Chapter X

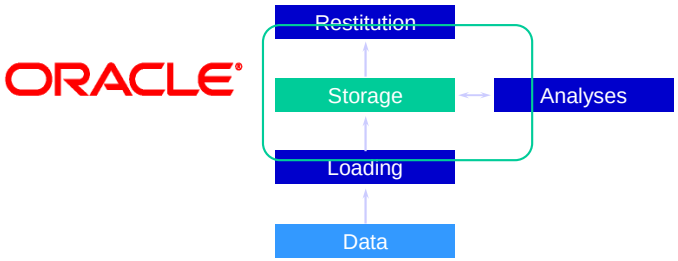
ORACLE Locator and ORACLE Spatial

Acknowledgement: Albert Godfrind, Oracle

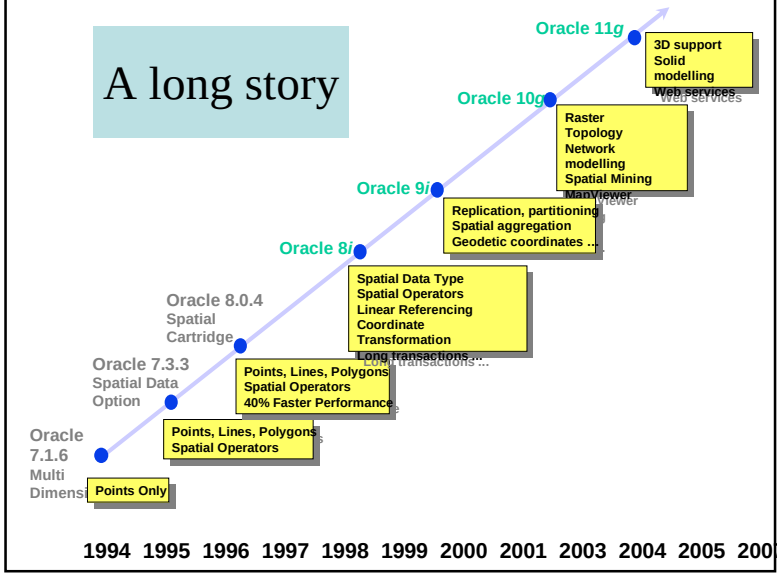
X – ORACLE Locator and Spatial

- 10.1 – Generalities
- 10.2 – Geometric Data Model
- 10.3 – Loading and indexing
- 10.4 – Spatial queries and analyses
- 10.5 – Geo Raster
- 10.6 – Network modeling
- 10.7 – MapViewer
- 10.8 – Oracle Spatial by Example
- 10.9 – Conclusions

Oracle and GIS

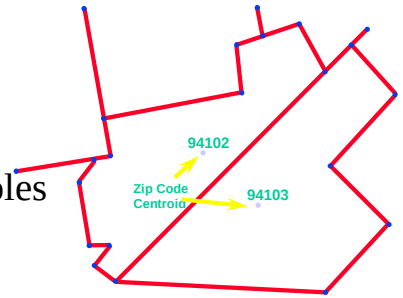


A long story



Example of geometric data

- Points
- Polylines
- Polygons
- Polygons with holes
- Circles
- Arcs of circles
- Composite Geometries



Oracle Locator

- All geometric objects
  - Points, lines, polygons
  - 2D, 3D, 4D
- Indexing (quadtrees and *r*-trees)
- Spatial queries
- Proximity queries
- Distances
- Projections

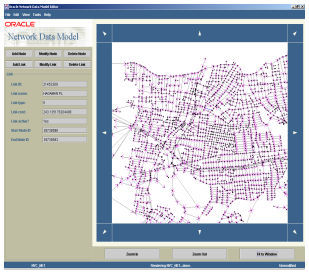
Oracle Spatial

= Locator + ...

- Geometric Transformations
- Spatial Aggregations
- Network Modeling
- Topology
- Raster
- Geocoder
- Spatial Data Mining
- 3D Types (LIDAR, TINS)
- Web Services (WFS, CSW, OpenLS)

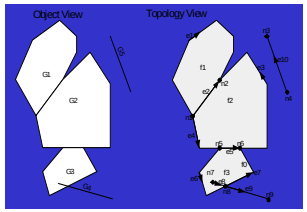
Oracle 11*g*: Networks

- A data model for networks
- Attributes at node and edge levels
- Applications
  - Transport
  - Logistics
  - Telephone, LBS
- Navigation



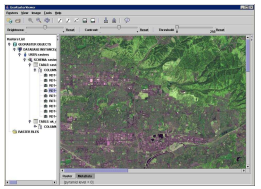
Oracle 11*g*: Topology

- Persistent model
  - Nodes, arcs, faces
  - Topological Relationships
  - Advanced consistency checking
- Data Model
  - Definition of topological primitives
  - New type SDO_TOPO_GEOMETRY
  - Co-existence with conventional data
  - Possibility of using both SDO_GEOMETRY and SDO_TOPO_GEOMETRY



Oracle 11*g*: Geo Raster

- New data type SDO_GEORASTER
  - Orthophotos, remote sensing
    - Multi-band, multi-layer
  - Meta-data in XML
  - Information de geo-referencing
  - GRID Data type
- Functionality
  - Storing and et indexing
  - Tiling
  - Construction of Image Pyramid
  - Selection and analyses
    - Manipulation at pixel level
  - Extraction and mosaicking



10.2 – Geometric Data Model

- A new geometric abstract data type
- Examples

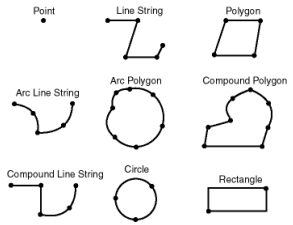
Creation of spatial tables

- Using the SDO_GEOMETRY type
- No limitation
- Any column can content any type of geometry

```
SQL> CREATE TABLE Cells (
  2>   Cell_id       NUMBER,
  3>   Cell_name     VARCHAR2(32),
  3>   Cell_type     NUMBER,
  4>   Location      SDO_GEOMETRY,
  5>   Covered_area  SDO_GEOMETRY);
```

Geometric Types

- Points and sets of points
- Polyline
- Polygon
- Chain of arcs
- Compound Polygons
- Circles
- Optimized rectangles

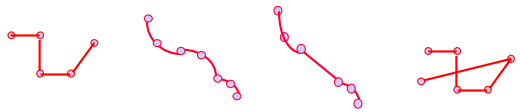


Geometric Primitives : Points

- Points (X_1, Y_1)
- 2, 3 or 4 dimensions

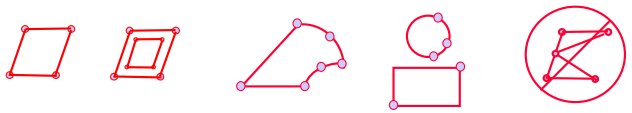
Geometric Primitives: Lines

- Lines (X_1, Y_1, ... X_n, Y_n)
- Can be made of broken lines or arcs of circles
- Do not delimitate areas
- A line can intersect itself

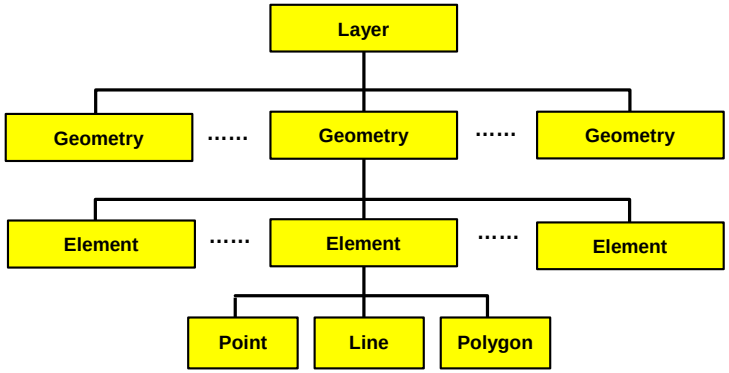


Geometric Primitives: Polygons

- Polygons (X_1, Y_1, ... X_n, Y_n)
- Boundaries must be closed (last point = first point)
- Can have holes
- Boundary does not intersect
- Can be made of broken lines or arcs of circles
- Other specific shapes (rectangle, circle)

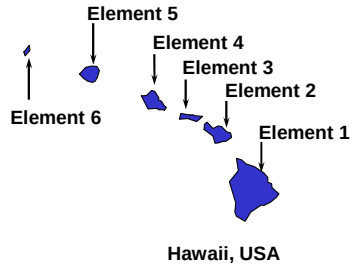


Elements, geometries, layers, …



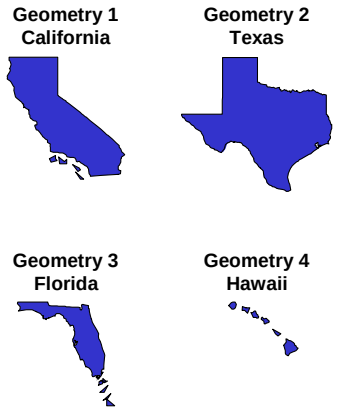
Element

- Constituting any geometric objects
- Element Type :
  - Point
  - Line
  - Polygon
- Ordered set of points



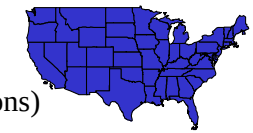
Geometry

- Represents a spatial object
- Made of an ordered set of elements
- Homogeneous or heterogeneous



Spatial layer

- Represent a geometric column within a table
  - Possibility of several geometric columns
- In general contents objects of same nature (= having same attributes)
  - "clients" layer (points)
  - “streets" layer (lines)
  - “municipalities" layer (polygons)



SDO_GEOMETRY type

- Structure of SDO_GEOMETRY

```
SDO_GTYPE          NUMBER
SDO_SRID           NUMBER
SDO_POINT          SDO_POINT_TYPE
SDO_ELEM_INFO      SDO_ELEM_INFO_ARRAY
SDO_ORDINATES      SDO_ORDINATE_ARRAY
```

- Example

```
SQL> CREATE TABLE states (
  2     state       VARCHAR2(30),
  3     totpop      NUMBER(9),
  4     geom        SDO_GEOMETRY);
```

SDO_GEOMETRY

- SDO_POINT_TYPE

```
x        NUMBER
y        NUMBER
z        NUMBER
```

- SDO_ELEM_INFO_ARRAY

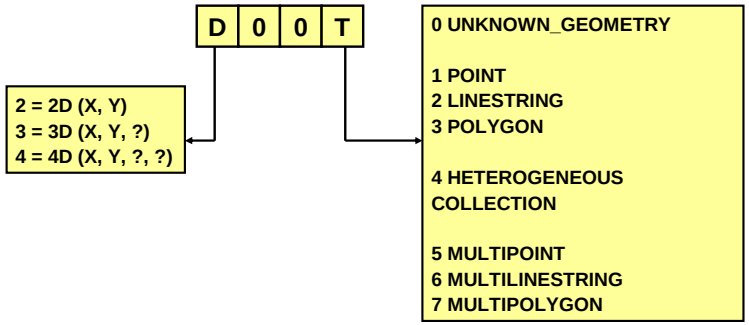
```
VARRAY (1048576) OF NUMBER
```

- SDO_ORDINATE_ARRAY

```
VARRAY (1048576) OF NUMBER
```

SDO_GTYPE

- Geometric nature



SDO_SRID

- SRID = Spatial Reference system ID
- More than 1000 different systems
- A common value: 8307
  - "Longitude/Latitude WGS84"
  - Used by GPS
  - Navteq and TeleAtlas data are WGS84-encoded
- All geometries must have the same SRID
- Different layers can have different SRID
- Automatic conversion when querying

SDO_POINT

- SDO_POINT_TYPE

```
x       NUMBER
y       NUMBER
z       NUMBER
```

- Optimized storage of points

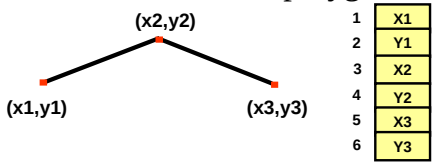
```
SQL> INSERT INTO TELEPHONE_POLES (col-1, …, col-n,
geom)
  2>    VALUES (attribute-1, …, attribute-n,
  3>      SDO_GEOMETRY (
  4>        2001, 8307,
  5>        SDO_POINT_TYPE (-75.2,43.7,null),
  6>        null, null)
  7>    );
```

SDO_ORDINATES

- Object with SDO_ORDINATE_ARRAY type

```
VARRAY (1048576) OF NUMBER
```

- Array of numbers
- Storage of coordinates for lines and polygons



- Two values per point (in 2D)

SDO_ELEM_INFO

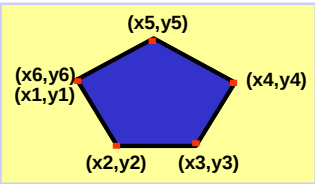
- SDO_ELEM_INFO_ARRAY

```
VARRAY (1048576) OF NUMBER
```

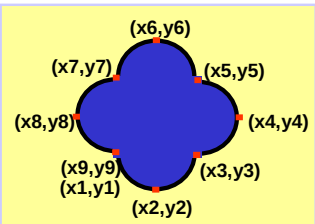
- Describe the components of a complex object
- Three values per element
- Ordinate offset: Position of the first number for this element in the SDO_ORDINATES array
  - Element type
  - Interpretation: Line, circle

Examples of polygons

Ordinate offset	Element type	Interpretation
1	1003	1

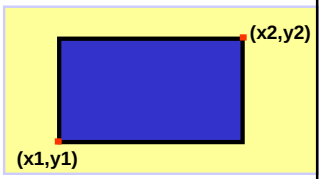


Ordinate offset	Element type	Interpretation
1	1003	2

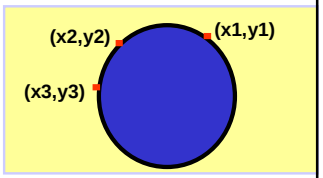


Examples of polygons

Ordinate offset	Element type	Interpretation
1	1003	3

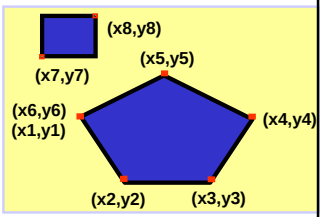


Ordinate offset	Element type	Interpretation
1	1003	4

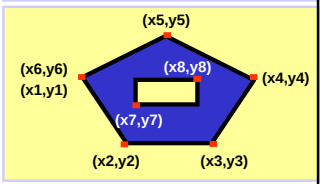


Multi-polygon and polygon with hole

Ordinate offset	Element type	Interpretation
1	1003	1
13	1003	3



Ordinate offset	Element type	Interpretation
1	1003	1
13	2003	3



Creating spatial tables

- Using the SDO_GEOMETRY type

```
SQL> CREATE TABLE Cells (
  2>   Cell_id       NUMBER,
  3>   Cell_name     VARCHAR2(32),
  3>   Cell_type     NUMBER,
  4>   Location      SDO_GEOMETRY,
  5>   Covered_area  SDO_GEOMETRY);
```

Metadata

- Minimum and maximum values for each dimension
- Tolerance for the layer (2 points are identical)
- Referencing system for the layer

```
SQL> INSERT INTO USER_SDO_GEOM_METADATA
  2>  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  3>  VALUES (
  4>   'ROADS',
  5>   'GEOMETRY',
  6>    SDO_DIM_ARRAY (
  7>       SDO_DIM_ELEMENT('Long', -180, 180, 0.5),
  8>       SDO_DIM_ELEMENT('Lat',   -90,  90, 0.5)),
  9>    8307 );
```

Construction of geometries

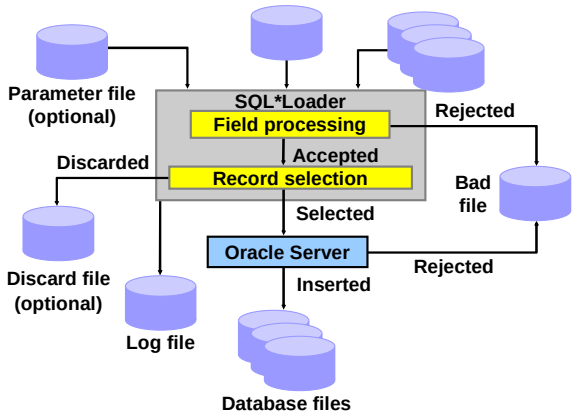
- Constructeur standard

```
SQL> INSERT INTO TELEPHONE_POLES (col-1, …, col-n,
geom)
  2>    VALUES (attribute-1, …, attribute-n,
  3>      SDO_GEOMETRY (
  4>        2001, 8307,
  5>        SDO_POINT_TYPE (-75.2,43.7,null),
  6>        null, null)
  7>     );
```

10.3 – Loading and Indexing

- Formats
- Loading
- Validation
- Creation of spatial indices

Using SQL*Loader



SQL*Loader for Points

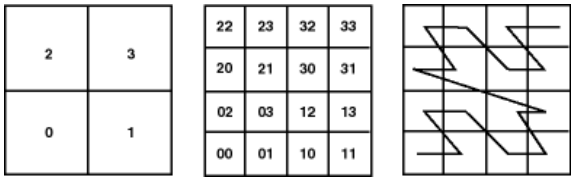
```
LOAD DATA
 INTO TABLE cities
 FIELDS TERMINATED BY '|' (
   CITY,
   STATE_ABRV,
   POP90,
   RANK90,
   LOCATION COLUMN OBJECT (
     SDO_GTYPE       INTEGER EXTERNAL,
     SDO_POINT COLUMN OBJECT (
       X             FLOAT EXTERNAL,
       Y             FLOAT EXTERNAL )
   )
 )
```

```
New York|NY|7322564|1| 2001|-73.943849000|40.669800000|
Los Angeles|CA|3485398|2| 2001|-118.411201000|34.112101000|
Chicago|IL|2783726|3| 2001|-87.684965000|41.837050000|
Houston|TX|1630553|4| 2001|-95.386728000|29.768700000|
Philadelphia|PA|1585577|5| 2001|-75.134678000|40.006817000|
San Diego|CA|1110549|6| 2001|-117.135770000|32.814950000|
```

R-tree

Minimum Bounding Rectangle Principle of indexing

Quadtree



Quadtree with Peano keys (Morton code)

HH codes

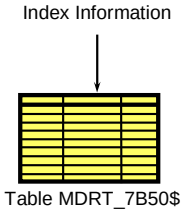
- HHCODEs (Helical Hyperspatial Codes)
- Peano space-filling curves
- Longitude/latitude/altitude/time

Selecting an index type

R-tree Indexing	Quadtree Indexing
The approximation of geometries cannot be fine-tuned. (Spatial uses the minimum bounding rectangles.	The approximation of geometries can be fine-tuned by setting the tiling level and number of tiles.
Index creation and tuning are easier.	Tuning is more complex, and setting the appropriate tuning parameter values can affect performance significantly.
Less storage is required.	More storage is required.
If your application workload includes nearest-neighbor queries (SDO_NN operator), R-tree indexes are faster.	If your application workload includes nearest-neighbor queries (SDO_NN operator), quadtree indexes are slower.
If there is heavy update activity to the spatial column, an R-tree index may not be a good choice.	Heavy update activity does not affect the performance of a quadtree index.
You can index up to four dimensions.	You can index only two dimensions.
An R-tree index is recommended for indexing geodetic data if SDO_WITHIN_DISTANCE queries will be used on it.	
An R-tree index is required for a whole-earth index.	

Creation R-tree-based index

```
create index CUSTOMERS_SIDX
   on CUSTOMERS (LOCATION)
 indextype is MDSYS.SPATIAL_INDEX;
```



Syntax for creating an index

```
CREATE INDEX <index-name>
  ON <table-name> (<column-name>)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX
  [PARAMETERS (
   'SDO_RTR_PCTFREE = <param_value>
    <storage_parameters> = <param_value> ... ')
  ] [PARALLEL [<parallel_degree>]];
```

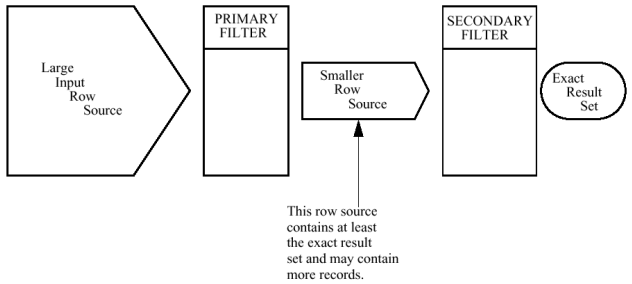
- More time-consuming than a normal index
- Depends from the number of objects

10.4 – Spatial queries and analyses

- Query execution
- Search based on spatial relations
- Search based on distances
- Search based on proximity
- Spatial joins
- Spatial functions
- Spatial aggregations

Query processing

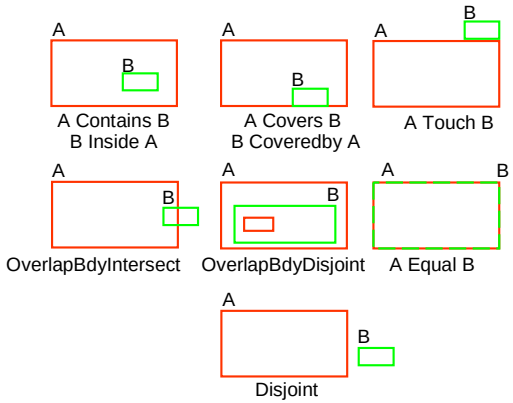
Figure 3–1 Query Model



Spatial queries

- With spatial predicates (Where Clause)
- Specific ORACLE operators :
  - SDO_INSIDE, SDO_TOUCH
  - SDO_WITHIN_DISTANCE
  - SDO_NN
  - Etc.
- The spatial index must exist… otherwise…

Topological Relations



Topological Predicates

- SDO_INSIDE
- SDO_CONTAINS
- SDO_COVERS
- SDO_COVEREDBY
- SDO_OVERLAPS
- SDO_TOUCH
- SDO_EQUAL
- *SDO_ANYINTERACT*

```
WHERE SDO_INSIDE ( <geometry-1>, <geometry-2> ) = 'TRUE
```

Function	Description
SDO_GEOM.RELATE	Determines how two objects interact.
SDO_GEOM.SDO_ARC_DENSIFY	Changes each circular arc into an approximation consisting of straight lines, and each circle into a polygon consisting of a series of straight lines that approximate the circle.
SDO_GEOM.SDO_AREA	Computes the area of a two-dimensional polygon.
SDO_GEOM.SDO_BUFFER	Generates a buffer polygon around a geometry.
SDO_GEOM.SDO_CENTROID	Returns the centroid of a polygon.
SDO_GEOM.SDO_CONVEXHULL	Returns a polygon-type object that represents the convex hull of a geometry object.
SDO_GEOM.SDO_DIFFERENCE	Returns a geometry object that is the topological difference (MINUS operation) of two geometry objects.
SDO_GEOM.SDO_DISTANCE	Computes the distance between two geometry objects.
SDO_GEOM.SDO_INTERSECTION	Returns a geometry object that is the topological intersection (AND operation) of two geometry objects.
SDO_GEOM.SDO_LENGTH	Computes the length or perimeter of a geometry.
SDO_GEOM.SDO_MAX_MBR_ORDINATE	Returns the maximum value for the specified ordinate of the minimum bounding rectangle of a geometry object.
SDO_GEOM.SDO_MBR	Returns the minimum bounding rectangle of a geometry.
SDO_GEOM.SDO_MIN_MBR_ORDINATE	Returns the minimum value for the specified ordinate of the minimum bounding rectangle of a geometry object.
SDO_GEOM.SDO_POINTONSURFACE	Returns a point that is guaranteed to be on the surface of a polygon.
SDO_GEOM.SDO_UNION	Returns a geometry object that is the topological union (OR operation) of two geometry objects.
SDO_GEOM.SDO_XOR	Returns a geometry object that is the topological symmetric difference (XOR operation) of two geometry objects.
SDO_GEOM.VALIDATE_GEOMETRY	Determines if a geometry is valid.
SDO_GEOM.VALIDATE_LAYER	Determines if all the geometries stored in a column are valid.
SDO_GEOM.WITHIN_DISTANCE	Determines if two geometries are within a specified Euclidean distance from one another.

SDO_RELATE

SDO_GEOM.RELATE

Purpose

This function examines two geometry objects to determine their spatial relationship.

Syntax

SDO_GEOM.RELATE (*layername1, SDO_GID1, mask, [layername2,] SDO_GID2*)

SDO_GEOM.RELATE (*layername1, SDO_GID1, mask, X_tolerance, Y_tolerance, SDO_ETYPE, num_ordinates, X_ordinate1, Y_ordinate1 [,...,Xn, Yn] [,SDO_ETYPE, num_ordinates, X_ordinate1, Y_ordinate1 [,...,Xn, Yn]]*)

- ANYINTERACT - Returns TRUE if the objects are not disjoint.
- CONTAINS - Returns TRUE if the second object is entirely within the first object and the object boundaries do not touch.
- COVEREDBY - Returns TRUE if the first object is entirely within the second object and the object boundaries touch at one or more points.
- COVERS - Returns TRUE if the second object is entirely within the first object and the boundaries touch in one or more places.
- DISJOINT - Returns TRUE if the objects have no common boundary or interior points.
- EQUAL - Returns TRUE if the objects share every point of their boundaries and interior, including any holes in the objects.
- INSIDE - Returns TRUE if the first object is entirely within the second object and the object boundaries do not touch.
- OVERLAPBDYDISJOINT - Returns TRUE if the objects overlap, but their boundaries do not interact.
- OVERLAPBDYINTERSECT - Returns TRUE if the object overlap, and their boundaries intersect in one or more places.
- TOUCH - Returns TRUE if the two objects share a common boundary point, but no interior points.

Generic Topological Operator

- SDO_RELATE together with a mask

```
WHERE SDO_RELATE (
     <geometry-1>, <geometry-2>, 'MASK=xxxx' ) =
'TRUE
```

- This mask can be 'INSIDE', 'CONTAINS', 'TOUCH', etc.
- Or a combination: 'INSIDE+COVEREDBY'

Examples

- What are parks totally located within the Wyoming state ?

```
SELECT p.name
  FROM us_parks p, us_states s
 WHERE s.state = 'Wyoming'
   AND SDO_INSIDE (p.geom, s.geom) = 'TRUE';
```

- Equivalent of:

```
   AND SDO_RELATE(p.geom,s.geom,'MASK=INSIDE') =
'TRUE';
```

Examples

- What are states which contents totally or partly the Yellowstone Park ?

```
SELECT s.state
  FROM us_states s, us_parks p
 WHERE SDO_ANYINTERACT (s.geom, p.geom) = 'TRUE'
   AND p.name = 'Yellowstone NP';
```

Spatial join: SDO_JOIN()

- Allows to find correlation between tables (topology or distance)
- Allow to compare all objects of a table with all objects of a second table
- Needs an index for each table
- Returns a table

SDO_JOIN function

```
SDO_JOIN( table_name-1, column_name-1,
          table_name-2, column_name-2
        [,'parameters'] [, preserve_join_order])
RETURN SDO_ROWIDSET;
```

```
SQL> DESC sdo_rowidset;
 SDO_ROWIDSET TABLE OF MDSYS.SDO_ROWIDPAIR
 Name           Null?    Type
 -------------- -------- -----------
 ROWID1                  VARCHAR2(24)
 ROWID2                  VARCHAR2(24)
```

Example:

- Associate to each “GOLD” client his sales region

```
SELECT s.id, c.id, c.name
  FROM customers c,
       sales_regions s,
       TABLE(SDO_JOIN(
             'customers', 'location',
             'sales_regions', 'geom',
             'mask=inside')) j
 WHERE j.rowid1 = c.rowid
   AND j.rowid2 = s.rowid
   AND c.customer_grade = 'GOLD'
 ORDER BY s.id, c.id;;
```

Computing length, area and distance

- SDO_AREA (g)
  - Area of a polygon
- SDO_LENGTH (g)
  - Length of a line, or perimeter of a polygon
- SDO_DISTANCE (g1,g2)
  - Distance between two objects

Examples

- What is the total area of the Yellowstone Park?

```
SELECT sdo_geom.sdo_area(geom,0.005,'unit=sq_km')
  FROM us_parks
 WHERE name = 'Yellowstone NP';
```

- What is the length of the Mississippi river?

```
SELECT sdo_geom.sdo_length(geom,0.005,'unit=km')
  FROM us_rivers
 WHERE name = 'Mississippi';
```

- What is the distance between Los Angeles and Frisco

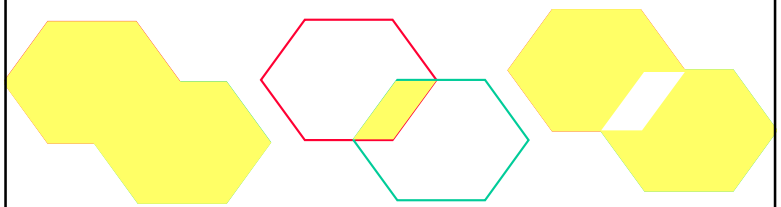
```
SELECT sdo_geom.sdo_distance(a.location, b.location,
0.005, 'unit=mile')
  FROM us_cities a, us_cities b
 WHERE a.city = 'Los Angeles'AND b.city = 'San Francisco';
```

Generation of new objects

- SDO_BUFFER (g, size)
  - Generate a buffer of the desire size
  - For internal buffer, size is negative
- SDO_CENTROID (g)
  - Gravity center (can be outside !!)
- SDO_CONVEXHUL (g)
  - Convex hull
- SDO_MBR (g)
  - Minimum bounding rectangle

Union, Intersection, Difference

- SDO_UNION (g1, g2)
- SDO_INTERSECTION (g1, g2)
- SDO_DIFFERENCE (g1, g2)



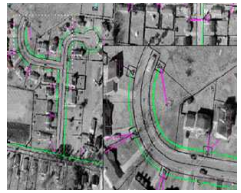
Spatial Aggregates

- Like SUM, COUNT, AVG ...
- Operate on a set of objects
- SDO_AGGR_MBR
  - MBR of a set of objects.
- SDO_AGGR_UNION
- SDO_AGGR_CENTROID
- SDO_AGGR_CONVEXHUL

10.5 – Geo Raster

A new abstract data type

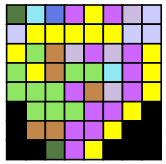
- Ortho-photos, remote sensing data, grids
  - Multi-bands, multi-layer
- An XML schema for associated metadata
  - Data source, layer information
- Geo-Referencing
  - Associated to pixels



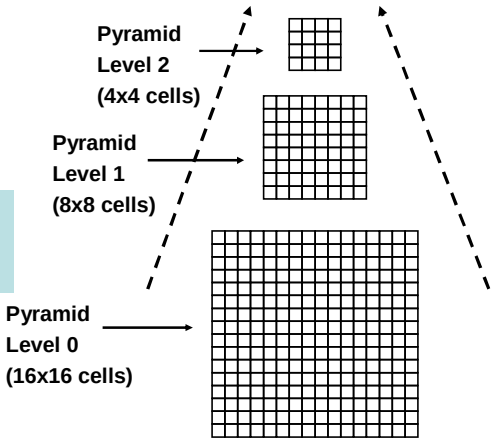
What is a Raster

- A 2-dimension array composed of cells or pixels (regularly spaced)
  - Orthophotos
  - Remote Sensing
  - Grids (SIG raster)
- Each cell/pixel has several values
  - Color
  - Frequency
  - Other …

2	5	4	9	1	9	7	6
6	1	1	1	1	1	6	6
1	3	8	7	9	7	9	1
3	1	8	3	3	5	9	1
3	3	3	9	8	7	9	1
0	3	3	3	9	9	1	0
0	8	8	9	9	1	0	0
0	0	2	9	1	0	0	0

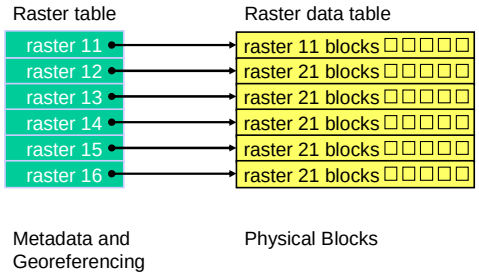


Pyramid



Physical storage

Separation « logical /physical »



Storage Model

A more complex example

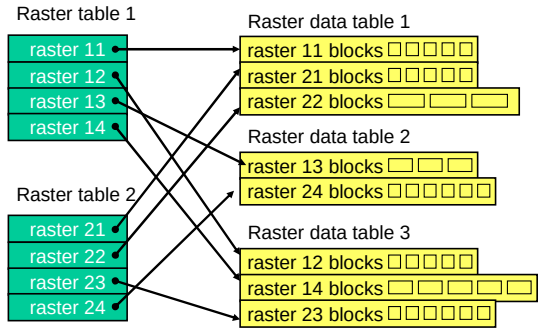


Figure 1–2 Physical Storage of GeoRaster Data

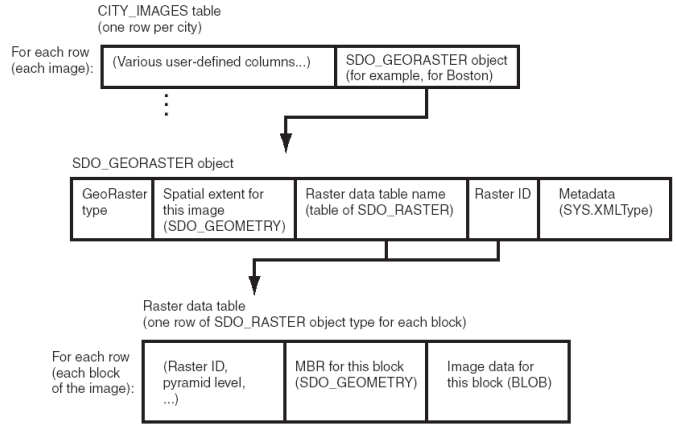
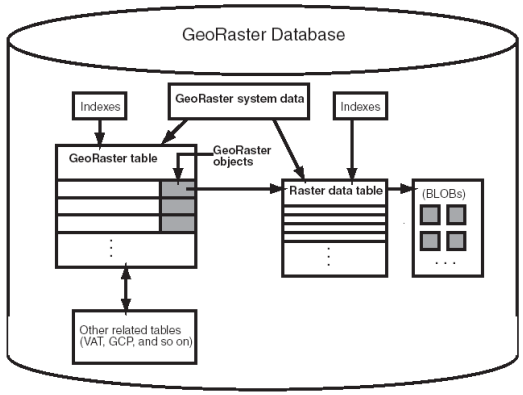


Figure 1–3 GeoRaster Data in an Oracle Database



Creation of raster tables

Creation of a table:

```
CREATE TABLE UK_RASTERS
    (ID           NUMBER PRIMARY KEY,
     SOURCE_FILE  VARCHAR2(80),
     DESCRIPTION  VARCHAR2(32),
     GEORASTER    SDO_GEORASTER)
```

Storage

```
CREATE TABLE UK_RASTERS_RDT_1 OF SDO_RASTER
  (PRIMARY KEY (
     RASTERID, PYRAMIDLEVEL, BANDBLOCKNUMBER,
     ROWBLOCKNUMBER, COLUMNBLOCKNUMBER))
  LOB(RASTERBLOCK) STORE AS (NOCACHE NOLOGGING);
```

GeoRaster Functions

- Insertion, updating, indexing and extraction
- Manipulation:
  - Generation of a resolution pyramid
  - Modification of format (Interleaving, blocking)
- Selection: geographic zone, band, pyramid
- Zooming in or out
- Mosaicking
- Access to pixel

GeoRaster Functions

- Format (import/export) :
  - TIFF/GeoTIFF
  - ESRI World File
  - JPEG
  - GIF
  - BMP
  - PNG

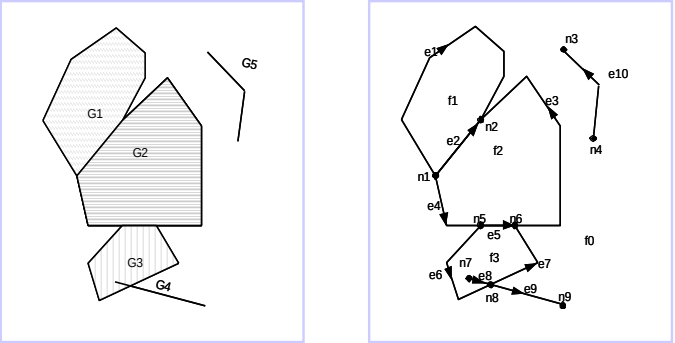
Table 1–3 Subprograms to Validate and Process GeoRaster Objects

Subprogram	Description
SDO_GEOR.validateGeoraster	Validates a GeoRaster object.
SDO_GEOR.schemaValidate	Validates a GeoRaster object's metadata against the GeoRaster XML schema.
SDO_GEOR.generateSpatialExtent	Generates a Spatial geometry that contains the spatial extent of the GeoRaster object.
SDO_GEOR.generatePyramid	Generates pyramid data for a GeoRaster object, which is stored together with the original data.
SDO_GEOR.deletePyramid	Deletes the pyramid data of a GeoRaster object.
SDO_GEOR.subset	Performs either or both of the following operations: (1) spatial crop, cut, or clip, or (2) layer or band subset.
SDO_GEOR.scale	Scales (enlarges or reduces) a GeoRaster object.
SDO_GEOR.scaleCopy	Scales (enlarges or reduces) a GeoRaster object and puts the result into a new object that reflects the scaling.
SDO_GEOR.changeFormat	Changes the storage format of an existing GeoRaster object (for example, changing the blocking, cell depth, or interleaving).
SDO_GEOR.changeFormatCopy	Makes a copy of an existing GeoRaster object using a different storage format (for example, changing the blocking, cell depth, or interleaving).
SDO_GEOR.georeference	Georeferences a GeoRaster object using specified cell-to-model transformation coefficients.
SDO_GEOR.mosaic	Mosaics GeoRaster objects into one GeoRaster object.

10.6 – Network modeling

- Spatial analysis must discover relations between objects
- Very time-consuming
- Structure for acceleration
- Tolerance and accuracy
- Topological primitives: nodes, arcs, faces

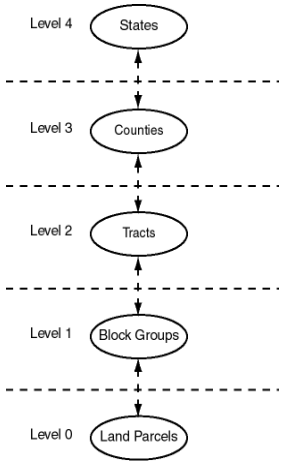
Geometry and Topology



Topology in Oracle

- Tables of topological primitives
  - Nodes, arcs, faces
  - Each primitive is stored only once
- Each primitive can be associated to one or several objects, f.i.
  - An arc can be the boundary of two parcels
  - An arc can be the boundary between a lake and a parcel
- Objects can have a hierarchical structure

Hierarchical structure



Creation of a Topology

```
SQL> EXECUTE SDO_TOPO.CREATE_TOPOLOGY('LAND_USE');
```

```
LAND_USE_NODE$                  TABLE
LAND_USE_EDGE$                  TABLE
LAND_USE_FACE$                  TABLE
```

Topological Primitives

- Three tables
  - <topology-name>_NODE$
  - <topology-name>_EDGE$
  - <topology-name>_FACE$
- And an extra table for relations.
  - <topology-name>_RELATION$

Creation of tables

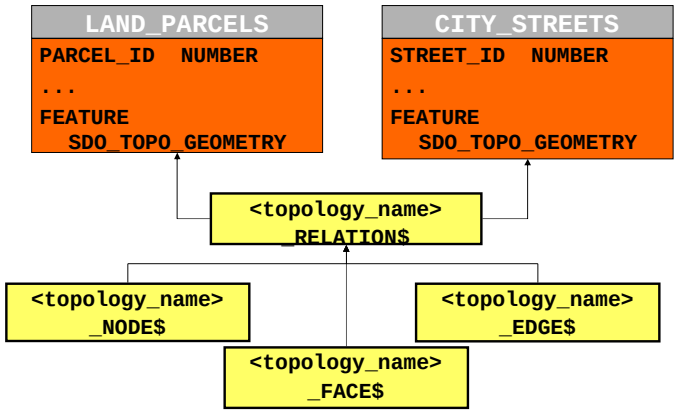
- Creation using *SDO_TOPO_GEOMETRY*

```
CREATE TABLE LAND_PARCELS (
  PARCEL_ID     NUMBER PRIMARY KEY,
  BLOCK_ID      NUMBER
  PARCEL_NAME   VARCHAR2(30),
  FEATURE       SDO_TOPO_GEOMETRY
);
```

- And then associated topology

```
SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER(
  'LAND_USE', 'LAND_PARCELS', 'FEATURE','POLYGON');
```

Relations between objects and primitives



Nodes Table

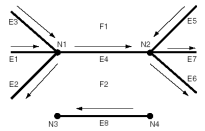
Table 1–3 Columns in the <topology-name>_NODE$ Table

Column Name	Data Type	Description
NODE_ID	NUMBER	Unique ID number for this node.
EDGE_ID	NUMBER	ID number (signed) of the edge (if any) associated with this node.
FACE_ID	NUMBER	ID number of the face (if any) associated with this node.
GEOMETRY	SDO_GEOMETRY	Geometry object (point) representing this node.

Edges table

Table 1–1 Columns in the <topology-name>_EDGE$ Table

Column Name	Data Type	Description
EDGE_ID	NUMBER	Unique ID number for this edge.
START_NODE_ID	NUMBER	ID number of the start node for this edge.
END_NODE_ID	NUMBER	ID number of the end node for this edge.
NEXT_LEFT_EDGE_ID	NUMBER	ID number (signed) of the next left edge for this edge.
PREV_LEFT_EDGE_ID	NUMBER	ID number (signed) of the previous left edge for this edge.
NEXT_RIGHT_EDGE_ID	NUMBER	ID number (signed) of the next right edge for this edge.
PREV_RIGHT_EDGE_ID	NUMBER	ID number (signed) of the previous right edge for this edge.
LEFT_FACE_ID	NUMBER	ID number of the left face for this edge.
RIGHT_FACE_ID	NUMBER	ID number of the right face for this edge.
GEOMETRY	SDO_GEOMETRY	Geometry object (line string) representing this edge.



Faces Table

Table 1–4 Columns in the <topology-name>_FACE$ Table

Column Name	Data Type	Description
FACE_ID	NUMBER	Unique ID number for this face.
BOUNDARY_EDGE_ID	NUMBER	ID number of the boundary edge for this face. The sign of this number (which is ignored for use as a key) indicates which orientation is being used for this boundary component (positive numbers indicate the left of the edge, and negative numbers indicate the right of the edge).
ISLAND_EDGE_ID_LIST	SDO_LIST_TYPE	Island edges (if any) in this face.
ISLAND_NODE_ID_LIST	SDO_LIST_TYPE	Island nodes (if any) in this face.
MBR_GEOMETRY	SDO_GEOMETRY	Minimum bounding rectangle (MBR) that encloses this face. (This is not required. However, if the MBR is specified and if a spatial R-tree index is defined on this geometry, the face can be retrieved more efficiently.)

Creating the topology

```
-- Create the topology. (Null SRID in this example.)
EXECUTE SDO_TOPO.CREATE_TOPOLOGY('LAND_USE_HIER', 0.00005);
-- Create feature tables.
CREATE TABLE land_parcels ( -- Land parcels (selected faces)
   feature_name VARCHAR2(30) PRIMARY KEY,
   feature SDO_TOPO_GEOMETRY);
CREATE TABLE block_groups (
   feature_name VARCHAR2(30) PRIMARY KEY,
   feature SDO_TOPO_GEOMETRY);
CREATE TABLE tracts (
   feature_name VARCHAR2(30) PRIMARY KEY,
   feature SDO_TOPO_GEOMETRY);
CREATE TABLE counties (
   feature_name VARCHAR2(30) PRIMARY KEY,
   feature SDO_TOPO_GEOMETRY);
CREATE TABLE states (
   feature_name VARCHAR2(30) PRIMARY KEY,
   feature SDO_TOPO_GEOMETRY);
```

Example PL/SQL

```
DECLARE
   land_parcels_id NUMBER;
   block_groups_id NUMBER;
   tracts_id NUMBER;
   counties_id NUMBER;
BEGIN
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('LAND_USE_HIER', 'LAND_PARCELS',
        'FEATURE','POLYGON');
   SELECT tg_layer_id INTO land_parcels_id FROM user_sdo_topo_info
   WHERE topology = 'LAND_USE_HIER' AND table_name = 'LAND_PARCELS';
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('LAND_USE_HIER', 'BLOCK_GROUPS',
        'FEATURE','POLYGON', NULL, land_parcels_id);
   SELECT tg_layer_id INTO block_groups_id FROM user_sdo_topo_info
        Topology Data Model Tables
        WHERE topology = 'LAND_USE_HIER' AND table_name = 'BLOCK_GROUPS';
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('LAND_USE_HIER', 'TRACTS',
        'FEATURE','POLYGON', NULL, block_groups_id);
   SELECT tg_layer_id INTO tracts_id FROM user_sdo_topo_info
        WHERE topology = 'LAND_USE_HIER' AND table_name = 'TRACTS';
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('LAND_USE_HIER', 'COUNTIES',
        'FEATURE','POLYGON', NULL, tracts_id);
   SELECT tg_layer_id INTO counties_id FROM user_sdo_topo_info
        WHERE topology = 'LAND_USE_HIER' AND table_name = 'COUNTIES';
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('LAND_USE_HIER', 'STATES',
        'FEATURE','POLYGON', NULL, counties_id);
END;
```

Topological queries

- Using (SDO_ANYINTERACT, SDO_INSIDE, SDO_CONTAINS, SDO_TOUCH, etc.)
- Example: find parcels neighboring another parcel

```
SELECT p1.parcel_name
  FROM land_parcels p1, land_parcels p2
 WHERE p2.parcel_name = 'OConnor Place'
   AND SDO_TOUCH (p1.feature, p2.feature) = 'TRUE';
```

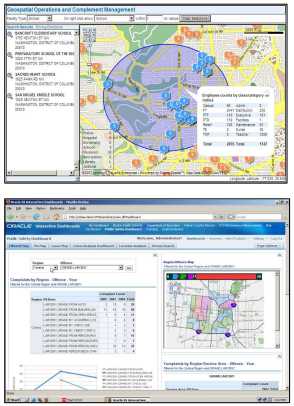
Extracting geometries

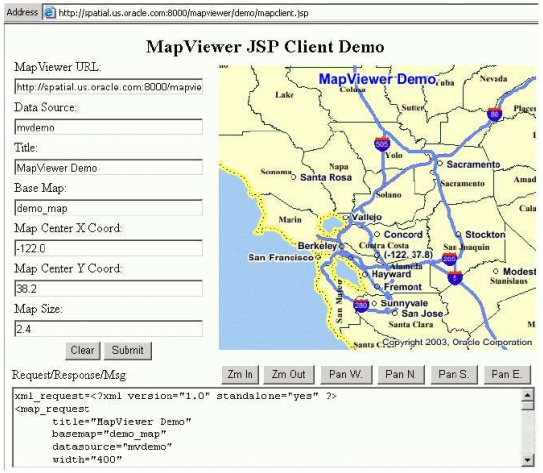
- GET_GEOMETRY() of the object with SDO_TOPO_GEOMETRY

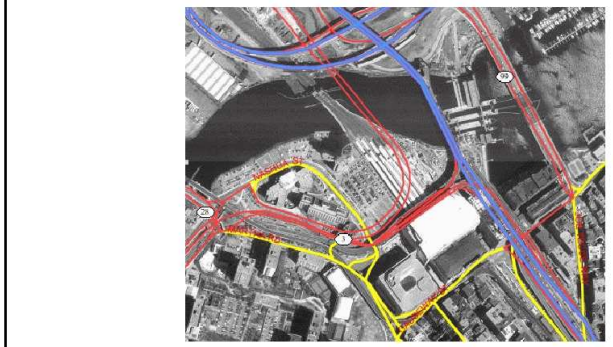
```
SELECT parcel_id, parcel_name,
       p.feature.get_geometry()
FROM   land_parcels p;
```

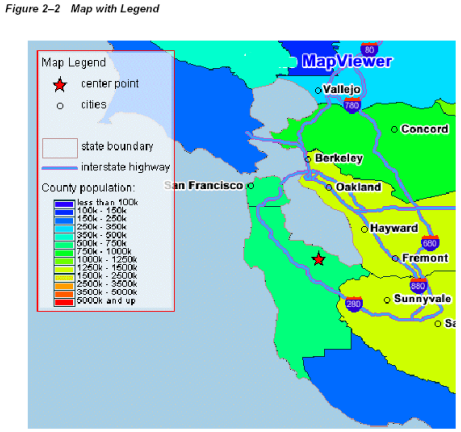
10.7 – Map Viewer Application Server

- Interfaces XML, Java et Javascript (Ajax)
- Tool for defining maps
- Thematic maps
- Formats PNG, GIF, JPEG, SVG
- Compatibility OGC WMS
  - Server and client levels










```
declare
   l_http_req utl_http.req;
   l_http_resp utl_http.resp;
   l_url varchar2(4000):= 'http://my_corp.com:8888/mapviewer/omserver';
   l_value varchar2(4000);
   img_url varchar2(4000);
   response sys.xmltype;
   output varchar2(255);
   map_req varchar2(4000);
begin
   utl_http.set_persistent_conn_support(TRUE);
   map_req := '<?xml version="1.0" standalone="yes"?>
   <map_request title="MapViewer Demonstration"
                datasource="mvdemo"
                basemap="course_map"
                Map Request Examples
                width="500"
                height="375"
                bgcolor="#a6cae0"
                antialiasing="false"
                format="GIF_URL">
      <center size="5" >
         <geoFeature>
            <geometricProperty>
               <Point>
                  <coordinates>-122.2615, 37.5266</coordinates>
               </Point>
            </geometricProperty>
         </geoFeature>
      </center>
   </map_request>';
```

Example of interactions between PL/SQL and Map Viewer

```
  l_http_req := utl_http.begin_request(l_url, 'POST', 'HTTP/1.0');
  --
  -- sets up proper HTTP headers
  --
  utl_http.set_header(l_http_req, 'Content-Type',
       'application/x-www-form-urlencoded');
  utl_http.set_header(l_http_req, 'Content-Length',
  length('xml_request=' ||   map_req));
  utl_http.set_header(l_http_req, 'Host', 'my_corp.com');
  utl_http.set_header(l_http_req, 'Port', '8888');
  utl_http.write_text(l_http_req, 'xml_request=' || map_req);
  --
  l_http_resp := utl_http.get_response(l_http_req);
  utl_http.read_text(l_http_resp, l_value);
  response := sys.xmltype.createxml (l_value);
  utl_http.end_response(l_http_resp);
  img_url := response.extract('/map_response/map_image/map_
       content/@url').getstringval();
  dbms_output.put_line(img_url);
end;
```

10.8 – Oracle Spatial by Example

- Thanks to
  - Richard L. Flores
  - Isinglass, Inc.
  - pleides100@yahoo.com

http://www.nocoug.org/download/2006-08/IntroOraSpatial.ppt

Scenario

- You wish to open an upscale beauty salon in central Contra Costa county, California, catering to wealthier, older women.
- You would like to be close to a major thoroughfare for ease of access.
- You don't want to be too close to any competitors.

Identify Types and Sources of Data Needed to Support Decision

- Competitors: Internet Search Engine
- Demographic (Age, Gender, Income): U.S. Census Bureau
- Roads: U.S. Geological Survey

Competitor Data: Table

```
CREATE TABLE beauty (id              NUMBER(38),
                     name            VARCHAR2(100),
                     full_address    VARCHAR2(100),
                     city_state      VARCHAR2(50),
                     street_number   VARCHAR2(10),
                     street_name     VARCHAR2(20),
                     street_type     VARCHAR2(15),
                     street_prefix   VARCHAR2(10),
                     street_suffix   VARCHAR2(10),
                     city            VARCHAR2(40),
                     state           VARCHAR2(2),
                     postal_code     VARCHAR2(16),
                     location        MDSYS.SDO_GEOMETRY);
```

Competitor Data: Spatial Metadata

```
INSERT INTO user_sdo_geom_metadata VALUES
  ('BEAUTY',                            -- Geometry Table
   'LOCATION',                          -- Geometry Column
    SDO_DIM_ARRY (
     SDO_DIM_ELEMENT ('LONGITUDE',      -- Longitude Text
                      -180,               --   Lower Boundary
                      180,                --   Upper Boundary
                      0.5),               --   Tolerance
     SDO_DIM_ELEMENT ('LATITUDE',       -- Latitude Text
                        -90,              --   Lower Boundary
                        90,               --   Upper Boundary
                        0.5)              --   Tolerance
     ),
     8307                                 -- (SRID) Datum:WGS84
  );
```

Competitor Data: Spatial Index

```
CREATE INDEX beauty_spatial_idx ON beauty (location)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

- R-Tree Index
- These are unlike regular Oracle indexes and special steps must be taken with their administration.

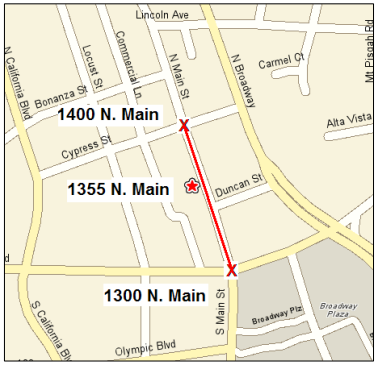
Competitor Data: Source

Extract list of competitors and their addresses from Search Engine.


```
Name
----------------
ID
NAME
FULL_ADDRESS
CITY_STATE
STREET_NUMBER
STREET_NAME
STREET_TYPE
STREET_PREFIX
STREET_SUFFIX
CITY
STATE
POSTAL_CODE
LOCATION
```

While very useful, it doesn't provide any directly mappable data.

Competitor Data: Geocoding



- The Geocoder will
  - Standardize Address Name and,
  - Using a database with the coordinates and street addresses of each intersection,
  - Interpolate the location of the given address.
- Oracle Spatial Option geocoder: added-cost
- Third party sells spatial database used to calculate the coordinates

Competitor Data: Geocoding

simplest_xmlrpc.pl "1355 N. Main, Walnut Creek, CA"

```
$VAR1 = [
     {
       'number' => '1355',
       'street' => 'Main',
       'lat' => '37.898365',
       'state' => 'CA',
       'city' => 'Walnut Creek',
       'zip' => '94596',
       'suffix' => '',
       'long' => '-122.060445',
       'type' => 'St',
       'prefix' => 'N'
     }
   ];
```

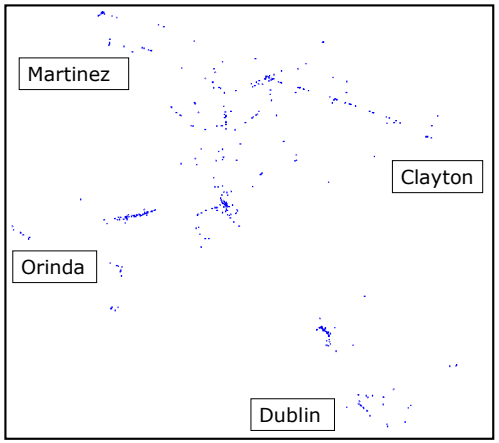
Name

ID
NAME
FULL_ADDRESS
CITY_STATE
STREET_NUMBER
STREET_NAME
STREET_TYPE
STREET_PREFIX
STREET_SUFFIX
CITY
STATE
POSTAL_CODE
LOCATION

Competitor Data: SDO_GEOMETRY Object-Relational Type

```
UPDATE beauty
   SET location      =
         SDO_GEOMETRY
          (2001,      -- Geometry Type: 2-D Point
           8307,           -- SRID, Datum: WGS84
           SDO_POINT_TYPE
             (-122.060445,  -- Longitude
              37.898365,    -- Latitude
              NULL),
            NULL,
            NULL
            )
 WHERE id = 430;
```

Competitor Data: Data Display

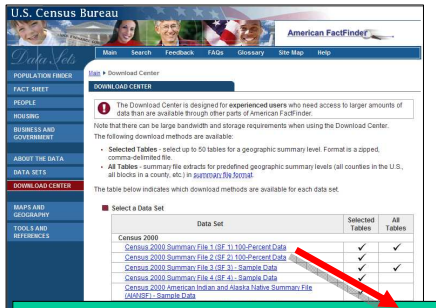


- eSpatial iSmart Explorer free on OTN
- OEM Spatial Index Advisor
- Oracle Mapviewer
- For serious users, many commercial products.

Non-Spatial Demographic Data: Table

```
CREATE TABLE census_data (
  CENSUS_TRACT       VARCHAR2(10)NOT NULL,
  MED_HOUSE_INCOME   NUMBER(38),
  GENDER_TOTAL       NUMBER(38),
  FEMALE_GE_40       NUMBER(38));
```

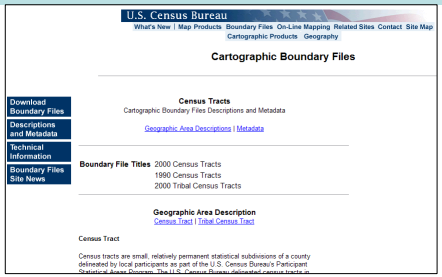
Non-Spatial Demographic Data: Source



- U.S. Census Bureau
- factfinder.census.gov
- "Download Center"
- Select detailed or summarized data by state, county, and census tract.

```
CENSUS_TRACT    MED_HOUSE_INCOME  FEMALE_GE_40 GENDER_TOTAL
--------------- ----------------  ------------ ------------
3010                       44871          975         3355
3020.02                    58769         1467         8475
                     * * * * * * *
```

Spatial Census Tract Data: Source



- www.census.gov/geo/www/cob/tr_metadata.html
- Has geographic boundaries of Census Tracts which can be loaded into Oracle Spatial.
- Choose state and "ARCVIEW Shapefile" format to download file for California. These files are sometimes called "ESRI Shapefiles".

Spatial Census Tract Data: Loading census_tract.sql

```
DROP TABLE CENSUS_TRACTS;

CREATE TABLE CENSUS_TRACTS (
  AREA           NUMBER,
  PERIMETER      NUMBER,
  TR06_D00_      NUMBER,
  TR06_D00_I     NUMBER,
  STATE          VARCHAR2(2),
  COUNTY         VARCHAR2(3),
  TRACT          VARCHAR2(6),
  NAME           VARCHAR2(90),
  LSAD           VARCHAR2(2),
  LSAD_TRANS     VARCHAR2(50),
  GEOM           MDSYS.SDO_GEOMETRY);
```

Spatial Census Tract Data: Loading

•In SQL*Plus:

connect spatial/spatial

@census_tracts.sql

•Run SQL*Loader:

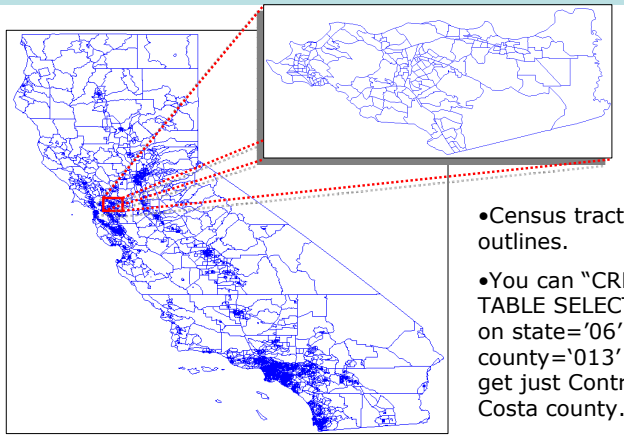
sqlldr spatial/spatial census_tracts

•In SQL*Plus:

connect spatial/spatial

EXECUTE SDO_MIGRATE.TO_CURRENT('CENSUS_TRACTS','GEOM)'

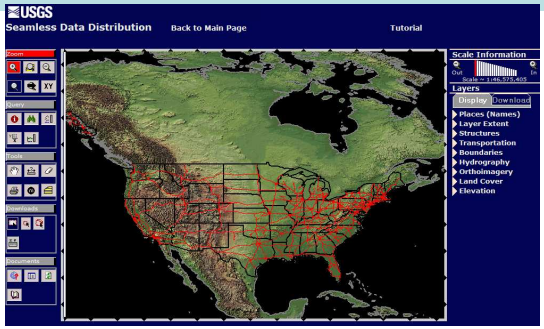
Spatial Census Tract Data: Display



•Census tract outlines.

•You can "CREATE TABLE SELECT AS" on state='06' and county='013' to get just Contra Costa county.

Road Data: Source

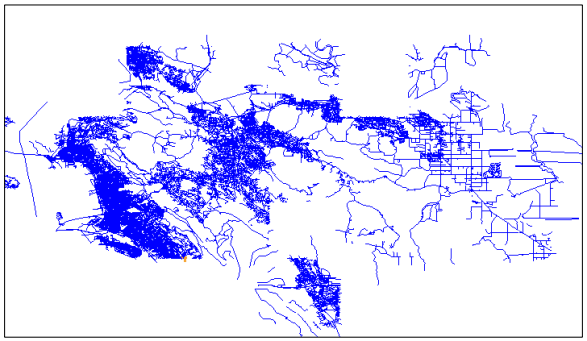


•seamless.usgs.gov

•Bureau of Transportation Statistics from U.S. Geological Survey.

•shapefiles

Road Data: Display



Analysis: Criteria Definition

- Within 2 miles of census tracts in which
  - The Median Household Annual Income is greater then $100K and
  - Over 30% of the people are women 40 years or older
- Within ½ mile of a major thoroughfare
- Not within ½ mile of a competitor

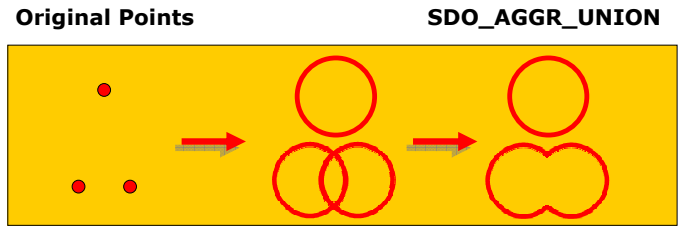
Analysis: Oracle Spatial Buffers

	Original Geometry	Buffered Geometry
Point		
Line String		
Polygon		

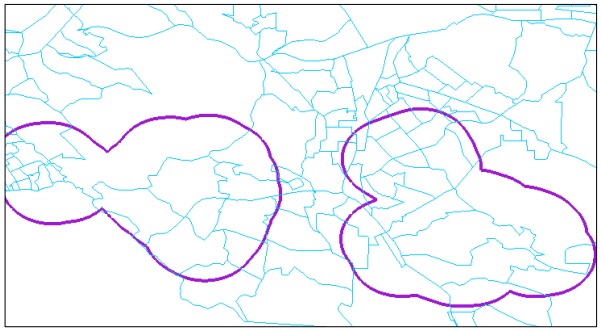
Analysis: Target Census Tract Buffer

```
CREATE TABLE target_tract_buffer AS
SELECT SDO_AGGR_UNION(SDOAGGRTYPE(
         SDOAGGRTYPE(
           SDO_GEOM.SDO_BUFFER(
           a.geom,              -- geometry column
           2.00,                -- Distance
           0.5,
           'arc_tolerance=0.005 unit=mile'), -- Units
         0.5)) geom
 FROM census_tracts a,
      census_data   b
WHERE b.census_tract                = a.name
  AND b.med_house_income            >=100000
  AND b.female_ge_40/b.gender_total >= 0.30
  AND a.state                       = '06'
  AND a.county                      = '013';
```

Analysis: Target Census Tract Buffer



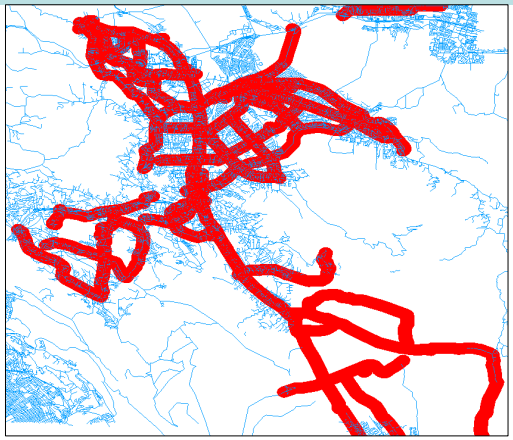
Analysis: Target Census Tract Buffer



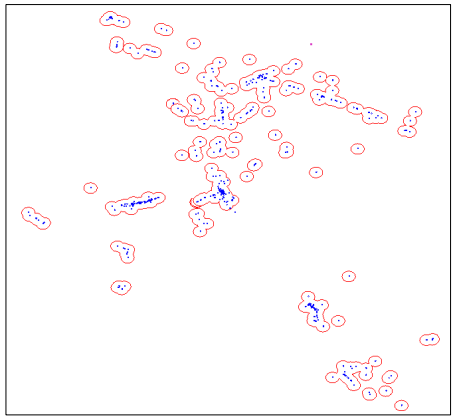
Analysis: Major Road Buffer

```
CREATE TABLE road_buffer AS
SELECT prefix, name, type, suffix,
       SDO_AGGR_UNION(
         SDOAGGRTYPE(
          SDO_GEOM.SDO_BUFFER(
           a.geom,              -- geometry column
           0.50,                -- Distance
           0.5,
           'arc_tolerance=0.005 unit=mile'), -- Units
         0.5)) geom
  FROM roads a
 WHERE (name = 'ACALANES' AND type = 'AVE')
    OR (name = 'ACALANES' AND type = 'RD')
                 * * * * * * * *
    OR (name = 'YGNACIO VALLEY' AND type = 'RD');
```

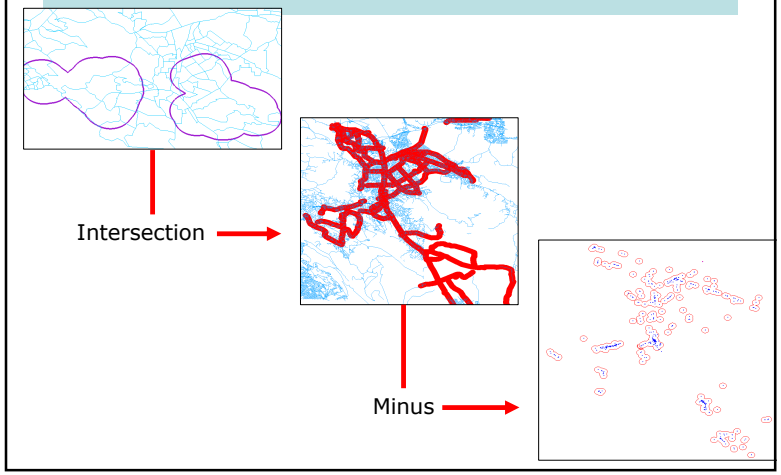
Analysis: Major Road Buffer



Analysis: Competitor Buffer

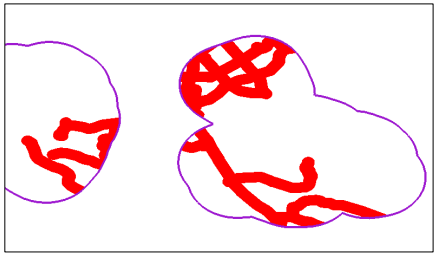


Analysis: Spatial Operations



Analysis: Spatial Operations

```
CREATE TABLE target_site_wocomp AS
SELECT SDO_AGGR_UNION(SDOAGGRTYPE(c.geom,0.5)) geom
  FROM (SELECT SDO_GEOM.SDO_INTERSECTION(
                a.geom, b.geom, 0.5) geom
          FROM target_tract_buffer a,road_buffer b)
       c);
```

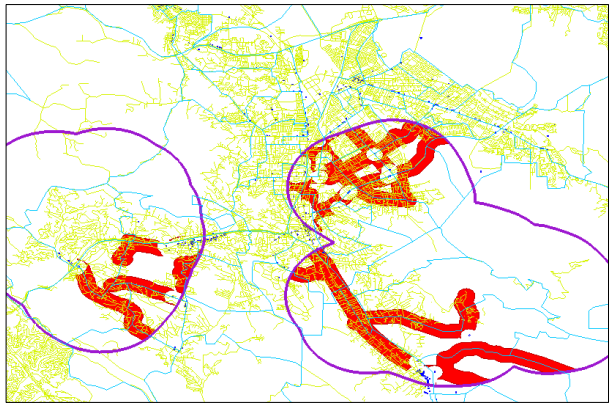


Analysis: Spatial Operations

```
CREATE TABLE target_site AS
SELECT SDO_AGGR_UNION(SDOAGGRTYPE(a.geom,0.5)) geom
  FROM (SELECT SDO_GEOM.SDO_DIFFERENCE(
               b.geom, c.geom, 0.5) geom
          FROM target_site_wocomp b,competitor_buffer
       c) a;

-- Create spatial metadata and index for target_site
--   and target_site_wocomp after creation.
```

Analysis: Final Display



10.9 – Final Remarks

- Total processing of spatial data and non-spatial data
- Integrated in most existing GIS

That's all Folks!!