Spatial a gyakorlatban

A korábban összefoglalt elméleti ismeretanyagot mindenképpen szerettem volna kipróbálni a gyakorlatban is. Sajnos az időm rövidsége és beállítási problémák miatt nem tudtam megoldani, hogy egy előre elkészített adatbázist betöltsek, majd ezen végezzek vizsgálatokat. Ezt azonban a későbbiek során mindenképpen pótolni szeretném.

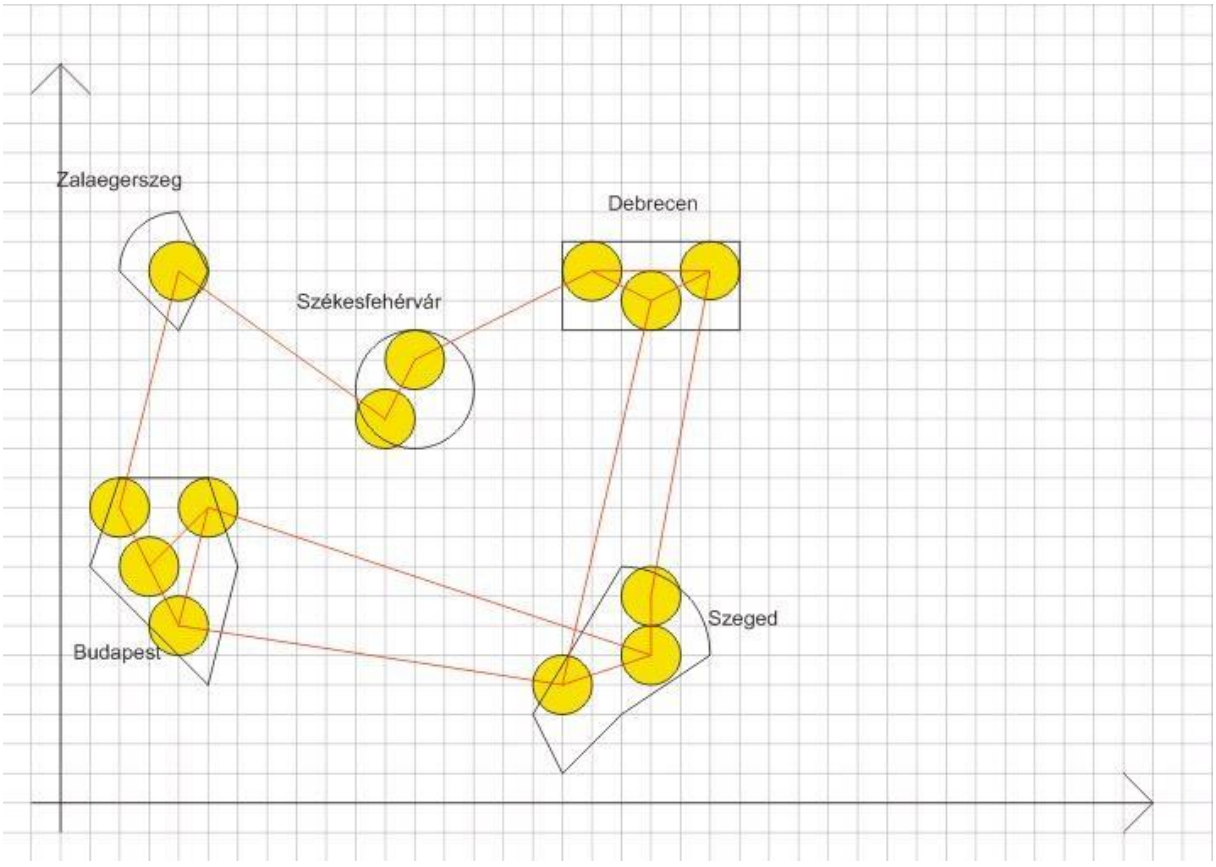
Így azonban egy általam kreált kis mintapélda volt az alapja a gyakorlati vizsgálataimnak.

Mintapélda

Egy képzeletbeli magyar távközlési szolgáltató internetelérést biztosít Magyarországon 5 szintén képzeletbeli városban, ezek Budapest, Zalaegerszeg, Székesfehérvár, Debrecen és Szeged. A szolgáltatónak van egy térképe, melyen fel vannak tüntetve a városok, a városokon belüli csomópontok, és a csomópontok közötti fizikai összeköttetések. Azt tudjuk még, hogy egy csomópont 1 km-es körzetében tud a szolgáltató szolgáltatni.

A szolgáltató ezeket az adatokat egy adatbázisban tárolja, és különböző összefüggések érdeklik, például milyen messze vannak egymástól a városok, vagy mekkora egy város lefedettsége.

A szolgáltató térképe így néz ki:



A megoldáshoz egy sql szkriptet írtam mely alant olvasható (a sorok feltöltésénél nem írtam ki mindent, hogy egy picit rövidebb legyen):

```
-- Halozat.sql

-- Táblák eldobása

DROP TABLE cities;

DROP TABLE nodes_cover;

DROP TABLE paths;

DROP TABLE nodes;

DELETE FROM user_sdo_geom_metadata;

-- Táblák létrehozása

CREATE TABLE cities (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(16),
  shape SDO_GEOMETRY);

CREATE TABLE nodes (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(5),
  shape SDO_GEOMETRY);

CREATE TABLE nodes_cover (
  id NUMBER PRIMARY KEY,
  shape SDO_GEOMETRY);

CREATE TABLE paths (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(12),
  startpoint NUMBER,
  endpoint NUMBER,
  shape SDO_GEOMETRY);

ALTER TABLE paths ADD (
  CONSTRAINT "start" FOREIGN KEY(startpoint)
      REFERENCES nodes(id),
  CONSTRAINT "end" FOREIGN KEY(endpoint)
      REFERENCES nodes(id));

ALTER TABLE nodes_cover ADD (
  CONSTRAINT "id" FOREIGN KEY(id)
        REFERENCES nodes(id));

-- Sorok létrehozása

-- Városok

INSERT into cities VALUES(1, 'Budapest', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1), SDO_ORDINATE_ARRAY(5,4, 6,8, 5,11, 2,11,
1,8, 5,4)));

INSERT into cities VALUES(2, 'Zalaegerszeg', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 7,2,2), SDO_ORDINATE_ARRAY(2,18, 4,16,
5,18, 4,20, 2.586,19.414, 2,18)));
```

```
INSERT into cities VALUES(3, 'Szekesfehervar', SDO_GEOMETRY(2003, NULL,
NULL, SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(9,14, 11,12,
13,14)));

INSERT into cities VALUES(4, 'Debrecen', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,3), SDO_ORDINATE_ARRAY(17,16, 23,19)));

INSERT into cities VALUES(5, 'Szeged', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 9,2,2), SDO_ORDINATE_ARRAY(19,8,
16,3, 17,1, 19,3, 22,5, 21.121,7.121, 19,8)));

COMMIT;

-- Csomópontok

INSERT INTO nodes VALUES(1, 'BP1', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(2,10,NULL), NULL, NULL));
INSERT INTO nodes VALUES(2, 'BP2', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(3,8,NULL), NULL, NULL));

...

INSERT INTO nodes VALUES(13, 'SZEG3', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(17,4,NULL), NULL, NULL));

COMMIT;

-- Utak

INSERT INTO paths VALUES(1, 'BP1-BP2', 1, 2, SDO_GEOMETRY(2002, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(2,10, 3,8)));
INSERT INTO paths VALUES(2, 'BP2-BP3', 2, 3, SDO_GEOMETRY(2002, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(3,8, 5,10)));

...

INSERT INTO paths VALUES(17, 'SZEG3-BP4', 13, 4, SDO_GEOMETRY(2002, NULL,
NULL, SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(17,4, 4,6)));

COMMIT;

-- Lefedettség feltöltése

INSERT INTO nodes_cover VALUES(1, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(1,10, 2,9, 3,10)));
INSERT INTO nodes_cover VALUES(2, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(2,8, 3,7, 4,8)));

...

INSERT INTO nodes_cover VALUES(13, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(16,4, 17,3, 18,4)));

COMMIT;

-- Metaadatok betöltése
INSERT INTO user_sdo_geom_metadata
    (TABLE_NAME,
     COLUMN_NAME,
     DIMINFO,
     SRID)
```

```
  VALUES (
  'cities',
  'shape',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 0, 30, 0.005),
    SDO_DIM_ELEMENT('Y', 0, 30, 0.005)
     ),
  NULL);

...

INSERT INTO user_sdo_geom_metadata
    (TABLE_NAME,
     COLUMN_NAME,
     DIMINFO,
     SRID)
  VALUES (
  'nodes_cover',
  'shape',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 0, 30, 0.005),
    SDO_DIM_ELEMENT('Y', 0, 30, 0.005)
     ),
  NULL);

COMMIT;
-- Indexek létrehozása

CREATE INDEX cities_index
   ON cities(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX nodes_index
   ON nodes(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX paths_index
   ON paths(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX nodes_cover_index
   ON nodes_cover(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

COMMIT;

-- Lekérdezések

-- Városok területei

SELECT name as city, SDO_GEOM.SDO_AREA(shape, 0.005) as area, 'km^2' as
measure
FROM cities;

-- Melyik csomópont melyik városban van?

SELECT node, city
FROM (SELECT n.name as node, c.name as city, SDO_GEOM.RELATE(c.shape,
'anyinteract', n.shape, 0.005) as inside
FROM cities c, nodes n)
WHERE inside = 'TRUE';
```

```
-- Két város közötti fizikai kapcsolatok?

SELECT c1.name as city1, c2.name as city2, p.name as path_name
FROM cities c1, cities c2, nodes n1, nodes n2, paths p
WHERE NOT(c1.id=c2.id) AND
    SDO_GEOM.RELATE(c1.shape, 'contains', n1.shape, 0.005) = 'CONTAINS' AND
    SDO_GEOM.RELATE(c2.shape, 'contains', n2.shape, 0.005) = 'CONTAINS' AND
    p.startpoint = n1.id AND
    p.endpoint = n2.id;

-- Fizikai kapcsolatok hossza?

SELECT p.name as path, SDO_GEOM.SDO_LENGTH(p.shape, m.diminfo) as length,
'km' as km
FROM paths p , user_sdo_geom_metadata m
WHERE m.table_name = 'PATHS' AND m.column_name = 'SHAPE';

-- Városok közötti távolságok?

SELECT c1.name as city1, c2.name as city2, SDO_GEOM.SDO_DISTANCE(c1.shape,
c2.shape, 0.005) as distance, 'km' as km
FROM cities c1, cities c2
WHERE c1.id < c2.id;


-- Debrecen lefedettsége?

SELECT city, SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_DIFFERENCE(shape, cover,
0.005), 0.005) / SDO_GEOM.SDO_AREA(shape, 0.005) * 100 as coverage, '%'
as meas
FROM( SELECT c.name as city,
      SDO_GEOM.SDO_UNION(SDO_GEOM.SDO_UNION(nc1.shape, nc2.shape, 0.005),
      nc3.shape, 0.005) as cover, c.shape as shape
      FROM cities c, nodes n1, nodes n2, nodes n3, nodes_cover nc1,
      nodes_cover nc2, nodes_cover nc3, user_sdo_geom_metadata m
      WHERE n1.id = nc1.id AND
            n2.id = nc2.id AND
            n3.id = nc3.id AND
            n1.id < n2.id AND
            n1.id < n3.id AND
            n2.id < n3.id AND
            SDO_GEOM.RELATE(c.shape, 'contains', n1.shape, 0.005) =
              'CONTAINS' AND
            SDO_GEOM.RELATE(c.shape, 'contains', n2.shape, 0.005) =
              'CONTAINS' AND
            SDO_GEOM.RELATE(c.shape, 'contains', n3.shape, 0.005) =
              'CONTAINS' AND
            c.name = 'Debrecen' AND
            m.table_name = 'NODES_COVER' AND
            m.column_name = 'SHAPE');
```

A kimenet pedig az alábbiak szerint alakult (a kommentek utólagosan kerültek bele a jobb átláthatóság kedvéért):

```
-- Városok területe:

CITY                    AREA MEAS
--------------- ---------- ----
Budapest                  22 km^2
Zalaegerszeg     7,14096678 km^2
Szekesfehervar   12,5663706 km^2
Debrecen                  18 km^2
Szeged           20,5671752 km^2

-- Melyik csomópont melyik városban van?

NODE  CITY
----- ---------------
BP1   Budapest
BP2   Budapest
BP3   Budapest
BP4   Budapest
ZEG1  Zalaegerszeg
SZFV1 Szekesfehervar
SZFV2 Szekesfehervar
DEB1  Debrecen
DEB2  Debrecen
DEB3  Debrecen
SZEG1 Szeged
SZEG2 Szeged
SZEG3 Szeged

-- Városok között milyen összeköttetések vannak?

CITY1            CITY2            PATH_NAME
---------------- ---------------- ------------
Budapest         Zalaegerszeg     BP1-ZEG1
Zalaegerszeg     Szekesfehervar   ZEG1-SZFV1
Szekesfehervar   Debrecen         SZFV2-DEB1
Debrecen         Szeged           DEB2-SZEG1
Debrecen         Szeged           DEB3-SZEG3
Szeged           Budapest         SZEG2-BP3
Szeged           Budapest         SZEG3-BP4

-- Összeköttetések hossza?

PATH             LENGTH KM
------------ ---------- --
BP1-BP2      2,23606798 km
BP2-BP3      2,82842712 km
BP2-BP4      2,23606798 km
BP3-BP4      4,12310563 km
BP1-ZEG1     8,24621125 km
ZEG1-SZFV1   7,81024968 km
SZFV1-SZFV2  2,23606798 km
SZFV2-DEB1   7,61577311 km
DEB1-DEB2             4 km
DEB1-DEB3    2,23606798 km
DEB2-DEB3    2,23606798 km
DEB2-SZEG1   11,1803399 km
DEB3-SZEG3   13,3416641 km
SZEG1-SZEG2           2 km
```

```
SZEG2-SZEG3  3,16227766 km
SZEG2-BP3    15,8113883 km
SZEG3-BP4    13,1529464 km

-- Városok egymástól mért legkisebb távolsága?

CITY1           CITY2              DISTANCE KM
--------------- --------------- ---------- --
Budapest        Zalaegerszeg              5 km
Budapest        Szekesfehervar   4,70820393 km
Budapest        Debrecen                 13 km
Budapest        Szeged           10,9141031 km
Zalaegerszeg    Szekesfehervar   5,15541753 km
Zalaegerszeg    Debrecen                 12 km
Zalaegerszeg    Szeged           16,9783599 km
Szekesfehervar  Debrecen         4,32455532 km
Szekesfehervar  Szeged                    8 km
Debrecen        Szeged                    8 km

-- Debrecen lefedettsége?

CITY              COVERAGE M
--------------- ---------- -
Debrecen        47,6401224 %
```

Tanulságok

Még egy ilyen egyszerű kis példában is, mint a fent vázolt, sok tapasztalatot lehet szerezni.

Jelentősen megkönnyíti például a lekérdezések összeállítását a Spatial függvények egymásba ágyazhatósága, mivel majdnem minden ilyen függvény visszatérése egy SDO_GEOMETRY elem. Ezáltal sokkal tömörebben és átláthatóbban lehet leírni egy lekérdezést, ahogy az az utolsó példánál is látható.

A lekérdezések összeállításánál vált érthetővé számomra a metaadatok jelentősége is. Több függvénynek két alakja is van, az egyik esetben elegendő egy toleranciaértéket megadni a számítás elvégzésére, ez látható például az utolsó lekérdezésnél. Azonban lehetőségünk van a függvény paramétereihez egy metaadatsort is hozzárendelni, így különböző koordinátarendszerekben definiált adatok esetén is el tudjuk végezni a műveletet anélkül, hogy nekünk kellene a konverziót elvégezni.