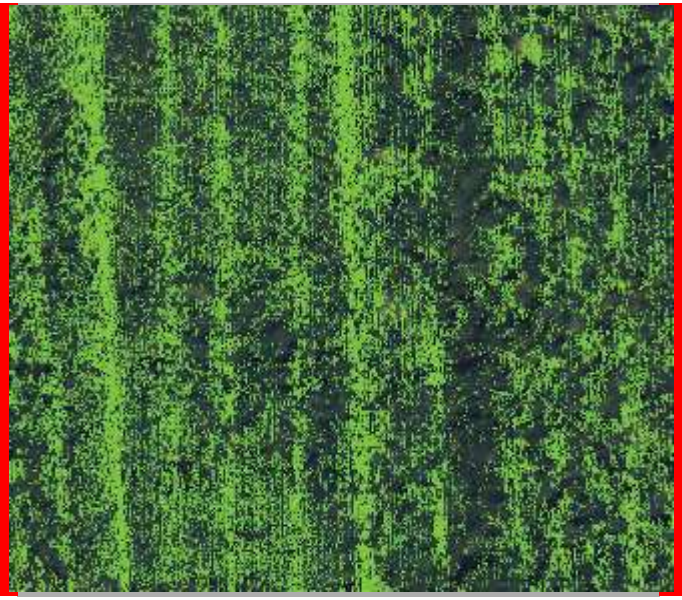


ORACLE®

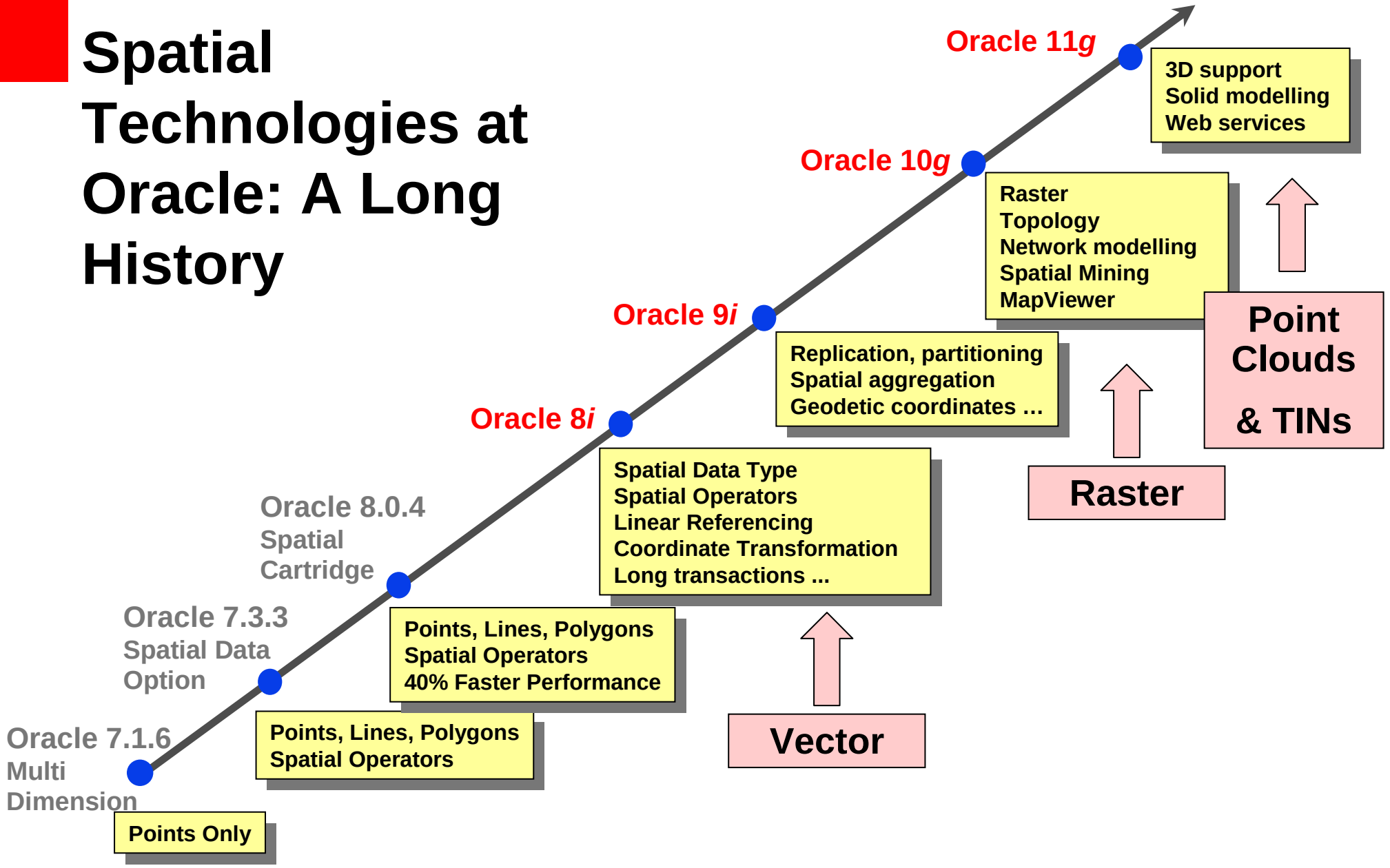
Oracle’s Point Cloud datatype

Albert Godfrind
Geospatial and Multimedia Technologies
Oracle Corporation

Agenda

- A short history of Oracle Spatial technologies
- Point clouds and Databases: the challenges
- Separating the "physical" from the "logical"
- SDO_PC type structure
- Physical type: the point cloud blocks
- Loading point clouds
- Processing functions
- TINs: the SDO_TIN and tin blocks structures
- Generating TINs from point clouds
- Conclusion

Spatial Technologies at Oracle: A Long History



Data Management Challenges of Point Clouds

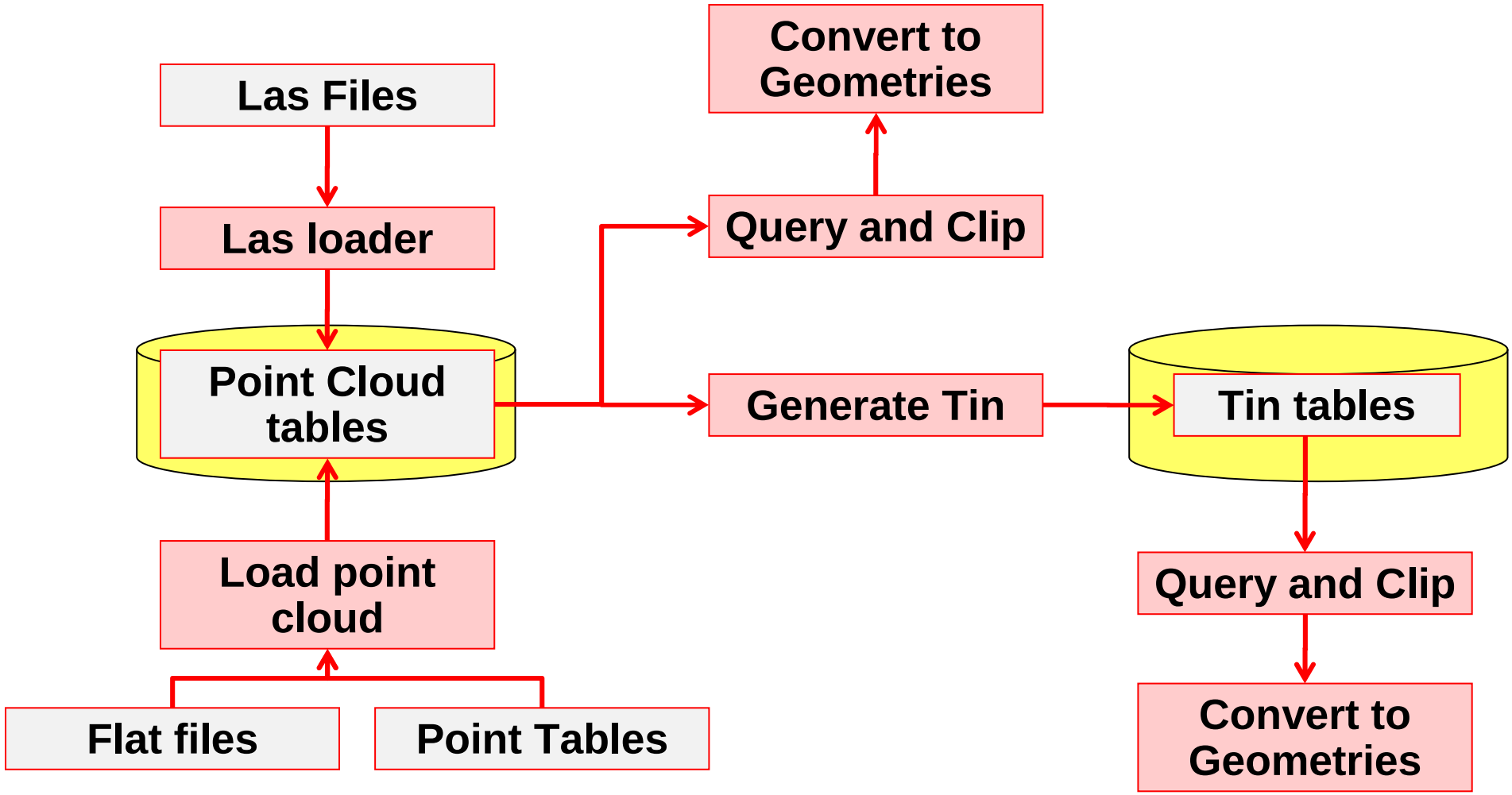
- **Volume**: increasing LiDAR densities with technology – Billions of points
  - Multi-return 150 KHz * 4 = 600k/sec
  - Full-waveform 250 KHz * 2048 =512,000,000/sec

- **But also:**
  - Lag-time from acquisition to analysis
  - Metadata access and management
  - Fusion with other geospatial data (terabytes)
  - Multi-user access and security
  - Versioning, Archiving

More Data Management Challenges

- Data Transformation
  - Surfaces (TIN, DEM, vector transformation)
- Projections
- Data integration
- Filtering, Visualization and Analysis
- Backup, Recovery and minimizing downtime

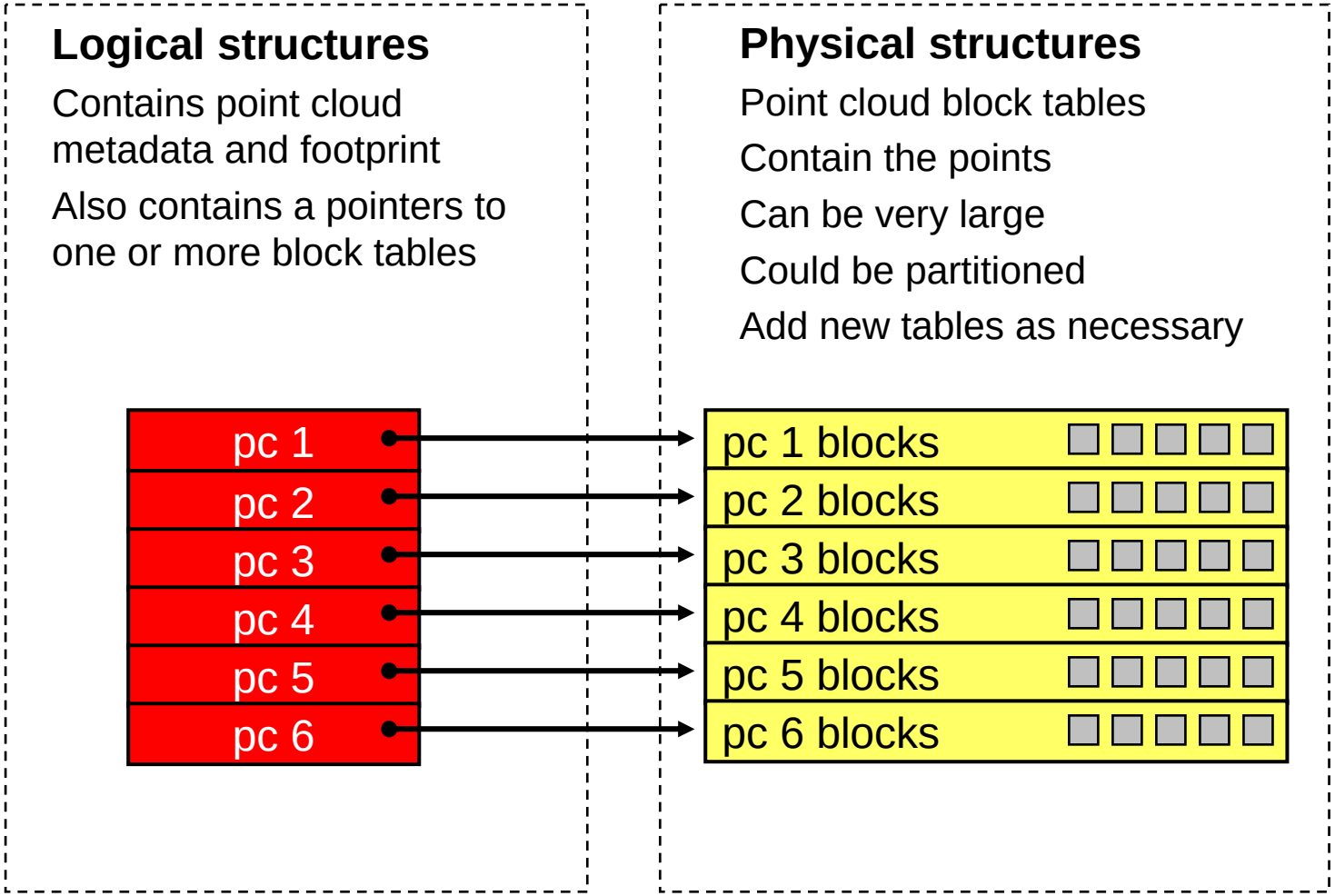
Process Flow



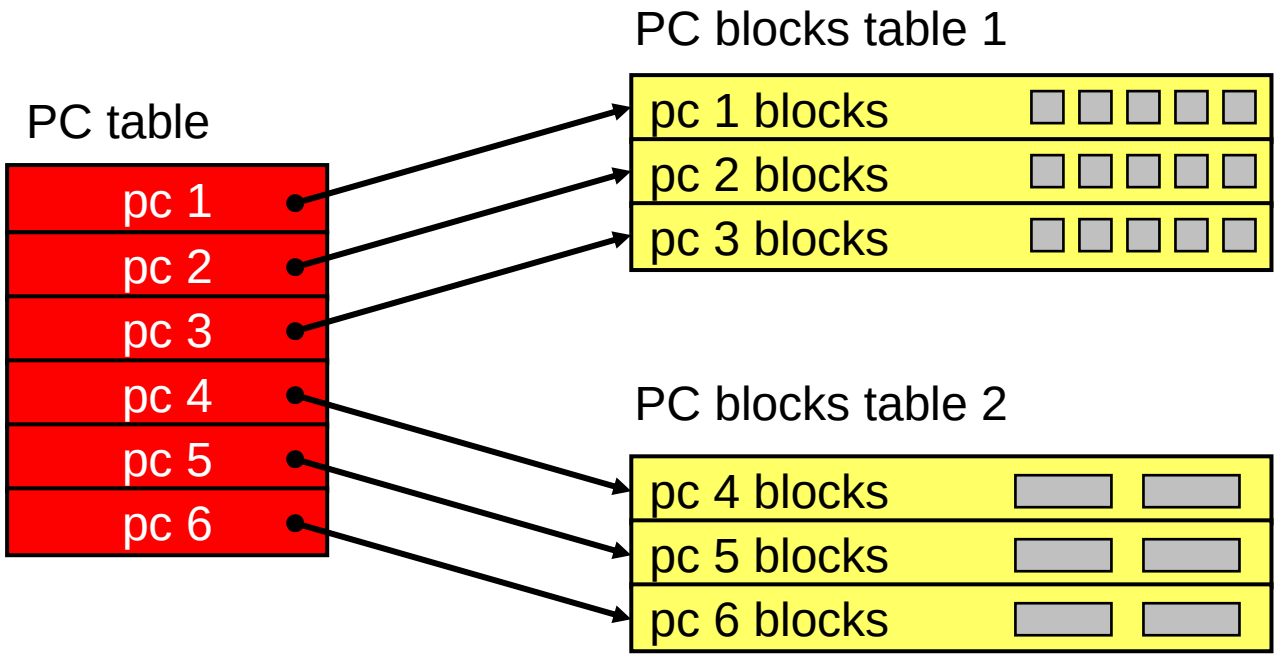
Storage Model for Point Clouds

- Separates logical from physical structures
- Logical structures
  - Tables containing an SDO_PC column
  - Contains generic attributes and footprint
  - Also contains a pointer to a PC block table
- Physical structures
  - “Block tables”
  - Contain point cloud blocks
  - Can be very large
  - Structure defined in SDO_PC_BLK object type

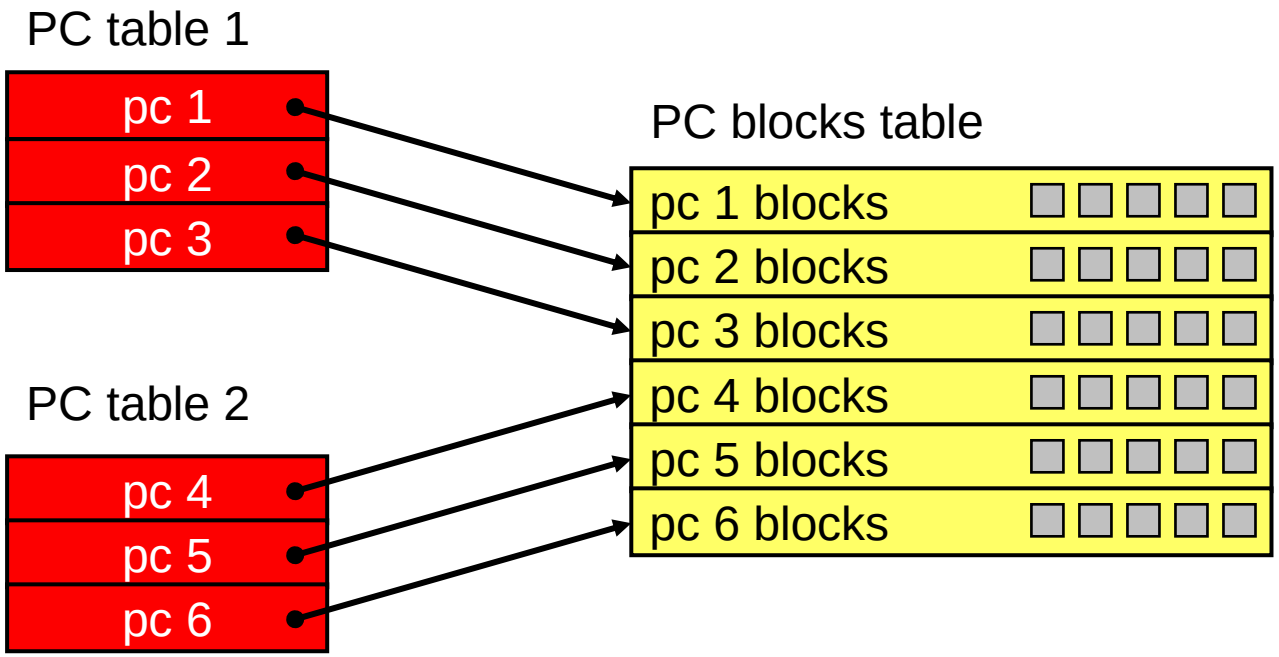
Storage Model



Storage Model



Storage Model



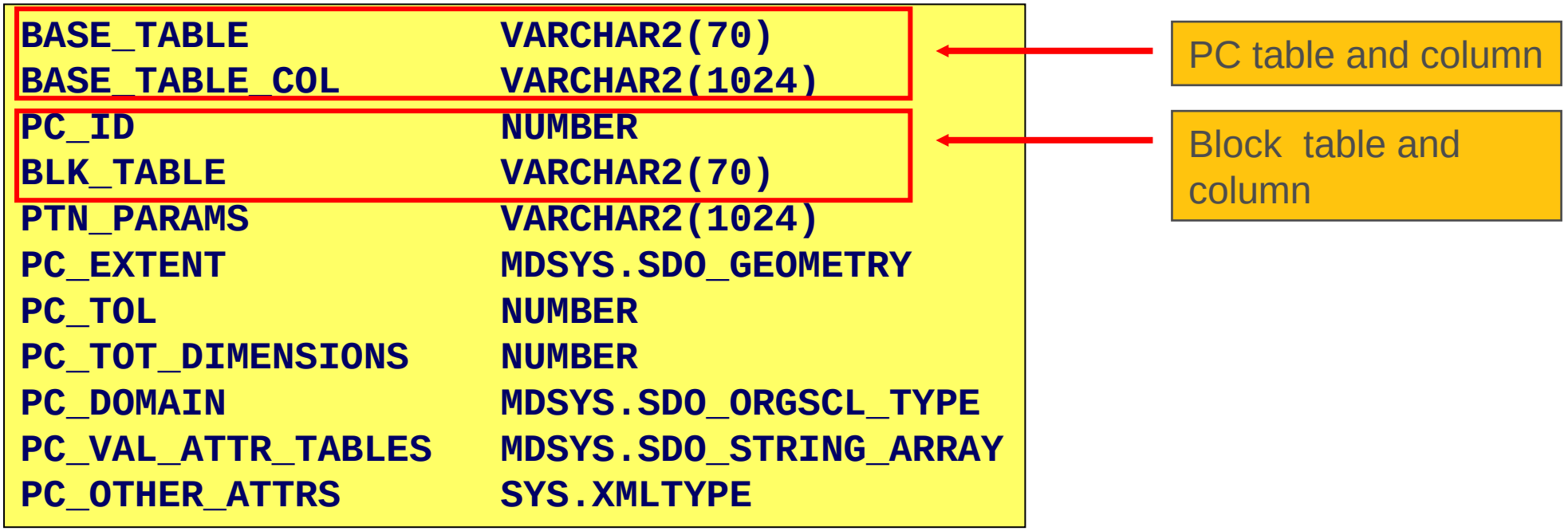
Creating Point Cloud Tables

- Use the SDO_PC type
- Can have any number of PC tables
- Include any combination of attributes
- Scenes can be searched on any attribute
- Also the spatial extent of the scene

```
CREATE TABLE LIDAR_SCENES(
  SCENE_ID         NUMBER PRIMARY KEY,
  COLLECTION_TS    TIMESTAMP,
  DESCRIPTION      CLOB,
  ... (any number of attributes) ...
  POINT_CLOUD      SDO_PC
);
```

The SDO_PC type

- PC_EXTENT is the footprint of the point cloud
- Needs a spatial index to support spatial searching



The SDO_PC type

Attribute	Explanation
BASE_TABLE	Name of the base table containing a column of type SDO_PC
BASE_TABLE_COL	Name of the column of type SDO_PC in the base table
PC_ID	ID number for the point cloud
BLK_TABLE	Name of the table that contains information about each block in the point cloud.
PTN_PARAMS	Parameters for partitioning the point cloud
PC_EXTENT	SDO_GEOMETRY object representing the spatial extent of the point cloud (the minimum bounding object enclosing all objects in the point cloud)
PC_TOL	Tolerance value for points in the point cloud.
PC_TOT_DIMENSIONS	Total number of dimensions in the point cloud. Includes spatial dimensions and any nonspatial dimensions, up to a maximum total of 9.
PC_DOMAINS	(Not currently used.)
PC_VAL_ATTR_TABLES	SDO_STRING_ARRAY object specifying the names of any value attribute tables for the point cloud
PC_OTHER_ATTRS	XMLTYPE object specifying any other attributes of the point cloud

Creating Point Cloud Block Tables

- Using the SDO_PC_BLK type

```
CREATE TABLE PC_BLK_01 OF SDO_PC_BLK (
  PRIMARY KEY (
     OBJ_ID, BLK_ID
  )
)
LOB(POINTS) STORE AS SECUREFILE
  (COMPRESS HIGH NOCACHE NOLOGGING);
```

- Define a primary key on the block id
- Use SECUREFILE lobs (new structure in 11g)
- Allows compression of the LOBs!
  - (also encryption and de-duplication)

The SDO_PC_BLK type

- Describes one block of points

```
OBJ_ID                NUMBER
BLK_ID                NUMBER
BLK_EXTENT            MDSYS.SDO_GEOMETRY
BLK_DOMAIN            MDSYS.SDO_ORGSCL_TYPE
PCBLK_MIN_RES         NUMBER
PCBLK_MAX_RES         NUMBER
NUM_POINTS            NUMBER
NUM_UNSORTED_POINTS   NUMBER
PT_SORT_DIM           NUMBER
POINTS                BLOB
```

- Contains the unique identifier of the block
  - Scene id (OBJ_ID, same as PC_ID) and block id (BLK_ID)

The SDO_PC_BLK type

Attribute	Explanation
OBJ_ID	ID number of the point cloud object
BLK_ID	ID number of the block.
BLK_EXTENT	Spatial extent of the block.
BLK_DOMAIN	(Not currently used.)
PCBLK_MIN_RES	Minimum resolution level at which the block is visible in a query. The block is retrieved only if the query window intersects the spatial extent of the block and if the minimum - maximum resolution interval of the query. Usually, lower values mean farther from the view point, and higher values mean closer to the view point.
PCBLK_MAX_RES	Maximum resolution level at which the block is visible in a query.
NUM_POINTS	Total number of points in the POINTS BLOB
NUM_UNSORTED_POINTS	Number of unsorted points in the POINTS BLOB
PT_SORT_DIM	Number of the dimension (1 for the first dimension, 2 for the second dimension, etc) on which the points are sorted.
POINTS	BLOB containing the points.

BLOB Structure

- The BLOB contains an array of points
- Each point encoded as
  - *d* 64-bit floating point numbers (*d* = the dimensionality of the point)
  - One 32 bit integer representing the point number
  - One 32 bit integer representing the partition number
- Future: compressed format
  - Storing coordinates as offsets from the origin of the block MBR
  - Using short integers

Initializing a 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
  - Unique identifier in that table

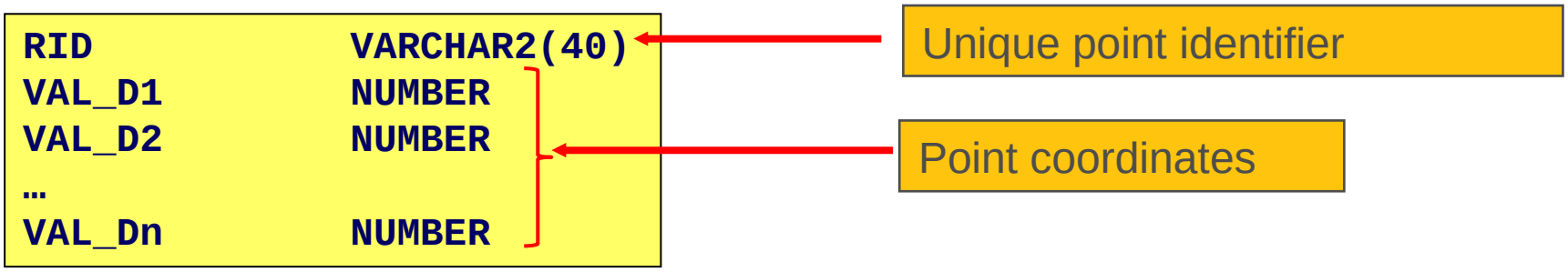
```
INSERT INTO LIDAR_SCENES (
  SCENE_ID, POINT_CLOUD)
VALUES (
  1,
  SDO_PC_PKG.INIT(
    BASETABLE          => 'LIDAR_SCENES',
    BASECOL            => 'POINT_CLOUD',
    BLKTABLE           => 'PC_BLK_01',
    PTN_PARAMS         => 'BLK_CAPACITY=1000',
    PC_TOL             => 0.005,
    PC_TOT_DIMENSIONS => 3,
    PC_EXTENT          =>
      SDO_GEOMETRY (2003, 4326, NULL,
        SDO_ELEM_INFO_ARRAY (1,1003,3),
        SDO_ORDINATE_ARRAY (-74, 40, -73, 41)
      )
  )
);
```

Loading a Point Cloud

- Load a point cloud from a flat table

```
DECLARE
  PC SDO_PC;
BEGIN
  SELECT POINT_CLOUD INTO PC
  FROM LIDAR_SCENES WHERE SCENE_ID = 1;
  SDO_PC_PKG.CREATE_PC (PC, 'INPUT_POINTS');
END;
/
```

- Structure of the input table



Loading a Point Cloud

```
CREATE TABLE input_points (
  rid             VARCHAR2(40),
  val_d1          NUMBER,
  val_d2          NUMBER,
  val_d3          NUMBER
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY data_files
  ACCESS PARAMETERS (
    FIELDS TERMINATED BY "," (
      rid,
      val_d1,val_d2,val_d3
    )
  )
  LOCATION ('input_points.dat')
);
```

- Input table could be a flat file
- Defined as an external table.

File "input_points.dat"

```
279, -73.999922, 40.000002, 74
280, -73.999921, 40.000002, 27
281, -73.999920, 40.000002, 76
282, -73.999919, 40.000002, 72
283, -73.999918, 40.000002, 91
284, -73.999917, 40.000002, 96
```

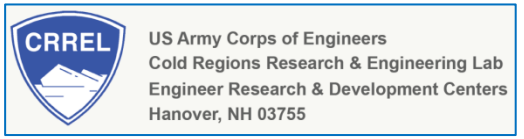
Getting LAS Data into Oracle

- Tests performed by Michael Smith and David Finnegan, US Army Corps or Engineers
- Presented at Oracle Spatial Users Conference, Tampa, Florida, April 23, 2009
- See http://download.oracle.com/otndocs/products/spatial/pdf/osuc2009_presentations/osuc2009_usace_smith.pdf

US Army Corps of Engineers
Cold Regions Research & Engineering Lab
Engineer Research & Development Centers
Hanover, NH 03755

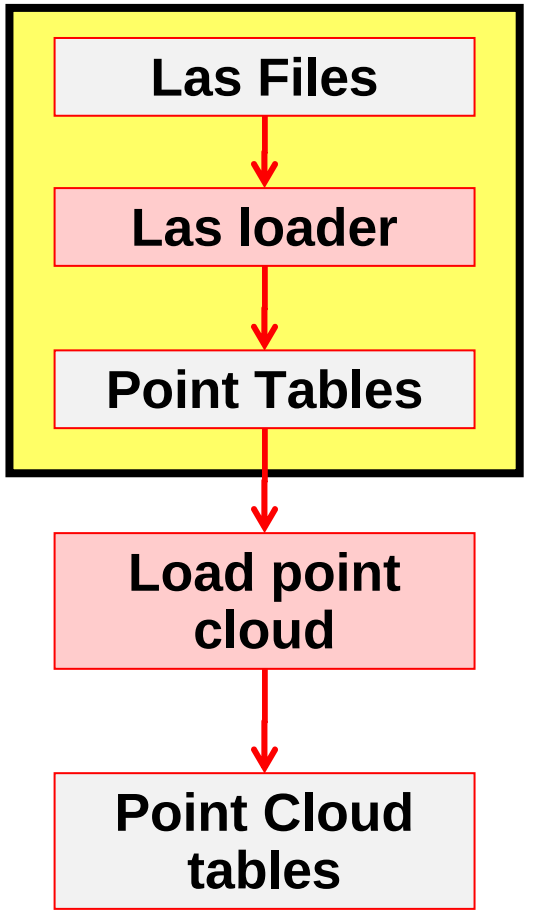
Getting LAS Data into Oracle

- Testing Machines:
  - Sun t5240 2 UltraSparcT2+ 1.2Ghz - 64Gb Ram
  - Sun x4150 x86 2 Xeon X5460 3.16GHz - 8Gb Ram
- Differences based on Chip Architecture
  - x86 faster than Sparc
- Speed of Temp location made a difference
  - moving temp tablespace from SAS to SSD to RAM yielded ~20-40% increase in speed
- Create PC without the results table
  - not needed and saved ~20% time
- Effect of Block Size
  - small increase with larger size



Converting LAS Data to Points

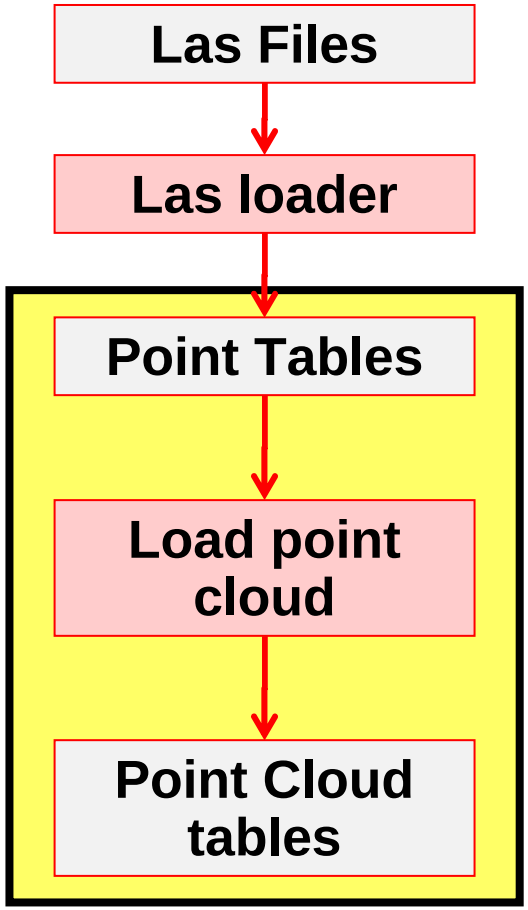
- Java LAS loader to Normal Table
  - Sparc: 3.8 Mpts/min
  - x86: 22.7 Mpts/min
- External Table and LibLAS to Normal Table
  - new in 11.1.0.7, preprocessor option
  - PREPROCESSOR exec_dir:'las2txt'
  - OPTIONS '--parse Mxyz –stdout'
  - Sparc: 41.2 Mpts/min
  - x86: 99.96 Mpts/min



US Army Corps of Engineers
Cold Regions Research & Engineering Lab
Engineer Research & Development Centers
Hanover, NH 03755

Creating Point Clouds

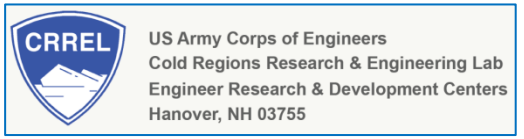
- Single Session
  - Sparc: 1.8 Mpts/min - 5k block size – ram temp
  - x86: 8.2 Mpts/min - 5k block size – ram temp
  - x86: 8.7 Mpts/min - 100K block size – ram temp
- Multiple Sessions
  - Used Apache Jmeter
  - Sparc:
    - 10, 20, 50, 100 simultaneous sessions
    - 1.2 – 2.5 Mpts/min
  - x86:
    - 6, 10, 20 sessions
    - 2.2 – 7.8 Mpts/min



CRREL
US Army Corps of Engineers
Cold Regions Research & Engineering Lab
Engineer Research & Development Centers
Hanover, NH 03755

Size Inflation!

- Original LAS File: 26 Mpts 505Mb
- Table and Index size:
  - Lobs (BasicFiles): 839.5 Mb
  - SecureFiles (no compression): 826.2 Mb
  - SecureFiles (medium compression): **223.4 Mb**



Processing Point Clouds

- Select scenes using spatial operators
- Use any spatial operator to search through point blocks
  - SDO_ANYINTERACT
  - SDO_NN
  - SDO_FILTER

Processing Point Clouds

- CLIP_PC (Clip Point Cloud)
  - 2D or 3D query window
  - Returns points for any block whose extent intersects the querywindow
  - Only points that intersect the query window are returned
  - Creates a new SDO_PC, can be stored or used in queries
- TO_GEOMETRY
  - Gets the points (as a Point Cluster) from a PC
  - Can be from a CLIP_PC operation

CLIP_PC: Clipping from a Point Cloud

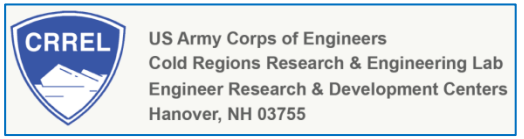
- Selects points from a point cloud that are within a spatial window.
- Can also select points based on specific dimension values
- Results in an array of point blocks

```
DECLARE
  PC SDO_PC;
BEGIN
  -- Get the scene to clip from
  SELECT POINT_CLOUD INTO PC
  FROM   LIDAR_SCENES
  WHERE  SCENE_ID = 1;

  -- Clip out the desired subset from the scene
  INSERT INTO CLIPPED_LIDAR_SCENES_BLOCKS
  SELECT * FROM TABLE (
    SDO_PC_PKG.CLIP_PC (
      INP             =>  PC,
      IND_DIM_QRY     =>  SDO_GEOMETRY(2003, 4326, NULL,
                            SDO_ELEM_INFO_ARRAY (1, 1003, 3),
                            SDO_ORDINATE_ARRAY (
                              -73.99996, 40.000066,
                              -73.99994, 40.000080
                            )
                          ),
      OTHER_DIM_QRY   =>  NULL,
      QRY_MIN_RES     =>  NULL,
      QRY_MAX_RES     =>  NULL
    )
  );
END;
/
```

Retrieving Point Cloud Data

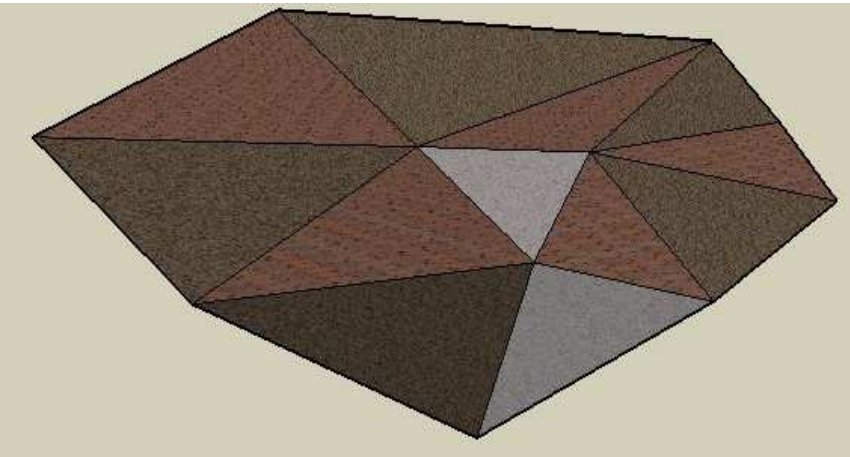
- 1 km circular buffer moving in data range
- Calculate average Z value, max Z value
- Sparc:
  - 30 sessions – avg: 8.46 sec / session
  - 300 sessions – avg: 8.64 sec / session
- x86:
  - 30 sessions – avg: 0.66 sec / session
  - 250 sessions – avg: 1.14 sec / session



Triangulated Irregular Networks (TINs)

SDO_TIN: Triangulated Irregular Network

- Representation of surfaces / terrains
- Contains a network of irregularly placed triangles
- Each point (triangle node) has X, Y and Z coordinates

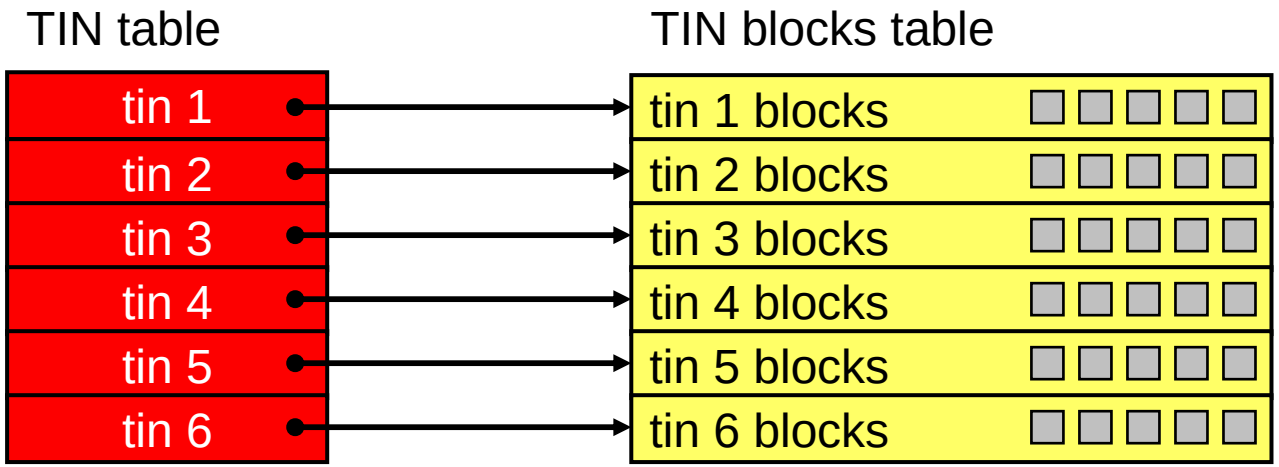


Node No	X	Y	Z
1	5	6	3
2	3	6	5
3	1	5	6
4	4	4	4
5	6	5	3
6	2	2	2
.	.	.	.

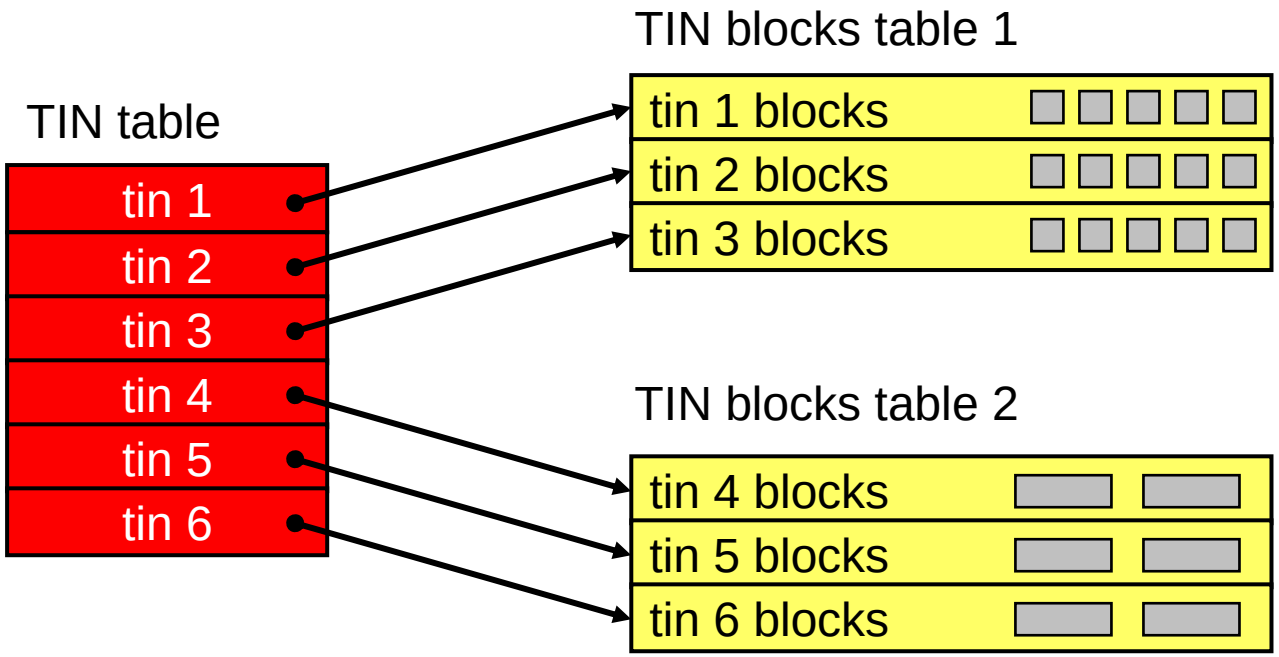
Storage Model

- Separates logical from physical structures
- Logical structures
  - Tables containing an SDO_TIN column
  - Contains metadata and footprint
  - Also contains a pointer to a TIN block table
- Physical structures
  - “Block tables”
  - Contain triangles
  - Can be very large
  - Structure defined in SDO_TIN_BLK object type

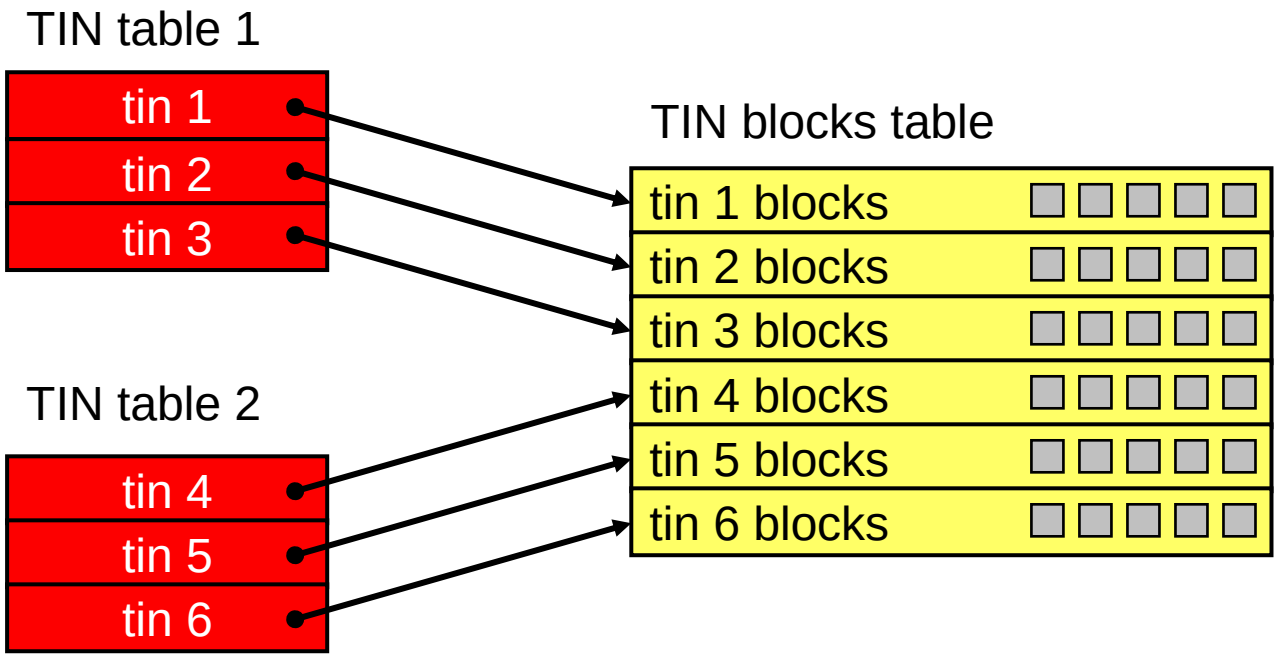
Storage Model



Storage Model



Storage Model



Creating TIN Tables

- Creating a TIN table

```
CREATE TABLE TINS (
  ID          NUMBER PRIMARY KEY,
  TIN         SDO_TIN
);
```

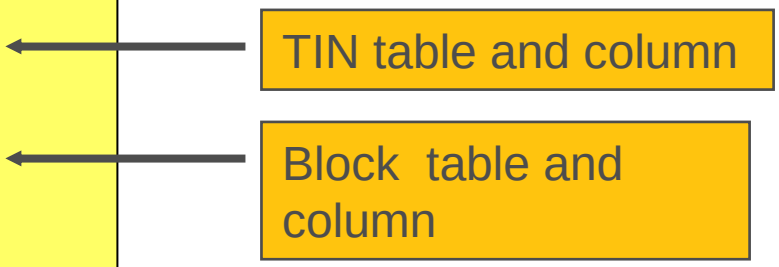
- Creating a TIN block table

```
CREATE TABLE TIN_BLOCKS_01 OF SDO_TIN_BLK (
  PRIMARY KEY (
     OBJ_ID, BLK_ID
  )
)
LOB(POINTS) STORE AS SECUREFILE (NOCACHE NOLOGGING)
LOB(TRIANGLES) STORE AS SECUREFILE (NOCACHE NOLOGGING);
```

The SDO_TIN type

- TIN_EXTENT is the footprint of the TIN
- May need a spatial index

```
BASE_TABLE              VARCHAR2(70)
BASE_TABLE_COL          VARCHAR2(1024)
TIN_ID                  NUMBER
BLK_TABLE               VARCHAR2(70)
PTN_PARAMS              VARCHAR2(1024)
TIN_EXTENT              MDSYS.SDO_GEOMETRY
TIN_TOL                 NUMBER
TIN_TOT_DIMENSIONS      NUMBER
TIN_DOMAIN              MDSYS.SDO_ORGSCL_TYPE
TIN_BREAK_LINES         MDSYS.SDO_GEOMETRY
TIN_STOP_LINES          MDSYS.SDO_GEOMETRY
TIN_VOID_RGNS           MDSYS.SDO_GEOMETRY
TIN_VAL_ATTR_TABLES     MDSYS.SDO_STRING_ARRAY
TIN_OTHER_ATTRS         SYS.XMLTYPE
```



The SDO_TIN_BLK type

- Describes one TIN block
- Contains a LOB with the points in that block.
- Contains another LOB with the triangles.

OBJ_ID	NUMBER
BLK_ID	NUMBER
BLK_EXTENT	MDSYS.SDO_GEOMETRY
BLK_DOMAIN	MDSYS.SDO_ORGSCL_TYPE
PCBLK_MIN_RES	NUMBER
PCBLK_MAX_RES	NUMBER
NUM_POINTS	NUMBER
NUM_UNSORTED_POINTS	NUMBER
PT_SORT_DIM	NUMBER
POINTS	BLOB
TR_LVL	NUMBER
TR_RES	NUMBER
NUM_TRIANGLES	NUMBER
TR_SORT_DIM	NUMBER
TRIANGLES	BLOB

BLOB Structure

- The *POINTS* blob contains an array of points
- Each point encoded as
  - *d* 64-bit floating point numbers (*d* = the dimensionality of the point
  - One 32 bit integer representing the point number
  - One 32 bit integer representing the partition number
- The *TRIANGLE* blob contains an array of triangles
  - Each triangle defined by 3 points
  - Points identified by their number in the POINTS blob

Package *SDO_TIN_PKG*

- Initialization of a TIN
  - sdo_tin_pkg.init()
- Loading a TIN from a point cloud
  - sdo_tin_pkg.create_tin()
- Extracting triangles in a spatial window
  - sdo_tin_pkg.clip_tin()
- Converting to SDO_GEOMETRY
  - sdo_tin_pkg. to_geometry()

Enhancements Under Consideration

- Refined granular access to SDO_PC
  - Create PC without needing input table
  - Insert/Update new Blocks of PC
  - Increase maximum dimensionality
- Enable query /update / indexing of values beyond spatial in PC

Integration with standard RDBMS features

- Moving massive amounts of LIDAR data between databases:
  - Use Transportable Tablespaces
- Storage control and scalability
  - Use ASM (Automatic Storage Management)
  - Dynamically add and use disk capacity
- Cluster and grid computing (RAC)
  - Dynamically add and remove processing nodes
  - Parallel processing and parallel queries
- Exadata Database Machines
  - Offload queries and I/Os to dedicated hardware

Open Source Enhancements

- Development of LibLAS library
  - Write SDO_PC from LAS
  - Write LAS from SDO_PC
  - Extend LibLAS to read other formats
    - TerraSolid .bin files
    - Others …..
  - Encourage 3rd party developers to make use of library
- Coordination with GDAL/OGR
  - LibLAS can currently write to OGR data types
  - GDAL can read/write SDO_RASTER
  - Enable GDAL to read from LibLAS as an OGR type

ALBERT GODFRIND

Spatial Solutions Architect

Geospatial and Multimedia Technologies

Data Server Division

Oracle Corporation

Greenside
400 av. Roumanille - BP 309
06906 Sophia-Antipolis
France

phone **+33 4 93.00.88.91**
fax **+33 4 93.00.88.44**
mobile **+33 6 09.97.27.23**
albert.godfrind@oracle.com
http://www.oracle.com/