Topologikus térbeli adatstruktúrák

Elek István

Topologikus térbeli adatstruktúrák

A könyv megjelenését a Magyar Tudományos Akadémia támogatta.

Lektorálta: dr. Nikovits Tibor (ELTE, Információs Rendszerek Tanszék)

ISBN 978-963-279-862-2

Témakör: *térinformatika, adatbázis-technológia*

Kiadja a Typotex Elektronikus Kiadó Kft.
Felelős vezető: Votisky Zsuzsa
Főszerkesztő: Horváth Balázs
Műszaki szerkesztő: Csépány Gergely
A szöveget gondozta: Czene István
Borítóterv: Szalay Éva
Készült a Kódex Könyvgyártó Kft. nyomdájában
Felelős vezető: Marosi Attila

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

Bevezetés

A topológia jól ismert fogalom a térinformatikával foglalkozók körében. Minden piacvezető szoftver tartalmaz funkciókat a topológia kezelésére, létrehozására, javítására. A térbeli lekérdezések miatt a topologikus adatok megléte alapvető adatminőségi követelmény.

A térképi adatok létrehozásakor, importjakor kell felépülnie a megfelelő topológiának, amely eléggé gyakran nem teremthető meg automatikusan. Raszteres adatok feletti fél automatikus (kézi) vektorizáláskor a topológia biztosítása alapvetően a vektorizálást végző személy szakértelmétől függ. Továbbá az export/import műveletekből származó adatok topológiája erősen függ az eredeti adatforrás minőségétől. Amennyiben az adatforrás strukturálatlanul tartalmazza a vektoros adatokat (spagettimodell), és topológiai hibákat tartalmaz, akkor az import során keletkezett adatok sem lesznek topologikus minőségűek.

Az importált adatok topológiájának létrehozása külön beavatkozást igényel, amennyiben az eredeti adathalmaz tartalmazott topológiai hibákat. A piacvezető GIS szoftverek funkcionalitása számos eszközt biztosít erre a célra. Nemcsak a drága kereskedelmi szoftverek, hanem az ingyenesen hozzáférhető nyílt forráskódú rendszerek is fel vannak készülve a topológiai hibák javítására, a topologikus minőségű adatrendszer megteremtésére.

A könyvben bemutatjuk a különböző vektoros adatmodelleket, a spagettimodellt, és a topológia megőrzésére alkalmas más modelleket is. Különösen sokat fogunk foglalkozni a poligonok topológiájával, amely a legösszetettebb ebben a vonatkozásban. A spagettimodellben is lehetséges topologikus minőségű adatrendszerek létrehozása, de az adatbázis szerkezetileg lehetővé teszi topológiai hibák létrejöttét. A spagettimodell nemcsak ezért korszerűtlen, hanem azért is, mert redundánsan tárolja az objektumok node-jait. A redundáns tárolás további hibák forrása lehet, ezért fontos olyan adatszerkezetek kidolgozása, amely megszünteti a redundanciát, és garantálja az egyszer már létrejött topológiai megőrzését. Ilyen adatszerkezeteket is ismertetni fogunk.

Be fogjuk mutatni a piacvezető adatbázis-kezelők által a térbeliség kezelésére kialakított fogalmakat és funkcionalitásokat. Kiválasztottunk egy meghatározót a neves adatbázis-kezelők közül, amely végül a Postgres/PostGIS lett. Látni fogjuk, hogy térbeli függvénykönyvtár a spagettimodellre épül. Viszonylag részletesen, de nem minden részletre kitérve, ismertetni fogjuk az OGC elvi megalapozását, és annak egy SQL-implementációját.

Részletesen ismertetünk egy redundanciamentes, a topológiát szerkezetéből következően megőrző adatstruktúrát. Ha korszerűbb, megbízhatóbb is a topológiát megőrző adatszerkezet, a spagettire kidolgozott függvénykönyvtár nem alkalmazható rá. Ez az adatszerkezet tehát nem OGC kompatibilis, de előnyös tulajdonságai miatt a tárgyalását indokoltnak találtuk. Ha bebizonyosodik, hogy a topologikus adatszerkezet a spagettinél jobb tulajdonságokkal rendelkezik, és performancia szempontból is megfelelő, akkor implementálni kell a teljes térbeli függvényhalmazt.

A könyv utolsó fejezetében részletesebben tárgyaljuk a vonalas objektumok mentén való helymeghatározást, noha nem csak a topológia szempontjából. Vonalas objektumokra is felvázolunk egy topológia megőrző adatszerkezetet, amely szintén nem OGC kompatibilis, de a redundanciamentes tárolás miatt úgy gondoltuk, hogy érdekes lehet. A vonalas objektumok menti helymeghatározás megtárgyalása során megmutatjuk, hogy a tisztán geometriai elemeken nyugvó leírás számos esetben nem kielégítő a valóság megfelelő leírására.

Úgy gondoljuk, hogy mindkét redundanciamentes topológiamegőrző adatstruktúra érdekes lehet bizonyos állami alapadatok (pl. kataszteri térképek, topográfiai térképek) tárolása szempontjából. Igaz, ma még egyetlen GIS szoftver sem követ ehhez hasonló struktúrát, hiszen vagy az OGC szabvány szerint működik, vagy saját implementációt használ, de a redundanciamentes struktúrából bármikor exportálható spagettitopológiájú adat (pl. ESRI shape, vagy Mapinfo MIF), és így bármely szoftverben előállítható az annak megfelelő munkaformátum. Egyébként bármely standard formátumú adat importjakor át kell esnünk ezen a procedurán, függetlenül attól, hogy OGC szabványt követ vagy sem.

Mintegy mellékesen bemutatunk néhány önkényesen kiválasztott, ismert adatformátumot, amely részben munkaformátum is (WKT), részben pedig csak adatcsere céljára szolgál (MIF/MID, DAT adatcsere formátum).

A könyv készítésekor azt az elvet követtük, hogy az angol terminológiát megtartottuk, mivel számos szónak jelenleg nincs még magyar fordítása,

és ahol lehetett, a szavak, fogalmak magyar megfelelőjét zárójelben feltüntettük az angol szavak mellett. Ettől persze kicsit vegyes lett a kép, de arra semmiképpen nem akartunk vállalkozni, hogy kísérletezzünk a fogalmak magyar fordításával. Remélhetőleg idővel kialakulnak a szakma által elfogadott, meghonosodott magyar szavak is.

2014. június, Budapest Dr. Elek István

1

A spagettimodell áttekintése

A vektoros adatmodell az ábrázolni kívánt objektumok általunk jellemzőnek, fontosnak tartott pontjainak helyvektorait tárolja, ami kétdimenziós esetben x, y koordinátákat, háromdimenziós esetben x, y, z koordinátákat jelent. Azt is tárolnunk, tudnunk kell, hogy a megadott koordináták milyen koordináta-rendszerben értelmezettek. Ez a tény feltételezi, hogy az a szoftver, amely tárolja a pontok koordinátáit, kezelni tudja a koordináta-rendszereket.

Minden, a pontnál bonyolultabb objektum, a helyvektorosan tárolt pontok sokaságából épül fel valamely összekötési szabály alapján (amely legegyszerűbb esetben lehet az összetartozó pontok beviteli sorrendje). A matematikai összefoglaló gráfelméleti fejezetével összhangban, a vektoros adatmodellben az ábrázolni kívánt objektumokat gráfokkal írjuk le. A jellegzetes pontok a gráf pontjai (vertexei, node-jai, magyarul csúcsai, csomópontjai), amelyek alkotják a vonalakat és a poligonokat.

1.1. Vektoros adatok létrehozása

Kérdés, hogy ki dönti azt el, hogy melyik pont jellegzetes illetve fontos, elvégre éppen ezeket akarjuk tárolni. A kérdés megválaszolásához vizsgáljuk meg a vektoros adatbevitel módszereit. Ha űrfelvételekről vagy légifotókról vektorizálunk, előbb értelmeznünk kell, hogy mit látunk. Ha felismertük a folyókat, utakat, házakat, akkor azok jellegzetes pontjai (ház sarkai, folyók, utak jellemző pontjai, úgy mint torkolat, útkereszteződés, stb.) egyszerűen megállapíthatók. A folyamat kiemelten fontos momentuma a felismerés, az emberi szemlélő, interpretátor tudása, tapasztalata, háttér ismeretei. Enélkül a vektorizálás lehetetlen. Az automatikus vektorizálás eddig megoldatlan mivolta éppen azt bizonyítja, hogy a vektorizálás igen magas szintű intelligenciát igényel, mint amilyen az interpretátor tapasztalata, térképészeti és légi-

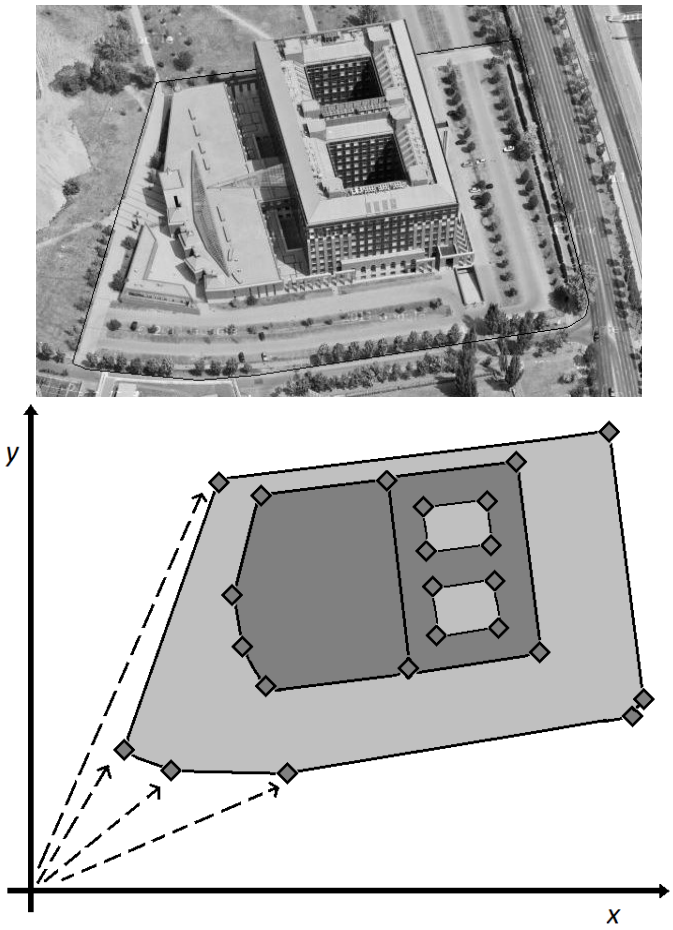
fotó értelmezési képességei. Ezek a legtöbb esetben fejlett képfeldolgozási, képfelismerési képességeket jelentenek, amelyekre mind ez idáig számítógépeket nem sikerült megtanítani. Az eddig legsikeresebb térképvektorizáló programok félautomatikus módon képesek csak működni, emberi szakértő folyamatos jelenléte mellett, aki olyan esetekben, amikor a program értelmezhetetlen grafikai szituációt észlel, beavatkozik.

Speciális esetekben persze lehetséges automatikus vektorizálás, mint például csak szintvonalakat tartalmazó raszteres térkép, de ha komplett térképek raszteres állományai képezik a vektorizálás tárgyát, a programok képtelenek kezelni a túlzottan bonyolult grafikai szituációt. Gondoljunk csak egy topográfiai térképre, amelyet színnel kitöltött felületek, eltérő vastagságú, mintázatú és színű vonalak, szintén sokféle pont, szimbólum, és végül számos felirat alkot. A különböző tematikák egy papírlapon való ábrázolása gyakran eltérő stílusú vonalak együttfutását eredményezi. A térkép olvashatósága miatt ilyenkor különböző ábrázolási konvenciók nehezítik a gépi felismerést. A térképet kevéssé ismerő ember számára is nehezen érthetőek ezek a helyzetek.

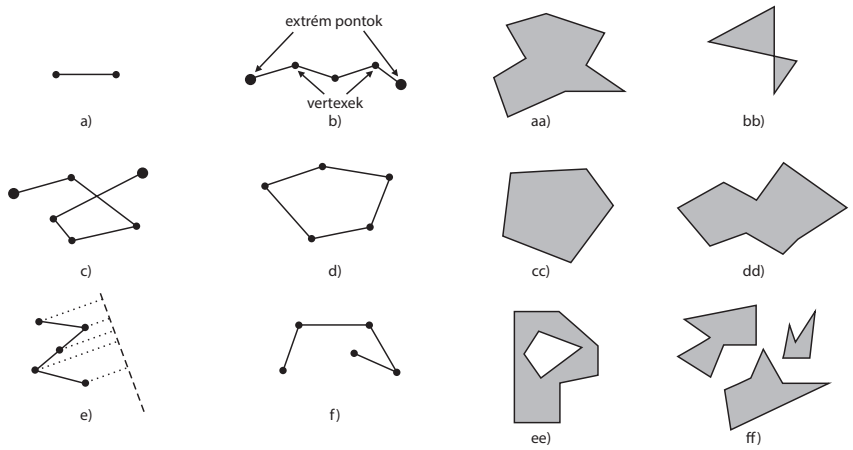
További vektorizálási szempont, hogy milyen célra készül a vektortérkép. Az előre definiált célokból következtetni lehet arra, hogy mely objektumot tekintsük fontosnak, és melyet csak háttérnek, vagy éppen elhanyagolhatónak. Példaként vegyük azt az esetet, amikor közigazgatási célból készítünk vektoros térképet szkennelt papírtérkép alapanyagból. Ilyenkor minden közigazgatási vonatkozású adat fontos (településhatár, megyehatár, stb.), ellenben a folyók pontos ábrázolást nem igényelnek, noha jelenlétük a tájékozódás szempontjából szükséges. A patakok elhanyagolhatók, nem is kell, hogy rákerüljenek a digitális térképre (még akkor sem, ha rajta vannak a papír alapanyagon).

Amikor papírtérkép a vektorizálás nyersanyaga, akkor is igaz, hogy valamikor valakinek értelmeznie kellett a képet, valakinek valamilyen előismeretei alapján ki kellett tűzni mérési pontokat. Összefoglalva tehát elmondhatjuk, hogy a vektoros adatmodellt követő digitális térképekben a helyvektorokon és az összekötési szabályon kívül nagy mennyiségű emberi tudás és szakértelem is benne van.

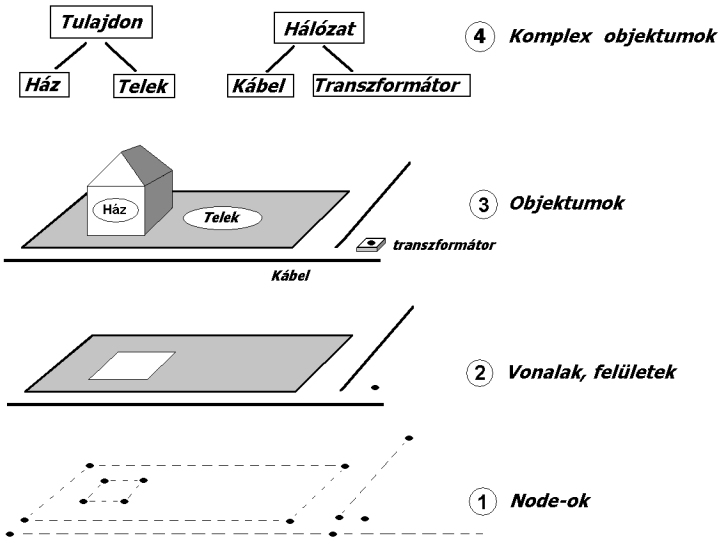
A 1.1. ábrán látható egy idealizált kép a valóság egy szeletéről, amelyet vektorosan írunk le. A valóságról alkotott modellünkhöz geometriai elemeket használunk, úgymint a nulla dimenziós pont, és az egydimenziós vonal.



1.1. ábra. A valóság egy darabja, az ELTE IK madártávlatból (fenti ábra), és ugyanez a vektoros adatmodellel leírva (alsó ábra). A helyvektoros leírás – mint látható – elhelyezte az ábrázolni kívánt objektumok felszínre merőleges vetületeinek jellemző pontjait egy koordináta-rendszerben. Az erősen kiemelt sarokpontok helyvektorait tároljuk, valamint a pontok összekötési szabályát



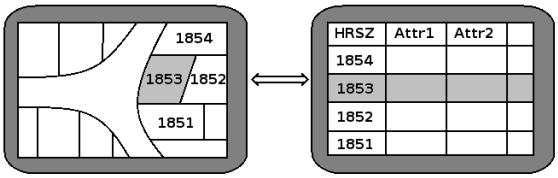
1.2. ábra. Példák egydimenziós objektumokra (bal oldali ábra): a) vonal szegmens (szakasz); b) vonallánc (polyline) extrém pontokkal (végpontokkal) és vertexekkel; c) nem kereszteződő vonallánc; d) zárt polyline; e) monoton polyline; f) nem monoton polyline.
Példák kétdimenziós objektumokra (jobb oldali ábra): aa) egyszerű poligon; bb) nem egyszerű poligon (ilyen poligon csak hibaként fordul elő a geoinformatikában); cc) konvex poligon; dd) monoton poligon; ee) poligon szigetstruktúra (lyukas poligon); ff) diszjunkt poligonokból álló régió



1.3. ábra. A vektoros rendszerek elemeiből (geometric primtives) egyre komplexebb objektumok, objektumcsoportok alkothatók: 1 – a node-ok szintje; 2 – vonalak, poligonok szintje; 3 – objektumok szintje; 4 – komplex objektumok szintje

Ennek speciális esete a vonallánc (polyline), amely egymáshoz kapcsolódó szakaszok sorozata (az összekapcsoltság már ebben az esetben is topológiai, vagyis a szomszédsággal kapcsolatos megszorításokat ró a modellre). A következő geometriai elem a (kétdimenziós) poligon, amelyet pontok sorozata épít fel (1.2. ábra). Ennél precízebb megfogalmazás, hogy egy poligon egymáshoz kapcsolódó szakaszok sorozata, amelynek kezdő és végpontja azonos, de legfőbb tulajdonsága a körbezárt terület. Az egymáshoz kapcsoltság, valamint a zártság a topológiára, vagyis a vonalak szomszédsági viszonyaira vonatkozó megszorítás.

Amint a 1.3. ábrán látható, a legelemibb építőelem, csak a pont tartalmaz koordinátaadatot. A többi objektum, a hierarchiában feljebb álló vonal, vonallánc és a poligon csak strukturális információt tartalmaz arról, hogy a magasabb hierarchiájú objektum milyen viszonyban van a hierarchiában alatta lévőkkel.



1.4. ábra. A grafikus és a leíró adatok kapcsolatának sematikus ábrája. A grafikus objektumok és a leíró adataik 1:1 típusú kapcsolatban vannak egymással

Ez egyben azt is jelenti, hogy a pont és koordinátáinak tárolása nem elegendő a geometria vektoros leírásához, hanem azt is tárolni kell, hogy a hierarchiában a pontok felett álló objektumok miként jönnek létre a pontokból alkotott szakaszok, és azok esetleges kapcsolódása által.

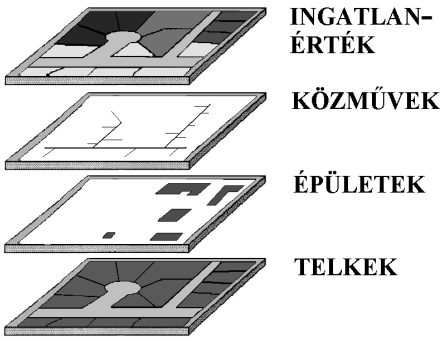
A vektoros objektumleírásnak számos előnye van, mint például:

- Csak ott van pont (vertex), ahol változás, azaz törés van a geometriában
- Helytakarékos grafikaleírás
- Objektum hierarchia felépítésének lehetősége (1.3. ábra)
- Korlátlan adatbázis-kapcsolati lehetőségek (1.4. ábra)

1.2. Kapcsolatok, rétegek, feature classok

Az adatbázis-kapcsolatot a 1.4. ábra mutatja sematikusan. A geometriai hierarchia bármely szintjéhez rendelhetünk attribútumadatokat leíró táblákat, ha azt a problémakör üzleti logikája (business logic) megkívánja.

A vektorosan tárolt adatokat a spagettimodellben általában logikai csoportokba szedve kezelik, attól függően, hogy milyen csoportosítást részesítünk előnyben (1.5. ábra). Egy lehetőség az önkényes csoportosítás, amely a tervező, rendszerépítő deklarációjától függ. Ő dönti el, hogy mely objektumok kerüljenek egy objektumcsoportba (feature class, layer). Ebben az esetben igen tág tere nyílik a tervezőnek, de egyben megnő a felelőssége is.

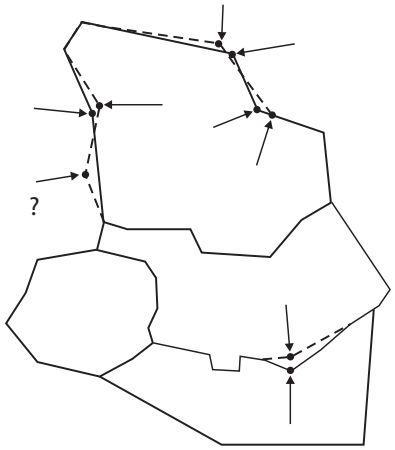


1.5. ábra. Objektumcsoportok létrehozásával rugalmasan használható adatszerkezet hozható létre. Az objektumorientált megközelítésű rendszerek ezeket *feature class*oknak hívja (tulajdonságosztályok), a hagyományos, CAD-terminológiát előnyben részesítő szoftvergyártók pedig inkább a *layer* elnevezést használják. Az ábrán egy számított feature class is látható, az **ingatlanérték**, amely a **telkek**, **épületek** és **közművek** osztályokból számítható

Egy másik lehetőség az objektumok valamilyen tulajdonságazonosság alapján történő besorolásán, osztályozásán alapuló csoportosítás. A csoportosítás lehet fix, vagyis nem változtatható, de lehet rugalmasan változtatható is, ami a pillanatnyi legyűjtési szempontrendszer alapján végzett csoportosítás.

A vektoros adatokat nemcsak koordináták szerint, hanem egymáshoz képest is megadhatjuk, sőt célszerű a szomszédsági, a kapcsolódási viszonyokat is figyelembe venni. Számos hétköznapi eset mutat „topologikus" megfontolásokat: „Hogyan találok oda a pályaudvarra?" kérdezi az eltévedt turista. „Ezen az utcán menjen végig a második kereszteződésig, ott forduljon jobbra, menjen addig, amíg az út nem keresztez egy vasúti átkelőt, majd a sín mellett haladva eléri az állomást." Egy másik, topológiát rejtő kérdés: „Mi van a szomszédom telke és az én telkem között?" „Semmi, csak a kerítés (a kerítés nem telek, hanem egy egydimenziós objektum), mert ha lenne, akkor nem volnánk szomszédok." A topológia tehát leírja az objektumok szomszédsági viszonyait, az objektumok kapcsolatrendszerét (átfedések, szakadások).

A spagettimodell egy olyan vektoros adatmodell, amely a node-okon és az összekötési szabályon kívül mást nem vesz figyelembe. Nem vizsgálja, hogy van-e közvetlen szomszédja egy poligonnak, és így vannak-e közös node-jai a szomszédos poligonoknak. Nem törődik a folytonossággal, és az egyes objektumok esetleges térbeli sorrendiségével, szomszédságával (1.6. ábra).



1.6. ábra. A spagettimodellben előforduló hibák egyike. Az ábrán a folytonos és a szaggatott vonalas poligonoknak egymáson kellene futniuk, de amint a nyilak mutatják, ez nem teljesül, sőt a **?** jellel jelölt nyíl olyan node-ot mutat a szaggatott poligonban, amelynek nincs is megfelelője a folytonos vonallal jelöltben.

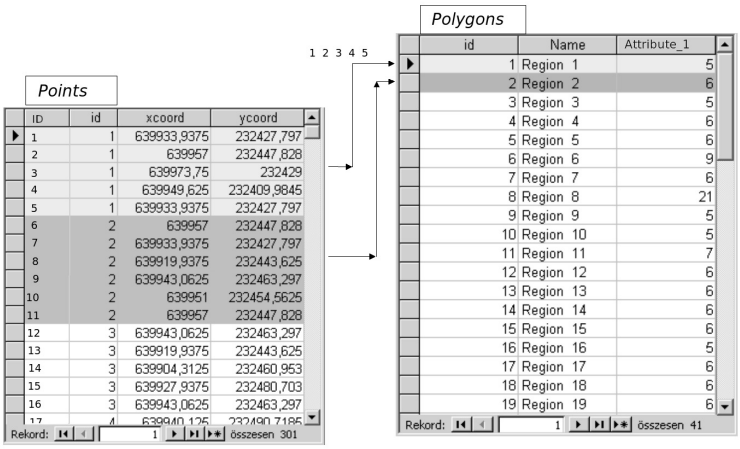
A legtöbb általános célú, vektoros rajzoló spagettimodellt követ. Előnye az egyszerűsége, amely egyben a hátránya is, mivel olyan laza minőségi kritériumok mellett, mint amilyet a spagettimodell kíván meg, nagyon könnyű rossz minőségű adatbázist létrehozni. Az egyszerűség persze nem zárja ki, hogy hozzáértők a spagettiből is professzionális minőséget hozzanak létre. Az általános rajzoló programok saját adatformátumot használnak, amely (többnyire) nem tartalmaz relációs adatbázisok elméletére épülő megfontolásokat, a szomszédságot, a kapcsolatrendszert leíró megszorításokat.

1.3. Spagettileírás relációs logikával

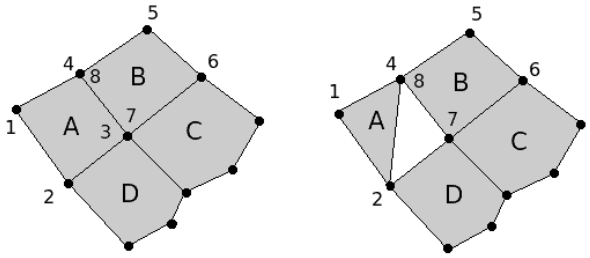
Meglepő, hogy a legtöbb térinformatikai szoftver a fentiek ellenére alkalmazza a spagettimodellt. Amikor poligonok zártságát, vonalláncok szakadásmentességét kívánjuk ellenőrizni, akkor azt a szoftverek külön utasításra megteszik, de az adatmodell nem tartalmaz erre vonatkozó garanciát, azt a node-koordináták összehasonlításával lehet megtenni. A 1.7. ábrán a spagettimodell egy lehetséges megvalósítása látható. Relációs adatbázis-kezelők tábláiban tároljuk a pontok és a poligonok adatait. A *Points* táblában a pontok azonosítói (*Points.ID*, elsődleges kulcs), és a pontok koordinátái vannak (*Points.xcoord*, *Points.ycoord*), valamint egy idegen kulcs (*Points.id*), amely a *Polygons* nevű tábla elsődleges kulcsára (*Polygons.id*) mutat. E mező alapján megállapítható, hogy az egyes poligonokat mely pontok alkotják. Például a *Polygons.id*=2 azonosítójú, és „Region 2" nevű poligont azok a pontok alkotják, amelyeknek *Points.id* mezője 2 értékű (lásd az 1.7. ábra szürkével kiemelt rekordjait). Figyeljük meg azonban, hogy a „Region 1" nevű poligont alkotó pontok között van olyan (*Points.ID*=2), amelynek koordinátái megegyeznek a a „Region 2" nevű poligon egyik node-jával (*Points.ID*=6). Ez a körülmény azt mutatja, hogy a találkozó poligonok határainak azonos koordinátájú pontjai redundáns módon vannak tárolva.

Az adatokból kiolvasható, hogy hézag- és átfedésmentes topológiájú az adatállomány, de a tárolás redundáns mivolta, valamint az a körülmény, hogy az adatbázis nem tartalmaz semmiféle adatot a szomszédságra vonatkozóan, lehetővé teszi, hogy a *Points* tábla bármely pontját megváltoztatva (eltolás, törlés) elromlik a topológia (1.8. ábra). Akkor lenne csak megőrizhető a topológia, ha az egyik poligonban bekövetkezett változás a másikban is bekövetkezne. A spagetti adatmodell erre nem képes.

A 1.8. ábrán látható továbbá, hogy az **A(1,2,3,4)** és **B(5,6,7,8)** poligon (zárójelben az őket alkotó node azonosítók). Töröljük ki **A**-ból a 3 számú node-ot. Ekkor **A(1,2,4)**-re változik, míg **B** változatlan marad. Így hézag keletkezik a poligonstruktúrában, mivel a 7 számú node a **B** poligonhoz tartozik, vagyis a spagettitopológia adatszerkezete nem biztosítja a topológia megőrzését. Természetesen szoftveres eszközökkel (kis környezetbe eső pontok figyelésével) elkerülhető effajta hibák elkövetése, de világosan látható a túl egyszerű szerkezet hiányossága. Előnye az egyszerűsége. Ennek köszönhetően számos jól ismert GIS-szoftver is spagetti adatszerkezetet használ. Amelyik szoftverrel a 1.8. ábrán látható jelenség előállítható, az spagetti adatmodellt használ.



1.7. ábra. A node-ok és a belőlük alkotott poligonok relációs táblák kapcsolatával leírhatók. Ez az egyszerű szerkezet azonban nem tartalmaz semmilyen információt a topológiáról. Vegyük észre, hogy az ábrán látható poligonok szomszédosak, így a *Points* tábla több azonos koordinátájú rekordot is tartalmaz



1.8. ábra. Spagetti adatszerkezetben el lehet rontani a hézag- és átfedésmentes topológiát. Az **A** jelű poligon 3 számú node-ját kitörölve hézag keletkezik, mivel a 7. számú node egy másik rekord a *Points* táblában. Az adatszerkezet tehát nem biztosítja a korrekt topológia fennmaradását

2

Az OGC-topológia kezelése

Ebben a fejezetben áttekintjük az Open Geospatial Consortium (OGC) ajánlásait, amely a létező rendszerek közös elvi alapja. Az áttekintés nem lesz teljes, mivel a szabvány számos olyan előírást is tartalmaz, amely nem része a könyvnek (pl. feliratok, fontok, jelkulcs). Akit az OGC ajánlásai minden részletre kiterjedően érdekelnek, az letöltheti a teljes szabványt a *http://www.opengeospatial.org/* oldalról.

A szabvány az ábrázolni kívánt objektumok és elemeik tárolásának adatbázis szemléletű leírását adja, amely ma már minden térinformatikai rendszer elvi alapját képezi. A piacvezető adatbázis-kezelő rendszerek is ennek a szabványnak megfelelően tárolják a térbeli információkat.

Mint látni fogjuk a specifikációk spagettitopológiát definiálnak. Objektumonként tárolják az alkotóelemeket (pontokat), így a redundanciamentesség nem biztosított. Ez a szabvány nem definiál olyan adatstruktúrát, amely a topológiát megőrizné, így nem nevezhető topologikusnak az adatszerkezet sem. Ez azonban nem jelenti azt, hogy ne lehetne korrekt topológiájú adatrendszereket létrehozni, elvégre a szakadásmentesség, a hézag- és átfedésmentesség így is biztosítható.

A következőkben áttekintjük az alkalmazott mozaikszavak, rövidítések magyarázatait (2.1. táblázat), és a jelölések rendszerét (2.2. táblázat).

2.1. Fogalommeghatározások

Boundary (határ) Azon pontok halmazát nevezzük boundarynak, ami valamely entitás határait adja meg. Leggyakrabban geometriai értelemben használjuk pontokkal definiált területek meghatározásához.

2.1. táblázat. Rövidítések

API	Application program interface
COM	Component object model
CORBA	Common object request broker architecture
DCE	Distributed computing environment
DE-9IM	Dimensionally extended nine-intersection model
FID	Feature ID column in the implementation of feature tables based on predefined data types
GID	Geometry ID column in the implementation of feature tables based on predefined data types
MM	Multimedia
NDR	Little endian byte order encoding
OLE	Object linking and embedding
RPC	Remote procedure call
SQL	Structured query language
SRID	Spatial reference system identifier
SRTEXT	Spatial reference system well known text
UDT	User defined type
UML	Unified modeling language
WKB	Well known binary
WKT	Well known text
WKTR	Well known text representation
XDR	Big endian byte order encoding

2.2. táblázat. Jelölések

nD	n-dimenziós, ahol n egész szám
$\Re^n$	n-dimenziós tér, ahol n egész
$\emptyset$	üres halmaz
$\cap$	közös rész
$\cup$	két vagy több halmaz uniója
$-$	két halmaz különbsége
$\in$	eleme egy halmaznak
$\notin$	nem eleme egy halmaznak
$\exists$	létezik
$\nexists$	nem létezik
$\forall$	minden
$\subset$	valódi részhalmaza
$\subseteq$	részhalmaza
$\Leftrightarrow$	akkor, és csak akkor
$\Rightarrow$	implikáció
$\ni$	oly módon
$f : D \rightarrow R$	D tartományból R tartományba leképező függvény
$\{X \mid s\}$	X halmaz, feltéve, hogy s feltétel igaz
$\wedge$	ÉS, logikai metszet
$\vee$	VAGY, logikai unió
$=$	egyenlő
$\neq$	nem egyenlő
$\leq$	kisebb vagy egyenlő
$<$	kisebb
$\geq$	nagyobb vagy egyenlő
$>$	nagyobb
$\ni$	oly módon, hogy …
∂	topologikus boundary operátor

Buffer (hatászóna) A buffer egy olyan geometric object (geometriai objektum), amelynek a távolsága egy adott geometric objecttől kisebb vagy egyenlő, mint egy előre megadott küszöbérték.

Coordinate (koordináta) A koordináta olyan n elemű számsorozat, amelyet n-dimenziós pontok pozíciójának megadására használunk. Egy referencia-koordinátarendszerben ehhez mértékegységet is meg kell adnunk.

Coordinate dimension (koordináta dimenzió) A coordinate dimension tengelyek vagy mértékek száma, amelyet pozíció megadására használunk egy adott koordináta-rendszerben.

Coordinate reference system (vonatkoztatási rendszer) A coordinate reference system egy koordináta-rendszer, amely a datum révén kapcsolódik a valós világhoz.

Coordinate system (koordináta-rendszer) A koordináta-rendszer azon szabályok összessége, amelyek alapján meg lehet adni bármely pont koordinátáit.

Curve (görbe) A görbe egy egydimenziós geometriai elem, amely folyamatosan leképez egy vonalat. Egy boundary is tekinthető görbének, ahol a görbe pontok sorozatából áll. Ha a görbe topológiai értelemben zárt, akkor a kezdő és végpontja azonos.

Direct position (pozíció) A pozíció leírása koordináták sorozatával egy adott vonatkoztatási rendszerben.

End point (végpont) Egy görbe végpontjának megadása. A kezdő- (start) és végpont (end point) a görbe irányítottságával együtt értelmezhető. Egy görbe tükrözése (flip) a kezdő- és végpontok felcserélését jelenti anélkül, hogy a görbe pontjainak rendjét megváltoztatnánk.

Exterior (külső határ) Különbségtétel a belső és a külső világ között. Ez a különbségtétel alkalmazandó a topologikus megfogalmazások esetére (külső, belső).

Feature (osztály) A világbeli jelenségek absztrakciója, amely lehet egy típusa vagy egy példánya (feature instance) a világ valamely dolgának.

Feature attribute (osztály leíró adatok) Egy feature valamely karakterisztikus adata. Ez lehet típus, név vagy valamely tartomány értéke, ami kapcsolódik a feature-höz. A geometria csak egy attribútumtípusa a feature-nek.

Geometric complex (komplex geometriai objektumok) Olyan komplex objektumok, amelyek elemi geometriai objektumokból állnak. Felbonthatók elemi geometriai objektumokra.

Geometry object Térbeli objektum, amely elemi geometriagyűjtemények halmaza.

Geometric primitives (geometriai elemek) Elemi geometriai objektumok, amelyek egyszerű, összefüggő, homogén elemei a térnek. Az elemi geometriai objektumok tovább nem bontható, legkisebb egységek.

Interior (belső határ) Különbségtétel a belső és a külső világ között. Belsőnek lenni azt jelenti, hogy kérdéses pozícióadatok egy geometriai objektumon belül vannak, de nincsenek rajta a határon (boundary).

Linear referencing system (lineáris referenciarendszer) Lineáris referencia szerinti pozicionáláskor egy adott referenciaponttól egy távolságot adunk meg egy útvonal mentén.

Point (pont) Nulla dimenziós elemi geometriai objektum, amely egy pozíciót reprezentál.

Simple feature (egyszerű osztály) Egyenes vonalszakaszok sorozatából, vagy pontok halmazára végzett síkbeli interpolációval kapott pontok sorozatából álló halmaz. Lényegében az osztály minden pontja előállítható kontrollpontok paraméterezése révén. A görbék valamennyi részének van két kontrollpontja (kezdő- és végpont). Bármely P pont előállítható a következő módon: $P = tP_0 + (1-t)P_1$, ahol $0 < t < 1$.

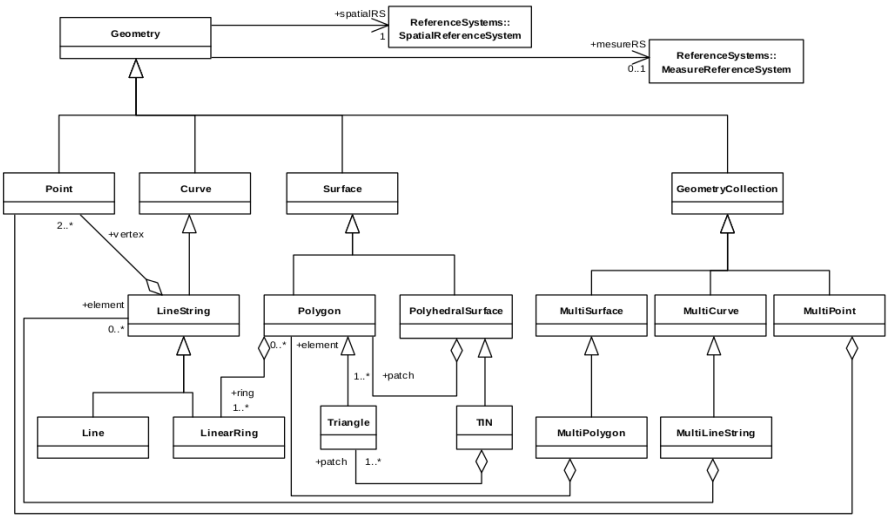
A felületek, poligonnak tekintve őket, felbonthatók háromszögek halmazára, amely három kontrollponttal adhatók meg (P_0, P_1, P_2). A háromszög bármely P pontja előállítható ezeket a pontokat tartalmazó vektoregyenlettel („barycentric coordinates"): $P = aP_0 + bP_1 + cP_2$, ahol a, b, c valós számok, $a, b, c > 0$, $a + b + c = 1$.

Start point (kezdőpont) Egy görbe kezdőpontja.

Surface (felület) Egy 2-dimenziós topologikus elemi geometriaobjektum, amely a síknak egy régióját reprezentálja.

2.2. A geometry object modell

Ebben részben áttekintjük a OGC architektúráját, amelyben megvizsgáljuk a Geometry object modellt, a well-know text geometria reprezentációját, a



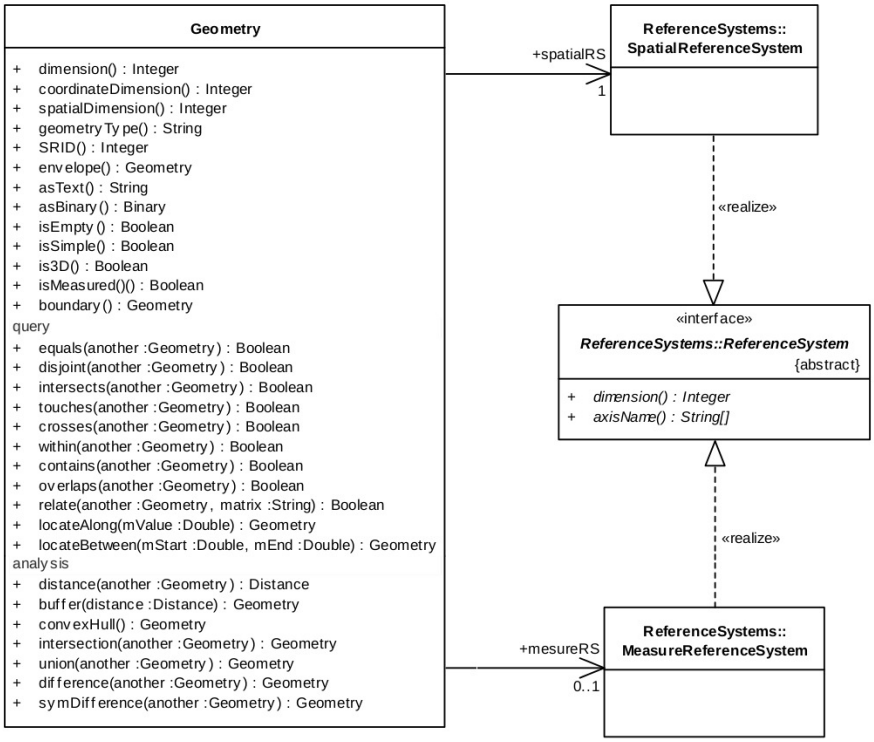
2.1. ábra. Az OGC-geometria osztályainak hierarchiája. Ez a modell 0, 1, 2 dimenziós objektumok gyűjteményeire lett kidolgozva, úgy, mint MultiPoint, MultiLineString és MultiPolygon, modellezve a Points, LineStrings és Polygons geometriai objektumokat. A MultiCurves és a Multisurfaces mint szuperosztályok lettek bevezetve, amelyek a collection interface általánosításai, hogy kezeljék a Curves és a Surfaces gyűjteményeket. A nem homogén gyűjtemények a GeometryCollection példányai

well-know binary geometria reprezentációját, és végül a well-known text térbeli vonatkoztatási rendszerének reprezentációját.

A geometry objektum modell Distributed Computing Platform független, és az UML-t használja. A 2.1. ábrán az OGC-geometria osztályainak hierarchiája látható. Az alap geometry osztálynak vannak alosztályai, úgy mint pont, görbe, felszín és geometria gyűjtemény. Minden geometriai objektum egy térbeli referenciarendszerben értelmezendő, amely leírja azt a koordináta-rendszert, amelyben a geometriai osztályt definiáltuk.

2.2.1. Geometry

Az OGC hierarchiájában a geometry a gyökérben helyezkedik el, és mivel egy abstract osztály, nem példányosítható. Példányosítható viszont a geometry osztály, 0, 1, 2 dimenziós geometriaobjektumokra korlátozva, amelyek



2.2. ábra. A geometry osztály műveleteinek elvi felépítése

2, 3, 4 dimenziós térben létezhetnek ($\Re^2, \Re^3, \Re^4$). A geometrynek R^2-ben vannak vannak pontjai x, y koordinátaértékekkel, R^3-ban x, y, z, vagy x, y, m értékekkel (ahol m mérték, amely lineáris referenciarendszerben a vonatkoztatási ponttól való „távolságot" adja meg, ami nem euklideszi távolság), R^4-ben pedig x, y, z, m értékekkel. A koordinátákat a referencia-koordinátarendszerben kell értelmezni. A z koordináta tipikusan magasságként tekinthető, bár lehet más is.

Műveletek geometriai objektumokon

A 2.2. ábra a geometry osztály műveleteit mutatja. Ezek a következők:

Dimension Típus: Integer. A geometriaobjektum a dimenziót magában foglalja. Kisebb vagy egyenlő lehet, mint a koordináta-rendszerének dimenzió-

ja. Inhomogén gyűjtemények esetén az objektum dimenziója kisebb vagy egyenlő, mint az őt magát tartalmazó objektum dimenziója.

Geometry type Típus: String. Az objektum példány visszatérési értéke az objektum típus neve.

SRID Típus: Integer. Térbeli referenciarendszer-azonosító (spatial reference system id) az adott geometriai objektum azonosítója.

Envelope Típus: Geometry. A minimális befoglaló négyszöget (MBB) jelenti az adott geometriai objektumhoz. Poligonok esetében ez a poligon befoglaló négyszögeinek sarokpontjait jelenti: [(MINX, MINY), (MAXX, MINY), (MAXX,MAXY), (MINX,MAXY), (MINX, MINY)]. Minimum z és m értékek is beletartozhatnak. Az envelope legegyszerűbb reprezentációja két direct position, az egyik a minimumokat, a másik a maximumokat tartalmazza.

AsText Típus: String. Well-known text export esetén ez reprezentálja a geometric objectet.

AsBinary Típus: Binary. Well-known binary export esetén ez reprezentálja a geometric objectet.

IsEmpty Típus: Integer. Visszatérési értéke 1, ha a geometric object üres. Logikai típusként (boolean) is értelmezők az értékek. Ekkor 1:TRUE, 0:FALSE.

IsSimple Típus: Integer. Értéke 1, ha a geometric objectnek nincsenek elfajult pontjai (önmagát metsző, önmagát érintő részek).

Is3D Típus: Integer. Értéke 1, ha a geometric objectnek vannak z koordinátaértékei.

IsMeasured Típus: Integer. Értéke 1, ha a geometric objectnek vannak m koordinátaértékei.

Boundary Típus: Geometry. Visszatérési értéke a geometriai objektum határa (boundary). Tekintettel a topológiai zártságra, ez az érték a geometriai objektum reprezentációját is jelenti elemi geometriákkal (geometric primitives).

Térbeli relációk geometriai objektumok között

A térbeli relációk visszatérési típusa egész, bár logikai (boolean) értékként is értelmezhető.

Equals (egyenlő) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben egyenlő" *geom*$_2$-vel.

Disjoint (szétválaszt) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben különálló" *geom*$_2$-vel.

Intersects (metszi) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben metszi" *geom*$_2$-t.

Touches (érinti) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben érinti" *geom*$_2$-t.

Crosses (kereszteződik) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben keresztezi" *geom*$_2$-t.

Within (benne van) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térbelileg benne van" *geom*$_2$-ben.

Contains (tartalmazza) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térbelileg tartalmazza" *geom*$_2$-t.

Overlaps (átfedi) (*geom*$_2$: Geometry) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben átfedi" *geom*$_2$-t.

Relate (kapcsolódnak) (*geom*$_2$: Geometry, intersectionPatternMatrix: String) Típus: Integer. Visszatérési értéke 1, ha a geometriai objektum „térben kapcsolódik" *geom*$_2$-vel.

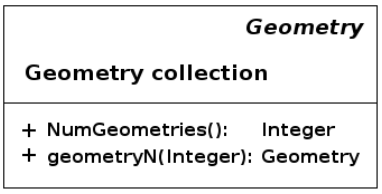
LocateAlong (helyet megállapít vonal mentén) (mValue:Double) Típus: Geometry. Visszatérési értéke egy geometriagyűjtemény, amely megfelel a specifikált *m* értéknek.

LocateBetween (helyet megállapít vonal mentén két pont között) (mStart: Double, mEnd:Double) Típus: Geometry. Visszatérési értéke egy geometriagyűjtemény, amely megfelel a specifikált *m* értékeknek.

Térbeli elemzésekhez támogató funkciók

Distance (távolság) (*geom*$_2$:geometry) Típus: double. Visszatérési értéke két pont legrövidebb távolsága, amely térbeli referenciarendszerben értendő.

Buffer (distance:double) Típus: geometry. Visszatérési értéke egy geometria, amely minden olyan pontot tartalmaz, amely kisebb vagy egyenlő a distance (távolság) értéknél. A számítások térbeli referenciarendszerben értendők.



2.3. ábra. A geometriagyűjtemény (geometry collection) és metódusai: *NumGeometries* – a gyűjteményben lévő geometriai objektumok száma; *GeometryN* – az *N*-edik geometry object a gyűjteményben

ConvexHull (konvex burok) Típus: geometry. Visszatérési értéke egy geometria, amely a geometriai objektum konvex burkát reprezentálja.

Intersection (metszőpont) (*geom*$_2$:geometry) Típus: geometry. Visszatérési értéke egy geometria, amely a két geometria egymást metsző pontjainak halmazából áll.

Union (egyesítés) (*geom*$_2$:geometry) Típus: geometry. Visszatérési értéke egy geometria, amely a két geometria egyesítéséből áll.

Difference (különbség) (*geom*$_2$:geometry) Típus: geometry. Visszatérési értéke egy geometria, amely a két geometria különbségéből áll.

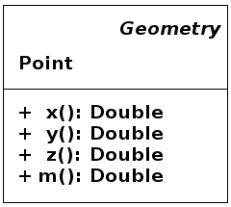
SymDifference (*geom*$_2$:geometry) Típus: geometry. Visszatérési értéke egy geometria, amely a két geometria szimmetrikus különbségéből áll.

Geometry collection (geometriagyűjtemény)

A Geometry collection (2.3. ábra) egy gyűjtemény, amely számos geometriai objektumot (geometry object) tartalmaz. Valamennyi geometriai objektumnak ugyanabban a térbeli referenciarendszerben kell lenni, akár csak a gyűjteménynek. Más egyéb megszorítás nincs a geometriai objektumokra nézve.

Point (pont)

A point (2.4. ábra) egy nulla dimenziós geometriai objektum, amely egy pozíciót határoz meg egy koordináta-rendszerben. Általában x és y koordinátákkal rendelkezik, de lehet z vagy m koordinátája is. A pont határai (boundary) üres halmaz.



2.4. ábra. A pont és metódusai: *x()* – a pont x koordinátáját adja vissza; *y()* – a pont y koordinátáját adja vissza; *z()* – a pont z koordinátáját adja vissza; *m()* – a görbe m koordinátáját adja vissza

Multipoint

A MultiPoint egy nulla dimenziós geometriaiobjektum-gyűjtemény.

Curve (görbe)

A Curve egy egydimenziós objektum (2.5. ábra), pontok sorozatából áll, amely a pontok közti interpoláció révén jön létre. Zárt, ha kezdő és végpontja azonos. A curve egy egydimenziós leképezése egy homomorf, valós, zárt intervallumnak:

$$D = [a,b] = \{t \in \Re | a \leq t \leq b\}$$

$$f : [a,b] \rightarrow \Re^n$$

ahol n a térbeli vonatkoztatási rendszer koordináta-dimenziójának a száma.

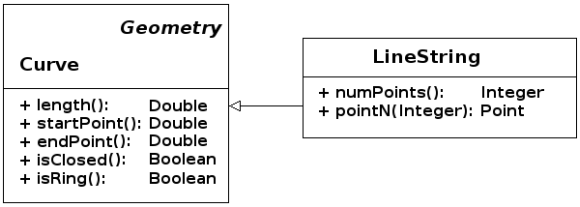
A curve egy altípusa a linestring, ahol a pontok közti interpoláció lineáris. A curve egyszerű, ha nem megy át kétszer ugyanazon a ponton:

$$\forall c \in Curve, \;\; [a,b] = c.Domain, c =: f : [a,b] \rightarrow \Re^n$$

$$c.IsSimple \Leftrightarrow \forall x_1, x_2 \in [a,b] : [f(x_1) = f(x_2) \wedge x_1 < x_2] \Rightarrow [x_1 = a \wedge x_2 = b]$$

A curve zárt, ha kezdő- és végpontja azonos:

$$c : IsClosed \Leftrightarrow [f(a) = f(b)]$$



2.5. ábra. A görbe és metódusai: *length* – A görbe hossza a referenciarendszerben mérve; *startpoint* – a görbe kezdő pontja; *endPoint* – a görbe végpontja; *isClosed* – értéke 1 (TRUE), ha a görbe zárt; *isRing* – értéke 1, ha a görbe zárt és simple (egyszerű)

A zárt curve határa üres halmaz:

$$c.IsClosed \Leftrightarrow [c.Boundary = \emptyset]$$

A curve akkor gyűrű, ha egyszerű és zárt.

LineString, Line, LinearRing

A LineString (2.6. ábra) egy olyan görbe, amelynek pontjai lineáris interpolációval vannak összekötve. Minden egyes pontpár egy vonalszegmenst (LineSegment) alkot. A Line (vonal) pontosan két pontból álló LineString. A LinearRing egy olyan LineString, amely zárt és egyszerű.

MultiCurve

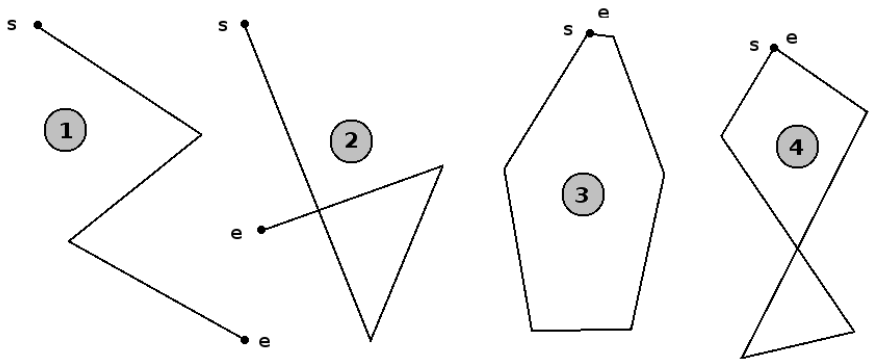
A MultiCurve (2.7. ábra) egy egydimenziós geometriai gyűjtemény, aminek az elemei görbék (curve).

MultiLineString

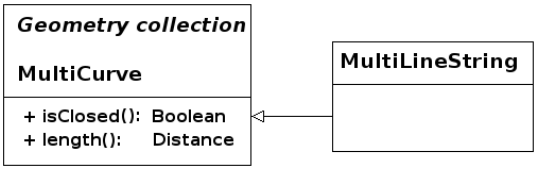
A MultiLineString egy olyan MultiCurve, amelynek elemei LineStringek. A 2.8. ábra példákat mutat MultiLineStringekre, ahol a határok a következők: $1 - s_1, e_2$; $2 - s_1, e_1$; $3 - \emptyset$.

Surface (felület)

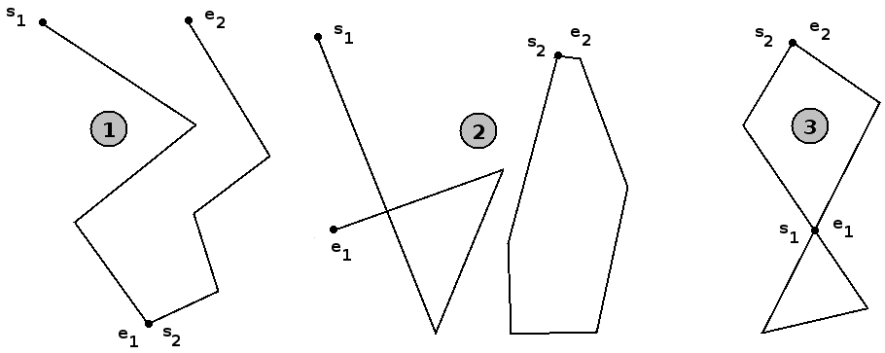
A surface egy kétdimenziós geometriai objektum, amely egyszerű „foltokból" áll. Ezeknek egy külső határa van (boundary), de lehet benne olyan folt is, amelynek 0 vagy több belső határa is van (sziget poligonok). 3 dimenzi-



2.6. ábra. Példák LineStringekre. 1 – egyszerű LinString; 2 – nem egyszerű LineString; 3 – egyszerű, zárt LineString (LinearRing); 4 – nem egyszerű zárt LineString. Az *s* a kezdő pontot, *e* a végpontot jelöli. Metódusai: *NumPoints* – visszatérési értéke a LineStringben lévő pontok száma; *PointN* – visszatérési értéke a LineString egy adott (pointN) pontja



2.7. ábra. A Multicurve és metódusai. *IsClosed* – visszatérési értéke 1 (TRUE), ha a MultiCurve zárt; *Length* – visszatérési értéke a Multicurve hossza, amely a tartalmazott görbék hosszainak összege



2.8. ábra. Példák MultiLineStringekre. 1 – egyszerű MultiLineString; 2 – nem egyszerű, kételemű MultiLineString; 3 – nem egyszerű, zárt, kételemű MultiLineString. Az s_i a kezdő, e_j a végpontokat jelöli

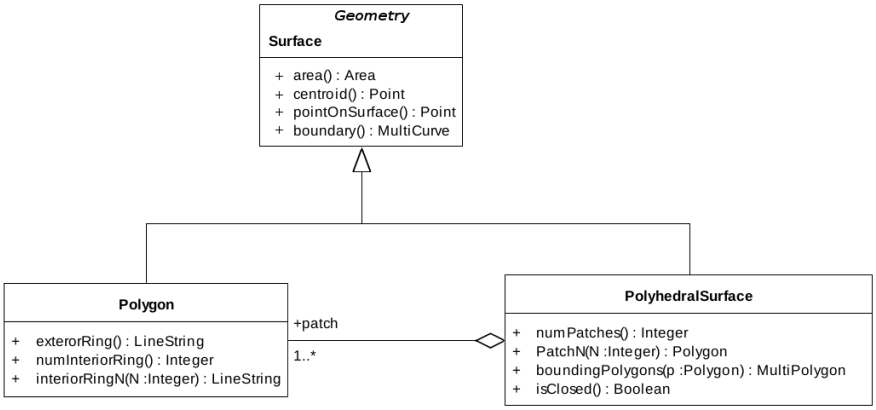
ós felszín esetében a felület izometrikus síktranszformációval képezhető le (affin transzformációval) kétdimenziós felszínné (ebben az esetben $z = 0$).

A polyhedral surfacet (poliéder felszín) úgy kell elképzelni, mint több egyszerű felületet, amelyek közös határvonalak mentén találkoznak (mintegy össze vannak varrva). 3 dimenzióban a polyhedral surface olyan egyszerű felületekből áll, amelyek eltérő normálisaik miatt egyenkénti affin transzformációval képezhető le a síkba. A transzformáció természetesen a találkozó határvonalakat találkozó vonalakba viszi át.

A poligon egy egyszerű sík felület. A polyhedral surface szintén egy egyszerű felület, amely több találkozó poligonból áll. A MultiSurface egy több polyhedral surfaceből álló halmaz, amelyben lehetnek akár lyukas felületek is.

Polygon (poligon), triangle (háromszög)

A poligon egy sík felszín, amelyet egy külső határ, és 0 vagy több belső határ definiál. Minden belső határ egy lyukat definiál a poligonon belül. A háromszög egy három, nem egy egyenesbe eső pontból álló poligon, melynek nincsenek belső határai (nincsenek benne lyukak). A poligonok külső határa egy LinearRing, amelynek körüljárási iránya az óramutató járásával ellentétes. A belső határ(ok), ha van, is LinearRing, de körüljárási iránya az óramutató járásával megegyező.



2.9. ábra. A surface és metódusai. *Area()*: A felület területét adja meg; *Centroid()*: a felület centroidját adja meg; *PointOnSurface()*: garantáltan a felületen lévő pont

A poligonokkal szembeni követelmények a következők:

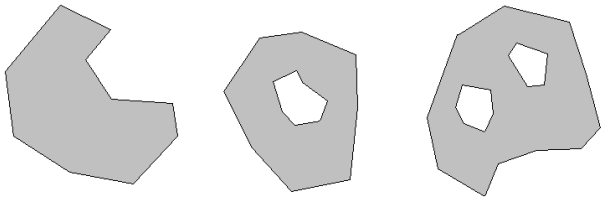
- a poligonok topologikusan zártak
- a poligon határa LinearRing-ek halmazából áll, amelyek a belső és külső határokat adják megadhatjuk
- nincs két egymást keresztező Ring a poligon határaiban, legfeljebb egyetlen közös pontjuk lehet (érintés)

$$\forall P \in Polygon,\ \ \forall c_1, c_2 \in P.Boundary,\ \ c_1 \neq c_2$$
$$\forall p, q \in Point,\ \ p, q \in c_1,\ \ p \neq q$$
$$[p \in c_2] \Rightarrow [\exists \delta > 0\ \ni [|p-q| < \delta] \Rightarrow [q\ \notin c_2]]$$

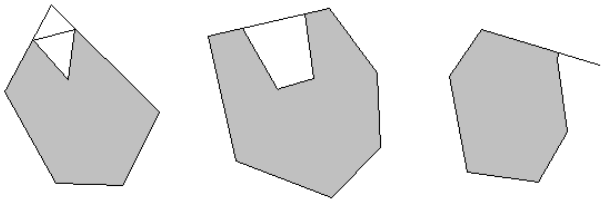
- egy poligonnak nem lehetnek metsző vonalai, nem lehetnek rajta tüskék vagy szakadások

$$\forall P \in Polygon,\ \ P = P.Interior.Closure$$

- a belső határa az egyes poligonoknak kapcsolódó pontok halmaza



2.10. ábra. Példák poligonokra



2.11. ábra. Példák nem megengedett poligonokra

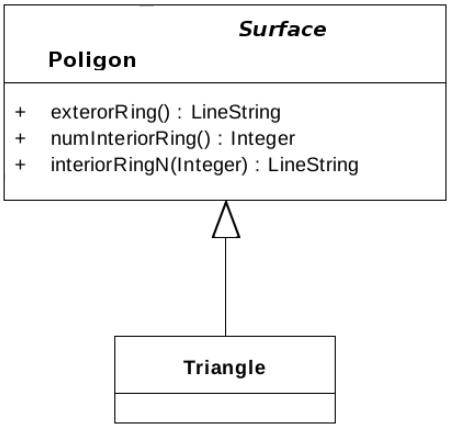
- egy egy vagy több lyukkal rendelkező poligon külső határa nem kapcsolódik. Minden lyuk definiál egy kapcsolódó komponenst a külső határhoz

A 2.10. ábra megengedett poligonokat, míg a 2.11. ábra nem megengedett poligonokat mutat. A 2.12. ábra a poligonokat és metódusaikat mutatja.

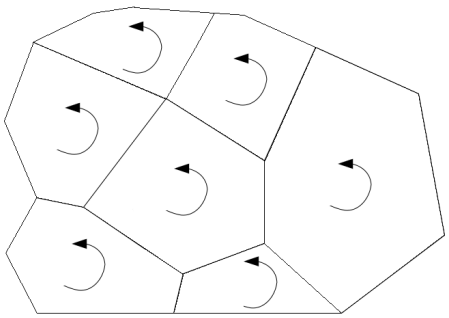
Polyhedral surface (poliéder felület)

A polyhedral surface egymással határos poligonok gyűjteménye, amelyek közös határszegmensekkel rendelkeznek. Valamennyi poligon, amely közös határvonallal rendelkezik, kifejezhetők, mint LineStringek gyűjteménye. A TIN (triangular irregular network) háromszög alakú foltokból álló polyhedral surface.

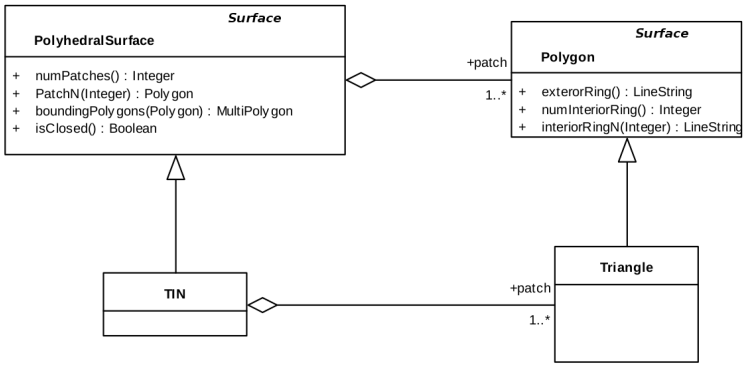
Bármely poligonokra igaz, melyeknek közös határvonaluk van, hogy a poligonok LinearRingjei a közös vonalszegmensen ellentétes irányban haladnak. Így minden határos poligon megfelelően irányított lesz. Egy ilyen polyhedral surface-t szemléltet a 2.13. ábra. A nyilak a poligonok körüljárási irányát mutatják.



2.12. ábra. Polygon és metódusai. ExteriorRing() – Visszatérési értéke a poligon külső gyűrűje (ring, LineString); NumInteriorRing() – az adott poligon belső gyűrűinek száma; InteriorRingN(N: integer) – Visszatérési értéke a poligon N-edik belső gyűrűje



2.13. ábra. A polyhedral surface konzisztens körbejárási irányokkal



2.14. ábra. A polyhedral surface és metódusai. *NumPatches* – visszatérési értéke a tartalmazott poligonok száma; *PatchN* – visszatérési értéke egy poligon tetszőleges körüljárási iránnyal; *BoundingPolygons* – visszatérési értéke egy poligongyűjtemény, amely a megadott felületen van; *IsClosed* – Visszatérési értéke 1 (TRUE), ha a poligon önmagában zárt, így tömör területet zár be (nincs benne lyuk)

MultiSurface

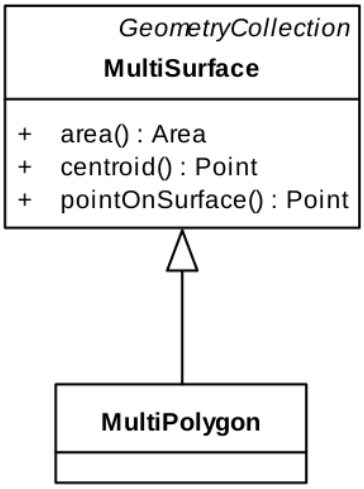
A MultiSurface egy kétdimenziós geometriagyűjtemény, amelynek elemei felületek. Valamennyi felületnek ugyanabban a referencia-koordinátarendszerben kell lenni. Belső határaik nem metszhetik egymást. Ha két felület határa közös pontokból áll, akkor egyesíthetők egyetlen egyszerű felületté. A MultiSurface heterogén geometriagyűjteményeket reprezentálhat, mint például poligonok és polyhedral surface-ek. A MultiSurface a műveletei a 2.15. ábrán láthatók.

Multipolygon

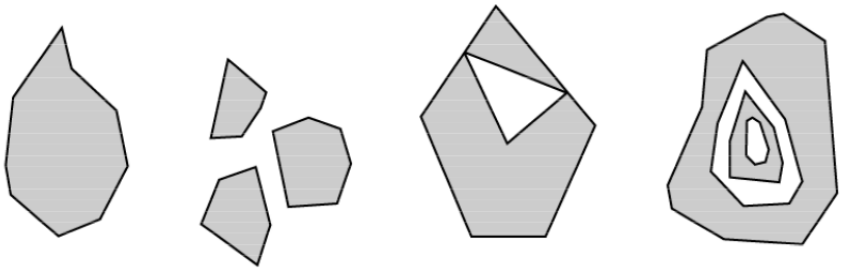
A MultiPolygon egy olyan MultiSurface, amelynek elemei kizárólag polygonok. A MultiPolygonokkal szember a következő követelmények vannak:

- bármely két poligon, amelyek elemei a MultiPolygonnak, belső határai (interior) nem metszhetik egymást.

$$\forall M \in MultiPolygon,\ \ \forall P_i, P_j \in M.Geometries(), i \neq j,$$
$$Interior(P_i) \cap Interior(P_j) = \emptyset$$



2.15. ábra. Multisurface műveletek: *Area(): Double* – visszatérési értéke a MultiSurface területe, amely a referencia-koordinátarendszer egységeiben értendő; *Centroid(): Point* – visszatérési értéke a multisurface centroidja (a centroid a multisurface-en kívülre is eshet); *PointOnSurface(): Point* – visszatérési értéke egy pont, amely biztosan a felületen van



2.16. ábra. Példák MultiPolygonokra

- bármely két polygon, amelyek elemei a MultiPolygonnak, nem keresztezhetik egymást, és csak véges számú pontjuk lehet közös (érintés)

$$\forall M \in MultiPolygon,\ \ \forall P_i, P_j \in M.Geometries()$$

$$\forall c_i, c_j \in Curvec_i \in P_i.Boundaries(), c_j \in P_j.Boundaries()$$

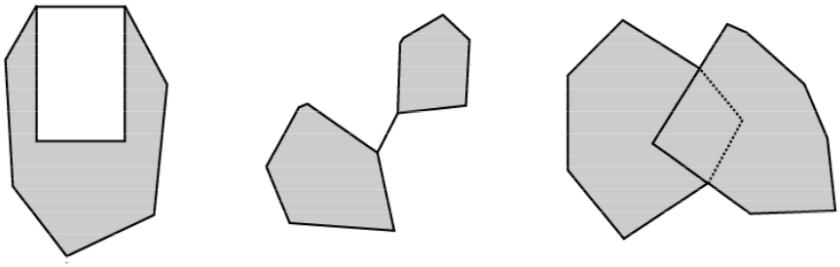
$$\exists k \in Integer \ni c_i \cap c_j = \{p_1, \ldots, p_k | p_m \in Point, 0 < m < k\}$$

- a MultiPolygon definíció szerint zárt
- a Multipolygonban nem lehetnek szakadások, tüskék (spike), lyukak

$$\forall M \in MultiPolygon, M = Closure(Interior(M))$$

- a MultiPolygon belső határa (interior), több mint egy poligon esetén, nincs csatlakozva; a MultiPolygon kapcsolódó komponenseinek belső határai egyenlőek a MultiPolygonban lévő poligonok számával

A MultiPolygon határa zárt görbék halmaza (Curves, LineStrings) a tartalmazott poligonoknak megfelelően. Minden egyes görbe (Curve) a MultiPolygon határában (boundary) megegyezik pontosan egy poligonnal, továbbá az összes poligon határából áll a MultiPolygon határa. A 2.16. ábrán példák láthatók MultiPolygonokra, míg a 2.17. ábrán nem megengedett MultiPolygonok láthatók.



2.17. ábra. Meg nem engedett MultiPolygonok

2.2.2. Térbeli relációs operátorok

A térbeli relációs operátorok boolean visszatérési értékű eljárások, amelyek arra szolgálnak, hogy megállapítsuk két geometriai objektum térbeli kapcsolatát. Ezt oly módon valósítják meg, hogy a geometriai objektumokat 2D-s referenciafelszínre vetítik, majd megvizsgálják, hogy van-e geometriai kapcsolat a két objektum között, pontosabban, hogy van e kapcsolat az objektumok határai (boundaries), belső (interior) és külső (exterior) határai között. A kapcsolatrendszert egy 3×3-as mátrixszal reprezentálják, amelynek neve *Intersection matrix*. Felmerülhet a kérdés, hogy a kapcsolatok vizsgálatakor miért vetítik az objektumokat 2D-be, és miért nem végzik el a műveleteket 3D-ben. Noha elvileg ez az eljárás elvégezhető lenne 3D-ben is, de a megnövekedett számítási idő miatt, performancia okokból, az egyszerűbb és lényegesen gyorsabb 2D-s műveleteket részesítik előnyben.

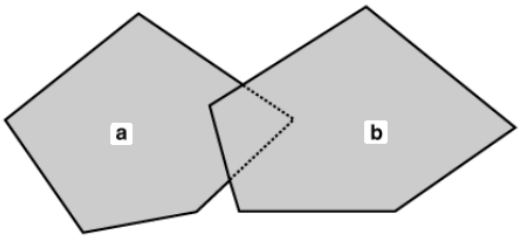
Egy geometriai objektum határa egy nála eggyel kisebb dimenziójú geometriai objektum. Például egy Point vagy MultiPoint határa üres halmaz; egy nem zárt Curve határa a görbe két végpontja; egy zárt görbe határa üres halmaz; egy MultiCurve határa azokból a pontokból áll, amelyek a tartalmazott görbék páratlan számú határaiból állnak; egy poligon határa gyűrűk (Rings) sorozatából áll; egy MultiPolygon határa a tartalmazott poligonok határainak (Rings) sorozatából áll.

Dimensionally Extended Nine-Intersection Modell (DE-9IM)

Legyen a egy geometriai objektum, amelyre $I(a), B(a), E(a)$ reprezentálja az a objektum belső határát (interior), határát (boundary) és külső határát (ex-

2.3. táblázat. Az intersection mátrix a DE-9IM modell szerint

	Interior	Boundary	Exterior
Interior	$dim(I(a) \cap I(b))$	$dim(I(a) \cap B(b))$	$dim(I(a) \cap E(b))$
Boundary	$dim(B(a) \cap I(b))$	$dim(B(a) \cap B(b))$	$dim(B(a) \cap E(b))$
Exterior	$dim(E(a) \cap I(b))$	$dim(E(a) \cap B(b))$	$dim(E(a) \cap E(b))$



2.18. ábra. Átfedő a, b poligonok

terior). Legyen $dim(x)$ az x-ben lévő geometriai objektumok dimenzióinak maximuma (-1, 0, 1, 2), ahol -1 reprezentálja a $dim(\emptyset)$ értéket.

Bármely két geometriai objektum metszete $I(a), B(a), E(a)$ egy kevert dimenziójú geometriai objektumsorozat lesz (x). Például két poligon határának (boundary) a metszete eredményezhet egy pontot és egy vonalat. A 2.3. táblázat a DE-9MI modell szerinti metszetmátrixot mutatja két geometriai objektum (a, b) között. A metszet dimezióinak kiszámítása sokféle képet mutathat. Vonal-terület (Line-Area) esetében a lehetséges értékek interior-interior reláció esetén {-1, 1}. Terület-terület (Area-Area) esetében a lehetséges interior-interior értékek {-1, 2}.

2.4. táblázat. Az intersection mátrix a, b átfedő poligonokra

	Interior	Boundary	Exterior
Interior	2	1	2
Boundary	1	0	1
Exterior	2	1	2

Alkossuk meg az intersection mátrixot (2.4. táblázat) a 2.18. ábrán látható két valóságos poligonra. Ha a két a,b objektumra vonatkozó térbeli reláció eredménye az intersection mátrix valamely értékével egyezik, akkor az állítás értéke igaz (TRUE). Emlékezzünk ezen alfejezet elején említett megállapításra, hogy a térbeli operátorok boolean típusú visszatérési értékű operátorok. Az intersection mátrix celláiban található kilenc lehetséges minta képezi a vizsgálódás alapjait. A lehetséges mintázatok p {T, F, *, 0, 1, 2} értékűek lehetnek. Jelentésük a következő:

$$p = T \Rightarrow dim(x) \in \{0,1,2\}$$

azaz $x \neq \emptyset$, és ahol x a két objektum metszetét jelenti.

$$p = T \Rightarrow dim(x) = -1$$

azaz $x \neq \emptyset$

$$p = T \Rightarrow dim(x) \in \{-1,0,1,2\}$$

Ez az eset érdektelen.

$$p = T \Rightarrow dim(x) = 0$$

$$p = T \Rightarrow dim(x) = 1$$

$$p = T \Rightarrow dim(x) = 2$$

A mintázatok a mátrix formánál jobban kezelhető formában is reprezentálhatók, ha a mátrix elemeit egy 9 elemű listaként fogjuk fel, ahol a lista elemei sorfolytonosan lettek leszármaztatva a mátrix soraiból. A következő példában egy kódrészleten mutatjuk be ezt a listás reprezentációt az átfedés (overlap) funkción keresztül:

```
char * overlapMatrix = "T*T***T**"
Geometry* a,b
Boolean b = a->Relate(b, overlapMatrix)
```

Nevesített térbeli relációk

A térbeli relációk legtöbbször néhány kiemelt, névvel elnevezett funkciót takarnak. A felhasználók számára emészthető függvényneveket használunk, ráadásul nem várható el, hogy mindezek SQL-szabvány szerinti szintaktikájával tisztában legyenek. Az egyszerűbb használat érdekében beszédes függvénynevek alkalmazásával oldjuk meg ezt a problémát.

Equals Legyen adva két geometriai objektum, a és b.

$$a.Equals(b) \Leftrightarrow a \subseteq b \wedge b \subseteq a$$

DE-9IM formában kifejezve:

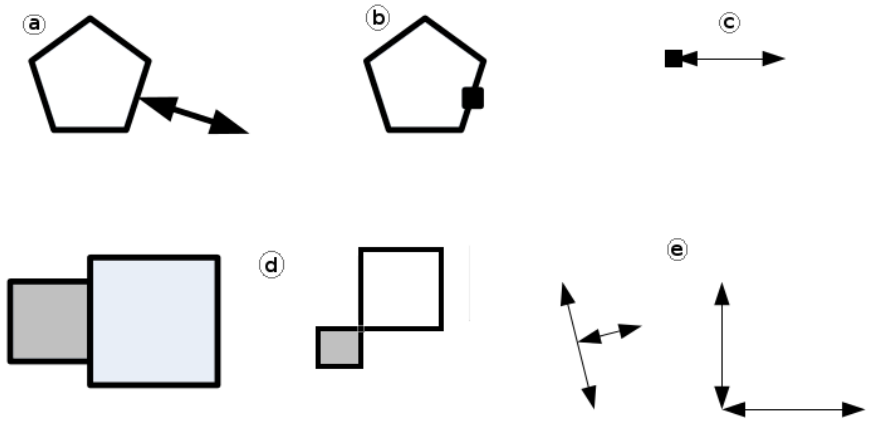
$$\begin{aligned} & a.Equals(b) \Leftrightarrow [\\ & (I(a) \cap I(b) \neq \emptyset) \wedge \\ & (I(a) \cap B(b) = \emptyset) \wedge \\ & (I(a) \cap E(b) = \emptyset) \wedge \\ & (B(a) \cap I(b) = \emptyset) \wedge \\ & (B(a) \cap B(b) \neq \emptyset) \wedge \\ & (B(a) \cap E(b) = \emptyset) \wedge \\ & (E(a) \cap I(b) = \emptyset) \wedge \\ & (E(a) \cap B(b) = \emptyset) \wedge \\ & (E(a) \cap E(b) \neq \emptyset)] \\ \Leftrightarrow & a.Relate(b, "TFFFTFFFT") \end{aligned}$$

Disjoin Legyen adva két geometriai objektum, a és b.

$$a.Disjoin(b) \Leftrightarrow a \cap b \neq \emptyset$$

DE-9IM formában kifejezve:

$$\begin{aligned} & a.Disjoin(b) \Leftrightarrow [\\ & (I(a) \cap I(b) = \emptyset) \wedge \\ & (I(a) \cap B(b) = \emptyset) \wedge \\ & (B(a) \cap I(b) = \emptyset) \wedge \\ & (B(a) \cap B(b) \neq \emptyset)] \\ \Leftrightarrow & a.Relate(b, "FF*FF****") \end{aligned}$$



2.19. ábra. Példák érintő geometriai objektumokra. a – Polygon/LineString; b – Polygon/Point; c – LineString/Point; d – Polygon/Polygon; e – LineString/LineString

Touches Legyen adva két geometriai objektum, a és b, amelyek érintik egymást. Érinthetik egymást a következő párok: poligon – poligon, vonal (line) – vonal, vonal – poligon, pont – poligon és pont – vonal (2.19. ábra). Pont – pont érintés nem létezik.

$$a.Touch(b) \Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (a \cap b) \neq \emptyset$$

DE-9IM formában kifejezve:

$$\begin{gathered} a.Touch(b) \Leftrightarrow [\\ (I(a) \cap I(b) = \emptyset) \wedge \\ (B(a) \cap I(b) \neq \emptyset) \vee \\ (I(a) \cap B(b) \neq \emptyset) \vee \\ (B(a) \cap B(b) \neq \emptyset)] \\ \Leftrightarrow [a.Relate(b,"FT*******") \vee \\ a.Relate(b,"F**T*****") \vee \\ a.Relate(b,"F***T****")] \end{gathered}$$

Crosses A keresztezés kapcsolat két geometriai objektum között csak Point/Line, Point/Area, Line/Line és Line/Area típusok között jöhet létre:

$$a.Cross(b) \Leftrightarrow [I(a) \cap I(b) \neq \emptyset \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)]$$

DE-9IM formában kifejezve:

$$Case\ a \in P,\ b \in L\ or$$

$$Case\ a \in P,\ b \in A\ or$$

$$Case\ a \in L,\ b \in A:$$

$$a.Cross(b) \Leftrightarrow [I(a) \cap I(b) \neq \emptyset \wedge I(a) \cap E(b) \neq \emptyset]$$

$$a.Relate(b, "T*T******")$$

$$Case\ a \in L,\ b \in L:$$

$$a.Cross(b) \Leftrightarrow dim(I(a) \cap I(b)) = 0$$

$$\Leftrightarrow a.Relate(b, "0********")$$

Within A within relációt következőképpen definiáljuk:

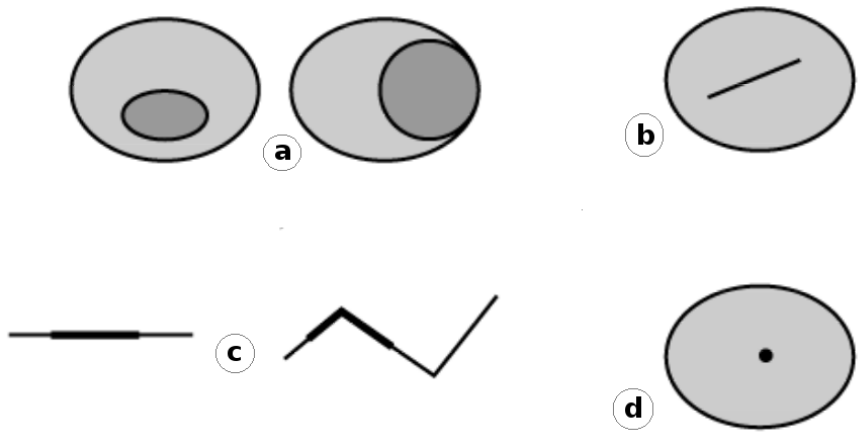
$$a.Within(b) \Leftrightarrow (a \cap b = a) \wedge (I(a) \cap E(b) = \emptyset)$$

DE-9IM formában kifejezve:

$$a.Within(b) \Leftrightarrow [I(a) \cap I(b) \neq \emptyset \wedge I(a) \cap E(b) = \emptyset \wedge B(a) \cap E(b) = \emptyset]$$

$$\Leftrightarrow a.Relate(b, "T*F**F***")$$

A 2.20. ábra példákat mutat a within relációra.



2.20. ábra. Példák *Within* (benne van) geometriai relációra. a – Polygon/Polygon; b – Polygon/LineString; c – LineString/LineString; d – Polygon/Point

Overlaps Az overlap funkciót Area/Area, Line/Line és Point/Point relációkra alkalmazzuk. A 2.21. ábra példákat mutat az overlap relációra. Az overlap-et a következőképpen definiáljuk:

$$a.Overlaps(b) \Leftrightarrow (dim(I(a)) = dim(I(b)) = dim(I(a) \cap I(b)))$$

$$\wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

DE-9IM formában kifejezve:

$$Case\ a \in P,\ b \in P\ or\ Caseqa \in A,\ b \in A:$$

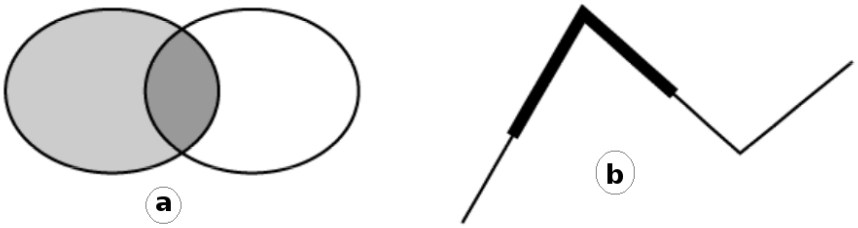
$$a.Overlaps(b) \Leftrightarrow (I(a) \cap I(b) \neq \emptyset)\ land$$

$$(I(a) \cap E(b) \neq \emptyset) \wedge$$

$$(E(a) \cap I(b) \neq \emptyset)$$

$$\Leftrightarrow a.Relate(b, "T*T***T**")$$

$$Case\ a \in L,\ b \in L:$$



2.21. ábra. Példák *Overlap* (átfed) geometriai relációra. a – Polygon/LineString; b – LineString/LineString

$$a.Overlaps(b) \Leftrightarrow (dim(I(a) \cap I(b) = 1) \wedge (I(a) \cap E(b) \neq \emptyset \wedge (E(a) \cap I(b) \neq \emptyset$$
$$\Leftrightarrow a.Relate(b, "1 * T * * * T * *")$$

Contains

$$a.Contains(b) \Leftrightarrow b.Within(a)$$

Intersects

$$a.Intersects(b) \Leftrightarrow !a.Disjoin(b)$$

Metódusok

- Equals(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg egyenlő Geom2-vel.
- Disjoin(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg szétválasztott Geom2-től.
- Intersects(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg metszi Geom2-t.
- Touches(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg érinti Geom2-t.
- Crosses(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg keresztezi Geom2-t.
- Within(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg benne van Geom2-ben.

- Contains(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg tartalmazza Geom2-t.
- Overlaps(Geom2:Geometry): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg átfedi Geom2-t.
- Relate(Geom2:Geometry, intersectionPatternMatrix:String): Integer. Visszatérési értéke 1 (TRUE), ha az adott geometriai objektum térbelileg kapcsolódik Geom2-höz, vizsgálván a geometriai objektumok belső határának, határának és külső határának metszetét.

2.2.3. Annotation text (feliratok)

Az OGC szabvány számos előírást tartalmaz a textek, feliratok paramétereinek tárolására, kezelésére. Valamennyi szoftvergyártó megoldja a textek tárolásával, pozicionálásával kapcsolatos problémákat, de ezek ma még kevéssé szabványosak. Az OGC szabvány erre a szabványosításra tesz javaslatot. Mivel azonban a kérdéskör kevéssé kapcsolódik könyvünk témájához, mármint a topologikus adatszerkezetekhez, ezért a kérdéskört nagyvonalúan átugorjuk. Részletes információk találhatók a OGC weboldalán: *http://www.opengeospatial.org/*.

2.3. SQL-implementációk

Vizsgáljunk meg, hogy az előbbi fejezetekben tárgyalt fogalmak alapján miként alkothatunk meg egy olyan adatbázist, amely egységbe foglalva kezeli a geometriát, a koordináta-rendszert és a leíró adatokat.

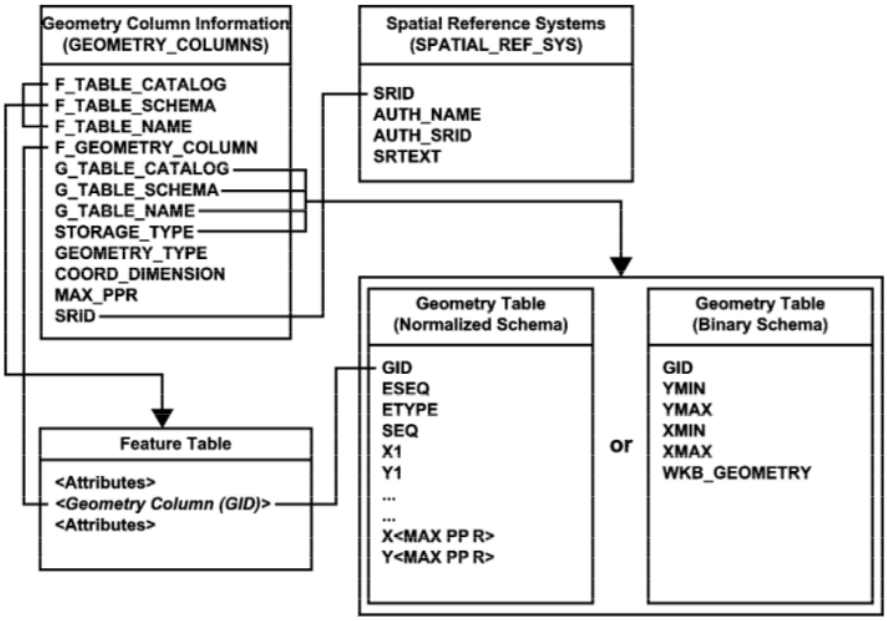
2.3.1. Egy SQL-implementáció meglévő típusok alapján

A 2.22. ábrán látható adatszerkezet egy olyan séma, amely a geometry és a spatial reference system kapcsolatrendszerének leírására szolgál előre definiált típusok alapján. A *Feature Table*-ben kapcsolódnak össze a geometria és az attribútumok. A mezők jelentésének magyarázata a 2.5. táblázatban látható.

Feature table és geometry column

Amint az a 2.22. ábrán látható, a *Feature Table*t és a *Geometry Table*t a GEOMETRY_COLUMNS kapcsolja össze. A GEOMETRY_COLUMNS a következőkből áll:

- feature table azonosító



2.22. ábra. Egy adatbázisséma a feature table leíráshoz, amely tartalmazza a geometry és a spatial reference táblákat

2.5. táblázat. Magyarázat a feature table-t leíró adatbázissémához

Táblanevek	**Leírás**
GEOMETRY_COLUMNS	Leírja a rendelkezésre álló feature table-ket és geometriai tulajdonságaikat
SPATIAL_REF_SYS	Leírja a koordináta-rendszert és a geometriai transzformációkat
FEATURE_TABLE	Tárolja az objektumok gyűjteményét, ahol az oszlopok az attribútumokat jelentik, a sorok pedig az egyes objektumokat reprezentálják
GEOMETRY_TABLE	A geometriát tárolja, amely lehet egy SQL numerikus vagy bináris típus

- a Geometry Column neve
- a térbeli referencia-koordinátarendszer azonosítója (SRID)
- a Geometry Column típusa
- a Geometry Column dimenziója
- a Geometry Table azonosítója, amely tárolja a geometriai objektumot
- a Geometry Table navigációjához szükséges információ

Spatial reference system

Minden Geometry Columnra és minden egyes geometriai entitásra egy és csak egy térbeli referencia-koordinátarendszer vonatkozhat. Ezt az információt tároljuk is (lásd 2.22. ábra). A SPATIAL_REF_SYS táblában nemcsak numerikus kódot (SRID), hanem szöveges leírást is tárolunk (SRTEXT) a referenciarendszerről, mint pl. HD72 EOV, vagy Longitude/latitude UTM zone 34. Ez a leírási mód összhangban van a WKT-ben (100. oldal) történő referenciarendszer-leírással.

Feature table

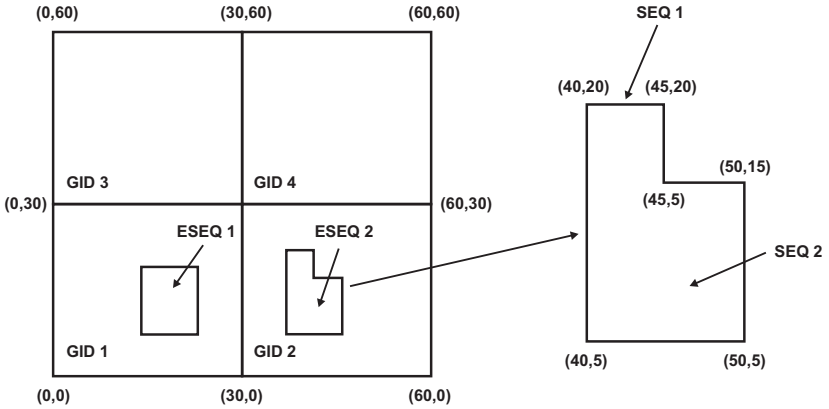
A feature a valós dolgok egy absztrakciója. A Feature Table egy sora a valós dolgok egy entitását reprezentálja. Leíró adatokat is tartalmaz az adott objektumról, valamint annak geometriáját leíró táblára (GEOMETRY_TABLE) is hivatkozik.

Geometry table

A normalizált geometry sémában a geometriai objektumok koordinátáit tároljuk az SQL-nek megfelelő numerikus típusokkal, ahogy azt a 2.23. ábrán láthatjuk. A geometriai objektumokat a GID kulcs révén azonosítjuk. Az objektumot felépítő geometriai elemek sorrendjét az ESEQ (element sequence), típusát az ETYPE, sorrendjét a SEQ (sequence number) adja meg.

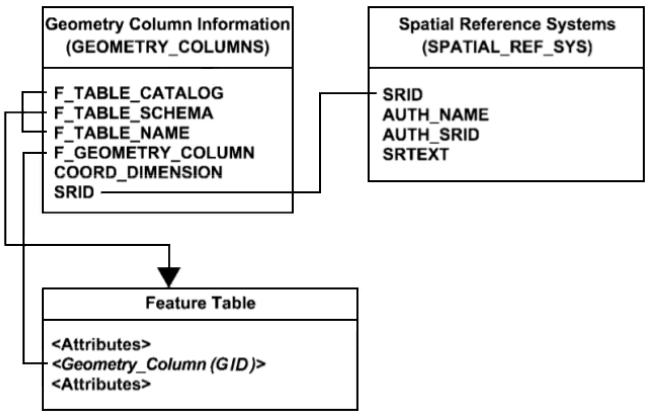
Ezek alapján a geometriai objektumok normalizált leírásának reprezentációját a következőképpen definiáljuk:

- ETYPE paraméter megadja a geometria típusát
- ESEQ paraméter azonosítja a geometriai objektumok elemeit, amelyek lehetnek többszörös elemek is



GID 1	ESEQ	ETYPE	SEQ	X0	Y0	X1	Y1	X2	Y2	X3	Y3	X4	Y4
1	1	3	1	0	0	0	30	30	30	30	0	0	0
1	2	3	1	10	10	10	20	20	20	20	10	10	10
2	1	3	1	30	0	30	30	60	30	60	30	30	0
2	2	3	1	40	5	40	20	45	20	45	15	50	0
2	2	3	1	50	15	50	5	40	5	Nil	Nil	Nil	0
3	1	3	1	0	30	0	60	30	60	30	30	0	30
4	1	3	1	30	30	30	60	60	60	60	30	30	30

2.23. ábra. Egy geometry table példa poligongeometriára



2.24. ábra. A *Geometry Type* sémája

- az elemek több részből is állhatnak, amelyeket SEQ paraméter azonosít
- a poligonok tartalmazhatnak lyukakat is, amint azt a geometry object modellben láthattuk (19. oldal)
- PolygonRing-ek különböző részekből történő összeállításakor ezeket be kell zárni. A részek egy rendezett listából származnak, amelyeknek sorrendjét a SEQ paraméter adja meg
- a nem haszált koordinátapárok értékeit *Nil*-re kell állítani, mivel ezen a módon tudjuk megállapítani a koordinátalista végét
- amikor egy geometriai objektum egy következő sorban folytatódik, akkor az adott sor utolsó pontja a következő sor első pontja kell legyen
- egy geometriai objektumban nincs felső határa az elemek számának, illetve a sorok számának egy elemre vonatkozóan

2.3.2. Egy SQL-implementáció geometriatípus alapján

A feature table, geometry és a spatial reference system együttest leíró adatbázisséma bevezetésével megjelenik az SQL-ben as *Geometry type*, a geometria típusú adat. A 2.24. ábrán látható a geometry type adattípus leírására kidolgozott séma. A táblák magyarázata a következő:

- A GEOMETRY_COLUMNS leírja a rendelkezésre álló feature táblákat és azok geometriai tulajdonságait.
- A SPATIAL_REF_SYS tábla leírja a geometry referencia-koordinátarendszerét és a szükséges transzformációkat.
- A feature table az objektumok gyűjteménye. A feature táblák sorai reprezentálják az egyes objektumokat, az oszlopai az attribútumokat. A feature table egy attribútuma a *Geometry Type* is.

Feature table és geometry column

A *feature table* és a *geometry columns* táblák a *GEOMETRY_COLUMNS* táblán keresztül vannak összekapcsolva azáltal, hogy minden Geometry columnnak van egy neve a *GEOMETRY_COLUMNS* táblában. A geometry columnshoz tárolt adatok a következők:

- a feature table azonosítója
- a geometry column neve
- a geometry column térbeli referenciarendszerének azonosítója (SRID)
- a geometry column dimenziója

Vegyük észre, hogy a *GEOMETRY_COLUMNS* táblában az új SQL típus, a Geometry Type, részhalmaza a meglévő SQL típusok alapján kreált *GEOMETRY_COLUMNS* tábla típusainak.

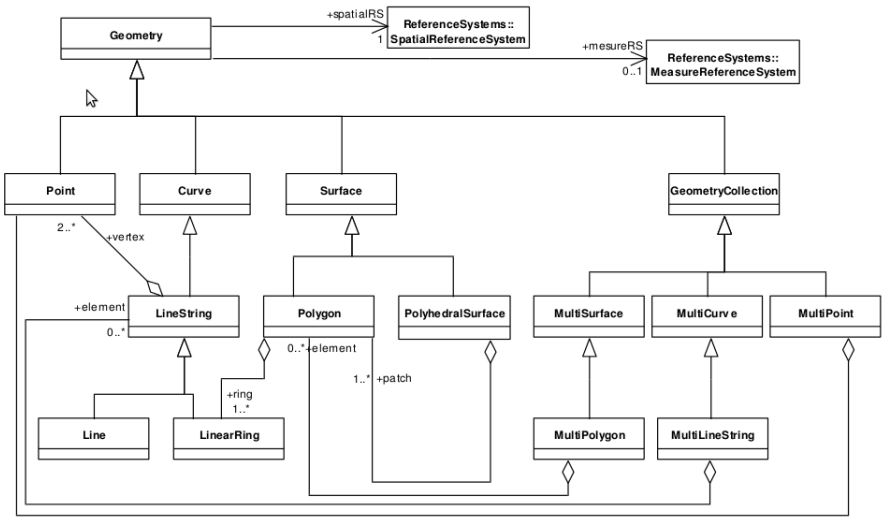
Spatial reference system

Minden geometry columnhoz tartozik egy térbeli referencia-koordinátarendszer, amely minden geometriai objektumokhoz hozzá van rendelve. Ennek révén tudjuk értelmezni a koordináták numerikus értékeit.

A *SPATIAL_REF_SYS* tábla tárolja az összes referenciarendszert, amely az adatbázisban elérhető. Az egyes referenciarendszerekre a spatial reference system azonosítóval (SRID) lehet hivatkozni. A referenciarendszer elnevezése (AUTH_NAME), kódja (AUTH_SRID) és WKT leírása (SRTEXT) is megtalálható. Az SRID egy egész szám, amely az adatbázison belül egyedi.

Feature table

A feature table a valóság reprezentációja. A tárolt objektumok attribútumait a tábla oszlopai, az egyes objektumokat a sorai reprezentálják. Az adott objektum geometriáját a geometry column tárolja, amely *Geometry type* típusú.



2.25. ábra. SQL-geometriatípus hierarchiája. A fő típus a Geometry és altípusok a Point, Curve és a Surface, valamint a Geometry collection. A MultiPoint, a MultiCurve és a MultiSurface a Geometry collection altípusai

User defined type

A user defined type (UDT) révén kiterjeszthető az SQL típusrendszerre, vagyis saját típusok definiálásra is van lehetőség. Így UDT típusú oszlopokat lehet létrehozni, és a rekordokat UDT típusú adatokkal feltölteni.

SQL-geometriatípus hierarchiája

A 2.25. ábrán az SQL geometry type hierarchiája látható. Az SQL-függvények és eljárások ennek a típusnak a figyelembe vételével lettek kidolgozva, ahogyan azt az OGC bevezető részében a Geometry Object Modelben leírtuk (19. oldal). Természetesen ezek WKT reprezentációja is adott.

2.4. Összegzés

Az OGC előírásait ma már a legtöbb GIS szoftver teljesíti, hasonlóan a relációs adatbázis-kezelő rendszerekhez. A Postgres ebben élen jár, de a Microsoft SQL servere is erre épül. Érdekes módon az ORACLE külön utakon jár, bár nyilvánvalóan hasonló fogalmakat használ a térbeliség leírásához.

Mint láthattuk, az OGC matematikailag korrekt módon alapozza meg a térbeliség kezelését. Világosan definiálja a topologikus minőséget. A térbeli objektumok valamennyi alkotóeleme az objektumhoz, és csak ahhoz tartozik, ily módon a szomszédos poligonok közös vertexei mindkét poligonban benne lesznek, ami mindenképpen redundánsnak mondható. Hasonlóan igaz ez a megállapítás a Curve, és a LineString objektumokra is, mivel az azonos nyomvonalon futó görbék redundánsan tartalmazzák a vertexeiket.

Az is világosan látható, hogy a térbeliség függvényei is ehhez a logikához illeszkednek. A szomszédos, vagy egymáson futó térbeli objektumok mit sem tudnak arról, hogy ők kiknek a szomszédai, mely objektumokkal vannak közös vertexeik. Ettől függetlenül e modell alapján is lehet (sőt kötelező) hézag- és átfedésmentes poligontopológiát előállítani szomszédos poligonokra. A szomszédság megállapítására a térbeli függvények nyújtanak segítséget.

Egy későbbi fejezetben látni fogunk olyan adatstruktúrákat, amelyek ezt a problémát is megoldják (63. oldal), vagyis szomszédos poligonok és egymáson futó LineStringek esetében nem tárolják redundánsan az azonos térbeli pozícióban lévő vertexeiket.

3

Ipari szoftverek megoldásai a térbeliség kezelésére

Nem fogjuk áttekinteni valamennyi, a piacon fellelhető adatbázis-kezelő rendszer megoldásait, de azért a legjellemzőbbet mindenképpen. Az előző fejezetben bemutattuk az OGC szabványt, amely a ma létező piaci szoftverek elvi hátterét adja. Ebben a fejezetben egy piacvezető, nyílt forráskódú adatbázis-kezelő, a Postgres/PostGIS térbeliséget kezelő megoldásait fogjuk áttekinteni. A Microsoft SQL server is az OGC szabvány szerint működik, megoldásai nagyon hasonlóak a bemutatandókhoz, ezért erre külön nem térünk ki. A térbeliséget más, nagy nevű adatbázis-kezelő rendszerek is kezelik, mint például az ORACLE, de saját elvi megoldásokat követnek, amelyek ugyan nem kevésbé színvonalasak, de az OGC előírásait nem mindenben követik. Terjedelmi korlátok okán ezek áttekintésére nem teszünk kísérletet.

A PostGIS a PostgreSQL adatbázis-kezelő rendszernek egy kiegészítő modulja. Vektoros és raszteres adatokat kezel. Itt és most csak vektoros adatok kezelésével foglalkozunk. Akit részletesebben érdekel a PostGIS, és a hozzá kapcsolódó térbeli technológia, az bővebb ismertetőt talál a http://tarsadalominformatika.elte.hu/tananyagok/gdal/index.html oldalon.

3.1. PostGIS térbeli objektum típusai

OGC szabvány szerinti Simple Feature for SQL objektumokat implementálta és bővítette ki a PostGIS. Kétféleképpen reprezentálhatók a geometriai objektumok:

- WKT, azaz Well-known-Text formátum, amely szöveges formátum (lásd a *Formátumleírások* című fejezetet, 99. oldal).
- WKB, azaz Well-known-Binary formátum, amely bináris formátum.

PostGIS-ben a következő WKT térbeli objektum típusok vannak:

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

A fenti Geometry objektumok kétdimenziós síkban vannak reprezentálva, így bizonyos számítások, mint például a távolság, hossz vagy terület, nem lesznek pontosak. Ennek oka, hogy ezek derékszögű koordinátarendszerre vonatkoznak, és a legtöbb geometry típuson alapuló térbeli függvény Descartes-algebra alapján dolgozik. Márpedig ha a Föld felületén mozgunk, akkor gömbi geometriát kell használni. (Persze nagy méretarányú térképek esetén (1:500-1:4000 méretarányig) a kétféle geometria közötti különbségből származó hiba elhanyagolható.)

A fent említettek miatt lett bevezetve a Geography típus, amely nem kétdimenziós síkra vonatkozik, hanem gömbi geometriát használ. Emiatt persze a Geography típuson alapuló térbeli függvények kiszámítása bonyolultabb és hosszadalmasabb, így csak a térbeli függvények egy részhalmazára lettek implementálva. Az egyik legnagyobb hátránya a Geography típusnak, hogy csak WGS84 (EPSG:4326) vetületi rendszerben levő adatokat képes kezelni.

3.2. A PostGIS felépítése

PostGIS két táblát használ metaadatok tárolására:

- SPATIAL_REF_SYS: vetületi rendszerek tárolásáért felelős.
- GEOMETRY_COLUMNS: ez a tábla az adatbázisban levő geometry adattípust felhasználó táblákról tárol információt.

3.2.1. Spatial_Ref_sys tábla felépítése

A több mint 3000 vetületi rendszert tartalmazó tábla lehetőséget ad vetületi rendszerek közötti transzformációra. Ezenkívül definiálhatunk saját vetületi rendszert PROJ4 formátumban. Erről több leírás található: https://trac.osgeo.org/proj/, valamint vetületi rendszerekre vonatkozó információ a http://spatialreference.org/ címen találhatunk. A legelterjedtebb vetületi rendszerek WGS84, ETRS89 és EOV/HD72 közül, amiket a PostGIS alapértelmezetten támogat. A Spatial_Ref_sys tábla felépítése a következő:

- SRID: Egész szám, amely egyértelműen meghatározza az adott vetületi rendszert (elsődleges kulcs)
- AUTH_NAME: Szabvány neve, amelyben definiálva lett. pl. EPSG
- AUTH_SRID: Szabvány által megadott egyedi azonosítója. pl.: EPSG:4326
- SRTEXT: Vetületi rendszer leírása WKT formátumban.
- PROJ4TEXT: PROJ4 szerinti leírása.

3.2.2. Geometry_columns tábla felépítése

A PostGIS 2.0 előtti verziókban Geometry_Columns nézetként volt definiálva, amely inkonzisztenciát okozott gyakorlatlan felhasználók számára, mivel egy tábla módosítás vagy törlése esetén, nem voltak szinkronban a Geometry_columns és geometriát felhasználó tábla szerkezete. Szerkezete a következő:

- F_TABLE_SCHEMA, F_TABLE_NAME: Melyik sémában található és mi az adott tábla neve.
- F_GEOMETRY_COLUMN: Geometriaoszlop neve.
- COORD_DIMENSION: Térbeli adatok dimenziószáma.
- SRID: Térbeli adatok melyik vetületi rendszerben vannak.
- TYPE: Térbeli objektumok típusa: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION.

Spatial tábla létrehozása

Két esetet fogunk bemutatni: új tábla létrehozása, és meglévő táblához geometry column hozzáadása.

- Új tábla létrehozása:

```
create table proba(id serial primary key,
         location geometry(POINT, 4326));
```

- Létező táblához geometry column hozzáadása:

```
alter table proba add column
         bbox geometry(POLYGON, 23700);
```

Adatbetöltés

Az *Shp2Pgsql* betöltő ESRI shapefile formátumú fájlokat tölt be adatbázisba. Paraméterei a következőek:

- (c|a|d|p)
- c: Betöltés előtt új tábla létrehozása, amely megegyezik a Shapefile sémájával.
- a: Új adatok hozzáadása meglevő táblastruktúrához (azt feltételezi, hogy shapefile sémája megegyezik tábla szerkezetével)
- d: Meglevő nevű tábla eldobása, ezután létrehoz új táblát.
- p: végrehajtható SQL kód a kimenetben.
- ?: Help parancsok megjelenítése.
- D: PostgreSQL Dump formátumának használata, amely csak a|c|d kapcsolókkal működik.
- s: forrásállomány vetületi rendszerének megadása.
- t: cél adat dimenzió számának meghatározása.

- i: Integer típus legyen 32bites függetlenül DBF fejléc definíciójától.
- k: Shapefile attribútumneveinek megőrzése (sok esetben ez problémát okozhat, ezért ha lehet, ne használjuk ezt a kapcsolót)
- I: GIST-index használata
- S: Egyszerű geometriák használata.
- w: WKT-formátum legyen a kimenetben.
- e: Tranzakció használatának kikapcsolása, azaz minden egyes objektum betöltését hajtsa végre azonnal.
- W: Karakterkódolásának megadása.
- N: Null geometria, azaz olyan objektumok, amelyek nem tartalmaznak geometriát, mit tegyen: insert (beszúrás), skip (kihagyás), abort (megszakítás)
- n: csak a DBF fájl tartalmának a betöltése.
- G: Geography típus használata Geometry helyett.
- T: tablespace név megadása.
- X: Index tábla tablespace-nek megadása.

Lássunk két példát:

1. ```
   shp2pgsql -d -D -s 4326 -I countries.shp
            public.countries > countries.sql
   ```

2. vagy betölthetjük egyenesen az adatbázisba „countries.sql" helyett a

   ```
   psql -U felhasználó_név -h host_név -d spatial_data
   ```

   paranccsal.

A *Pgsql2Shp* alkalmazás ESRI shapefile formátumban menti ki az adatokat. Kapcsolói a következőek:
   ```

- f: Kimeneti fájl nevének megadása.
- h: host név meghatározása.
- p: port szám meghatározása.
- u: adatbázis-kezelő felhasználó neve.
- P: felhasználóhoz tartozó jelszó.
- g: geometriaoszlop nevének megadása (mivel egy adott táblához több geometria típusú oszlop is tartozhat.
- b: bináris kurzor használata az adatokon végig olvasásához (jellemzője: gyors)
- r: Nyers mód, azaz struktúra megtartásával képezi le az adatokat Shapefile-ba.

Ezenkívül kötelező meghatározni az adatbázis nevét, séma és tábla nevét, amit le kívánunk kérdezni, összefoglalva tehát:

```
pgsql2shp [kapcsolók] [adatbázis_név] [schema][tábla_név]
```

amely egy példában a következő:

```
pgsql2shp -f countries.shp -h localhost -p 5432
-u mici -p mez -g hatar -b spatial_data public.countries
```

3.2.3. PostGIS függvények

Alapértelmezetten WKT típusú objektumokra nyújtanak támogatást, valamint kiterjesztett, EWKT típusú objektumokra.

A WKT-re vonatkozóan négy alcsoportja van a függvényeknek:

- Kezelési: ilyen a geometriaoszlop hozzáadás, SRID lekérdezés.
- Kapcsolati, viszony: tartalmazási viszonyokat támogató függvények.
- Feldolgozási: ide tartozik a centroid, befoglaló téglalap, buffer zóna, terület, súlypont lekérdezése és létrehozása.
- Tartalmi: Adott geometriára vonatkozó információk lekérését támogató függvények.
- Konstruktorok: Geometria létrehozási függvények.

Kezelési függvények

- AddGeometryColumn(séma, tábla, oszlop, srid, típus, dimenzió) Geometriaoszlop hozzáadása meglévő táblához.
- DropGeometryColumn(séma, tábla, oszlop) Meglévő geometriaoszlop eltávolítása.
- ST_SetSRID(geometry, srid) Adott geometriára vonatkozó vetületi rendszer beállítása.

Kapcsolati függvények

Távolsági függvények

- ST_Distance(geometry, geometry) returns float: Két geometriaobjektum közti távolság meghatározása.
- ST_DWithin(geometry, geometry, float) returns boolean: Két geometria közti megadott távolság között van-e.

Geometriai függvények

- ST_Disjoint(geometry, geometry) returns boolean: Két geometria befoglaló téglalapjai geometria viszonyban vannak-e egymással.
- ST_Intersects(geometry, geometry) returns boolean: Két gometria metszi-e egymást.
- ST_Touches(geometry, geometry) returns boolean: Két geometria érinti-e egymást.
- ST_Crosses(geometry, geometry) returns boolean: Két keresztezi-e egymást.
- ST_Within(geometryA, geometryB) returns boolean: geometryA benne van-e geometryB-ben.
- ST_Overlaps(geometry, geometry) returns boolean: Geometriák fedike egymást.
- ST_Contains(geometryA, geometryB) returns boolean: geometryB része-e geometryA-nak.

- ST_Covers(geomtryA, geometryB) returns boolean: geometryB valódi része-e geometryA-nak.
- ST_CoveredBy(geometryA, geometryB) returns boolean: geometryA valódi része-e geometryB-nek.
- ST_Relate(geometry, geometry, kapcsolati-mátrix) returns boolean: Két geometriának valamilyen geometriai viszonya van-e egymással mátrix alapján.
- ST_Relate(geoemtry,geometry) returns boolean: Két geometriának van-e valamilyen viszonya egymással.

Feldolgozási függvények

- ST_Centroid(geometry) returns POINT: adott geometria súlypontjának lekérdezése.
- ST_Area(geometry polygon) returns float: adott geometria területének lekérdezése.
- ST_Length(geometry linestring) returns float: adott geometria hosszának lekérdezése.
- ST_PointOnSurface(geometry) returns POINT: egy tetszőleges pont lekérdezése, amely rajta vagy benne van az adott geometrián.
- ST_Boundary(geometry) returns LineString: zárt vonalakból álló geometry, amely az adott objektum határvonala.
- ST_Buffer(geometry, float sugár-hossza, integer quad_seq-számossága) returns geometry.
- ST_Buffer(geometry, float sugár-hossza, string paraméter) returns geometry: paraméterek szóközzel elválasztva felsorolásban:
  - quad_segs=#: hasonló mint az előző függvény.
  - endcap=round|flat|square: round-lekerekített, flat-egyszerű, square-szögletes buffer vég.
  - join=round|mitre|bevel: összekötő-szakaszoknál milyen legyen a buffer lekerekített|szögletes|levágott.

- ST_ConvexHull(geometry) returns geometry: adott geometriára vonatkozó befoglaló téglalapot adja vissza.
- ST_SymDifference(geometryA, geometryB) returns geometry: Két geometria különbsége.
- ST_Union(geometry, geometry) returns geometry: Két geometria uniója.
- ST_Union(geometries) returns geometry: geometriai objektumok uniója.
- ST_MemUnion(geometries) returns geometry: hasonló mint ST_Union csak memóriafelhasználás szempontjából gazdaságosabb.

Tartalmi függvények

- ST_AsText(geometry) returns WKT: WKT formátumban adja vissza az adott geometriát.
- ST_AsBinary(geometry) returns WKB: WKB formátumban adja vissza az adott geometriát.
- ST_SRID(geometry) returns integer: Adott geometriára vonatkozó vetületi rendszer azonosítóját adja.
- ST_Dimension(geometry) returns integer: Adott geometriára vonatkozó dimenziószámot adja.
- ST_Envelope(geometry) returns geometry: Adott geometriára vonatkozó befoglaló téglalapot adja.
- ST_IsSimple(geometry) returns boolean: Egyszerű-e az adott geometria.
- ST_IsEmpty(geometry) returns boolean: Adott geometria tartalmaz-e pontokat.
- ST_IsClosed(geometry) returns boolean: Adott geometria zárt-e.
- ST_IsRing(geometry) returns boolean: Adott görbe zárt-e.

- ST_NumGeometries(geometry) return integer: Adott geometry collection-nek számosságát adja.
- ST_GEometryN(geometry, integer sorszám) returns geometry: Geometry collection-nek a megadott számosságú geometriáját adja.
- GeometryType(geometry) returns string: Adott geometria típusát adja.

Konstruktorok

- ST_GeomFromText(WKT text, srid) returns geometry: Geometriaobjektum létrehozása WKT-vel.
- ST_GeomFromWKB(bytearray WKB, srid) returns geometry: Geometriaobjektum létrehozása WKB-val.

Kiegészítő függvények

- DropGeometryTable(séma, tábla): geometriaoszlop törlése.
- UpdateGeometrySRID(séma, tábla, oszlop, srid): geometriaoszlop vetületi rendszerének felülírása.
- update_geometry_stats(tábla, oszlop): statisztikai adatok frissítése az adott geometriaoszlopról, QUERY tervező számára fontos művelet. Ezek után javasolt a VACUUM ANALYZE-t lefuttatni.

Operátorok

- $A = B$: Egyenlőségi vizsgálat, azonban csak a befoglaló téglalapot vizsgálja, előnye, hogy gyors.
- $A\& < B$: A befoglaló téglalapja fedésben van B befoglaló téglalapjával vagy a bal oldalán van-e.
- $A\& > B$: Hasonló mint $A\& < B$ csak a jobb oldalon van-e A befoglaló téglalapja.
- $A << B$: A szigorúan B bal oldalán van.
- $A >> B$: A szigorúan B jobb oldalán van.
- $A\& < |B$: A befoglaló téglalapja átfedésben van B befoglaló téglalapjával vagy alatta van-e.

- $A|\&>B$: Hasonló, mint $A\&<|B$ csak B-hez képes felette van-e.
- $A<<|B$: A befoglaló téglalapja szigorúan B alatt van.
- $A|>>B$: A befoglaló téglalapja szigorúan B felett van.
- $A=B$: Geometriailag megegyezik-e A, B-vel pontonként (nem csak befoglaló téglalap vizsgálat).
- $A@B$: B befoglaló téglalapja tartalmazza-e A befoglaló téglalapját.
- $A\ B$: A befoglaló téglalapja tartalmazza-e B befoglaló téglalapját.
- $A\&\&B$: A befoglaló téglalapja átfedésben van B befoglaló téglalapjával.

4

Redundanciamentes topologikus adatszerkezetek

Ebben a fejezetben olyan adatstruktúrákat fogunk bemutatni, amelyek szerkezetükből adódóan megőrzik a topológiát, és nem tárolnak geometriai elemeket (pontokat) redundáns módon. Ezt úgy érik el, hogy a poligonokat, vonalláncokat alkotó node-okat tárolják, és az egyes komplexebb objektumok (poligonok) csak hivatkoznak az őket alkotó pontokra. Ennélfogva a szomszédos poligonok azonos node-jai csak egyszer lesznek tárolva, ezáltal megszűnik a redundáns tárolás, és a szomszédos poligonok között véletlen hézagok, átfedések létrejöttének az esélye.

Amint látni fogjuk, ez a logika kiterjeszthető a nem egy rétegen, egy feature classban tárolt adatokra is. Így akár különböző rétegek node-jait, amelyek azonos nyomvonalon futnak, sem kell redundánsan tárolni. Ezáltal a klasszikus réteg vagy feature class fogalom is lazulni fog.

4.1. Topológialeírás relációs adatbázis-kezelőkkel

A relációs modellel szemben a következő elvárásokat támasztjuk poligonok leírása esetében:

- tárolja a geometriai elemek topológiai viszonyait
- ne engedjen meg hibás eseteket (hézag- és átfedésmentesség)
- a poligonok ne metsszék egymást
- olyan eseteket is tároljon, amikor egy poligonban egy másik poligon benne van (pl. sziget egy tóban, úszó telkek a lakótelepeken, stb.).

- tárolja a geometriai entitások attribútumait: egy poligon milyen színű, egy vonal szaggatott-e, stb.
- a tárolt topológiával kapcsolatos kérdésekre könnyen válaszoljon: egy pont hol van, egy polyline mit metsz, egy poligon mivel szomszédos, milyen másik poligonokat tartalmaz, stb.

4.1.1. Pont-kontúr-poligon struktúra

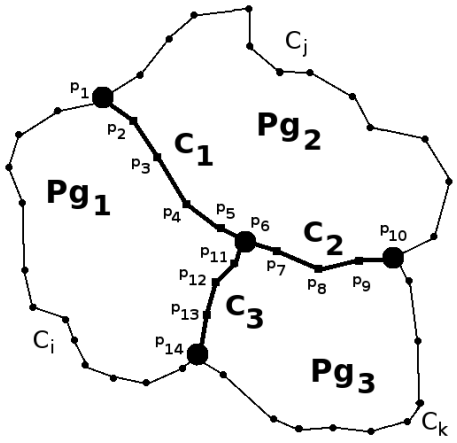
A fenti kritériumokat relációs adattáblák kapcsolatrendszerével fogjuk leírni, amely mintegy automatikusan biztosítja a poligontopológia megőrzését. Lássunk erre egy példát. Vizsgáljuk meg a 4.1. ábrát. Építsük fel a $\mathbf{Pg_1}$, $\mathbf{Pg_2}$ és $\mathbf{Pg_3}$ poligonokat az őket alkotó vonalláncokból, vagyis a $\mathbf{Pg_1}$ poligont $C_1, C_3 \cdots C_i$ kontúrokból, $\mathbf{Pg_2}$ poligont $C_1, C_2 \cdots C_j$ kontúrokból, és végül a $\mathbf{Pg_3}$ poligont $C_2, C_3 \cdots C_k$ kontúrokból. A C_1 kontúrt a $p_1, p_2, p_3, p_4, p_5, p_6$ pontok, a C_2 kontúrt a $p_6, p_7, p_8, p_9, p_{10}$ pontok, a C_3 kontúrt pedig a $p_6, p_{11}, p_{12}, p_{13}, p_{14}$ pontok alkotják.

Hozzunk létre négy relációs táblát (4.2. ábra). Az elemi geometriát, vagyis a pontok azonosítóit, és x,y koordinátáit a *Points* tábla tartalmazza. A *Points.id-point* a *Points* táblában elsődleges kulcs. A *Contours* tábla ezekre a pontokra hivatkozva épül fel, mivel a *Contours.id-point* mezője, mint idegen kulcs, hivatkozik a *Points* tábla megfelelő rekordjára. Ezen kívül még tartalmaz egy *point-order* nevű mezőt, amely az összekötési sorrendet határozza meg.

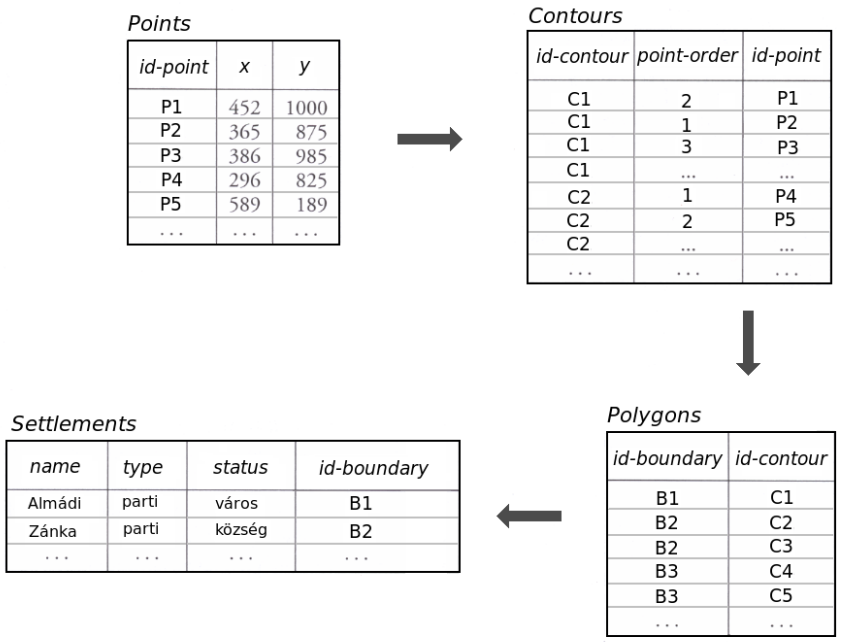
A *Polygons* tábla a *Contours* rekordjaiból épül fel. A *Polygons* tábla tartalmaz egy *id-boundary* nevű mezőt, amely elsődleges kulcs ebben a táblában, valamint egy *id-contour* nevű mezőt, amely mint idegen kulcs, a *contours* tábla megfelelő rekordjára mutat. A *Polygons* tábla annyi vonalláncból áll, ahány határvonala van a poligonnak.

Végül a *Settlements* tábla, amely a poligonok neveit (vagy bármely azonosítóját) tartalmazza, a *Polygons* tábla azonosítóira hivatkozik (*id-boundary*), mint geometriai összetevőre, a többi adata viszont leíró tartalmú (name, type, status). Vegyük észre, hogy ebben az esetben egy node megváltoztatása a *Points* táblában (pl. eltolása vagy törlése) nem rontja el a topológiát, ha az eredetileg létezett, nem változtatja meg a szomszédsági viszonyokat. Egy ilyen nagy hatékonyságú struktúra már képes biztosítani topologikus tulajdonságok megőrzését.

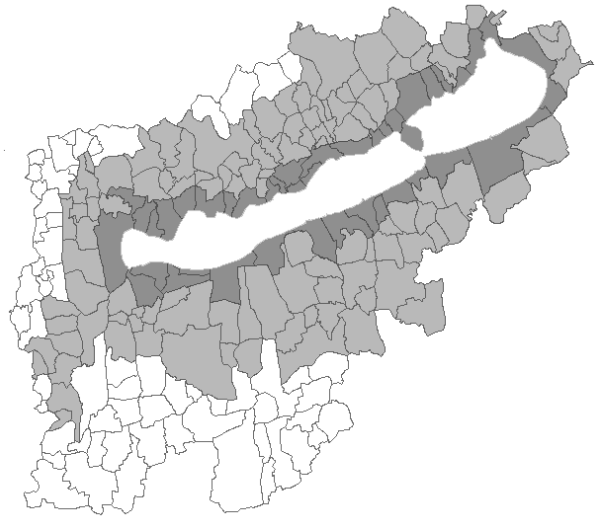
Példaképpen adjunk meg egy hagyományos SQL-parancsot, amely a *Settlements* táblából kiválasztja a „type" mező alapján a Balaton-parti települé-



4.1. ábra. Poligonokat felépítő kontúrok rendszere. Ha csak a pontokat tároljuk a koordinátáikkal, és a kontúrok csak hivatkoznak az őket felépítő pontokra, valamint a poligonok csak hivatkoznak az őket felépítő kontúrokra, akkor egyetlen pontot sem tárolunk redundánsan. További, a topológia megőrzését segítő tulajdonság, hogy egy pont elmozdítása az összes tőle függő objektumra ugyanazt a hatást fogja gyakorolni, vagyis megőrződik a topológia



4.2. ábra. A relációs táblák kapcsolatrendszere

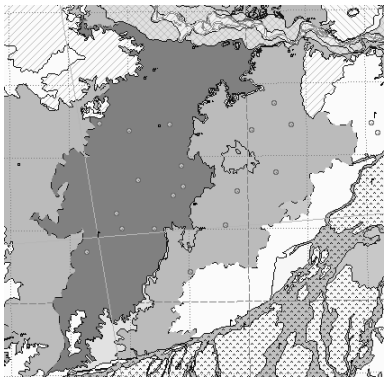


4.3. ábra. A parti települések lekérdezésének eredménye a sötétszürkével kitöltött poligonok halmaza

seket. Hangsúlyozni szeretnénk, hogy a lekérdezés eredménye a táblák kapcsolatrendszerének köszönhetően térbelileg is értelmezhető. Nyilvánvalóan még nem való egy végfelhasználónak, de az már látható, hogy ilyen parancsokból felépíthető egy olyan függvénykönyvtár, amely szintaktikailag térbeli lekérdezésként fogalmazható meg. Az SQL-parancs tehát a következő, amelynek eredményét a 4.3. ábra mutatja:

```
SELECT polygons.id-contour
FROM settlements, polygons, contours, points
WHERE type='Parti'
AND settlements.id-boundary=polygons.id-boundary
AND polygons.id-contour=contours.id-contour
AND contours.id-point=points.id-point
ORDER BY polygons.id-contour, point-order
GROUP BY name
```

Vizsgáljuk meg gyakorlati szempontból a 4.1. ábrán látható poligonstruktúrát. Sem a layer, sem a feature class logikájú rendszerekben nem lehetséges, hogy egy poligonnak eltérő színű határoló vonalai legyenek, vagyis



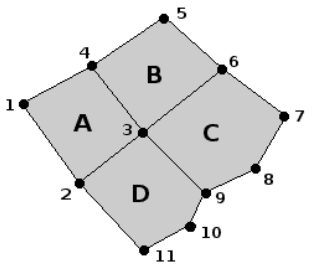
4.4. ábra. Egy földtani térképrészlet, ahol egy poligont különböző típusú vonalak határolnak. A hagyományos layer/feature class logikájú rendszerekben egy objektumnak csak egyféle határoló vonala lehetséges

vizualizációs attribútumokat csak az egész poligonra vonatkozóan adhatunk meg. Márpedig a gyakorlatban ilyen igény felmerülhet. Különösen geológiai térképeken láthatunk ilyen poligonokat (4.4. ábra). Ennek a földtani térképezésre jellemző okai vannak, a részletek jelen pillanatban nem érdekesek. Az azonban érdekes, hogy ilyen probléma megoldását a térinformatikai rendszerek gyakorlati művelői úgy oldják meg, hogy a poligonréteget szétvágják vonalláncok rendszerére, ekkor pedig már lehetségessé válik az eltérő színezés.

Az adatbázis-konzisztencia szempontjából ez a megoldás nemcsak, hogy nem elegáns, de kifejezetten kockázatos. A poligonok szétvágásával létrejövő vonallánc-halmaz elszakadt az eredeti poligonhalmaztól, és ha abban valamilyen változás áll be, akkor annak a vonallánc-halmazra történő átvezetéséről gondoskodni kell. Arról már nem is beszélve, hogy ezzel a „megoldással" többszörös redundanciát viszünk az adatbázisba. Mennyi hibaforrás!

Vegyük észre, hogy a 4.1. ábrán látható struktúrában azonban lehetséges eltérő színeket megadni a kontúroknak minden nagyobb erőfeszítés nélkül. Ráadásul szükségtelen redundáns rétegeket létrehozni, poligonokat vonalláncokká szétszedni, a különböző rétegek összhangjáról gondoskodni.

Hasonló gyakorlati probléma, hogy a földmérésben különböző minőségi osztályt jelentenek egy háznak azok a sarokpontjai, amik telekhatárra esnek, mint azok azok, amelyek a telek belsejében vannak. Egy poligonnal repre-



4.5. ábra. Az A, B, C, D poligonok, és az őket alkotó {1,2,3,4,5,6,7,8,9,10,11} pontok láthatók

zentált telek esetében a pontok ilyetén megkülönböztetése a pontok megduplázása nélkül nem lehetséges. A későbbiekben be fogunk mutatni egy olyan adatszerkezetet, amely a poligonok és a pontok közvetlen kapcsolatát írja le kontúrok beiktatása nélkül.

4.1.2. Pont-poligon struktúra

A fentebb bemutatott szerkezetnél egyszerűbb struktúrát mutatunk be ebben a fejezetben. Koordinátákat ebben az esetben is csak a pontokhoz fogunk rendelni. A poligonok hivatkozni fognak ezekre a pontokra, így redundáns tárolás ebben az esetben sem áll fenn.

Mielőtt megnéznénk a táblák kapcsolatrendszerét, vessünk egy pillantást 4.5. ábrára. Hozzunk létre három adattáblát (4.6. ábra). A *Points* tábla tartalmazza a pontok egyedi azonosítóit (*Points.Ptid*) és koordinátákat (*Points.Xcoord*, *Points.Ycoord*).

```
CREATE TABLE Points
(Ptid INT NOT NULL IDENTITY,
Xcoord FLOAT NULL, Ycoord FLOAT NULL,
PRIMARY KEY (Ptid))
```

A *Polygons* táblában poligononként annyi rekord van, ahány node-ból áll a poligon. Egy idegen kulcsa van (*Points.Ptid*), amely azokra a rekordokra mutat (*Points.Ptid*) a *Points* táblában, amelyek alkotják a poligonokat.

```
CREATE TABLE Polygons
(idPolygons INT NOT NULL IDENTITY, idPoint INTEGER NOT NULL,
```

Points

Ptid	Xcoord	Ycoord
1	a1	b1
2	a2	b2
3	a3	b3
4	a4	b4
5	a5	b5
6	a6	b6
7	a7	b7
8	a8	b8
9	a9	b9
10	a10	b10
11	a11	b11

Polygons

idPolygons	Ptid	name	pointOrder
1	1	A	1
2	2	A	2
3	3	A	3
4	4	A	4
5	3	B	3
6	4	B	4
7	5	B	5
8	6	B	6
9	3	C	3
10	6	C	6
11	7	C	7
12	8	C	8
13	9	C	9
14	2	D	2
15	3	D	3
16	9	D	9
17	10	D	10
18	11	D	11

Pgs

idPg	name	Attributes
1	A	Attrib-1
2	B	Attrib-2
3	C	Attrib-3
4	D	Attrib-4

4.6. ábra. A 4.5. ábrán látható A, B, C, D poligonok reprezentációja a *Points*, *Polygons*, *Pgs* táblákkal

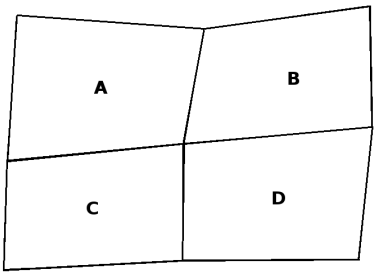
```
name VARCHAR(50), pointOrder INTEGER NULL,
PRIMARY KEY (idPolygons) FOREIGN KEY (idPoint)
```

A *Pgs* tábla annyi rekordból áll, ahány poligon van. Az idegen kulcsa azokra a *Polygons* nevű táblában lévő azonosítókra mutat, amelyek a poligont alkotják.

```
CREATE TABLE Pgs
(idPg INT NOT NULL IDENTITY, name varchar(50),
PRIMARY KEY (idPg), FOREIGN KEY(idPolygons))
```

4.1.3. Spagetti konverziója topologikus struktúrába

Mivel szinte minden létező adatunk valamilyen spagetti topológiájú struktúrában áll rendelkezésünkre, ezért ezek átalakítása az előző részben vázolt topologikus struktúrába alapvető feladat. Amint azt az OGC ajánlásban láthattuk, a poligonok kezdő és végpontjait, amelyek ugyanazok a pontok is, tárolják a spagetti szerkezetek. Az első lépés ezen pontok kettős tárolásának megszüntetése. Könnyen áttekinthető a Mapinfo MIF formátuma, ezért egy MIF/MID példán mutatjuk be a folyamatot. A MIF/MID formátum részletesebb ismertetése a 100. oldalon olvasható. A problémakör teljesen hasonló ESRI Shape formátum importja esetén is, csak az *shp* formátum nehezebben



4.7. ábra. Az A, B, C, D poligonok grafikai megjelenése

tekinthető át. A legtöbb térbeli funkciót is tartalmazó adatbázis-kezelő rendszer közvetlenül képes importálni az *shp* formátumot. Az eredmény teljesen hasonló lesz a következőkben ismertetettekhez.

MIF import

Vegyük a 4.7. ábrán látható egyszerű esetet. Ennek a MIF/MID-ben tárolt formáját a 4.8. ábra mutatja. Kreáljunk egy átmeneti táblát (*importedTable*, amely azért átmeneti, mert a konverzió végeztével töröljük), amibe betöltjük a MIF/MID importált adatait:

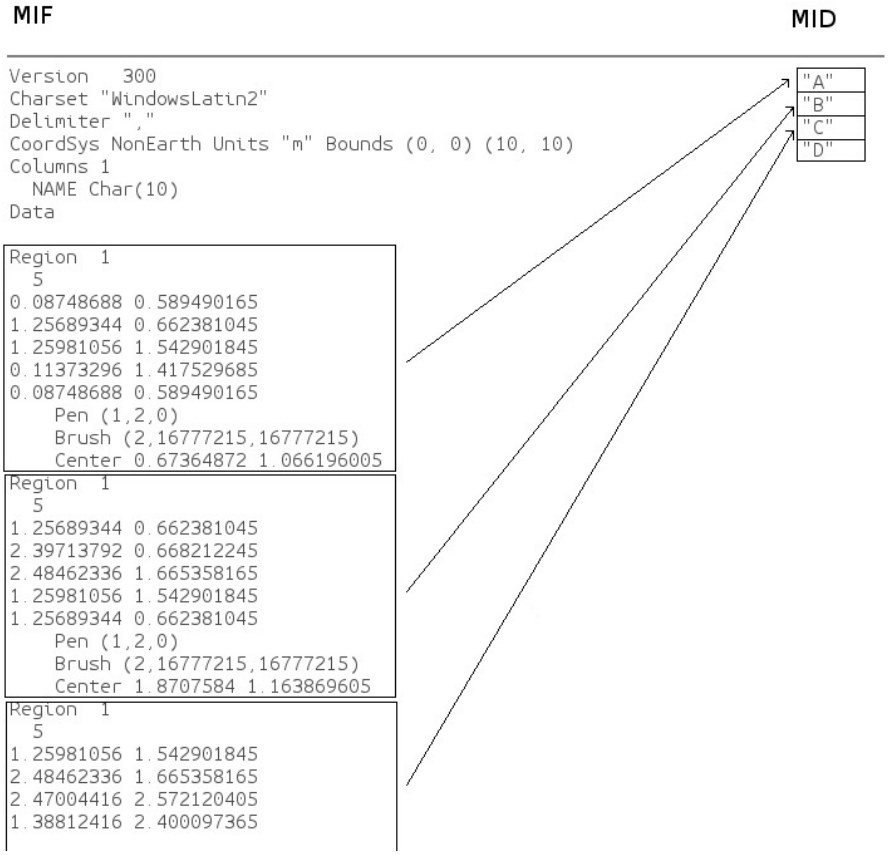
```
CREATE TABLE importedPoints
(idPoint INT NOT NULL IDENTITY,
x FLOAT NOT NULL, y FLOAT NOT NULL, name VARCHAR(50),
pointOrder INTEGER NOT NULL, PRIMARY KEY (idPoint))
```

majd töltsük fel adatokkal:

```
INSERT INTO importedPoints
(x, y, name, pointOrder)
VALUES (xcoord, ycoord, name, pOrder)
```

Dupla végpontok kiszűrése

Az átmeneti *importedTable* nevű táblában a MIF fájlnak megfelelően vannak az adatok, vagyis a node-ok koordinátái, valamint az adott poligon leíró adatai, ami jelen esetben a poligonok neve. A MIF specifikációnak megfelelően, itt a poligonok azonos kezdő- és végpontjai is szerepelnek. Ezeket szűrjük ki:



4.8. ábra. Az *A, B, C, D* poligonok leírása MIF/MID fájlokkal. A bal oldali rész a MIF-et, a jobb oldali a MID fájl tartalmát mutatja. Amint látható, a MIF fájlban egy bekeretezett adatcsoport, amely node-ok koordinátáit tartalmazza, megfelel a MID fájl egy sorának, amely az adott poligonhoz tartozó leíró adatokat tartalmazza, ami jelen esetben a poligonok neve (*A, B, C, D*)

```
SELECT name, x, y, MIN(pointorder) FROM importedPoints
GROUP BY name, x, y ORDER BY MIN(pointorder)
```

Szomszédos poligonok közös node-jainak kiszűrése

Ezután azokat a közös node-okat is ki kell szűrni, amelyek a szomszédos poligonok közös sarokpontjai. Ehhez először létrehozunk egy átmeneti táblát (*PolygonsPoints*) a következő módon:

```
CREATE TABLE PolygonsPoints
(idPoint INTEGER NOT NULL IDENTITY,
name VARCHAR(50) NOT NULL, x FLOAT NOT NULL,
y FLOAT NOT NULL, pointOrder INTEGER NOT NULL,
PRIMARY KEY (idPoint))
```

majd betöltjük a megmaradt, már nem redundáns pontokat:

```
INSERT INTO PolygonsPoints (name, x, y, pointOrder)
VALUES(x-values, y-values, pOrder)
```

A *Points* tábla létrehozása és feltöltése

Miután minden azonos pontot kiszűrtünk, kreáljuk meg a már topologikus referenciák létrehozásához szükséges, node-okat tartalmazó adattáblát:

```
CREATE TABLE Points (idPoint INT NOT NULL IDENTITY,
x FLOAT NULL, y FLOAT NULL,
PRIMARY KEY (idPoint))
```

Gyűjtsük le a szükséges adatokat az *importedPoints* nevű táblából:

```
SELECT s1.x, s1.y
FROM (SELECT name, x, y FROM importedPoints
GROUP BY name, x, y)
AS s1 GROUP BY s1.x, s1.y
```

majd ez alapján töltsük fel a *Points* táblát:

```
INSERT INTO Points (x, y) VALUES(x-values, y-values)
```

A *Polygons* és a *Pgs* táblák létrehozása és feltöltése

A *Polygons* tábla feltöltéséhez szükség van még az *importedPoints* táblára is. Előbb hozzuk létre a *Polygons* táblát:

```
CREATE TABLE Polygons
(idPg INT NOT NULL IDENTITY, idPoint INTEGER NULL,
name VARCHAR(50) NOT NULL, pointOrder INTEGER NOT NULL,
PRIMARY KEY (idPg))
```

majd a *Pgs* táblát:

```
CREATE TABLE Pgs
(idPg INT NOT NULL IDENTITY, name VARCHAR(50) NOT NULL,
PRIMARY KEY (idPg))
```

Ne felejtsük el, hogy a *Polygons* tábla poligononként annyi rekordot tartalmaz, ahány node-ból áll a poligon. Ezek a rekordok azonban csak a *Points* tábla megfelelő rekordjaira mutatnak, koordinátaértékeket nem tartalmaznak. Megadják azonban a node-ok összekötési sorrendjét (*Polygons.pointOrder* mező). A táblák feltöltéséhez gyűjtsük le a szükséges adatokat a következő SQL paranccsal:

```
SELECT name, x,y, pointorder FROM polygonsPoint
```

Ezeket töltsük be a *Polygons* táblába:

```
INSERT INTO Polygons
(name, idpoint, pointorder)
VALUES(name, idPoint, pointOrder)
```

majd a *Pgs* táblába is, amelynek annyi rekordja van, mint ahány poligon:

```
INSERT INTO Pgs SELECT name FROM Polygons GROUP BY name
```

Vegyük észre, hogy a *Pgs* tábla az, amely a poligonok azonosítóin kívül a poligonok leíró adatait is tartalmazhatja (pl. helyrajzi szám, település név, stb.). Az utolsó lépés a két ideiglenes tábla törlése (*importedPoint*, *PolygonsPoins*).

Ezek után tekintsük meg a 4.9. ábrát, amely a 4.7. ábra poligonjainak topologikus adatszerkezetben történő leírását mutatja. Figyeljük meg, hogy a

Points

	idPoint	x	y
	1	0,08748688	0,589490165
	2	1,25689344	0,662381045
	3	2,39713792	0,668212245
	4	0,11373296	1,417529685
	5	1,25981056	1,542901845
	6	2,48462336	1,665358165
	7	1,38812416	2,400097365
	8	0,17788992	2,499229525
	9	2,47004416	2,572120405

Polygons

	idPg	idPoint	name	pointOrder
	1	1	A	0
	8	2	A	1
	9	5	A	2
	16	4	A	3
	2	2	B	0
	7	3	B	1
	10	6	B	2
	15	5	B	3
	3	5	C	0
	6	6	C	1
	11	9	C	2
	14	7	C	3
	4	4	D	0
	5	5	D	1
	12	7	D	2
	13	8	D	3

Pgs

	idPg	name
	1	A
	2	B
	3	C
	4	D

4.9. ábra. Fiktív poligonok (4.7. ábra) leírása topologikus adatszerkezettel. A *points* tábla csak pontokat, azok azonosítóit és koordinátáit tartalmazza. Rájuk hivatkozik a *Polygons* tábla, amely annyi rekordból áll, mint ahány node-ból egy-egy poligon. Végül a *Pgs* tábla annyi rekordból áll, ahány poligon van

Points tábla mit sem tud arról, hogy mely táblák hivatkoznak rá. A példában a *Polygons* tábla hivatkozik rá, amely poligonokat ír le, de bármely más tábla is hivatkozhatna a *Points* tábla bármely rekordjára, nemcsak poligonokat leíró, hanem vonalláncokat (polyline) leíró is. Ez a lehetőség azért figyelemre méltó, mert a gyakorlatból jól ismert eset, amikor poligonok határán futnak vonalláncok (pl. vízfolyás két megye határa, két település határán fut az autópálya, stb.) eddig semmiképpen nem voltak leírhatók redundanciamentesen. Mindegyik objektumcsoportnak tárolni kellett a saját node-jait. A fenti topologikus modellben erre nincs szükség. Ezzel voltaképpen erősen felpuhul a réteg (feature class) fogalom. Elegendő tárolni az objektumainkat alkotó pontokat, minden egyebet a kapcsolatrendszer, a referenciák rendszere határoz meg.

4.1.4. Megjelenítés a topologikus adatstruktúrákból

A topologikus adatszerkezetből való rajzolás nem triviális probléma. A vektorgrafikus rajzoló könyvtárak lehetővé teszik ugyan poligonok, vonalak, pontok és szövegek kirajzolását a canvasra, de a spagettimodellhez hasonló logikával. Egy poligont a node-jai és az összekötési szabály definiál, vagyis nem vizsgálja, hogy egy pont egy vagy több poligonhoz tartozik. A rajzo-

lás poligonról poligonra történik, azaz az adott poligon node-jait köti össze, aztán veszi a következő poligont, és így tovább.

Ha rajzolni akarunk a topologikus adatstruktúrából, akkor abból előbb spagettit kell előállítani, (például valamilyen SQL paranccsal), majd az így kapott adathalmazt átadni a rajzoló függvényeknek. A következő SQL parancs topologikus adatszerkezetből spagettibe történő lekérdezést mutatja, amelynek eredményét aztán átadhatjuk a rajzoló függvénynek (például .NET esetén a DrawPolygonnak).

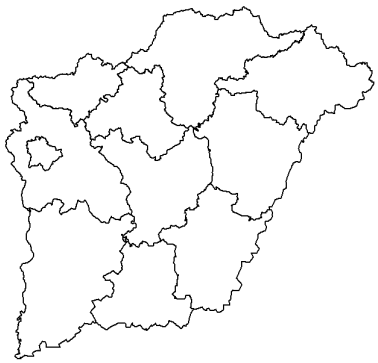
```
SELECT name, points.idpoint, x, y, polygons.pointorder
FROM points, polygons
WHERE polygons.name=pgs.name
and Points.idPoint=polygons.idpoint
ORDER BY polygons.pointorder
```

Ezek után lássunk néhány valódi adatot, amelyet topologikus adatbázisból rajzoltunk ki (Budapesti kerületek: 4.10. ábra, Magyarország megyéi a Dunától keletre: 4.11. ábra, Magyarország járásai a Dunától keletre: 4.12. ábra, Budapest XI. kerületének egy részlete: 4.13. ábra).

4.10. ábra. Budapesti kerületek poligonjai

4.1.5. A vonaltopológia egy relációs modellje

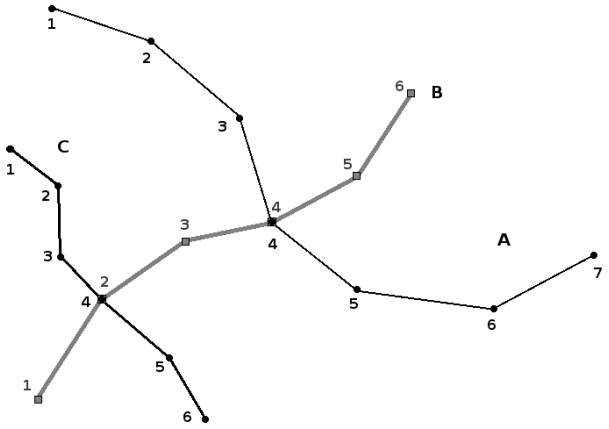
A fentiekhez hasonlóan, amikor poligonokat írtunk le relációs táblák kapcsolat rendszerével, vonalas geometria elemek is reprezentálhatók redundanciamentesen. A spagettimodellben a keresztező vonalak közös node-jaik mindkét objektumban tárolódnak, így redundancia keletkezik (4.14. ábra).



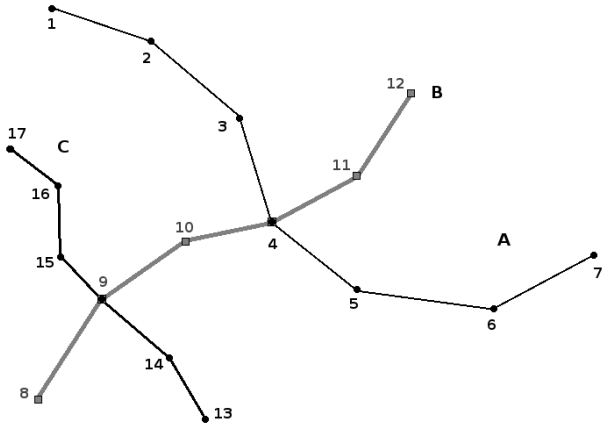
4.11. ábra. Magyarország megyéi a Dunától keletre

4.12. ábra. Járások poligonjai a Dunától keletre

4.13. ábra. Egy város egy részének telkei



4.14. ábra. Kereszteződő vonalak. A **C** jelű görbe 4. node-ja, és a **B** jelű görbe 2. pontja közös. Ugyancsak közös a **B** jelű görbe 4. node-ja, és az **A** jelű görbe 4. node-ja



4.15. ábra. Kereszteződő vonalak közös pontjait nem tároljuk kétszer. Azonosítóik alapján állapítható meg, hogy egy pont egy vagy több vonal létrehozásában játszik-e szerepet

A 4.15. ábrán láthatók a vonalakat alkotó node-ok, melyek közül néhány több vonalhoz is tartozik (a 4. és 9. jelű pontok)). Ha a pontokat, koordinátáikkal együtt egy csak pontokat tartalmazó táblában, a vonalakat reprezentáló hivatkozásokat pedig egy másik táblában tároljuk, akkor ez a redundancia megszüntethető.

A *Points*, *Lines* és *Plines* táblák létrehozása

Hozzunk létre egy csak pontokat tartalmazó táblát (*Points*), amely a pontok azonosítóit és koordinátáit tartalmazza. Hozzunk létre továbbá egy *Lines* és egy *Plines* nevű táblát. A *Lines* tartalmazza a vonalat alkotó pontok hivatkozásait, mégpedig szám szerint annyit, ahány pontból áll az adott vonal. A *Plines* nevű tábla a vonalobjektumot tartalmazza a leíró adataival és a *Lines* táblára való hivatkozással (4.16. ábra). Ez az adatszerkezet nem tartalmaz redundáns pontokat, így egy-egy node megváltoztatása (mozgatása vagy törlése) nem okoz szakadást az eredetileg szakadásmentes vonaltopológiában. Azt azért látni kell, hogy ha egy közös pontot törlünk a *Points* táblából, akkor a kereszteződő vonalak többé nem kereszteződnek (egymás felett haladnak el).

Points

Ptid	Xcoord	Ycoord
1	x1	y1
2	x2	y2
3	x3	y3
4	x4	y4
5	x5	y5
6	x6	y6
7	x7	y7
8	x8	y8
9	x9	y9
10	x10	y10
11	x11	y11
11	x12	y12
11	x13	y13
11	x14	y14
11	x15	y15
11	x16	y16
11	x17	y17

Lines

idLines	Ptid	name	pointOrder
1	1	A	1
2	2	A	2
3	3	A	3
4	4	A	4
5	5	A	5
6	6	A	6
7	7	A	7
8	8	B	1
9	9	B	2
10	10	B	3
11	4	B	4
12	11	B	5
13	12	B	6
14	13	C	1
15	14	C	2
16	9	C	3
17	15	C	4
18	16	C	5
19	17	C	6

Plines

idLn	name	Attributes
1	A	Attrib-1
2	B	Attrib-2
3	C	Attrib-3
⋮	⋮	⋮

4.16. ábra. Vonalak redundanciamentes tárolását biztosító adatszerkezet

Vegyük észre, hogy ez az adatszerkezet a redundanciamentes tároláson kívül további előnyöket is tartogat. Útvonal-optimalizáló algoritmusok számára nagy segítség lehet ez az adatszerkezet. A függelékben ismertetett példa (lásd 130. oldal) esetében adottnak vesszük az érintendő pontok meglétét, miközben egy valóságos térinformatikai rendszerben (amely spagettitopológiájú network modellt használ), egyáltalán nem nyilvánvaló, hogy egy adott pontból indul-e többfelé is él. Ennek megkereséséről külön gondoskodni kell valamely térbeli keresőfunkció segítségével. A relációs táblákban tárolt vonalhálózatok esetében fel sem merül, hogy van-e elágazási lehetőség az adott pontban, hiszen nincs szükség esetlegesen eltérő rétegekben, feature classokban tárolt vonalak közös részeinek megkeresésére.

4.1.6. Összegzés

A redundanciamentes topologikus adatszerkezet létrehozásának folyamatát didaktikai szempontok alapján mutattuk be. A valóságban, egy konkrét implementációban a folyamat kevéssé szegmentált. Az átmeneti táblák léte inkább a jobb érthetőséget szolgálta, semmint a célszerűséget. A gyakorlatban a folyamat egyszerűbben is végrehajtható, tömörebb, összetettebb SQL-parancsokkal. Ezek megértése azonban kevésbé lett volna követhető. Ezért választottuk a folyamat bemutatásnak ezt a kissé iskolás módját.

Az OGC szabvány alapján lehetséges ugyan korrekt topológiájú adatrendszerek létrehozása, de a szabvány nem garantálja a topológia megőrzését és a redundanciamentes tárolást. A előző részben arra mutattunk egy megoldást, hogy lehetséges poligon topológiát megőrző adatstruktúra létrehozása. Sajnálatos módon ez a struktúra nem követi az OGC szabványt, mivel a redundanciamentes tárolásból következően ellentmond az OGC szabvány előírásainak. Amennyiben a performanciatesztek megfelelő sebességűnek bizonyulnak az OGC alapú létező megoldásokkal összehasonlítva, akkor el kell gondolkozni a szabvány továbbfejlesztésén.

További következmény, hogy a térképészet megjelenítési szempontjai többnyire nem egyeztethetők össze a topológialeírás követelményeivel (nemcsak a topológia megőrző adatstruktúrákéival, hanem már az OGC szabvány előírásaival sem). El kell fogadnunk, amit már hallgatólagosan régóta tudunk, hogy a kartográfiai célú adatrendszerek nem alkalmasak a térinformatika céljaira. Fordítva is igaz ez a megállapítás: a térinformatikai célú adatrendszerek csak korlátozott mértékben, és jelentősebb utómunkálatok (kartografálás) után válnak csak alkalmassá térképészeti célú felhasználásra. Számos gyakorlati példa bizonyítja, hogy a papírtérképek digitalizálása, ha kartográfiai szempontok lettek megállapítva minőségi követelményként, a térinformatika számára használhatatlan eredményt produkált. Az ilyen adatrendszerek nem többek, mint a papírtérképek digitális változatai. Amennyiben térinformatikai felhasználás a cél, akkor rengeteg utómunkálatot kell még elvégezni ahhoz, hogy az eredmény szakadás-, hézag- és átfedésmentes topológiájú legyen. Arról nem is beszélve, hogy az a digitális térkép, amely strukturális okokból nem ad lehetőséget attribútumadatok egyszerű hozzákapcsolására, alig használható bármire a megjelenítésen kívül. Egy profán hasonlattal élve, a jövendőbeli házunk ott van bármely Tüzép-telepen, csak ki kell válogatni a szükséges elemeket, és össze kell állítani olyan struktúrába, hogy funkcionálisan megfeleljen egy házzal szembeni követelményrendszernek.

4.2. Poligontopológia-leírás duális gráfokkal

Az előző fejezetben azt mutattuk be, hogy miként lehet poligonok viszonyrendszerét relációs adatbázis-kezelők tábláinak kapcsolataival leírni. Más lehetőségek is vannak poligonok szomszédsági viszonyainak leírására. Most egy jól ismert, de – érdekes módon – a gyakorlatban ritkán alkalmazott módszert mutatunk be, amely duális gráfokkal (lásd 121. oldal, Gráfelmélet összefoglaló) írja le a poligonok kapcsolatrendszerét. Ez a módszer csak

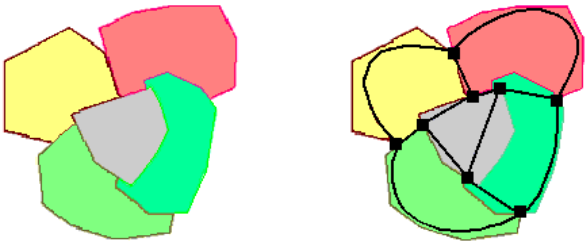
poligonok leírására alkalmazható, szemben a relációs modellel, amely vonalláncokra és poligonokra egyaránt alkalmazható.

A topologikus tárolás elemei – az eddigiekhez hasonlóan – poligonok, vonalak és pontok. A kérdés az, hogy hogyan tároljuk ezeket az objektumokat úgy, hogy a szomszédsági viszonyok eleve bele legyenek építve az adatszerkezetbe. A spagettimodellben a helyes topológiáról magunknak kell gondoskodnunk, az magától nem teljesül (természetesen szoftveres megoldások lehetnek, amelyek arra kényszerítik a felhasználót, hogy ne vétsen topológiai hibát). A topologikus adatszerkezetnek a következőkre kell képesnek lennie:

- tárolja a geometriai elemek topológiai viszonyait
- ne engedjen meg hibás eseteket, azaz a poligonok közt ne legyenek rések (hacsak nem olyan természetűek, mint például városhatárok), azok ne metsszék egymást, stb.
- olyan eseteket is tároljon, amikor egy poligonban egy másik poligon benne van, ugyanis ilyen esetek a térképeken előfordulnak (pl. sziget egy tóban).
- tárolja a geometriai entitások attribútumait: egy poligon milyen színű, egy vonal szaggatott-e, stb.
- a tárolt topológiával kapcsolatos kérdésekre könnyen válaszoljon: egy pont hol van, egy polyline mit metsz, egy poligon mivel szomszédos, milyen másik poligonokat tartalmaz, stb.

Mivel a geometriai objektumok típusa pont, vonallánc és poligon lehet, ezek mind külön térképi rétegek leírására használhatók. Amint láttuk, pontok esetében elegendő azok azonosítóit és helyét (koordinátáit) tárolnunk. A pontok nemcsak összetettebb geometriai objektumok elemei lehetnek, hanem önálló pontszerű entitásai egy információs rendszernek. Ilyen esetekben a pontok grafikai megjelenése tematikus tartalmat is hordozhat. A grafikus megjelenés legtöbbször valamilyen egyezményes grafikus jellel történik, amelyet valamely jelkészletből, ábrázolási konvenció gyűjteményből választunk ki (jelkulcs).

A vonalláncokat célszerű gráfokkal leírni, ahol a vertexek a vonalláncokat alkotó pontok, az élek pedig egy-egy node összekötöttségét reprezentálják. Ezekhez is rendelhetők a megjelenítésre vonatkozó grafikus stílusok (vastag



4.17. ábra. Példa egy síkgráfra. Az ábra bal oldalán a poligonok, az ábra jobb oldalán a területek által meghatározott síkgráf látható

vonal, szaggatott vonal, több párhuzamos vonalból álló egység), de ezek nem érintik a vonalláncokat alkotó node-ok elhelyezkedését.

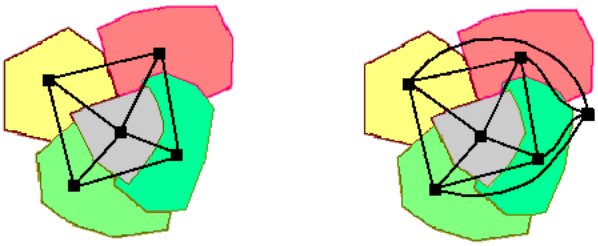
A poligonokat egy speciális, összetett gráfstruktúrával ábrázoljuk, amelyet síkgráfnak neveznek. A síkgráf háromféle elemtípusból áll: terület, él és elágazási pont. A terület reprezentálja a poligonok területeit, az élek pedig a területeket elválasztó polyline-okat, az elágazási pontok pedig a polyline-ok találkozási pontjai lesznek.

4.2.1. Síkgráfok

A síkbeli struktúrát valójában két gráfban tároljuk. Az egyik gráf csúcsai az elágazási pontok, ezek között a pontok között élek pedig ott lesznek, ahol ezen a pontok között polyline-ok vannak. Ezekben az élekben tároljuk azt is, hogy tőle jobbra és balra milyen területek vannak. Nevezzük síkgráfnak ezt a gráfot (4.17. ábra).

A másik gráf a síkgráf duálisa, amelyben a csúcsok a területek, az élek pedig azt jelentik, hogy egy területnek egy másik a szomszédja. A területhez tároljuk az őt határoló polyline-okat is, egyszóval azt a valódi poligont, amit az reprezentál. A duális gráf szerepe, hogy a síkgráffal együtt leírja a poligonok közti topológiai viszonyokat és azok geometriáját is. Szomszédságra, elérhetőségre vonatkozó kérdésekre így már könnyen válaszolhatunk (ezzel a struktúrával azonban még nem tudunk egymásba ágyazott területeket tárolni).

Definiáljunk ezért a valós területeken kívül egy új típust, a befoglaló területeket. Ezek a valóságban külön poligonként nem megjelenő területek.



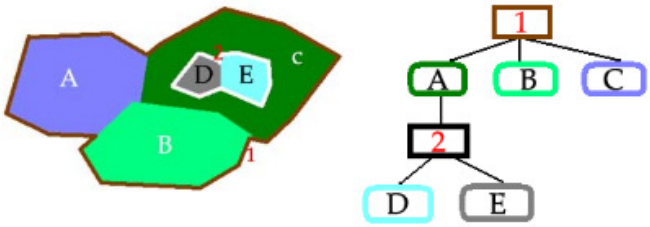
4.18. ábra. A 4.17. ábrán lévő síkgráf duális gráfja és a duális gráf a befoglaló területekkel kiegészítve

Ez tulajdonképpen egy befoglaló terület, egy olyan poligonhalmaz kontúrja, amelynek minden elemét el tudjuk érni az őket leíró síkgráf duális gráfjában (4.18. ábra). A legkülső befoglaló terület tulajdonképp a poligonstruktúra kontúrja, vagyis legfelső szinten az egész kép, a többi szinten pedig egy teljes lyuk egy valódi poligonon belül.

4.2.2. Befoglalási fa

A valós és befoglaló területeket hierarchiába rendezhetjük, és fával ábrázolhatjuk. A fa gyökerében a kép kontúrját jelölő befoglaló terület lesz, a gyerekei azok a poligonok, amelyek a duális gráfban elérhetőek belőle. Így minden valós területet hozzárendeltük az őt közvetlenül tartalmazó befoglaló területhez, másrészről az egész képet tartalmazó befogó területen kívül minden befoglaló terület egy valós terület alá tartozik (4.19. ábra).

A befoglalási fában a gyökérben az egész területstruktúra kontúrja lesz. A páros szinteken befoglaló, a páratlanokon valós területek vannak. Egy befoglaló terület gyerekei azok a valós területek, amelyek belőle a duális gráf bejárásával elérhetők. A befoglalási fát nem egy teljesen külön adatszerkezetben tároljuk, hanem a duális gráfot egészítjük ki úgy, hogy az leírja a tartalmazásokat is. Ha egy poligonon belül van egy beágyazott poligonrendszer, akkor ennek kontúrja lesz a befoglaló területe. A befoglaló területek is megjelennek a duális gráfban, mégpedig úgy, hogy egy befoglaló területnek szomszédja lesz az összes olyan poligon, amely az adott poligonrendszer szélén van. A területekhez tároljuk azt is, hogy befoglaló vagy valós területet írnak-e le.



4.19. ábra. A valós és a befoglaló területek kontúros jelöléssel. A valós területek betűvel, a befoglalók számmal címkézve, illetve a struktúrájához tartozó befoglalási fa

Ha valós területet írnak le, akkor tartalmazza a határoló irányított éleket negatív körüljárási irány szerint, és a beágyazott befoglaló területek listáját. Ha az adott terület befoglaló terület, akkor tartalmazza a határoló irányított éleket pozitív körüljárási irány szerint, és a beágyazott valós területek listáját. A síkgráfban a szélen lévő éleknek eddig csak az egyik oldalán voltak területek, mostantól a másik szomszédját is beállítjuk, mégpedig a befoglaló területre. Így a síkgráf és duális gráf élei meg fognak egyezni abban az értelemben, hogy a síkgráf éleinek a duális gráf egy másik szerepét adja meg, azaz, hogy az él által reprezentált polyline két területet választ el.

Mivel az élek megegyeznek a síkgráfban és duális gráfjában, ezért azokat nem is tároljuk kétszer. A duális gráfba azzal teszünk be egy élt, hogy a bal és jobb oldalára eső területeket beállítjuk. Ez azért jó, mert könnyű a használat során egyikről másikra áttérni, valamint egy vonallánc változás egyszerre jelenik meg mindenütt, így ilyen változtatással nem képzelhetőek el topológiailag érvénytelen állapotok. Ezért mondjuk azt, hogy a duális gráf reprezentáció topologikus adatmodellt eredményez.

A síkgráf élei irányítatlanok, de egy élben a polyline-t mindkét irányítás szerint tároljuk, hogy az építés során tudjunk hivatkozni mindkét irányításra. A reprezentációban a pontokhoz tároljuk a pont koordinátáját, és a kimenő irányított éleket pozitív körüljárás szerint. Az élek irányítatlan éleket írnak le, de hozzárendelünk egy tetszőleges irányítást, ami szerint leírjuk a kezdő- és végpont azonosítóját, valamint a bal és jobb oldali terület azonosítóját. Ha

nincs jobb vagy bal terület, akkor a befoglaló terület azonosítóját használjuk. Ezenkívül tároljuk az élhez tartozó polyline-t is.

Az eddig leírt adatszerkezet előnye, hogy támogat minden számunkra fontos műveletet, megőrzi a topológiai helyességet. A következőkben áttekintjük az előállítás módját.

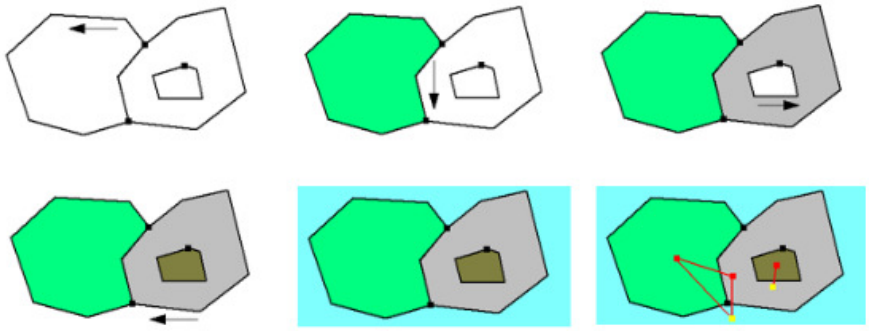
4.2.3. Síkgráf és duális gráf előállítása

A síkgráf előállításának a bemenete egy helyes polyline-halmaz, amelyek a poligonok topológiáját meghatározzák.

- kigyűjtjük a vonalláncok végpontjait, ezekből létrehozzuk a gráf pontjait és tároljuk, hogy a síkon melyik pontot jelölik.
- az összes vonalláncot egy éllel reprezentáljuk
- az él két végpontjához létrehozott pontok éllistáiba beszúrjuk az élt, figyelve arra, hogy az élek sorrendjének rendezettsége megmaradjon

A síkgráf előállítása után a duális gráf előállítása következik. Ehhez először az élek feldolgozásával előállítjuk a területeket. Ehhez dolgozzuk fel az éleket valamilyen $e_1, e_2 \dots e_{|E|}$ sorrendben. Az i-edik lépésben a következőket tesszük (4.20. ábra):

1. nézzük meg, hogy az e_i mindkét oldalán van-e már terület
2. ha igen, akkor menjünk tovább
3. ha nem, akkor:
   (a) hozzunk létre egy új T területet az e_i megfelelő oldalára
   (b) járjuk körbe a területet úgy, hogy az e_i a körüljárás első éle olyan irányítás szerint, hogy T az e_i bal oldalára esik
   (c) a pontokban mindig azon az élen menjünk tovább, ami a körüljárási irány szerint rendezett éllistában azután következik, amin bejöttünk
   (d) a bejárt élek mindegyikének bal oldalára állítsuk be a T területet
   (e) a T-hez tároljuk az így kapott körbejárás szerint azt a polyline-t, ami határolja. Tulajdonképp ez lesz a poligon, amit a T jelöl



4.20. ábra. A kiinduló polyline halmaz; első lépés iránya; a síkgráf létrehozásának lépései; a végső síkgráf és a duális gráf területei; a duális gráf csúcsai és élei (sötétek a valós területeket jelölő csúcsok, világosak a befoglalók)

A területfelépítés algoritmusának eredményeképp nemcsak a területek és az azok közti szomszédságok alakultak ki, hanem a poligonok is, amelyek a területet határolják. A körüljárási irányból kiderül, hogy valós, vagy befoglaló területről van szó. Ha ennek a poligonnak a csúcsai helyes körüljárás szerint tárolódnak a határoló poligonban, akkor a terület valós terület, különben határoló. Ezt a poligon előjeles területének kiszámításával tudjuk ellenőrizni. Ha az összes befoglaló területet megkaptuk, akkor nincs más dolgunk, mint a befoglalási fát megkonstruálni. Ezt a következőképp tehetjük:

1. rendezzük a befoglaló területeket területük szerint csökkenő sorrendbe

2. az első befoglaló terület, azaz a teljes kép, kivételével megkeressük a fában azt, hogy egy befoglaló terület melyik valós területbe tartozik, ezek listáját tegyük bele a duális gráfban a területet reprezentáló csúcsba

3. az aktuális befoglaló területből duális gráf bejárással elérhető területeket rendeljük hozzá a befoglaló területhez, ezek listáját tegyük bele a duális gráfban a befoglalási területet reprezentáló csúcsba

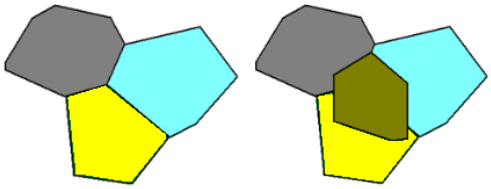
A rendezés miatt nem fordulhat elő olyan, hogy egy befoglaló terület tartalmaz egy megelőző befoglaló területet, így nem tudjuk elrontani az egymásba ágyazást. Ennek az adatszerkezetnek sok előnyös tulajdonsága van:

- Mivel semelyik élt nem tároljuk kétszer, ezért a topológiát nem tudjuk elrontani egy vonallánc egyszerű megváltoztatásával, ha átmetszünk vele egy másik vonalláncot. Ha azonban erre figyelünk, akkor ezen túl egy él változtatása mindkét poligonra vonatkozik, melyet elválaszt.
- Az adatszerkezetben könnyű lekérdezni egy poligon szomszédait, hiszen ahhoz csak a duális gráfban kell a poligont reprezentáló terület szomszédait lekérdezni.
- Támogatja az akárhány szinten beágyazott területeket, és a tartalmazást is könnyű vizsgálni a kiegészítő befoglalási fa vizsgálatával.

4.2.4. Síkgráf megváltoztatása

A felépített adatszerkezettel kapcsolatosan meg kell vizsgálni, hogy a változtatásokat mennyire nehéz elvégezni benne. Éleket törölni könnyű feladat, ilyenkor azt kell megtenni, hogy megnézzük, milyen területeket határolt el az él. Ha valós területeket, akkor azokat össze kell vonni egy közös területté. Ha valós és befoglaló területet, akkor a valós területet a befoglaló területhez csatoljuk abban az értelemben, hogy a szomszédai innentől azokon az éleken keresztül, amikkel eddig hozzá csatlakoztak, most a befoglaló területhez fognak. Miután ezt megtettük, még meg kell nézünk, hogy van-e olyan csúcs, amely csak két él találkozási pontjában van. Ha igen, akkor ezeket töröljük, és a két élt, ami a szomszédsági listájában van, összevonjuk.

A síkgráfok hátránya, hogy a felépített struktúrába új csúcsok beszúrása esetén a duális gráfban többszörös élek lesznek, ezért a csúcsok beszúrását önmagában nem szabad megengedni, csak akkor, ha azt élek beszúrása is követi, és új területek jönnek létre. Még nagyobb körültekintést igényel új poligonok beszúrása területek közé a lokális topológia megváltoztatásával, mert ez szinte az egész gráf újraépítését követeli (4.21. ábra) meg.



4.21. ábra. Példa a síkgráf megváltoztatására: a középső poligon beszúrása a gráfba új csúcsok és élek létrehozásával és törlésével jár, érdemesebb az elejétől felépíteni, mint az eredeti struktúrát változtatni

5

A geometriai megközelítés fonákságai

Az előző fejezetekben láthattuk, hogy a térbeliség leírása a földfelszín ábrázolandó objektumai geometriai elemeinek meghatározásán, leírásán alapul. Minden objektum alapeleme a pont. Ezek megfelelő szerkezetben való tárolása teszi lehetővé komplexebb objektumok (vonalláncok, poligonok) leírását. Sőt a sokat tárgyalt topológia is a pontok, a komplexebb objektumokat alkotó node-ok meghatározására épül.

Csakhogy egy tetszőleges hely megadásának nem mindig a pont geometriai megadása (x, y, z vagy λ, ϕ, z koordináták) a legcélravezetőbb módja, miközben az egzakt helymeghatározás mégis csak a pontok koordinátáinak ismeretén alapul. Ennek az ellentmondásnak a feloldásával foglalkozik ez a fejezet.

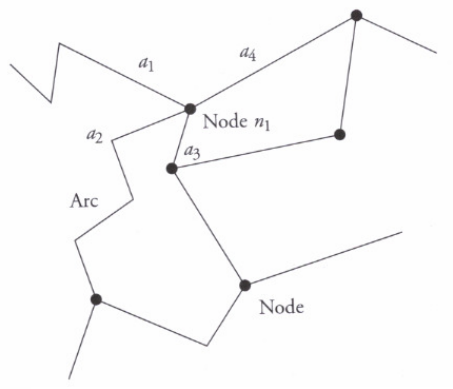
5.1. A hálózatmodell

A hálózat modell arra hivatott, hogy vonalak hálózatával írjunk le hálózatos működésű dolgokat, úgy mint közlekedésszervezés, szállítmányozás, logisztikai problémák, közműhálózatok, kapcsolatrendszerek, stb. Minden esetben arról van szó, hogy gráfokkal modellezzük az üzleti logika által definiált csomópontokból felépülő hálózatokat (5.1. ábra). Megkülönböztetünk arcot (ívet) és vonalláncot (polyline-t).

Az ábráról látható, hogy célszerű lehet, különösen útvonal optimalizációs problémák esetében, a sok node-ból álló vonalláncokat arcoknak nevezve másként kezelni, elvégre ezeken a vonallánc szakaszokon nincs elágazás.

5.1.1. Lineáris referencia

A helymeghatározás többféle módon is lehetséges. A jól ismert x, y, z koordináták megadása mellett a postai cím alapján történő helymeghatározás is



5.1. ábra. Node-ok, polyline-ok, arcok (ívek) alkotják a hálózatokat

régóta alkalmazott módszer. Ha valamelyik ismerősünk a címe helyett a lakhelye koordinátáit adná meg, meglepődnénk, mert bizonyos problémakörben a postai cím a helymeghatározás elfogadott rendje. A pozíció megadásának ez a módja csak a GPS-szel rendelkezők számára nyújt segítséget.

A vonalas létesítmények, mint a közmű- és közlekedési hálózatok (út, vasút) valamely pontjának térbeli leírására használjuk a következő módszert, amely a hálózati modell egyik igen általános felhasználása. Vegyük például az utak mentén lévő kilométerköveket, vagy a folyók mentén a folyamkilométereket. Ezek a helyek pontos koordinátákkal rendelkeznek. Számos esetben azonban a helymeghatározás egy-egy ilyen mérföldkőhöz képest történik (pl. a 123. folyamkilométertől 150 méterre folyásirányban). Utakon történt események (pl. balesetek) helyének megállapítására sem használjuk az abszolút koordinátákat, hanem valamilyen relatív meghatározást adunk, mint pl. baleset történt a 4-es út 52. és 53-as kilométere között, félúton. A folyamkilométerek, kilométerkövek, mint szelvénybeosztások, akár pontoknak is tekinthetők.

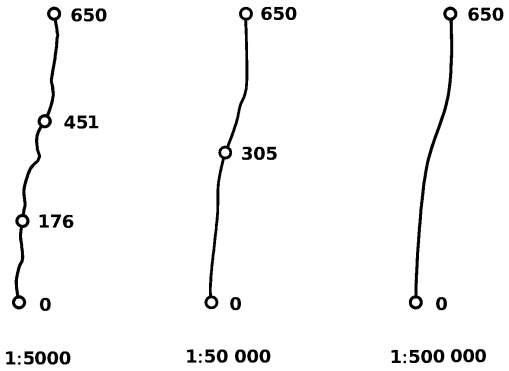
A hálózatot reprezentáló vonalláncokat alkotó node-ok azonban egyáltalán nem biztos, hogy egybeesnek a szelvénybeosztás pontjaival (képzeljünk el 10 km hosszú nyílegyenes vasúti vonalszakaszt, amelynek egy kezdő- és egy végnode-ja van. Teljesen felesleges lenne oda is node-ot tenni, ahol nincs változás a geometriában (5.2. ábra). A szelvénybeosztás kilométerenként tartalmaz egy pontot, így akár ez alapján generálhatnánk vonalláncokat. Ez sem

5.2. ábra. A Hódmezővásárhely és Orosháza közötti 47.sz. úton több nyílegyenes szakasz is van. A leghosszabb, 10 km-es szakasz kilométer beosztását mutatja az ábra (nagy fekete pontok). Kézenfekvő, hogy itt nincsenek node-ok, viszont az is nyilvánvaló, hogy a node-okra építve nem lehet szakaszokat elkülöníteni egy ilyen vonalszakaszon [forrás: Google Maps]

lenne jó megoldás, mert a geometria szempontjából mindenképpen redundáns node-ok tömegét hoznánk létre hosszú egyenes szakaszokon, ellenben kis görbületi sugarú szakaszokon az ilyen logika nem írná le megfelelő pontossággal a geometriát.

Tovább bonyolítja a helyzetet, hogy előfordulhatnak változások a geometriában (pl. nagyobb, többszintes csomópontot építenek ki több útvonal találkozásában, vagy véglegesen eltereli a folyót egy gát, erőmű megépülése miatt). Ilyenkor akár jelentősebb hosszváltozás is bekövetkezhet, ami ha a vonalas logikát követjük, valamennyi kilométerszelvény áthelyezését igényelné. Ez nyilván képtelenség.

Végül is mindegy, hogy milyen módszerrel határozunk meg az azonosítani kívánt helyeket, ha ezek egymással összhangban vannak, és azonos ered-



5.3. ábra. A különböző méretarányból és az eltérő üzleti logikából származó lineáris referencia alapú megközelítés sematikusan ábrázolva

ményre vezetnek. Szakterületenként eltérhetnek ezek az azonosítási módszerek. Lássunk ezek közül néhányat:

- földrajzi koordináták
- útszelvényszám
- szelvényazonosítóhoz képesti távolság
- városhatártól mért távolság
- kilométerkövektől mért távolság
- kereszteződésektől mért távolság
- nevezetes helytől mért távolság

A különböző szakterületek az üzleti logikából következően sokszor eltérő méretarányú térképről származó geometriai alapra építették az információs rendszerüket. A különböző méretarány, a sokféle üzleti logika, a szelvénybeosztás problémakörét mutatja a 5.3. ábra.

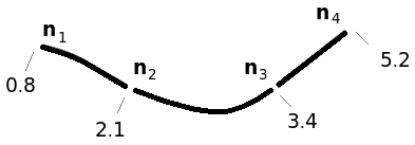
A 5.4. ábrán látható vonallánc leírható a klasszikus georeferencia által, az n_1, n_2, n_3, n_4 node-ok koordinátáival (5.1. táblázat), és ugyanazon helyek lineáris referencia (szelvényezés) szerinti leírásával (pl. az adott hely távolsága a kezdő szelvényszámtól, 5.2. táblázat).

5.1. táblázat. Az n_1, n_2, n_3, n_4 node-ok georeferencia táblázata

n_id	**X**	**Y**
n_1	625212	385261
n_2	625230	385286
n_3	625248	385266
n_4	625267	385274

5.2. táblázat. Az n_1, n_2, n_3, n_4 nodeok a lineáris referencia (szelvényezés) szerinti táblázata

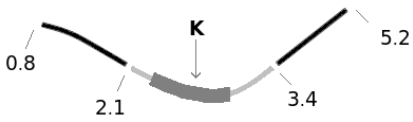
n_id	**távolság**
n_1	0.8
n_2	2.1
n_3	3.4
n_4	5.2



5.4. ábra. Egy vonalláncot alkotó node-ok (n_1, n_2, n_3, n_4), és a szelvényezés logikája szerinti ugyanazon helyek

5.3. táblázat. A **K** jelű szegmens meghatározása a lineáris referencia szerint

id	**vonal_id**	start	end
K	67	2.2	3.1

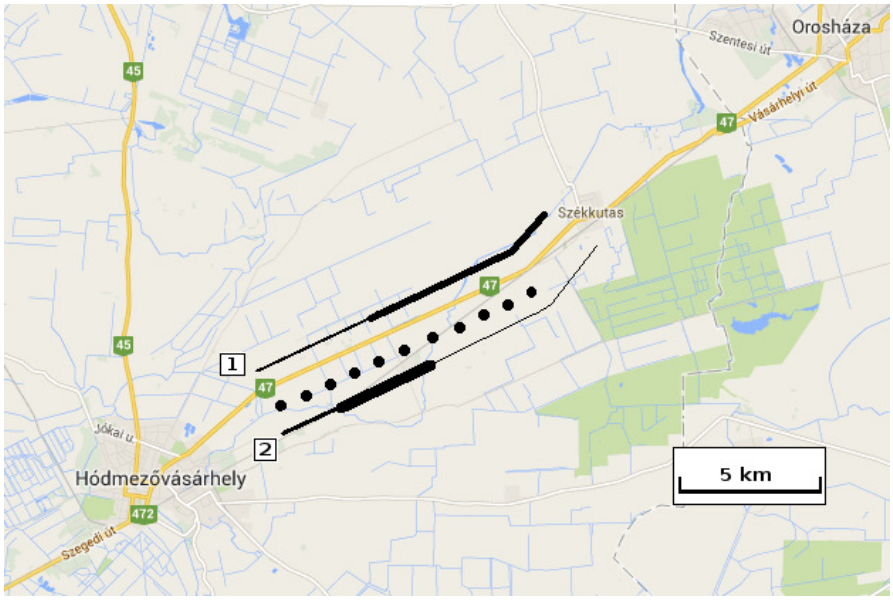


5.5. ábra. Egy vonallánc, amelyen láthatók a szelvényezés helyei, és egy olyan **K** jelű vonalszakasz, amelyet a 5.3. számú lineáris referencia táblázatban definiáltunk

5.1.2. Dinamikus szegmentálás

A rétegszemléletű megközelítésben a grafikus adatok szegmentáltsága, valamely üzleti logika szerinti szakaszoltsága a grafikus adatbázis permanens része. Az objektumorientált szemlélet előretörésével a grafikus adatok valamely (tetszőleges) attribútum szerinti dinamikus szegmentálásának lehetősége rugalmas rendszerépítést tesz lehetővé. Különösen érdekes ez a lineáris referencia szerinti logikájú rendszerekben, ahol a szegmensek megadása a lineáris referencia szerint valósítható meg (5.3. táblázat és 5.5. ábra).

Egy vonalláncokból álló hálózat tetszőleges attribútumadat szerinti szegmentálása még inkább rávilágít arra, hogy a vonalláncok szegmentálása nem alapozható a geometriai leírást jelentő node-okra. Tegyük fel, hogy szegmentálni szeretnénk, és eltérő vonalvastagsággal megjeleníteni a 5.6. ábrán látható, 47. sz. utat. Mivel az út nyílegyenes, ezért a kérdéses szakaszon nincsenek node-ok. Változnak viszont az attribútumadatok. Ahányféle leíró adat van, annyiféle szegmentálási eredmény keletkezhet. Nyilvánvaló, hogy nem lehet a szegmentálás és a megjelenítés alapja a vonal geometriáját leíró node-ok sorozata.



5.6. ábra. A Hódmezővásárhely és Orosháza közötti 47.sz. út egyik nyílegyenes szakaszának szegmentálása több attribútumadat szerint. A *1.* jelű szakaszon két szegmens van. A vékonyabb azt jelzi, hogy az út kétsávos, a vastagabb azt, hogy négysávos. A *2.* jelű szakaszon három szegmens van, amelyek az aszfaltburkolat eltérő korát mutatják (a vékony 1998 előtti, a vastagabb 1998 és 2010 közötti, míg a legvastagabb 2010 utáni) [forrás: Google Maps]

5.2. Összegzés

A fenti példákból levonható az a következtetés, hogy a vonalas objektumok geometria alapú leírása csak a topológia szempontjából tekinthető kielégítőnek. Az üzleti logika és a belőle fakadó szegmentálási kényszer nem alapozható a topologikus leírásra. Az OGC szabvány measure-nek nevezi a problémát, és egy pont x, y, z koordinátája mellé, mint opcionális negyedik dimenziót is megengedi, vagyis elvileg egy pontot akár négy koordináta is meghatározhat ($P[x, y, z, m]$).

Tovább nehezíti a helyzetet, hogy a szelvénybeosztás nem alkalmazható bármely generalizáltsági szintű térképre, ugyanis a különböző generalizáltságú térképek vonalhosszainak összege biztosan eltérő lesz. Világosan kell látnunk, hogy a térképészet (vagyis a megjelenítés) szempontjai nem egyeztethetők össze a lineáris referencia logikájával sem.

A kérdés továbbra is az, hogy egy vonallánc node alapú, topologikus leírására kialakított adatbázist alkalmassá lehet-e tenni a lineáris referencia kezelésére. Egy járható út az, ha megtartjuk a topológialeírásra szolgáló táblákat és kapcsolatrendszerüket, és létrehozunk egy olyan táblát, amely megtartja az előbbi táblák közül a node-okat tartalmazó tábla rekordjait (a példában *Points* volt a tábla neve), pontosabban rámutat ezekre a rekordokra, és kiegészül olyan rekordokkal, amelyek a lineáris referencia vonatkoztatási pontjai. Amennyiben dinamikus szegmentálási eredmények is szükségesek, akkor ezek a pontok is, mint új rekordok, bekerülnek a táblába. Így voltaképpen egy „hibrid" táblát kapunk, amely hivatkozásokat és további pontokat tartalmaz. Ilyenkor persze geometriai szempontból a bekerült új pontok szükségtelenek a topológia leírásához (ahhoz egyébként nem is használjuk fel őket), viszont minden olyan pontot tartalmaznak, amelyek a szegmentálás végrehajtásához szükségesek.

6

Formátumleírások

Ebben a fejezetben ismertetünk néhány vektorgrafikus textformátumot. A Mapinfo MIF/MID formátumát azért, mert jól követhető ebben a text fájlban a különböző geometriaelemek kezelése, valamint a poligontopológia spagetti logikájú leírása. A topologikus adatstruktúra kialakításának folyamatát MIF fájlok segítségével mutattuk be, így már csak ezért is érdemes egy kicsit részletesebben megismerkedni vele. A WKT (well known text) formátum hasonlóan a MIF-hez, textfájl. Szerepe az open source adatbázis-kezelőkben, különösen as PostGIS-ben fontos, így bemutatásra érdemes.

6.1. A Well-known formátumok

6.1.1. A well-known text geometria reprezentációja

Minden OGC szerinti geometriai objektumnak van ú.n. Well-known Text reprezentációja. A well-known text (WKT) egy jelölő nyelv (text markup language), amely vektoros geometriai adatok, térbeli referenciarendszerek és ezek közötti transzformációk leírására való. Létezik a bináris változata is (Well-known binary, WKB). E formátumok segítségével geometriai adatokat tárolhatunk relációs adatbázisokban, mint például a Postgres vagy az MS SQL server. Mindkét formátum pontos specifikációja az OGC szabványban olvasható (http://www.opengeospatial.org), általános formája következő:

$$< geometrytype > [coordinatetype][coordinatelist]$$

A WKT megengedett geometriai objektum típusai (*geometry type*) a pont (point), a vonal (line), a poligon (polygon), a TIN és a poliéder (polyhedron). Multigeometriák és geometriai gyűjtemények is megengedettek, amiket az

OGC-t bemutató fejezetben ismertettünk (15. oldal). A geometriai objektumok lehetnek $2D(x,y)$, $3D(x,y,z)$ és $4D(x,y,z,m)$ dimenziósak.

A *coordinate type* megadja, hogy a geometriának van z vagy m dimenziója. Ha itt Z-t adunk meg, akkor ez 3D-s lesz magasságadattal, ha M-t, akkor ez egy mérték lesz valamilyen referenciarendszerben mérve. HA ZM érték szerepel, akkor az mindkét adat meglétét jelöli (vagyis az objektum 4D-s lesz). Ha üresen hagyjuk, akkor azt jelöli, hogy az adat 2D-s. A *coordinate list* egy vesszővel elválasztott, duplapontos számokból álló lista, amely a geometriai objektum vertexeit adja meg. A 6.1. táblázat geometriai reprezentációkat mutat WKT-ben.

6.1.2. A well-known binary geometria reprezentációja

Minden OGC szerinti geometriai objektumnak van egy ú.n. Well-Known Binary reprezentációja, amely ugyanazt írja le, mint a well-known text, csak bináris formában. Ennek ismertetését mellőzzük. Akit mélyebben érdekel, az a részletes leírást megtalálhatja az OGC-specifikációkban.

6.1.3. A well-known text spatial reference reprezentációja

A WKT, mint fentebb említettük nemcsak geometriai objektumok, hanem a referencia-koordinátarendszerek és az ezek közötti transzformációk leírására is alkalmas. Minden entitásnak van egy nagybetűvel írt kulcsszava (pl. DATUM, UNIT, SPHEROID, stb.), amelyet kapcsos zárójelek ([]) között követ a definíció és a vesszővel elválasztott paraméterek. A 6.2. és a 6.3. táblázatokban tekintsük át ezeket.

6.2. Mapinfo MIF/MID

A Mapinfo cég a MIF/MID formátumot eredetileg az adatvesztés nélküli adatcsere miatt dolgozta ki. A MIF a Mapinfo Data Interchange Format rövidítése. Akkoriban még nem létezett a mára standarddá vált ESRI shape formátum. A létező és legnépszerűbb AutoCad DXF formátum viszont, ami szintén adatcsere célokat szolgált, akkoriban még nem volt képes adatvesztés nélküli adatexportra. Ez indokolta a MIF formátum kidolgozását. Részletes, angol nyelvű dokumentáció található a *http://mapinfo.mif* weboldalon.

A következőkben röviden áttekintjük, hogy miként kezeli a MIF a poligonokat, vonalakat, pontokat, szövegeket és egyéb attribútumadatokat és azok struktúráját, a vetületi rendszereket és grafikus adatok térbeli kiterjedését. Noha, text fájl lévén, nem adatbázis logikájú a felépítése, de nagyon alkalmas arra, hogy belőle adatbázis épüljön. Egy másik lényeges tulajdonsága,

6.1. táblázat. Geometriareprezentációk WKT-ben

Geometria	**Szöveges reprezentáció**	**Leírás**
Point	null	üres pont
Point	POINT (150 40)	egy pont
Point	MULTIPOINT((14 12),(22 28))	multipont két ponttal
Point	Point Z (150, 40, 358)	3D-s pont
Point	Point ZM (150, 40, 358, 12)	3D-s pont 12-es mérték értékkel
Line	null	üres vonal
Line	LINESTRING (14 40, 15 42, 16 45)	LineString 3 ponttal
Line	MULTILINESTRING ((176 40, 176 42) (176 42, 178 48))	multilinestring két linestringgel
Polygon	null	üres poligon
Polygon	POLYGON ((156530509 41490812, 156522356 41487473, 156519301 41490217, 156526058 41493118, 156530509 41490812))	egyszerű poligon
Polygon	MULTIPOLYGON (((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 70, 80 60, 60 60)))	multipoligon két poligonnal
Tin	TIN Z (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 0 0, 0 0 0)) ((0 0 0, 1 0 0, 0 0 1, 0 0 0)) ((1 0 0, 0 1 0, 0 0 1, 1 0 0)))	tetraéder 4 ponttal
Polyhedron	POLYHEDRON (((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)), ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))	poliéder
Geometry collection	GEOMETRYCOLLECTION (POINT(4 6), POINT(8 10), LINESTRING(14 16, 27 30))	geometriagyűjtemény, amely két pontból és egy LineStringből áll

6.2. táblázat. Matematikai transzformációk leírása WKT-ben

<math transform>	=	<param mt> | <concat mt> | <inv mt> | <passthrough mt>
<param mt>	=	PARAM_MT ["<classification name>" ,<parameter>*]
<parameter>	=	PARAMETER["<name>", <value>]
<value>	=	<number>
<concat mt>	=	CONCAT_MT [<math transform> ,<math transform>*]
<inv mt>	=	INVERSE_MT[<math transform>]
<passthrough mt>	=	PASSTHROUGH_MT [<integer>, <math transform>]

hogy minden olyan adatot tartalmaz, amely a korrekt topológia visszaépítéséhez szükséges. Topológia leírása azonban spagetti szemléletű. Sok szempontól (de nem minden szempontból) megfelel az OGC szabvány követelményeinek.

A MIF/MID formátum két fájlból áll. Az egyik a MIF fájl, amely fájl fejlécből, az adatstruktúra leírásából, a vetület megnevezéséből és paramétereiből, valamint az adatok befoglaló téglalapjának koordinátáiból áll. A MID fájl csak attribútumadatokat tartalmaz, a MIF-ben leírt adatstruktúra szerinti rendben. A MIF/MID export egy rétegből egy MIF/MID fájlpárt készít.

6.2.1. File header

A MIF fájl a következő paraméterekkel kezdődik:

```
VERSION n
Charset ,,characterSetName''
[ DELIMITER ,,<c>'' ]
[ UNIQUE n,n.. ]
[ INDEX
n,n.. ]
[ COORDSYS...]
[ TRANSFORM...]
COLUMNS n
<name> <type>
<name> <type>
.
.
DATA
```

6.3. táblázat. Koordináta-rendszer leírása WKT-ben

<coordinate system>	=	<horz cs> | <geocentric cs> |
	=	<vert cs> | <compd cs> |
	=	<fitted cs> | <local cs>
<horz cs>	=	<geographic cs> | <projected cs>
<projected cs>	=	PROJCS["<name>", <geographic cs>, <projection>, <parameter>,* <linear unit> ,<twin axes>,<authority>]
<projection>	=	PROJECTION["<name>" ,<authority>]
<geographic cs>	=	GEOGCS["<name>", <datum>, <prime meridian>, <angular unit> ,<twin axes> ,<authority>]
<datum>	=	DATUM["<name>", <spheroid>,<to wgs84> ,<authority>]
<spheroid>	=	SPHEROID["<name>", <semi-major axis>, <inverse flattening> ,<authority>]
<semi-major axis>	=	<number>
<inverse flattening>	=	<number>
<prime meridian>	=	PRIMEM["<name>", <longitude> ,<authority>]
<longitude>	=	<number>
<angular unit>	=	<unit>
<linear unit>	=	<unit>
<unit>	=	UNIT["<name>", <conversion factor> ,<authority>]
<conversion factor>	=	<number>
<geocentric cs>	=	GEOCCS["<name>", <datum>, <prime meridian>, <linear unit> ,<axis>, <axis>, <axis>, <authority>]
<authority>	=	AUTHORITY["<name>", "<code>"]
<vert cs>	=	VERT_CS["<name>", <vert datum>, <linear unit>, <axis>, ,<authority>]
<vert datum>	=	VERT_DATUM["<name>",<datum type> ,<authority>]
<datum type>	=	<number>
<compd cs>	=	COMPD_CS["<name>", <head cs>, <tail cs> ,<authority>]
<head cs>	=	<coordinate system>
<tail cs>	=	<coordinate system>
<twin axes>	=	<axis>, <axis>
<axis>	=	AXIS["<name>", NORTH | SOUTH | EAST | WEST | UP | DOWN | OTHER]
<to wgs84s>	=	TOWGS84[<seven param>]
<seven param>	=	<dx>, <dy>, <dz>, <ex>, <ey>, <ez>, <ppm>
<dx>	=	<number>
<dy>	=	<number>
<dz>	=	<number>
<ex>	=	<number>
<ey>	=	<number>
<ez>	=	<number>
<ppm>	=	<number>
<fitted cs>	=	FITTED_CS["<name>", <to base>, <base cs>]
<to base>	=	<math transform>
<base cs>	=	<coordinate system>
<local cs>	=	LOCAL_CS["<name>", <local datum>, <unit>, <axis>, ,<axis>* ,<authority>]
<local datum>	=	LOCAL_DATUM["<name>",<datum type> ,<authority>]

ahol a *VERSION* a Mapinfo verziószámát határozza meg. A *Charset* a character set megnevezését adja meg (magyar karakterek esetében ez többnyire *WindowsLatin2*. A *DELIMITER* a MID fájlban lévő adatok elválasztására szolgáló karakter adja meg (*c* értéke többnyire *tabulátor*, *vessző(,)*, vagy *pontosvessző(;)*. A default értéke a *tabulátor*. A *UNIQUE* utcahálózatoknál kap szerepet, ezért ezt nem ismertetjük. Az *INDEX* megadja, hogy mely oszlopok vannak indexelve (ez, mint tudjuk, a gyors keresés miatt lehet fontos).

A *COORDSYS* az adott réteg koordináta-rendszerét határozza meg, annak minden vetületi paraméterével. Annyiféle rendszert ír le, ahányat képes kezelni a Mapinfo. (Többek között ezért fontos a *VERSION* megadása.) Szintaktikája többféle lehet:

```
CoordSys Earth
[ Projection type,
datum,
unitname
[ , origin_longitude ]
[ , origin_latitude
]
[ , standard_parallel_1 [ , standard_parallel_2 ] ]
[ , azimuth ]
[ , scale_factor
]
[ , false_easting ]
[ , false_northing
[ , range
]
] ]
[ Affine Units unitname, A, B C, D, E, F ]
[ Bounds ( minx, miny) ( maxx, maxy) ]
```

A non-earth típushoz a következő paraméterezést adja meg:

```
CoordSys Nonearth
[ Affine Units unitname, A, B C, D, E, F ]
Units unitname
Bounds ( minx, miny) ( maxx, maxy)
```

A *TRANSFORM* záradék koordináta-transzformáció megadását is lehetővé teszi, ha ezt be akarjuk építeni a MIF fájlba. Ezen kívül még sokféle

koordináta-rendszer megadása lehetséges, azonban ennek részleteivel a továbbiakban nem foglalkozunk.

A *COLUMNS* paraméter az adatoszlopok számát adja meg. A column name az oszlop nevét, a column type az oszlop típusát, hogy az character and decimal típusú, és végül a mező hosszát (ahol ez a típusból következően szükséges). A következő mezőtípusok megengedettek:

- char (width), string típus megadott hosszal
- integer, 4 bytos egész típus
- smallint, rövid (2 bytos) egész típus, amelynek értéke -32767 és +32767 közötti értékeket vehet fel
- decimal (width, decimals), decimális, ahol a hosszat és a tizedes értékek számát adjuk meg
- float, lebegőpontos szám
- date, dátum típus
- logical, logikai típus

Lássunk egy példát:

```
COLUMNS
3
STATE char (15)
POPULATION integer
AREA decimal (8,4)
```

ahol 3 adatoszlop van. A *STATE* mező string típusú, 15 karakter hosszúsággal, a *POPULATION* mező 4 bytos egész típusú, az *AREA* mező 8 jegyű decimális szám, 4 tizedes jeggyel. Ezek a MID fájl tartalmát is befolyásolják. A string típusú adatok a MID fájlban idézőjelek között jelennek meg.

6.2.2. Data section

A geometriai adatokat tartalmazó data section a *DATA* kulcsszóval kezdődik. (Nem keverendő össze az attribútumadatokkal, amelyeket a MID fájl tartalmazza). Ha nincs grafikus adat az adott rétegben, akkor a *DATA* kulcsszó után következő sorban a *NONE* kulcsszó szerepel. Amennyiben vannak grafikus adatok is, akkor azok a következő típusúak lehetnek:

- point (pont)
- line (vonal)
- polyline (vonallánc)
- region (poligon)
- arc (ív)
- text (szöveg)
- rectangle (négyszög alakú terület)
- rounded rectangle (lekerekített sarkú, négyszög alakú terület)
- ellipse (ellipszis alakú terület)
- multipoint (pontsorozat)
- collection (gyűjtemény)

Point

A pont objektumoknak két paramétere van, az *x* és *y* koordináta, valamint opcionálisan megadható, hogy milyen szimbólummal jelenjen meg. A *shape* paraméter megadja a szimbólum alakját (pl. kör, csillag, négyszög, stb.), a *color* a színét, a *size* a méretét.

```
POINT x y
[ SYMBOL (shape, color, size)]
```

Line

A **line** objektumoknak négy paramétere van: a vonal kezdő és végpontjainak koordinátái. Opcionálisan megadható a vonalak vastagsága (*width*), mintázata (*pattern*) és színe (*color*) a *PEN* paraméter megadásával.

```
LINE x1 y1 x2 y2
[ PEN (width, pattern, color)]
```

Polyline

A **polyline** vonalak sorozatából állnak, ahol az egyik vonal végpontja azonos a következő vonal kezdőpontjával. A polyline egy meglehetősen komplex objektum, mert nemcsak vonalak, hanem vonalláncok sorozatából állnak. Ebben az esetben használjuk a *MULTIPLE* kulcsszót. Ebben az esetben szekciónként meg kell adnunk a *numpts* paramétert, amely megadja, hogy az adott szekció hány pontból áll. Opcionálisan megadhatjuk a vonal vastagságát, mintázatát, színét. Megadhatjuk továbbá a *SMOOTH* paramétert, amely arra utal, hogy megjelenítéskor simított görbét kreáljon (spline-okkal) vagy sem.

```
PLINE [ MULTIPLE numsections ]
numpts1
x1 y1
x2 y2
:
[
numpts2
x1 y1
x2 y2
]
:
[ PEN (width, pattern, color)]
[ SMOOTH ]
```

Region

A **region** objektum egy vagy több poligonból áll. A *numpolygons* paraméter adja meg, hogy hány poligonból áll a régió. A magyar szaknyelv az ilyen régiókat (poligonokat) lyukas poligonnak, az ingatlan-nyilvántartás úszó telkeknek nevezi ezeket a struktúrákat. A leggyakoribb esetben, amikor csak egy poligon van a régióban, akkor a *REGION 1* paraméterezést látjuk.

```
REGION numpolygons
numpts1
x1 y1
x2 y2
:
[
numpts2
x1 y1
```

```
x2 y2
]
:
[ PEN (width, pattern, color)]
BRUSH (pattern, forecolor, backcolor)
[ CENTER x y ]
```

A *numpnts* azt adja meg, hogy a régión belüli, soron következő poligon hány node-ból áll. Ezt követően az x, y koordinátapárok jönnek. A *PEN* paraméter a poligonok határvonalainak vastagságát, a vonal mintázatát és színét adják meg. A *BRUSH* paraméter a kitöltés mintázatát, háttér és előtér színét adja meg. (A háttérszín olyankor kap szerepet, ha átlátszóvá tesszük a mintázatot). A *CENTER* paraméter a poligon centroidjának koordinátáit adja meg.

Az **arc** (ív) meghatározásához szükséges ismerni a befoglaló téglalapja átlósan szemközti koordinátáit (x_1, y_1, x_2, y_2) az *a* kezdő, és a *b* záró szögeket (mivel az *arc* egy ellipszis egy szelete). A *PEN* a vonal szélességét (*width*), mintázatát, azaz vonaltípusát (*pattern*) és színét (*color*) adja meg.

```
ARC
x1 y1 x2 y2
a b
[ PEN (width, pattern, color)]
```

A **text** objektumok maximum 255 karakter hosszúságú stringekből állnak. Az x_1, y_1, x_2, y_2 paraméterek a text helyét határozzák meg a térképen (térképi koordinátákban). A *Font*-tal a szöveg fontkészletét adhatjuk meg, A *spacing* paraméterrel a szóközt (1, 1.5, 2), a *justify*-jal a szöveg igazítását, *Angle*-lel a szöveg irányát, *Label line*-nal pedig azt lehet megadni, hogy húzzon-e vonalat a beszúrási ponttól a szöveg kezdetéhez.

```
TEXT
,,textstring''
x1 y1 x2 y2
[ FONT...]
[ Spacing {1.0 | 1.5 | 2.0}]
[ Justify {Left | Center | Right}]
[ Angle text_angle]
[ Label Line {simple | arrow} x y ]
```

A **Rectangle**-lel egy téglalapot lehet megadni a szemközti sarokpont koordinátákkal.

```
RECT
x1 y1 x2 y2
[ PEN (width, pattern, color)]
[ BRUSH (pattern, forecolor, backcolor)]
```

A **Rounded rectangle**-lel lekerekített szélű téglalapot lehet megadni. Az *a* paraméterrel a lekerekítés mértéke szabályozható.

```
ROUNDRECT
x1 y1 x2 y2
a
[ PEN (width, pattern, color)]
[ BRUSH (pattern, forecolor, backcolor)]
```

Az **ellipse**-szel egy ellipszist lehet megadni a befoglaló téglalap koordinátáival.

```
ELLIPSE x1 y1 x2 y2
[ PEN (width, pattern, color)]
[ BRUSH (pattern, forecolor, backcolor)]
```

A **multipoint** objektum számos x, y koordinátapárból álló pontsorozatot adható meg.

```
MULTIPOINT num_points
x1 y1
x2 y2
x3 y3 ...
```

A **collection** objektum egy gyűjtemény, amely többféle objektum halmazából állhat.A *num_parts* paraméter az objektumcsoportok számát adja meg.

```
Collection format
Collection num_parts
Region
.......
Pline
......
Multipoint
..........
```

Az érthetőség kedvéért lássunk egy példát:

```
COLLECTION num_parts
Region
Pline
Multipoint
EXAMPLE:
Collection 3
Region
3
5
4.850832 10.077456
5.850832 11.077456
6.850832 13.077456
12.850832 19.077456
4.850832 10.077456
4
-5.149168 0.077456
-4.149168 1.077456
-3.149168 3.077456
-5.149168 0.077456
4
14.850832 20.077456
15.850832 21.077456
16.850832 23.077456
14.850832 20.077456
Pen (1,2,0)
Brush (2,16777215,16777215)
Center 8.850832 14.577456
Pline 3
-7.149168 0.077456
-3.149168 -2.922544
-2.149168 2.077456
Pen (1,2,0)
Multipoint
2
-6.149168 -0.922544
-5.149168 0.077456
Symbol (35,0,12)
```

6.2.3. Pen, Brush, Symbol, Font Codes

A *PEN* záradék vonalas objektumok (line, polyline, arc, poligon határoló vonal) stílusát adja meg, amely jelenti a vonalak vastagságát (width), a vo-

naltípust (pattern) és a színt (color). A vonaltípust egy szám jelöli, amely a Mapinfo vonalstílus táblájában elfoglalt hely sorszámát jelenti.

```
PEN (width, pattern, color)
```

A *BRUSH* záradék kitöltött felületi objektumok (poligonok, téglalapok, ellipszisek) kitöltését szabályozza. A *pattern* paraméter a kitöltés mintázatát, a *forecolor* a kitöltés színét, a *back color* a háttérszínt adja meg. Mintázat egy sorszámot takar, amely a Mapinfo kitöltési stílusait tartalmazó táblázatban elfoglalt helyet jelenti.

```
Brush (pattern, forecolor [, backcolor ])
```

A *SYMBOL* záradék a pontszerű objektumok megjelenési stílusát adja meg. A *shape* paraméter egy index, amely a Mapinfo pontszerű szimbólumait leíró tábla valamely sorára mutat. A *color* a szimbólum szinét, a *size* a méretét adja meg.

```
SYMBOL (shape, color, size)
```

A *FONT CODES* záradék szövegek megjelenésének leírására szolgál. Hosszas ismertetésétől eltekintünk, mert kevéssé kapcsolódik a topológia problémakörhöz. Megemlítését azonban szükségesnek tartottuk.

6.2.4. MID file

A MID fájl a MIF-ben megadott grafikai elemekhez (poligonokhoz, vonalakhoz, pontokhoz) tartozó alfanumerikus leíró adatok tárolására való. A MIF fájl headerjében megadott adatszerkezetnek megfelelő rendben tárolja az adatokat. Egy sor a MID fájlban egy rekordot jelent az adatbázisban. String típusú adatokat idézőjelek között (többi adattípus idézőjel nélkül), valamilyen, a MIF-ben megadott határoló karakterrel elválasztva tárol. Ahány grafikus objektumunk van, annyi sora van a MID fájlnak is. Egy sorban, egymás után következnek a MIF-ben megadott szerkezetű adatok, abban a sorrendben, ahogy a MIF a struktúra leírását tartalmazza. Amennyiben hiányzik a MID fájl, az attribútum nélküli grafikát jelent.

6.3. A DAT szabvány elemei

A DAT egy közel 200 oldalas, geodéziai célú magyar szabvány, amely digitális térképek szabványos tárolására és adatcseréjére használatos. A digitális

alaptérkép objektumai geometriájának leírásához szükséges elemek a pont, a vonal és a felület, valamint a felületet alkotó határ (vagy határok) és az utóbbit alkotó határvonalak. Itt szerepel még az adatféleségek adatrekordjainak megszűnésére vonatkozó időtáblázat. A geometriai alapelemek táblázataiban szerepelnek azok topológiai kapcsolatai is (melyekből egy térinformatikai rendszerben a topológiát képezni lehet), így külön topológiai táblázatokra nincs szükség.

A táblázatok tartalmazzák az adatmezők nevét, típusát és hosszát, továbbá az adatmező szöveges megnevezését és a hozzá kapcsolódó egyéb jellemzőket (mint például: kötelező vagy opcionális használat) valamint hivatkozásokat. Minden egyes táblázathoz megadásra kerül az adattábla kulcsa, az egyes adatmezőkre vonatkozó értéktartományok, illetve lehetséges értékek. Ezután következik a hivatkozó és a hivatkozott táblázatok felsorolása. A leírás végén szerepel a táblázatok karbantartásának módja. Számos esetben a táblázatok után szerepelnek még a geometriai alapelemekre vonatkozó szabályok és megjegyzések szövegei.

A DAT nem OGC kompatibilis. Noha kimeneti formátuma egy ASCII fájl, de ez a formátum nem kompatibilis a WKT formátummal. Céljait tekintve sem illeszkedik az OGC-hez, mivel az OGC szabványban csak a matematikai megalapozás, és az adatbázis-technológiai előírások szerepelnek. A DAT-ban viszont, geodéziai indíttatású szabvány lévén, számos geodéziai vonatkozású előírás is szerepel (pl. középhiba, vagy a határvonal kiemelt kezelése, számos attribútumadat kezelésének előírása, mint pl. művelési ág, iktatószám, stb.). Adatcsere-formátumként Magyarországon meghatározó jelentőségű.

A teljes szabványt itt nem ismertetjük, hanem csak azokat a részeit, amelyek a topologikus tárolás problémájához kötődnek.

6.3.1. Geometriai alapelemek és topológiájuk táblázatai

- T_PONT: Pont geometriai alapelemek táblázata
- T_VONAL: Vonal geometriai alapelemek táblázata
- T_HATARVONAL: Határvonal geometriai alapelemek táblázata
- T_HATAR: Határ geometriai alapelemek táblázata
- T_FELULET: Felület geometriai alapelemek táblázata
- T_OSSZEKOT_IV: Pontok összekötése adatainak táblázata

- T_MEGSZ_DATUMG: Geometriai alapelem adatrekordok érvényessége megszűnésének táblázata

6.3.2. Az objektumokat leíró táblázatok

- T_OBJ_ATTRAA: Vízszintes és 3D geodéziai alappontok és attribútumaik táblázata
- T_OBJ_ATTRAB: Magassági geodéziai alappontok és attribútumaik táblázata
- T_OBJ_ATTRAC: Részletpontok és attribútumaik táblázata
- T_OBJ_ATTRBA: Közigazgatási egységek és attribútumaik táblázata
- T_OBJ_ATTRBB: Közigazgatási alegységek és attribútumaik táblázata
- T_OBJ_ATTRBC: Földrészletek és attribútumaik táblázata
- T_OBJ_ATTRBE: Alrészletek, művelési ágak és attribútumaik táblázata
- T_OBJ_ATTRBF: Termőföld-minőségi osztályok és attribútumaik táblázata
- T_OBJ_ATTRBG: Egyéb önálló ingatlanok (EÖI) és attribútumaik táblázata
- T_OBJ_ATTRBH: Szolgalmi joggal érintett területek attribútumai
- T_OBJ_ATTRBI: Földminősítési mintaterek attribútumai
- T_OBJ_ATTRCA: Épületek és attribútumaik táblázata
- T_OBJ_ATTRCB: Épületek tartozékai és attribútumaik táblázata
- T_OBJ_ATTRCC: Kerítések, támfalak, földművek és attribútumaik táblázata
- T_OBJ_ATTRCD: Tereptárgyak, egyedi építmények és attribútumaik
- T_OBJ_ATTRCE: Köztéri szobrok, emlékművek, emlékhelyek és attribútumaik táblázata

- T_OBJ_ATTRDA: Közlekedési létesítmények azonosítópontjai és attribútumaik táblázata
- T_OBJ_ATTRDB: Belterületek közlekedési létesítményei és attribútumaik táblázata
- T_OBJ_ATTRDC: Külterületek közlekedési létesítményei és attribútumaik táblázata
- T_OBJ_ATTRDD: Vasutak és más kötöttpályás közlekedési létesítmények és attribútumaik táblázata
- T_OBJ_ATTRDE: Légiforgalmi létesítmények és attribútumaik táblázata
- T_OBJ_ATTRDF: Közlekedés műtárgyai (I.) és attribútumaik táblázata
- T_OBJ_ATTRDG: Közlekedés műtárgyai (II.) és attribútumaik táblázata
- T_OBJ_ATTREA: Távvezetékek, függőpályák tengelyvonalai és attribútumaik táblázata
- T_OBJ_ATTREB: Távvezetékek, függőpályák műtárgyai és attribútumaik táblázata
- T_OBJ_ATTRFA: Folyóvizek és állóvizek és attribútumaik táblázata
- T_OBJ_ATTRFB: Vízi közművek és attribútumaik táblázata
- T_OBJ_ATTRFC: Vízügyi műtárgyak és attribútumaik táblázata
- T_OBJ_ATTRGA: Szintvonalak és attribútumaik táblázata
- T_OBJ_ATTRGB: Domborzati alakzatok és attribútumaik táblázata
- T_OBJ_ATTRGC: Digitális domborzatmodell és attribútumainak táblázata
- T_OBJ_ATTRHA: Felmérési munkaterületek és attribútumaik táblázata

- T_OBJ_ATTRHB: DAT adatbázis kezelési egységek és attribútumaik táblázata
- T_OBJ_ATTRHC: Térség jellegű területek és attribútumaik táblázata
- T_OBJ_ATTRIA: EOV rendszerbe transzformált raszteres állományok és attribútumaik táblázata
- T_OBJ_ATTRIB: EOV rendszerbe transzformálható raszteres állományok és attribútumaik táblázata
- T_OBJ_ATTRIC: Egyéb, nem transzformálandó raszteres állományok és attribútumaik táblázata

Megjegyzések:

- Minden táblázatban az adatrekord érvényessége megszűnésének dátumát szerepeltetni kell (*megsz_datum* adatmező) az *F_DATUMOK* formátum szerint. Érvényben lévő adatrekord esetén a *megsz_datum* adatmező értéke nulla (0).
- Minden táblázatban az objektum előzőleg ill. legutóbb érvényes (és az adott pillanatban már érvénytelen) adatrekordjának azonosító sorszámát szerepeltetni kell abban az adatmezőben, amelyik az „elozo_" karakterekkel kezdődik és az illető táblázatban használt objektumazonosító sorszám adatmezőjének nevével folytatódik. Ha az objektumnak nincs előzőleg érvényes adatrekordja, akkor ez az adatmező a saját rekordjának sorszámát tartalmazza.
- Minden táblázat adatainak karbantartása azonos módon történik az alábbiak szerint:
  - Új rekord a soron következő azonosító sorszámmal és a *megsz_datum* 0-val való feltöltésével kerül a táblázatba.
  - Valamely objektum megszűnése, vagy valamely adatrekord bármelyik adatmezőjének megváltozása esetén az adatrekord *megsz_datum* nevű adatmezőjébe be kell írni a megszűnésnek vagy az adatrekord érvényessége megszüntetésének dátumát.
  - Objektum megszűnése esetén más teendő nincs.

  - Ha egy adatrekord érvényessége azért szűnt meg, mert valamelyik adatmezeje megváltozott, akkor a táblázatban új adatrekordot kell nyitni a soron következő azonosító sorszámmal. Ebben az érvénytelenített adatrekord változatlan adatmezőinek értékeit és megváltozó adatmezőjének új értékét kell szerepeltetni.

- A *T_OBJ_ATTRIA*, *T_OBJ_ATTRIB* és *T_OBJ_ATTRIC* táblázatok ún. háttéradatokat tartalmaznak, amelyek az MSZ 7772-1 szabvány tartalmához képest többletet jelentenek.

6.3.3. Adatminőségi jellemzők táblázatai

- T_AHASZN_REG: Tényleges adathasználat regisztrálandó adatainak táblázata
- T_ATTRBIZN: Attribútumféleségek meghatározási bizonytalanságának táblázata
- T_ATTRELTER: Attribútumértékek eltérési minőségadatainak táblázata
- T_BIZALOM: Bizalmas adatkezelés adatainak táblázata
- T_DAT_KORL: DAT-táblázatok és konkrét objektumok használatkorlátozási adatainak táblázata
- T_EREDET: Eredet adatminőségi jellemzőinek gyűjtőtáblázata
- T_HITELES: Hitelesítés és állami átvétel adatainak táblázata
- T_KONZISZT: Adatkonzisztencia jellemzőinek táblázata
- T_MUVELET_REG: DAT-ban végzett műveletek regisztrálandó adatainak táblázata
- T_OSADATAL12: Ősadatállomány minőségadatainak táblázata
- T_QGEOMETRIA: Geometriai adatok minőségének táblázata
- T_TELJESSEG: Adatok teljességét leíró táblázat

Megjegyzések:

- E Függelék táblázataiban foglaltak fogalmi tárgyalását és részleteit az MSZ 7772-1 szabvány „10. Adatminőség" fejezete írja le.
- Az adatminőségi jellemzők táblázatainak karbantartása a táblázatok értelemszerű bővítésével történik.
- A hivatkozott táblázatok nevei az „Egyéb jellemzők" oszlopban találhatók, vagy külön leírásra kerültek.
- A hivatkozó táblázatok külön leírásra kerültek.
- A táblázatokban szereplő dátumokat az *F_DATUMOK* formátum szerint kell értelmezni.

6.3.4. Az objektumok és attribútumaik osztályozásának kódtáblázatai

- T_ATTR_CSOP: Attribútumok csoportjainak kódtáblázat
- T_OBJ_CSOP: Objektumcsoportok kódtáblázata
- T_OBJ_FELS: Objektumféleségek kódtáblázata
- T_OBJ_OSZT: Objektumosztályok kódtáblázata

Megjegyzések:

- Az MSZ 7772-1 szabvány szerint az egyéb önálló ingatlanok (EÖI) a DAT-adatbázis részét képezik, de nincsenek önálló objektumcsoportba osztva. Az EÖI adatait – differenciált kezelhetőségük érdekében – a jelen szabályzat BG jelű külön objektumcsoportban kezeli (lásd a T_OBJ_CSOP és a T_ATTR_CSOP táblázatokat). Az objektumcsoportba öt objektumféleség tartozik (lásd a T_OBJ_FELS táblázatot).
- Az MSZ 7772-1 szabványhoz képest a jelen szabályzat – az állami alapadat és az alapadat jellegű adatok mellett – háttéradat használatával lehetőséget kínál a raszteres állományok kezelésére is. Ennek attribútumai az I kódú, "Raszteres háttérinformáció" elnevezésű objektumosztály (lásd a T_OBJ_OSZT táblázatot) IA, IB és IC kódú objektumcsoportjaiban találhatók (lásd a T_ATTR_CSOP és a T_OBJ_CSOP táblázatokat). Az IA három, az IB négy, az IC pedig három objektumféleséget tartalmaz (lásd a T_OBJ_FELS táblázatot).

6.3.5. Gyűjtőtáblázatok

- T_CEG: Cégek adatainak táblázata
- T_CIM: Postacímek táblázata
- T_CIM_KULFOLD: Külföldi címek táblázata
- T_FELIRAT: Magyarázó szövegek, feliratok és névrajzi megírások táblázata
- T_HELYREALL_GY: Alappontok helyreállítási adatainak gyűjtőtáblázata
- T_HELYREALL_GYIO: Iránypontok és őrpontok helyreállítási adatainak gyűjtőtáblázata
- T_HELYSZIN_GY: Alappontok helyszínelési adatainak gyűjtőtáblázata
- T_HELYSZIN_GYIO: Iránypontok és őrpontok helyszínelési adatainak gyűjtőtáblázata
- T_IRANYPONT_GY: Vízszintes alappontok iránypontjainak gyűjtőtáblázata
- T_KERESZT_GY: Vonalas létesítmények keresztezési és metszési pontjainak gyűjtőtáblázata
- T_MAS_RENDSZER_GY: Más (pl. régi) rendszerű koordináták és magasságok gyűjtőtáblázata
- T_MUVMIN_FOKOZAT: Művelési ág és minőségi osztály szerinti fokozattábla
- T_ORPONT_GY: Magassági alappontok őrpontjainak gyűjtőtáblázata
- T_SZEMELY: Személyek adatainak táblázata
- T_SZIMBOLUM: Jelkulcsi jelek elforgatási és elhelyezési adatainak gyűjtőtáblázata

6.3.6. DAT adatcsere-formátum

DAT adatcsere-formátum a digitális alaptérkép cserére felajánlott adatainak formátumát írja elő. A DAT adatcsere-formátum előírása az adattáblázatok felsorolása.

1. Minden adattáblázat változó hosszúságú rekordokból áll.
2. Minden rekord egy táblázatsor, melynek tartalmát és maximális hosszát a megfelelő adattáblázat nevéhez tartozó leírás és az adatokat követő elválasztójel szerint kell értelmezni.
3. Az adatcsere állományt kötelezően címrekorddal kell kezdeni. A címrekord az alábbi adatmezőkből áll:
   (a) az adatcsere-állomány fájljának kiterjesztés nélküli neve (AN, 8)
   (b) az adatcsere-állomány valamilyen megnevezése (AN, 22),
   (c) az adatcsere-formátum verziójának száma (AN, 6),
   (d) az adatátadó szervezet megnevezése (AN, 25),
   (e) az adatátadásért felelős személy neve (AN, 20),
   (f) az adatátvevő szervezet megnevezése (AN, 25),
   (g) az adatátvételért felelős személy neve (AN, 20),
   (h) az adatcsere-állomány adatcserehordozóra való felírásának dátuma (N, 8).
4. Két adattáblázat adatrekordjait a soron következő táblázat nevét tartalmazó rekord választja el, amely rekord változó hosszúságú a táblázat nevének hosszától függően.
5. Nem feltétlenül része az adatcsere-állománynak az olyan adattáblázat neve, amelynek egyetlen rekordja sem szerepel az adatcsere-állományban.
6. A rekordok maximális hossza egy-egy adattáblázaton belül egyenlő az adattáblázat leírásában szereplő adatmezők hosszainak és az adatokat követő * határoló jelek számának összegével.
7. Ha az adatcsere-állományban üres (NULL) adatmező szerepel, akkor ezt a határoló jel kitevésével szükséges jelölni.

8. A teljes adatcsere-állomány egyetlen ASCII fájlban található. A fájl neve az adatcsere-állomány rövid (8-karakteres) azonosítója, amely megegyezik a (3) a) pont szerinti megnevezéssel. Kiterjesztése: .dat.

9. Az adatcsere-állomány konkrét tartalma szempontjából a csereobjektumok halmaza a meghatározó, vagyis hogy mely konkrét objektumok (objektumelőfordulások) képezik az adatcsere tárgyát.

10. Az adatcsere-állomány kötelezően előírt tartalmát képezik a csereobjektumoknak az objektumleíró adattáblázatokból származó rekordjai és az azok által hivatkozott táblázatokban érintett rekordok (a kódtáblázatok esetében legtöbbször maguk a táblázatok, teljes tartalommal).

11. Az adatcsere-állomány részét nem kötelezően képezhetik azok az adattáblázatok (ill. rekordjaik), amelyeknek egyetlen rekordjára sincs (ill. amelyekre nincs) közvetlen vagy közvetett hivatkozás az objektumleíró táblázatok rekordjai között.

12. A numerikus adatmezők maximális hossza – az N jelölést vessző után követő szám – magában foglalja az adat egész és tizedes jegyeinek számát, a tizedes pontot és az esetleges előjelet is.

13. Az alfanumerikus adatmezők maximális hossza az AN jelölést vessző után követő számmal egyezik meg.

14. Az adatcsere-formátum minden adata után * határolójelet kell tenni (a táblázatnevek után is).

15. Az adatcsere-formátum minden rekordja „kocsi vissza, soremelés" karakterekkel (ASCII kódjuk 13 és 10) fejeződik be. Ezért a rekord belsejében (pl. egy szövegben) ezek a karakterek nem fordulhatnak elő.

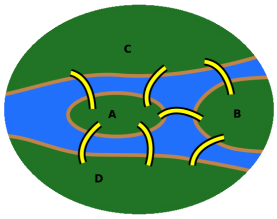
7

Gráfelmélet-összefoglaló

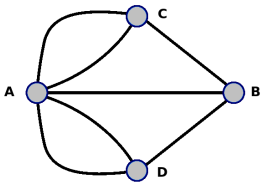
Az első gráfelméleti munka a Szentpétervári Tudományos Akadémia közleményeiben jelent meg 1736-ban. A szerző Leonhard Euler (1707-1783) svájci matematikus, aki ekkor az akadémia meghívására Oroszországban dolgozott. Königsberg (ma Kalinyingrad) a Balti-tenger közelében a Pregel-folyó két partján fekszik. A folyón itt két sziget is van, amelyeket hidak kötnek össze a partokkal, amint az 7.1. ábrán látható. Akkoriban a folyó A és B szigeteit hidak kötötték össze egymással és a partokkal is (C, D). A königsbergiek kedvenc kérdése volt, hogy valaki otthonról indulva tehet-e olyan sétát, hogy minden hídon pontosan egyszer menjen át, és a séta végén hazaérjen.

Euler teljes általánosságban oldotta meg a feladatot. Egy olyan ábrát készített, melyben a városrészeknek pontok, a hidaknak a pontokat összekötő vonalak (nem feltétlenül egyenesek) felelnek meg (7.2. ábra).

Az ilyen pontokból és vonalakból álló alakzatokat gráfoknak nevezzük. A pontok a gráf csúcsai (node, vertex), a vonalak a gráf élei (edge). A kérdés tehát így fogalmazható meg: valamely csúcsból kiindulva bejárhatjuk-e a



7.1. ábra. A königsbergi hidak sematikusan ábrázolva



7.2. ábra. A königsbergi hidak gráfja

gráf éleit úgy, hogy minden élen pontosan egyszer haladjunk át, és végül a kiindulási pontba érjünk vissza?

Ha valaki a kérdéses útvonalon halad, és eljut valamely csúcsba, akkor onnan, más útvonalon, ki is kell lépnie, és végül utolsó lépésként belép a kezdőpontba. Így minden pontba a sétálónak ugyanannyiszor kell belépni, mint kilépni. A feladat megoldhatóságához tehát szükséges, hogy minden pontban páros számú él találkozzon. Ez a königsbergi hidak problémája esetében nem teljesül, ezért a feladat nem oldható meg.

A feladat megoldásában lényeges volt az egy csúcsban találkozó élek száma. Ezt a számot a továbbiakban a csúcs fokának, fokszámának nevezzük.

Jelöljük a G gráf (7.3. ábra) pontjainak halmazát $V(G)$-vel. A pontokat általában kis- vagy nagy betűkkel $u, v, \ldots$, vagy akár egyszerűen számokkal jelöljük. A G gráf élei halmazának szokásos jelölése $E(G)$.

$$V(G) = \{v_1, v_2, v_3, v_4, v_5\}$$

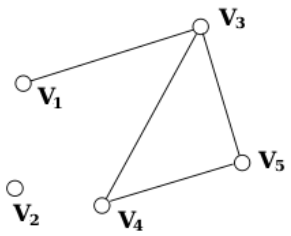
és

$$E(G) = \{(v_1, v_3), (v_3, v_4), (v_3, v_5), (v_4, v_5)\}.$$

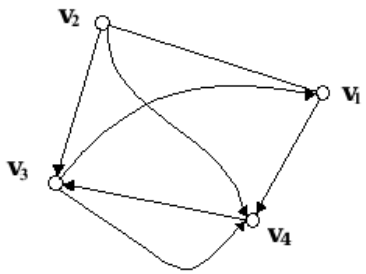
Egy élt két végpontja megadásával jelölünk. Amennyiben a gráf irányítatlan, úgy a végpontok neveinek sorrendje lényegtelen.

Irányított gráfok

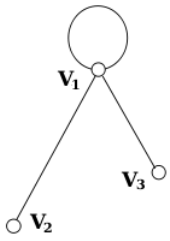
Az ún. irányított gráfok esetében az élek sorrendje lényeges. Az éleket egyenes szakasznak, vonallláncnak, vagy görbe vonalnak is ábrázolhatjuk (7.4. ábra).



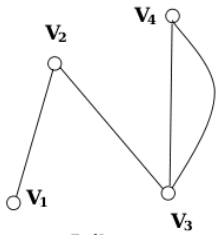
7.3. ábra. A G gráf csúcsai és élei



7.4. ábra. Példa irányított gráfra



7.5. ábra. Példa hurkot is tartalmazó gráfra: $V(G) = \{v_1, v_2, v_3\}; E(G) = \{(v_1.v_1), (v_1, v_2), (v_1, v_3)\}$



7.6. ábra. Példa többszörös élt is tartalmazó gráfra

Hurok

Előfordulhat olyan él is, melynek mindkét végpontja ugyanaz a pont. Az ilyen élt huroknak nevezzük (az 7.5. ábrán ilyen a v_1 pont).

Többszörös élek

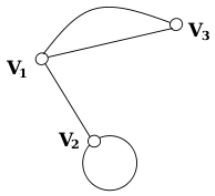
Két csúcs között több élt is húzhatunk, ezt többszörös élnek nevezzük. Ilyenek az 7.6. ábrán a v_3 -t és a v_4 csúcsokat összekötő élek. A gráfok ábrázolásakor két él az ábrán metszheti egymást, de ez a metszéspont a gráfnak nem csúcsa.

Szomszédsági mátrix

A G gráfhoz hozzá tudunk rendelni egy őt leíró mátrixot. $i = 1,2,\ldots,m$ a mátrix sorainak száma, $j = 1,2,\ldots,n$ a mátrix oszlopainak száma. Ha a mátrixnak m sora és n oszlopa van, $m \times n$ típusú mátrixnak nevezzük. A mátrixban levő a_{ij} számok a mátrix elemei. Az a_{ij} elem az i-edik sor j-edik oszlopának eleme. Ha $m = n$, a mátrixot négyzetesnek mondjuk.

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{1n} \\ \vdots & & & \\ a_{m1} & a_{m1} & \ldots & a_{mn} \end{pmatrix}$$

Nézzük meg a három csúcspontú G gráfot. A gráfot ismerjük, ha meg tudjuk mondani, hogy melyik pont melyik ponttal van éllel összekötve, és hányszoros éllel. Hozzunk létre egy mátrixot, ahol írjuk be mindegyik mezőbe az azt meghatározó csúcsokat összekötő élek számát. Az 7.7. ábrán látható G



7.7. ábra. A G gráf szomszédsági viszonyait az (7.1) számú szomszédsági mátrix írja le

gráf szomszédsági mátrixának nevezzük az (7.1) számú mátrixot. E mátrix ismeretében meg tudjuk rajzolni a gráfot.

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 0 \end{pmatrix} \tag{7.1}$$

Általában egy n vertexű gráf szomszédsági mátrixa az $A = (a_{ij})$ egy $n \times n$-es mátrix, melyben a_{ij} a v_i és v_j csúcsokat összekötő élek száma, $i, j = 1, 2, \ldots n$. Így az A elemei nem negatív egész számok. A főátlóban levő elemek összege a gráfban levő hurkok számát adja. Világos az is, hogy az irányítatlan gráf szomszédsági mátrixa szimmetrikus a főátlóra, vagyis $a_{ij} = a_{ji}$, míg az irányított gráf mátrixa nemszimmetrikus (pl. 7.4. ábra). Ennek szomszédsági mátrixa az (7.2) mátrix.

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \tag{7.2}$$

Ha egy gráf pontjait más sorrendben indexezzük, akkor a szomszédsági mátrixban bizonyos sorok és oszlopok felcserélődnek, míg a gráf ugyanaz marad. A mátrix adatszerkezeti szempontból a programozási nyelvekben jól ismert kétdimenziós tömb.

A gráf egy csúcsában találkozó élek számát a csúcs fokának nevezzük. Ez egy nem negatív egész szám (amely 0 is lehet). Ez esetben a pont izolált pont, nincs összekötve egyetlen más ponttal sem (pl. az 7.3. ábrán a v_2 pont).

Mivel minden él két ponthoz – vagy hurok esetében ugyanazon ponthoz kétszer – csatlakozik, így a fokszámok összege az élek számának kétszerese. A gráf fokainak összege páros szám, tehát a páratlan fokú pontok száma páros.

Élmátrix

A gráfok megadásának másik módja az élmátrix, mely a gráfok éleit tárolja, például egy $G[1..E]$ egydimenziós tömbben, ahol E az élek száma. Matematikai szempontból ez sorvektor, vagyis olyan mátrix, melynek csak egy sora van. A G elemei számpárok, pl. $G[k] = (v_i, v_j)$, vagyis a k-adik él a v_i és v_j csúcsokat köti össze. Itt nehezebb megállapítani két pont összekötöttségét, viszont könnyebb, pl. az összes élt felsorolni. Ritka kapcsolati mátrix esetén ez a forma könnyebben programozható.

Súlyozott gráf

Az élekhez lehet számot is rendelni ez a súlyozott gráf, mátrixa a súlymátrix. Ilyen, pl. egy csővezetékrendszer, ahol megadjuk az egyes csövek áteresztő képességét. Úthálózatokra vonatkoztatva, az élhez rendelt súly lehet akár az élen való áthaladás ellenállása, amely így egyirányú utcák esetben a csúcspontok közötti távolság (megengedett haladási irányban történő mozgás esetén), vagy végtelen nagy (a kötelező haladási iránnyal szemben).

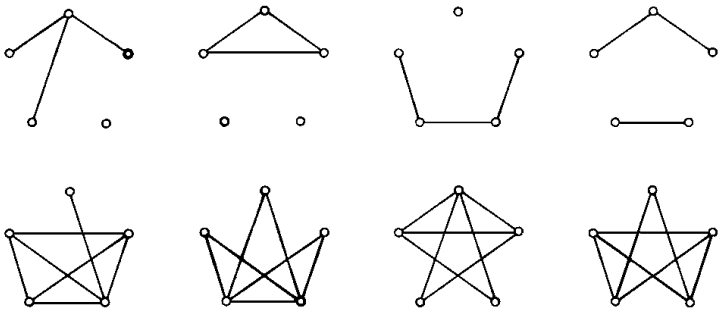
Komplementer gráf

Egy adott gráfhoz rendeljünk egy másik gráfot úgy, hogy a pontok halmaza mindkét gráfban ugyanaz, de a második gráfban két pontot akkor és csak akkor kötünk össze, ha az első gráfban a két pont nincs összekötve. Ezt a gráfot az eredeti G gráf komplementer gráfjának nevezzük és $\overline{G}$-sal jelöljük (7.8. ábra).

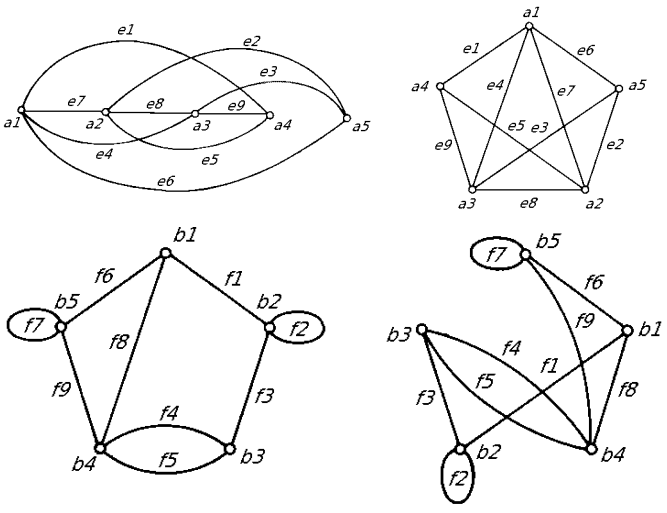
Izomorfia Ha a G_1 gráf minden pontjának és élének megfeleltethető a G_2 gráfnak pontosan egy pontja, illetve éle, valamint a G_2 gráf minden pontjának és élének megfeleltethető a G_1 gráfnak pontosan egy pontja, illetve éle, akkor a gráfok izomorfak (7.9. ábra).

Teljes n-gráfok

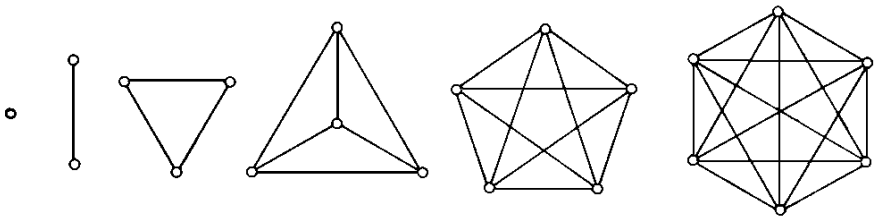
Vizsgáljuk most meg egy egyszerű gráf pontjainak fokszámát. Csak az az eset érdekes, ha a gráfnak legalább két pontja van. Ha a gráf n-pontú, akkor mindegyik pont fokszáma $0, 1, 2, \ldots, n-1$ lehet. Ha egy pont fokszáma 0,



7.8. ábra. Gráfok (felső sor) és komplemenseik (alsó sor) [8]



7.9. ábra. Példák izomorf gráfokra [8]



7.10. ábra. Példa teljes gráfokra [8]

akkor az a pont nincs más ponttal összekötve, ha pedig egy pont fokszáma $n-1$, akkor ez a pont minden más ponttal össze van kötve. Ezért nem fordulhat elő, a gráfban 0-fokú és $(n-1)$-fokú pont is legyen. A kettő közül legfeljebb egyik lehet. Így azt kapjuk, hogy mivel n pontunk van, de a fokszámra csak $n-1$ különböző eset lehetséges, ezért legalább egy fokszámnak legalább kétszer kell előfordulnia. Ha egy gráfnak van legalább két pontja, akkor van két azonos fokú pontja.

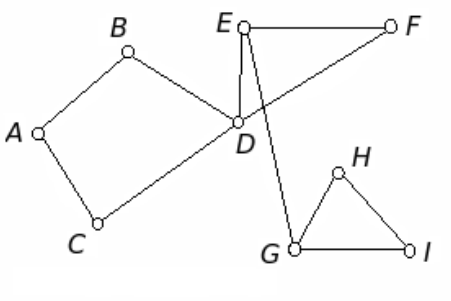
Hány éle van az n-pontú egyszerű gráfnak, amelynek bármely két különböző pontját él köti össze? Az ilyen gráf neve teljes n-gráf (7.10. ábra). A teljes n-gráf minden pontjának foka $n-1$. A teljes n-gráf éleinek száma $n(n-1)/2$. Egy gráf és komplementere teljes gráfot alkotnak.

Egyszerű gráfok

Az általános gráfban két pont többszörösen is össze lehet kötve, illetve a hurokél is megengedett. A továbbiakban csak olyan gráfokról lesz szó, melyekben nincs sem többszörös él, sem hurok. Az ilyen gráfokat egyszerű gráfoknak nevezzük. Az egyszerű gráfok szomszédsági mátrixában csupa 0 és 1 áll, és a főátló minden eleme 0.

Út, vonal, kör, séta

Ábrázolja a G gráf egy városrész úthálózatát. A gráfelméletben útnak nevezzük az éleknek olyan egymáshoz csatlakozó sorozatát, mely egyetlen szögponton sem halad át egynél többször (7.11. ábra). Ez természetesen azt is jelenti, hogy minden élen legfeljebb egyszer haladunk át. Konkrét feladatokban legtöbbször utakat keresünk, mert azok száma véges, és általában rövid úton szeretnénk egyik pontból a másikba jutni.



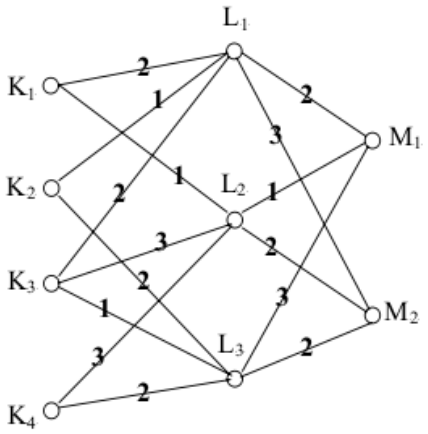
7.11. ábra. Az ábrán lévő gráf esetében *ABDEGH* út, de az *ABDEFDEGIH* nem. AB, BD, DE, EF, FD, DC vonal, de nem út. Az AB, BD, DC, CA élsorozat kör, vagyis a kiindulási pontba visszatérő vonal. [8]

Vonalnak nevezzük a gráf éleinek olyan egymáshoz csatlakozó sorozatát, melyben minden él különböző, ám egyes pontokon többször is áthaladhatunk. Például AB, BD, DE, EF, FD, DC vonal, de nem út (7.11. ábra). Vonalról van szó, ha egy részgráfot a toll felemelése nélkül, minden élt csak egyszer érintve meg tudunk rajzolni.

Még általánosabb fogalom a séta. Sétának nevezzük gráfban az $AB, BD, \ldots$ egymáshoz csatlakozó élek sorozatát (7.11. ábra), ahol az élek és pontok nem feltétlenül különbözők. Ha két pont között van séta, akkor biztos, hogy út is van. Az AB, BD, DC, CA élsorozat a kiindulási pontba visszatérő vonal, melyben minden él és pont, a kezdő és végponttól eltekintve, egyszer szerepel. Az ilyen élsorozatot körnek nevezzük. Ha egyik pontból a másikba két út is vezet, akkor a gráfban biztosan van kör.

Legyen A egy gráf szomszédsági mátrixa, akkor a $K(k) = A^k$. Ekkor az A mátrix k-adik hatványaként kapott mátrix $K(k)_{ij}$ eleme, a gráf i pontjából a j pontjába vezető k hosszúságú séták számát adja. Itt a séta hosszán az i ponttól a j csúcspontig történő haladás közben bejárt élek számát kell érteni. Valóban, maga a szomszédsági mátrix is ezt mutatja: mely szögpontból melyik szögpontba lehet egyetlen élen haladva eljutni. Nyilvánvalóan abba a szögpontba, amelyikkel közvetlenül össze van kötve éllel.

A következő példa Balogh [2] jegyzetéből származik. Nehezebb átlátni Nehezebb átlátni a mátrix önmagával való többszöri beszorzása eredményeként kapott mátrix elemeinek értelmét. A mátrixok szorzása megértésének



7.12. ábra. Vasúti példa [2]

egyszerűsítése céljából lássunk egy könnyen érthető példát, s az általánosítás érdekében tekintsünk el a mátrixok négyzetes voltától. Legyen K, L, M három ország, melyek egyes városait vasút köti össze.

Legyenek K országban K_1, K_2, K_3, K_4; L országban L_1, L_2, L_3 és M országban M_1 és M_2 városok. Az egyes városok között közlekedő vonatok számát írjuk rá a városokat jelző pontokat összekötő szakaszokra. Mátrixok segítségével is ábrázolhatjuk az utazási lehetőségeket. Lássuk először a K-ból L-be közlekedő vonatokat (ha K_i és L_j között nem közlekedik vonat, akkor 0-t írunk a megfelelő helyre).

Ábrázoljuk a K-ból L-be közlekedő vonatokat az 7.1. táblázatban, és **A**-val jelölve mátrixként (7.3 mátrix). A mátrix i-edik sorának j-edik eleme azt mutatja, K_i-ből hány vonat megy L_j-be.

Tekintsük át az L-ből M-be menő vonatokat a 7.2. táblázattal. A megfelelő mátrixot jelöljük **B**-vel (7.4 mátrix). Ebben a mátrixban az i-edik sor j-edik eleme azt mutatja, hogy L_i városból hány vonat megy M_j városba.

Az 7.12. ábra szerint K országból M országba csak valamelyik L-beli városon át juthatunk el. A kérdés az, hogy hány különböző módon juthatunk el K_1-ből M_1-be. Az 7.12. ábra alapján K_1-ből L_1-be menni az kétféleképpen lehetséges. Innen ugyancsak kétféleképpen utazhatunk M_1-be. Ez tehát 2×2 lehetőség. K_1-ből L_2-be egy vonatjárat van, L_2-ből M_1-be ugyancsak

7.1. táblázat. A K_i-ből L_j-be közlekedő vonatok táblázata

	L_1	L_2	L_3
K_1	2	1	0
K_2	1	0	2
K_3	2	3	1
K_4	0	3	2

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 3 & 2 \\ 0 & 3 & 2 \end{pmatrix} \tag{7.3}$$

7.2. táblázat. A M_i-ből L_j-be közlekedő vonatok táblázata

	M_1	M_2
L_1	2	3
L_2	1	2
L_3	3	2

$$\mathbf{B} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 3 & 2 \end{pmatrix} \tag{7.4}$$

7.3. táblázat. A K_i-ből M_j-be közlekedő vonatok táblázata

	M_1	M_2
K_1	c_{11}	c_{12}
K_2	c_{21}	c_{22}
K_3	c_{31}	c_{32}
K_4	c_{41}	c_{42}

egy. Így ezen az úton csak 1×1 módon juthatunk M_1-be. K_1-ből nem vezet vonat L_3-ba, így K_1-ből L_3-an át M_1-be jutni 0 lehetőség. (Írhatnánk 0×3-at is, mert L_3-ból három lehetőség van M_1-be jutni.) Összesen tehát $2 \times 2 + 1 \times 1 + 0 \times 3 = 5$ különböző lehetőségünk van arra, hogy K_1-ből M_1-be utazzunk.

A feladatot általánosabban megfogalmazva, határozzuk meg az 7.3. táblázat elemeit, ahol c_{ij} az a szám, amely azt mutatja, hogy K_i-ből hány különböző módon juthatunk el M_j-be. Tudjuk, hogy $c_{11} = 5$. Határozzuk meg például a c_{22}-t!

Írjuk egymás mellé az **A**, **B** és a **C** mátrixokat.

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 3 & 2 \\ 0 & 3 & 2 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 3 & 2 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \\ c_{41} & c_{42} \end{pmatrix} \tag{7.5}$$

Már ismerjük a c_{11} meghatározásának módját: $c_{11} = 2 \cdot 2 + 1 \cdot 1 + 0 \cdot 3 = 5 = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31}$, és a feladat alapján hasonló módon számítjuk ki a c_{22}-t: $c_{22} = 1 \cdot 3 + 0 \cdot 2 + 2 \cdot 2 = 7$, vagyis $c_{22} = a_{21} \cdot b_{12} + a_{22} \cdot b_{22} + a_{23} \cdot b_{32}$. Vegyük észre, hogy c_{11}-et az **A** mátrix első sora elemeinek a **B** mátrix első oszlopának megfelelő elemeivel való szorzása, és a kapott három kéttényezős szorzatot összeadása által kapjuk meg. Ugyanezt láthatjuk a c_{22} esetén is.

Két mátrixból ily módon előállított új mátrixot az előbbiek szorzatának nevezzük. Itt nem részletezve a mátrixok szorzásának szabályait (mivel a függelék későbbi részeiben erre kitérünk), definiáljuk két mátrix szorzatát:

Az **A** $m \times n$ típusú és a **B** $n \times p$ típusú mátrix **AB** szorzatán azt a **C** $m \times p$ típusú mátrixot értjük, melyben

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj},$$

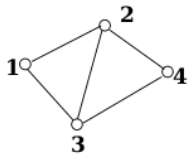
ahol $i = 1,2,\ldots m; j = 1,2,\ldots,p$.

Mint látjuk, a szorzás definíciója nem bármely két mátrixra vonatkozik, hanem csak olyanokra, ahol az első tényezőnek annyi oszlopa van, ahány sora a másodiknak. A példában szereplő **A** és **B** mátrixok szorzata **C** mátrix (7.6 formula).

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 3 & 2 \\ 0 & 3 & 2 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 3 & 2 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 5 & 8 \\ 8 & 7 \\ 10 & 14 \\ 9 & 10 \end{pmatrix} \tag{7.6}$$

A mátrixszorzás fontos tulajdonsága, hogy nem szükségszerűen kommutatív, vagyis $\mathbf{AB} \neq \mathbf{BA}$, ugyanakkor asszociatív, tehát $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$. A gráfokat leíró szomszédsági mátrixok hatványozása szempontjából ez fontos megállapítás. Könnyen belátható, hogy a négyzetes szomszédsági mátrixok hatványozása önmagával történő többszöri beszorzással egyszerűen megoldható.

Vegyünk egy négypontos egyszerű gráfot, melynek szomszédsági mátrixa **A** (7.13. ábra). Jelöljük a csúcspontjait rendre arab számokkal, így ezek azonosíthatók a gráf mátrixának sor- és oszlopszámaival.



7.13. ábra. A négypontos **A** gráf és szomszédsági mátrixa (7.7)

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{7.7}$$

Számoljuk ki a $K(2) = \mathbf{A}^2$ és a $K(3) = \mathbf{A}^3$ mátrixokat (7.8. formula).

$$K(2) = \mathbf{A^2} = \begin{pmatrix} 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 1 \\ 1 & 2 & 3 & 1 \\ 2 & 1 & 1 & 2 \end{pmatrix}; \qquad K(3) = \mathbf{A^3} = \begin{pmatrix} 2 & 5 & 5 & 2 \\ 5 & 4 & 5 & 5 \\ 5 & 5 & 4 & 5 \\ 2 & 5 & 5 & 2 \end{pmatrix} \tag{7.8}$$

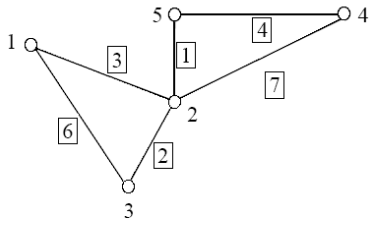
Ezen mátrixok elemei a sor és oszlopszámoknak megfelelő vertexek között való közlekedés közben két, illetve három élen történő áthaladások számát adják. A kapott mátrixok ugyanúgy, mint az irányítatlan gráfok szomszédsági mátrixai, szimmetrikusak a főátlóra. Vegyük észre, hogy itt nem a bejárható utak számát, hanem a séták számát kaptuk meg. Amennyiben az i-edik pontból a j-edikbe vezető $k = 1,2,\ldots n-1$ lépéses utak számát szeretnénk megtudni, rögzíteni kell a séta közben érintett pontokat, s csak azokat tartani meg, melyekben minden érintett vertex csak egyszer fordul elő.

Nézzünk egy térinformatikához közeli példát. Cél a legrövidebb utak meghatározása bizonyos csomópontok között. Ábrázoljuk a csomópontokat és az őket összekötő utakat súlyozott gráffal, ahol az élekhez számok vannak rendelve. A konkrét feladattól függően a gráf lehet irányítatlan vagy akár irányított. Többféle algoritmus létezik a probléma megoldására (Warshall-algoritmus, Dijkstra-algoritmus, stb.). Példaként ismerkedjünk meg a Warshall-algoritmussal. Legyen adott egy egyszerű öt vertexű súlyozott gráf (7.14. ábra). Szomszédsági mátrixát jelöljük A-val. A ∞ jelek az él nélküli pontpárokat jelölik.

Vezessünk be egy új mátrixműveletet:

$$A[k]_{ij} = min\,\{A[k-1]_{ij}, A[k-1]_{ik} + A[k-1]_{kj}\},$$

ahol a $\mathbf{A[k]}$ a mátrixok sorozata $k = 1,\ldots n$-ig számítandó, ahol n az $\mathbf{A}$ mátrix mérete $(n \times n)$, és $\mathbf{A[0]} = \mathbf{A}$. Vagyis a súlymátrixból kiindulva (amikor $k = 0$) rendre előállítjuk az $\mathbf{A[1]}, \mathbf{A[2]}, \ldots, \mathbf{A[n]}$ mátrixokat a fenti képlet szerint, vagyis az új – magasabb sorszámú mátrix – ij indexű elem értékéül az előző mátrix ij-edik eleme, valamint az ik és kj indexű elemek összegének összehasonlítása után a kisebbiket vesszük. Az $\mathbf{A[n]}$-re kapott eredmény mátrix a legrövidebb utakat adja meg. Esetünkben $n = 5$, vagyis a súlymátrix alapján az ötödik menetben $k = 5$-re a (7.10. számú) mátrixot kapjuk.



7.14. ábra. *G* egy egyszerű, öt vertexű gráf, súlyozott élekkel, melynek szomszédsági mátrixa **A** (7.9 formula)

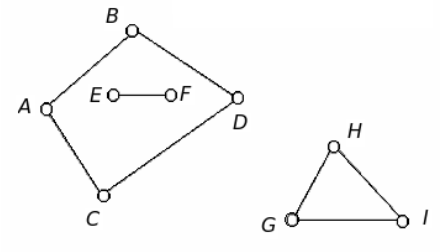
$$\mathbf{A} = \begin{pmatrix} 0 & 3 & 6 & \infty & \infty \\ 3 & 0 & 2 & 7 & 1 \\ 6 & 2 & 0 & \infty & \infty \\ \infty & 7 & \infty & 0 & 4 \\ \infty & 1 & \infty & 4 & 0 \end{pmatrix} \tag{7.9}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 3 & 5 & 8 & 4 \\ 3 & 0 & 2 & 5 & 1 \\ 5 & 2 & 0 & 7 & 3 \\ 8 & 5 & 7 & 0 & 4 \\ 4 & 1 & 3 & 4 & 0 \end{pmatrix} \tag{7.10}$$

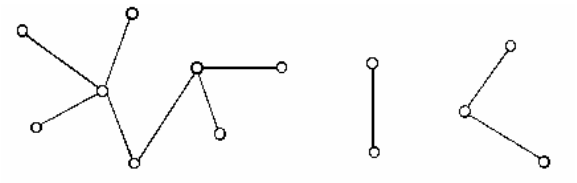
Ez tehát a keresett útmátrix. A harmadik sor negyedik oszlopának eleme $\mathbf{A}[\mathbf{5}]_{34} = 7$ például azt jelenti, hogy a gráf 3-as vertexéből a 4-esbe vezető legrövidebb út hossza 7 egységnyi (3-2-5-4 pontokon át vezet).

Összefüggő gráfok

Egy gráfot összefüggőnek mondunk, ha bármely pontjából bármely másik pontjába eljuthatunk úton (elég azt megállapítani, hogy egy tetszőlegesen kiválasztott pontjából az összes többihez van-e út). Az eddig vizsgált gráfok mind összefüggőek voltak, míg az 7.15. ábrán lévő gráf nem összefüggő. Ha egy gráf nem összefüggő, akkor van olyan pontja, melyből nem vezet út minden másik pontba. A gráf egy darabja az összes olyan pontok (és az őket összekötő élek) halmaza, melyek kölcsönösen elérhetők úton. Ezt a gráf egy összefüggő komponensének nevezzük.



7.15. ábra. Nem összefüggő gráfok

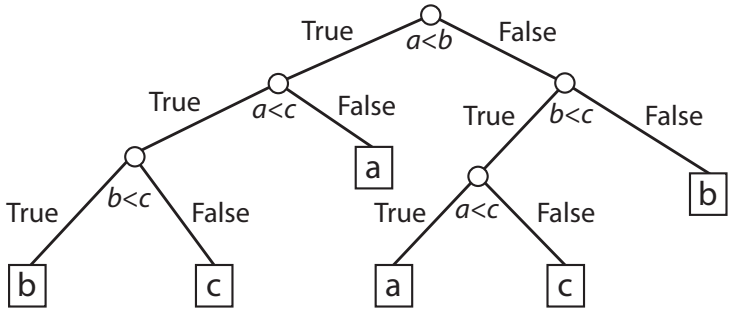


7.16. ábra. Fák és erdők [8]

Fák, erdők

Az összefüggő, körmentes egyszerű gráfokat fának nevezzük. Az olyan nem összefüggő gráfot, melynek összefüggő komponensei fák, erdőnek nevezzük (7.16. ábra).

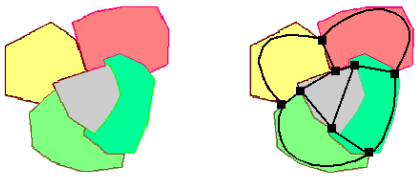
- Fában két pont között pontosan egy út létezik. Létezik, mert a gráf összefüggő, és pontosan egy, mert ha kettő lenne, kör is lenne.
- Ha fához hozzáteszünk egy élt, már nem lehet fa. Az (ij) él hozzávételével ugyanis kört kapnánk, hiszen eddig is volt a két csúcs között út, és most lett még egy.
- Fából egy élt elhagyva már nem lehet fa. Ha ugyanis fa lenne, akkor abba az iménti élt visszatéve megint csak fát kapnánk (az eredetit), ami az előző bekezdés szerint nem lehet.
- Úgy is fogalmazhatnánk, hogy adott számú vertexű fa a lehető legkevesebb élt tartalmazza, ami szükséges az összefüggőséghez.



7.17. ábra. Bináris fa

Az 7.16. ábrán látható fák mindegyikében van elsőfokú pont. Vajon minden esetben így van ez? Válasszuk a fa egy tetszőleges pontját, nevezzük A-nak (ez lehet elsőfokú vagy akármilyen más), és induljunk el valamely irányba, a ponthoz illeszkedő valamely élen (legalább egy ilyen él van, mivel a gráf összefüggő). Mivel nincs kör, így soha nem térhetünk vissza olyan pontba, ahol egyszer már jártunk, ezért a gráf végessége miatt az útnak egyszer vége lesz. Nem tudunk tovább menni, azaz ez a pont elsőfokú. Legyen ez a B pont. Most ebből a B elsőfokú pontból indulva, ugyanilyen módon, ismét lesz vége a körmentes gráfnak, kapunk egy újabb elsőfokú pontot, amely lehet az A (ha elsőfokú volt), vagy valamely más, például a C. Ebből következik, hogy egynél többpontú fában van legalább két elsőfokú pont.

Meg tudjuk-e mondani, hogy egy n-pontú fának, hány éle van? Az 7.16. ábrán látható esetekben a 8-pontúnak 7 éle van; a 2-pontúnak 1 éle van; a 3-pontúnak 2 éle van. Általában az n-pontú fának $n-1$ éle van. $n=1$-re igaz az állítás, az 1-pontú fának 0 éle van. Tegyük fel, hogy igaz k pontú fára, és igazoljuk, hogy akkor igaz $k+1$-pontúra is. Növeljük a k-pontú fát egy ággal. Mivel a fa körmentes, összefüggő, egyszerű gráf, így a fa bármely választott pontjából csak egy új, eddig nem létező pontba húzhatunk egy élt. Eggyel nőtt a fa pontjainak száma, és eggyel nőtt az élek száma is, így az állítás igaz.



7.18. ábra. Egy gráf és duálisa

A fák alkalmasak arra, hogy számrendszereket szemléltessenek Különösen nagy szerepük van az úgynevezett bináris fáknak, melyek a kettes számrendszert szemléltetik. Itt egy vertex egy kérdést jelent, melyre igaz (*i*) vagy hamis (*h*) a válasz, és e szerint megyünk tovább a bal vagy a jobb oldali élen. Példaként ábrázoljuk bináris fával azt az algoritmust, mely három szám közül kiválasztja a középsőt (7.17. ábra). Itt a kérdés a fa csúcspontja mellett feltüntetett számpárok viszonyának igaz vagy hamis volta. A fa valamely téglalappal jelzett elsőfokú pontja tartalmazza a választ (a konkrét számadatoktól függően).

Síkba rajzolható gráf

Egy gráfot akkor nevezünk síkba rajzolhatónak, ha lerajzolható úgy a síkban, hogy az élei nem metszik egymást. Euler szerint egy összefüggő, síkba rajzolt gráf csúcsainak és tartományainak száma (beleértve a külső, nem korlátos tartományt is) 2-vel nagyobb az éleinek számánál.

Duális gráf

Egy síkba rajzolható gráf duális gráfja alatt azt a gráfot értjük, aminek csúcsai az eredeti gráf tartományai, és azok a csúcsok vannak összekötve, amik megfelelői szomszédosak voltak (7.18. ábra). Síkba rajzolható gráf duálisa is síkba rajzolható gráf.

Irodalomjegyzék

[1] D. Arctur – M. Zeiler: „Designing Geodatabases”, ESRI, 2004

[2] Balogh Sándor: „Logikai elemek és kapcsolások; Gráfeméleti alapfogalmak”, 2004, http://mek.oszk.hu/02900/02901/

[3] R. Burke: „Getting to know ArcObjects”, ESRI Press, 2003

[4] J. Celko: „SQL felsőfokon”, Kiskapu Kiadó, 2002, ISBN 963 9301 20 5

[5] Elek István: „Bevezetés a geoinformatikába”, ELTE Eötvös Kiadó, 2006

[6] Elek István: „Térképek, adatbázisok, információs rendszerek”, ELTE Eötvös Kiadó, 2011, ISBN 978 963 312 039 2

[7] ESRI Shapefile Technical Description – ESRI White Paper, July 1998

[8] Hajnal P.: „Gráfelmélet”, Poligon Kiadó, 2003

[9] J. Han – M. Kamber: „Adatbányászat”, Panem Könyvkiadó, 2004, ISBN 963 545 394 9

[10] Laurini – Thomson: „Fundamentals of Spatial Information Systems”, Academic Press, 1992

[11] Lovász L. – Pelikán J. – Vesztergombi K.: „Diszkrét matematika”, TypoTEX Kiadó, 2006

[12] A. Mitchell: „The ESRI Guide to GIS Analysis”, ESRI, 1999

[13] MacEachren – Taylor: „Visualization in Modern Cartography”, Elsevier, 2005

[14] P. Rigaux – M. Scholl – A. Voisard: „Spatial Databases with Application to GIS”, Morgan Kaufmann Publishers, 2002

[15] Hanan Samet: „The Design and Analysis of Spatial Data Structures", Addison-Wesley, 1994, ISBN 0-201-50255-0

[16] R. Tomlinson: „Thinking About GIS", ESRI Press, 2007

Webes források:

[17] Nguyen Thai Binh: GDAL, PostgreSQL- PostGIS, http://tarsadalominformatika.elte.hu/tananyagok/gdal/index.html, 2013. ELTE TÁMOP 4.1.2.A/1-11/1-2011-0056

[18] *OpenGIS*® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, http://www.opengeospatial.org/standards/sfs/06-103r4_Implementation_Specification_for_Geographic_Information_-_Simple_feature_access_-_Part_1_Common_Architecture_v1.2.1.pdf, Copyright©2010 Open Geospatial Consortium Inc., All Rights Reserved

[19] Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option, http://www.opengeospatial.org/standards/sfs/06-103r4_Implementation_Specification_for_Geographic_Information_-_Simple_feature_access_-_Part_2_SQL_Option_v1.2.1.pdf, Copyright©2010 Open Geospatial Consortium Inc., All Rights Reserved

[20] The Open Source Geospatial Foundation, http://www.osgeo.org/, 2014

[21] Cartographic Projections Library, https://trac.osgeo.org/proj/ 2014

[22] Arnulf Christl: Introduction of Spatial Data Management with PostGIS, 2005, Bonn, Germany, http://www.mapbender.org/presentations/Spatial_Data_Management_Arnulf_Christl/html/img0.html