

Pojmy související s GIS

Matematická kartografie - disciplína zabývající se převodem zemského povrchu do roviny.

Ekvidistantní zobrazení - nezkresluje délky určité soustavy. Zpravidla touto soustavou bývají zeměpisné poledníky nebo rovnoběžky. Nelze definovat ekvidistantní zobrazení, které by nezkreslovalo žádné délky.

Ekvivalentní zobrazení - nezkresluje plochy, zkreslení úhlů je však zde poměrně značné, což se projevuje zejména ve tvarech ploch.

Konformní zobrazení - ponechává nezkreslené úhly, značně jsou však zde zkreslovány plochy.

Zemský povrch - geometricky složitý útvar geoid, proto je modelován rotačním elipsoidem, který je určen hlavní a vedlejší poloosou. Pro různá kartografická zobrazení je jsou používány různé elipsoidy, v poslední době je snaha používat elipsoid WGS84.

	Bessel	Hayford	Krasovskij	IAG 1967	WGS-84
rok	1841	1909	1940	1967	1984
a [m]	6377397.155	6378388	6378245	6378160	6378137
b [m]	6356078.963	6356911.946	6356863.019	6356774.516	6356752.314

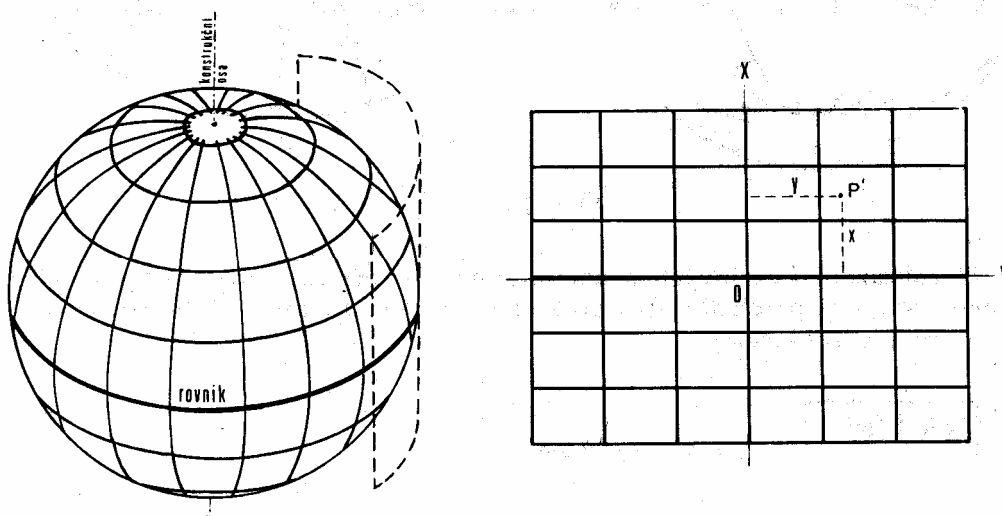
Geodetické datum: model zemského tělesa (koule, elipsoid..), jeho umístění (orientace) vůči zemskému tělesu a datum určení.

Geografické souřadnice – určení polohy bodu na ploše elipsoidu pomocí zeměpisné šířky φ a zeměpisné délky λ . Šířka φ je definována jako úhel mezi normálou k ploše elipsoidu a rovinou rovníku ($-90^\circ, 90^\circ$) k severnímu a jižnímu pólu. Délka λ je úhel mezi rovinou základního poledníku (meridiánu) a poledníku daného bodu ($0^\circ, 360^\circ$) nebo ($-180^\circ, 180^\circ$).

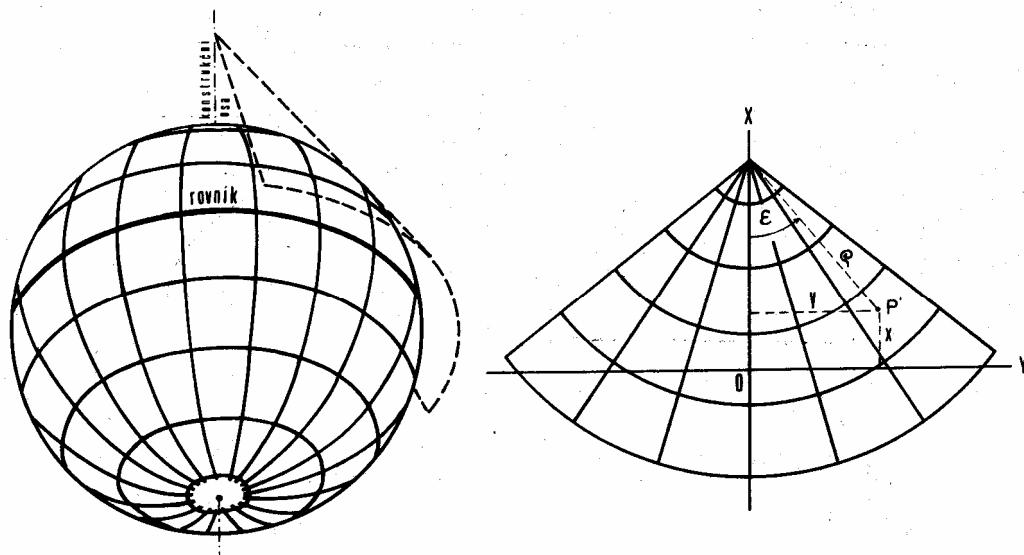
Geocentrické souřadnice X, Y, Z - prostorový souřadný systém s počátkem ve středu elipsoidu, osa X je vložena do průsečíku rovníku a roviny základního (nultého) poledníku, osa Z spojuje střed elipsoidu a severní pól a osa Y leží v rovině rovníku otočena o 90° proti směru hodinových ručiček od osy X .

Rovinné souřadnice – určení polohy v rovině pomocí dvojice rovinných souřadnic X, Y v ortogonálním souřadném systému. V některých zobrazeních (zobrazení UTM) se používá symbolika E, N (Easting, Northing) tj. rovinné souřadnice narůstající k východu resp. k severu.

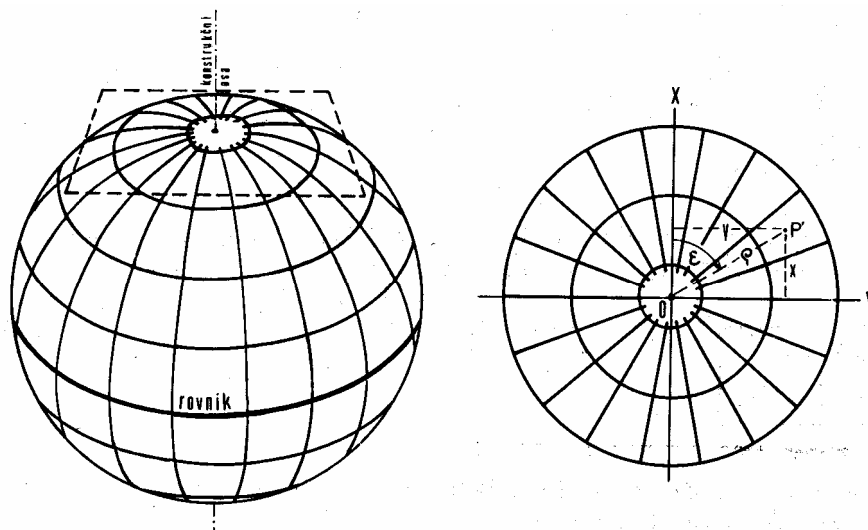
Základní typy převodu geografických do rovinných souřadnic:



Válcové zobrazení



Kuželové zobrazení



Azimutální zobrazení

Souřadný systém – je soubor těchto údajů :

- Geodetické datum (elipsoid, referenční bod, datum určení)
- Souřadný systém geografických souřadnic φ, λ , (včetně volba základního poledníku)
- Zobrazovací rovnice do rovinných souřadnic
- Souřadný systém rovinných souřadnic X, Y

Příklad : (<http://gis.zcu.cz/kartografie/konference2001/sbornik/veverka/veverka-referat.htm>)

Civilní souřadnicový systém S-JTSK je určen – Besselovým elipsoidem z roku 1841 s referenčním bodem Herrmanskogel, zeměpisné délky se určují od ferrského poledníku, zobrazovací rovnice dvojitého konformního kuželového zobrazení v obecné poloze (Křovákovo zobrazení) s volbou délkového faktoru 0.9999 pro snížení vlivu délkového zkreslení.

Vojenský souřadnicový systém S-42 je určen – Krasovského elipsoidem z roku 1942 s referenčním bodem Pulkovo, zeměpisné délky se měří od Greenwiche, zobrazovací rovnice Gaussova-Krügerova zobrazení s opakovatelností vždy pro šestistupňové poledníkové pásy, vložení osy X vždy do obrazu středového poledníku příslušného pásu, s úpravou souřadnice Y přičtením konstanty 500 km a dále předřazení čísla pásu (3 nebo 4) před posunutou souřadnicí Y . Od r. 2005 je nahrazen WGS84 – zobrazení UTM (Universal Transversal Mercator)

Souřadný systém WGS 84 - World Geodetic System 1984 je globální geocentrický geodetický systém pevně spojený se zemským tělesem. Počátek a orientace jeho os X, Y, Z jsou realizovány pomocí 12 pozemských stanic se známými přesnými souřadnicemi, které nepřetržitě monitorují dráhy družic systému GPS-NAVSTAR. Systém byl původně definován Ministerstvem obrany USA pro obranné účely, dnes je celosvětově používanou technologií prostorové lokalizace.

Cassini-Soldner – Válcové zobrazení na Zachově elipsoidu s referenčními body Sv. Štěpán, resp. Gusterberg. Definováno v Rakousko Uherské monarchii pro stabilní katastr v měřítkách 1:2800 a 1:1440 (dodnes používaná).

Transformace souřadných systémů:

Postup 1:

- Zdrojové souřadnice $[X', Y']$ převedeme na geografické souřadnice zdrojového systému $[\varphi, \lambda]$
- $[\varphi, \lambda]$ korigujeme do cílových geografických souřadnic $[\varphi', \lambda']$, $\varphi' = \varphi + \Delta\varphi$, $\lambda' = \lambda + \Delta\lambda$. $\Delta\lambda$ a $\Delta\varphi$ určíme podle znalosti identických bodů ve zdrojovém a cílovém zobrazení (transformační klíč).
- $[\varphi', \lambda']$ převedeme do cílového rovinného zobrazení

Postup 2:

- Ze znalosti identických bodů ve zdrojovém a cílovém rovinném zobrazení $[X_i, Y_i] \rightarrow [X_i', Y_i']$ určíme koeficienty polynomiální transformace a tuto potom použijeme pro jednotlivé body. Používáme polynomy do 3. stupně, vyšší stupeň vede k nestabilitě řešení (omezený počet platných cifer)

- Mapování - vytváření map měřeními nebo fotogrammetrickým mapováním pomocí geodetických základů - bodů geodetických sítí.
- Dálkový průzkum Země (DPZ) - získávání informací o zemském povrchu a jeho blízkém okolí pomocí snímacích zařízení (kamery, skenery) umístěných v letadlech nebo družicích.
- Topografická mapa - je grafická prezentace (zobrazení) části zemského povrchu se standardizovaným obecným obsahem (voda, lesy, komunikace..)
- Měřítko mapy a úroveň územní podrobnosti obsahu geografického informačního systému
- Tématická mapa - zobrazení geografických dat a jevů v topografickém podkladu pomocí grafické reprezentace prostorových dat: bodů, linií a areálů. Metody reprezentace: bodové značky, lokalizované kartodiagramy, kartodiagramy, symbolologie čar, kartogramy.

Mapové dílo v ČR

Mapy velkých měřítek do 1:5000

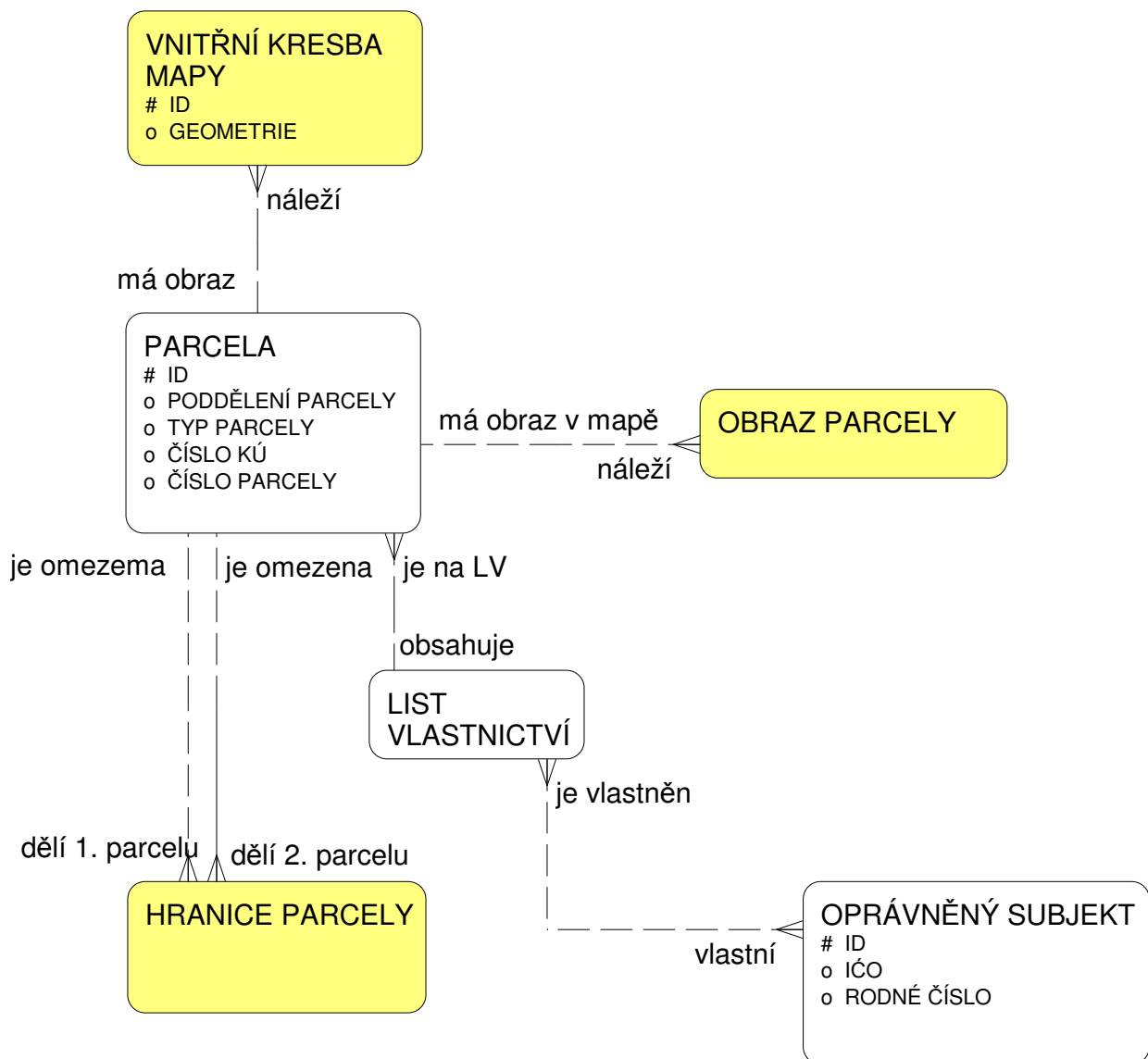
- katastrální mapy (mapy stabilního katastru) v systému Cassini-Soldner (počátek Gusterberg v Čechách, Sv. Štěpán na Moravě) v sáhových měřítkách 1:2880, 1:1440, 1:720 (měřítko je odvozeno ze vztahu 1 jitro - 1600 čtverečních sáhů - je zobrazeno jako čtvereční palec), ale v dekadických měřítkách
- katastrální mapy v S-JTSK (systém jednotné trigonometrické sítě katastrální - Křovák), měřítko 1:1000 ve městech (intravilán), 1:2000 v extravilánu vznikaly po roce 1928
- Státní mapa odvozená v měřítku 1:5000, systém JTSK, obsah: vlastnické hranice, polohopis (vnitřní kresba)
- Digitální katastrální mapa - mapu vedenou digitálně postupně vytvářejí katastrální úřady

Státní mapové dílo velkých měřítek v České republice vznikalo v průběhu dvou století. Mapové dílo je charakteristické svou rozmanitostí a rozdílnou kvalitou (především vzhledem k přesnosti a aktuálnosti mapy). Tento stav je způsoben programovým rušením vlastnických vztahů v minulosti a několika (neúspěšnými) pokusy geodetů mapové dílo sjednotit.

Mapy středních měřítek 1 : 10000 až 1 : 200 000

- Základní mapa středního měřítko - v měřítkách 1:10000, 1:25000, 1:50000, 1:100000, 1:200000 s obsahem topografické mapy
- Topografická mapa GŠ ČSA, měřítko 1:25000 (v některém území i 1:10000)

Příklad 1 - Informační systém o nemovitostech

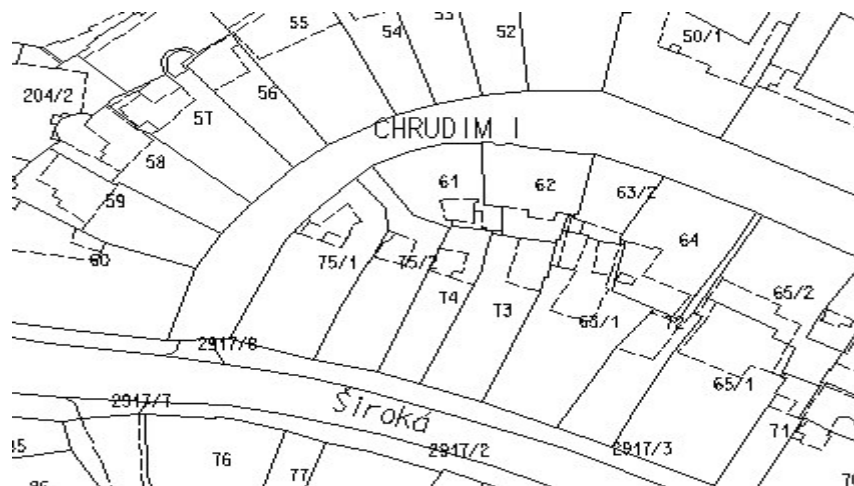


Uvažujme „klasický“ informační systém o nemovitostech s datovým modelem v relačním databázovém systému, entity systému jsou navrženy v E-R diagramu. Klasický IS je schopen reagovat na dotazy typu:

- kdo vlastní parcelu ..?
- jaké parcely vlastní osoba ...?; jakou cenu mají parcely, které vlastní osoba ...?

GIS jsou navrhovány tak, aby byly schopny reagovat na dotazy typu:

- kde se nalézá parcela ...?
- jaké typy parcel se nalézají v daném regionu ...?
- Jakou výměru parcel vlastní daný Oprávněný subjekt

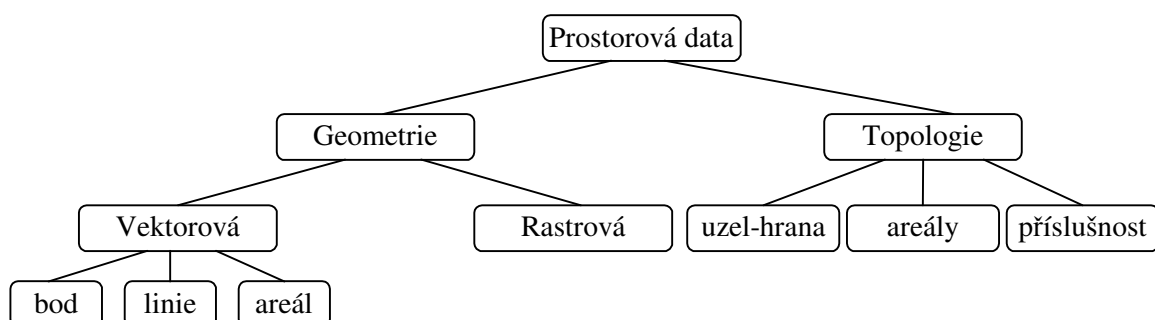


GIS je jakýkoliv manuálně nebo počítačově založený soubor postupů užívaných k ukládání a manipulování geograficky vztažených dat. *Geograficky vztažená data* mají dvě složky:

- fyzikální rozměr respektive třídu (průměrná výška stromů v lese, počet obyvatel města, šířka silnice respektive typ sídla, typ vegetace, geomorfologický typ, apod.)
- prostorovou lokalizaci ve vztahu ke zvolenému souřadnému systému (polární souřadnice, souřadnice ve zvoleném systému kartografického zobrazení)

Typy GIS – tradiční dělení

- Land Information System (LIS), Land Related Information System (LRIS), územně orientovaný informační systém - speciální případ GIS v podrobnosti velkého měřítka, který zahrnuje vlastnické vztahy (hranice parcel a informace o vlastnících parcel)
- Geoinformační systém - systém pracující s daty, která lze lokalizovat v území, ale ne vždy je lze považovat za geografická (umístění vodovodního šoupátka, dopravní značky).
- Grafický informační systém - systém pracující s obrazovými daty, která nemá smysl lokalizovat v nějakém (jednotném) prostoru.
- V posledních letech toto dělení ztrácí smysl po geoinformačních systémech je požadována komplexní funkcionality.



Vektorová data - reprezentují objekty pomocí datových struktur, jejichž základní položkou je bod 2-rozměrného spojitého (euklidovského) prostoru. Termínem “spojitý” myslíme spojitý, až na technické omezení použité počítačové aritmetiky.

Rastrová data - podmnožina 2D prostoru je pravidelně rozdělena (většinou čtvercovou) sítí, každý element této sítě je nositelem tématické části (geografické) informace. Prostorová lokalizace je určena indexem elementární složky sítě, popřípadě jeho zobrazením do cílového (kartografického) souřadného systému.

Grid data - Základem této reprezentace je opět pravidelná síť položená na 2D prostor. Rozdíl oproti rastrovým datům spočívá v tom, že tématická část informace je získávána na základě předem definované sítě, kterou je rozděleno zájmové území.

Topologie - vymezuje vztahy mezi entitami (objekty) systému, aniž by obsahovala umístění objektu v prostoru. Například informační systémy o spojení míst silniční sítě nevyžadují přesné umístění uzlů v prostoru, pracují pouze s relací na množině všech uzlů.

Jak reprezentovat prostorovou složku dat v GISech?

Povšimněme si objektů na *obr. 1*. Je vidět, že fragment mapy obsahuje prostorové informace těchto typů.

- bod
- lomená čára (linie)
- oblast (areál)

(Texty a značky budeme považovat také za bodové mají svůj referenční bod a velikost textu/značky je stále stejná, nemění se s měřítkem).

```
typedef struct
{
    long x;
    long y;
}
pointT;
```

Pevná řádová čárka je výhodná, vyhneme se problémům typu „dva různé body mají nulovou vzdálenost“, ale na druhé straně nelze počítat jen v celočíselné aritmetice.

```
double pointDist
(
    pointT *p1,
    pointT *p2
)
{
    double r;

    r=sqrt ((p2->x-p1->x) * (p2->x-p1->x) +
            (p2->y-p1->y) * (p2->y-p1->y) );
    return (r);
}
```

Špatně!!!

```
typedef struct
{
    long double;
    long double;
}
DPointT;
```

```
double DPointDist
(
    DPointT *p1,
    DPointT *p2
) ...
```

Datové sklady prostorových GIS

Pro fyzickou reprezentaci je možné použít vlastních datových struktur a ukládat je přímo ve file systému operačního systému. Přes nesporné výhody tohoto přístupu, jako je optimalizace uložení prostorové složky informace, převažují nevýhody, zejména aplikační závislost.

Není jednotný standard ukládání vektorové geometrie.

Nejpoužívanější veřejné formáty:

Shape file (systém ARC/INFO fy. ESRI)

Geograficky vztažená informace je obsažena ve trojici souborů

- *.shp prostorová informace
- *.shx prostorový index
- *.dbf popisná informace a topologické vazby

Základní podporované geometrické primitivy:

Point	bod
MultiPoint	body
MultiLine	lomené čáry
Polygon	Areál

Vše 2D a 3D varianty.

Definice neobsahuje symbologii (barva, síla, styl linií, výplň polygonu). Tu obsluhuje aplikace na základě popisných informací.

DGN file, norma IGDS (Intergraph, Bentley)

Geometrická informace je obsažena v souboru *.DGN, soubor sám o sobě nenese popisnou informaci, ta je uložena v relační databázi (nebo *.dfb souboru), DGN soubor obsahuje pouze tzv. „link“ = identifikace v souboru/databázi.

Základní geometrické primitivy:

CELL_HEADER_ELM	Hlavička buňky
LINE_ELM	Úsečka
LINE_STRING_ELM	Lomená čára
SHAPE_ELM	Polygon
TEXT_ELM	Text
TEXT_NODE	Textový uzel
CURVE_ELM	Křivka
CMPLX_STRING_ELM	Komplexní linie
CMPLX_SHAPE_ELM	Komplexní polygon
ELLIPSE_ELM	Elipsa
ARC_ELM	Oblouk
POINT_STRING_ELM	Body
BSPLINE_ELM	B-spline
DIMENSION_ELM	Kóta

(komplexní linie se mohou skládat z různých segmentů – např. lomených čar a oblouků).

Definice obsahuje symbologii geometrických primitiv.

Typ CELL může opět obsahovat buňku.

Podobné vlastnosti mají i ostatní CAD formáty (DXF, DWG..)

ORACLE 7.x (Spatial Data Option)

Geometrie je reprezentována čtyřmi tabulkami:

_SDOLAYER	- obsahuje služební údaje pro prostorovou indexaci
_SDODIM	- obsahuje rozsah pro jednotlivé dimenze geometrie
_SDOGEOM	- obsahuje vlastní geometrii
_SDOINDEX	- obsahuje prostorové indexy objektů v SDOGEOM

možné typy geometrie jsou: bod, lomená čára a areál,

Jednalo se o první pokus o standardizaci geometrie – ten se vlivem denormalizace uložení souřadnic ukázal jako slepá ulička v současné době není rozvíjen.

ORACLE 8x, 9x – datový typ SDO_GEOMETRY dále rozlišitelný podle typu na:

- UNKNOWN_GEOMETRY	- neznámá geometrie
- POINT	- bod
- LINESTRING	- lomená čára
- POLYGON	- areál (oblast)
- COLLECTION	- kolekce geometrií
- MULTIPOINT	- body
- MULTILINESTRING	- více linií
- MUTIPOLYGON	- více areálů

Línie jsou tvořeny sekvencemi bodů a kruhových oblouků.

Typ COLLECTION nemůže obsahovat typ COLLECTION.

Definice neobsahuje symbologii geometrických primitiv.

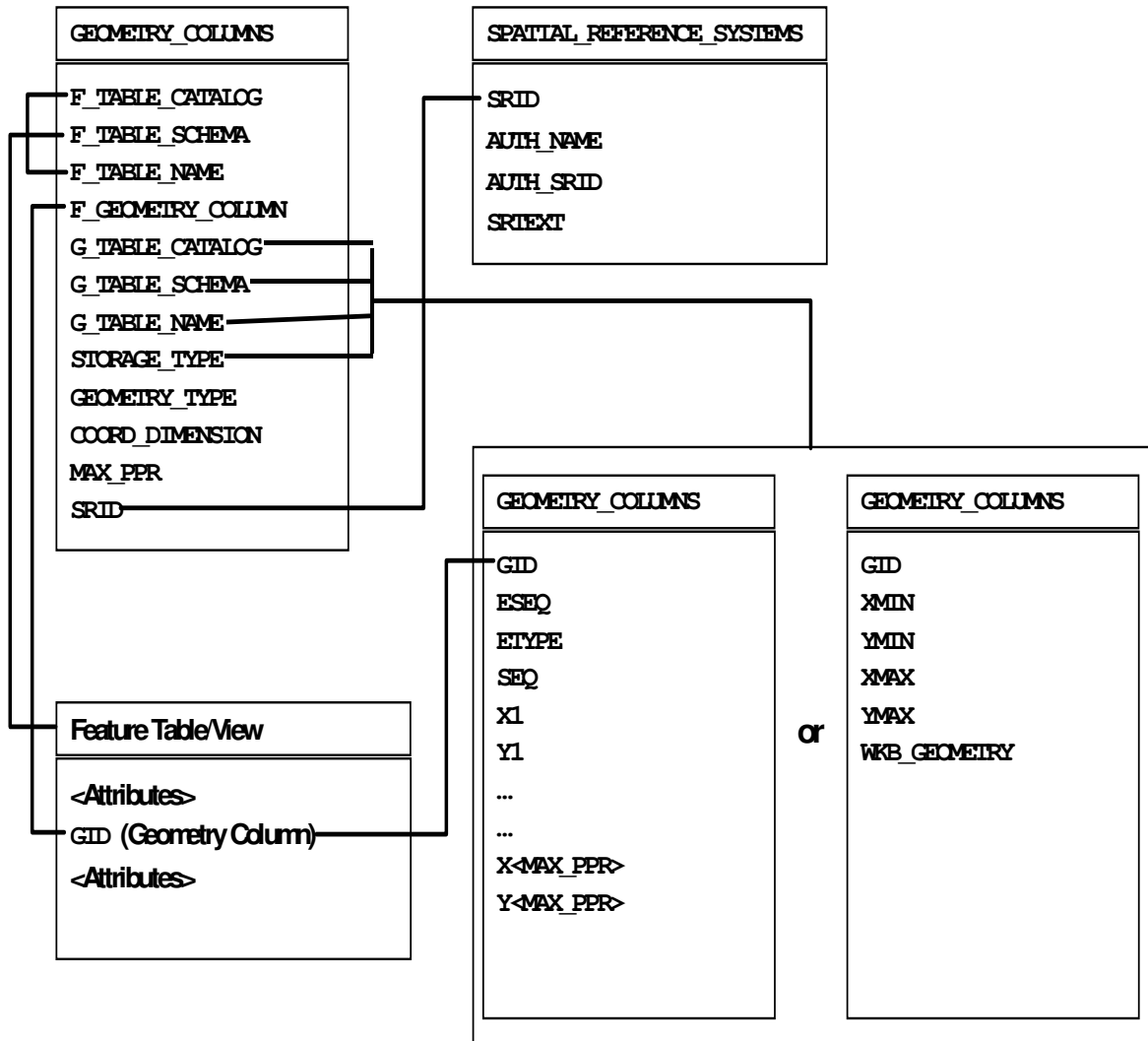
Pokus o standardizaci uložení prostorové složky:

Open GIS Consortium, Inc. – sdružení soukromých, veřejných organizací (universit ..) se zájmem vybudování „standardu“ struktur a služeb v prostorových datech.

Our Vision is a world in which everyone benefits from geographic information and services made available across any network, application, or platform.

Our Mission is to deliver spatial interface specifications that are openly available for global use.

OpenGIS® Simple Features Specification For SQL



```

// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
    double x;
    double y;
};

LinearRing {
    uint32 numPoints;
    Point points[numPoints];
}

enum wkbByteOrder {
    wkbXDR = 0,           // Big Endian
    wkbNDR = 1           // Little
Endian
};

```

```

WKBPoint {
    byte        byteOrder;
    uint32      wkbType;           // 1
    Point      point;
};

WKBLineString {
    byte        byteOrder;
    uint32      wkbType;           // 2
    uint32      numPoints;
    Point      points[numPoints];
};

WKBPolygon {
    byte        byteOrder;
    uint32      wkbType;           // 3
    uint32      numRings;
    LinearRing  rings[numRings];
};

```

```

WKBMultiPoint {
    byte        byteOrder;
    uint32      wkbType;           // 4
    uint32      num_wkbPoints;
    WKBPoin    WKBPoints[num_wkbPoints];
};

```

```

WKBMultiLineString {
    byte        byteOrder;
    uint32      wkbType;           // 5
    uint32      num_wkbLineStrings;
    WKBLineString WKBLineStrings[num_wkbLnStrgs];
};

```

```

wkbMultiPolygon {
    byte        byteOrder;
    uint32      wkbType;           // 6
    uint32      num_wkbPolygons;
    WKBPolygon  wkbPolygons[num_wkbPolygons];
}

```

```

WKBGeometry {
union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBGeometryCollection  collection;
    WKBMultiPoint           mpoint;
    WKBMultiLineString     mlinestring;
    WKBMultiPolygon        mpolygon;
    }
};

WKBGeometryCollection {
    Byte                byte_order;
    uint32              wkbType;                // 7
    uint32              num_wkbGeometries;
    WKBGeometry        wkbGeometries[num_wkbGeoms];
}

```

Základní datové typy WKB neumožňují vykreslit mapu

- symbologie geometrických elementů
- reprezentace bodových prvků
- texty (velikost, rotace, font ...)

Rekurzivní definice WKBGeometry je nutná a vyžaduje tento fakt zohlednit při vývoji aplikací!!

Příklad – rekurzivní zpracování geometrie

```
int processWKGB(byte *ge)
{
    int i, j, nP;
    int newPos, size;
    byte *pg;

    pg=(byte *)ge;

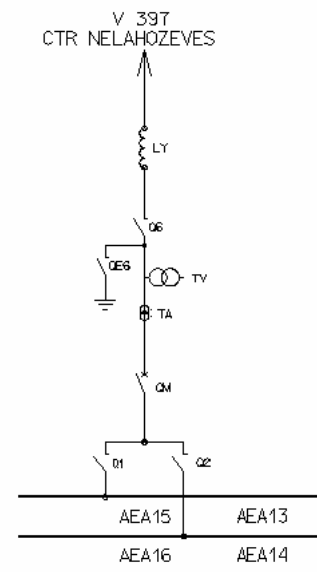
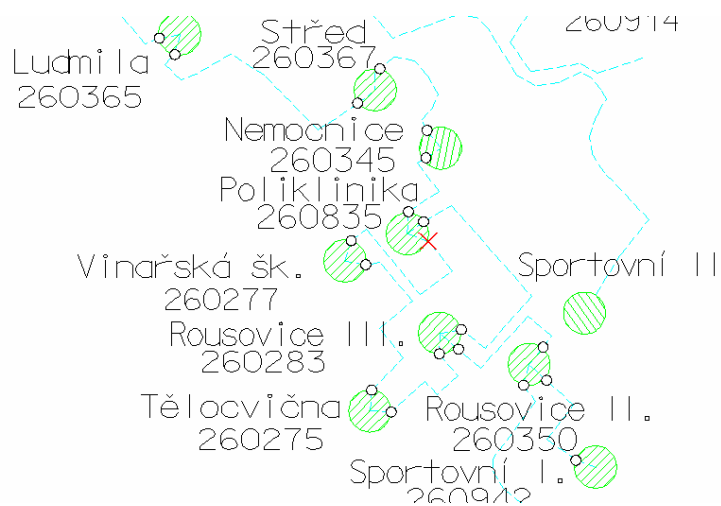
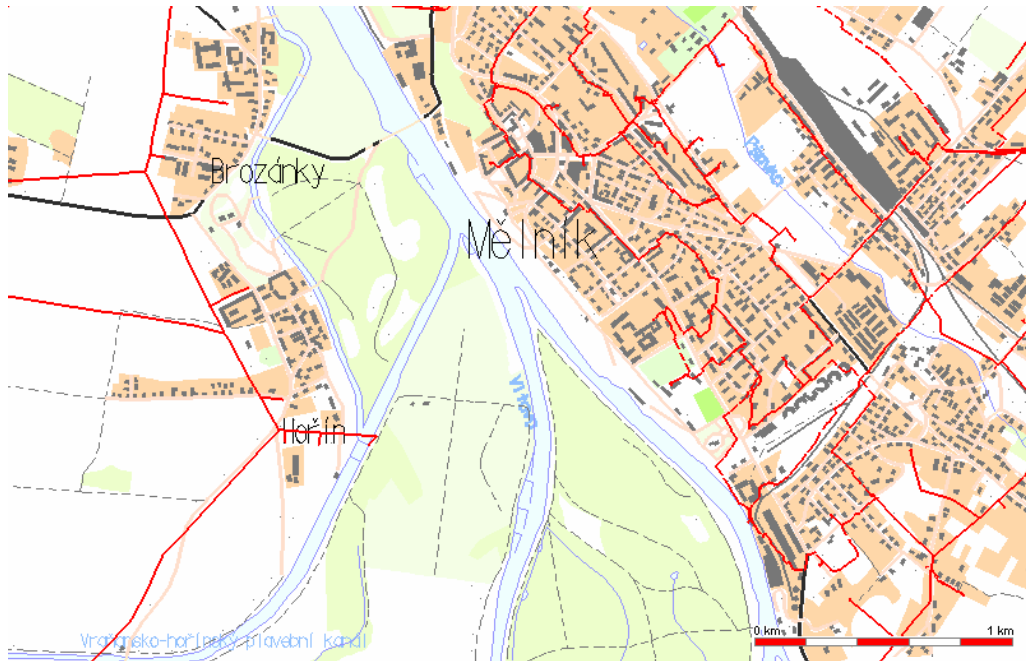
    ...
    switch (getWKbType(ge)) // rozskok podle typu
    {
        case wkbPoint:
            ...
        case wkbLineString:
            doSomethingWithLineString(pg);
            size=sizeofWKBLstr(pg);
            break; ...

        case wkbGeometryCollection:
            newPos=sizeof(byte)+sizeof(uint32)*2;
            size=newPos;
            nP=pg->num_wkbGeometries;
            for (i=0; i<nP; ++i)
            {
                pg=&pg[newPos];
                newPos=processWKGB(pg);
                size+=newPos;
            }
            break;
        default:
            error("Unsupported ..");
            break;
    }
    return(size); }
```

Norma WKB má i formu výměnného formátu v XML – GML – Geographic Markup Language. Ta zahrnuje definici geometrie ve formátu GML, popisnou část informace nijak neomezuje. Vzhledem k masivní podpoře XML parserů mnoha vývojových prostředí se zřejmě jedná „formát budoucnosti“.

Příklad GML:

```
<?xml version="1.0"?>
<GeometryCollection
xmlns:gml="http://www.opengis.net/gml">
  <PSVN_USK>
    <ID>10225914</ID>
    <L_PARAM>271527</L_PARAM>
    <gml:LineString srsName="null">
      <gml:coordinates cs="," decimal="."
        ts="">-595427397,-1075666207
              -595438694,-1075937499
      </gml:coordinates>
    </gml:LineString>
  </PSVN_USK>
  <PSVN_USK>
    <ID>10239989</ID>
    <L_PARAM>671864</L_PARAM>
    <gml:LineString srsName="null">
      <gml:coordinates cs="," decimal="." ts="">
        -594758645,-1075683985
        -595382726,-1075607457
        -595425487,-1075601997
      </gml:coordinates>
    </gml:LineString>
  </PSVN_USK>
  .
  .
```

Dokumentační systémy správců sítí jsou často založeny na CAD prostředcích:

- bohatší repertoár vyjadřovacích prostředků
- ověřené prostředky pro pořizování dat
- součástí geometrie je i její symbologie (grafická reprezentace)

–
Geometrické elementy v CAD systému (část normy IGDS – Bentley Systems, Intergraph)

IGDS	WKB
CELL_HEADER_ELM	WKBGeometryCollection
LINE_ELM	WKBLineString
LINE_STRING_ELM	WKBLineString
SHAPE_ELM	WKBPolygon
TEXT_ELM	WKBPoint
TEXT_NODE	WKBPoint
CURVE_ELM	WKBLineString
CMPLX_STRING_ELM	WKBLineString
CMPLX_SHAPE_ELM	WKBPolygon
ELLIPSE_ELM	WKBPolygon
ARC_ELM	WKBLineString
POINT_STRING_ELM	WKBMultiPoint
BSPLINE_ELM	WKBLineString
DIMENSION_ELM	WKBGeometryCollection
MULTILINE_ELM	WKBMultiLineString

Definice 1 - Problém vyhledání

Bud' V konečná množina objektů typu T_1 , q objekt typu T_2 . Problém vyhledání je funkce $search(q, V)$, která vrací odpověď typu T_3 . \square

Příklad 2 - Problém příslušnosti prvku k množině

Položme $T_1 = T_2$, a $T_3 = \{T, F\}$. Vrací-li funkce $search(q, V)$ hodnotu T v případě, že $q \in V$ a hodnotu F v případě $q \notin V$, potom říkáme, že funkce $search$ řeší problém příslušnosti pro množinu V . \square

Příklad 3 - Rozsahový dotaz na uspořádané množině

Necht' (V, \leq) je úplně uspořádaná množina, $T_2 = T_1 \times T_1$, $T_3 = exp(T_1)$. Je-li $q = [low, high]$ a vrací-li funkce $search$ takovou množinu $R \subseteq V$, že pro všechna $x \in R$ platí $low \leq x \leq high$, potom říkáme, že funkce $search$ řeší problém rozsahového výběru na množině V . \square

Příklad 4 - Rozsahový dotaz na body ve 2D prostoru

Nechť $V \subseteq E_2$ a $|V| < \infty$ (konečná množina bodů euklidovského 2D prostoru). Typem T_1 je tedy typ „bod ve dvourozměrném prostoru“. Bud' dále $T_2 = T_1^2$ takový typ, že pro každé $x = [x_{min}, y_{min}, x_{max}, y_{max}]$ typu T_2 , je

$$x_{min} \leq x_{max} \text{ a } y_{min} \leq y_{max}.$$

(Typ dotazů jsou obdélníky (okna) rovnoběžné s osami souřadného systému). Typ odpovědí T_3 je opět množina takových bodů z V , které leží uvnitř dotazového obdélníku. Funkce $search(q, V)$ vrací všechny prvky množiny V , které leží uvnitř obdélníku q , tedy je-li $q = [x_{min}, y_{min}, x_{max}, y_{max}]$ potom se jedná o všechny body $b = [x, y] \in V$, pro které je:

$$x_{min} \leq x \leq x_{max} \text{ a } y_{min} \leq y \leq y_{max}$$

řeší problém rozsahového dotazu na body ve 2D prostoru.

□

Příklad 5 - Rozsahový dotaz na obdélníky ve 2D prostoru

Bud' V konečná množina obdélníků ve 2D prostoru takových, že jejich strany jsou rovnoběžné s osami souřadného systému. Typem T_1 je tedy typ „čtveřice souřadnic“ $[x_{min}, y_{min}, x_{max}, y_{max}]$. Nechť dále je $T_2 = T_1$. Typ dotazů jsou opět obdélníky (okna), které jsou rovnoběžné s osami souřadného systému. Typ odpovědí T_3 je množina obdélníků z V , které incidují obdélníkem dotazu. Funkce $search(q, V)$, která pro

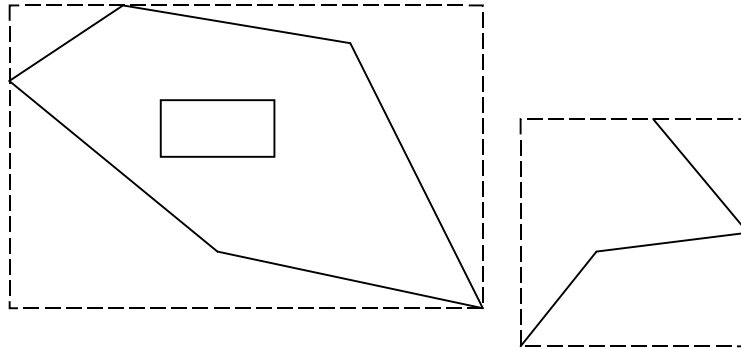
$q = [x_{minQ}, y_{minQ}, x_{maxQ}, y_{maxQ}]$ vrací

$[x_{min}, y_{min}, x_{max}, y_{max}] \in V$ s vlastností

$$x_{min} \leq x_{maxQ} \wedge y_{min} \leq y_{maxQ} \wedge x_{minQ} \leq x_{max} \wedge y_{minQ} \leq y_{max}$$

řeší problém rozsahového dotazu na obdélnících 2D prostoru. □

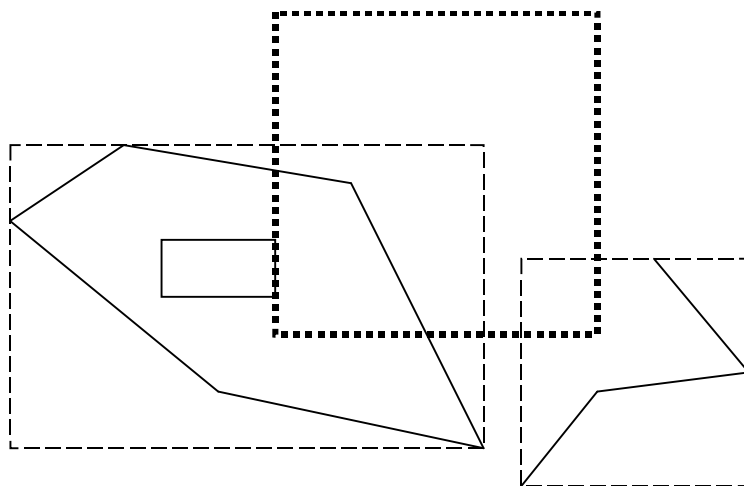
Jednotný zdroj pro indexování geometrických objektů je minimální omezující obdélník geometrického objektu rovnoběžný s osami souřadného systému – MBR (minimal bounding rectangle):



tedy minima, resp. maxima lomových (definičních) bodů

xmin,ymin,xmax,ymax

V naprosté většině případů vystačíme s obdélníkovým dotazem:



Metoda, která realizuje tento dotaz je často nazývána primárním filtrem (ORACLE). Metoda která realizuje přesnou odpověď je nazývána filtrem sekundárním.

Běžné indexovací metody (tj. ty které jsou implementovány v RDBMS – např. B⁺ stromy) poskytují efektivní aparát pro vyhledávací problémy:

- příslušnosti k množině
- rozsahový dotaz

neposkytují samy o sobě aparát vhodný k prostorovým dotazům:

- rozsahový dotaz na obdélníky ve 2D prostoru

Příklad:

Máme soubor intervalů (1D obdélníků), a dotaz bude opět interval. Odpovědí budou všechny intervaly, které s dotazem incidují (mají neprázdný průnik)

Podmínka pro incidenci:

$$\begin{aligned} [x_{\min}, x_{\max}] \cap [x_{\min Q}, x_{\max Q}] &\neq \emptyset \\ \Leftrightarrow \\ x_{\min} &\leq x_{\max Q} \wedge x_{\max} \geq x_{\min Q} \end{aligned}$$

Indexovací metoda, která podporuje „první zásah“ nám nepomůže, neboť nejhorší případ dotazu vede prohledání celého souboru.

Problémy vyhledávání rozdělíme na dvě hlavní třídy:

- **problém statický**
- **problém dynamický**

Statický:

- *build(V)* **vybuduje podpůrné struktury pro množinu V**
- *search(q, V)* **odpoví na vyhledávací dotaz**

Dynamický:

- *insert(x, V)* **vloží do množiny V nový objekt x**
- *delete(x, V)* **vymaže z množiny V objekt x**
- *search(q, V)* **odpoví na vyhledávací dotaz**

Statický problém lze řešit podobně jako dynamický problém opakovaným použitím funkce *insert*. □

Funkce *search* bývá většinou rozdělena na dvě části, a to

- *init(q, V)* **inicializace dotazu**
- *fetch(x)* **vrací jeden objekt z množiny V**

práce potom probíhá podle jednoduchého schématu

```
init(q, V);  
while(fetch(x) == SUCCESS)  
{  
    zpracuj_objekt(x);  
} □
```

Poznámka 1

Všechny uvedené příklady lze triviálně řešit jedním průchodem množiny V , tedy v lineární časové složitosti $O(|V|)$. Uvádění jiných metod má tedy smysl pouze v případě, že tento základní odhad nějak zlepšíme. \square

Pro rozsahové výběry se většinou studuje časová složitost „zásahu“ prvního objektu, který splňuje podmínku rozsahového výběru.

Algoritmus 1 - Vyhledání klíče v binárním stromu

1. Vstup: kořen stromu nod , klíč k .
2. Je-li strom prázdný, potom končíme vyhledávání "neúspěchem".
3. Je-li $key(nod) = k$, potom končíme "úspěchem".
4. Je-li $k < key(nod)$, pokračujeme krokem 1 pro $lSon(nod)$.
5. Je-li $k > key(nod)$, pokračujeme krokem 1 pro $rSon(nod)$.

Algoritmus 2 - Vkládání klíče do binárního stromu

1. Vstup: klíč k .
2. Procházíme strom, jako bychom hledali klíč k , dokud nenarazíme na volnou pozici, tedy končíme bodem 2 předešlého algoritmu.
3. Do volné pozice vložíme klíč k .

Algoritmus 3 - Rozsahové vyhledání v binárním stromu

1. Vstup: interval $[min, max]$, kořen stromu nod .
2. Patří-li $key(nod)$ do intervalu $[min, max]$, pošleme jej na výstup a aplikujeme algoritmus na $lSon(nod)$ a $rSon(nod)$.
3. Je-li $max < key(nod)$, aplikujeme algoritmus na $lSon(nod)$.
4. Je-li $min > key(nod)$, aplikujeme algoritmus na $rSon(nod)$.

Zlepšení časové složitosti spočívá v tom, že v určitých fázích algoritmů jsme schopni rozhodnout, kterou větev stromu můžeme bez rizika vynechat. Potíže způsobuje skutečnost, že v jistých případech může být strom degenerovaný (např. $lSon(nod) = \emptyset$ pro všechny uzly nod). Degenerace nastává tehdy, když jednotlivé prvky vstupují do stromu v nevhodném pořadí. V případě statické verze vyhledávacích problémů lze vybudovat tzv. optimální binární strom (na vstupu procedury *build* známe celou množinu V).

Definice 2 - Optimální strom

Strom nazveme optimální, liší-li se počty uzlů v podstromech $lSon(nod)$ a $rSon(nod)$ maximálně o 1 pro jeho každý uzel nod . \square

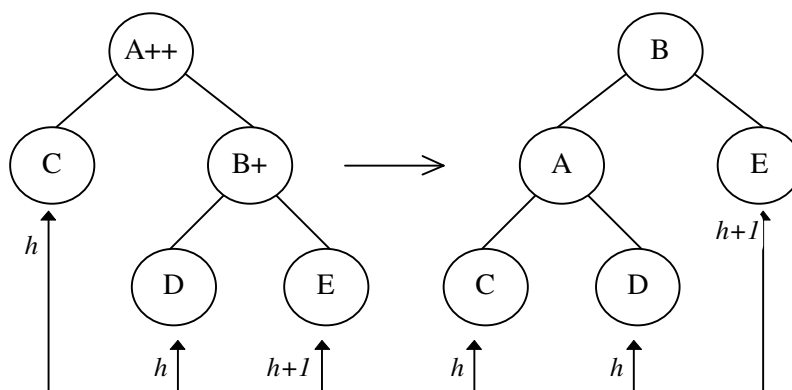
Poznámka 2 – Hloubka optimálního stromu obsahujícího $|V|$ klíčů je $\log(|V|)$.

Algoritmus 4 - Vybudování optimálního binárního stromu

- 1. Vstup: množina klíčů V , kořen stromu nod .**
- 2. Je-li $V = \emptyset$, skonči.**
- 3. Rozděli množinu V na dvě disjunktní množiny $V_1, \{med(V)\}, V_2$ tak, že $med(V)$ je medián množiny V , klíče z V_1 jsou menší než $med(V)$ a klíče z V_2 jsou větší než $med(V)$.**
- 4. Definuj kořen stromu jako $med(V)$.**
- 5. Aplikuj algoritmus na množinu V_1 pro levý podstrom $lSon(nod)$.**
- 6. Aplikuj algoritmus na množinu V_2 pro pravý podstrom $rSon(nod)$. \square**

Definice 3 - Vyvážené stromy

Binární strom nazveme vyvážený, liší-li se hloubky $lSon(nod)$ a $rSon(nod)$ maximálně o 1 pro jeho každý uzel nod (hloubkou stromu rozumíme maximální délku cesty od kořene k listu). \square



Poznámka 3 – Hloubka vyváženého stromu obsahujícího $|V|$ klíčů je $\approx \log(|V|)$.

Definice 4 - $BB[\alpha]$ stromy

Bud' $0 < \alpha < 1/2$. Binární strom T patří do třídy $BB[\alpha]$ stromů, platí-li pro jeho každý uzel nod , podstrom $Tree(nod)$ s kořenem nod , levý podstrom $lSon(nod)$

$$\alpha \leq |lSon(nod)|/|Tree(nod)| \leq 1-\alpha. \square$$

pokud byl v nějakém okamžiku podstrom definovaný uzlem nod optimální, pak k porušení podmínky z definice $BB[\alpha]$ stromů musí dojít k minimálně $c \cdot |Tree(nod)|$ vložení/mazání uzlů do/z příslušného podstromu (c je konstanta závislá pouze na parametru α).

Definice 5 – B⁺-stromy

B-strom řádu m je strom s těmito vlastnostmi:

- každý uzel má $\leq m$ synů
- každý uzel, s výjimkou kořene a listů, má $\geq m/2$ synů
- kořen má minimálně 2 syny, pokud není list
- všechny listy jsou na stejné úrovni
- nelistový uzel s k syny obsahuje $k - 1$ klíčů \square

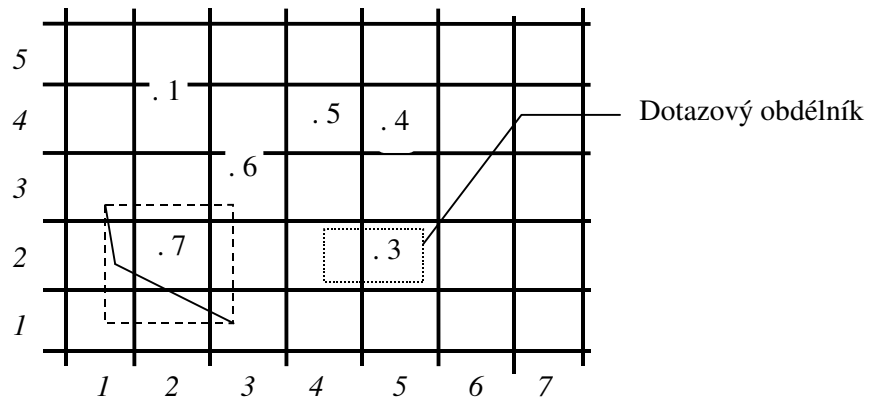
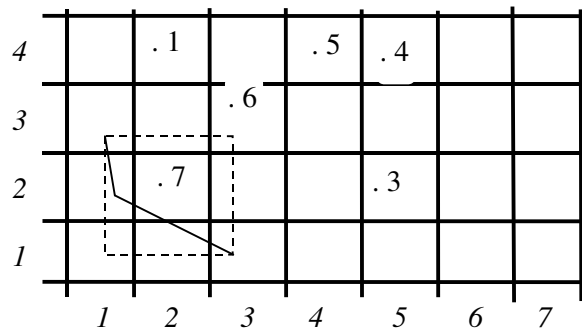
Hlavní myšlenka spočívá ve tvaru uzlů:

$$p_0 \text{ key}_1 p_1 \dots p_{k-1} \text{ key}_k p_k$$

kde p_i je ukazatel na uzel pro s všechny klíči key s vlastností:

$$key_{i-1} \leq key \leq key_i$$

Metoda „GRID“



Prostorový dotaz v GRIDu, prohledáváme pouze čtverce (4,2) a (5,2), pro efektivní přístup ke čtvercům použijeme libovolnou vyhledávací metodu (strukturu).

Realizace GRID metody v prostředí SQL

```
create table GTABLE
(
  GID number,
  XMIN number,
  YMIN number,
  XMAX number,
  YMAX number,
  WKB_GEOMETRY blob,
  .
  .
  constraint GTABLE_PK primary KEY (GID)
);

create table GTABLE_IDX
(
  GID number,
  GRID_X number,
  GRID_Y number
);

alter table GTABLE_IDX add constraint
  GTABLE_IDX_PK primary key (gid,grid_x,grid_y);

alter table GTABLE_IDX add constraint
  GTABLE_IDX_fk1
  foreign key (GID) references GTABLE(GID)
  ON DELETE CASCADE;

create index GTABLE_IDX_I1
  on GTABLE_IDX(grid_x, grid_y);
```

```

create trigger gtable_spatial
before insert or update of x,y on GTABLE for
each row
begin
  xfrom:=GET_GRID_X(:NEW.XMIN);
  xto  :=GET_GRID_X(:NEW.XMAX);
  yfrom:=GET_GRID_Y(:NEW.YMIN);
  yto  :=GET_GRID_Y(:NEW.YMAX);

  pro xfrom<=i<=xto a yfrom<=j<=yto
  begin
    INSERT INTO GTABLE_IDX VALUES (:NEW.GID, i, j);
  end;

end;
/

```

(funkce GET_GRID_X/Y vrací gridové indexy)

```

Create table SPATIAL_QUERY
(
  id int,
  xmin int,
  ymin int,
  xmax int,
  ymax int,
  constraint SPATIAL_QUERY_PK
  primary key (id)
);

```

```

CREATE TABLE SPATIAL_QUERY_IDX
(
  query_id int,
  grid_x int,
  grid_y int,
  constraint SPATIAL_QUERY_IDX_PK
    primary key (query_id,grid_x, grid_y),
  constraint SPATIAL_QUERY_IDX_FK
    foreign key query_id references
    SPATIAL_QUERY(ID)
    on delete cascade
);

create trigger SPATIAL_QUERY_SPATIAL
before insert of id on SPATIAL_QUERY
for each row
xfrom,xto,yfrom,yto,i,j int;
BEGIN
  xfrom:=GET_GRID_X(:NEW.XMIN);
  xto:=GET_GRID_X(:NEW.XMAX);
  yfrom:=GET_GRID_Y(:NEW.YMIN);
  yto:=GET_GRID_Y(:NEW.YMAX);

  pro xfrom<=i<=xto a yfrom<=j<=yto
  BEGIN
    INSERT INTO SPATIAL_QUERY_IDX
      VALUES (:NEW.ID,i,j);
  END;

END;
/

```


Prostorový dotaz pro obdélník x_{min} , y_{min} , x_{max} , y_{max} provedeme následovně:

1) Identifikace dotazu:

Z databáze získáme nový (jednoznačný) klíč dotazu, například ze sekvence `select query.nextval from query_sequence.`

2) Inicializace dotazu:

```
insert into spatial_query
values (id, xmin, ymin, xmax, ymax),
tím se vlivem triggeru SPATIAL_QUERY_SPATIAL
automaticky vloží identifikace gridových čtverců to
tabulky spatial_query_idx
```

3) Prostorový dotaz:

```
select ...
from
  gtable           A,
  gtable_idx       B,
  spatial_query_idx C
where
  A.GID=B.GID      AND
  B.grid_x=C.grid_x AND
  B.grid_y=C.grid_y AND
  C.query_id=id;
```

4) Ukončení prostorového dotazu:

```
delete from spatial_query where id=id;
```

Jaký mechanismus odstraňuje řádky z tabulky `spatial_query_idx`?

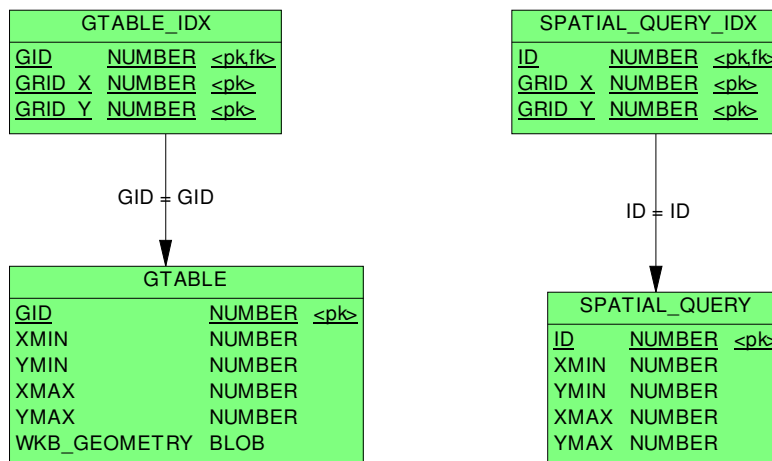
Exekuční plán prostorového dotazu GRID

The screenshot shows the TOAD interface with a SQL query and its execution plan. The query is as follows:

```
select /*+rule*/ A.gid
from
  gtable          A,
  gtable_idx      B,
  spatial_query_idx C
where
  A.GID=B.GID      AND
  B.grid_x=C.grid_x AND
  B.grid_y=C.grid_y AND
  C.id=0;
```

The execution plan is shown in the 'Explain Plan' tab:

- SELECT STATEMENT Optimizer=HINT: RULE
 - NESTED LOOPS
 - NESTED LOOPS
 - INDEX (RANGE SCAN) OF SPATIAL_QUERY_IDX_PK (UNIQUE)
 - TABLE ACCESS (BY INDEX ROWID) OF GTABLE_IDX
 - INDEX (RANGE SCAN) OF GTABLE_IDX_I1 (NON-UNIQUE)
 - INDEX (UNIQUE SCAN) OF GTABLE_PK (UNIQUE)



Výhody vs. nevýhody GRID metody.



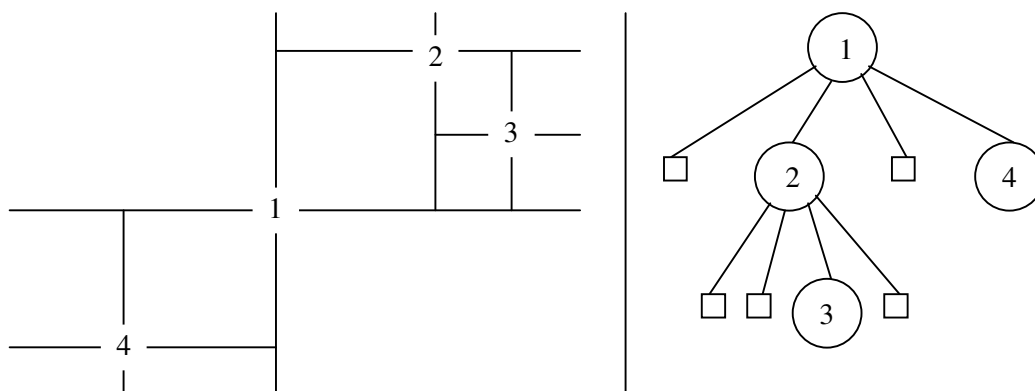
- velmi snadná implementace v prostředí RDBMS
- snadné rozšíření na více dimenzí (?)
- relativně snadná (resp. řešitelná implementace neobdélníkových dotazů)



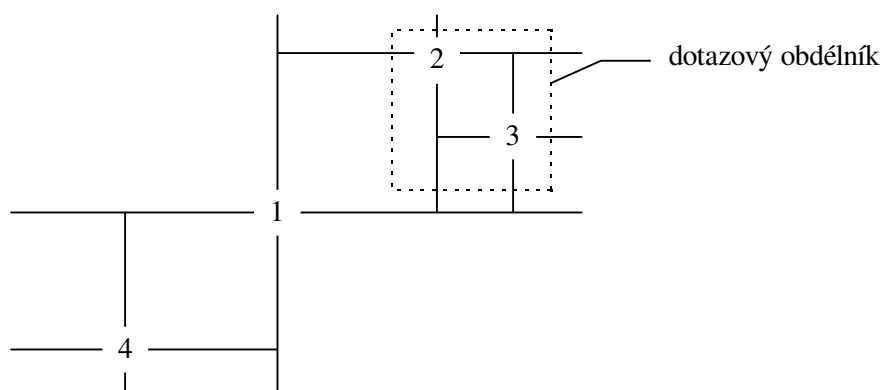
- netriviální odhad velikosti GRIDových čtverců, špatná volba má dramatické důsledky
- nepravidelné chování při řádově rozdílné velikosti geometrických objektů

Quad Tree – kvartérní strom

- $urSon(nod)$ - klíče tohoto podstromu jsou „vpravo nahoře“ od bodu $key(nod)$
- $ulSon(nod)$ - klíče tohoto podstromu jsou „vlevo nahoře“ od bodu $key(nod)$
- $lrSon(nod)$ - klíče tohoto podstromu jsou „vpravo dole“ od bodu $key(nod)$
- $llSon(nod)$ - klíče tohoto podstromu jsou „vlevo dole“ od bodu $key(nod)$



Obr. 8 - Quad Tree - Kvartérní strom



Obr. 9 - Rozsahový dotaz v Quad-Tree pro body. Podstrom „4“ neprohledáváme.

Algoritmus – Vkládání do Quad-Tree

Vkládání je stejné, jako v případě obyčejných binárních stromů. Hledáme tedy bod v kvartérním stromu, dokud nenajdeme volnou pozici. Do ní vložíme nový klíč.

Algoritmus - Rozsahový dotaz v Quad-Tree pro body

1. Vstup: kořen stromu nod a obdélník $q=[xmin,ymin,xmax,ymax]$.

2. Je-li $Tree(nod)=\emptyset$, skonči."

3. Je-li $key(nod)$ v dotazovém obdélníku q , pošli jej na výstup a aplikuj algoritmus na všechny jeho čtyři podstromy.

4. Vyber podstromy, pro které budeš aplikovat algoritmus ($x()$ značí x-ovou souřadnici klíče):

- je-li $x(key(nod)) > xmax \wedge y(key(nod)) > ymax$, potom aplikuj algoritmus na větev $lSon(nod)$,
- je-li $x(key(nod)) > xmax$, potom aplikuj algoritmus pouze na větev $lSon(nod)$ a $ulSon(nod)$,
- a analogicky pro ostatní případy.

Definice 6 - k-D strom

Úroveň *level(nod)* uzlu *nod* binárního stromu rozumíme délku cesty k tomuto uzlu od kořene stromu.

Bud' (S, \leq) uspořádaná množina,

$k > 0$,

$$x = (x_0, \dots, x_i, \dots, x_{k-1}), \quad y = (y_0, \dots, y_i, \dots, y_{k-1}) \in S^k$$

Říkáme, že

$$x \leq_i y, \text{ jestliže } x_i \leq y_i$$

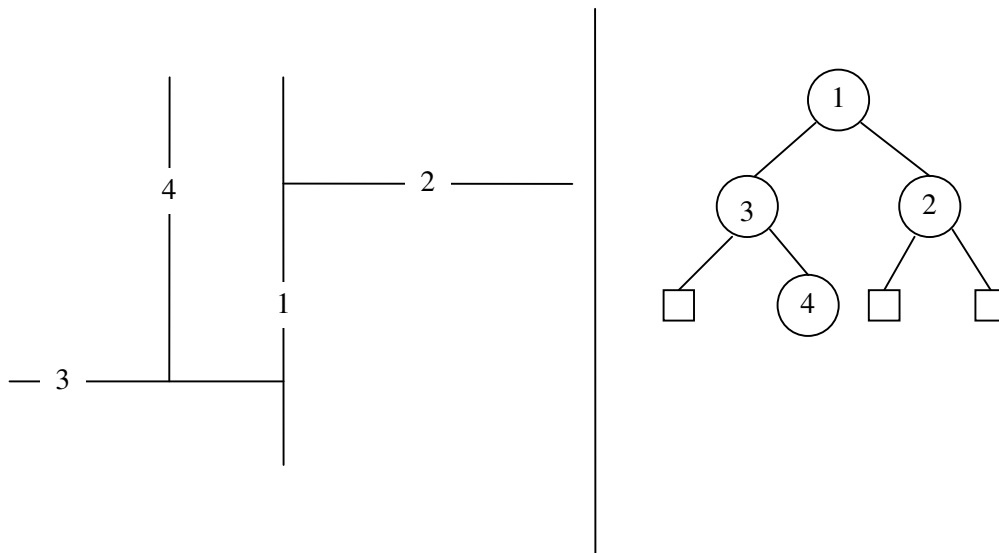
k-D stromem nad S nazveme binární strom, jehož uzly jsou k-tice z S^k , a kde pro každý uzel *nod*, jeho levý podstrom $lSon(nod)$ a všechny uzly tohoto podstromu *nodL* platí:

$$nodL \leq_i nod \quad \text{kde } i = level(nod) \bmod k$$

Analogická podmínka musí být splněna i pro pravý podstrom $rSon(nod)$. \square

Algoritmus – Vložení bodu do 2-d stromu

Analogicky binárním stromům, hledáme ve stromu „bod“ dokud nenarazíme na volnou pozici.



Obr. 10 - 2-d strom

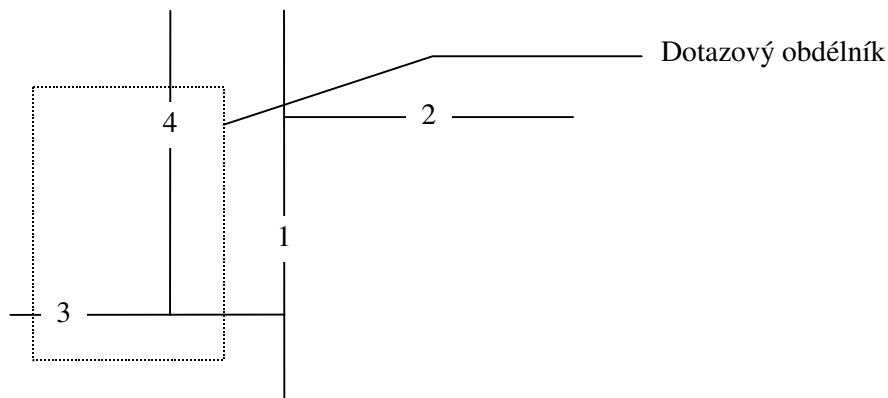
Algoritmus - Rozsahový dotaz pro body ve 2-d stromu

Tree(nod)* označuje podstrom z uzlu *nod

key(nod)* označuje bod v uzlu *nod

***x()* značí *x-ovou* souřadnici klíče bodu**

- 1. Vstup: Kořen *nod* a obdélník $q=[xmin,ymin,xmax,ymax]$.**
- 2. Je-li $Tree(nod)=\emptyset$, skonči.**
- 3. Je-li $key(nod)$ v dotazovém obdélníku, pošli jej na výstup a aplikuj algoritmus na oba dva syny.**
- 4. Vyber syny, pro které budeš aplikovat algoritmus a to podle úrovně ve které se nachází uzel *nod*:**
 - **je-li $level(nod) \bmod 2=0 \wedge x(key(nod)) > xmax$, potom aplikuj algoritmus na větev $ISon(nod)$,**
 - **analogicky pro další možné případy.**



**Obr. 11 - Rozsahový dotaz v 2-d-stromu pro body.
Podstrom „2“ neprohledáváme.**

Vyvažování multidimensionálních stromů je komplikované. Nedají se totiž provádět rotace jako v klasických binárních stromech (srovnej s obr. 5), protože v každém patře stromu měníme srovnávací kritérium.

Pomocí $BB[\alpha]$ techniky lze však k -D stromy udržovat vyvážené pomocí částečné reorganizace. K tomu potřebujeme techniku pro budování optimálního 2-D stromu.

Algoritmus -Vybudování optimálního 2-D stromu

1. Vstup: množina bodů V , kořen stromu nod a $l \in \{x, y\}$
2. Je-li $V = \emptyset$, skonči.
3. Rozděl množinu V na po dvou disjunktní množiny $V_1, \{med_l(V)\}, V_2$ tak, že $med_l(V)$ je takový bod, že jeho x -ová souřadnice je medián množiny l -ových souřadnic z V , l -ové souřadnice z V_1 jsou menší než $med_l(V)$ a l -ové z V_2 jsou větší než $med_l(V)$.
4. Definuj kořen stromu jako $med_l(V)$.
5. Je-li l rovno x potom přiřaď $l=y$ jinak $l=x$
6. Aplikuj algoritmus na množinu V_1 pro levý podstrom $lSon(nod)$.
7. Aplikuj algoritmus na množinu V_2 pro pravý podstrom $rSon(nod)$.□

Metodu k-D stromů lze použít i na obdélníky, které můžeme považovat za 4D body. Použijeme tedy 4-D strom.

Algoritmus 7 - Rozsahový výběr pro obdélníky ve 4-d stromu

1. Vstup: kořen stromu *nod*, dotazový obdélník $q=[x_{min},y_{min},x_{max},y_{max}]$.
2. Je-li $Tree(nod)=\emptyset$, skonči.
3. Jsou-li obdélníky q a $key(nod)$ incidentní, pošli $id(nod)$ na výstup a aplikuj algoritmus na $lSon(nod)$ a $rSon(nod)$.
4. Podle úrovně, ve které se nacházíš ve stromu, se rozhodni, zda můžeš vynechat nějakou větev, např.:

Je-li $level(nod) \bmod 4 = 0$ a $x_{min}(nod(key)) > x_{max}(q)$

aplikuj algoritmus pouze pro $lSon(nod)$.

Analogicky pro další úrovně, v každé se dá za jistých podmínek jedna větev vynechat.

Pevný kvartérní strom (*non-pointer Quad Tree*)

Zájmové území je postupně děleno na obdélníkové části a podle nich je jim přidělován „klíč“

1000	2000	
3000	4100	4200
	4300	4400

Obr. 12 - Číslování obdélníků-dlaždic v *non-pointer Quad Tree*.

Obdélník bude mít index takové dlaždice „pevné struktury“ která je jeho nadmnožinou a je nejmenší s touto vlastností.

- Pro libovolný obdélník R označme $Q(R)$ jeho klíč v *non-pointer QuadTree*.
- Pro libovolný klíč K označme jeho „nenulovou“ část, tedy levý podřetězec symbolem $NZ(K)$.
- Délku znakového řetězce K označme $strlen(K)$.
- Podřetězec řetězce K z levé strany délky l označme $lsubstr(K,l)$.

Tvrzení

Bud'te A, B libovolné obdélníky, jejichž strany jsou rovnoběžné s osami souřadného systému. Necht' dále $A \cap B \neq \emptyset$. Označíme-li $l = \min\{\text{strlen}(\text{NZ}(Q(A))), \text{strlen}(\text{NZ}(Q(B)))\}$, potom:

$$l_{\text{substr}}(Q(A), l) = l_{\text{substr}}(Q(B), l)$$

□.

Algoritmus - Vyhledání obdélníků v non-pointer QuadTree

1. Vstup – obdélník $S = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]$.
2. Pošli na výstup všechny obdélníky A , pro které:

$$l_{\text{substr}}(Q(A), \text{strlen}(\text{NZ}(Q(S)))) = \text{NZ}(Q(S)) \wedge A \cap S \neq \emptyset$$

3. Pošli na výstup všechny obdélníky A , pro které:

$$Q(A) = P \wedge A \cap S \neq \emptyset$$

kde P jsou všechny klíče, které jsou na cestě od $Q(S)$ ke kořenu, tj. v $Q(S)$ zprava postupně nahrazujeme nenulové číslice nulami.

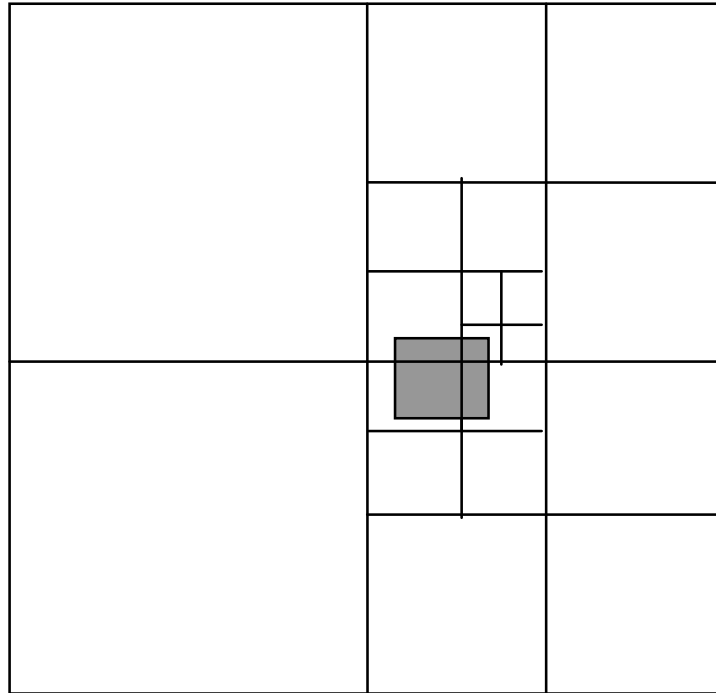
Tento postup má jednu nevýhodu, v případě, že dotaz inciduje se středem území, potom procházíme v bodě 2 všechno. Této nevýhodě se vyhneme dekomponováním dotazu.

Algoritmus Dekompozice dotazu v non-pointer QuadTree

1. Vstup – dotazový obdélník S .
2. Rozděl obdélník S na obdélníky S_1 a S_2 ($S_1 \cup S_2 = S$) podle takové souřadnice x resp. y , která způsobila klíčování v *non-pointer quadTree*, tj. takovou, která ohraničuje nějaký čtverec v *non-pointer quadTree* a prochází dotazovým obdélníkem S . V případě, že taková souřadnice neexistuje potom obdélník S neděl a konec.
3. Aplikuj krok 2. na čtverce S_1 a S_2 podle druhé souřadnice.

Následuje příklad, na kterém demonstrujeme hlavní výhody této metody:

- Velmi snadná implementace v prostředí SQL - tedy relačních databází
- „jeden objekt“ = „jeden klíč“, znamená, že prostorová indexace je zabezpečena přímo v geometrické tabulce. Prostorový výběr nevyžaduje součin, či spojení s dalšími tabulkami.



Dekompozice dotazu - 4 obdélníky s klíči:

2330000000,2343000000,4110000000,4120000000

```

SELECT  ID FROM KM_ALL WHERE
(
  (SPAT_KEY BETWEEN '2330000000' AND '2335000000') OR
  (SPAT_KEY BETWEEN '2343000000' AND '2343500000') OR
  (SPAT_KEY BETWEEN '4110000000' AND '4115000000') OR
  (SPAT_KEY BETWEEN '4120000000' AND '4125000000')
) OR
SPAT_KEY IN
('0000000000',
 '2000000000',
 '2300000000',
 '2340000000',
 '4000000000',
 '4100000000'
)
AND (xmax>=-642646042) AND (ymax>=-1114990337) AND
    (xmin<=-569087654) AND (ymin<=-1070777051)

```

Intervalové stromy

Za zaměstnance podniku máme data v tabulce:

Identifikace	Počátek_prac_poměru	Konec_prac_poměru

a potřebujeme řešit dotazy typu “kdo všechno byl zaměstnán v daném období?”. □

Definice 7 - Intervalový strom

Intervalový strom pro množinu $V = \{[x_1, x_2], \dots, [x_{2n-1}, x_{2n}]\}$ je tvořen:

- binárním vyhledávacím stromem pro $2n$ počátečních a koncových bodů
- každý uzel nod (definovaný bodem $x(nod)$) navíc obsahuje seznam intervalů $I(nod) \subseteq V$ takový, že $x(nod)$ je bodem každého intervalu $I(nod)$
- levý (pravý) podstrom uzlu nod je intervalový strom pro intervaly, jejichž pravé (levé) koncové body jsou menší (větší) než $x(nod)$

□

Již z definice intervalového stromu je zřejmé, že je nutné jej budovat postupně.

Algoritmus 9 - Budování intervalového stromu

1. Vstup: množina intervalů V .
2. Vybuduj binární strom T pro počáteční a koncové body z V .
3. Zařazujeme postupně všechny intervaly co nejvýše do stromu T .

Poznámka 6

Každý uzel nod dělí množinu V do tří skupin. Jednak jsou to intervaly ležící vlevo od $x(nod)$, jednak intervaly ležící vpravo od $x(nod)$ a konečně intervaly které bod $x(nod)$ obsahují.□

Algoritmus - Vyhledání intervalů v intervalovém stromu

1. Vstup: interval dotazu $q=[xminQ,xmaxQ]$, kořen stromu nod

2. Je-li nod list, potom skonči.

3. Je-li $x(nod)$ v intervalu q , potom

- 3.1. $l(nod)$ na výstup
- 3.2. Aplikuj algoritmus pro $lSon(nod)$ a $rSon(nod)$

4. Je-li $maxQ < x(nod)$ potom:

- 4.1. Projdi seznam $l(nod)$ a na výstup pošli ty intervaly z $l(nod)$, které incidují s q .
- 4.2. Aplikuj algoritmus na $lSon(nod)$

5. Je-li $minQ > x(nod)$ potom:

- 5.1. Projdi seznam $l(nod)$ a na výstup pošli ty intervaly z $l(nod)$, které incidují s q .
- 5.2. Aplikuj algoritmus na $rSon(nod)$.

Poznámka 7

Seznam $l(nod)$ lze reprezentovat dvěma provázanými seznamy $L(nod)$ a $R(nod)$, ve kterých jsou levé a pravé koncové body. $L(nod)$ je uspořádán vzestupně, $R(nod)$ je uspořádán sestupně. Toho lze využít v krocích 4.1. a 5.1. tak, že procházení seznamu lze ukončit prvním neúspěchem.□

Intervalové stromy pro obdélníky

Pro jednu osu (třeba x) vybudujeme intervalový strom s tím rozdílem, že seznamy $I(nod)$ budou 2D intervaly, tedy obdélníky. Tím dostaneme primární strukturu pro 2D variantu intervalového stromu.

Sekundární struktura bude tvořena intervalovými stromy pro druhou (y -ovou) osu v každém uzlu nod primárního intervalového stromu pro y -ové intervaly ze seznamu $I(nod)$.

Rozsahový výběr v takto vybudované struktuře potom probíhá stejně, jako v 1D variantě s následujícím rozdílem:

- 3.1. Použije se sekundární struktura pro výběr podle y – interval
- 4.1. Zohlední se y - intervaly při posílání na výstup
- 5.1. Zohlední se y - intervaly při posílání na výstup

SB⁺ stromy

Jsou modifikací B⁺ stromů

SB⁺ strom je B⁺ strom z počátečních a koncových bodů intervalů a navíc:

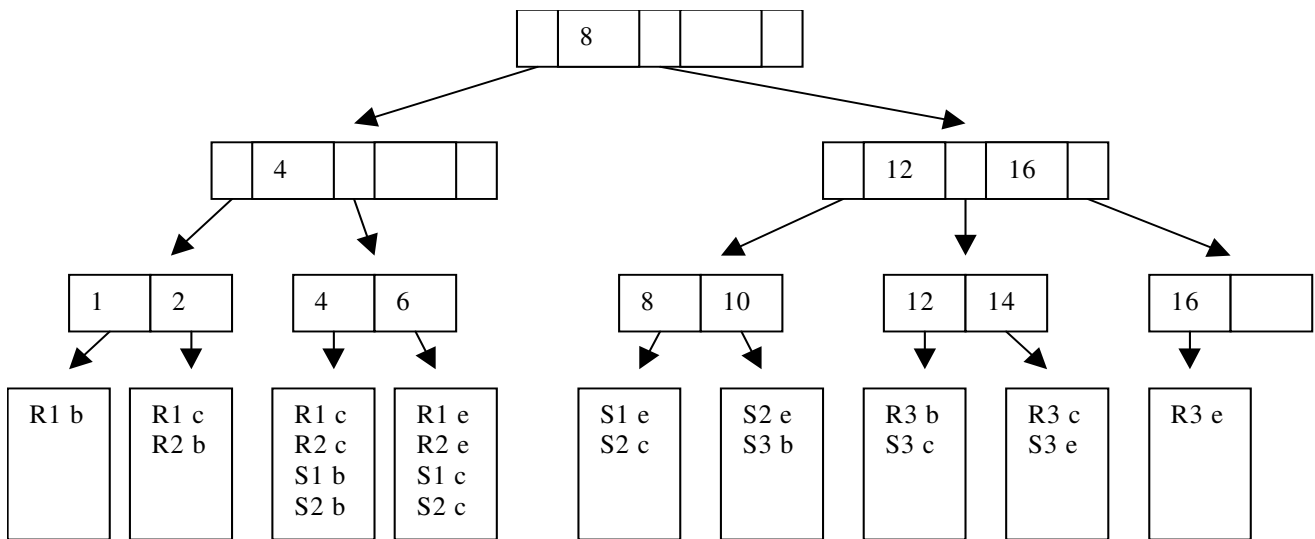
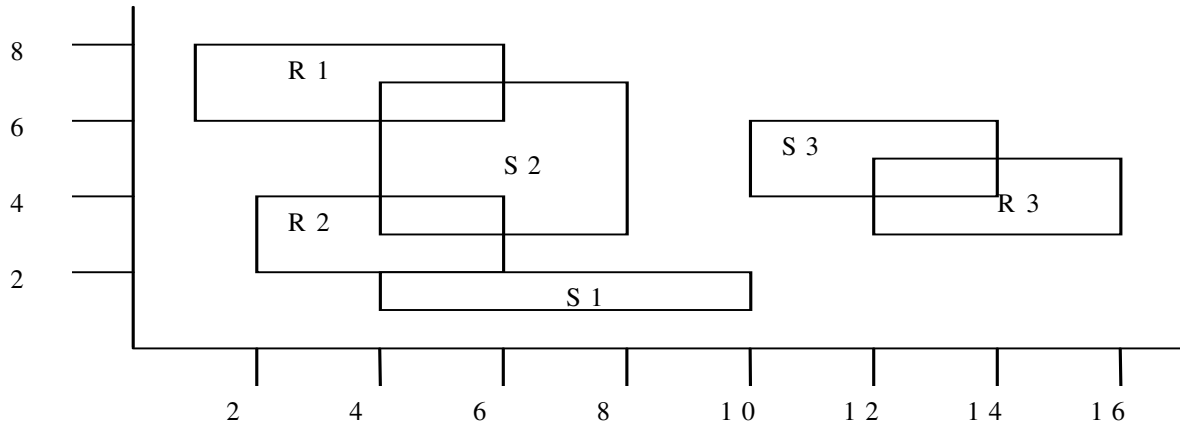
- V SB⁺ stromech jsou k listům přidány seznamy identifikátorů intervalů, které jsou incidentní s klíčem v listu (tj. nějakým počátkem resp. koncem nějakého intervalu).
- S každým identifikátorem je pamatován příznak, který označuje zda v se jedná o počáteční hodnotu intervalu, koncovou hodnotu intervalu, popřípadě zda interval touto hodnotou prochází.

Listy SB⁺ stromech jsou tedy tvořeny strukturami, které můžeme popsat následujícím způsobem:

```
typedef struct      // seznam intervalů
{
    long idInterval; // identifikace intervalu
    char incidType;  // 'b' - počátek,
}                  // 'c' průchozí, 'e' konec
sbListT;

typedef struct      // list SB+ stromu
{
    long point;     // hodnota bodu
    sbListT list[1]; // seznam incid.intervalů
}
sbLeafT;
```

SB⁺ strom



Algoritmus – Incidence intervalů v SB^+ stromech

1. Vstup dotazový interval $[xmin, xmax]$. Existující SB^+ strom S .

2. Najdi ve stromu takový list, že pro bod ip který reprezentuje tento list platí

$$ip = \min\{i; i \in S, i \geq xmin\}$$

3. Pro $ip \leq i \leq xmax$ pošli na výstup identifikace intervalů ze seznamu listu reprezentovaným bodem i (identifikace se mohou opakovat, posíláme jen jednou).

Algoritmus – Vkládání intervalů do SB^+ stromu

1. Vstupní interval $[x_{min}, x_{max}]$, jeho identifikace I .
2. Najdi ve stromu takový list, že pro bod ip který reprezentuje tento list platí $ip = x_{min}$.
3. Jestliže v kroku 2. jsme takový list nenašli, potom:
 - 3.1. Vlož do stromu bod x_{min} standardní metodou pro B^+ stromy
 - 3.2. Necht' pip je bezprostřední předchůdce x_{min} , nip bezprostřední následník x_{min} v SB^+ stromu.
 - 3.3. Polož $x_{min}.seznam = pip.seznam \cap nip.seznam$ bez ohledu na příznak typu incidence.
 - 3.4. Polož příznak typu incidence = 'c' pro všechny intervaly z $x_{min}.seznam$.
4. Kroky 2.-3. pro x_{max} .
5. Pro všechny listy SB^+ stromu takové, že pro jejich body ip platí $x_{min} \leq ip \leq x_{max}$:
 - 5.1. Je-li $ip = x_{min}$ potom přidej do $ip.seznam$ identifikaci I a příznak typu incidence 'b'.
 - 5.2. Je-li $ip = x_{max}$ potom přidej do $ip.seznam$ identifikaci I a příznak typu incidence 'e'.
 - 5.3. Je-li $x_{min} < ip < x_{max}$ potom přidej do $ip.seznam$ identifikaci I a příznak typu incidence 'c'.

Poznámka

Vícerozměrný problém řešíme vybudováním indexových struktur pro každou osu. Pro vícerozměrný výběr potom musíme vytvořit výstup jako průnik výstupů pro každou osu.

Poznámka

Strukturu SB⁺ stromu můžeme velmi efektivně použít na řešení incidence objektů v dotazovém okně, tedy na dotaz typu: *Všechny dvojice objektů, které mohou mít neprázdný průnik a leží v daném dotazovém okně.* Takový dotaz řešíme snadnou modifikací algoritmu v kroku 3..

Poznámka

Metoda SB⁺ stromu je okamžitě použitelná v relačních databázích indexovými tabulkami typu:

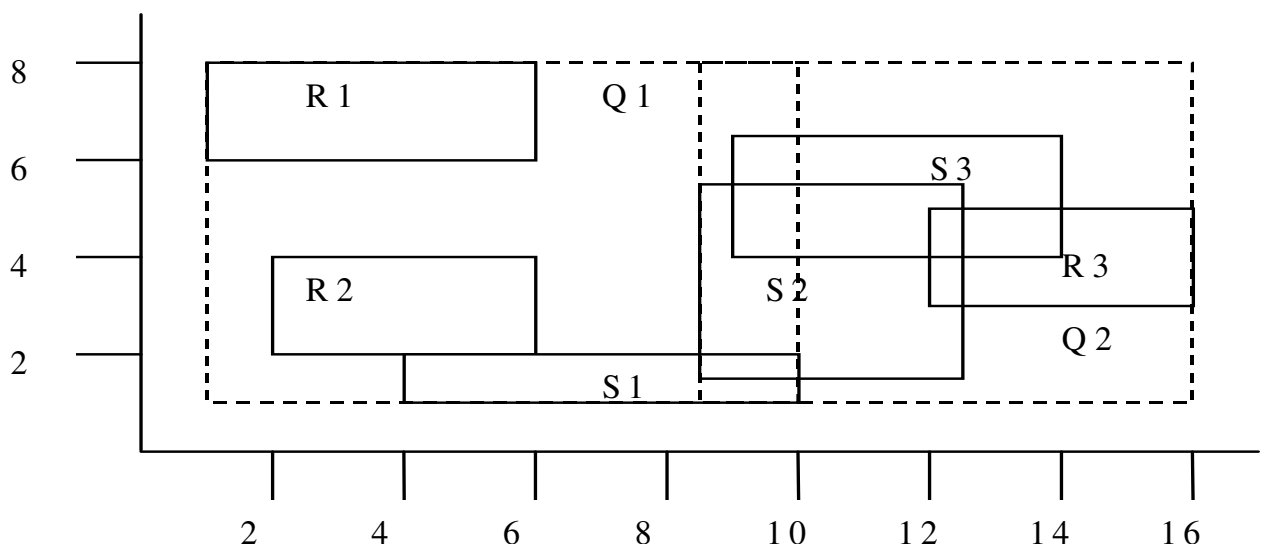
```
create table table_idx
(
  idInterval int,
  point      int,
  incidence  varchar(1)
);
```

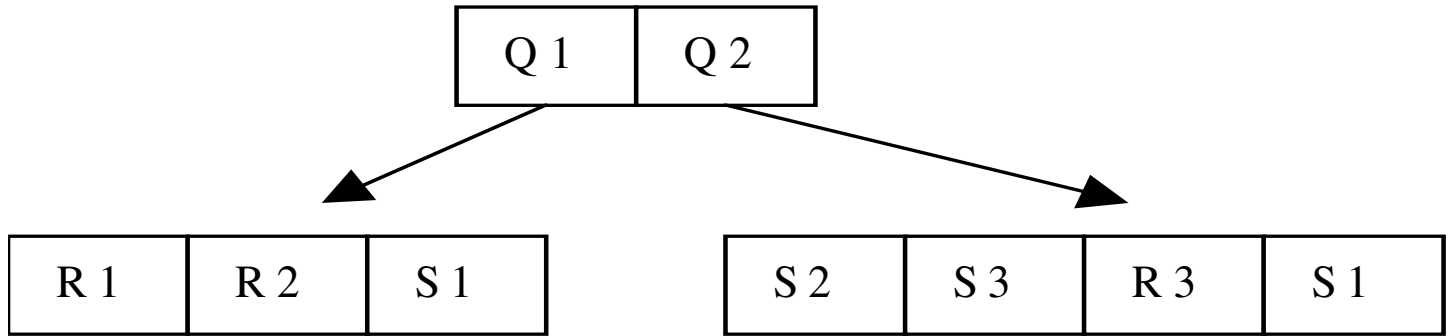
R-Stromy (R-tree)

Analogie k B-stromům

M – maximální počet klíčů v uzlu, $m \leq M/2$ – minimální počet uzlů.

- Každý uzel obsahuje minimálně m klíčů a maximálně M klíčů pokud není kořen.
- Klíče v R-stromech jsou obdélníky s ukazateli na synovské uzly, v listech obdélníky s ukazateli na geometrické prvky.
- Pro synovské uzly platí, že jejich klíče (tj. obdélníky) jsou uvnitř “otcovského” obdélníku.
- Listy stromu jsou na téže úrovni.
- Kořen obsahuje minimálně dva klíče, pokud není list.





Algoritmus – Vyhledání klíčů obdélníků v R- stromech

1. Vstup *uzel* R-stromu *R*. Dotazový obdélník *Q*.
2. Je-li *uzel* list, potom všechny klíče incidentní s *Q* na výstup.
3. Jinak aplikuj algoritmus na syny takových *klíčů* z *uzel*, pro které je *klíč* incidentní s *Q*.

Algoritmus – Vkládání klíčů do R- stromů

1. Vstup, klíč $Key=(MBR, ID)$
2. Vyhledej list *N*.
 - 1.1 Polož $N:=$ kořen stromu.
 - 1.2 Je-li *N* list návrat.
 - 1.3 Necht' klíč *F* v *N* jehož obdélník vyžaduje nejmenší rozšíření takové, aby obsahoval *MBR* vstupujícího klíče. Rozšiř jeho *MBR* a pokračuj 1.4.
 - 1.4 $N:=$ synovský uzel na který ukazuje *F*, a pokračuj krokem 1.2
3. Přidej *Key* do vybraného uzlu *N*.
4. Je-li počet klíčů v *N* menší, nebo roven *M* konec, jinak rozděl uzel *N* na dva nové uzly. Je-li *N* kořen, vytvoř nový kořen se dvěma novými klíči, jinak odstraň z rodičovského uzlu původní klíč a nahraď jej dvěma novými klíči a polož $N:=$ rodič *N*.
5. Opakuj 4.

Najdi dva obdélníky (možná incidentní) s následujícími vlastnostmi:

- Sjednocení obou obdélníků je původní obdélník
- Oba obdélníky obsahují zhruba stejný počet klíčů
- Oba obdélníky se překrývají co nejméně

Algoritmus dělení uzlu (kvadratická složitost)

1. Vyber první dva obdélníky

1.1 Pro každou dvojici klíčů k,l vytvoř minimální obdélník j obsahující oba klíče a polož $p(k,l)=Plocha(j)-Plocha(k)-Plocha(l)$.

1.2 Vyber dvojici obdélníků k,l s maximem $p(k,l)$, zařaď je do první a druhé skupiny.

2. V případě, že jedna skupina obsahuje tak málo obdélníků, že pro zachování podmínky minima m musí obsahovat všechny nezařazené obdélníky, zařaď do ní zbývající obdélníky a konec.

3. Pro všechny nezařazené obdélníky spočítej rozdíl ploch o které se zvětší obdélníky první a druhé skupiny začleněním nezařazeného obdélníku.

4. Vyber obdélník z 3. který má maximální rozdíl ploch a zařaď ho do skupiny, jejíž celkový obdélník se rozšíří méně. Pokračuj krokem 2.

Algoritmus dělení uzlu (lineární složitost)

1. Vyber první dva obdélníky

- 1.1 Pro každou dimenzi najdi klíče s maximem minima a minimem maxima, stanov „separační vzdálenost“ mezi těmito klíči (minimum minus maximum).
- 1.2 Normalizuj separační vzdálenost tak, že vzdálenost intervalů podělíš rozsahem všech klíčů v dané dimenzi.
- 1.3 Vyber dvojici k, l s největší normalizovanou separační vzdáleností, zařaď je do první a druhé skupiny.

2. V případě, že jedna skupina obsahuje tak málo obdélníků, že pro zachování podmínky minima m musí obsahovat všechny nezařazené obdélníky, zařaď do ní zbývající obdélníky a konec.

3. Vezmi další nezařazený klíč a zařaď jej do takové skupiny, jejíž MBR vyžaduje menší rozšíření.

Funkce a operace nad geometrickými objekty

OGC:

Konstrukční operace – vytvářejí nové instance geometrie

`LineFromText`

`PolygonFromText`

.

.

Příklad:

```
INSERT INTO
```

```
  Countries (Name, Location)
```

```
VALUES
```

```
('Kenya',
```

```
  PolygonFromText('POLYGON ((x y, x y, x y,  
  ..., x y))',14))
```

'POLYGON (x y, x y, x y, ..., x y)' – WKT (the well-known textual representation)

Podpůrné a manipulační funkce:

`Dimension (g Geometry) : Integer`

Vrací dimenzi geometrie

`GeometryType (g Geometry) : String`

Vrací typ geometrie

`AsText (g Geometry) : String`

Konverze do textové reprezentace

`AsBinary (g Geometry) : Binary`

Konverze do binární reprezentace

`IsEmpty (g Geometry) : Integer`

Testuje prázdnou geometrii

`IsSimple (g Geometry) : Integer`

Test na jednoduchou geometrii geometrii

`Envelope (g Geometry) : Geometry`

MBR omezující obdélník geometrie

`X (p Point) : number`

x projekce bodu

`Y (p Point) : number`

y projekce bodu

NumPoints(l LineString) : Integer

Počet bodů v lomené čáře.

PointN(l LineString, n Integer) : Point

n-tý bod v lomené čáře.

ExteriorRing(p Polygon) : LineString

Obalová hranice areálu

NumInteriorRing(p Polygon) : Integer

Počet vnitřních hranic areálu

**InteriorRingN(p Polygon, n Integer) :
LineString**

n-tá vnitřní hranice areálu

NumGeometries(g GeomCollection) : Integer

Počet geometrických elementů ve složené geometrii.

**GeometryN(g GeomCollection, n Integer) :
Geometry**

n-tý geometrický prvek ve složené geometrii

Měřicí funkce:

`Length(1 LineString) : number`

Délka lomené čáry (neproblematická operace)

`Length(1 Polygon) : number`

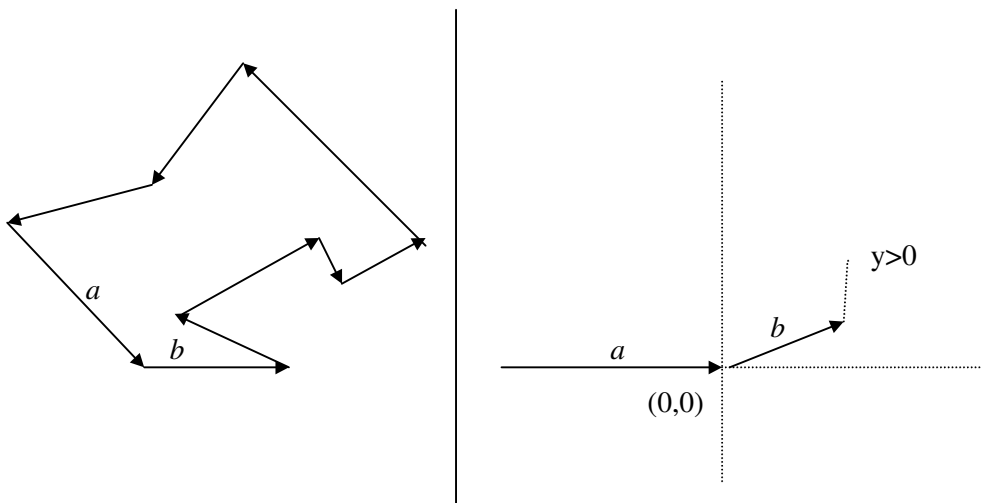
Plocha hranic areálu

`Area(1 Polygon) : Double Precision`

$$A = 1/2 \sum_{\text{Hranice}} \sum_{\text{Body}} (x_i - x_{i+1})(y_i + y_{i+1})$$

Algoritmus - Určení orientace polygonu

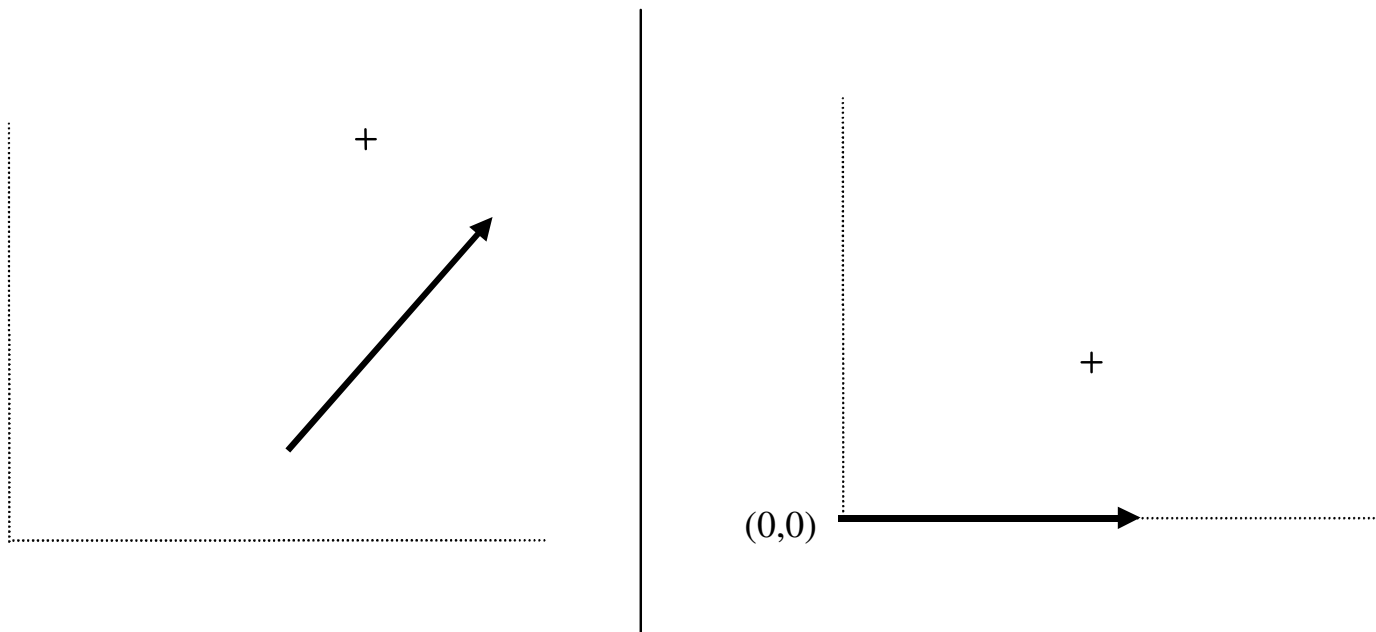
1. Vyber z hranic oblastí takovou hranici a tři po sobě jdoucí její body tak, aby střední bod měl minimální souřadnici y (ze všech souřadnic y v polygonu) a první bod měl souřadnici y větší, než bod prostřední.
2. Rotuj souřadnou soustavu tak, aby orientovaná úsečka definovaná prvními dvěma body splynula s osou x v kladném směru.
3. Znaménko souřadnice y posledního bodu určuje orientaci polygonu.



Distance(g1 Geometry, g2 Geometry) : Double Precision

Vzdálenost dvou geometrických objektů

Poloha bodu vůči úsečce:

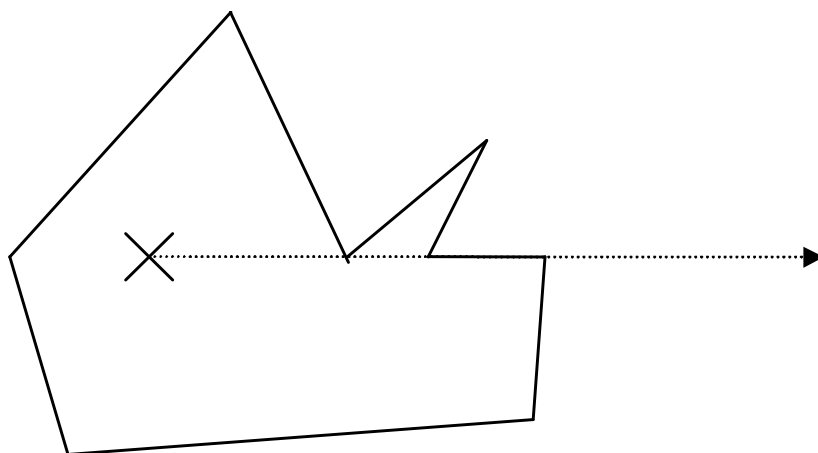


Rotace bodu za účelem určení jeho polohy vzhledem k úsečce

$$x' = x \cdot \cos(\varphi) - y \cdot \sin(\varphi)$$

$$y' = x \cdot \sin(\varphi) + y \cdot \cos(\varphi)$$

Poloha bodu vůči polygonu:



Úniková polopřímka z polygonu

Algoritmus - Bod v polygonu

1. Najdi v bodech polygonu bod, jehož y -ová souřadnice je různá od souřadnice bodu, který testujeme. Jestliže neexistuje, ukonči proceduru s výsledkem **LEŽÍ_NA_HRANICI**. Necht' pp je polopřímka vycházející z testovaného bodu rovnoběžná s osou x v kladném směru.
2. $nPrus := 0$
3. Od vybraného bodu postupně procházej všechny úsečky a proved' body 4 – 7.
4. Leží-li testovaný bod na úsečce, ukonči proceduru s výsledkem **LEŽÍ_NA_HRANICI**.
5. Má-li úsečka vlastní průsečík s polopřímkou pp , potom $nPrus := nPrus + 1$
6. Končí-li úsečka na polopřímce a začíná-li mimo polopřímku, stanov podle počátku úsečky $odkud := POD$ nebo $odkud := NAD$
7. Začíná-li úsečka na polopřímce a končí-li mimo polopřímku a pokračuje-li do jiné poloroviny, než je stav proměnné $odkud$, potom $nPrus := nPrus + 1$.
8. Je-li $nPrus$ sudý, ukonči proceduru s výsledkem **VNĚ**.
9. Je-li $nPrus$ lichý, ukonči proceduru s výsledkem **UVNITŘ**

Množinové operace

`Intersection (g1 Geometry, g2 Geometry) : Geometry`

`Difference (g1 Geometry, g2 Geometry) : Geometry`

`Union (g1 Geometry, g2 Geometry) : Geometry`

`SymDifference(g1 Geometry, g2 Geometry) : Geometry`

`Buffer (g1 Geometry, d Double Precision) : Geometry`

`ConvexHull(g1 Geometry) : Geometry`

Bod – bod

Triviální operace. Pozor, pro příslušnost bodu k množině je nutné použít vhodnou přístupovou metodu k prostorovým datům.

Bod – lomená čára

Vzájemná poloha úsečka X bod.

Bod – oblast

Poloha bodu vůči polygonu

Lomená čára – lomená čára

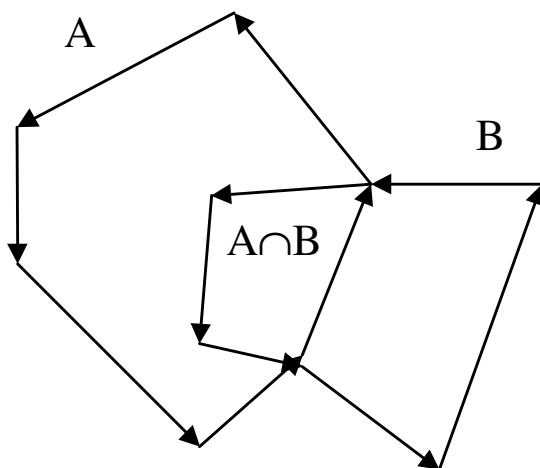
Poloha dvou úseček

Lomená čára – oblast

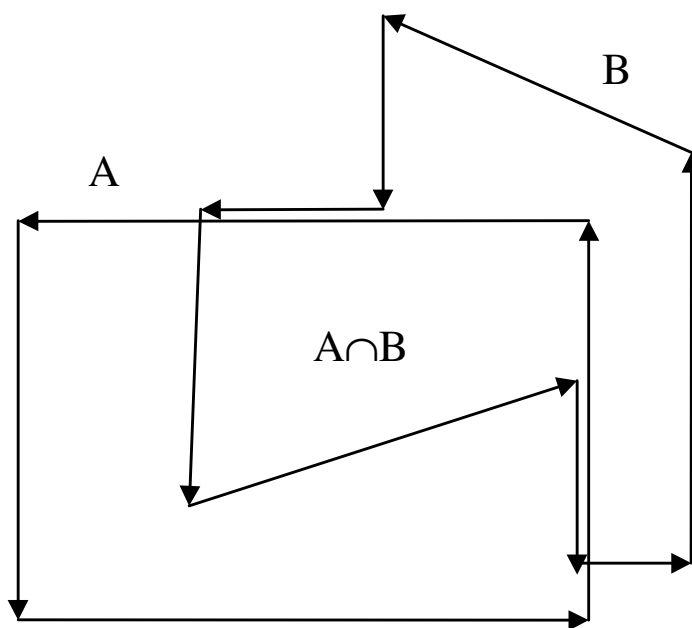
Algoritmus 13 - Průnik lomené čáry s oblastí.

1. Vstup: oblast a lomená čára.
2. Ze vstupní lomené čáry vytvoř seznam P segmentů lomené čáry takových, které buď neprotínají hranice oblasti, nebo jsou celé tečné.
3. Ze seznamu P vytvoř seznam $S \subseteq P$ takový, že libovolný vnitřní bod každého segmentu z S leží uvnitř oblasti.
4. Zřetěz segmenty z S do “co nejdelších” lomených čar, a výsledek pošli na výstup

Oblast – oblast



Průnik dvou oblastí



Průnik oblastí s tečnými hranicemi

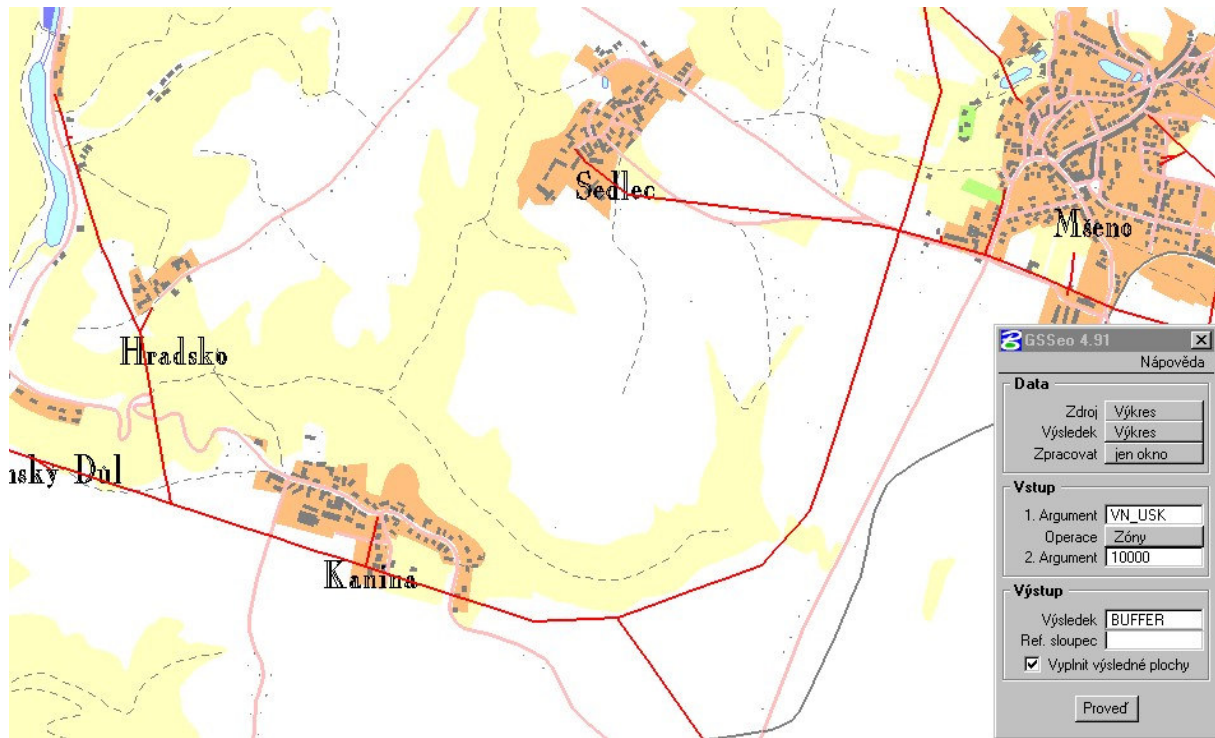
Algoritmus 15 - Průnik dvou oblastí

- 1. Vstup: dvě orientované oblasti.**
- 2. Všechny hrany hranic oblastí modifikuj tak, aby se vzájemně neprotínaly, mohou však splývat s hranami hranic z druhé oblasti. Potom mají tyto vlastnosti**
 - hrana splývá s jinou z druhé oblasti
 - hrana leží celá uvnitř druhé oblasti
 - hrana leží celá vně oblasti
- 3. Do seznamu zařaď ty hrany, které buď, leží celé v druhé oblasti, nebo splývají s nějakou hranou z druhé oblasti, se kterou mají stejnou orientaci, (totožné hrany jen jednou).**
- 4. Z vybudovaného seznamu zřetěz hranice výsledné oblasti a výsledek pošli na výstup.**

Nástin důkazu, že 4. Je uskutečnitelný...

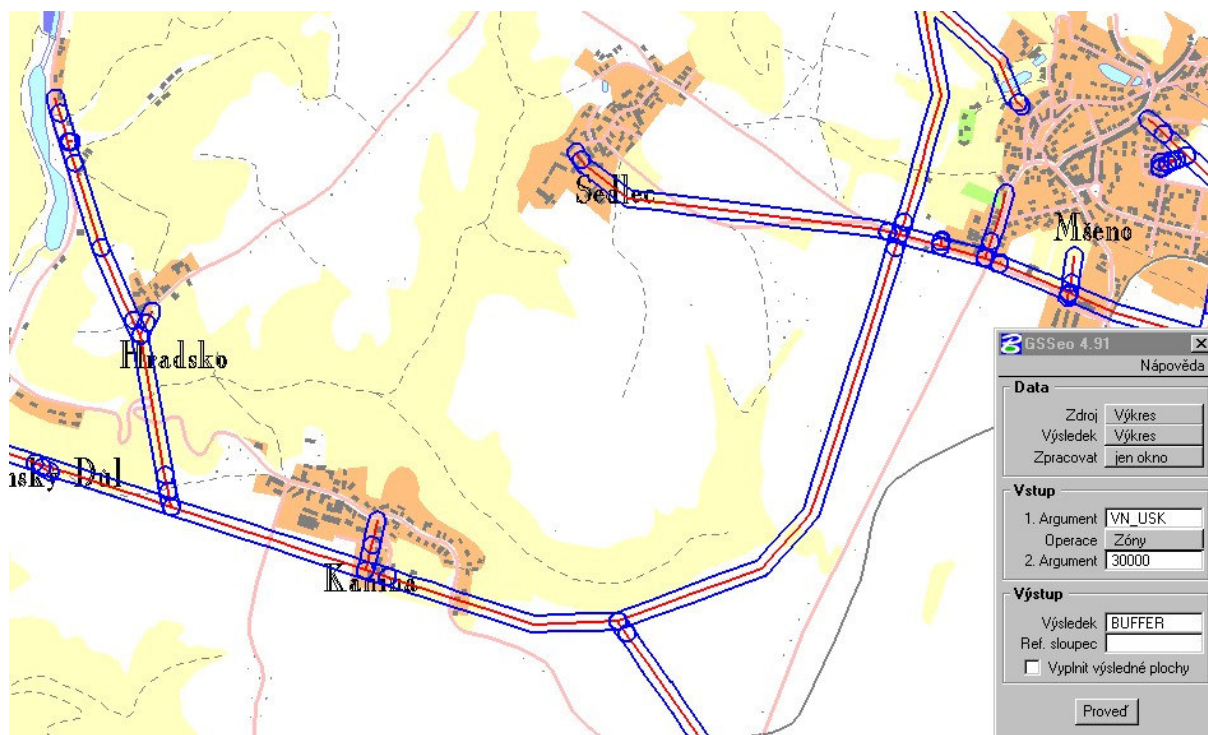
Příklad (systém GeoStore – GEOVAP):

V GIS systému máme (mimo jiné ...) vrstvu lesů reprezentované jako areály a vrstvu venkovních úseků vysokého napětí. Zajímá nás, kde lesy zasahují do bezpečnostního pásma 10 metrů kolem úseků vysokého napětí, a to jen pro určitý výřez území.



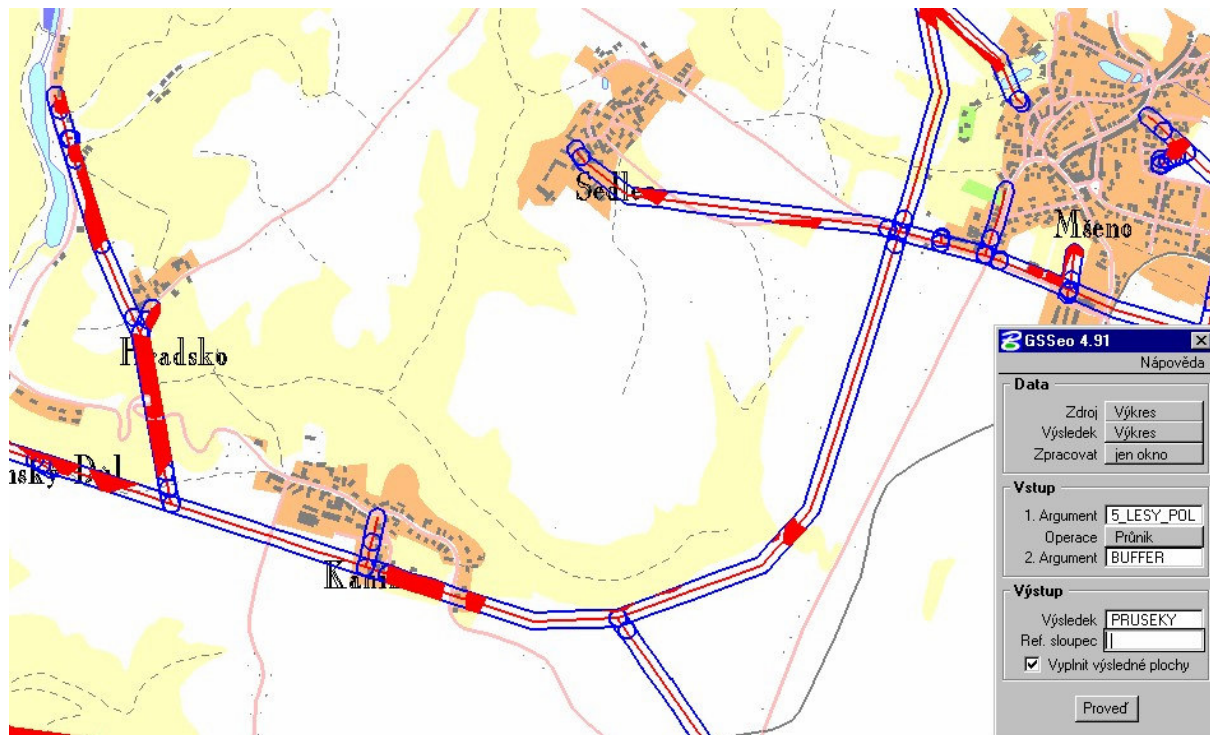
Obr 1.

Zvolíme úlohu „Zóny“, vyplníme vstupní argumenty úlohu – tabulka VN_USK (úseky) a poloměr 10000 mm. Po stisknutí tlačítka „Proveď“ jsou liniové prvky z tabulky VN_USK „obaleny“ zónami o zvoleném poloměru. Výstup bude mít nastavenou tabulku BUFFER. (Obr. 2).



Obr. 2

Zvolíme úlohu „Průnik“, první argument bude tabulka, ve které jsou lesy, druhým argumentem bude tabulka BUFFER (po proběhnutí 1) je nastavena tato tabulka na zónách). Volitelně můžeme zvolit možnost vyplnění výsledných areálů. Po provedení dostaneme kýžený výsledek (Obr. 3).



Obr. 3.

Transformace souřadných systémů:

Vstup: Dva seznamy bodů, které si „odpovídají“

$$\{ [x_1, y_1] \dots [x_n, y_n] \}$$

$$\{ [u_1, v_1] \dots [u_n, v_n] \}'$$

(Transformujeme $[u, v]$ do $[x, y]$ s co nejmenší chybou)

Výstup: Parametry transformačních rovnic

Lineární:

$$\begin{aligned} x &= f(u, v) = a_1 u + b_1 v + c_1 \\ y &= g(u, v) = a_2 u + b_2 v + c_2 \end{aligned}$$

Bilineární:

$$\begin{aligned} f(u, v) &= a_1 u + b_1 v + c_1 uv + d_1 \\ g(u, v) &= a_2 u + b_2 v + c_2 uv + d_2 \end{aligned}$$

Kvadratická, kubická, obecná polynomiální ...

Podmínka:

$$\sum \text{dist}^2([x_i, y_i], [f(u_i, v_i), g(u_i, v_i)]) = \min$$

kde

$$\text{dist}^2([x_1, y_1], [x_2, y_2]) = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

Lineární regrese:

Příklad pro lineární transformaci

$$\Sigma (a_1 u_i + b_1 v_i + c_1 - x_i)^2 + (a_2 u_i + b_2 v_i + c_2 - y_i)^2 = \min$$

$$d\Sigma/da_1 = \Sigma 2 (a_1 u_i + b_1 v_i + c_1 - x_i) \cdot u_i = 0$$

$$d\Sigma/db_1 = \Sigma 2 (a_1 u_i + b_1 v_i + c_1 - x_i) \cdot v_i = 0$$

$$d\Sigma/dc_1 = \Sigma 2 (a_1 u_i + b_1 v_i + c_1 - x_i) = 0$$

Soustava normálních rovnic (pro $g(u,v)$ analogicky):

$$a_1 \Sigma u_i^2 + b_1 \Sigma u_i v_i + c_1 \Sigma u_i = \Sigma x_i u_i$$

$$a_1 \Sigma u_i v_i + b_1 \Sigma v_i^2 + c_1 \Sigma v_i = \Sigma x_i v_i$$

$$a_1 \Sigma u_i + b_1 \Sigma v_i + c_1 \cdot n = \Sigma x_i$$

Rastrová data v GIS



Typy rastrových dat používaných pro GIS technologie jsou stejná jako v počítačové grafice:

- binární
- polotónová
- víceúrovňová

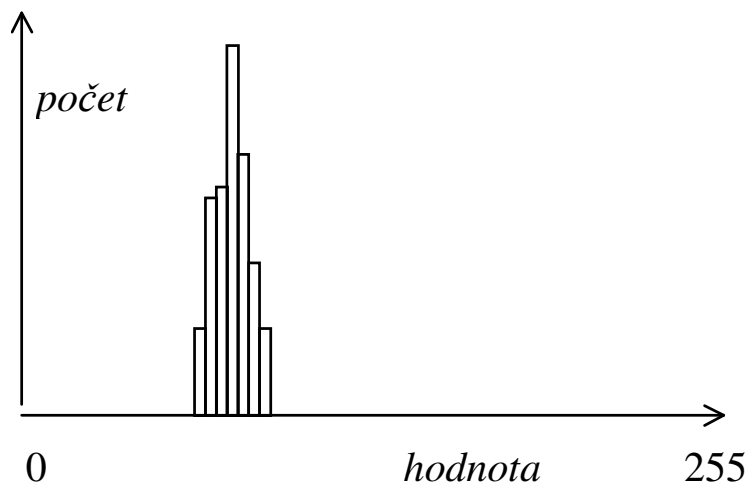
Pro další práci očíslováme sousedy pixelu P následujícím způsobem:

3	2	1
4	P	0
5	6	7

Sousedé 0,2,4,6 se nazývají *přímí* (d-) sousedé pixelu p.
Sousedé 1,3,5,7 se nazývají *nepřímí* (i-) sousedé pixelu p.

Definice - Histogram obrazu.

Necht' f je polotónový obraz barev 1..M. Jeho *histogramem* rozumíme konečnou posloupnost $h(f)=(h_1..h_M)$, kde, h_i je počet pixelů s barvou i . □



Obr. 18 - Histogram obrazu

Definice - Matice sousednosti

Necht' f je polotónový obraz barev 1..M. Jeho maticí *susednosti* rozumíme čtvercovou $M \times M$ matici $CM(f)=\{cm_{ij}\}$, kde, cm_{ij} je počet (přímou) sousedících pixelů o barvě i a j . □

Lineární filtrace

Bud' f polotónový obraz, $M > 0$. Položme

$$g(x,y) = H(M,x,y)$$

kde H je libovolná funkce, která v konstantním čase počítá novou hodnotu pixelu $g(x,y)$ z okolí pixelu (x,y) o rozměru M . □

Funkce H bývá někdy váženým průměrem pixelů a lze ji vyjádřit:

$$H(M,x,y) = \sum_{i=-M,M} \sum_{j=-M,M} h(i,j) * f(x+i,y+j)$$

a) Filtry zhlazující filtry

a)

1	1	1
1	1	1
1	1	1

b)

1	2	1
2	4	2
1	2	1



b) Filtry, které se snaží „zostřit“ obraz

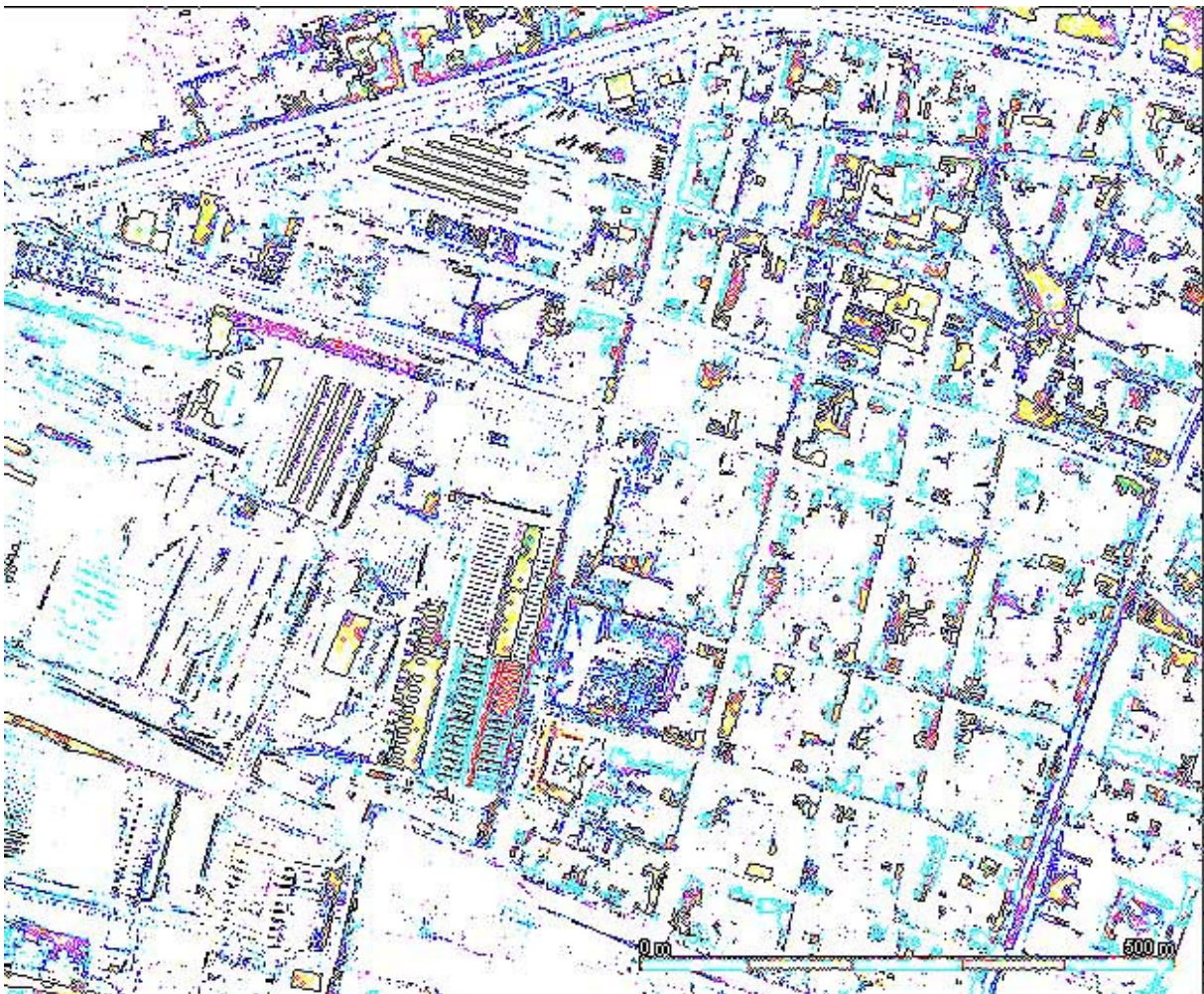
-1	-1	-1
-1	n	-1
-1	-1	-1



c) Detektory hran

Základním filtr této třídy přiřadí novému pixelu největší absolutní hodnotu ze dvou výsledků:

1	2	1	1	0	-1
0	0	0	2	0	-2
-1	-2	-1	1	0	-1



Původní obraz



Zhlazovací filtr:



Zostrující filtr:

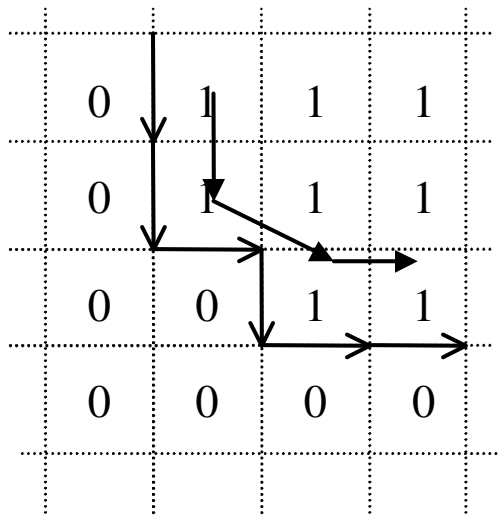


Detektor hran:



Kontura (hranice) oblasti v binárním obrazu

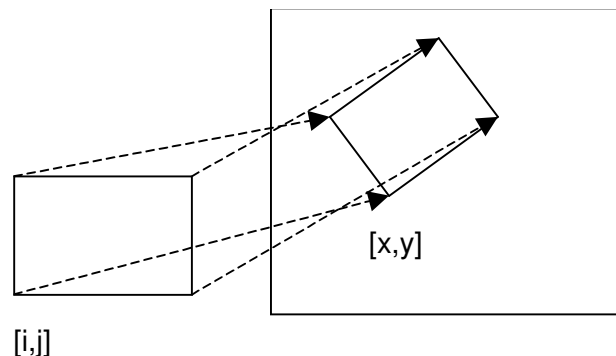
Necht' O je libovolná oblast (množina složená z jedniček) v binárním obrazu, Konturou (hranicí) oblasti O rozumíme všechny pixely patřící této oblasti, které mají nulového d -souseda. \square



→ vektorově

→ rastrově

Transformace obrazu



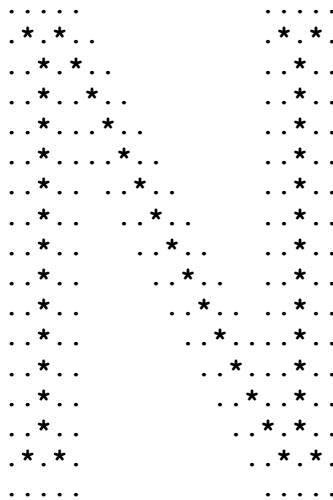
4. Určíme bodové objekty ve zdrojovém obrazu, jejichž (kartografické) souřadnice jsou známy. Například přesně odečtené z mapy, geodeticky zaměřené apod.
5. určíme transformační funkce z nového do starého obrazu (komerční produkty většinou poskytují polynomiální transformaci založené na metodě nejmenších čtverců).

$$\begin{aligned}i &= F(x,y) \\j &= G(x,y)\end{aligned}$$

kde (i,j) značí souřadný systém originálního obrazu, (x,y) souřadný systém obrazu nového.

6. fázi procházíme cílový obraz, pomocí transformačních funkcí F a G se „díváme“ do originálu a počítáme hodnotu pixelu. Podle toho, z jakého okolí zdrojového pixelu určíme výslednou hodnotu pixelu nového, se hovoří o metodách
 - nejbližší soused
 - bilineární transformace
 - konvoluce okolí $M \times M$

První případ prostě přenesou hodnotu pixelu do nového obrazu, v dalších případech se výsledná hodnota se počítá z jistého okolí pixelu v originálním obrazu.



Obr. 20 - Skelet binárního obrazu: binární obraz je reprezentován znaky ., jeho skelet znaky *.

Definice 12 - Skelet

Necht' R je množina pixelů, B její hranice (kontura), P bod v R . *Nejbližší sused bodu P na hranici B* je bod M z B takový, že pro každý bod M' z B , M' je různý od M je vzdálenost PM' větší nebo rovna vzdálenosti PM . Má-li bod P více než jednoho nejbližšího suseda, nazývá se *bodem skeletu množiny R* . Sjednocení všech bodů skeletu tvoří *skelet množiny R* .□

Algoritmus 16 - Určení skeletu

1. Urči hranici (konturu) $B(R)$ množiny R .
2. Urči množinu násobných pixelů $M(R)$ v hranici $B(R)$
3. Je-li $B(R) = M(R)$, skonči.
4. Polož $R = R - (B(R) - M(R))$ a pokračuj krokem 1.

(a)

A	A	A
0	P	0
B	B	B

(b)

A	A	A
A	P	0
A	0	2

(c)

A	A	C
0	P	2+
B	B	C

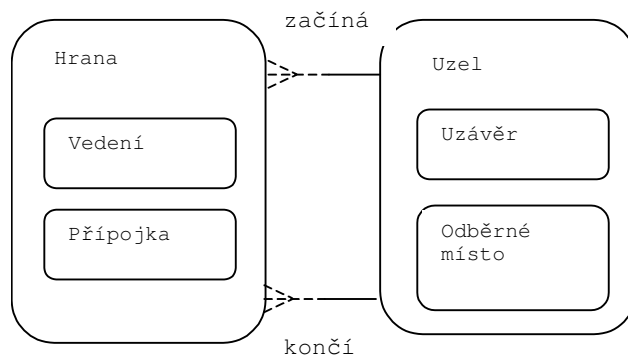
Pixel označený 2 je hraniční pixel, pixel označený 2+ je hraniční nebo násobný pixel.

(a), (b): Alespoň jeden pixel ze skupiny pixelů A, B je nenulový
(c): Alespoň jeden pixel ze skupiny C musí být nenulový. Pokud jsou oba nenulové, pak může být hodnota pixelů ve skupinách A i B libovolná. Pokud je jeden pixel skupiny C nulový, musí být alespoň jeden pixel skupiny A i skupiny B nenulový.

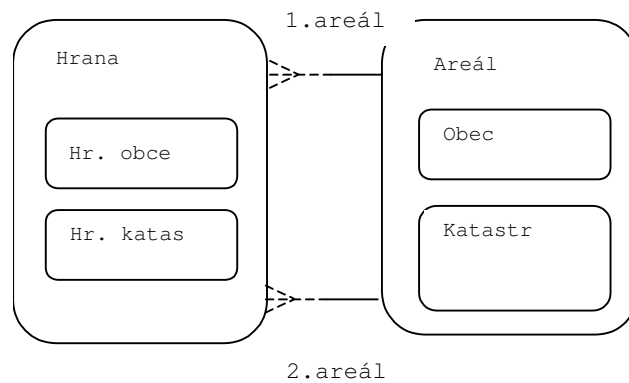
Topologické úlohy:

Byly efektivně řešeny před tím, než byly vůbec GIS technologie vymezeny, přesto je můžeme považovat za součást analytického jádra topologicky orientovaných GIS.

Základní datová struktura síťového grafu:



Základní datová struktura areálového grafu:



Nejčastější úlohy:

- Trasování grafu - vyber všechny uzly/hrany, které jsou “napájeny” z daného uzlu, hodně používaná úloha pro dispečery sítí, modelování situací „co se stane, když?“.
- Nejkratší cesta z uzlu do uzlu. Používá se klasický Dijkstrův algoritmus. Lze výhodně využít vlastnosti, že uzly grafu jsou prostorově lokalizovány.

Algoritmus „Minimální cesta (Best search)“

Nechť (U, H) je graf s nezáporně ohodnocenými hranami, váhu libovolné hrany $h \in H$ označme $w(h)$. Nechť dále každý uzel $u_i \in U$ má 2D souřadnice (x_i, y_i) . Pro libovolné uzly u, v označme $d(u, v)$ jejich vzdálenost v E_2 . Pro libovolnou hranu $h_i = (u_i, v_i)$ nechť dále je $d(u_i, v_i) \leq w(h)$ – platí trojúhelníková nerovnost metrického prostoru E_2 . Potom pro libovolné uzly $u_0, v \in U$ následující postup vede k nalezení minimální cesty z u_0 do v .

Nechť $u_0, u_1, \dots, u_n = v$ je kritická cesta.

1. Inicializace: Pro každý uzel $u_i \in U$, $u_i \neq u_0$ položme $d_path_i = \infty$, $d_path_0 = 0$, a položme každý uzel $u_i \in U$ $c_path_i = NULL$.
2. Výběr pivota: Položme $c_path_i = d_path_i$ pro takový u_i , pro který je $c_path_i = NULL$, $d_path_i < \infty$ a pro který je $d_path_i + d(u_i, v)$ minimální. Když neexistuje – končíme, cesta neexistuje. Je-li $u_i = v$, potom konec c_path_i je délka minimální cesty a provedeme zpětný chod.
3. Expanze: Pro všechny uzly $u_k \in U$ takové, že existuje hrana $h_k \in H$, $h_k = (u_i, u_k)$ (u_i je pivot z předchozího kroku) položme $d_path_k = \min\{d_path_k, c_path_i + w(h_k)\}$ a pokračujeme 2.

A) Má-li u_i přiřazeno dočasné ohodnocení, které je rovno hodnotě kritické cesty, potom bude někdy vybrán jako pivot.

B) Je-li u_i vybrán jako pivot a je-li mu přiřazena trvalá hodnota kritické cesty potom u_{i+1} nebyl vybrán jako pivot před ním.

C) u_{i+1} je potom přiřazena hodnota kritické cesty

D) u_n je přiřazena hodnota kritické cesty

A) C) a D) jsou triviální, B) SE dá dokázat (snad)

– Problém obchodního cestujícího.

Topologicko-geometrické úlohy:

- **Vytváření topologických vazeb na základě geometrických vlastostí objektů; jedná se o vytvoření příslušného typu grafu (uzel-hrana, hrana-hrana, areálový graf). Hojně se využívá přístupových metod pro geometrické objekty.**
- **Generování oblastí z hran areálového grafu.**
- **Identifikace hran areálového grafu.**
- **Generování vyšších územních celků areálového grafu.**
- **Kontrola konzistence geometrických a topologických vlastností dat (kontrola shody umístění uzlu a konce hrany s ním incidentní, kontrola křížení hran, kontrola stupňů uzlových bodů a další kontroly)**

