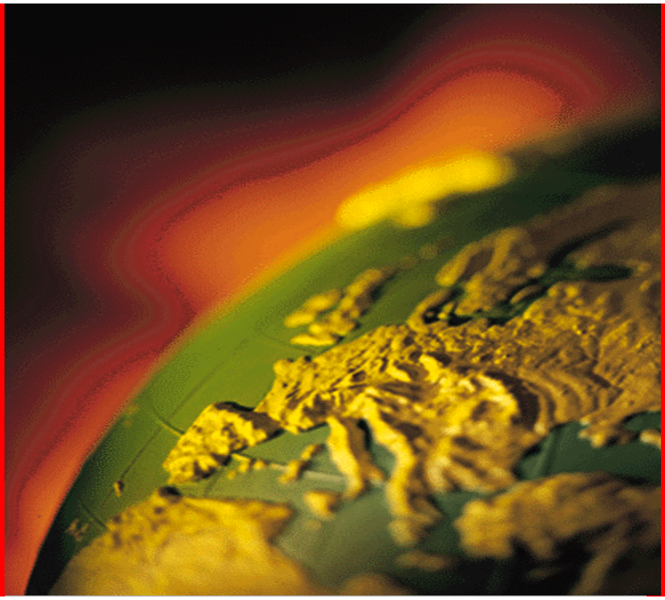


ORACLE®

Introduction to performance of Oracle Spatial databases

Albert Godfrind
Geospatial and Multimedia Technologies
Oracle Corporation

OGh

Oracle gebruikersclub Holland
onderdeel van Oracle Benelux User Group

Agenda

- What is performance anyway ?
- What affects database performance
- Methods and techniques
- Dealing with large volulmes
- And what about rasters ?
- New 11g types – Point clouds and tins
- Conclusion

What is “Performance” ?

What is “good” performance ?

Different meaning for different people

- “Batch” operations
  - Throughput
  - Turn-around time
- Interactive access
  - Response time
  - When is a (spatial) request “complete” ?
    - Results start to appear ?
    - All results are returned ?
- A response time of 1 minute …
  - Is acceptable for a batch report
  - Is unacceptable for an online HTML report

Performance is NOT important!

- Price / Performance is important
  - Provide the right performance at the right cost
- Scalability is even more important
  - Must be able to support larger workloads by increasing hardware capacity

Economics of System Performance

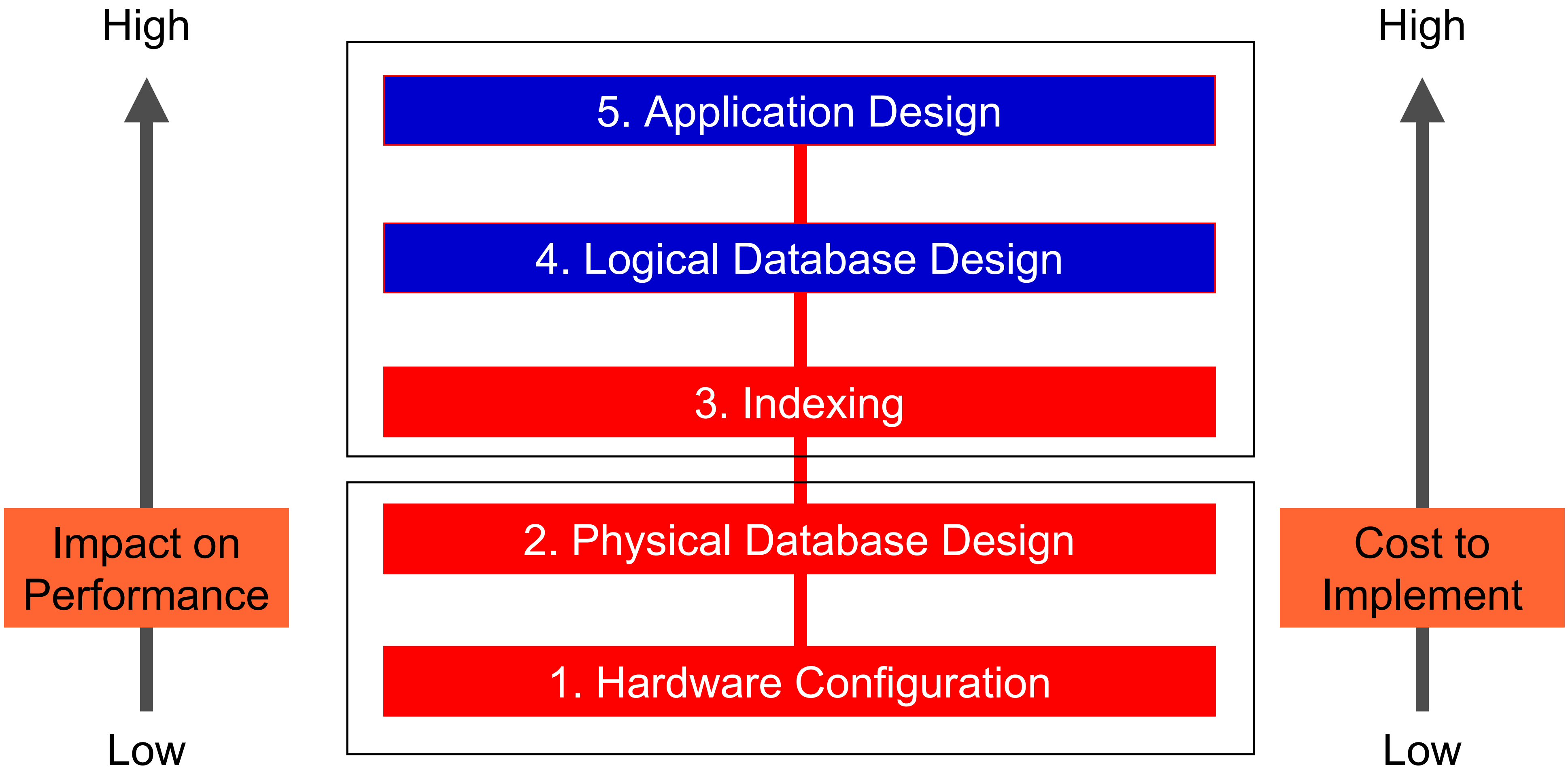
Two common approaches for solving performance problems

1. Throw *hardware* at the problem
   - BMM (Buy More Memory)
   - Replace CPUs with faster ones
   - More CPUs: multi-core, SMP, RAC
   - …
2. Throw *people* at the problem
   - Change database design
   - Change application
   - …

Consider this:

How many man-days can I afford for the cost of an additional CPU ?

What Affects Database Performance



Design *before* you Tune !

- Know your application
  - Transaction mix (batch / online)
  - Kind of queries (simple/complex)
  - …
- Know your data
  - Type of features
  - Complexity
  - Projections
  - …

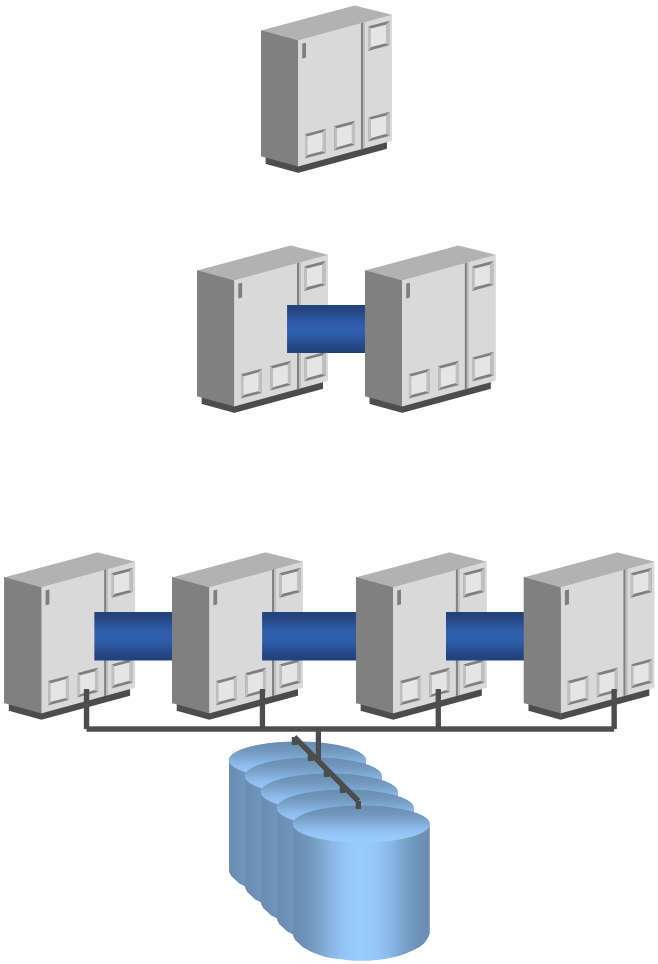
Benchmark *before* you Deploy !

- Use a representative sample of the actual data
  - Good mix of geometries of various complexity
- Spatial processing proportional to geometry complexity
  - If too simple geometries: will underestimate hardware needs
  - If too complex: will overestimate needs

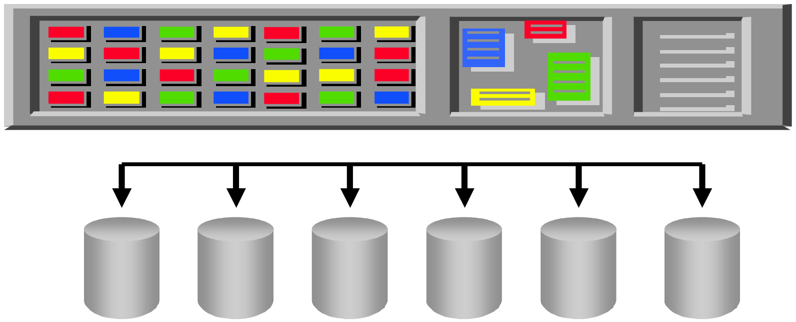
Database Tuning Methods

1. Hardware Configuration

- What you change …
  - Add a CPU board
  - Add more memory
  - Use faster disks
  - …
- Minimal impact on application availability
- Possible only if application scales!

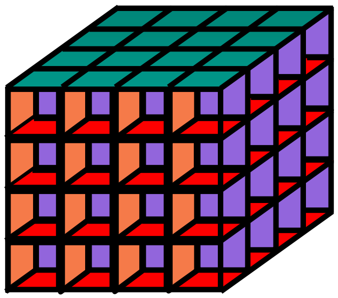


2. Physical Database Design



- What you change …
  - Block size
  - Object distribution in tablespaces
  - Tablespace distribution on disks
  - Buffer cache and memory pools
  - “Pinning” objects in memory
  - …
- Needs some application/database shutdown
- File copies, export/import, etc …

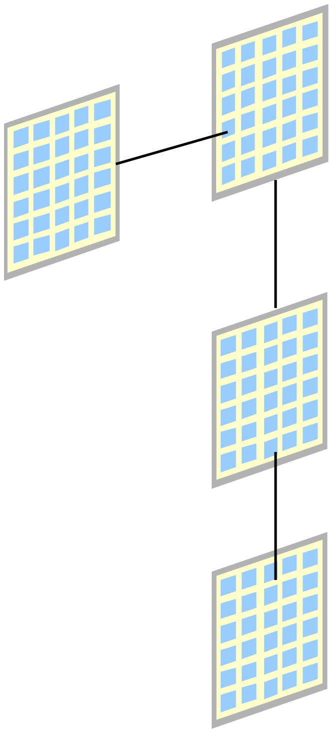
3. Indexing



- What you change …
  - Create new indexes
  - Remove / alter existing indices
  - Table and index partitioning
- *Single most important factor that affects performance before touching application or logical database design.*
- Requires a knowledge of application design

4. Logical Database Design

- What you change …
  - Normalize or de-normalize tables
  - Split / combine tables
  - All features in one table vs. multiple tables
- Major impact on applications!
- Complex implementation
  - Need full re-test
- Can have a large pay-off



Know Your Application

- Cartography ?
  - Your goal is to produce detailed maps as quickly as possible
- Spatial Analysis ?
  - Your goal is to find relationships between spatial features (topology, distance, closeness, etc.)

Application needs and Logical Database Design

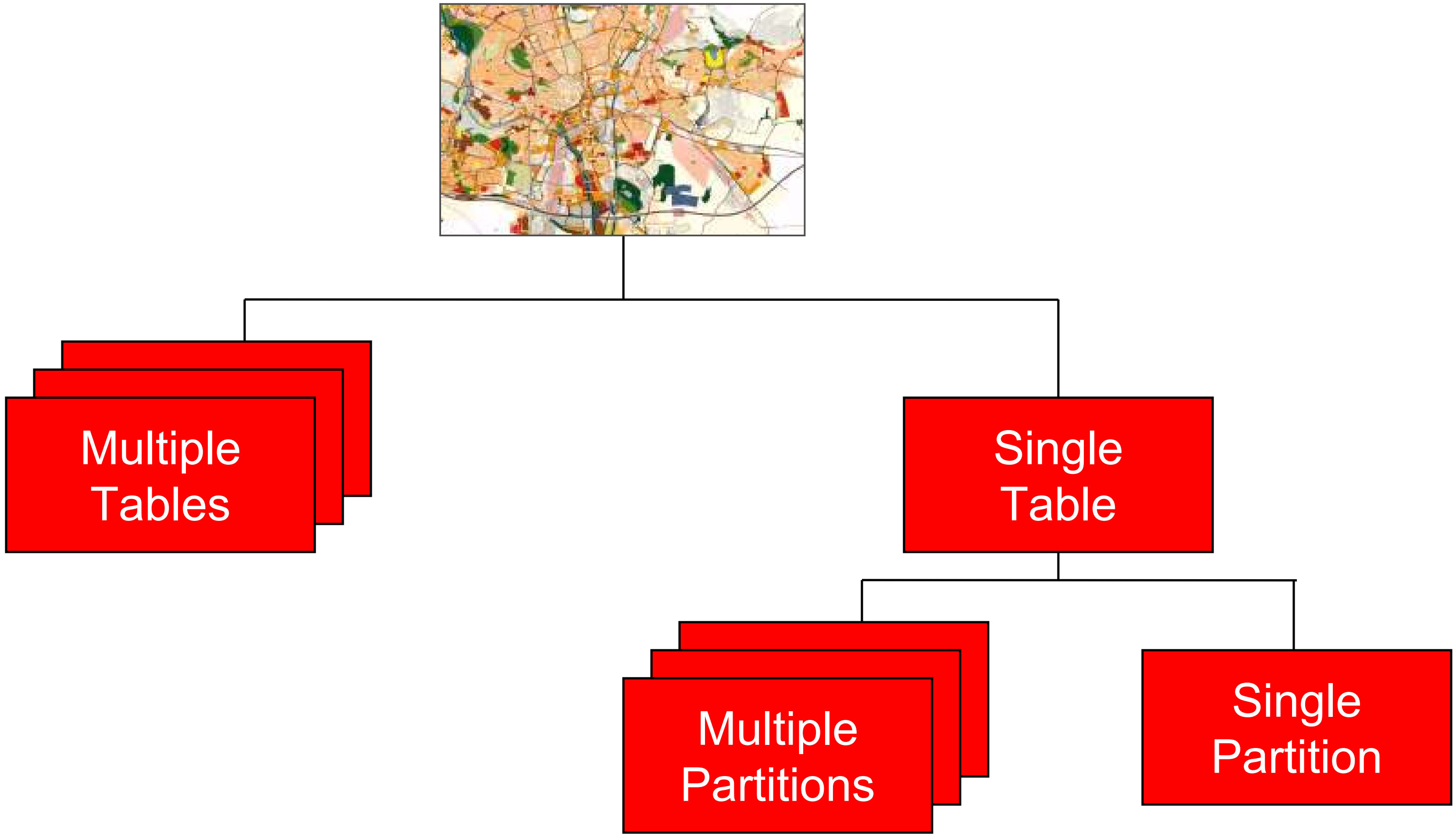
How do you model your spatial features as database tables ?

- One extreme:
  - All spatial layers are in a single table
  - Good if your goal is to generate maps quickly
  - Few database queries
- Other extreme:
  - Each spatial layer is a separate table
  - Good if you perform complex queries

Know Your Data

- Volume
  - Sizing
  - May need to partition
- Complexity
  - May need multiple representations
  - Different scale levels
  - Different coordinate systems

Layers vs. Tables vs. Partitions



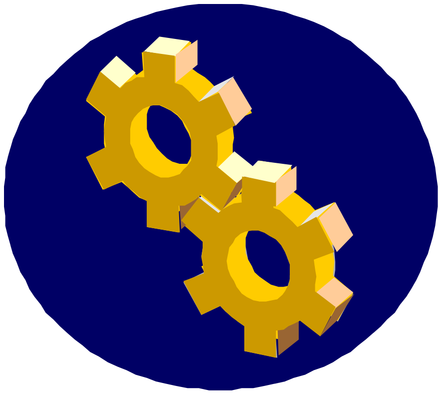
Sizing

- Data
  - Geometries difficult to size
  - Storage space depends on complexity (number of vertices)

  ⇨ *Load a subset and extrapolate*
- Indexes
  - Rtree indexes are easy to size
  - Storage space depends on number of objects

  ⇨ *Pre-compute the size*

5. Application Design



- What you change …
  - The way the application uses the database
  - SQL syntax
  - Relational vs. navigational queries
  - SQL vs. PL/SQL
- May not be able to do anything
  - If using "off-the-shelf" tools!

Types of Spatial Queries

- Simple queries
  - Rectangular filter queries
  - Display: pan, zoom
- Complex
  - Relate, within_distance, nearest neighbors
  - Spatial joins
  - Spatial calculations
- Mixing spatial and other predicates
  - Spatial predicates should be applied first by query optimizer
  - May need “hints” if not

Write your queries the right way !

- General syntax:
  - SDO_INSIDE (geometry_1, geometry_2) = ‘TRUE’
  - Find all “geometry_1” that are inside “geometry_2”
- “geometry_1” is the column you search
  - *Must appear* ***first*** *in your query*
- “geometry_2” is your query “window”
  - *Must appear* ***second*** *in your query*
- Do not swap them
  - May still get the right results
  - But will be much slower!

Write your queries the right way !

- Installing a mobile telephone antenna less than 50 meters from a school is prohibited …
- “*find all schools that are less than 50 meters from a specific antenna*”

```
SELECT s.school_id, …
FROM schools s, antennas a
 WHERE a.antenna_id = :1
   AND SDO_WITHIN_DISTANCE(
        s.location, a.location, ‘distance=50’) = ‘TRUE’;
```

- “*find all antennas that are less than 50 meters from a specific school*”

```
SELECT s.antenna_id, …
FROM schools s, antennas a
 WHERE a.school_id = :1
   AND SDO_WITHIN_DISTANCE(
        a.location, s.location, ‘distance=50’) = ‘TRUE’;
```

A good spatial query is a query you avoid !

- Pre-compute spatial relations and store them
- For example: Administrative boundaries
- Example:
  - A US county is always in a state
  - Just do a relational join between counties and states
- Example:
  - A customer belongs to one sales region
  - Store the sales region id in the customer table
  - Fill once using a spatial query
  - Update automatically via trigger

Combining spatial and non-spatial predicates

- Example: “*get all customers in a given age range that live within a given region*”
- If age range is wide and region is narrow, the right choice is to use the spatial index first
- If age range is very narrow and the region is wide, the right choice is probably to to use the “age” index first
- If the optimizer does not do what you want, use hints

```
SELECT /*+ index (customers age_index) */ …
  FROM customers
 WHERE age between :1 and :2
   AND SDO_WITHIN_DISTANCE(
         location, :3, ‘distance=:4 unit=km’) = ‘TRUE’;
```

A Word on Statistics

- All queries use the “cost-based” optimizer
- The optimizer chooses the optimal query plan and access paths based on the selectivity of the predicates
- Selectivity determined by statistics
  - Distribution of values in each column (histograms)
  - Statistics are automatically collected and maintained by the database
- No statistics available for *spatial* data distribution
  - Spatial predicates use a fixed (hard-coded) selectivity

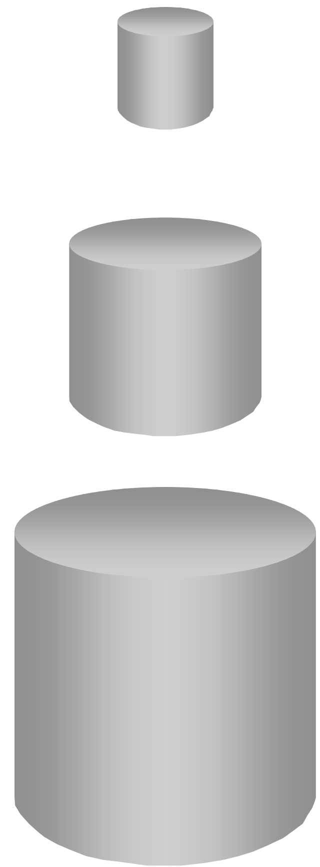
Coordinate Systems and Performance

- Option 1: All data in the same coordinate system
  - No transformations necessary
- Option 2: Multiple coordinate systems
  - Transform on the fly
  - Hold multiple representations
  - Trade off: CPU vs. storage
- Spatial queries in projected vs geodetic data
  - Projected significantly faster
  - Simpler maths, uses less CPU

Large Volumes of Data

Dealing with large data volumes

- How *large* is *large* ?
  - 100’s of thousands is normal
  - Millions is interesting
  - 10’s of millions is serious
  - 100’s of millions is large
  - Billions is very large
- What *is* the problem with large volumes ?
  - They mean big structures
    - Cumbersome to manage
  - Long operations
    - Data reload, refresh
    - Index rebuilds
    - Archiving, …



Partitioning: Divide and Conquer

Two reasons for partitioning

For manageability

- Break large problems into manageable pieces
- Can load / rebuild individual partitions
- Can load / rebuild multiple partitions concurrently

For performance

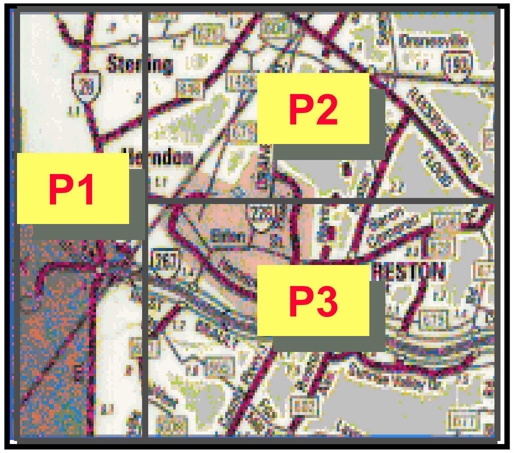
- Query parallelism
- Partition elimination

- Can partition tables, or indexes, or both
  - Also spatial indexes
- Transparent to applications
  - But: Choosing a good partitioning criteria is critical

Partitioning a Spatial Index

- Solves the problem of building large spatial indexes
- Building multiple small R-trees is faster than building a single large R-tree
- Can build the index in parallel mode
  - Multiple index partitions built in parallel
- Can build the index one partition at a time
  - In case of failure, only rebuild the failed partition
- Can use "partition exchange" mechanism
  - To swap in a new set of data very quickly
- Applications can use query parallelism
  - When searching multiple partitions

Spatial Partitioning

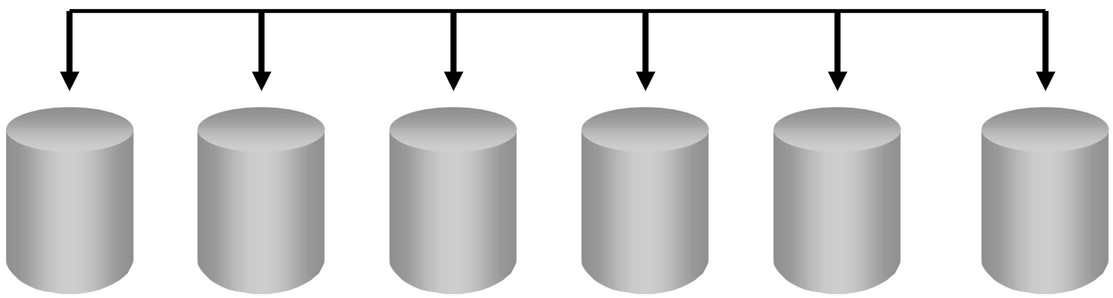


Some assembly required!

- Define a partition map
  - A set of geographical regions
- Add a *partition key* column to the table
- Associate a geometry with a region
  - Based on centroid, first/last vertex, largest area, …
  - Fill the *partition key* from the region id
- Partition according to the *partition key*
- Automate using a trigger

Spatial partitioning and performance

- Always need to check all partitions
- Partition elimination done by indexing code
- Cost per partition is minimal (1ms per partition – 1.5Ghz CPU)
- But: can still add up if lots of partitions!



Other Notes

Spatial Clustering

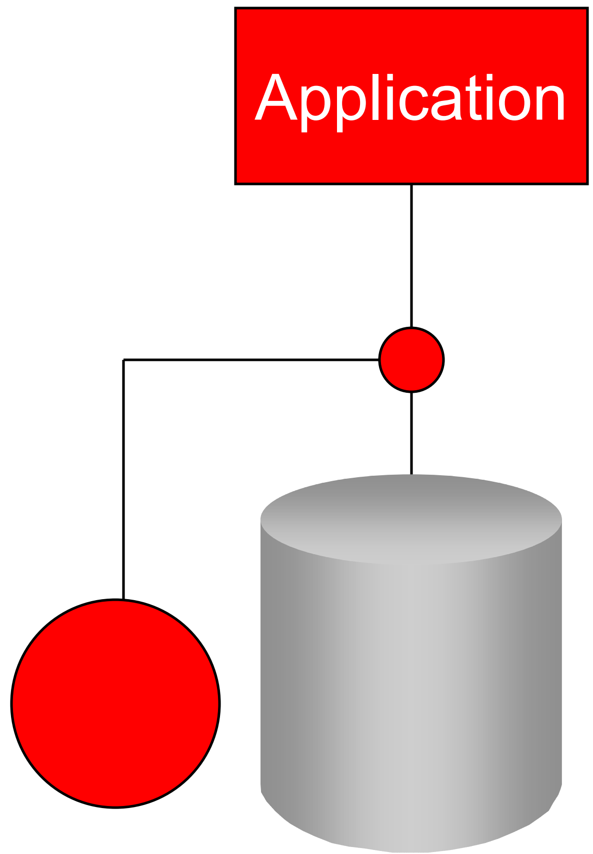
"Spatially" close features are stored "physically" close

Some assembly required!

- Add a *cluster key* column to the table
- Derive a *cluster key* from each geometry
  - Based on centroid, hhcode, …
- Sort by cluster key and load
- Automate using a trigger
- Clustering is *not* maintained – need to reorganize

Tracing

- Need to find out about the SQL queries sent to the database
- Useful to reproduce / diagnose problems
- Can use your own logging
- Can use Oracle's "SQL Trace" facility
- Tracing another process can be tricky
  - Identify the connection to trace
  - May use multiple connections
  - May start and stop connections dynamically



Data Quality

- Geometries can contain errors
- Most common errors:
  - “Duplicate” points
  - Incorrect polygon orientation
  - Self-touching polygons
- No impact on performance, but
  - Queries may fail
  - Queries may return wrong results
  - Spatial calculations may loop …
- Oracle 10g and 11g less permissive than 9i

Data Quality

- No automatic validation!
- Use the validation procedures
  - VALIDATE_GEOMETRY_WITH_CONTEXT()
- Correct the errors
  - Common errors correctable
  - RECTIFY_GEOMETRY()

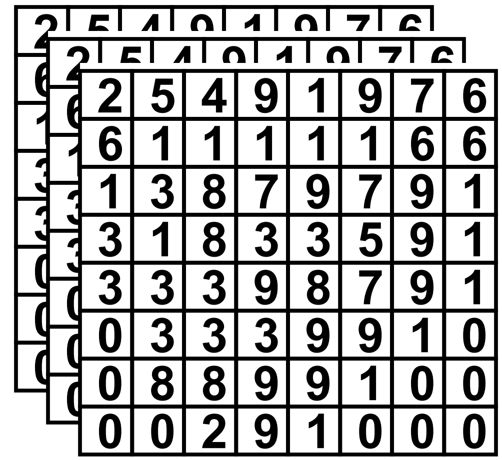
Rasters

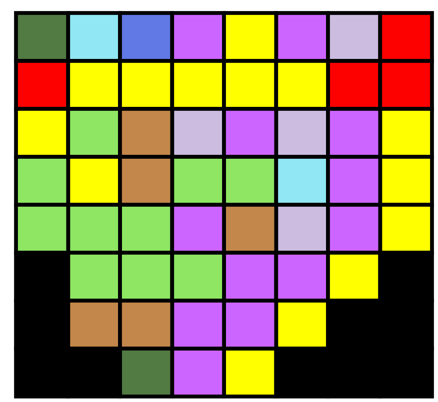
And what about rasters ?

- Raster data typically means imagery: satellite or aerial
- Large volumes
  - Counting in terabytes instead of gigabytes
  - GeoTIFF Images of 500MB are common
- Important parameters:
  - Database design: logical vs physical
  - Raster blocking
  - Pyramiding
  - Compression
- Loading
  - Or: "*How long does it take me to move a ton of bricks ?* "

What is a Raster ?

- Two dimensional array of regularly spaced elements (pixels or cells)
  - Orthophotos
  - Remote Sensing
  - Gridded data (raster GIS)
- Image data is collected by a variety of technologies
  - Satellite remote sensing
  - Airborne photogrammetry
  - Sonar
- Digital images can be composed of one or more bands
  - Bands often represent an interval of wavelengths along the electromagnetic spectrum
  - Band data can be simultaneously recorded



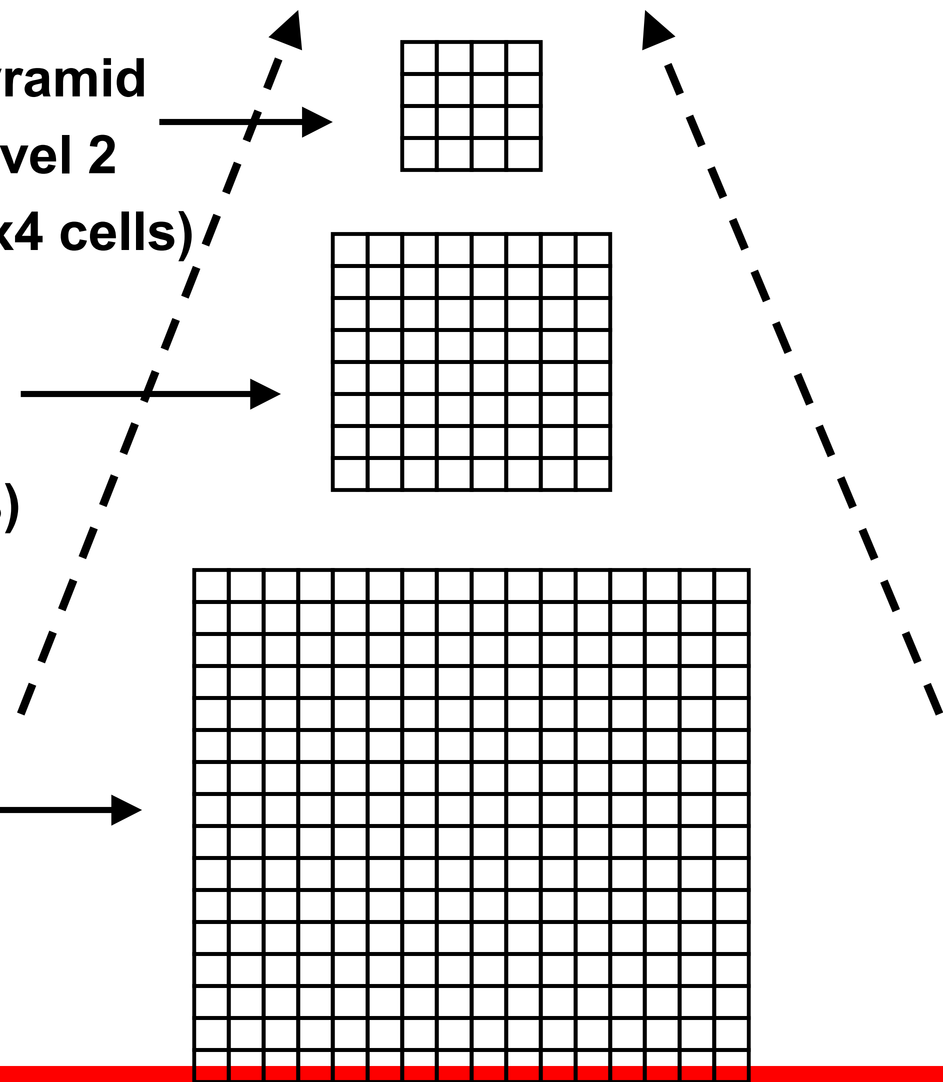


Resolution Pyramid

Pyramid Level 2 (4x4 cells)

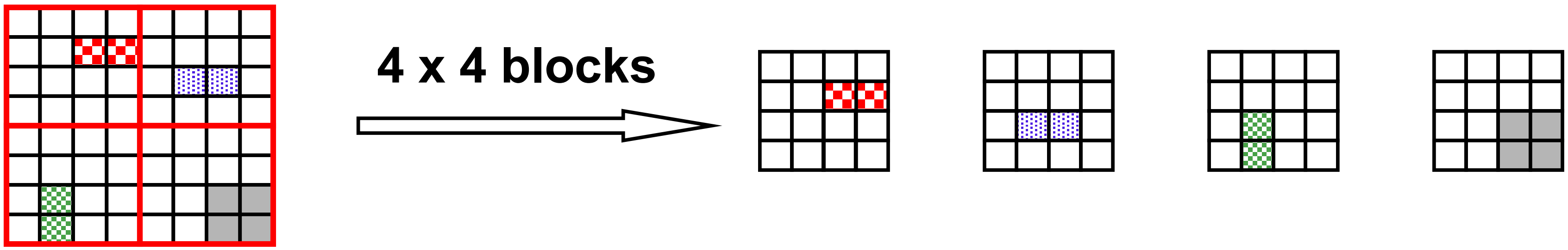
Pyramid Level 1 (8x8 cells)

Pyramid Level 0 (16x16 cells)



Blocking

- A GeoRaster image can be composed of an extremely large number of cells
- It is more efficient in terms of storage and retrieval to break large images into smaller blocks
- In GeoRaster, users/applications can determine how data is blocked
  - Specify rows, columns, and optionally bands



Loading Large Rasters …

Or: how long does it take to shift a ton of bricks ?

- Using one wheelbarrow ?
- Using several wheelbarrows ?
- But is the door wide enough ?

Storage Model

Separates Logical from Physical structures

Logical structures

Contains raster metadata and footprint

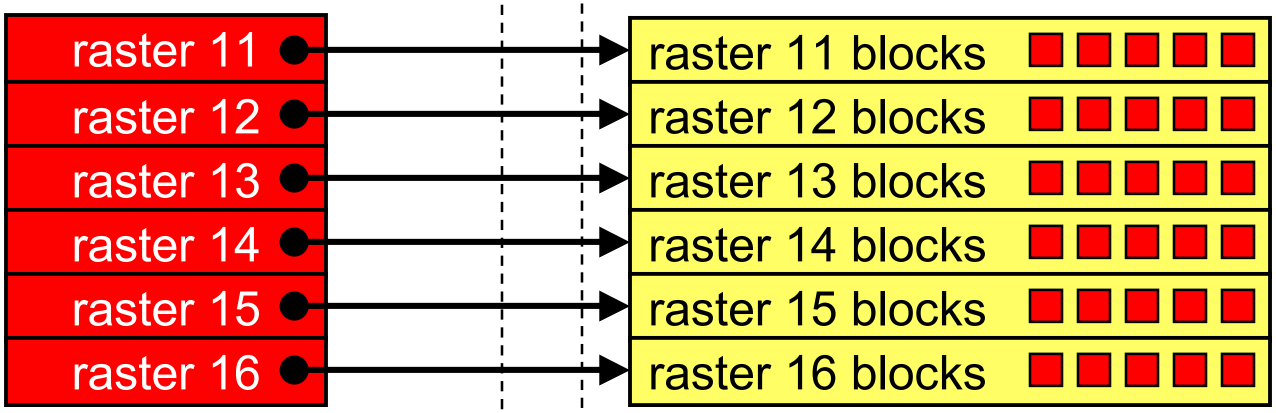
Also contains a pointers to one or more RDT's

Physical structures

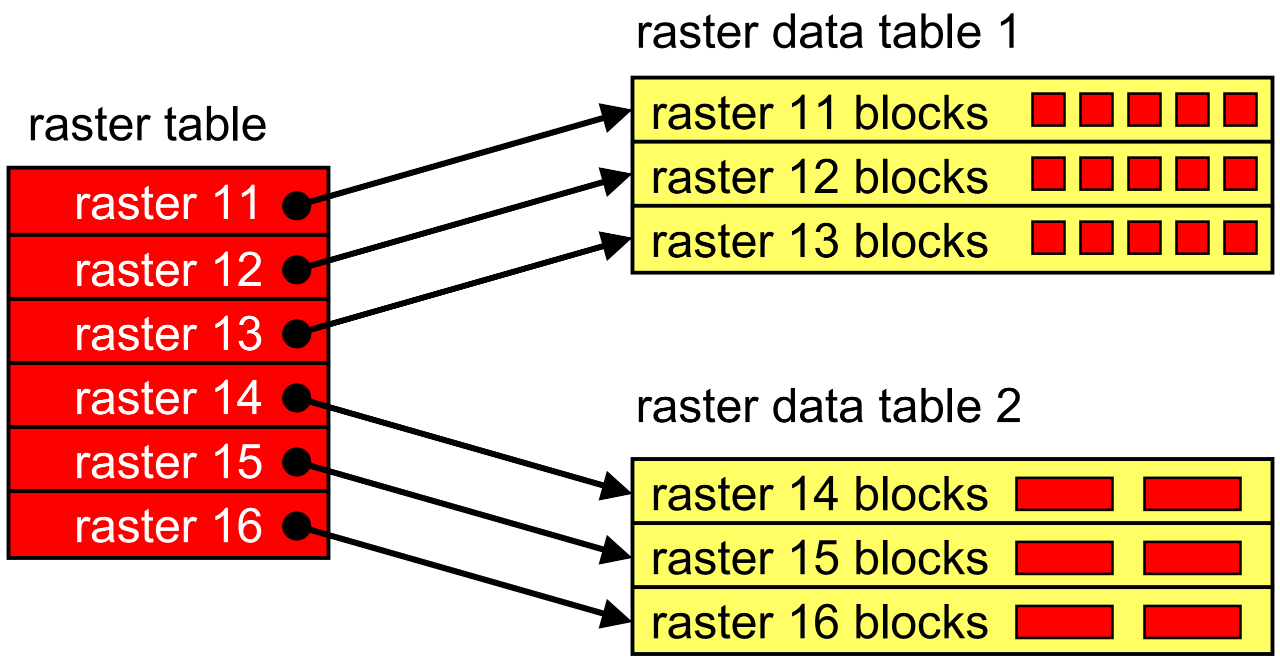
"Raster data tables" or RDTs

Contain raster pixels

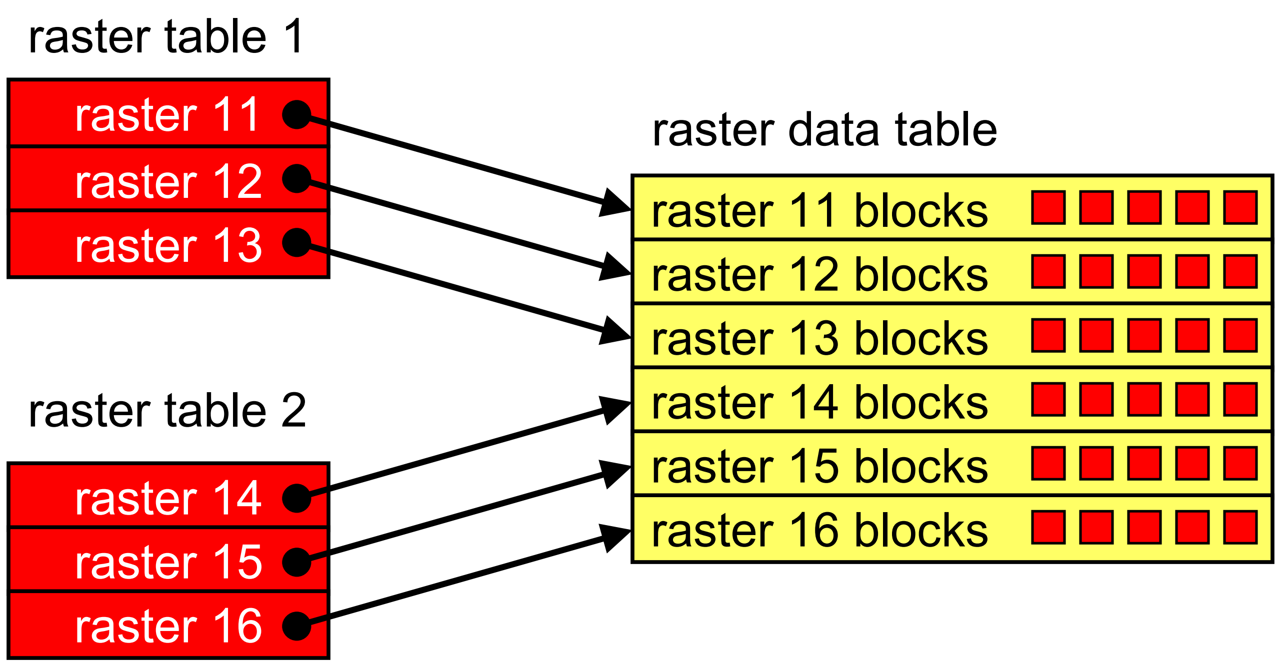
Can be very large



Storage Model

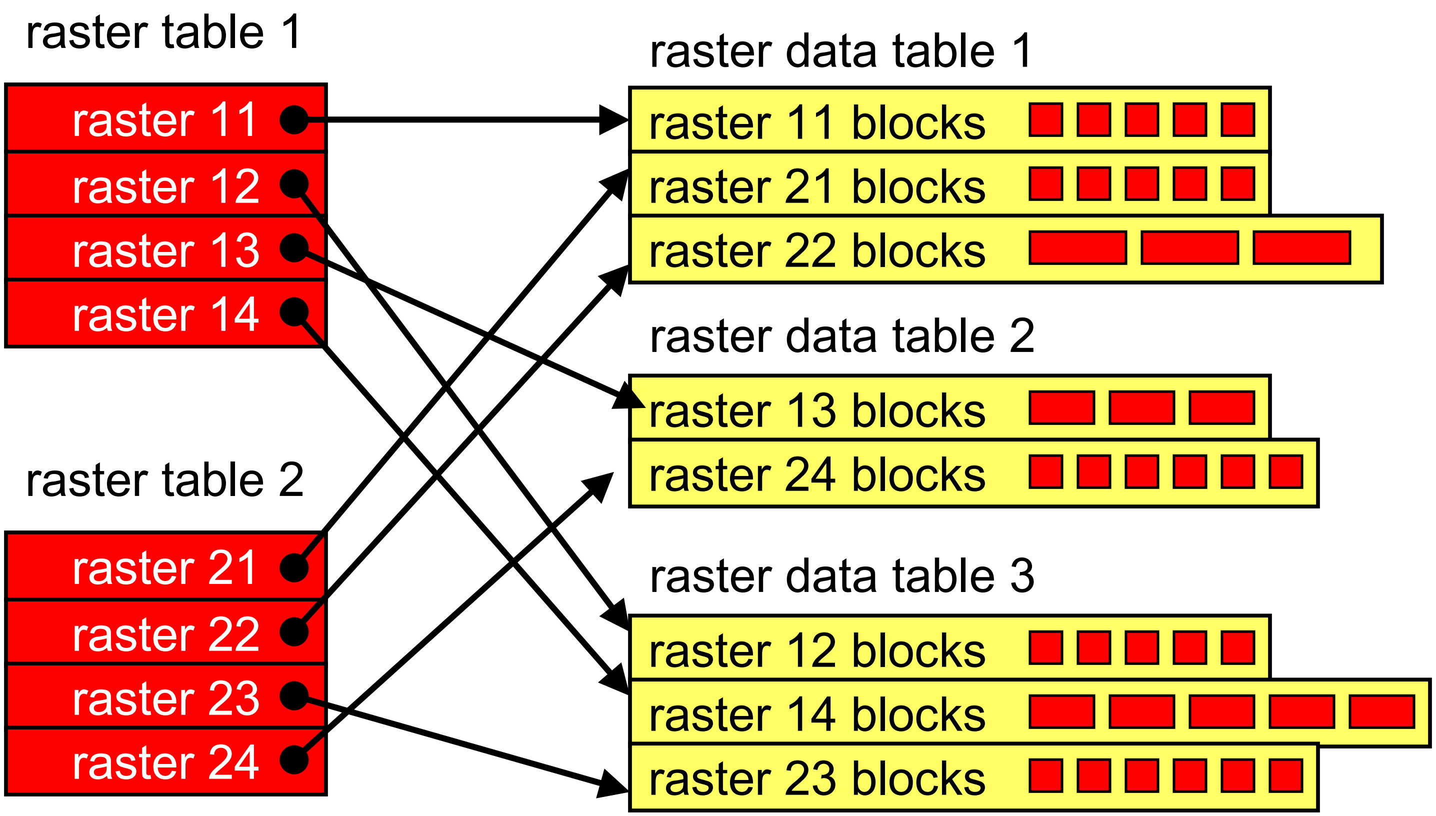


Storage Model



Storage Model

A more complex example



GeoRaster Compression

JPEG Compression

- Lossy compression
- For rasters with cellDepth=8BIT_U and no more than 4 bands per block
- JPEG-B or JPEG-F mode
- Control the compression level using the quality parameter
  - 0 (max compression) to 100 (no compression)

DEFLATE Compression

- Lossless compression
- Uses the ZLIB format

JPEG2000 and MrSid Compression

- Via a plugin from LizardTech
- Also loader for JP2/SID files

New 3D Types (11g)

- Point Clouds (LIDAR)
- TINS

- Both can be very large
- 100's of millions of points to describe a scene
- Storage mechanism similar to GeoRaster
  - Separates logical from physical storage
  - Blocking of physical storage

Conclusion

- Spatial is not different
  - All well-known design and tuning principles apply
- Spatial is different
  - Complexity
  - Spatial clustering and partitioning
  - Spatial indexing

"If you know how to use Oracle, you know 80% of how to manage spatial data with Oracle Spatial"

ALBERT GODFRIND

Spatial Solutions Architect

Geospatial and Multimedia Technologies

Data Server Division

Oracle Corporation	**Greenside**	**phone**	**+33 4 93.00.88.91**
	400 av. Roumanille - BP 309	**fax**	**+33 4 93.00.88.44**
	06906 Sophia-Antipolis	**mobile**	**+33 6 09.97.27.23**
	France	**albert.godfrind@oracle.com**	
		http://www.oracle.com/	