Design of Geographic Information Systems class 9

Thomas Bittner
bittner3@buffalo.edu

Recap: R-trees

R-trees

George Kollios

- An external memory tree
- Index nodes and data (leaf) nodes
- All leaf nodes appear on the same level
- Every node contains between m and M entries
- The root node has at least 2 entries (children)
- (only deal with Minimum Bounding Rectangles - **MBR**s)

Example

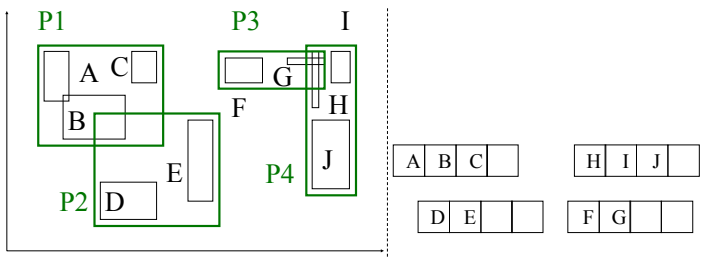
George Kollios

- 4 entries per page; group nearby rectangles to parent MBRs; each group -> disk page

Example

George Kollios

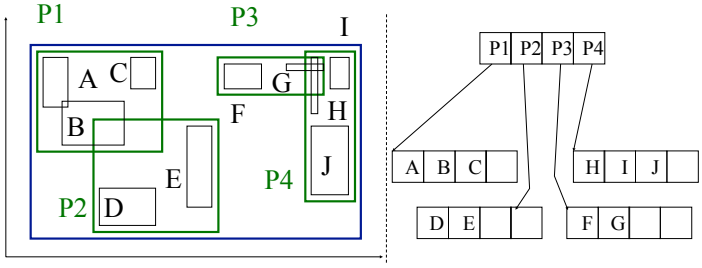
- F=4



Example

George Kollios

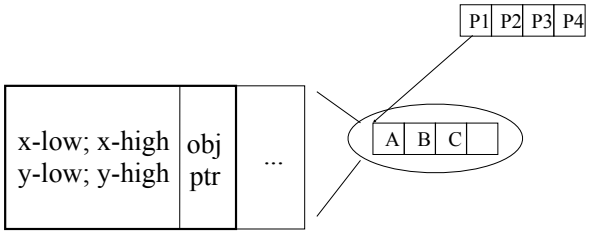
- F=4



R-trees - format of nodes

George Kollios

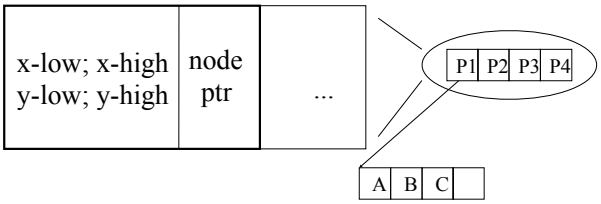
- {(MBR; obj_ptr)} for leaf nodes



R-trees - format of nodes

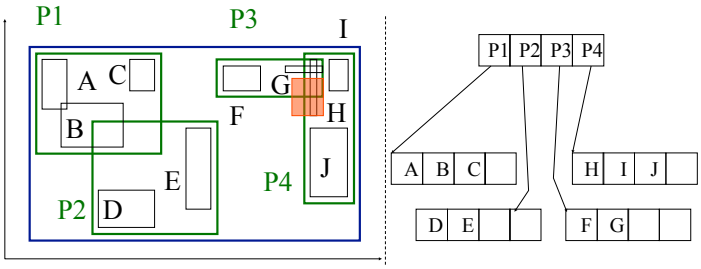
George Kollios

- {(MBR; node_ptr)} for non-leaf nodes



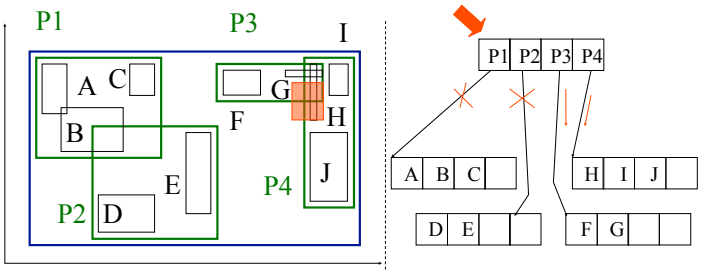
R-trees:Search

George Kollios



R-trees:Search

George Kollios



R-trees:Search

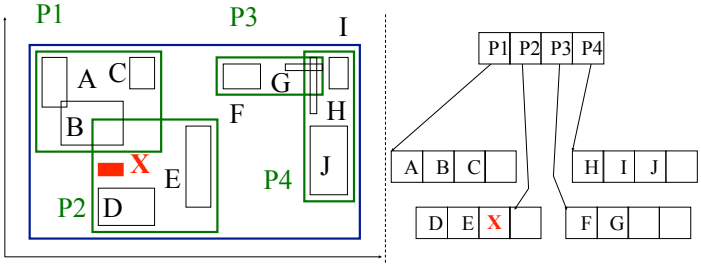
George Kollios

- Main points:
  - every parent node completely covers its 'children'
  - a child MBR may be covered by more than one parent - it is stored under ONLY ONE of them. (ie., no need for dup. elim.)
  - a point query may follow multiple branches.
  - everything works for **any(?)** dimensionality

R-trees:Insertion

George Kollios

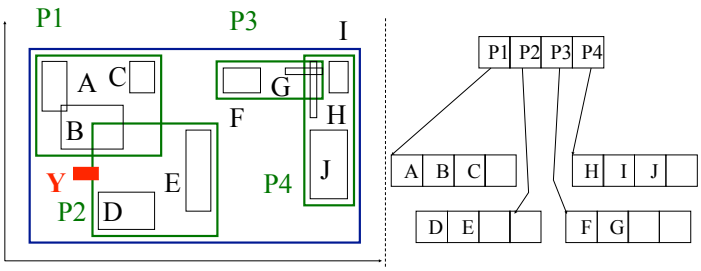
Insert X



George Kollios

R-trees:Insertion

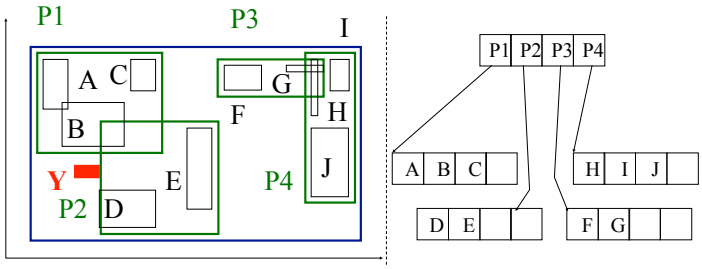
Insert Y



George Kollios

R-trees:Insertion

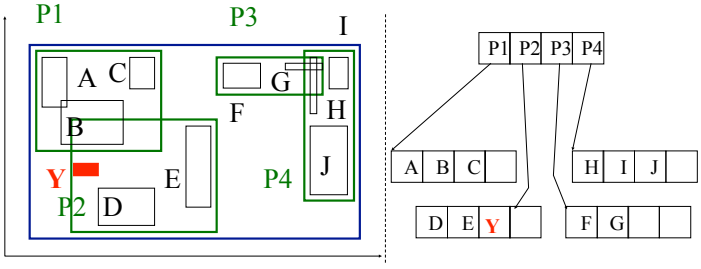
- Extend the parent MBR



George Kollios

R-trees:Insertion

- Extend the parent MBR



George Kollios

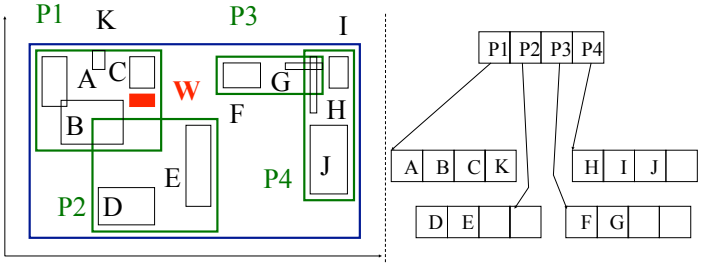
R-trees:Insertion

- How to find the next node to insert the new object?
  - Using ChooseLeaf: Find the entry that needs the least enlargement to include Y. Resolve ties using the area (smallest)
- Other methods ...

George Kollios

R-trees:Insertion

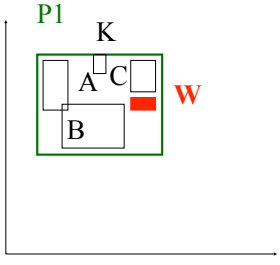
- If node is full then Split : ex. Insert w



George Kollios

R-trees:Split

- Split node P1: partition the MBRs into two groups.

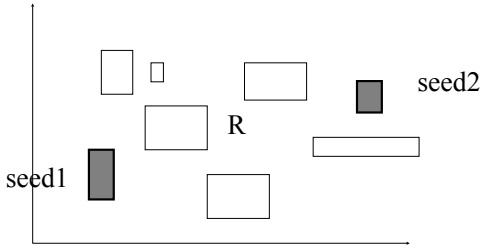


- (A1: plane sweep, until 50% of rectangles)
- A2: 'linear' split
- A3: quadratic split
- A4: exponential split: 2^{M-1} choices

George Kollios

R-trees:Split

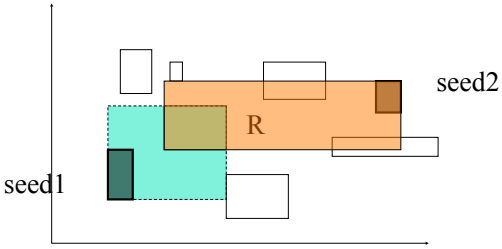
- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'



George Kollios

R-trees:Split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed':
- 'closest': the smallest increase in area



George Kollios

R-trees:Insertion

- Use the **ChooseLeaf** to find the leaf node to insert an entry E
- If leaf node is full, then **Split,** otherwise insert there
  - Propagate the split upwards, if necessary
- Adjust parent nodes

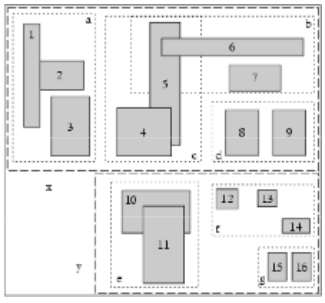
George Kollios

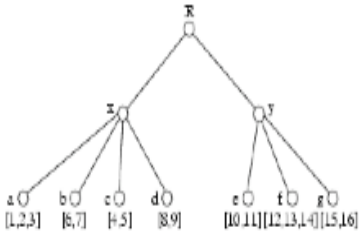
R-Trees:Deletion

- Find the leaf node that contains the entry E
- Remove E from this node
- If underflow:
  - Eliminate the node by removing the node entries and the parent entry
  - Reinsert the orphaned (other entries) into the tree using **Insert**

The lecture notes are based on slides by Shashi Shekhar

Spatial Objects with R-Tree





The lecture notes are based on slides by Shashi Shekhar

Spatial Objects with R-Tree

- Properties of R-trees
  - Balanced
  - Nodes are rectangle
    - child's rectangle within parent's
    - possible overlap among rectangles!
  - Other properties in section 4.2.2
- Implementation of find operation
  - Search root to identify relevant children
  - Search selected children recursively
- Ex.: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5

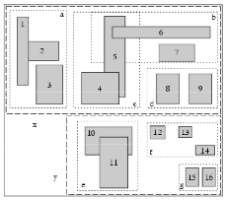
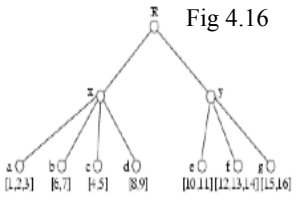
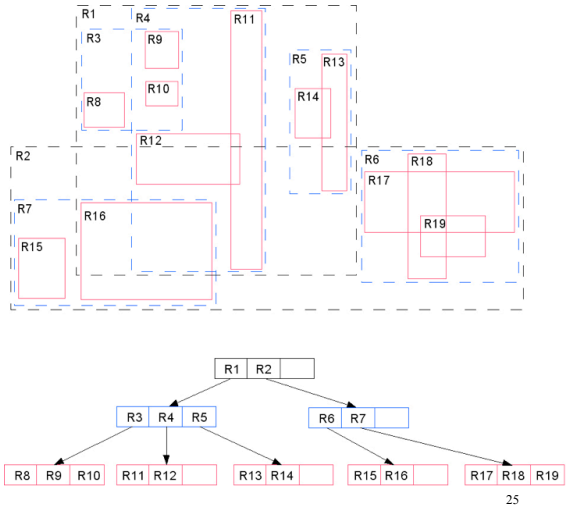


Fig 4.15

Fig 4.16

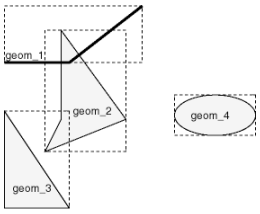


The lecture notes are based on slides by Shashi Shekhar



Spatial queries in Oracle Spatial

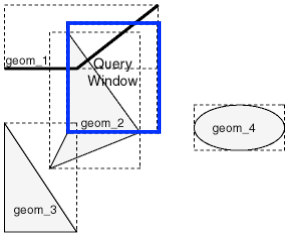
Minimal bounding boxes



remember:

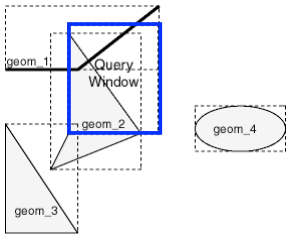
In a spatial R-tree index, each geometry is represented by its minimum bounding rectangle (MBR).

Query windows and minimal bounding boxes



A typical spatial query is to request all objects that lie within a **query window**

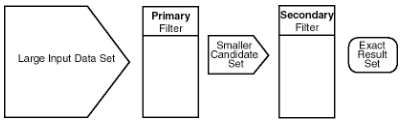
Query windows and minimal bounding boxes



the query window covers

- **parts of geometries** geom_1 and geom_2,
- **part of the MBR** for geom_3
- **none** of the actual geom_3 geometry.

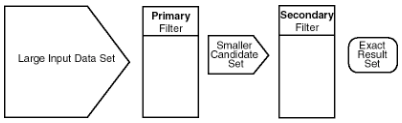
Query Model



Spatial uses a **two-tier query model** to resolve spatial queries

- The **primary filter**
  - Compares **geometry approximations** (MBRs) to reduce computation complexity
  - Because the primary filter compares geometric approximations, it returns a superset of the exact result set.
  - Permits fast selection of candidate records to pass along to the secondary filter.
  - Is considered a lower-cost filter.

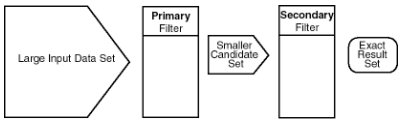
Query Model

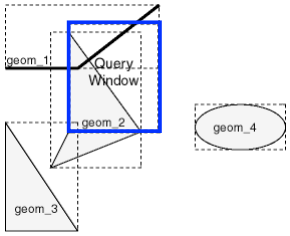


Spatial uses a **two-tier query model** to resolve spatial queries

- The **secondary filter**
  - applies exact computations to geometries that result from the primary filter.
  - yields an accurate answer to a spatial query.
  - is computationally expensive, but it is only applied to the primary filter results, not the entire data set.

Query Model





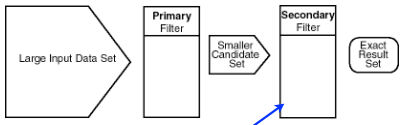
primary filter

- geom_1, geom_2, and geom_3

secondary filter

- geom_1 and geom_2

Spatial Relationships and Filtering



Spatial uses secondary filters to determine the spatial relationship between entities in the database.

Spatial Relationships and Filtering

The spatial relationship is based on geometry locations. The most common spatial relationships are based on topology and distance.

Topology

The *boundary* of an area consists of a set of curves that separates the area from the rest of the coordinate space.

The *interior* of an area consists of all points in the area that are not on its boundary.

Example of a topological relation:
two areas are said to be adjacent if they share part of a boundary but do not share any points in their interior.

Spatial Relationships and Filtering

The spatial relationship is based on geometry locations. The most common spatial relationships are based on topology and distance.

Distance

The distance between two spatial objects is the minimum distance between any points in them.

Two objects are said to be *within a given distance* of one another if their distance is less than the given distance.

Spatial Relationships and Filtering

To determine spatial relationships, Spatial has several secondary filter methods:

- The SDO_RELATE operator evaluates topological criteria.
- The SDO_WITHIN_DISTANCE operator determines if two spatial objects are within a specified distance of each other.
- The SDO_NN operator identifies the nearest neighbors for a spatial object.

Spatial Relationships and Filtering: SDO_RELATE

The SDO_RELATE operator implements a nine-intersection model for categorizing binary topological relationships between points, lines, and polygons.

- Each spatial object has an interior, a boundary, and an exterior.
- boundaries of geometric primitives
  - The boundary consists of points or lines that separate the interior from the exterior.
  - lines
    - The boundary of a line string consists of its end points;
    - if the end points overlap (that is, if they are the same point), the line string has no boundary.

Spatial Relationships and Filtering: SDO_RELATE

The SDO_RELATE operator implements a nine-intersection model for categorizing binary topological relationships between points, lines, and polygons.

- Each spatial object has an interior, a **boundary**, and an exterior.
- boundaries of geometric primitives
  - The boundary consists of points or lines that separate the interior from the exterior.
  - multi-lines
    - The boundaries of a multiline string are the end points of each of the component line strings;
    - if the end points overlap, only the end points that overlap an odd number of times are boundaries.

Spatial Relationships and Filtering: SDO_RELATE

The SDO_RELATE operator implements a nine-intersection model for categorizing binary topological relationships between points, lines, and polygons.

- Each spatial object has an interior, a **boundary**, and an exterior.
- boundaries of geometric primitives
  - The boundary consists of points or lines that separate the interior from the exterior.
  - polygons
    - The boundary of a polygon is the line that describes its perimeter.
    - outer boundaries
    - boundaries of holes

Spatial Relationships and Filtering: SDO_RELATE

The SDO_RELATE operator implements a nine-intersection model for categorizing binary topological relationships between points, lines, and polygons.

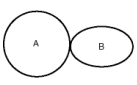
- Each spatial object has an **interior**, a boundary, and an **exterior**.
- the interior consists of points that are in the object but not on its boundary
- the exterior consists of those points that are not in the object.

Spatial Relationships and Filtering: SDO_RELATE

Given that an object A has three components (a boundary Ab, an interior Ai, and an exterior Ae), any pair of objects has nine possible interactions between their components.

Pairs of components have an empty (0) or not empty (1) set intersection.

The set of interactions between two geometries is represented by a nine-intersection matrix that specifies which pairs of components intersect and which do not.



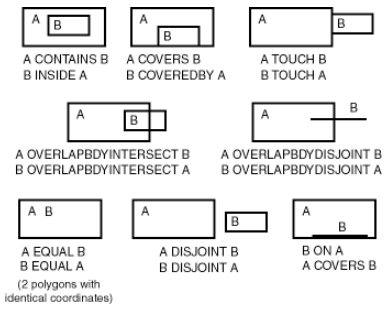
A TOUCH B

9-Intersection Matrix

nine-intersection matrix for two polygons that are adjacent to one another.

This matrix yields the following bit mask, generated in row-major form: "101001111".

Spatial Relationships and Filtering: SDO_RELATE



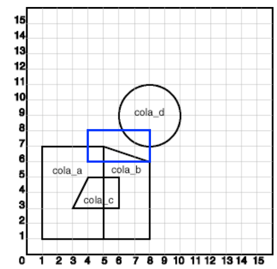
Spatial Relationships and Filtering: SDO_RELATE

Spatial uses the following names:

- DISJOINT -- The boundaries and interiors do not intersect.
- TOUCH -- The boundaries intersect but the interiors do not intersect.
- OVERLAPBDYDISJOINT -- The interior of one object intersects the boundary and interior of the other object, but the two boundaries do not intersect. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- OVERLAPBDYINTERSECT -- The boundaries and interiors of the two objects intersect.
- EQUAL -- The two objects have the same boundary and interior.
- CONTAINS -- The interior and boundary of one object is completely contained in the interior of the other object.
- COVERS -- The interior of one object is completely contained in the interior or the boundary of the other object and their boundaries intersect.
- INSIDE -- The opposite of CONTAINS. A INSIDE B implies B CONTAINS A.
- COVEREDBY -- The opposite of COVERS. A COVEREDBY B implies B COVERS A.
- ON -- The interior and boundary of one object is on the boundary of the other object (and the second object covers the first object). This relationship occurs, for example, when a line is on the boundary of a polygon.
- ANYINTERACT -- The objects are non-disjoint.

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).



the areas of interest for four colas

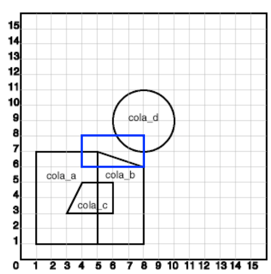
Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

ANYINTERACT -- The objects are non-disjoint.

```
expected results:

cola_b, cola_a, cola_d
```

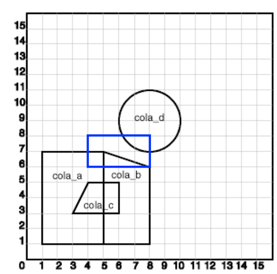


the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
             SDO_GEOMETRY(2003, NULL, NULL,
               SDO_ELEM_INFO_ARRAY(1,1003,3),
               SDO_ORDINATE_ARRAY(4,6, 8,8)),
               'mask=ANYINTERACT') = 'TRUE';
```

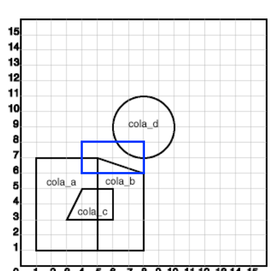


the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
             SDO_GEOMETRY(2003, NULL, NULL,
               SDO_ELEM_INFO_ARRAY(1,1003,3),
               SDO_ORDINATE_ARRAY(4,6, 8,8)),
               'mask=ANYINTERACT') = 'TRUE';
```

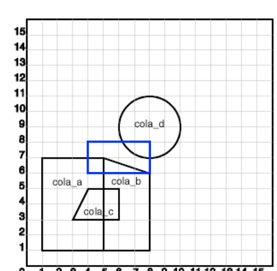


the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
             SDO_GEOMETRY(2003, NULL, NULL,
               SDO_ELEM_INFO_ARRAY(1,1003,3),
               SDO_ORDINATE_ARRAY(4,6, 8,8)),
               'mask=ANYINTERACT') = 'TRUE';
```



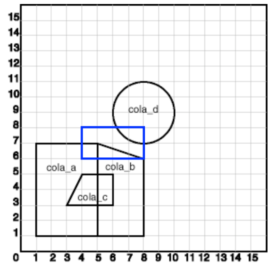
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8)),
              'mask=ANYINTERACT') = 'TRUE';
```

SDO_RELATE(geometry1, geometry2, param);



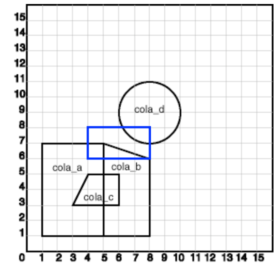
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8)),
              'mask=ANYINTERACT') = 'TRUE';
```

SDO_RELATE(geometry1, geometry2, param);



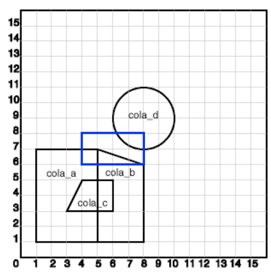
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8)),
              'mask=ANYINTERACT') = 'TRUE';
```

SDO_RELATE(geometry1, geometry2, param);



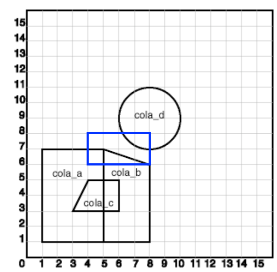
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8)),
              'mask=ANYINTERACT') = 'TRUE';
```

SDO_RELATE(geometry1, geometry2, param);



the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

Spatial uses the following names:

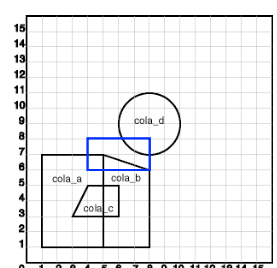
- DISJOINT -- The boundaries and interiors do not intersect.
- TOUCH -- The boundaries intersect but the interiors do not intersect.
- OVERLAPBDYDISJOINT -- The interior of one object intersects the boundary and interior of the other object, but the two boundaries do not intersect. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- OVERLAPBDYINTERSECT -- The boundaries and interiors of the two objects intersect.
- EQUAL -- The two objects have the same boundary and interior.
- CONTAINS -- The interior and boundary of one object is completely contained in the interior of the other object.
- COVERS -- The interior of one object is completely contained in the interior or the boundary of the other object and their boundaries intersect.
- INSIDE -- The opposite of CONTAINS. A INSIDE B implies B CONTAINS A.
- COVEREDBY -- The opposite of COVERS. A COVEREDBY B implies B COVERS A.
- ON -- The interior and boundary of one object is on the boundary of the other object (and the second object covers the first object). This relationship occurs, for example, when a line is on the boundary of a polygon.
- ANYINTERACT -- The objects are non-disjoint.

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_RELATE(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8)),
              'mask=ANYINTERACT') = 'TRUE';
```

```
    MKT_ID NAME
---------- --------------------------------
         2 cola_b
         1 cola_a
         4 cola_d
```



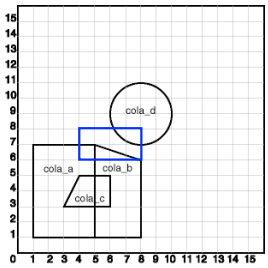
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_RELATE

This example finds geometries that have the ANYINTERACT relationship with a query window: a rectangle with lower-left, upper-right coordinates (4,6), (8,8).

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_ANYINTERACT(c.shape,
            SDO_GEOMETRY(2003, NULL, NULL,
              SDO_ELEM_INFO_ARRAY(1,1003,3),
              SDO_ORDINATE_ARRAY(4,6, 8,8))
            ) = 'TRUE';
```

```
       MKT_ID NAME
 ---------- --------------------------------
          2 cola_b
          1 cola_a
          4 cola_d
```



the areas of interest for four colas

Spatial Operators, Procedures, and Functions

- Spatial operators, such as SDO_FILTER and SDO_RELATE, provide optimum performance because they use the spatial index. (Spatial operators require that the geometry column in the first parameter have a spatial index defined on it.)
- Spatial operators must be used in the WHERE clause of a query.
- The first parameter of any operator specifies the geometry column to be searched, and the second parameter specifies a query window.

Spatial Relationships and Filtering

The spatial relationship is based on geometry locations. The most common spatial relationships are based on topology and distance.

To determine spatial relationships, Spatial has several secondary filter methods:

- The SDO_RELATE operator evaluates topological criteria.
- The SDO_WITHIN_DISTANCE operator determines if two spatial objects are within a specified distance of each other.
- The SDO_NN operator identifies the nearest neighbors for a spatial object.

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

SDO_WITHIN_DISTANCE

determines if two spatial objects, A and B, are within a specified distance of one another.

constructs first a distance buffer, D_b, around the reference object B. then checks that A and D_b are non-disjoint.

The distance buffer of an object consists of all points within the given distance from that object:
(The dashed lines represent distance buffers. Notice how the buffer is rounded near the corners of the objects)

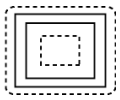
distance buffers for

a point,

a line, and

a polygon (with a hole)

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

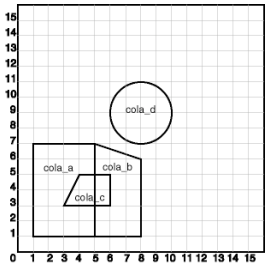


polygon with a hole:

- the large rectangle is the exterior polygon ring
- the small rectangle is the interior polygon ring (the hole).
- The dashed line outside the large rectangle is the buffer for the exterior ring,
- the dashed line inside the small rectangle is the buffer for the interior ring.

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)



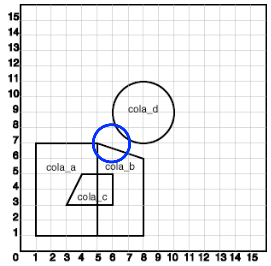
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

Expected results:

```
cola_b, cola_a, cola_d
```



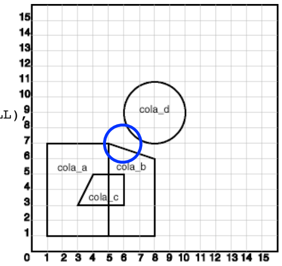
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_WITHIN_DISTANCE(c.shape,
            SDO_GEOMETRY(2001, NULL,
            SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),
              'distance=1') = 'TRUE';
```

```
SDO_WITHIN_DISTANCE(geometry1, aGeom,
                    'distance = <some_dist_val>')
```



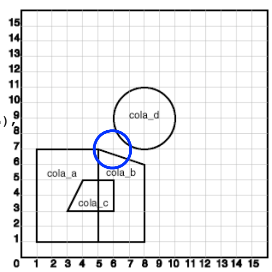
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_WITHIN_DISTANCE(c.shape,
            SDO_GEOMETRY(2001, NULL,
            SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),
              'distance=1') = 'TRUE';
```

```
SDO_WITHIN_DISTANCE(geometry1, aGeom,
                    'distance = <some_dist_val>')
```



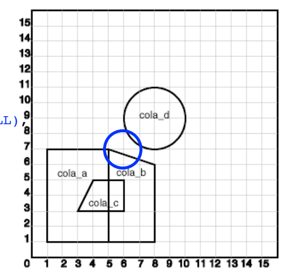
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_WITHIN_DISTANCE(c.shape,
            SDO_GEOMETRY(2001, NULL,
            SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),
              'distance=1') = 'TRUE';
```

```
SDO_WITHIN_DISTANCE(geometry1, aGeom,
                    'distance = <some_dist_val>')
```



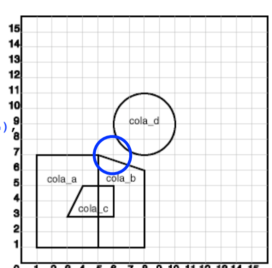
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_WITHIN_DISTANCE(c.shape,
            SDO_GEOMETRY(2001, NULL,
            SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),
              'distance=1') = 'TRUE';
```

```
SDO_WITHIN_DISTANCE(geometry1, aGeom,
                    'distance = <some_dist_val>')
```



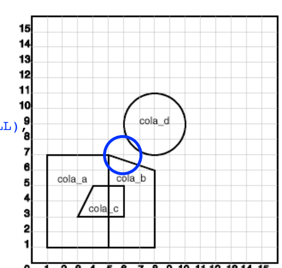
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_WITHIN_DISTANCE

This example finds geometries within a 1-unit buffer around the point (6,7)

```
SELECT c.mkt_id, c.name
  FROM cola_markets c
  WHERE SDO_WITHIN_DISTANCE(c.shape,
            SDO_GEOMETRY(2001, NULL,
            SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),
              'distance=1') = 'TRUE';
```

```
MKT_ID NAME
---------- --------------------------------
     2 cola_b
     1 cola_a
     4 cola_d
```



the areas of interest for four colas

Spatial Relationships and Filtering

The spatial relationship is based on geometry locations. The most common spatial relationships are based on topology and distance.

To determine spatial relationships, Spatial has several secondary filter methods:

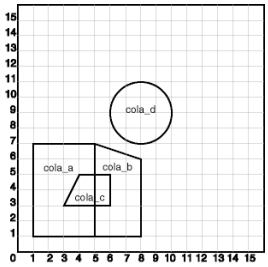
- The SDO_RELATE operator evaluates topological criteria.
- The SDO_WITHIN_DISTANCE operator determines if two spatial objects are within a specified distance of each other.
- The SDO_NN operator identifies the nearest neighbors for a spatial object.

Spatial Relationships and Filtering: SDO_NN

- The SDO_NN operator returns a specified number of objects from a geometry column that are closest to a specified geometry (for example, the five closest restaurants to a city park).
- In determining how close two geometry objects are, the shortest possible distance between any two points on the surface of each object is used.

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).



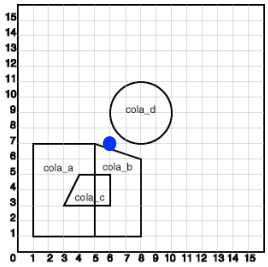
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

expected results:

cola_b, cola_d, cola_a, cola_c



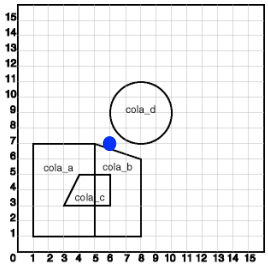
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE';
```

SDO_NN(geometry1, geometry2, param [, number]);



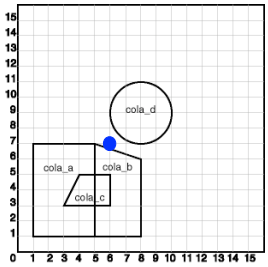
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE';
```

SDO_NN(geometry1, geometry2, param [, number]);



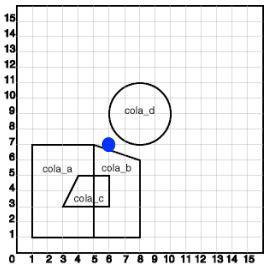
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE';
```

SDO_NN(geometry1, geometry2, param [, number]);



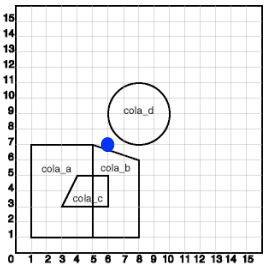
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE';
```

SDO_NN(geometry1, geometry2, param [, number]);



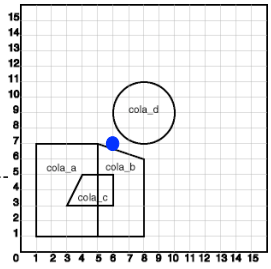
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE';
```

```
    MKT_ID NAME
---------- --------------------------------
         2 cola_b
         4 cola_d
         1 cola_a
         3 cola_c
```



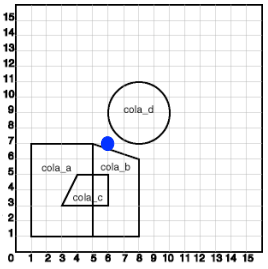
the areas of interest for four colas

Spatial Relationships and Filtering: SDO_NN

This example returns geometries in the order of their distances to the point (6,7).

```
SELECT c.mkt_id, c.name
   FROM cola_markets c
   WHERE SDO_NN(c.shape,  sdo_geometry(2001, NULL,
      sdo_point_type(6,7,NULL), NULL,  NULL)) = 'TRUE'
     AND ROWNUM <=2;
```

```
    MKT_ID NAME
---------- --------------------------------
         2 cola_b
         4 cola_d
```



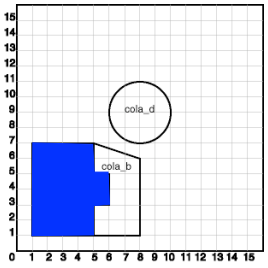
the areas of interest for four colas

Spatial Unions and Intersections

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

Expected result?

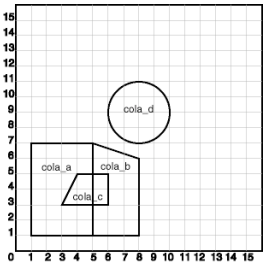


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SELECT
   SDO_GEOM.SDO_UNION(c_a.shape,
                      c_c.shape, 0.005)
   FROM cola_markets c_a, cola_markets c_c
   WHERE c_a.name = 'cola_a' AND
         c_c.name = 'cola_c';
```

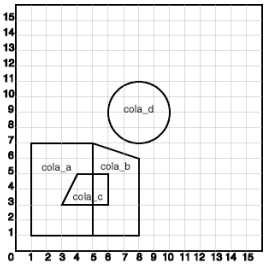


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SELECT
   SDO_GEOM.SDO_UNION(c_a.shape,
                      c_c.shape, 0.005)
   FROM cola_markets c_a, cola_markets c_c
   WHERE c_a.name = 'cola_a' AND
         c_c.name = 'cola_c';
```



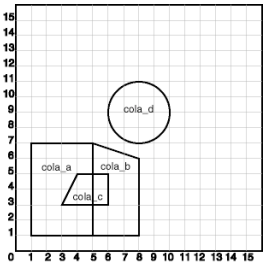
the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SELECT
   SDO_GEOM.SDO_UNION(c_a.shape,
                      c_c.shape, 0.005)
   FROM cola_markets c_a, cola_markets c_c
   WHERE c_a.name = 'cola_a' AND
         c_c.name = 'cola_c';
```

We address distinct geometries in the same geometry column.

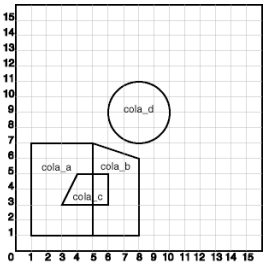


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SELECT
   SDO_GEOM.SDO_UNION(c_a.shape,
                      c_c.shape, 0.005)
   FROM cola_markets c_a, cola_markets c_c
   WHERE c_a.name = 'cola_a' AND
         c_c.name = 'cola_c';
```

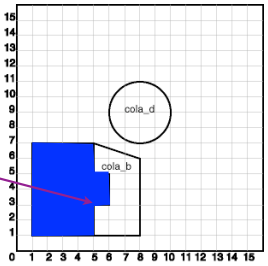


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

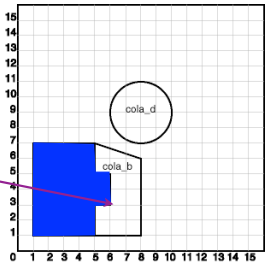


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

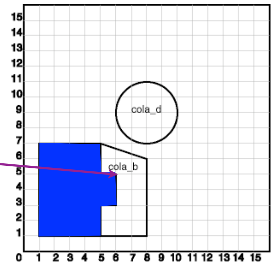


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

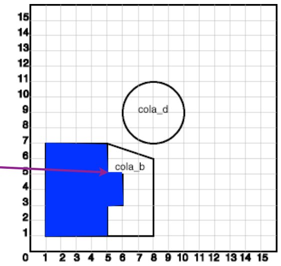


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

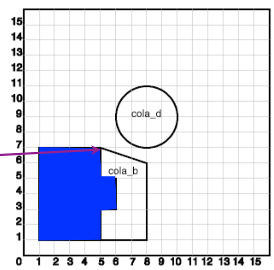


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

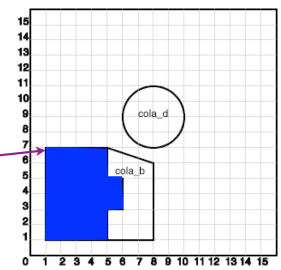


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

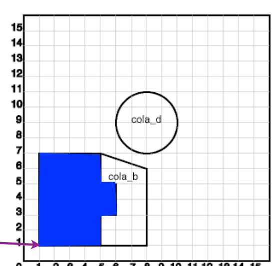


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```

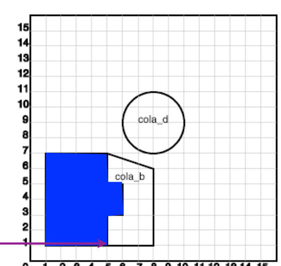


the areas of interest for four colas

Computing unions and intersections

This example query returns the topological union of the geometries of 'cola_a' and 'cola_c'.

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1),
SDO_ORDINATE_ARRAY(
5, 3,
6, 3,
6, 5,
5, 5,
5, 7,
1, 7,
1, 1,
5, 1,
5, 3))
```



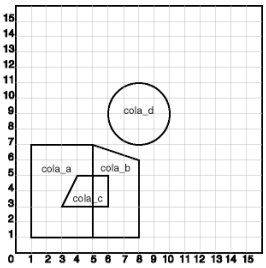
the areas of interest for four colas

Other spatial queries

Other spatial queries

This example returns the area of cola_d.

```
SELECT c.name,
       SDO_GEOM.SDO_AREA(c.shape, 0.005)
FROM cola_markets c
       WHERE c.name = 'cola_d';
```



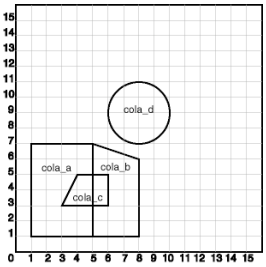
the areas of interest for four colas

Other spatial queries

This example returns the area of cola_d.

```
SELECT c.name,
       SDO_GEOM.SDO_AREA(c.shape, 0.005)
FROM cola_markets c
       WHERE c.name = 'cola_d';
```

```
NAME       SDO_GEOM.SDO_AREA(C.SHAPE,0.005)
---------- --------------------------------
cola_d     12.5663706
```



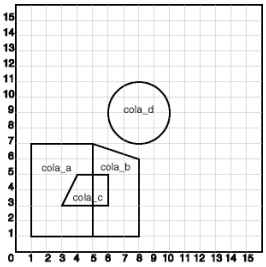
the areas of interest for four colas

Other spatial queries

This example returns the area of cola_d.

```
SELECT c.name,
       SDO_GEOM.SDO_AREA(c.shape, 0.005)
FROM cola_markets c
       WHERE c.name = 'cola_d';
```

You can find more geometry functions which you need for lab 7 in the file Oracle_Geometry_Functions.html on ublearns



the areas of interest for four colas