7

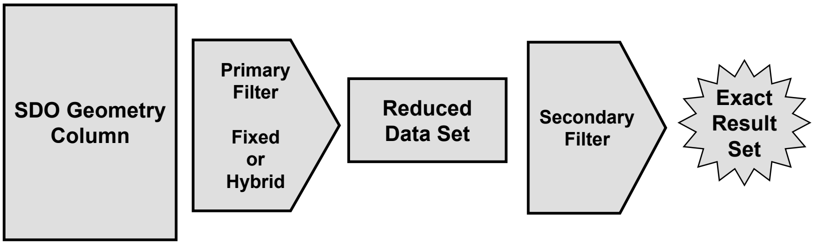
Object Relational Model
Spatial Queries

Objectives

After completing this lesson, you should be able to do the following:

- **Describe a spatial query and a spatial join**
- **Explain the query model**
- **Describe and compare spatial operators and functions**
- **Use the spatial operators and functions to perform spatial queries and joins**
- **Understand the topological relationships used by the spatial operators**

Query Model



SDO Geometry Column

Primary Filter

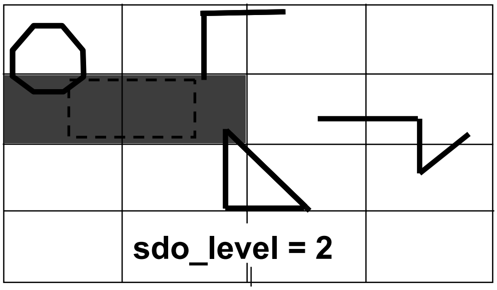
Fixed or Hybrid

Reduced Data Set

Secondary Filter

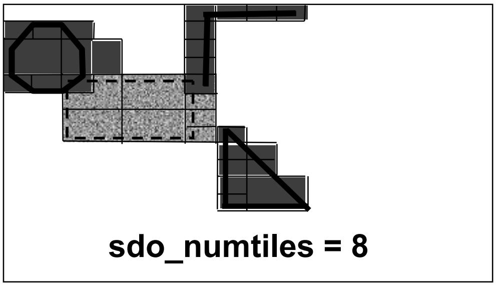
Exact Result Set

Primary Filter Example - FIXED



- Compares fixed sized tiles that approximate the area of interest with fixed sized tiles that approximate each geometry
- Result is not exact because comparing approximations

Primary Filter Example - HYBRID



- Hybrid filter:
  - First does a fixed tile comparison
  - Then does a variable tile comparison
  - Result is not exact because still comparing geometry approximations

Spatial Operators vs Functions

- Spatial operators:
  - Take advantage of spatial indexes
  - Require that spatial index exists on the first geometry specified in the operator
  - Only in the WHERE clause
- Spatial Functions:
  - Do not take advantage of spatial indexes
  - Could be used on small tables that are not spatially indexed
  - Can be used in the SELECT list or WHERE clause

Spatial Operators vs Functions

Operators

- SDO_FILTER
  - Performs a primary filter only
- SDO_RELATE
  - Performs a primary and secondary filter
- SDO_WITHIN_DISTANCE
  - Generates a buffer around a geometry and performs a primary and optionally a secondary filter
- SDO_NN
  - Returns nearest neighbors

Functions

- SDO_GEOM.RELATE
  - To determine the relationship between two geometries
  - To perform a spatial query without using a spatial index (i.e. on a small table)
- SDO_GEOM.WITHIN_DISTANCE
  - Generates a buffer around a geometry and performs a secondary filter

The SDO_FILTER operator

```
boolean := MDSYS.SDO_FILTER
 ( <geometry-1>,
   <geometry-2>,
   'QUERYTYPE=<querytype>
    [other optional parameters]'
 )
```

- Performs an approximate query (primary filter only)
- Returns TRUE or FALSE

Required arguments

- **GEOMETRY-1**
  - **Must be a column in a table**
  - **Must be of type SDO_GEOMETRY**
  - **Must be indexed**
- **GEOMETRY-2**
  - **Variable or column in a table**
  - **Must be of type SDO_GEOMETRY**
  - **If querytype=JOIN, must be a column in a table and indexed (very rare)**
- **QUERYTYPE**
  - **Valid values are WINDOW or JOIN (very rare)**

Optional arguments

- **IDXTAB1**
  - **Index table to associate with first geometry in operator.**
  - **By default, the primary index table is used.**
- **IDXTAB2**
  - **Index table to associate with the second geometry in operator**
  - **By default, the primary index table is used.**
  - **Only supported if QUERYTYPE=JOIN**

Optional arguments (continued)

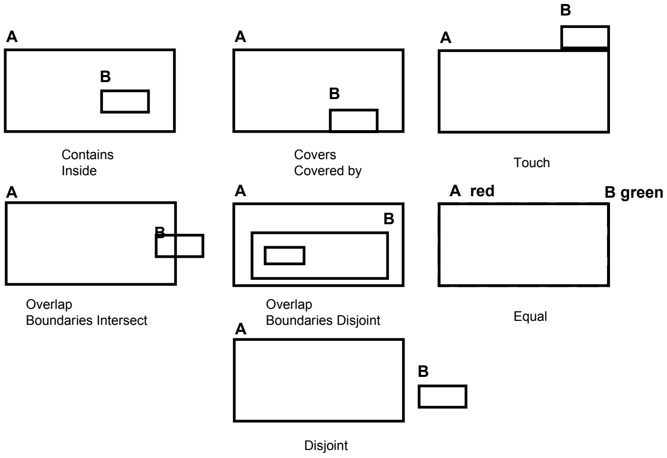
- LAYER_GTYPE
  - Set to POINT if querying POINT only columns (for optimal performance)
  - Otherwise, NOTPOINT (default)

SDO_FILTER Example

- Find all cities in a selected rectangular area
- Result is approximate

```
select c.city, c.pop90
  from cities c
 where mdsys.sdo_filter (
        c.location,
        mdsys.sdo_geometry (2003, null, null,
          mdsys.sdo_elem_info_array (1,1003,3),
          mdsys.sdo_ordinate_array (-109,37,-102,40)),
       'querytype=WINDOW layer_gtype=POINT') = 'TRUE';
```

Spatial (topological) relationships



Spatial (topological) relationships (cont.)

- **ANYINTERACT**
  - **Returns TRUE if not disjoint**
  - **Optimal mask**
  - **Only mask that takes advantage of center tile optimization**

The SDO_RELATE operator

```
MDSYS.SDO_RELATE
 ( <geometry-1>,
   <geometry-2>,
   ‘MASK=<mask>
    QUERYTYPE=<querytype>
    [other optional parameters]’
 ) = ‘TRUE’
```

- Performs a primary and secondary filter
- Returns TRUE or FALSE

Required arguments

- GEOMETRY-1
  - Must be a column in a table
  - Must be of type SDO_GEOMETRY
  - Must be indexed
- GEOMETRY-2
  - Variable or column in a table
  - Must be of type SDO_GEOMETRY
  - If querytype=JOIN, must be a column in a table and indexed (very rare)

Required arguments (cont.)

- MASK
  - Identify spatial relationship to test
  - Must be UPPER CASE in 8.1.5, case doesn't matter in 8.1.6+
  - OR'ed masks do not work in 8.1.5, fixed in 8.1.6+

    (i.e. INSIDE+COVEREDBY)
- QUERYTYPE
  - Valid values are WINDOW or JOIN (very rare)

Optional arguments

- IDXTAB1
  - Index table to associate with first geometry in operator.
  - By default, the primary index table is used.
- IDXTAB2
  - Index table to associate with the second geometry in operator
  - By default, the primary index table is used.
  - Only supported if QUERYTYPE=JOIN

Optional arguments (continued)

- **LAYER_GTYPE**
  - **Set to POINT if querying POINT only columns (for optimal performance)**
  - **Otherwise, NOTPOINT (default)**

SDO_RELATE - A window query

- **Find all counties in the state of New Hampshire**

```
select c.county, c.state_abrv
  from counties c,
       states s
 where s.state = 'New Hampshire'
   and mdsys.sdo_relate (c.geom, s.geom,
    'mask=INSIDE+COVEREDBY querytype=WINDOW') = 'TRUE';
```

SDO_RELATE - Another window query

- Find all counties around county Passaic

```
select c1.county, c1.state_abrv
  from counties c1,
       counties c2
 where c2.state = 'New Jersey'
   and c2.county = 'Passaic'
   and mdsys.sdo_relate (c1.geom, c2.geom,
        'mask=TOUCH querytype=WINDOW') = 'TRUE';
```

SDO_RELATE - A window query

- Find all cities in a selected rectangular area

```
select c.city, c.pop90
  from cities c
 where mdsys.sdo_relate (
        c.location,
        mdsys.sdo_geometry (2003, null, null,
          mdsys.sdo_elem_info_array (1,1003,3),
          mdsys.sdo_ordinate_array (-109,37,-102,40)),
        'mask=ANYINTERACT querytype=WINDOW layer_gtype=POINT')='TRUE';
```

Note: for point in polygon queries use the ANYINTERACT mask if you don’t mind returning points which fall on the boundary; ANYINTERACT makes use of center tile optimization.

SDO_RELATE and PL/SQL

- **Find total population in a selected rectangular area**

```
set serveroutput on;
declare
 rectangle mdsys.sdo_geometry;
 total_population number;
begin
 rectangle := mdsys.sdo_geometry (2003, null, null,
      mdsys.sdo_elem_info_array (1,1003,3),
      mdsys.sdo_ordinate_array (-109, 37, -102, 40));
 select sum(c.totpop) into total_population
   from counties c
  where mdsys.sdo_relate (c.geom, rectangle,
         'mask=ANYINTERACT querytype=WINDOW') = 'TRUE';
dbms_output.put_line('Population = '||total_population||'.');
end;
```

SDO_RELATE - window query

- **Find all interstates that interact with a county**

```
select i.highway
  from interstates i, counties c
 where c.state = 'New Jersey' and c.county = 'Passaic'
   and mdsys.sdo_relate (i.geom, c.geom,
       'mask=ANYINTERACT querytype=WINDOW') = 'TRUE';
```

- **Find all interstates that interact with selected counties**

```
select /*+ ordered */ i.highway
  from counties c, interstates i
 where c.state = 'Arizona' and poppsqmi < 10
   and mdsys.sdo_relate (i.geom, c.geom,
       'mask=ANYINTERACT querytype=WINDOW') = 'TRUE';
```

SDO_RELATE - join query

- Select all the city,county pairs

```
select city, county
  from counties c, cities i
where mdsys.sdo_relate (i.location, c.geom,
                        'mask=ANYINTERACT
                         querytype=JOIN
                         LAYER_GTYPE=POINT') = 'TRUE';
```

****NOTE** In 8.1.6 analyzing and computing statistics on both the index tables help the execution plan of a join query.**

Do not analyze the index tables in 8.1.5.

The SDO_WITHIN_DISTANCE operator

```
boolean := MDSYS.SDO_WITHIN_DISTANCE
 ( <geometry-1>,
   <geometry-2>,
   ‘DISTANCE=<n>,
    QUERYTYPE=<querytype>
    [other optional parameters]’
 )
```

- Performs an exact or approximate query
- Euclidean distance only
- Returns TRUE or FALSE

Arguments

- GEOMETRY-1
  - Must be a column in a table
  - Must be of type SDO_GEOMETRY
  - Must be indexed
- GEOMETRY-2
  - Variable or column in a table
  - Must be of type SDO_GEOMETRY
  - Will be buffered by distance
- DISTANCE (required)
  - The distance (expressed in the units used for the coordinate system)

Optional arguments (continued)

- LAYER_GTYPE (optional)
  - Set to POINT if querying POINT only columns (for optimal performance)
  - Otherwise, NOTPOINT (default)
- QUERYTYPE (optional)
  - If FILTER, primary filter query only (approximate results)

    NOTE Does not work in 8.1.5, fixed in 8.1.6

SDO_WITHIN_DISTANCE Examples

- **Find all cities within a distance from an interstate**

```
select c.city
  from cities c, interstates i
 where highway = 'I 170'
   and mdsys.sdo_within_distance (
        c.location, i.geom,
       'distance=0.5 layer_gtype=POINT') = 'TRUE';
```

- **Find intersates within a distance from a city**

```
select i.highway
  from interstates i, cities c
 where city = 'Tampa'
   and mdsys.sdo_within_distance (
        i.geom, c.location,'distance=0.5') = 'TRUE';
```

LOCATOR_WITHIN_DISTANCE Example

- **Find all cities within 50 miles of Chicago**

```
select c1.city
  from cities c1, cities c2
 where c2.city = 'Chicago'
   and mdsys.locator_within_distance (
        c1.location, c2.location,
       'distance=50 units=MILE') = 'TRUE';
```

- **LOCATOR_WITHIN_DISTANCE is part of intermedia (which is free)**
- **Can execute radius queries against point only layers (don't need to specify LAYER_GTYPE = POINT, automatically implied).**
- **Has a UNITS parameter (MILE, FT, METER), very accurate in 8.1.6**
- **In 8.1.6, data must be in long/lat WGS84**

The SDO_GEOM.RELATE Function

```
boolean := MDSYS.SDO_GEOM.RELATE
 ( <geometry-1>, <diminfo-1>,
   '<mask>',
   <geometry-2>, <diminfo-2> )
or
boolean := MDSYS.SDO_GEOM.RELATE
 ( <geometry-1>,'<mask>',<geometry-2>,<tolerance> )
```

- Performs an exact query (secondary filter)
- Returns TRUE or FALSE for an ANYINTERACT mask
- Returns the matching relationship if any other mask or FALSE
- Can be used in the select list

SDO_GEOM.RELATE Parameters

- GEOMETRY1
- DIMINFO1
  - From user_sdo_geom_metadata table
- MASK
  - Mask for operator
- GEOMETRY2
- DIMINFO2
  - From user_sdo_geom_metadata table
- TOLERANCE
  - The tolerance value to be used

SDO_GEOM.RELATE Function

- **Determine relationship of counties and states**

```
select c.county, mdsys.sdo_geom.relate
   (s.geom,
'determine',
    c.geom, 0.00000005)
   from states s, counties c
 where s.state = 'New Jersey'
    and s.state = c.state;

COUNTY                          RELATIONSHIP
------------------------------- --------------
Atlantic                        COVERS
Cape May                        COVERS
Cumberland                      COVERS
Essex                           CONTAINS
```

Note: this is simplified syntax

SDO_GEOM.RELATE Function

- **Determine relationship of states and counties (reverse relationship of previous slide)**

```
select c.county, mdsys.sdo_geom.relate
   (c.geom,
    'determine',
    s.geom, 0.00000005)
   from states s, counties c
 where s.state = 'New Jersey'
    and s.state = c.state;

COUNTY                          RELATIONSHIP
------------------------------- --------------
Atlantic                        COVEREDBY
Cape May                        COVEREDBY
Cumberland                      COVEREDBY
Essex                           INSIDE
```

SDO_GEOM.RELATE Function

- **Find all counties around New Jersey**

```
select c.county, c.state
  from states s, counties c
 where s.state = 'New Jersey'
   and mdsys.sdo_geom.relate
     (s.geom,
      'touch',
      c.geom, 0.00000005) = 'TOUCH';
```

- ***The function does not take advantage of spatial indexes !***

The SDO_GEOM.WITHIN_DISTANCE Function

```
boolean := MDSYS.SDO_GEOM.WITHIN_DISTANCE
  (<geometry-1>, <diminfo-1>,
   <distance>,
   <geometry-2>, <diminfo-2> )
or
boolean := MDSYS.SDO_GEOM.WITHIN_DISTANCE
  (<geometry-1>, <distance>, <geometry-2>,
<tolerance> )
```

- **Performs an exact query**
- **Euclidean distance only**
- **Returns TRUE or FALSE**
- **Does NOT use the spatial indices**

SDO_GEOM.WITHIN_DISTANCE Parameters

- **GEOMETRY1**
- **DIMINFO1**
  - **From user_sdo_geom_metadata table**
- **DISTANCE**
  - **The distance (expressed in the units used for the coordinate system)**
- **GEOMETRY2**
- **DIMINFO2**
  - **From user_sdo_geom_metadata table**
- **TOLERANCE**
  - **The tolerance value to be used**

SDO_GEOM.WITHIN_DISTANCE Function

- **Find all cities within a distance from an interstate**

```
select c.city
  from cities c, interstates i
 where highway = 'I 170'
   and mdsys.sdo_geom.within_distance (
        c.location,
        0.5,
        i.geom, 0.00000005 ) = 'TRUE';
```

- ***The function does not take advantage of spatial indices !***

SDO_GEOM.WITHIN_DISTANCE Function

- **Find all interstates within a distance from a city**

```
select i.highway
  from interstates i, cities c
 where city = 'Tampa'
   and mdsys.sdo_geom.within_distance (
        i.geom,
        (select diminfo
           from user_sdo_geom_metadata
          where table_name = 'INTERSTATES'
            and column_name = 'GEOM'),
        0.5,
        c.location,
        (select diminfo
          from user_sdo_geom_metadata
         where table_name = 'CITIES'
           and column_name = 'LOCATION')      ) = 'TRUE';
```

Syntax for flattening the varrays

```
SELECT *
FROM TABLE (SELECT a.geom.sdo_ordinates
            FROM states a
            WHERE state = 'California');
```

```
SELECT *
FROM TABLE (SELECT a.geom.sdo_elem_info
            FROM states a
            WHERE state = 'California');
```

- **Can only flatten one varray at a time**

Summary

In this lesson, your should have learned how to:

- Describe a spatial query and a spatial join
- Explain the query model
- Describe and compare spatial operators and functions
- Use the spatial operators and functions to perform spatial queries
- Understand the topological relationships used by the spatial operators

Practice 7-1 Overview

This practice covers the following topics:

- Perform various queries on the layers previously loaded and indexed
  - STATES
  - COUNTIES
  - INTERSTATES
  - CITIES