SPATIAL DATA IN ORACLE

Oracle Geometry

OGC specification vs Oracle model

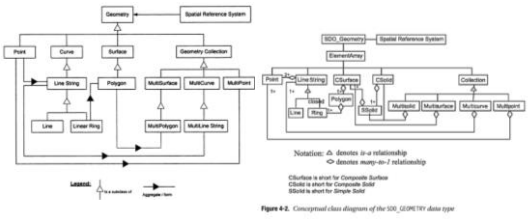


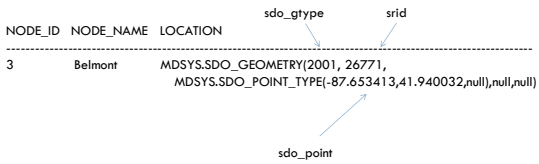
Figure 4-2. *Conceptual class diagram of the SDO_GEOMETRY data type*

sdo_geometry

```
describe sdo_geometry;
user type definition
---------------------------------------------
TYPE SDO_GEOMETRY           AS OBJECT (
           SDO_GTYPE        NUMBER,
           SDO_SRID         NUMBER,
           SDO_POINT        SDO_POINT_TYPE,
           SDO_ELEM_INFO    SDO_ELEM_INFO_ARRAY,
           SDO_ORDINATES    SDO_ORDINATE_ARRAY,
… )
```

sdo_geometry

```
select node_id, node_name, location from ctanode where node_name = 'Belmont';
```



sdo_gtype

format: D**00**T (different for linear referenced geometry)

D Dimension
- D = 2: two-dimensional
- D = 3: three-dimensional
- D = 4: four-dimensional

T Shape
- T = 0 without type
- T = 1 point
- T = 2 line
- T = 3 polygon/surface
- T = 4 collection

T Shape
- T = 5 multipoint
- T = 6 multiline
- T = 7 multipolygon/surface
- T = 8 solid
- T = 9 multisolid

For collections, D is upper bound on dimension of elements.

points

```
select sdo_geometry('point(-79 41)', 26771) from dual;

MDSYS.SDO_GEOMETRY(2001,26771,MDSYS.SDO_POINT_TYPE(-79,41,null),null,null)
```

- for single points use sdo_point (faster)
- sdo_point always has three coordinates
- for higher-dimensional data: sdo_elem_info, sdo_coordinates
- gtype 2001

```
TYPE SDO_GEOMETRY           AS OBJECT (
             SDO_GTYPE      NUMBER,
             SDO_SRID       NUMBER,
             SDO_POINT      SDO_POINT_TYPE,
             SDO_ELEM_INFO  SDO_ELEM_INFO_ARRAY,
             SDO_ORDINATES  SDO_ORDINATE_ARRAY,
… )
```

SRID (Spatial Reference System)

- in table mdsys.cs_srs

```
select cs_name, srid, wktext
from mdsys.cs_srs
where wktext like 'PROJCS%' and cs_name like '%Illinois%';
```

- types:
  - geodetic
  - projected

  (geo-referenced)
  - local (non-georeferenced), e.g. CAD/CAM

```
select cs_name, srid, wktext
from mdsys.cs_srs
where wktext like '%LOC%';
```

sdo_elem_info / sdo_ordinates

- sdo_elem_info: structure of geometric data
- sdo_ordinates: geometric data

simple: point, line, polygon (including multi)
versus
complex: compound, voided, collection

Simple Geometries

descriptor triplet:

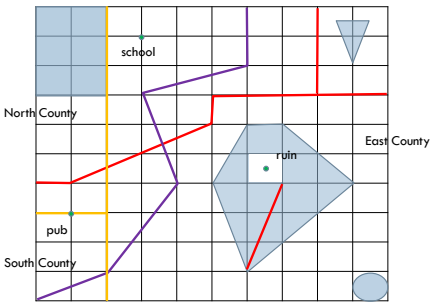
(offset, element_type, interpretation)

offset = 1 (for simple, starting position in ordinates)

element_type: 1 (point, gtype 1)
2 (line, gtype 2)
1003 (polygon, gtype 3)

interpretation: point: 1
line/polygon: 1for straight lines, 2 for arcs
polygon: 3 for rectangle, 4 for circle

my_world



simple geometry examples

'ruin', sdo_geometry(2001, 262156, sdo_point_type(6.5, 4.5, null), null, null)

'purple street', sdo_geometry(2002, 262156, null,
sdo_elem_info_array(1,2,1), sdo_ordinate_array(0,0,2,1,4,4,3,7,6,8,6,10))

'rectangle lake', sdo_geometry(2003, 262156, null,
sdo_elem_info_array(1,1003,3), sdo_ordinate_array(0,7, 2, 10))

'triangle lake', sdo_geometry(2003, 262156, null,
sdo_elem_info_array(1,1003,1), sdo_ordinate_array(9,8,9.5, 9.5, 8.5, 9.5, 9,8))
-- note that first vertex is repeated to get linear ring

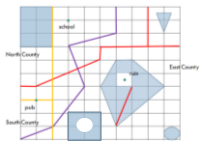
Exercise: south county
circle lake

voided polygons (polygons with holes)

- 1003 for outer ring, 2003 for inner ring
- outer rings: counter-clockwise, inner rings: clockwise

'poly lake', sdo_geometry(2003, 262156, null,
sdo_elem_info_array(1,1003, 1, 13, 2003, 1),
sdo_ordinate_array(6,1,9,4,7,6,6,6,5,4,6,1, 6,4,6,5,7,5,7,4,6,4)));

Exercise:



compound lines

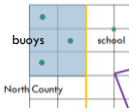
- line segment can be
  - straight (type 1)
  - arced (type 2)
- similarly a polygon can have arced or straight sides
- can be encoded as "compound" object
  - compound line:
    - element type: 4,
  - compound polygon:
    - 1005 (compound outer polygon),
    - 2005 (compound inner polygon)
  - interpretation: number of elements

collections

- homogeneous (all the same type)
  - multiline types: 2005 (points), 2006 (lines), 2007 (polygons)
- heterogeneous (different types)
  - collection type: 2004

'red street', sdo_geometry(2006, 262156, null,
sdo_elem_info_array(1,2,1,13,2,1),
sdo_ordinate_array(0,4,1,4,5,6,5,7,8,7,10,7,8,7,8,10))

Exercise:



In 3-D

- points (3001),
- lines (3002),
- surfaces (3003),
- solids (3008)

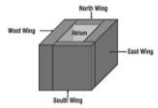


Figure 4-29. *Modeling the simple solid of Figure 4-28 as a composite solid*

Listing 4-42. *Example SQL for Composite Solid of Figure 4-29*

```
SQL> INSERT INTO geometry_examples (name, description, geom) VALUES
(
  '3-D Composite Solid of 4 simple solids',
  '3-dimensional composite solid ',
  SDO_GEOMETRY
  (
    3008,     -- SDO_GTYPE Format: DOOT. Set to 3008 for a 3-dimensional Solid
    NULL,     -- No coordinate system
    NULL,  -- No data in SDO_POINT attribute
    SDO_ELEM_INFO_ARRAY(
      1,   -- Offset of the composite solid element
      1006, --- Etype for a composite solid
      4,  --Number of simple solids making up the composite.
        --The simple solid descriptors next.
        1,   1007, 3,  -- Simple solid as a solid box
        7,   1007, 3,  -- Simple solid as a solid box
        13, 1007, 3,  -- Simple solid as a solid box
        19, 1007, 3   -- Simple solid as a solid box
    ),
    SDO_ORDINATE_ARRAY
    (
      -- min-corner and max-corner  for the  West wing
      2,0,1,  3.5,3,4,
      -- min-corner and max-corner for the East wing,
      3.5, 0,1,  4,3,3.5,
      -- min-corner and max-corner for the North wing,
      2.5,0,0,  3.5,3,4.5,
      -- min-corner and max-corner for the South wing,
      2.5,0,0.5,  4,3,4
    )
  )
);
```

Example from Pro Oracle Spatial

Spatial Operators

Spatial Indexes

- Before we can use spatial operators, we need to build spatial indexes
- Before we can build spatial indexes, we need to give the system the geometric metadata

Geometric Metadata and Spatial Index

```
insert into user_sdo_geom_metadata
       (table_name, column_name, srid, diminfo)
values ('my_poi', 'location', 262156,
       sdo_dim_array(sdo_dim_element('X',0,10, 0.1),
       sdo_dim_element('Y',0,10, 0.1)));
```

Warning: names are converted to uppercase, so for a delete you need to refer to `MY_POI'

```
create index my_poi_idx on my_poi(location)
indextype is mdsys.spatial_index;
```

even better: specify geometry:

```
create index my_poi_idx on my_poi(location)
indextype is mdsys.spatial_index
parameters('layer_gtype=multipoint');
```

Spatial Operations: within_distance

- SDO_WITHIN_DISTANCE(<loc>, <loc>, <param>)
  - param = 'DISTANCE = 2 UNIT = mile'
  - closest distance between two objects
  - returns 'TRUE' or 'FALSE' (strings, not Booleans)

```
select a.poi_name, b.poi_name
from my_poi a, my_poi b
where sdo_within_distance(a.location, b.location,
       'DISTANCE = 1 UNIT = MILE') = 'TRUE';
```

Exercise:
- find points of interest close to a road
- find points of interest close to a lake
- find points of interest and what county they lie in

Spatial Operations: sdo_nn

- SDO_NN(<loc>, <loc>, <param> [, <number>])
  - nearest neighbors
  - param = 'sdo_num_res = k' : restrict to k closest
  - use of this operator orders output, can use rownum to restrict
  - second location must be unique, otherwise error message that spatial index is needed
  - returns 'TRUE' or 'FALSE'

```
select a.poi_name, b.lake_name
from my_poi a, my_lake b
where sdo_nn(a.location, b.shape) = 'TRUE'
    and b.lake_name = 'circle lake';
```

Exercise: find two closest lakes to the pub

Spatial Operations: sdo_nn

Example: find two closest lakes to the pub

```
select a.poi_name, b.lake_name
from my_poi a, my_lake b
where  a.poi_name = 'pub' and
       sdo_nn(b.shape, a.location, 'sdo_num_res = 2') = 'TRUE'
       rownum <= 2;
```

what happens if we add

```
       b.lake_name != 'rectangle lake'
```

also, problem if conditions are added that use index (messes up ordering)

Spatial Operations

- SDO_NN_DISTANCE(number)
  - auxiliary function computed by sdo_nn containing distance
  - number refers to fourth optional parameter of sdo_nn

```
select a.poi_name, b.lake_name, sdo_nn_distance(1)
from my_poi a, my_lake b
where a.poi_name = 'pub' and
    sdo_nn(b.shape, a.location, 'sdo_num_res = 3', 1) = 'TRUE' and
    b.lake_name != 'rectangle lake' and
    rownum <= 2;
```

Exercise: find closest points of interest to the purple street and list them ordered by distance

Topological Relationships

- SDO_RELATE

  topological relationships: contains, overlap, …

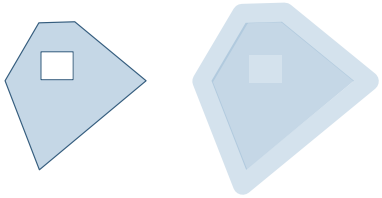
For preprocessing:

- SDO_BUFFER

  create buffer zone around geometry
- SDO_FILTER

  filter out by MBR

Buffer



Creating Buffers

```
drop table my_buff_street;
create table my_buff_street as
select street_id, street_name,
    sdo_geom.sdo_buffer(geom, 0.5, 0.5, 'UNIT = MILE') geom
from my_street;

delete from user_sdo_geom_metadata
       where table_name = 'MY_BUFF_STREET';
insert into user_sdo_geom_metadata
select 'my_buff_street', 'geom', diminfo, srid
from user_sdo_geom_metadata
where table_name = 'MY_STREET';

drop index my_buff_street_idx;
create index my_buff_street_idx on my_buff_street(geom)
indextype is mdsys.spatial_index;
```

Exercise: find points of interest within 0.5 miles of a street.

Filtering

- SDO_FILTER(<loc>, <loc>)
  - returns 'TRUE' if the minimum bounding rectangles of geometries overlap

Exercise: find lakes whose MBRs overlap

Relations:

- SDO_INSIDE(A,B)
  - if A with boundary lies in interior of B
  - same as SDO_CONTAINS(B,A)
- SDO_COVEREDBY(A,B)
  - if interior of A lies in interior of B and boundaries intersect
  - same as SDO_COVERS(B,A)
- SDO_TOUCH(A,B)
  - interiors of A and B are disjoint, but boundaries intersect
- SDO_EQUAL(A,B)
  - A and B are equal
- SDO_ANYINTERACT(A,B)
  - any of the above are true, i.e. the interior and boundary of A share intersect the interior and boundary of B

Exercises

- find all points of interest in East County
- find all points of interest on the boundary of South County
- find all lakes in East County
- list all lakes and the counties they belong to
- find all ferries (streets in lakes)

How about:

- find all streets passing through south county
- find all points of interest on islands (land surrounded by lake)

Relations:

- Overlap
  - SDO_OVERLAPS(A,B)
    - A contains interior points in both the interior and exterior of B and vice versa
  - SDO_OVERLAPBDYINTERSECT(A,B)
    - same as overlap and boundaries intersect
  - SDO_OVERLAPBDYDISJOINT(A,B)
    - same as overlap but boundaries do not intersect (how is that possible?)
- Disjoint
  - not anyinteract

Exercise: construct examples to test these operators

Also: ON

- SDO_ON(A,B)
  - if A is a linestring lying on the boundary of B

Alternative: SDO_RELATE

- SDO_RELATE(<loc>, <loc>, 'MASK = ? ') = 'TRUE'
  - ? can be any of the topological relationships: inside, contains, …
  - ? can also be several topological relationships separated by +, e.g. 'MASK = inside+touch'

Exercise: write query for finding all lakes in a county (even if they share boundary)

Operations on Geometries

- SDO_GEOM.SDO_INTERSECTION(A,B, <tol>)
- SDO_GEOM.SDO_UNION(A,B , <tol>)
- SDO_GEOM.SDO_DIFFERENCE(A,B , <tol>)
- SDO_GEOM.SDO_XOR(A,B , <tol>)
  (symmetric difference: A-B u B-A)

Exercise: test with different geometries: what's the union of two lines, the intersection of two lines, the difference of two lines, difference of polygon and line, …

Create a county NorthSouth that combines North county and South county.

Write a clipping function: given a geometry and a window (x1, y1, x2, y2) return the geometry clipped to that window.

Functions on Geometries

- sdo_geom.sdo_area(<geom>, <tol> [, <param>])
  - area of a region
  - can specify units: `unit = sq_yard' or 'unit = sq_mile', etc.
- sdo_geom.sdo_length(<geom>, <tol> [, <param>])
  - length of a curve
- sdo_geom.sdo_volume
- sdo_geom.sdo_mbr
  - returns MBR

Exercises

- calculate how many miles of the red street lie in North county
- what's the total area of islands
- which counties would a straight road between the pub and the school pass through?
- write a function to check whether you have to cross a given road to get from one point of interest to another