PC** data type

Logical structures

Contains point cloud metadata and footprint

Also contains a pointers to one or more block tables

Physical structures

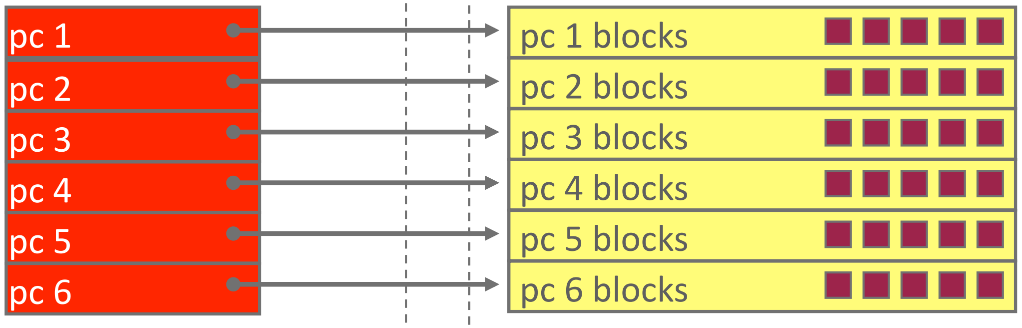
Point cloud block tables

Contain the points

Can be very large

Could be partitioned

Add new tables as

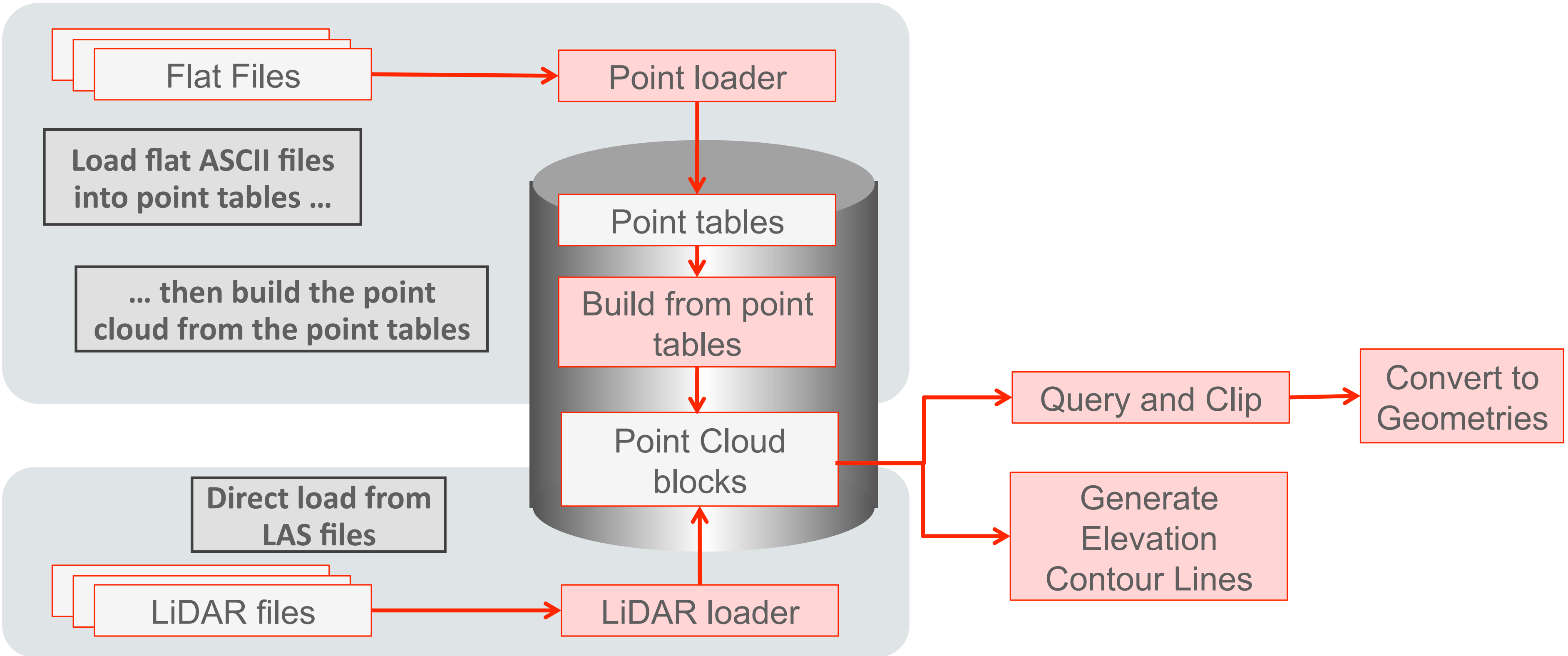



```
CREATE TABLE point_clouds (
  id          NUMBER,
  capture_ts  TIMESTAMP,
  point_cloud SDO_PC
);
```

```
CREATE TABLE pc_blocks
OF SDO_PC_BLK (
  PRIMARY KEY (
     obj_id, blk_id
  )
);
```

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

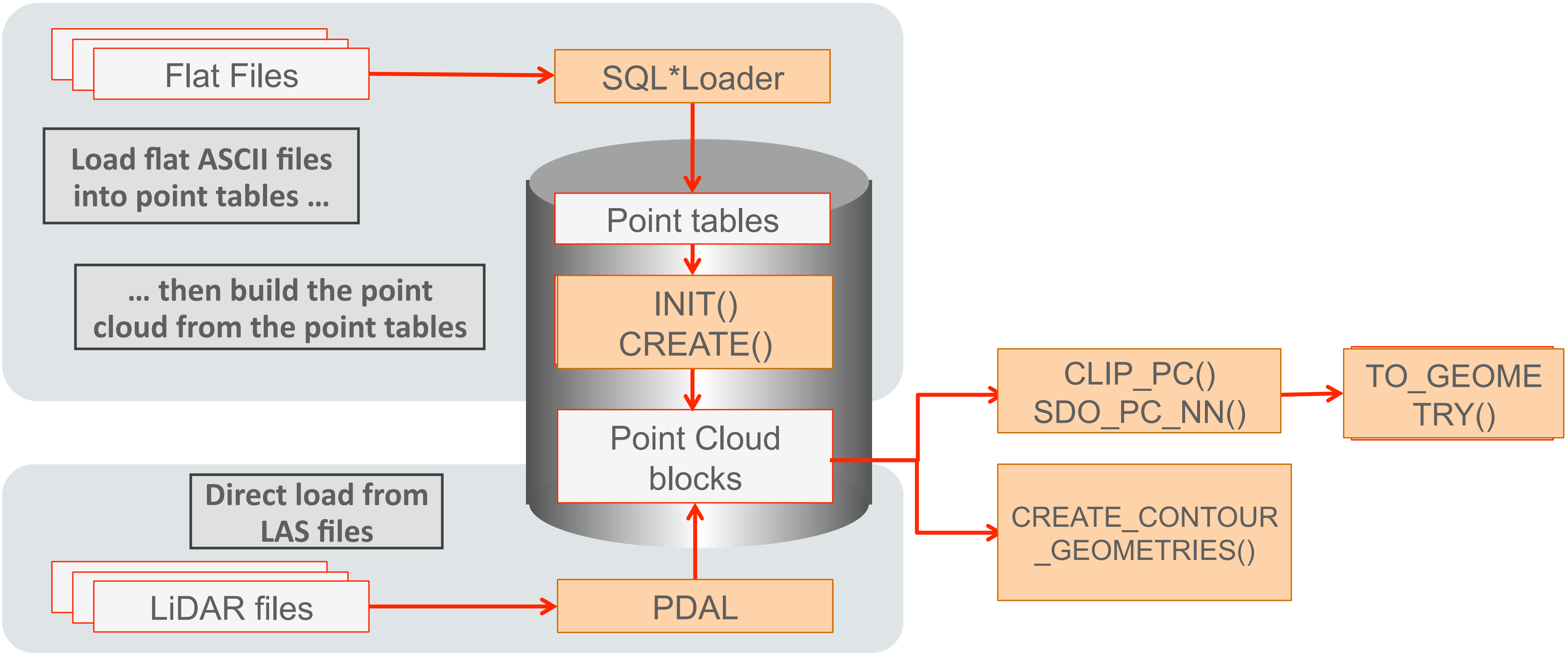
Workflow for **Blocked** Model



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Workflow for **Blocked** Model



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Loading (1/3): Load the Flat File into a Point Table

- Input is a csv text file
- Load using SQL*Loader

```
CREATE TABLE input_points (
  rid             VARCHAR2(40),
  val_d1          NUMBER,
  val_d2          NUMBER,
  val_d3          NUMBER
)
```

```
sqlldr scott/tiger
  control=point_table.ctl
  data=input_points.dat
  direct=true
  columnarrayrows=20000
  streamsize=2560000
  multithreading=true
```

```
LOAD DATA
 TRUNCATE
 INTO TABLE input_points
 FIELDS TERMINATED BY ',' (
   rid sequence,
   val_d1,
   val_d2,
   val_d3
)
```

```
-73.999922, 40.000002, 74
-73.999921, 40.000002, 27
-73.999920, 40.000002, 76
-73.999919, 40.000002, 72
-73.999918, 40.000002, 91
-73.999917, 40.000002, 96
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Loading (2/3): Initialize the Point Cloud

- Define the **structure** and **organization** of the point cloud
  - Resolution, dimensions, extent
  - Block capacity
- Specify the **location** of the blocks for each point cloud
  - Name of the point blocks table
- The unique identifier of the point cloud is automatically generated.
- Use default spatial partitioning (RTREE)

```
INSERT INTO pcs (ID, POINT_CLOUD)
VALUES (
  1001,
  sdo_pc_pkg.init(
    basetable          => 'PCS',
    basecol            => 'PC',
    blktable           => 'BLOCKS',
    ptn_params         =>
      'blk_capacity=10000',
    pc_extent          =>
      sdo_geometry(2003, null, null,
        sdo_elem_info_array(1,1003,3),
        sdo_ordinate_array(
          289020.90, 4320942.61,
          290106.02, 4323641.57
        )
    ),
    pc_tol             => 0.05,
    pc_tot_dimensions  => 3,
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Initialize the Point Cloud for **Hilbert** Spatial Partitioning

```
sdo_pc_pkg.init(
  basetable          => 'PCS',
  basecol            => 'PC',
  blktable           => 'BLOCKS',
  ptn_params         => 'blk_capacity=10000',
  pc_extent          =>
    sdo_geometry(2003, null, null,
      sdo_elem_info_array(1, 1003, 3),
      sdo_ordinate_array(289020.90, 4320942.61, 290106.02, 4323641.57)
    ),
    pc_tol             => 0.05,
    pc_tot_dimensions  => 3,
    pc_domain          => null,
    pc_val_attr_tables => null,
    pc_other_attrs     => xmltype(
      '<opc:sdoPcObjectMetadata
         xmlns:opc="http://xmlns.oracle.com/spatial/vis3d/2011/sdovis3d.xsd"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         blockingMethod="Hilbert R-tree">
       </opc:sdoPcObjectMetadata>')
);
```

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Loading (3/3): Create the Point Cloud from the Point Table

- **Read** the points from the point table
- **Spatially cluster** the points as specified during initialization (RTREE or HILBERT)
- Generate and fill the PC **blocks**
- Also creates the spatial index over the block extents.

```
DECLARE
  PC SDO_PC;
BEGIN
  SELECT POINT_CLOUD INTO PC
  FROM PCS WHERE ID = 1001;
  SDO_PC_PKG.CREATE_PC (
    PC,
    'INPUT_POINTS'
  );
END;
/
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Incremental Loading

```
$JAVA_HOME/bin/java -Xms2048m
 -classpath /…/ojdbc6.jar:/…/sdoutl.jar
  oracle.spatial.util.Las2SqlLdrIndep
 1
 BLOCKS
 BLOCK_ID_SEQ
 /…/ahn_bench023090_01.las
 10000
 jdbc:oracle:thin:@//host:port/service
 largepc
 largepc
 100000000
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

```
Connecting to jdbc:oracle:thin:@//host:port/sid...
Reading 100,000,000 points...   23.324 s. Sorting:   65.362 s. Writing:  164.608 s
Reading  99,995,084 points...   28.793 s. Sorting:   63.231 s. Writing:  187.621 s
Reading  99,997,274 points...   38.275 s. Sorting:  138.074 s. Writing:  216.841 s
Reading  99,991,996 points...   50.051 s. Sorting:  141.184 s. Writing:  236.634 s
Reading  99,990,636 points...   54.492 s. Sorting:  161.936 s. Writing:  240.436 s
Reading  99,990,636 points...   58.812 s. Sorting:  149.797 s. Writing:  244.515 s
Reading  99,990,637 points...   58.353 s. Sorting:  148.561 s. Writing:  242.785 s
Reading  99,990,638 points...   60.001 s. Sorting:  151.770 s. Writing:  242.187 s
Reading  99,990,639 points...   61.795 s. Sorting:  153.668 s. Writing:  243.461 s
Reading  99,990,640 points...   60.733 s. Sorting:  150.150 s. Writing:  243.077 s
Reading  99,990,641 points...   60.419 s. Sorting:  151.012 s. Writing:  244.065 s
Reading  99,999,834 points...   62.434 s. Sorting:  151.908 s. Writing:  246.490 s
Reading  99,990,124 points...   58.342 s. Sorting:  152.249 s. Writing:  245.693 s
Reading  99,999,835 points...   57.816 s. Sorting:  151.361 s. Writing:  244.788 s
Reading  99,990,145 points...   56.882 s. Sorting:  153.129 s. Writing:  245.248 s
Reading  99,997,246 points...   56.489 s. Sorting:  152.962 s. Writing:  244.226 s
Reading  99,992,320 points...   58.604 s. Sorting:  150.968 s. Writing:  253.848 s
Reading  99,990,606 points...   59.011 s. Sorting:  162.808 s. Writing:  253.697 s
Reading  99,990,607 points...   56.012 s. Sorting:  151.588 s. Writing:  255.625 s
Reading  99,990,608 points...   57.417 s. Sorting:  152.561 s. Writing:  256.142 s
Reading     139,854 points...    0.110 s. Sorting:    0.097 s. Writing:    0.375 s
---------------------------------------------------------------------------------
Read  2,000,000,000 points...1,078.165 s. Sorting:2,854.376 s. Writing:4,752.362 s
Time elapsed........................ 8,708.131 s
```

2 Billion points loaded in just about 2.5 hours @ 230,000 points/second

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: **CLIP_PC**

- Returns the **blocks** that match the query window
- Blocks only contain the **points** that match the window

```
select *
from table(
  sdo_pc_pkg.clip_pc(
    inp            => (select pc from pcs where id = 1),
    ind_dim_qry    => sdo_geometry(2003, null, null,
                        sdo_elem_info_array(1, 1003, 3),
                        sdo_ordinate_array(10, 10, 14, 14)
                      ),
     other_dim_qry => null,
     qry_min_res   => null,
     qry_max_res   => null
  )
);
```

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Return Individual **Points**

```
select query_points.x, query_points.y, query_points.z
from table (
        sdo_pc_pkg.clip_pc(
        inp           => (select pc from pcs where id = 1),
        ind_dim_qry   => sdo_geometry(2003, null, null,
                           sdo_elem_info_array(1, 1003, 3),
                           sdo_ordinate_array(10, 10, 14, 14)
                         ),
        other_dim_qry => null,
        qry_min_res   => null,
        qry_max_res   => null)
     ) query_blocks,
     table (
       sdo_util.getvertices(
         sdo_pc_pkg.to_geometry(
           pts         => query_blocks.points,
           num_pts     => query_blocks.num_points,
           pc_tot_dim  => 3,
           srid        => null
         )
       )
     ) query_points;
```

- Convert each block into a **SDO_GEOMETRY** object
- Extract points from that object into a stream **of X, Y, Z columns**

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Filter on additional dimensions (Z and Intensity)

```
select query_points.x, query_points.y, query_points.z, query_points.w intensity
from table(
        sdo_pc_pkg.clip_pc(
        inp            => (select pc from pcs where id = 1),
        ind_dim_qry    => sdo_geometry(2003, null, null,
                            sdo_elem_info_array(1, 1003, 3),
                            sdo_ordinate_array(10, 10, 14, 14)
                          ),
        other_dim_qry => sdo_mbr(
                            lower_left  => sdo_vpoint_type(123,    1),
                            upper_right => sdo_vpoint_type(123, 1000)),
        qry_min_res    => null,
        qry_max_res    => null
     ) query_blocks,
     table(
       sdo_util.getvertices(
         sdo_pc_pkg.to_geometry(
           pts         => query_blocks.points,
           num_pts     => query_blocks.num_points,
           pc_tot_dim => 4,
           srid        => null
         )
       )
     ) query_points;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Filter on Non-Blocked Dimensions

```
select query_points.x, query_points.y, query_points.z, query_points.w Intensity, -- followed by v5 - v11
  out.non_blocked_dim
from
  table(
    sdo_pc_pkg.clip_pc(
      inp           => (select pc from pcs where id = 1),
      ind_dim_qry   => sdo_geometry(
                         2003,
                         null,
                         null,
                         sdo_elem_info_array(1, 1003, 3),
                         sdo_ordinate_array(10, 10, 14, 14)),
      other_dim_qry => sdo_mbr(
                         lower_left  => sdo_vpoint_type(123,    1),
                         upper_right => sdo_vpoint_type(123, 1000)),
      qry_min_res   => 1,
      qry_max_res   => 1)) query_blocks,
  table(
    sdo_util.getvertices(
      geometry => sdo_pc_pkg.to_geometry(
                    pts         => query_blocks.points,
                    num_pts     => query_blocks.num_points,
                    pc_tot_dim => 4,
                    srid        => null ,
                    get_ids     => 1))) query_points ,
  pcs_out out
where
  out.ptn_id   = query_points.v5 and    -- v5 is blk_id, v6 is pt_id
  out.point_id = query_points.v6;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying in parallel (multiple windows): **CLIP_PC_PARALLEL**

```
with candidates AS (
  select blocks.blk_id, subqueries.ind_dim_qry, subqueries.other_dim_qry
  from
    pc_blocks blocks,
    (
      select 1 min_res, 1 max_res, :window ind_dim_qry, cast(null as sdo_mbr) other_dim_qry from dual
      union all
      select 2 min_res, 5 max_res, :window ind_dim_qry, cast(null as sdo_mbr) other_dim_qry from dual
    ) subqueries
  where
    blocks.obj_id = 1 and
    blocks.pcblk_min_res between min_res and max_res and
    SDO_ANYINTERACT(blocks.blk_extent, subqueries.ind_dim_qry) = 'TRUE'
)
select /*+ parallel(32) */ *
from table (
    sdo_pc_pkg.clip_pc_parallel(
      blocks => cursor(select * from candidates),
      inp    => (select pc from pcs where id = 1)
    )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Finding Nearest Points: **SDO_PC_NN**

```
select rownum pt_pos,
  sdo_geometry (3001, null, sdo_point_type(x, y, z), null, null) pts
from table (
  sdo_util.getvertices(
    sdo_pc_pkg.to_geometry(
      pts => sdo_pc_pkg.sdo_pc_nn(
               pc     => (select pc from pcs where pc_id = 1),
               center => sdo_geometry(3001, null,
                 sdo_point_type(15, 15, 30), null, null),
               n      => 3200
             ),
      num_pts    => 3200,
      pc_tot_dim => 3,
      srid       => null,
      blk_domain => null,
      get_ids    => 1
    )
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Finding Nearest Points: **SDO_PC_NN_FOR_EACH**

```
with candidates AS (
  select
    blocks.blk_id,
    SDO_GEOM.SDO_INTERSECTION(subqueries.ind_dim_qry, blocks.blk_extent, 0.05),
    subqueries.other_dim_qry
  from
    blocks blocks,
    (
      select 1 min_res, 1 max_res, :window ind_dim_qry, cast(null as sdo_mbr) other_dim_qry from dual
      union all
      select 2 min_res, 5 max_res, :window ind_dim_qry, cast(null as sdo_mbr) other_dim_qry from dual
    ) subqueries
  where
    blocks.obj_id = 1 and
    blocks.pcblk_min_res between min_res and max_res and
    SDO_ANYINTERACT(blocks.blk_extent, subqueries.ind_dim_qry) = 'TRUE'
)
select /*+ parallel (2) */ *
from table(
    sdo_pc_pkg.sdo_pc_nn_for_each(
      blocks      => cursor(select * from candidates),
      pc          => (select pc from pcs where id = 1),
      n           => 10,
      max_dist    => 10,
      qry_min_res => 1,
      qry_max_res => 1
    )
)
order by obj_id, blk_id, pt_id, neighbor_rank;
```

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

The “Flat” Model

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Flat Storage Model: **Simple Point Tables**

- "Back to basics…" a **simple flat relational model**
- X, Y, Z and other attributes stored as **ordinary columns**
- **No index** needed on Exadata
- Can partition on X, Y ,Z

```
CREATE TABLE  points (
  x         NUMBER,
  y         NUMBER,
  z         NUMBER,
  intensity NUMBER,
  returnval NUMBER,
  red       NUMBER,
  green     NUMBER,
  blue      NUMBER
);
```

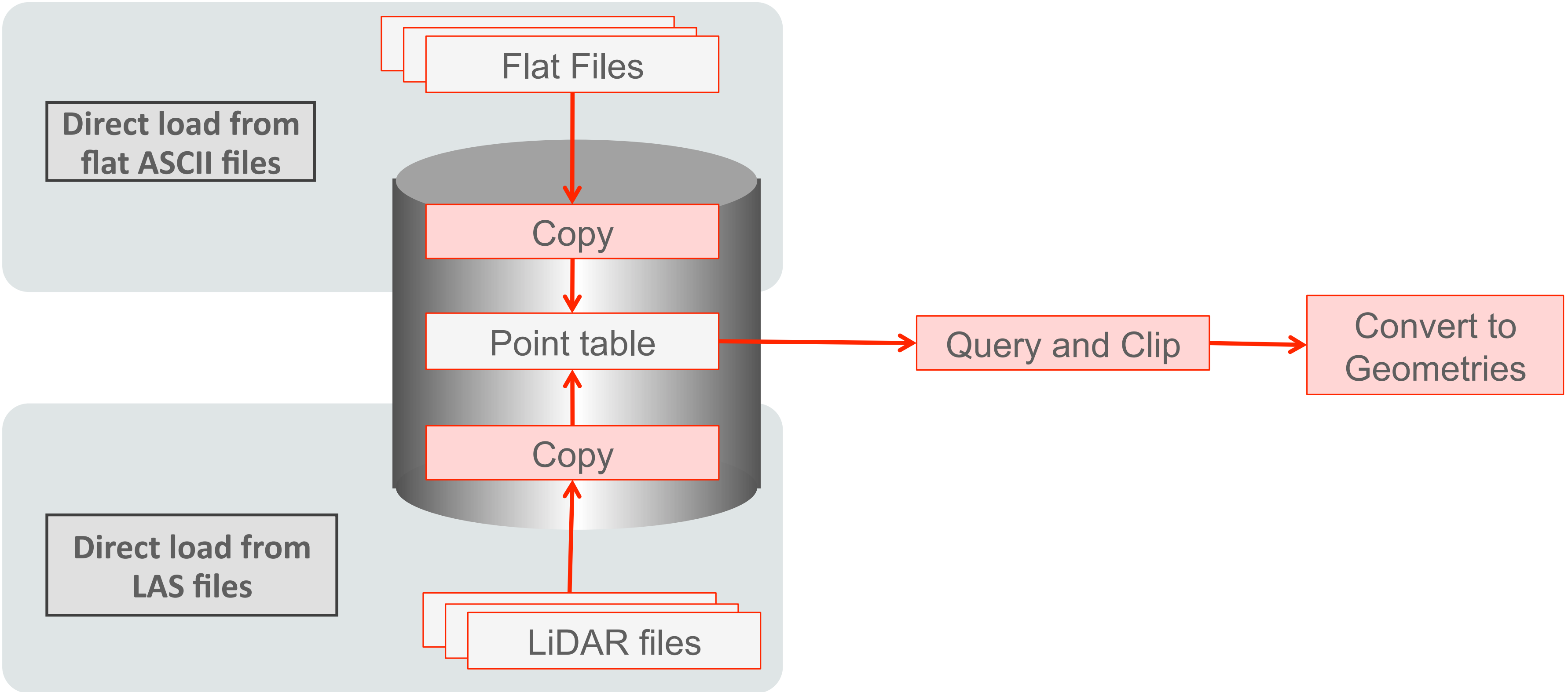
Takes advantage of intelligent Exadata storage servers

- HCC compression, for extremely high compression rates
- Parallel Enabled Smart Scan for extreme performance, including spatial queries.

ORACLE

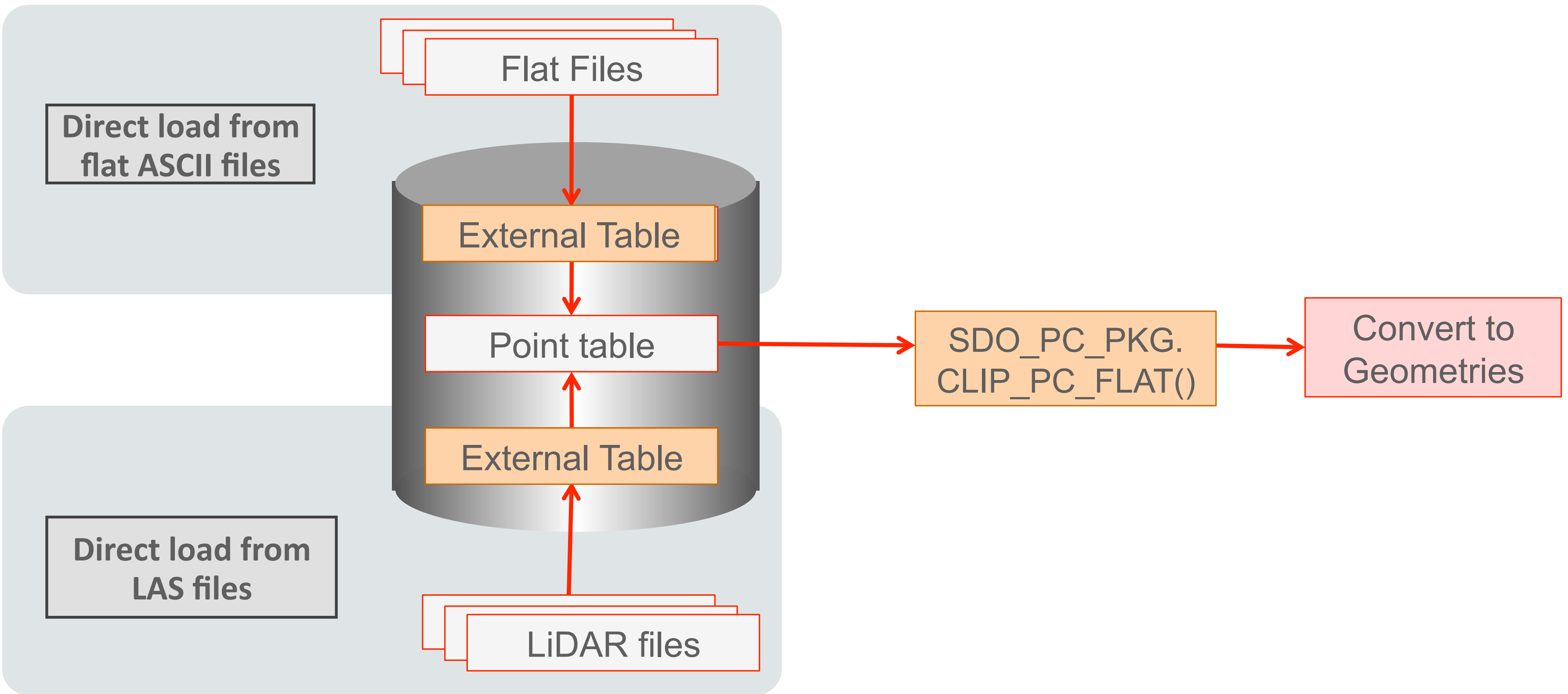
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Workflow for **Flat** Model



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Workflow for **Flat** Model

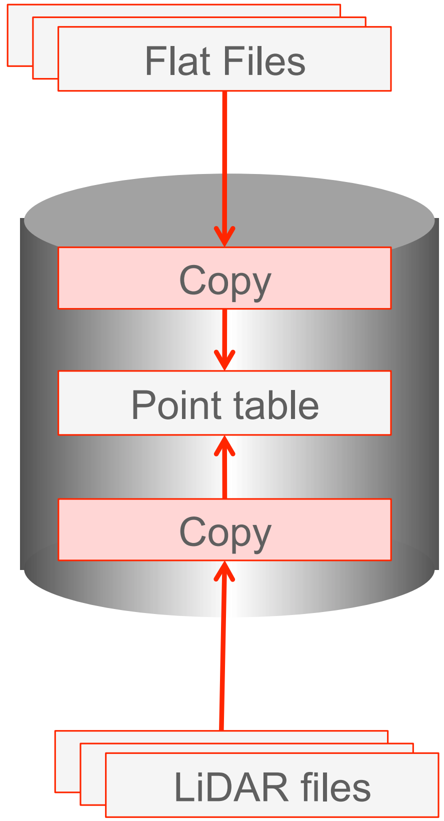


Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Loading: Simple and Direct

- Define flat files and LAS files as **external tables**
- Pre-processor **extracts points** from LAS files
- Loading is a simple **data copy** from the external table to the point table

```
CREATE TABLE points AS
SELECT ...
FROM ...
```



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

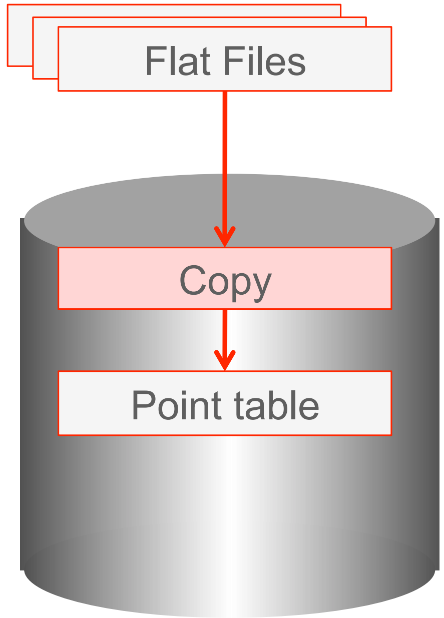
Loading from Flat CSV File(s)

External table for CSV files

```
CREATE TABLE points_csv_ext (
  x         NUMBER,
  y         NUMBER,
  z         NUMBER,
  intensity NUMBER,
  returnval NUMBER,
  red       NUMBER,
  green     NUMBER,
  blue      NUMBER
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY las_data_dir
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE
    FIELDS TERMINATED BY ",")
  LOCATION (
   'lidar_data_01.dat',
   'lidar_data_02.dat'
  )
)
REJECT LIMIT UNLIMITED
PARALLEL;
```

Any number of input files

```
CREATE TABLE points
   NOLOGGING
   COMPRESS FOR QUERY HIGH
   PARALLEL 96
AS
SELECT x,y,z, intensity,
  returnval, red, green, blue
FROM points_las_ext;
```



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

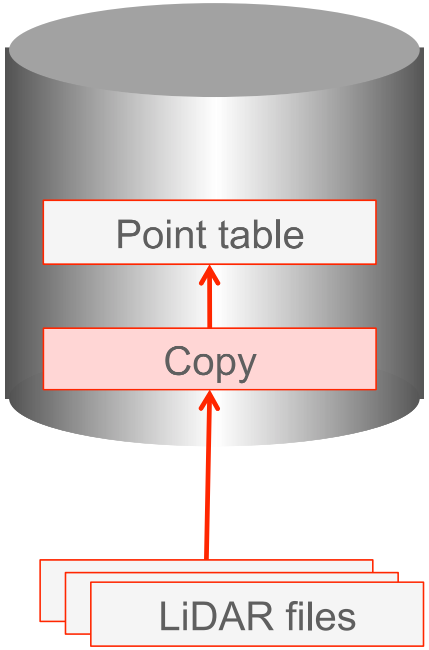
Loading from LAS File(s)

External table for LAS files

```
CREATE TABLE points_las_ext (
  x         NUMBER,
  y         NUMBER,
  z         NUMBER,
  intensity NUMBER,
  returnval NUMBER,
  red       NUMBER,
  green     NUMBER,
  blue      NUMBER
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY las_data_dir
  ACCESS PARAMETERS (
    PREPROCESSOR 'preprocessor.sh'
    RECORDS DELIMITED BY NEWLINE
    FIELDS TERMINATED BY ",")
  LOCATION (
   'lidar_data_01.las',
   'lidar_data_02.las'
  )
)
REJECT LIMIT UNLIMITED
PARALLEL;
```

Filter for LAS data

```
CREATE TABLE points
    NOLOGGING
    COMPRESS FOR QUERY HIGH
    PARALLEL 96
AS
SELECT x,y,z, intensity,
  returnval, red, green, blue
FROM points_las_ext;
```



Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

The Pre-processor Directive

preprocessor..sh

```
#!/bin/bash
$JAVA_HOME/bin/java -classpath $ORACLE_HOME/md/jlib/sdoutl.jar \
  oracle.spatial.util.Las2SqlLdr $1
```

- Read and decode the LAS file
- Write output as simple CSV format
- Can also use las2txt or a PDAL workfkow

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: Direct Filtering

- Returns the **points** that match the query window
- Simple box queries

```
select *
from points
where x between 0 and 2047
and y between 0 and 2047;
```

- Include also any attribute filtering

```
select *
from points
where x between 0 and 2047
and y between 0 and 2047
and intensity > 1.5;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: **CLIP_PC_FLAT**

- Returns the **points** that match a geometric query window

```
select *
from table (
  SDO_PC_PKG.CLIP_PC_FLAT(
    geometry =>
      SDO_GEOMETRY(2003, NULL, NULL,
        SDO_ELEM_INFO_ARRAY(1,1003,3),
        SDO_ORDINATE_ARRAY(0,0,2047,2047)
      ),
    table_name    => 'POINTS',
    tolerance     => 0.05,
    other_dim_qry => null,
    mask          => null
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: **SDO_PointInPolygon**

- Returns the **points** that match a geometric query window

```
select *
from table (
  sdo_PointInPolygon(
    CURSOR(
      select x, y, z from points
    ),
    SDO_GEOMETRY(2003, NULL, NULL,
      SDO_ELEM_INFO_ARRAY(1,1003,3),
      SDO_ORDINATE_ARRAY(0,0,2047,2047)
    ),
    0.05
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: SDO_PointInPolygon

- Returns the **points** that match the query window (a circle)

```
select *
from table (
  sdo_PointInPolygon(
    CURSOR(
      select x, y, z from points
    ),
    SDO_GEOMETRY(2003, NULL, NULL,
      SDO_ELEM_INFO_ARRAY(1, 1003, 4),
      SDO_ORDINATE_ARRAY(10, 12, 12, 10, 14, 12)
    ),
    0.05
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: **SDO_PointInPolygon**

- Include any additional filtering

```
SELECT x, y, z
FROM TABLE(
  sdo_PointInPolygon(
    CURSOR(
      select x, y, z from points
      where x between 10 and 14
      and y between 10 and 14
    ),
    SDO_GEOMETRY(2003, NULL, NULL,
      SDO_ELEM_INFO_ARRAY(1, 1003, 4),
      SDO_ORDINATE_ARRAY(10, 12, 12, 10, 14, 12)
    ),
    0.05
  )
);
```

ORACLE

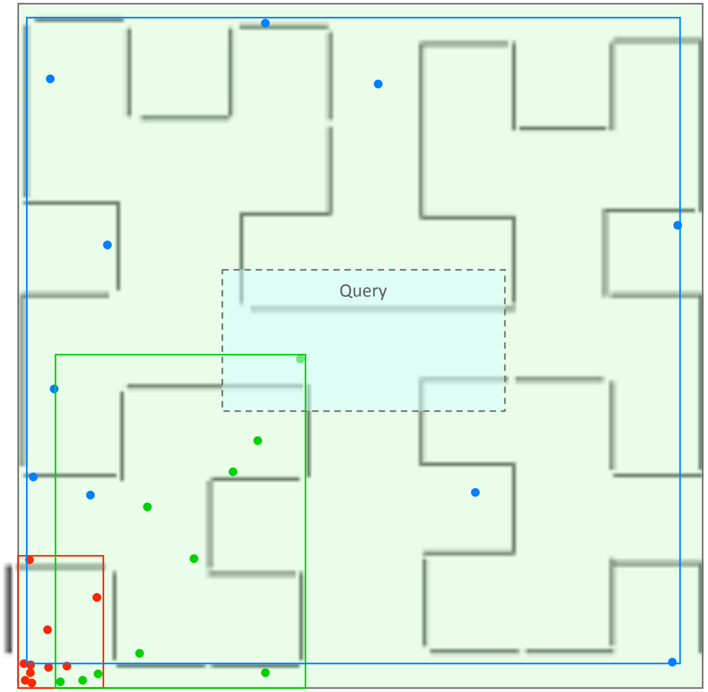
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

The “Hybrid” Model

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Index-Organized Table (IOT) with Spatial Ordering

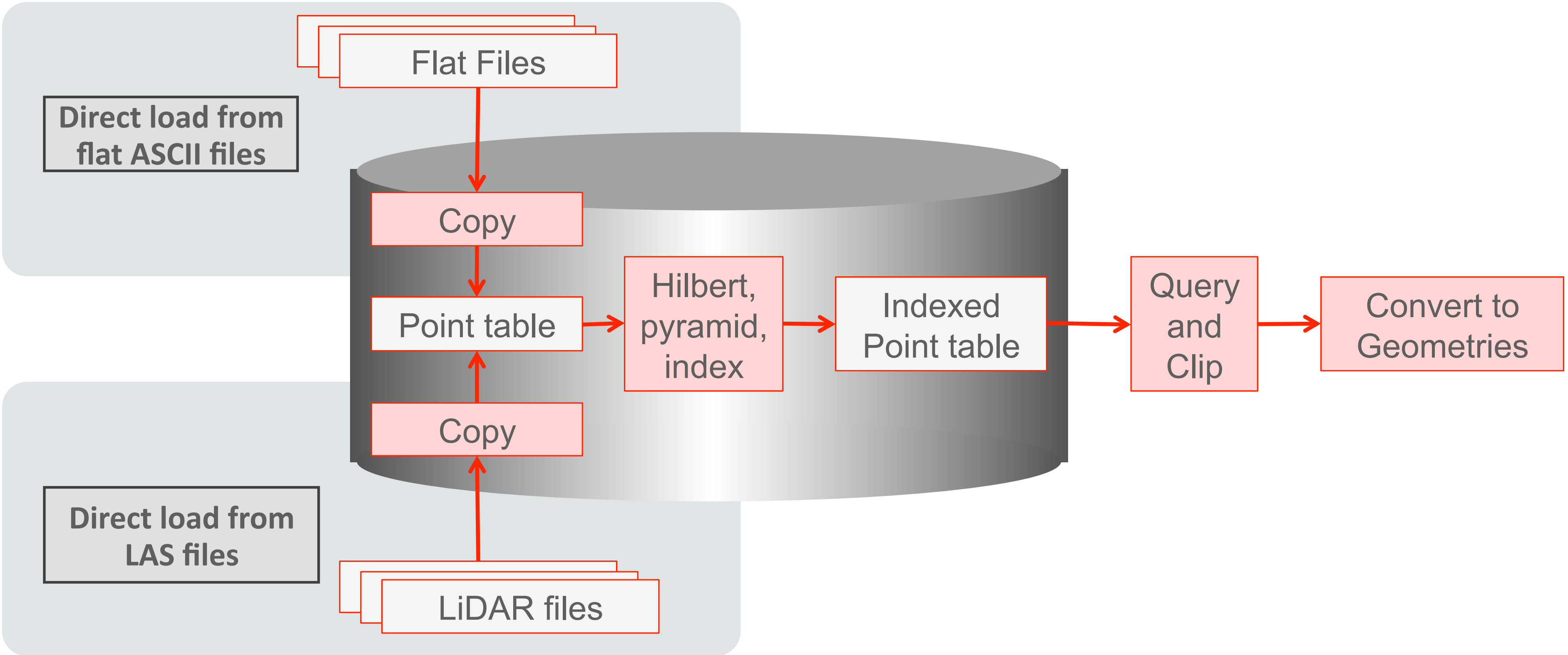
- Motivation:
  - Blocked model scalability on non-Exadata
- Spatial Partitioning
  - IOT, by Hilbert
- No BLOB Encoding



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Workflow for **Hybrid** Model



ORACLE

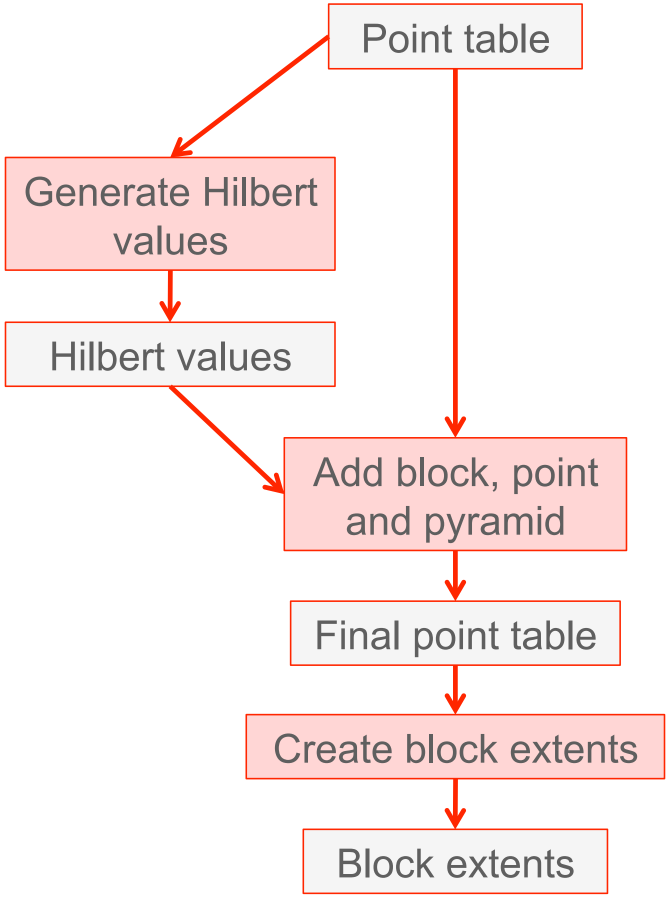
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Loading Process

Assume flat input table

```
CREATE TABLE input_points (
  rid              VARCHAR2(40),
  val_d1           NUMBER,
  val_d2           NUMBER,
  val_d3           NUMBER,
  ...
)
```

1) Generate Hilbert values, if necessary
2) Add block id, point id, pyramid level
3) Create block extent table
4) Index block extent table



ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

(1/4) Generate Hilbert Values

```
CREATE TABLE points$hilbert (
  rid,
  d,
  CONSTRAINT pk_points$hilbert PRIMARY KEY (d))
    ORGANIZATION INDEX
as
select *
from
    table(
      sdo_pc_pkg.generate_hilbert_vals(
        id_xy => cursor(select rowid, val_d1, val_d2 from points)
      )
    );
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

(2/4) Generate Blocking & Pyramiding Info

```
CREATE TABLE points$final (
  blk_id, pt_id, d, p,
  val_d1, val_d2, val_d3,
  CONSTRAINT pk_points$final PRIMARY KEY (p, blk_id, pt_id))
    ORGANIZATION INDEX
as
select
  t1.blk_id, t1.pt_id, t1.hilbert, t1.p,
  t2.val_d1, t2.val_d2, t2.val_d3
from
  table(
    sdo_pc_pkg.generate_hbp_vals(
      id_hilbert   => cursor(select rid, d from points$hilbert),
      blk_capacity => 10000)
  ) t1,
  points t2
where t2.rowid = t1.rid;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

(3/4) Generate Block MBRs

```
create table points$blocks (
  p, blk_id, num_points, max_d, blk_extent,
  CONSTRAINT pk_points$blocks PRIMARY KEY (p, max_d))
as
select
  p, blk_id, count(*), max(d),
  SDO_GEOMETRY(
    2003, NULL, NULL,
    SDO_ELEM_INFO_ARRAY(1, 1003, 3),
    SDO_ORDINATE_ARRAY(min(val_d1),min(val_d2),max(val_d1),max(val_d2))
  )
from points$final
group by p, blk_id;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

(4/4) Create Spatial Index

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES (
  'POINTS$BLOCKS',
  'BLK_EXTENT',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 0, 100000, 0.05),
    SDO_DIM_ELEMENT('Y', 0, 100000, 0.05)),
  null
);
```

```
create index sdo_idx_points$blocks on points$blocks(blk_extent)
 indextype is mdsys.spatial_index;
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Querying: Use the **Block Extents**

- Use a spatial query against the block extents

```
select *
from points$final
where blk_id in (
  select blk_id
  from points$blocks
  where sdo_anyinteract (
    blk_extent,
    SDO_GEOMETRY(2003, NULL, NULL,
      SDO_ELEM_INFO_ARRAY(1, 1003, 3),
      SDO_ORDINATE_ARRAY(0, 0, 2047, 2047)
    )
  ) = 'TRUE'
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

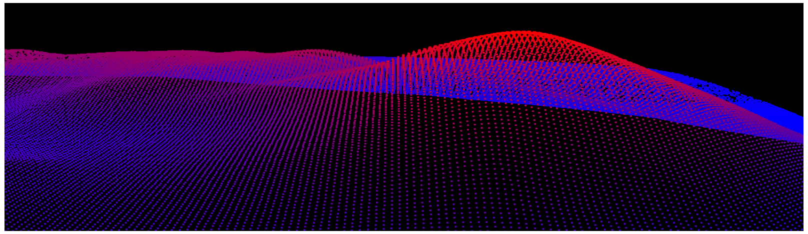
Querying: Use **SDO_PointInPolygon** for finer filtering

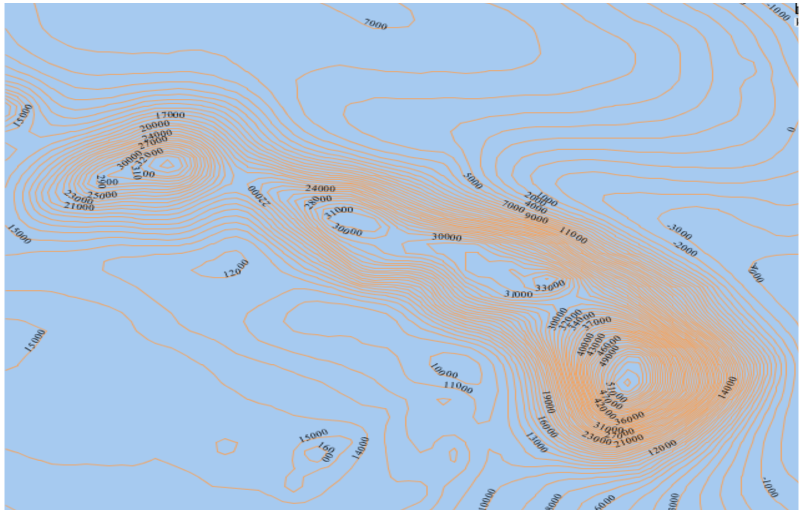
```
with p as (
  select *
  from points$final p, points$blocks b
  where p.blk_id = b.block_id
  and sdo_anyinteract (b.blk_extent, :query_window) = 'TRUE'
)
select *
from table (
  sdo_PointInPolygon(
    CURSOR(select * from p),
    :query_window,
    0.05
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Generate Contour Lines





Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Generate Contour Lines on **blocked** model

- Returns a list of geometry objects (SDO_GEOMETRY_ARRAY)

```
select *
from table (
  sdo_pc_pkg.create_contour_geometries(
    pc                  => (select pc from pcs where id = 1),
    sampling_resolution => 10,
    elevations          => sdo_ordinate_array(100, 200, 300),
    region              => sdo_geometry(2003, null, null,
                             sdo_elem_info_array(1, 1003, 3),
                             sdo_ordinate_array(0, 0, 999, 999)
                           )
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Generate Contour Lines on **flat** model

- Returns a list of geometry objects (SDO_GEOMETRY_ARRAY)

```
select *
from table (
  sdo_pc_pkg.create_contour_geometries(
    pc_flat_table       => 'POINTS',
    srid                => null,
    sampling_resolution => 10,
    elevations          => sdo_ordinate_array(100,200,300),
    region              => sdo_geometry(2003, null, null,
                             sdo_elem_info_array(1, 1003, 3),
                             sdo_ordinate_array(0, 0, 999, 999)
                           )
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Generate Contour Lines on **hybrid** model

- Returns a list of geometry objects (SDO_GEOMETRY_ARRAY)

```
select *
from table (
  sdo_pc_pkg.create_contour_geometries(
    pc_flat_table       => 'POINTS$FINAL',
    srid                => null,
    sampling_resolution => 10,
    elevations          => sdo_ordinate_array(100),
    region              => sdo_geometry(2003, null, null,
                             sdo_elem_info_array(1, 1003, 3),
                             sdo_ordinate_array(0, 0, 999, 999)
                           )
  )
);
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Generate and Save Contour Lines

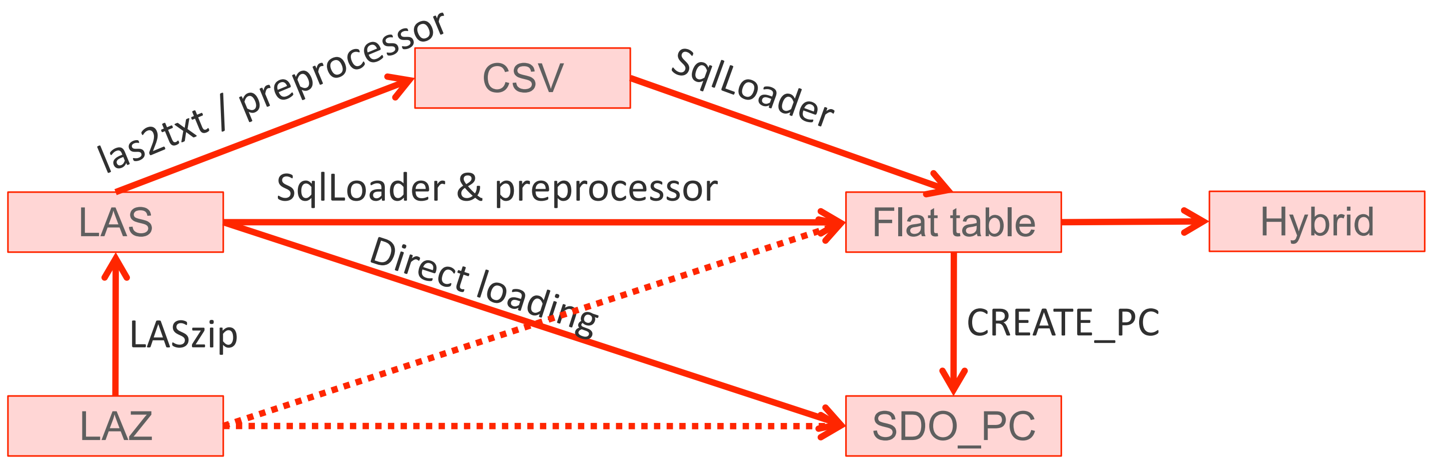
```
create table contours (
  id number,
  elevation number,
  geom sdo_geometry
);
```

```
declare
  contours sdo_geometry_array;
  elevations sdo_ordinate_array :=
    sdo_ordinate_array(
       100,  200,  300,  400,  500,  600,  700,  800,  900, 1000);
begin
  contours :=
    sdo_pc_pkg.create_contour_geometries(
      pc_flat_table => ('POINTS'),
      sampling_resolution => 10,
      elevations => elevations,
      region =>
        sdo_geometry(2003, null, null,
          sdo_elem_info_array(1, 1003, 3),
          sdo_ordinate_array(-1000, -1000, 999, 999)
        )
      );
  for i in 1..elevations.count loop
    insert into contours (id, elevation, geom)
    values (i, elevations(i), contours(i));
  end loop;
end;
/
```

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Summary of Loading Processes



ORACLE
Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Conclusions

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Performance of Point Cloud Loading/Creation

Blocked R-tree (1)	~ $O(n^{3/2})$
Blocked Hilbert (2)	Scales better: Near linear (except for Hilbert sorting component O(nlogn))
Blocked Hilbert JDBC Client (3)	Faster: Lower overhead, due to limited batch size… (1Mpts/s)
Flat Exadata (4)	Fastest: Approx. linear, lower overhead, no indexing… (640Bpts LAZ in 4:39h on Exadata X4-2 Full Rack)
Flat Others (5)	$O(n\,log\,n)$ ((x, y) sorting)
Hybrid Hilbert (6)	Similar to blocked (2)

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

EXADATA X4-2 Half Rack
96 cores

Load Performance

Compression	CSV File 2,835,027,995 Rows	1000 LAS Files 3,265,110,000 Rows
Query Low	1 min 20 sec 35,662,849 rows/sec	5 min 27 sec 9,985,045 rows/sec
Query High	2 min 2 sec 23,385,475 rows/sec	6 min 41 sec 9,547,105 rows/sec
Archive Low	2 min 3 sec 23,008,290 rows/sec	7 min 49 sec 6,961,855 rows/sec
Archive High	2 min 26 sec 19,408,353 rows/sec	14 min 36 sec 3,723,044 rows/sec

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Performance of Point Cloud Queries

Blocked (7)	Great Scalability: ~ O(# touched blocks)
Flat Exadata (8)	Great Scalability: ~ O(# Pts in query MBR)
Flat Others (9)	Similar to (8), except, needs a B-tree
Hybrid Hilbert (10)	Faster: Similar to (7), except lower overhead, due to no decoding

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

EXADATA X4-2 Half Rack
96 cores

Query Performance

- Random box queries on a data set of 2.9 billion points

Query ID	Rows Returned	Query Low Rows/Sec	Query High Rows/Sec	Archive Low Rows/Sec	Archive High Rows/Sec
1	100,234	75,934	56,951	58,961	21,648
2	101,914	76,627	56,935	57,578	19,944
3	107,318	96,682	58,965	58,009	14,046
4	1,013,301	858,729	509,196	484,833	125,563
5	1,044,341	976,019	561,473	558,471	133,547
6	1,080,314	871,220	534,808	524.424	145,398
7	10,053,844	2,072,240	2,047,626	2,171,456	1,180,028
8	10,085,246	2,136,704	2,041,547	2,178,238	1,097,415
9	101,757,599	2,351,146	2,220,327	2,302,279	2,344,645

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Compression and Point Cloud Models

- Blocked
  - Secure File Compression: HIGH, MEDIUM, LOW
- Flat
  - HCC (Exadata)
- Hybrid
  - "Exadata *Hybrid Columnar Compression* (EHCC) is not allowed on *Index Organized Tables* (IOT)"

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Using EXADATA HCC
(Hybrid Columnar Compression)

Impact of Compression

Compression	Rows	Size GB	Compression Ratio
No Compression	2,853,027,995	285.80	--
Query Low	2,853,027,995	36.9	7.74 x
Query High	2,853,027,995	12.67	22.55 x
Archive Low	2,853,027,995	12.65	22.59 x
Archive High	2,853,027,995	9.28	30.79 x

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Pro's and Con's of each Model

Storage model	Pro	Con
Blocked	• Storage (compression) • Scaling • Indexing • DB functionalities • Complex queries	• Loading (create blocks) • Block overhead in queries (noticeable in simple queries)
Flat	• Faster loading • DB functionalities • Dynamic schema (→ blocked) • Simple queries	• Storage (except Exadata) • Limits to scaling (except Exadata) • Indexing (except Exadata)
Hybrid	•Faster queries (→ blocked) •More scalable queries (→ flat) •Dynamic schema (→ blocked)	• No compression (no HCC with IOT)

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Where do we go from here ?

- Derivation of 3D models
  - Classification, conflation with data from other sources
- Web-based or service-based rendering
  - Visual inspection, etc.
  - Using the full resolution of the dataset or parts thereof (pyramiding)
- Selective data dissemination
  - Extract subsets for analysis by external tools
- In-database processing and analytics
  - Change detection in multi-temporal point clouds (buildings, vegetation, ...)

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

The Need for Interoperability

- Data distribution / file structures:
  - LAS and LAZ
- Application and tools:
  - PDAL
- Web Services:
  - OGC standard proposed: Web Point Cloud Service (WPCS)

asprs
THE IMAGING & GEOSPATIAL INFORMATION SOCIETY

pdal

OGC® Making location count.

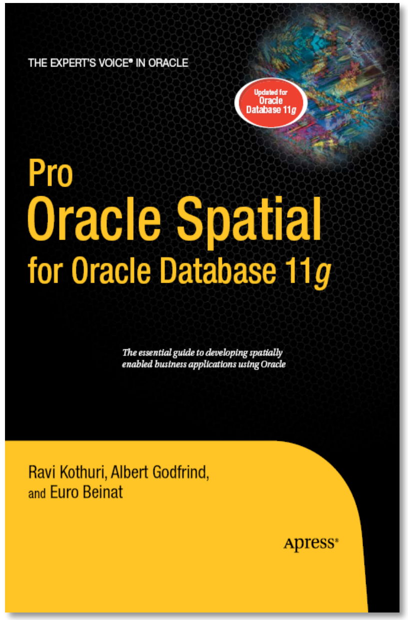
ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

To find out more …

Oracle Technology Network

www.oracle.com/goto/spatial

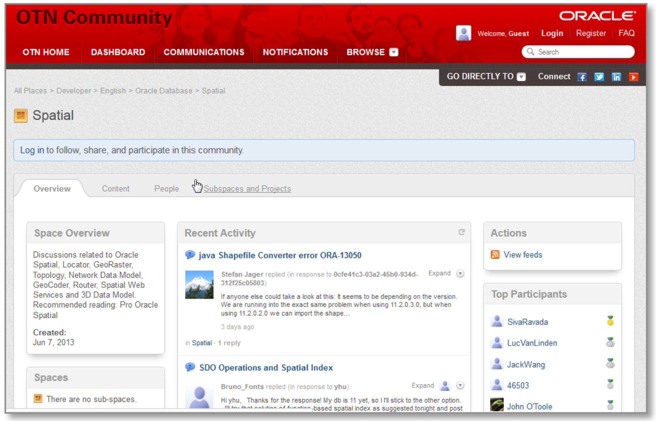


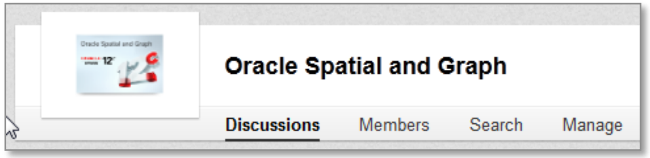
Examples, white papers, downloads, discussion forum, sample data ....

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

More resources

- Blogs
  - https://blogs.oracle.com/oraclespatial
- Developer forums on OTN
  - https://community.oracle.com/community/database/oracle-database-options/spatial
- LinkedIn community
  - „Oracle Spatial and Graph“ group
- Google+ community
  - „Oracle Spatial and Graph SIG“





Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

Hardware and Software

Engineered to Work Together

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |