**4.2 Querying Spatial Data**

This section describes how the structures of a Spatial layer are used to resolve spatial queries and spatial joins.

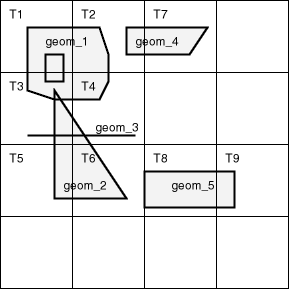
Spatial uses a two-tier query model with primary and secondary filter operations to resolve spatial queries and spatial joins, as explained in [Section 1.6](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_intro.htm#g1000087). The term *two-tier* is used to indicate that two distinct operations are performed in order to resolve queries. If both operations are performed, the exact result set is returned.

**4.2.1 Spatial****Query**

An important concept in the spatial data model is that each geometry is represented by a set of exclusive and exhaustive tiles. This means that no tiles overlap each other (**exclusive**), and the tiles fully cover the object (**exhaustive**).

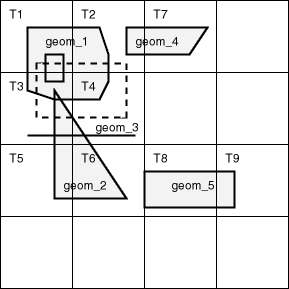
Consider the following layer containing several objects in [Figure 4-4](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005081). Each object is labeled with its geometry name (geom\_1 for the polygon with a hole, geom\_2 for the triangular polygon, geom\_3 for the line, and geom\_4 and geom\_5 for other polygons). The relevant tiles are labeled with T*n* (T1, T2, and so on)

***Figure 4-4 Tessellated Layer with Multiple Objects***

  
[Description of the illustration tess\_mult\_obj.gif](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/img_text/tess_mult_obj.htm)

A typical spatial query is to request all objects that lie within a defined fence or window. A **query window**is shown in [Figure 4-5](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005179) by the dotted-line box. A dynamic query window refers to a fence that is not defined in the database, but that must be defined before it is used.

***Figure 4-5 Tessellated Layer with a Query Window***

  
[Description of the illustration tess\_query\_win.gif](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/img_text/tess_query_win.htm)

**4.2.1.1 Primary****Filter Operator**

The [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator implements the primary filter portion of the two-step process involved in the product's query processing model. The primary filter uses the index data only to determine a set of candidate object pairs that may interact. The syntax is as follows:

SDO\_FILTER(geometry1 MDSYS.SDO\_GEOMETRY, geometry2 MDSYS.SDO\_GEOMETRY,

  params VARCHAR2)

Where:

* *geometry1* is a column of type MDSYS.SDO\_GEOMETRY in a table. *geometry1* must be spatially indexed.
* *geometry2* is an object of type MDSYS.SDO\_GEOMETRY. *geometry2* may or may not come from a table. If it comes from a table, it may or may not be spatially indexed.
* *params* is a quoted string of keyword value pairs that determine the behavior of the operator. See the [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037) for a list of parameters.

The following examples perform a primary filter operation only. They will return all the geometries shown in [Figure 4-5](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005179) that have an index tile in common with one of the index tiles that approximates the query window: tiles T1, T2, T3, and T4. The result of the following examples are geometries with IDs 1013, 1243, 12, and 501.

[Example 4-2](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005295) performs a primary filter operation without inserting the query window into a table. The window will be indexed in memory and performance will be very good.

***Example 4-2 Primary Filter with a Temporary Query Window***

SELECT A.Feature\_ID FROM TARGET A

WHERE sdo\_filter(A.shape, mdsys.sdo\_geometry(2003,NULL,NULL,

                                       mdsys.sdo\_elem\_info\_array(1,1003,3),

                                       mdsys.sdo\_ordinate\_array(x1,y1, x2,y2)),

                        'querytype=window') = 'TRUE';

Note that *(x1,y1)* and *(x2,y2)* are the lower-left and upper-right corners of the query window.

In [Example 4-3](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005306), a transient instance of type SDO\_GEOMETRY was constructed for the query window instead of specifying the window parameters in the query itself.

***Example 4-3 Primary Filter with a Transient Instance of the Query Window***

SELECT A.Feature\_ID FROM TARGET A

WHERE sdo\_filter(A.shape, :theWindow,'querytype=window') = 'TRUE';

[Example 4-4](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005314) assumes the query window was inserted into a table called WINDOWS, with an ID of WINS\_1.

***Example 4-4 Primary Filter with a Stored Query Window***

SELECT A.Feature\_ID FROM TARGET A, WINDOWS B

WHERE B.ID = 'WINS\_1' AND

  sdo\_filter(A.shape, B.shape,'querytype=window') = 'TRUE';

If the B.SHAPE column is not spatially indexed, the [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator indexes the query window in memory and performance is very good.

If the B.SHAPE column is spatially indexed with the same SDO\_LEVEL value as the A.SHAPE column, the [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator reuses the existing index, and performance is very good or better.

If the B.SHAPE column is spatially indexed with a different SDO\_LEVEL value than the A.SHAPE column, the [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator reindexes B.SHAPE in the same way as if there were no index on the column originally, and then performance is very good.

**4.2.1.2 Primary and****Secondary Filter Operator**

The [SDO\_RELATE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78531) operator performs both the primary and secondary filter stages when processing a query. This operator can be used only if a spatial index has been created on two dimensions of data. The syntax of the operator is as follows:

SDO\_RELATE(geometry1  MDSYS.SDO\_GEOMETRY,

            geometry2  MDSYS.SDO\_GEOMETRY,

            params     VARCHAR2)

Where:

* *geometry1* is a column of type MDSYS.SDO\_GEOMETRY in a table. *geometry1* must be spatially indexed.
* *geometry2* is an object of type MDSYS.SDO\_GEOMETRY. *geometry2* may or may not come from a table. If it comes from a table, it may or may not be spatially indexed.
* *params* is a quoted string of keyword value pairs that determine the behavior of the operator. See the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037) for a list of parameters.

The following examples perform both primary and secondary filter operations. They return all the geometries in [Figure 4-5](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005179) that lie within or overlap the query window. The result of these examples is objects 1243 and 1013.

[Example 4-5](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005358) performs both primary and secondary filter operations without inserting the query window into a table. The window will be indexed in memory and performance will be very good.

***Example 4-5 Secondary Filter Using a Temporary Query Window***

SELECT A.Feature\_ID FROM TARGET A

   WHERE sdo\_relate(A.shape, mdsys.sdo\_geometry(2003,NULL,NULL,

                                       mdsys.sdo\_elem\_info\_array(1,1003,3),

                                      mdsys.sdo\_ordinate\_array(x1,y1, x2,y2)),

                        'mask=anyinteract querytype=window') = 'TRUE';

Note that *(x1,y1)* and *(x2,y2)* are the lower-left and upper-right corners of the query window.

[Example 4-6](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm#i1005370) assumes the query window was inserted into a table called WINDOWS, with an ID of WINS\_1.

***Example 4-6 Secondary Filter Using a Stored Query Window***

SELECT A.Feature\_ID FROM TARGET A, WINDOWS B

WHERE B.ID= 'WINS\_1' AND

        sdo\_relate(A.shape, B.shape,

          'mask=anyinteract querytype=window') = 'TRUE';

If the B.SHAPE column is not spatially indexed, the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator indexes the query window in memory and performance is very good.

If the B.SHAPE column is spatially indexed with the same SDO\_LEVEL value as the A.SHAPE column, the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator reuses the existing index, and performance is very good or better.

If the B.SHAPE column is spatially indexed with a different SDO\_LEVEL value than the A.SHAPE column, the [SDO\_FILTER](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#BJAFBCFC) operator reindexes B.SHAPE in the same way as if there were no index on the column originally, and then performance is very good.

**4.2.1.3 Within-Distance Operator**

The [SDO\_WITHIN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i77653) operator is used to determine the set of objects in a table that are within*n* distance units from a reference object. This operator can be used only if a spatial index has been created on two dimensions of data. The reference object may be a transient or persistent instance of MDSYS.SDO\_GEOMETRY (such as a temporary query window or a permanent geometry stored in the database). The syntax of the operator is as follows:

SDO\_WITHIN\_DISTANCE(geometry1  MDSYS.SDO\_GEOMETRY,

                    aGeom      MDSYS.SDO\_GEOMETRY,

                    params     VARCHAR2);

Where:

* *geometry1* is a column of type MDSYS.SDO\_GEOMETRY in a table. *geometry1* must be spatially indexed.
* *aGeom* is an instance of type MDSYS.SDO\_GEOMETRY.
* *params* is a quoted string of keyword value pairs that determines the behavior of the operator. See the [SDO\_WITHIN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i77653) operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037) for a list of parameters.

The following example selects any objects within 1.35 distance units from the query window:

SELECT A.Feature\_ID

  FROM TARGET A

  WHERE SDO\_WITHIN\_DISTANCE( A.shape, :theWindow, 'distance=1.35') = 'TRUE';

The distance units are based on the geometry coordinate system in use. The distance units are those specified in the UNIT field of the well-known text (WKTEXT) associated with the coordinate system. (The WKTEXT is explained in [Section 5.4.1.1](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_cs_concepts.htm#i890447).) If you are using a geodetic coordinate system, the units are meters. If no coordinate system is used, the units are the same as for the stored data.

The [SDO\_WITHIN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i77653) operator is not suitable for performing spatial joins. That is, a query such as *Find all parks that are within 10 distance units from coastlines* will not be processed as an index-based spatial join of the COASTLINES and PARKS tables. Instead, it will be processed as a nested loop query in which each COASTLINES instance is in turn a reference object that is buffered, indexed, and evaluated against the PARKS table. Thus, the [SDO\_WITHIN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i77653) operation is performed *n* times if there are *n* rows in the COASTLINES table.

For non-geodetic data, there is an efficient way to accomplish a spatial join that involves buffering all the geometries of a layer. This method does not use the [SDO\_WITHIN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i77653) operator. First, create a new table COSINE\_BUFS as follows:

CREATE TABLE cosine\_bufs UNRECOVERABLE AS

   SELECT SDO\_BUFFER (A.SHAPE, B.DIMINFO, 1.35)

     FROM COSINE A, USER\_SDO\_GEOM\_METADATA B

     WHERE TABLE\_NAME='COSINES' AND COLUMN\_NAME='SHAPE';

Next, create a spatial index on the SHAPE column of COSINE\_BUFS. Then you can perform the following query:

SELECT a.gif, b.gid FROM parks A cosine\_bufs B

  WHERE SDO\_Relate(A.shape, B.shape, 'mask=ANYINTERACT querytype=JOIN') ='TRUE';

**4.2.1.4 Nearest Neighbor Operator**

The [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator is used to identify the nearest neighbors for a geometry. This operator can be used only if a spatial index has been created on two dimensions of data. The syntax of the operator is as follows:

SDO\_NN(geometry1  MDSYS.SDO\_GEOMETRY,

       geometry2  MDSYS.SDO\_GEOMETRY,

       param      VARCHAR2

       [, number  NUMBER]);

Where:

* *geometry1* is a column of type MDSYS.SDO\_GEOMETRY in a table. *geometry1* must be spatially indexed.
* *geometry2* is an instance of type MDSYS.SDO\_GEOMETRY.
* *param* is a quoted string of a keyword value pair that determines how many nearest neighbor geometries are returned by the operator. See the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037) for information about this parameter.
* *number* is the same number used in the call to [SDO\_NN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i80529). Use this only if the[SDO\_NN\_DISTANCE](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i80529) ancillary operator is included in the call to [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067). See the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067)operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037) for information about this parameter.

The following example finds the two objects from the SHAPE column in the COLA\_MARKETS table that are closest to a specified point (10,7). (Note the use of the optimizer hint in the SELECT statement, as explained in the Usage Notes for the [SDO\_NN](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#i78067) operator in [Chapter 10](http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_operat.htm#g1001037).)

SELECT /\*+ INDEX(cola\_markets cola\_spatial\_idx) \*/

 c.mkt\_id, c.name  FROM cola\_markets c  WHERE SDO\_NN(c.shape,

   mdsys.sdo\_geometry(2001, NULL, mdsys.sdo\_point\_type(10,7,NULL), NULL,

   NULL),  'sdo\_num\_res=2') = 'TRUE';

**4.2.1.5 Spatial Functions**

Spatial also supplies functions for determining relationships between geometries, finding information about single geometries, changing geometries, and combining geometries. These functions all take into account two dimensions of source data. If the output value of these functions is a geometry, the resulting geometry will have the same dimensionality as the input geometry, but only the first two dimensions will accurately reflect the result of the operation.

**4.2.2 Spatial Join**

A **spatial join** is the same as a regular join except that the predicate involves a spatial operator. In Spatial, a spatial join takes place when you compare all the geometries of one layer to all the geometries of another layer. This is unlike a query window that only compares a single geometry to all geometries of a layer.

In a spatial join, all tables must have the same type of spatial index (that is, R-tree or quadtree) defined on the geometry column; and if they have quadtree indexes, the SDO\_LEVEL value must be the same for all the indexes.

Spatial joins can be used to answer questions such as, *Which highways cross national parks?*

The following table structures illustrate how the join would be accomplished for this example:

PARKS(    GID VARCHAR2(32), SHAPE MDSYS.SDO\_GEOMETRY)

HIGHWAYS( GID VARCHAR2(32), SHAPE MDSYS.SDO\_GEOMETRY)

The primary filter would identify pairs of *GID* values from the PARKS and HIGHWAYS tables that interact in their index entries. The query that performs the primary filter join is:

SELECT A.GID, B.GID

  FROM PARKS A, HIGHWAYS B

  WHERE sdo\_filter(A.shape, B.shape, 'querytype=join') = 'TRUE';

The original question, asking about highways that cross national parks, requires the secondary filter operator to find the exact relationship between highways and parks.

The query that performs this join using both primary and secondary filters is:

SELECT A.GID, B.GID

  FROM parks A, highways B

  WHERE sdo\_relate(A.shape, B.shape,

           'mask=ANYINTERACT querytype=join');

**4.2.3 Cross-Schema Operator Invocation**

You can invoke spatial operators on an indexed table that is not in your schema. Assume that user A has a spatial table T1 (with index table IDX\_TAB1) with a spatial index defined, that user B has a spatial table T2 (with index table IDX\_TAB2) with a spatial index defined, and that user C wants to invoke operators on tables in one or both of the other schemas.

If user C wants to invoke an operator only on T1, user C must perform the following steps:

1. Connect as user A and execute the following statements:

GRANT select on T1 to C;

GRANT select on idx\_tab1 to C;

1. Connect as user C and execute a statement such as the following:

SELECT a.gid

  FROM T1 a

  WHERE sdo\_filter(a.geometry, :theGeometry, 'querytype=WINDOW') = 'TRUE';

If user C wants to invoke an operator on both T1 and T2, user C must perform the following steps:

1. Connect as user A and execute the following statements:

GRANT select on T1 to C;

GRANT select on idx\_tab1 to C;

1. Connect as user B and execute the following statements:

GRANT select on T2 to C;

GRANT select on idx\_tab2 to C;

1. Connect as user C and execute a statement such as the following:

SELECT a.gid

  FROM T1 a, T2 b

  WHERE b.gid = 5 AND

        sdo\_filter(a.geometry, b.geometry, 'querytype=WINDOW') = 'TRUE';

来源： <<http://docs.oracle.com/cd/B10501_01/appdev.920/a96630/sdo_index_query.htm>>