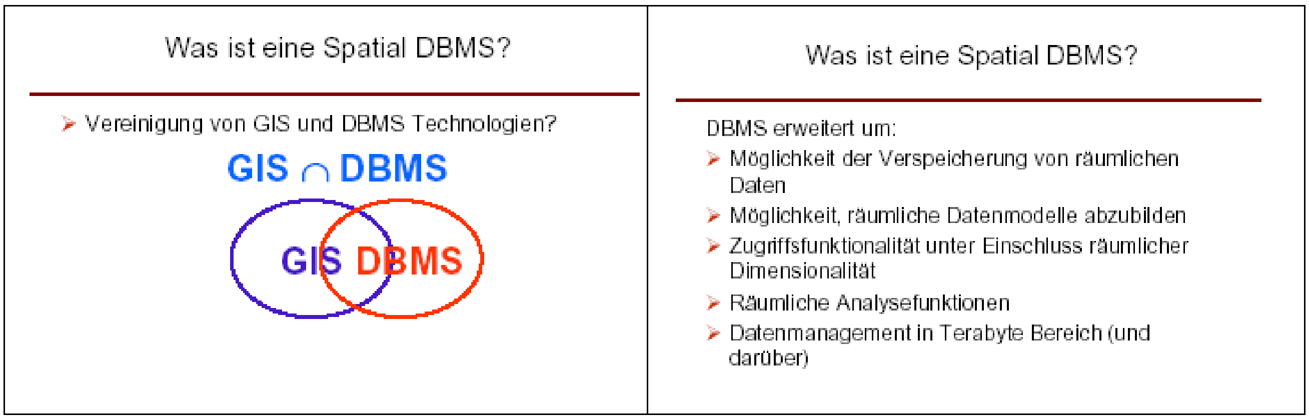


Was ist eine Spatial DBMS?

- Vereinigung von GIS und DBMS Technologien?

GIS ∩ DBMS

Was ist eine Spatial DBMS?

DBMS erweitert um:

- Möglichkeit der Verspeicherung von räumlichen Daten
- Möglichkeit, räumliche Datenmodelle abzubilden
- Zugriffsfunktionalität unter Einschluss räumlicher Dimensionalität
- Räumliche Analysefunktionen
- Datenmanagement in Terabyte Bereich (und darüber)

Abbildung 1: Was ist eine Spatial DBMS?

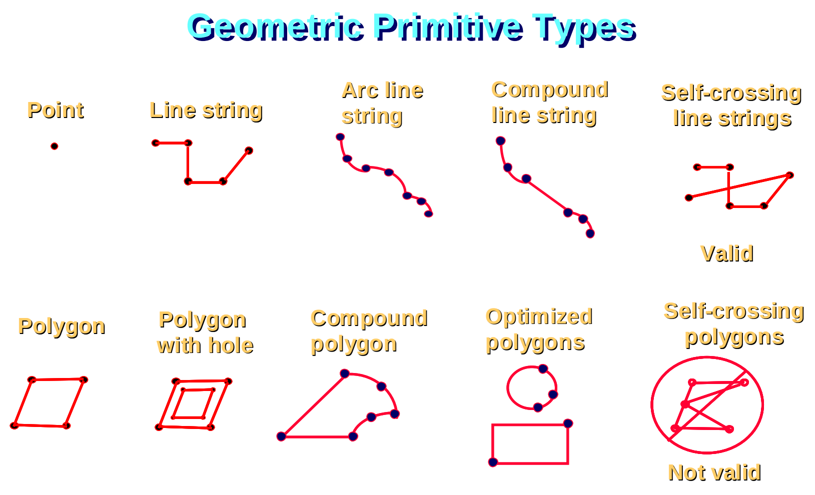


Abbildung 2: Geometrien

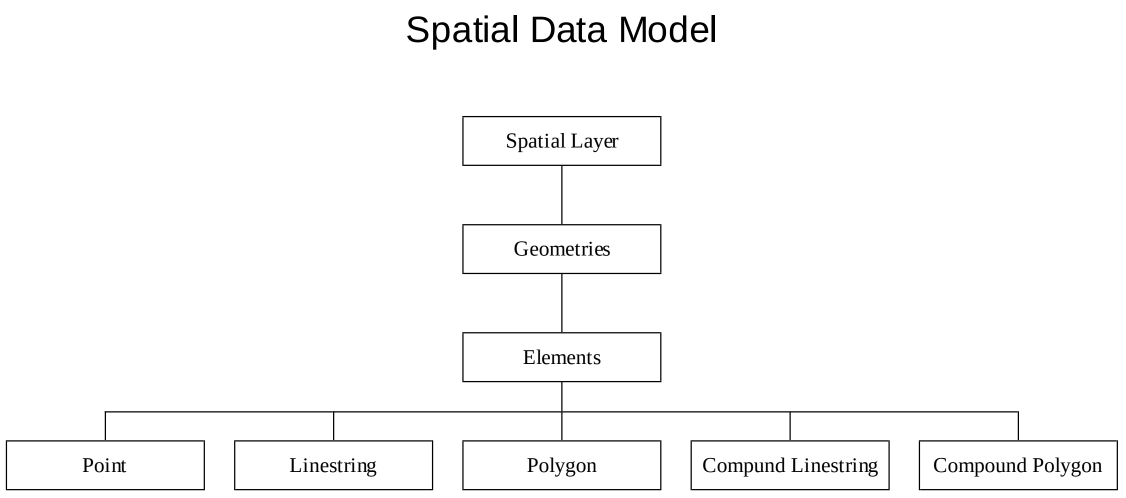


Abbildung 3: Spatial Data Model

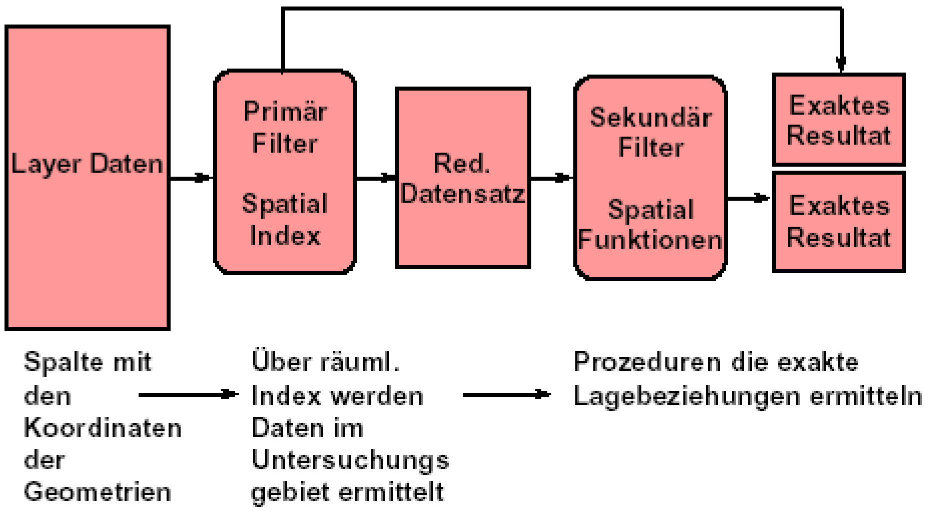

Layer Daten
Primär Filter
Spatial Index
Red. Datensatz
Sekundär Filter
Spatial Funktionen
Exaktes Resultat
Exaktes Resultat
Spalte mit den Koordinaten der Geometrien
Über räuml. Index werden Daten im Untersuchungs gebiet ermittelt
Prozeduren die exakte Lagebeziehungen ermitteln

Abbildung 4: Filter und Refinement (Optimiertes Abfrage Modell)

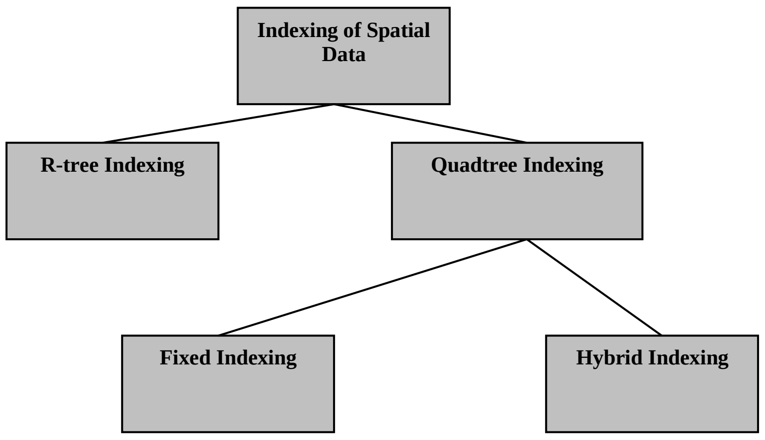
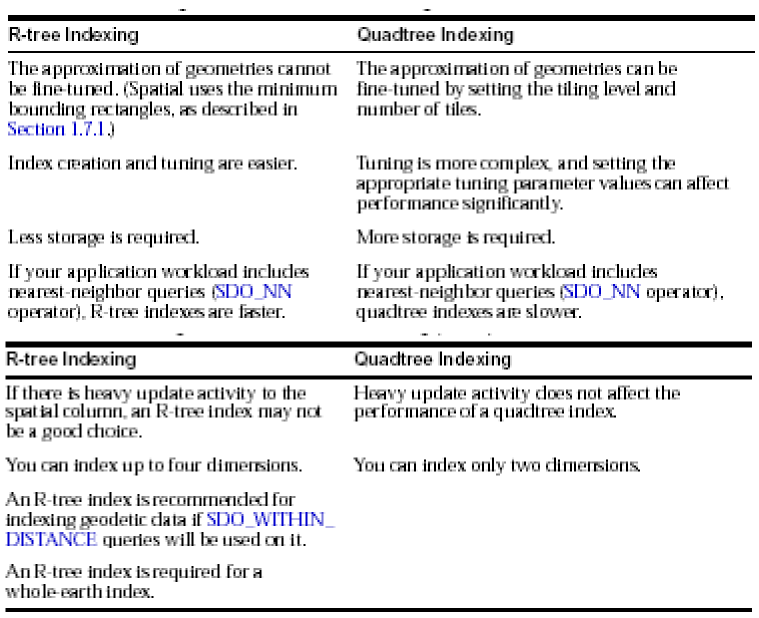


Abbildung 5: Indexing of Spatial Data



R-tree Indexing	Quadtree Indexing
The approximation of geometries cannot be fine-tuned. (Spatial uses the minimum bounding rectangles, as described in Section 1.7.1.)	The approximation of geometries can be fine-tuned by setting the tiling level and number of tiles.
Index creation and tuning are easier.	Tuning is more complex, and setting the appropriate tuning parameter values can affect performance significantly.
Less storage is required.	More storage is required.
If your application workload includes nearest-neighbor queries (SDO_NN operator), R-tree indexes are faster.	If your application workload includes nearest-neighbor queries (SDO_NN operator), quadtree indexes are slower.
If there is heavy update activity to the spatial column, an R-tree index may not be a good choice.	Heavy update activity does not affect the performance of a quadtree index.
You can index up to four dimensions.	You can index only two dimensions.
An R-tree index is recommended for indexing geodetic data if SDO_WITHIN_DISTANCE queries will be used on it.	
An R-tree index is required for a whole-earth index.	

Abbildung 6: Choosing R-tree or Quadtree Indexing

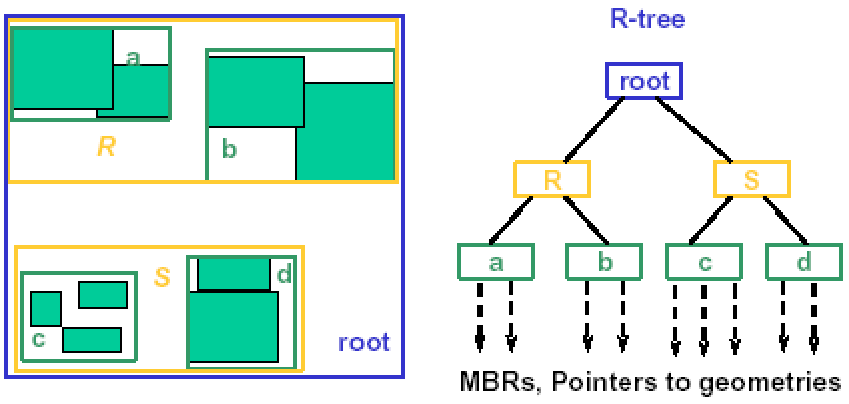


Abbildung 7: Rtree Index

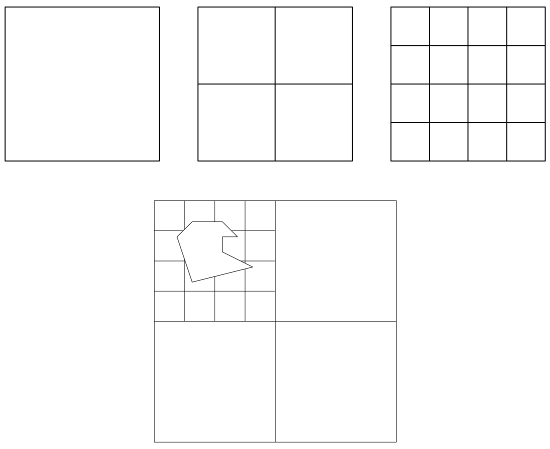
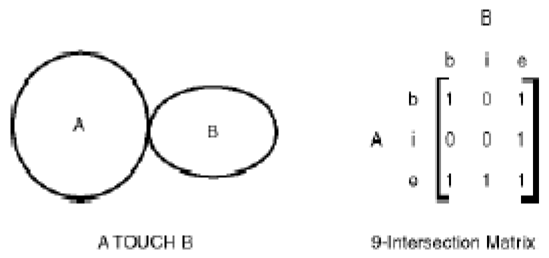
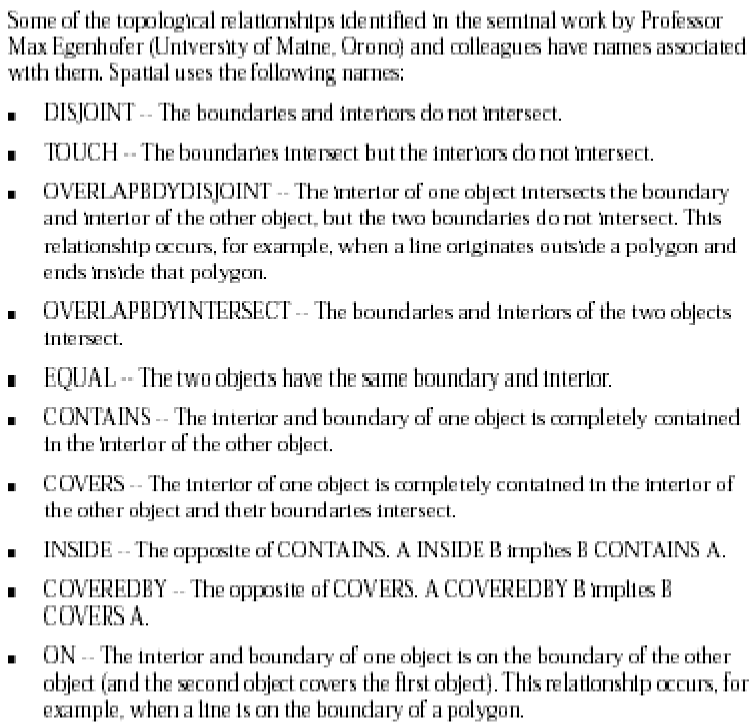


Abbildung 8: Quad Index




Some of the topological relationships identified in the seminal work by Professor Max Egenhofer (University of Maine, Orono) and colleagues have names associated with them. 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 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.

Abbildung 9: Spartial Relations

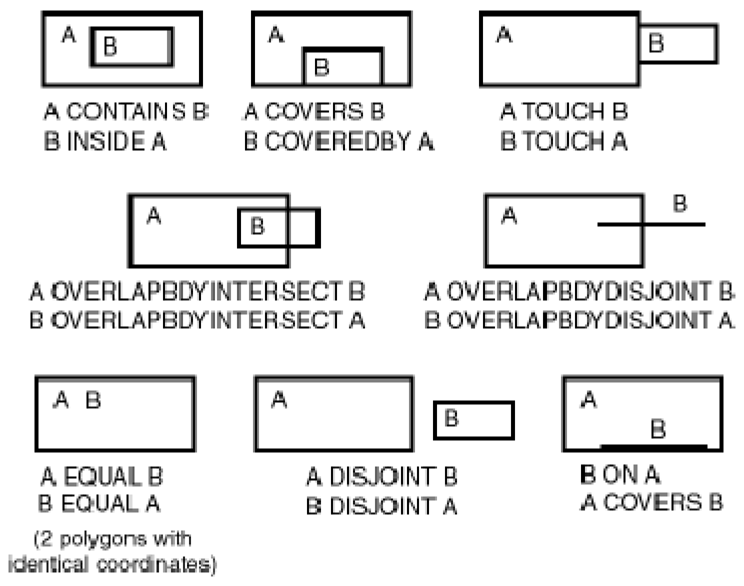


Abbildung 10: Spartial Relations - Examples

```
CREATE TYPE sdo_geometry AS OBJECT (
 SDO_GTYPE NUMBER,
 SDO_SRID NUMBER,
 SDO_POINT SDO_POINT_TYPE,
 SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
 SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);
```

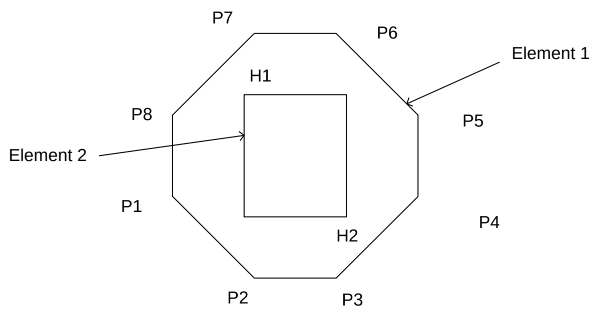
Abbildung 11: SDO_GEOMETRY Objekt

SDO_E TYPE	SDO_ INTER PRE TATION	Meaning
0	0	Unsupported element type. Ignored by the Spatial functions and procedures.
1	1	Point type.
1	N > 1	Point cluster with N points.
2	1	Line string whose vertices are connected by straight line segments.
2	2	Line string made up of a connected sequence of circular arcs. Each circular arc is described using three coordinates: the arc's starting point, any point on the arc, and the arc's end point. The coordinates for a point designating the end of one arc and the start of the next arc are not repeated. For example, five coordinates are used to describe a line string made up of two connected circular arcs. Points 1, 2, and 3 define the first arc, and points 3, 4, and 5 define the second arc, where point 3 is only stored once.
3	1	Simple polygon whose vertices are connected by straight line segments.
3	2	Polygon made up of a connected sequence of circular arcs that closes on itself. The end point of the last arc is the same as the start point of the first arc. Each circular arc is described using three coordinates: the arc's start point, any point on the arc, and the arc's end point. The coordinates for a point designating the end of one arc and the start of the next arc are not repeated. For example, five coordinates are used to describe a polygon made up of two connected circular arcs. Points 1, 2, and 3 define the first arc, and points 3, 4, and 5 define the second arc. The coordinates for points 1 and 5 must be the same, and point 3 is not repeated.
	3	Rectangle type. A bounding rectangle such that only two points, the lower-left and the upper-right, are required to describe it.
3	4	Circle type. Described by three points, all on the circumference of the circle.
4	N > 1	Line string with some vertices connected by straight line segments and some by circular arcs. The value, N, in the Interpretation column specifies the number of contiguous subelements that make up the line string. The next N triplets in the SDO_ELEM_INFO array describe each of these subelements. The subelements can only be of SDO_ETYPE 2. The last point of a subelement is the first point of the next subelement, and must not be repeated.
5	N > 1	Simple polygon with some vertices connected by straight line segments and some by circular arcs. The value, N, in the Interpretation column specifies the number of contiguous subelements that make up the polygon. The next N triplets in the SDO_ELEM_INFO array describe each of these subelements. The subelements can only be of SDO_ETYPE 2. The end point of a subelement is the start point of the next subelement and must not be repeated. The start and end points of the polygon must be the same.

Abbildung 12:SDO_INTERPRETATION

```
SQL> INSERT INTO TELEPHONE_POLES
  2>    VALUES (attribute_1, …. attribute_n,
  3>      MDSYS.SDO_GEOMETRY (
  4>        3001, null,
  5>        MDSYS.SDO_POINT_TYPE (-75.2,43.7,200),
  6>        null, null));
```

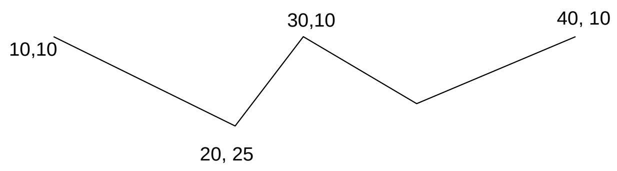
Abbildung 13: Anlegen eines Punktes im Raum



```
Element 1= [P1(6,15), P2(10,10), P3(20,10), P4(25,15), P5(25,35), P6(19,40),
P7(11,40), P8(6,25), P1(6,15)]
Element 2= [H1(12,15),  H2(15,24)]

INSERT INTO tabelle VALUES
  ('OBJ_1', MDSYS.SDO_GEOMETRY(3, NULL,NULL,
            MDSYS.SDO_ELEM_INFO_ARRAY(1,3,1, 19,3,3),
            MDSYS.SDO_ORDINATE_ARRAY(6,15, 10,10, 20,10, 25,15, 25,35,
                          19,40, 11,40, 6,25, 6,15, 12,15, 15,24)));
```

Abbildung 14: Polygon



```
SQL> INSERT INTO LINES VALUES (
  2>     attribute_1, …. attribute_n,
  3>     MDSYS.SDO_GEOMETRY (
  4>       2002, null, null,
  5>       MDSYS.SDO_ELEM_INFO_ARRAY (1,2,1),
  6>       MDSYS.SDO_ORDINATE_ARRAY (
  7>         10,10, 20,25, 30,10, 40,10))
);
```

Abbildung 15: Linienzug

```
SQL> INSERT INTO USER_SDO_GEOM_METADATA
  2>  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  3>  VALUES (
  4>   'ROADS',
  5>   'GEOMETRY',
  6>    MDSYS.SDO_DIM_ARRAY (
  7>       MDSYS.SDO_DIM_ELEMENT('Long', -180, 180, .005),
  8>       MDSYS.SDO_DIM_ELEMENT('Lat',   -90,  90, .005)),
  9>    8307);
```

Abbildung 16: Geometrie

```
CREATE INDEX ROADS_FIXED ON ROADS(GEOMETRY)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS('SDO_LEVEL = 6, SDO_NUMTILES=12');
```

Abbildung 17: Index

Finde alle Städte, die im Rechteck, das durch die Koordinaten (10,10) und (90,30) definiert ist, liegen:

```
select stadtbezeichnung
   from stadt s
   where mdsys.sdo_filter (
      s.location,                                     -- Geometry1
      mdsys.sdo_geometry (2003, null, null,           -- Geometry2
         mdsys.sdo_elem_info_array (1, 3, 3),
         mdsys.sdo_ordinate_array (10, 10, 90, 30)),
      'querytype=WINDOW layer_gtype=POINT') = 'TRUE';
```

Abbildung 18: SDO_FILTER

Gesucht sind alle Städte, die innerhalb des Bereichs der durch die Punkte (10, 10) und (110, 65) aufgespannt wird.

```
select s.stadtbezeichnung
   from stadt s
   where mdsys.sdo_relate (
      s.location,                                     -- Geometry1
      mdsys.sdo_geometry (3, null, null,              -- Geometry2
         mdsys.sdo_elem_info_array (1, 3, 3),
         mdsys.sdo_ordinate_array (10, 10, 110,65)),
      'mask=ANYINTERACT querytype=WINDOW layer_gtype=POINT') = 'TRUE';
```

Abbildung 19: SDO_RELATE

Gesucht ist die Gesamtzahl der Bevölkerung aller Städte, die im Rechteck, das durch die Punkte (10,10) und (100,65) aufgespannt wird.

```
set serveroutput on;
declare
 rectangle mdsys.sdo_geometry;
 total_population number;
begin
 rectangle := mdsys.sdo_geometry (3, null, null,
      mdsys.sdo_elem_info_array (1,3,3),
      mdsys.sdo_ordinate_array (10,10, 100,65));
 select sum(stadtbevoelkerung) into total_population
   from stadt
  where mdsys.sdo_relate (location, rectangle,
         'mask=ANYINTERACT querytype=WINDOW') = 'TRUE';
dbms_output.put_line('Population = '||total_population||'.');
end;
/
```

Abbildung 20: Bevölkerung in Gebiet

```
select sum(s.stadtbevoelkerung)
  from bezirk b, stadt s
 where b.bezirksbezeichnung = 'B2'
 and mdsys.SDO_CONTAINS(b.ausmass, s.location) = 'TRUE';
```

Abbildung 21: SDO_GEOM.RELATE

Gesucht sind alle Städte, die in einem Umkreis von 30 Einheiten um die Stadt S4 existieren:

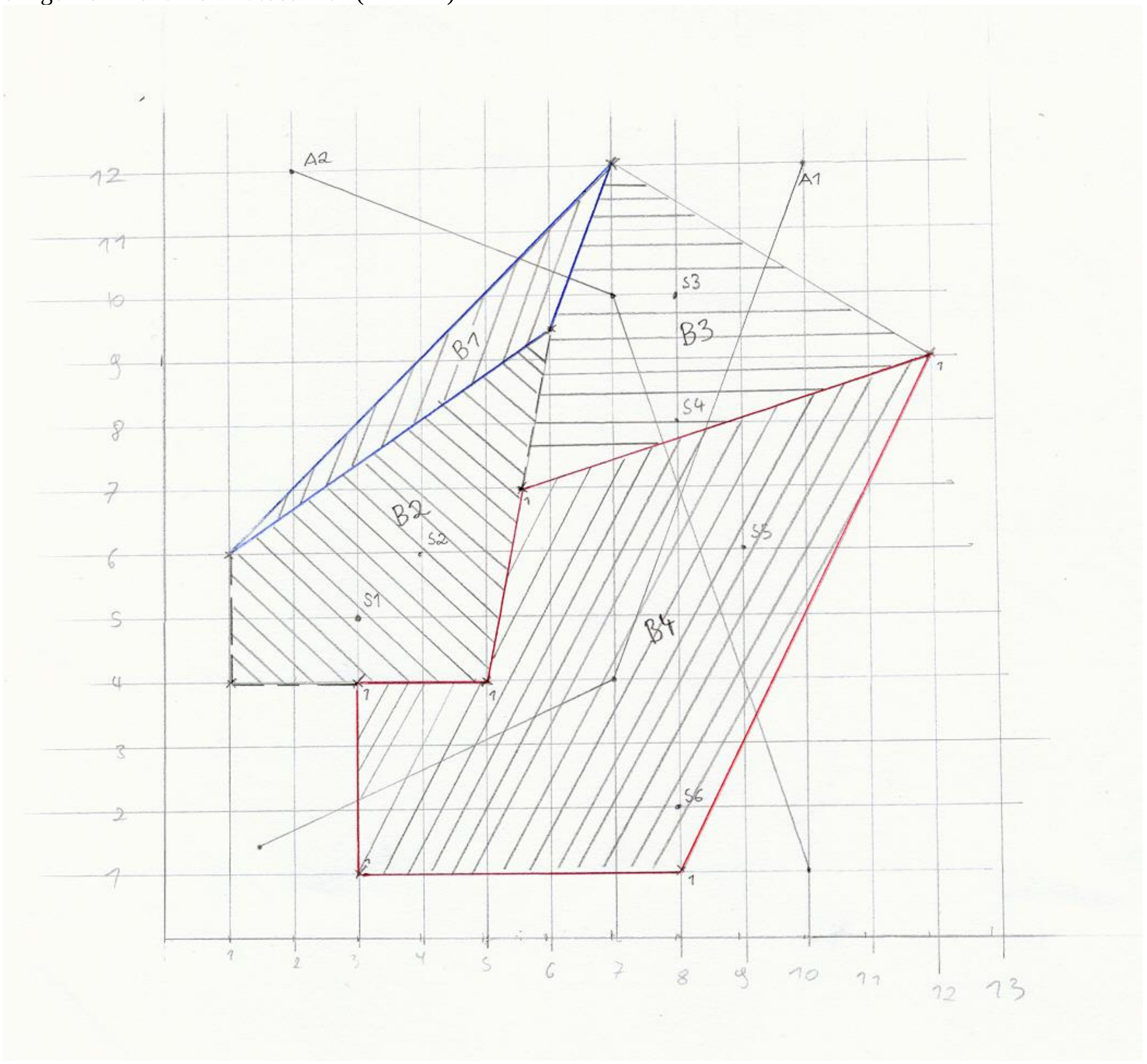
```
select s2.stadtbezeichnung
  from stadt s1, stadt s2
  where s1.stadtbezeichnung = 'S4'
  and mdsys.locator_within_distance (
         s1.location, s2.location,
          'distance = 30') = 'TRUE';
```

Abbildung 22: SDO_WITHIN_DISTANCE

Gesucht sind die 2 nächsten Städte zur Autobahn A1:
```
select /*+ ordered */
  s.stadtbezeichnung
  from autobahn a, stadt s
 where a.autobahnbezeichnung = 'A1'
   and mdsys.sdo_nn (
        s.location,                 -- Geometrie 1
        a.bahn,                     -- Geometrie 2
       'sdo_num_res=2') = 'TRUE';
```
Abbildung 23: SDO_NN

Gesucht ist die Länge der Autobahn A1.:
```
select a.autobahnbezeichnung, sdo_geom.sdo_length (a.bahn,m.diminfo)
  from autobahn a, user_sdo_geom_metadata m
  where m.table_name = 'AUTOBAHN' and m.column_name='BAHN' and a.autobahnbezeichnung='A1';
```
Abbildung 24: SDO_GEOM.LENGTH

Gegeben sei eine Landkarte mit Bezirken (B1 - B4). In den Bezirken befinden sich Städte (S1 - S6). Durch einige Bezirke führen Autobahnen (A1 - A2)



```
drop table stadt;
create table stadt (
  location    mdsys.sdo_geometry,
  stadtbezeichnung        varchar2(40),
  stadtbevoelkerung       number);

insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(30, 50, NULL),
NULL, NULL), 'S1', 100000);
insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(40, 60, NULL),
NULL, NULL), 'S2', 50000);
insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(80, 100, NULL),
NULL, NULL), 'S3', 20000);
insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(80, 80, NULL),
NULL, NULL), 'S4', 200000);
insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(90, 60, NULL),
NULL, NULL), 'S5', 1000);
insert into stadt values (mdsys.sdo_geometry(1, NULL, MDSYS.SDO_POINT_TYPE(80,20, NULL),
NULL, NULL), 'S6', 1000);

drop table autobahn;
create table autobahn (
  bahn            mdsys.sdo_geometry,
  autobahnbezeichnung varchar2(20));

insert into autobahn values(mdsys.sdo_geometry(2002, null, null, mdsys.sdo_elem_info_array(1,2,1),
mdsys.sdo_ordinate_array (15,15, 70,40, 100,120)), 'A1');
insert into autobahn values(mdsys.sdo_geometry(2002, null, null, mdsys.sdo_elem_info_array(1,2,1),
mdsys.sdo_ordinate_array (20,120, 70,100, 100,10)), 'A2');

drop table bezirk;
create table bezirk (
  ausmass  mdsys.sdo_geometry,
  bezirksbezeichnung    varchar2(20),
  bundesland     varchar2(20),
  bezirksbevoelkerung   number);

insert into bezirk values(mdsys.sdo_geometry(3, NULL, NULL, mdsys.sdo_elem_info_array(1,3,1),
mdsys.sdo_ordinate_array(10,60, 70,120, 60,95,10,60)), 'B1', 'OOE', 500000);
insert into bezirk values(mdsys.sdo_geometry(3, NULL, NULL, mdsys.sdo_elem_info_array(1,3,1),
mdsys.sdo_ordinate_array(10,40,50,40, 60,95, 10,60,10,40)), 'B2', 'NOE', 700000);
insert into bezirk values(mdsys.sdo_geometry(3, NULL, NULL, mdsys.sdo_elem_info_array(1,3,1),
mdsys.sdo_ordinate_array(55,70, 60,95,70,120,120,90, 55,70)), 'B3', 'OOE', 400000);
insert into bezirk values(mdsys.sdo_geometry(3, NULL, NULL, mdsys.sdo_elem_info_array(1,3,1),
mdsys.sdo_ordinate_array(30,10,30,40,50,40,55,70,120,90,80,10,30,10)), 'B4', 'NOE', 1000000);

delete from USER_SDO_GEOM_METADATA where table_name='STADT';
INSERT INTO USER_SDO_GEOM_METADATA
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES (
   'STADT',
   'LOCATION',
    MDSYS.SDO_DIM_ARRAY (
       MDSYS.SDO_DIM_ELEMENT('Long', 0,120,0.005),
       MDSYS.SDO_DIM_ELEMENT('Lat',  0,120,0.005)),
    NULL);
```

```
delete from USER_SDO_GEOM_METADATA where table_name='BEZIRK';
INSERT INTO USER_SDO_GEOM_METADATA
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES (
   'BEZIRK',
   'AUSMASS',
    MDSYS.SDO_DIM_ARRAY (
       MDSYS.SDO_DIM_ELEMENT('Long', 0,120,0.005),
       MDSYS.SDO_DIM_ELEMENT('Lat',  0,120,0.005)),
    NULL);

delete from USER_SDO_GEOM_METADATA where table_name='AUTOBAHN';
INSERT INTO USER_SDO_GEOM_METADATA
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES (
   'AUTOBAHN',
   'BAHN',
    MDSYS.SDO_DIM_ARRAY (
       MDSYS.SDO_DIM_ELEMENT('Long', 0, 20,0.005),
       MDSYS.SDO_DIM_ELEMENT('Lat',  0, 20,0.005)),
    NULL);

drop index stadt_fixed;
CREATE INDEX STADT_FIXED ON STADT(LOCATION)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX;

drop index BEZIRK_FIXED;
CREATE INDEX BEZIRK_FIXED ON BEZIRK(AUSMASS)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX ;
```

Abbildung 25: Beispieltabellen

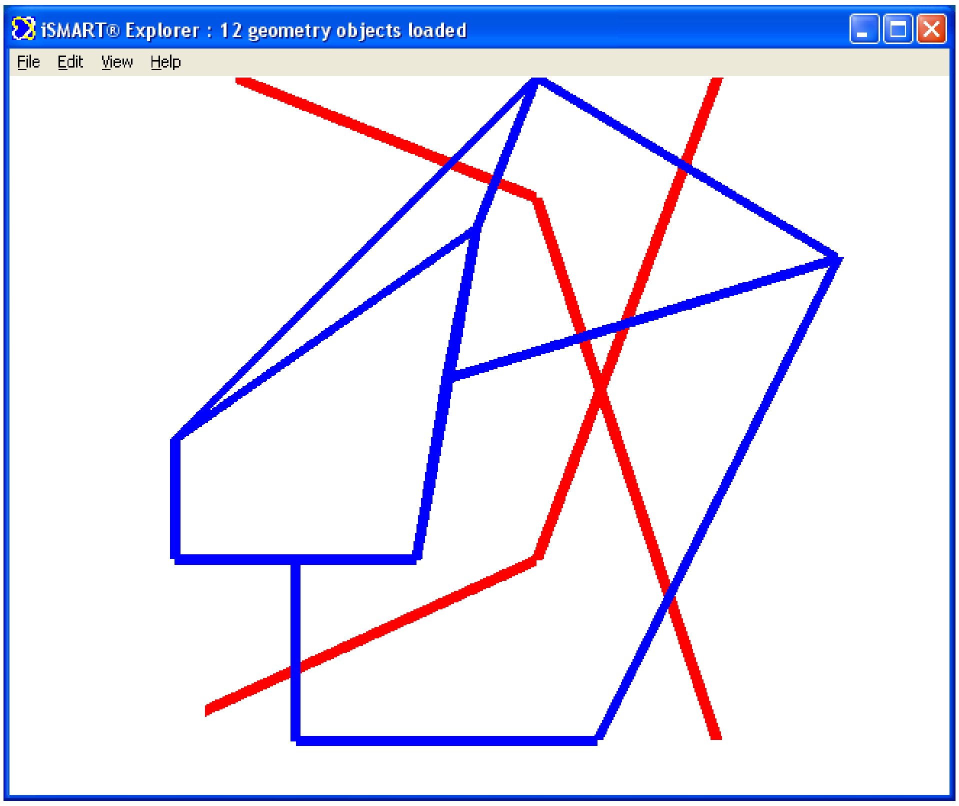


Abbildung 26: Beispieltabellen – dargestellt mit iSMARTExplorer