PDB výpisky

Prostorové DB

Definice 2.2.1 Pracovní definice prostorového databázového systému: Prostorové databázové systémy jsou databázové systémy, jejich DDL a DML zahrnují *prostorové datové typy*, *prostorové datové typy* jsou podpořeny i na *implementační úrovni*, takže je možné efektivně provádět operace indexace, vyhledávání, spojování (join),...

Geometrické modely musejí být schopny zachytit to, co chceme uložit. Jednak tedy oddělené entity a potom také skupiny entit, které spolu nějak prostorově souvisejí. Pro oddělené entity jsou **geometrické modely** typicky tyto:

- **body** — města,...
- **lomené úsečky** — řeky, silnice, vedení,...
- **uzavřená lomená úsečka** (polygon) — ohraničení oblastí
- **oblast** (vyplněná uzavřená lomená úsečka) — les, jezero, město,
- ...

Pro **popis prostoru** potom můžeme uvážit např.:

- **plošné oddíly/mapy** — podobně jako vyplněné oblasti, ale modelují i sousednost, souvislost, apod. (např. katastrální mapy, městské části, územní pokrytí, Voronoi,...)
- **sítě v prostoru** — jedná se grafové záležitosti, které umožní modelovat silnice, železnice, elektrické sítě, telefonní vedení apod.
- **vnořené plochy**
- **digitální modely terénu** apod.

V počítači nejsme schopni zachytit reálná čísla, pouze čísla racionální a velmi často jen čísla desetinná, s různou mírou přesnosti. **Zachycujeme tedy diskrétní prostor.**

Řešení problematiky je v tom, že v průběhu geometrických operací se již nadále neprovádí další výpočty průsečíků. Dochází tak k oddělení typů a operací nad prostorovými daty a ošetření číselných problémů korektně ke geometrickému modelu. Mezi hlavní přístupy, jak toto řešit je možné uvést:

- **Simplexy** – nejmenší nevyplněné objekty dané dimenze. Často se tedy označují jako d-simplexy. 0-simplex je potom bod, 1-simplex je úsečka, 2-simplex je trojúhelník, 3-simplex je čtyřstěn atd. Lze jednoduše vypozorovat, že d-simplex sestává z d+1 simplexů rozměru d-1. Takové tvořící elementy se potom nazývají styky (faces). Kombinace simplexů do složitějších struktur je povolena jen tehdy, pokud průnik libovolných dvou simplexů je styk.
- **Úplné popisy**, či **deskriptory** (**realms**) jsou vlastně jakýmsi souhrnným popisem všech objektů v databázi. Formálněji je to potom množina bodů, úseček, případně vyšších celků, které mají tyto vlastnosti:
    - každý (koncový) bod je bodem sítě
    - každý koncový bod úsečky (složitějšího útvaru) je bodem sítě
    - žádný vnitřní bod úsečky (složitějšího útvaru) není zaznamenán v síti
    - žádné dvě úsečky (složitější útvary) nemají ani průsečík, ani se nepřekrývají

Dají se stanovit kritéria, která mohou nějaký **návrh či systém ohodnotit** z hlediska kvality a úplnosti podpory práce s prostorovými daty. Kritéria jsou:

- **„vzhledy“ datových položek** musí být uniformní v rámci množinových operací nad množinami objektů tvořící data i případný výsledek;
- **systém musí obsahovat** formální definice dat a funkcí nad prostorovými datovými typy;
- v předchozím bodě zmíněné definice musejí **zohledňovat aritmetiku s konečnou přesností**;
- v systému musí být zahrnuta **podpora pro konzistentní popis prostorově souvisejících objektů** — objekty úzce souvislé, nebo dokonce těsně sousedící musí využívat pro popis shodné podčásti,...;
- **definice dat a operací by měla být nezávislá** na konkrétním SŘBD, ale přitom s daným SŘBD úzce spolupracující.

Relační algebru musím rozšířit o další typy (a operace, viz níže) tak, aby kromě atomických typů bylo možné zaznamenat i prostorová data. Vzhledem k předchozímu zvolíme jistou minimální množinu konstrukcí, které umožní sestavit prakticky libovolný objekt ve 2D:

- **bod** (point)
- **lomená úsečka** (line)
- **region**, který se dále dělí na
    - **uzavřená lomená úsečka** (polygon, pgon) — význam hranice, obdoba kružnice
    - **ohraničená plocha** (area) — plošně vyplněná oblast, obdoba kruhu

✉ Typy PGON, AREA, REG, EXT (REG + úsečka – objekty, které nejsou bezrozměrné), GEO (všechny).

V zásadě máme 3 **kategorie operací**, které je možné aplikovat:

1. **predikáty** — vstupem operací jsou různé hodnoty, z nichž alespoň jedna je nějakého z typů GEO a výsledkem je pravdivostní hodnota;
2. **geometrické relace** — vstupem je hodnota, či množina hodnot jednoho či více typů ze skupiny GEO a výsledkem je jedna či více hodnot z typu GEO, nebo relace, které takové typy obsahuje alespoň v jednom atributu;
3. **výpočetně náročné operace** — i operace z předchozí kategorie mohou být velmi náročné, ale typově se jedná o zjišťovaní vztahů, v této kategorii jsou výsledkem nově získané hodnoty různých i negeometrických typů, jedná se i o operace množinové.

Dále je třeba rozšířit stávající operace, aby pracovaly s novými typy:

- projekce
- selekce
- fúze (projekce se spojením)
- „windowing“
- Ořezání

Mapa, což je vlastně tabulka s prostorovými daty v relační databázi s podporou prostorových dat, je potom množina n-tic, kde jeden z atributů je právě typu region.
Algebra nad takovýmito objekty potom obsahuje (z hlediska prostorových dat) operátory **vložení**, **primitivní operace nad regiony** a **množinové geometrické operace**.

ROSE (Robust Spatial Extension)

- **algebra** navržená pro podporu prostorových databázových systémů vycházejících (do jisté míry) z relačních systémů.
- **vychází z deskriptorů** (realms) a složitější objekty získává jejich skládáním. Základem jsou tedy body, úsečky a oblasti (plochy, které mohou mít i „díry“).

Vnoření objektů:

- uvnitř (plošně) — může se dotýkat ohraničujícího objektu
- hranově vnořený — nedotýká se žádnou hranou
- vrcholově vnořený — nedotýká se ani vrcholy

Disjunkce:

- plošně disjunktní — můžou se dotýkat
- hranově disjunktní — nedotýkají se hranou
- zcela (vrcholově) disjunktní — nedotýkají se ani vrcholy

Definice 3.1.1 R-cyklus *je taková uzavřená lomená úsečka, která je vytvořena podle pravidel ukládání deskriptorů (realms), kde lomená úsečka je tvořena posloupností* n *úseček* $s_1, \ldots, s_n$ *a zároveň platí, že konec úsečky* s_i *je shodný se začátkem úsečky* $s_{(i+1) \bmod n}$. *Přitom se žádné dvě různé úsečky* s_i, s_j, *kde* $i, j \in \{1, \ldots, n\}$ *nikde neprotínají.*

Definice 3.1.2 R-plocha f *je dvojice* (c, H) *taková, že* c *je R-cyklus,* $H = \{h_1, \ldots, h_m\}$ *je množina R-cyklů a platí:*

- $\forall i \in \{1, \ldots, m\}$: h_i *je hranově vnořený v* c;
- $\forall i, j \in \{1, \ldots, m\}$, $i \neq j$: h_i *a* h_j *jsou hranově disjunktní;*
- *žádný jiný cyklus není možné ze segmentů popisující plochu* f *dále vytvořit.*

Pozn.: poslední podmínka zaručuje jednoznačnost reprezentace.

Definice 3.1.3 *Nechť* $f = (f_0, F)$ *a* $g = (g_0, G)$ *jsou dvě R-plochy. Říkáme, že* f *je plošně obsažena (vnořena) v* g *právě tehdy když:*

- f_0 *je plošně vnořena v* g_0 *a zároveň*
- $\forall g \in G$:
  - g *je plošně disjunktní s* f_0 *nebo*
  - $\exists f \in F$: g *je plošně vnořené v* f.

- specifikuje typ *regions* tak, že to je *množina hranově disjunktních R-ploch*
- rozšiřuje definici plošného vnoření z dvou R-ploch na dvě množiny R-ploch, aby bylo možné hovořit o plošném vnoření u instancí typu regions.

Definice 3.1.4 *Nechť* F, G *jsou dvě hodnoty typu regions, potom* F *je* plošně vnořena *v* G *právě tehdy když:* $\forall f \in F\ \exists g \in G$: f *je plošně vnořena v* g.

- ROSE specifikuje krom typů pro reprezentaci prostorových objektů i všechny operace s hodnotami těchto typů.
- má určité **nevýhody**:
  - chybějí operace pro vytvoření nového geometrického uskupení (voronoi, střed, konvexní obálky);
  - integrace DBS a deskriptorů není jednoduchá (při změně atributů deskriptoru je třeba vyhledat příslušný objekt a u něj změnit hodnoty atributů definující deskriptor).

Určování vztahů

- Pro 0D, 1D a 2D objekty existuje 52 platných kombinací pro určování vztahů mezi nimi
- Kombinací následujících operací lze realizovat všech 52:
  - 5 operací: **dotek**, **uvnitř**, **přes**, **přesah**, **disjunkce**;
  - 3 operátory na **extrakci hranice**.

Požadavky na práci s prostorovými datovými typy

- **Existence prostorových datových typů**
- **Grafické znázornění výsledků** — i když je možné hodnoty prostorových typů reprezentovat textově, tak především grafická reprezentace je žádána a očekávána.
- **Grafická kombinace několika výsledků** — jednotlivé dotazy často poskytují jen dílčí výsledky, kombinace více dotazů může tak poskytnout mnohem lepší obrázek; např. zobrazím část území a na něm jen cesty a sídla, v dalším kroku však chci doplnit řeky apod.
- **Zobrazení i s kontextem** — výběrem jednoho města jistě nemyslíme zobrazení jen údajů po jeho hranice, ale i přilehlé okolí, protože další náš zájem může být právě tímto směrem.
- **Kontrola stavu displeje** — díky různým operacím na displeji (výběr objektů do dotazů apod.) je možné, že dojde k poškození některých grafických objektů, případně je možné, že poslední dotaz nedodal informaci, co

bychom potřebovali, potom je nutné obnovit původní stav displeje, případně se vrátit o několik kroků zpět apod.
- **Dialog** — systém by měl vést s uživatelem/programátorem dialog, neboť výsledek často není získán jediným dotazem.
- **Různé typy zobrazení** — např. drátový model v 3D, stínovaný, realistický…
- **Legenda, popisky** —také údaje o měřítku
- **Změna měřítka**
- **Výběr podoblastí**

GUI

Za jistý standard by se dal považovat, který má typicky 3 okna.

1. textové okno pro textovou reprezentaci objektů prostorových datových typů i standardních typů;
2. grafické okno pro grafické zobrazení objektů prostorových datových typů a vstupy do dotazů;
3. textové okno pro vkládání dotazů a zobrazování systémových hlášení.

Interakce uživatele se systémem tak probíhá jak na úrovni textové, tak na kombinované, kdy dotaz je částečně formulován textem a doplněn je interakcí na úrovni grafické.

Reprezentace hodnot prostorových datových typů

PDT mají nestandardní a proměnlivou velikost. Je navíc nutné udržet kompatibilitu mezi dvěma hledisky:

Hledisko SŘBD

- **prostorové datové typy stejné jako hodnoty jiných typů** — pro SŘBD by bylo ideální, aby nové typy mohl zpracovávat jako stávající, což není možné;
- **různorodá (hodně velká) velikost dat** — velikost jednotlivých datových položek se může i pro hodnoty jednoho typu významně lišit a může nabývat vysokých hodnot;
- **hodnoty na disku (i více stránek)** — zatímco typicky ukládáme do jedné stránky i několik řádek tabulky relační DB, tak jedna datová položka prostorového datového typu může sama zabrat i několik stránek;
- **možnost natažení do operační paměti** — zpracování hodnot v různých operacích vyžaduje přítomnost dat v operační paměti, pro určité extrémy by to nemuselo být dosaženo;
- **základní operace specializované dle typu** — jak již bylo naznačeno výše, jedna a ta samá operace z konceptuálního pohledu může a často i má zcela odlišnou implementaci podle konkrétních typů prostorové DB a v extrémech i podle hodnot jednoho a téhož typu.

Hledisko prostorové algebry

- **hodnoty odpovídají ADT programovacího jazyka** — hodnoty prostorových datových typů se v principu musejí mapovat na konkrétní datové struktury konkrétního, implementačního programovacího jazyka;
- **jde o nějakou datovou strukturu (typicky složitou)** — kritické je to, že se jedná o vnitřně strukturovaný typ, i hierarchicky, což dále ztěžuje manipulaci;
- **podpora algoritmů numerické geometrie**
- **algoritmy neoptimalizovány pro jeden algoritmus, ale tak, aby podporovaly stejně všechny** — vyladění algoritmu pro jednu, či jen několik kombinací parametrů může být fatální pro sestavy, kdy se tyto kombinace v datech prakticky nevyskytují.

Fyzické uložení dat

- Pro prostorová data se volí takový způsob uložení, aby byl konzistentní pro různou velikost dat.
- Pro každá prostorová data dojde k **vyčlenění té částí dat, které se nemění s charakterem vkládaného objektu**. Ostatní data se ukládají mimo, do zvláštních datových stránek, které tak nebrání rychlému zpracování

Krom základních informací je dobré ukládat i:

- **statická data proměnlivé délky** — PSS (plane sweep sequence), ta musí být uložena a často ve standardizované formě, aby jejich další zpracování bylo možné optimalizovat;

- **aproximace** — řada operací je výpočetně náročná, pokud se jedná o obecná data, pro jistá konkrétní data (např. kružnice, hyperkrychle, apod.) však mohou být jednoduchá, navíc vhodně zvolená aproximace může mít konstantní prostorovou náročnost, takže je často uložena v konstantní části také;
- **uložené hodnoty unárních funkcí** (plocha, průměr, střed,...) — hodnoty, které lze spočítat v době ukládání dat do DB by měly být všechny spočteny a uloženy, aby nedocházelo k jejich opakovanému výpočtu v době běhu dotazů/aplikace.

Indexování

Ve 2D, 3D atd. nelze z reprezentace hodnot jednoznačně určit uspořádání – předchůdce a následníka (resp. souseda). Řešením jsou metody indexace.

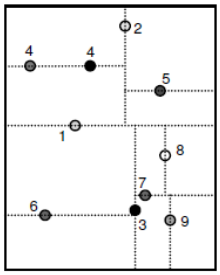
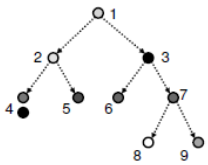
Mapování do 1D

... a následné užití známých 1D indexačních algoritmů.

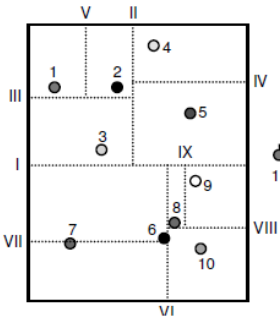
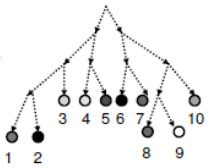
- Ztrácíme sousednost
- Transformace nemusí být realizovatelná
- Výsledek dotazů nemusí být transformovatelný zpět.

Stromy dělící prostor

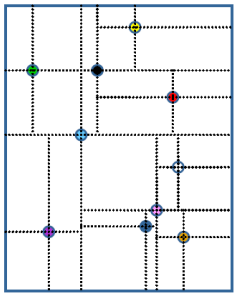
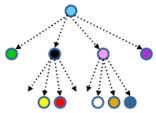
- **K-D-Tree**
  - **body jsou v uzlech stromu**
  - dělení prostoru hyperplochami (ve 2D = přímkami) na nejvyšší úrovni na 2 části
    - hyperplocha je rovnoběžná s osami souřadného systému ✉ eliminace 1 souřadnice
    - v každé hyperploše je bod (může být i více)
  - Operace:
    - Vyhledávání: OK
    - Vkládání: OK (Nevhodné vkládání může vést k degradaci stromu)
    - Mazání: Problém - musí dojít k znovuvložení celého podstromu

- **Adaptivní K-D-Tree**
  - **na každé straně hyperplochy je stejně bodů**
  - **body jsou v listech**
  - stanovená hranice počtu bodů v listech
  - eliminace nevýhody K-D-Tree v pořadí vkládání
  - Operace:
    - Vyhledávání: OK
    - Vkládání: OK
    - Mazání: OK (stačí odebrat dělicí plochu)

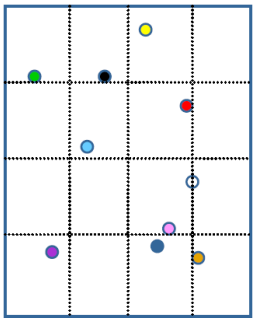
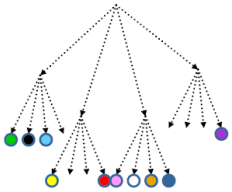



- **BSP Tree**
  - **jako Adaptivní K-D-Tree, ale plochy jsou nerovnoběžné se souřadným systémem**
  - **není adaptivní!** (tedy odolný proti změně dat) ✉ vyšší nároky na paměť
  - Binary Space Partitioning
  - Dělení tak dlouho, dokud počet bodů v podprostoru neklesne pod danou hodnotu
- **Quad-Tree**
  - Jako K-D-Tree, ale v místě bodu dělí na 2^n podstromů (n = dimenze prostoru)
  - Podstromy nejsou vyvážené a mohou se vyskytnout prázdné větve
  - Algoritmus vhodný pro: Body a regiony

- **Point Quad Tree**
  - Dělí v bodech
- **Region Quad Tree**

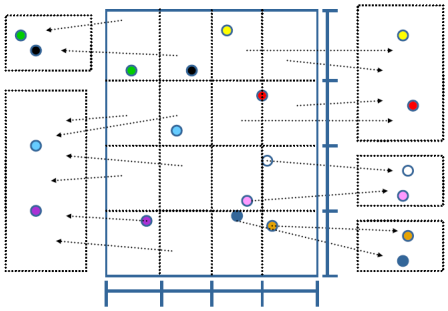
- Dělí v prostoru na prakticky shodné části

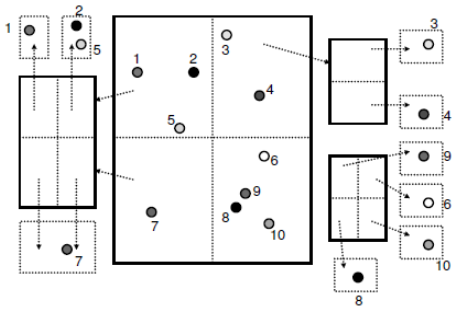
PM-quad-tree - regiony

Adaptivní hashování

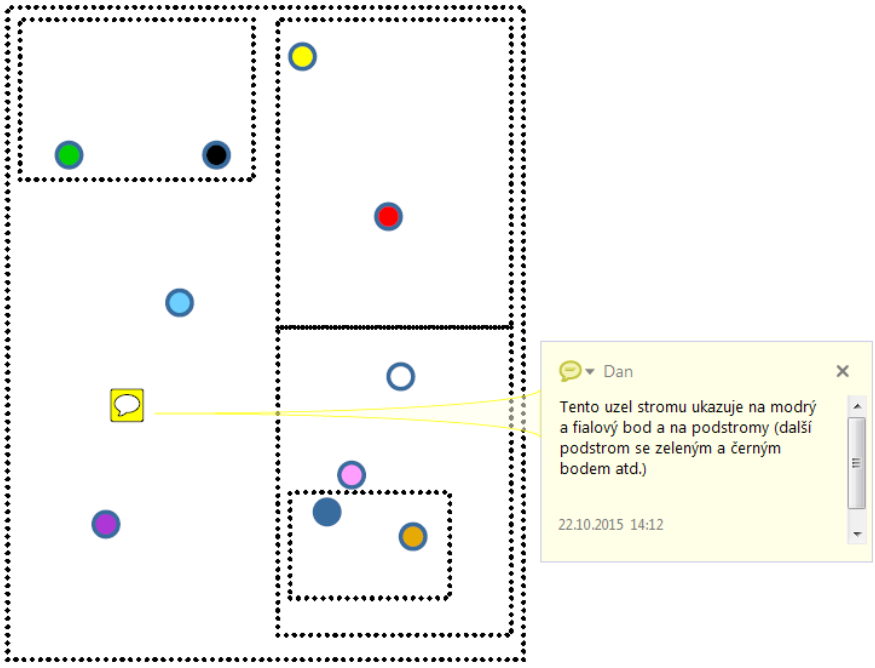
- **Grid File**
  - Prostor pokryt n-rozměrnou mřížkou (nemusí být pravidelná)
  - Adresář řadí každou buňku (i více) k datové jednotce (*bucket*)
  - Adresář (je velký) i mřížka jsou na disku (jen 2 přístupy na disk)
  - Počet řádků a sloupců je mocnina dvou
  - Operace:
    - Vyhledávání: OK
    - Vkládání: OK
      - Není lokální – nutná dělicí hyperplocha ke zvětšení adresáře
      - při vkládání shluku bodů může dojít k přetečení bucketu
    - Mazání: OK
      - Není lokální - odstranění hyperplochy je třeba prověřit
      - Pokud je dat málo, je možný zánik bucketu (protože data zanikla s ním nebo byla přesunuta)



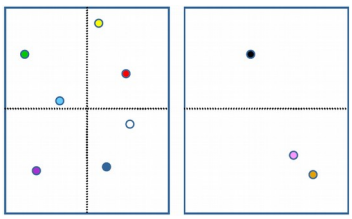
- **EXCELL**
  - **jako grid file, dělí na stejně velké jednotky**
  - **plošné dělení** (velký adresář, později hierarchie, přetokové stránky)
- **Two-Level Grid File**
  - **Dvě úrovně mřížky** – první odkazuje na druhou, což je Grid File sám o sobě
  - Operace:
    - Vkládání: OK (Je často lokální, ale přetečení není úspěšně vyřešeno)
    - Mazání: OK (Je často lokální)



- **BANG File**
  - Balanced And Nested GF, **datové buňky se překrývají**
  - Základem je interpolační Grid File – řeší exponenciální růst adresáře při nerovnoměrném rozmístění dat
  - Kombinace stromové a hashovací metody – hybridní (ale hlavně adaptivní)
    - Datová jednotka je v buňce, která je uložena ve vyváženém stromě
    - V buňce je předem dané maximum a minimum (ne 0) – pokud je víc bodů, vytvoří se nový podprostor a vyváží se strom



- **Twin Grid File**
  - **2x GF vedle sebe** – výsledkem jsou rovnoměrně rozložená data (využití prostoru až 90% oproti 69% samotného GF)
  - Jeden z GF je primární, druhý přetokový – jejich role se mohou libovolně prohazovat a hustoty mříží se dají měnit
    - Body se dávají implicitně do primárního GF. Pokud by přetekl, dají se do přetokového.
  - Operace:
    - Vyhledávání: OK (Možnost plné paralelizace)
    - Vkládání: OK (Možnost částečné paralelizace)

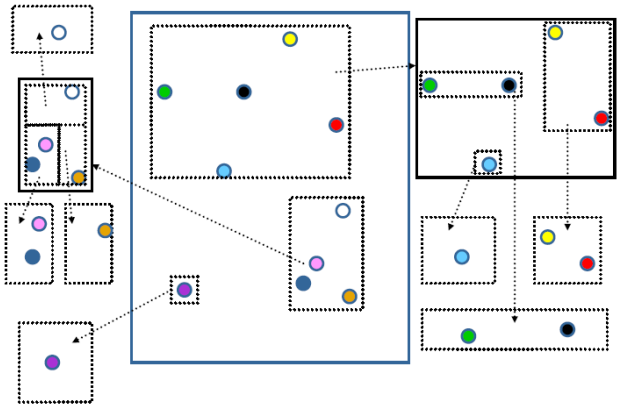


Vícerozměrné lineární hashování

- Malé nebo žádné adresáře ✉ vejdou se do paměti
- **MOLHPE**
  - ukazatele pro růst dat
  - datové jednotky stejných rozměrů
  - nevhodné pro nelineární distribuci
- **Quantile hashing**
  - jako MOLHPE, ale nerovnoměrně distribuovaná data "zrovnoměrňuje"
- **Z-hashing**

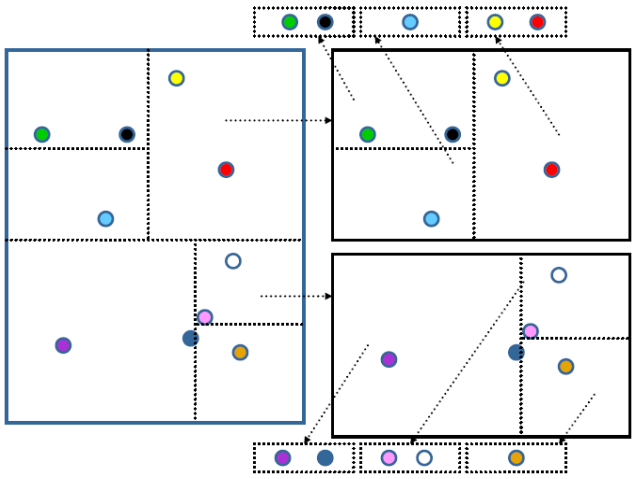
Hybridní hashování

- **Buddy tree**
  - **adresář je stromová struktura**
    - obecně nevyvážený strom s min. 2 položkami
    - ve stromě jsou ukazatele na nižší úrovně; v listech jsou ukazatele na buckety
    - **dělí se rekurzivně** hyperplochami rovnoběžnými s osami souřadného systému
    - Ve vnitřních uzlech se prostor omezí na MBB (minimum bounding box) vnitřních bodů - výraznější selektivita

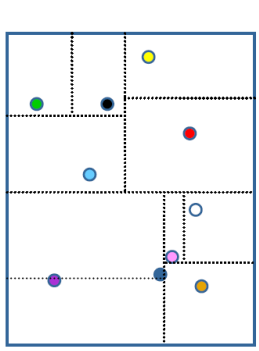


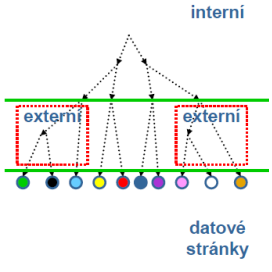
Přístup k bodům - stromy

- **K-D-B-Tree**
  - K-D Tree + B-Tree
  - Adaptivní K-D Tree
  - Balancovaný (B-Tree)
  - Operace:
    - Vkládání: OK
      - Může způsobit rozdělení, ale existují heuristiky pro optimální rozdělení
      - Propagace stromem
    - Mazání: OK (Slučování při podtečení)



- **hB-Tree**
  - Holey brick tree
  - Dělení uzlu je víceatributové
    - Fraktální struktura
    - BANG File
  - DAG – i když se jde ve stromu po jiných hranách, lze dojít do stejného místa
  - Paralelní verze
- **LSD tree**
  - **Adaptivní K-D Tree** ✉ výškově vyvážený
  - **Stromová (adresářová) struktura je externí**
  - Local Split Decision
  - Nejen pro prostorová data



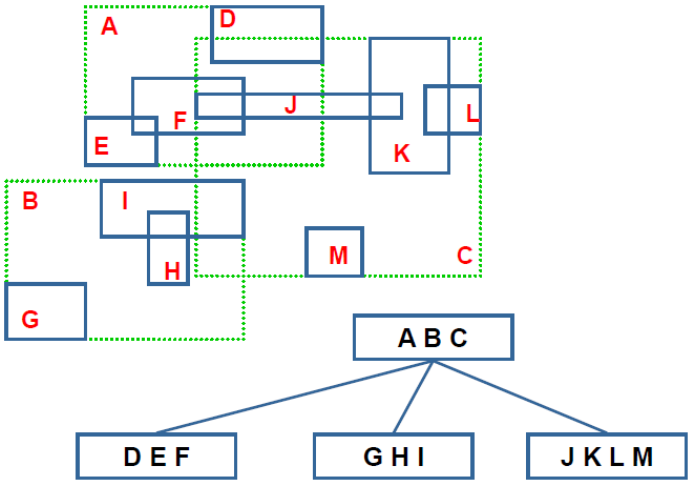


Indexování prostorových objektů

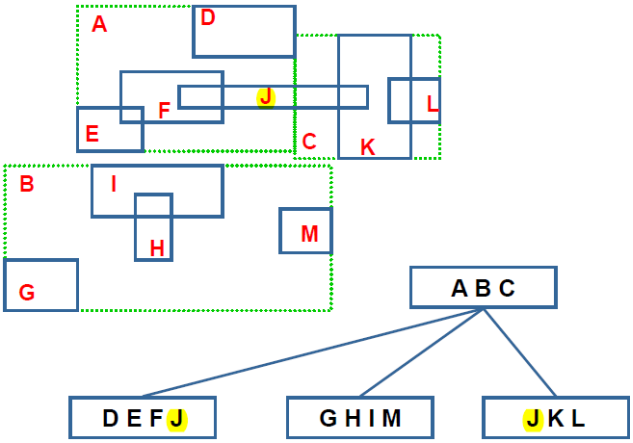
3 hlavní metody přístupu:

1. **Transformace** – mapování objektů na bezrozměrné objekty vícedimenzionálních prostorů, přesněji v *k*×*n*-rozměrném prostoru (obdélník ve 2D se tak stává bodem ve 4D);
    - Nedostatečná metoda – některé dotazy jsou nerealizovatelné, mapování je často složité nebo nemožné, výsledky se špatně interpretují
2. **Překrývání** — indexační struktury se překrývají, takže vzniká více vyhledávacích cest
    - Implementačně se buňky překrývají svými hranicemi.
    - V praxi tak dochází k tomu, že algoritmus je stejný, jen počet prohledávacích cest se zvětšuje, protože dopředu není jasné, ve které buňce je nakonec objekt uložen.
3. **Ořezávání** — aby nedocházelo k překrytí, tak se objekty rozdělí, ale tím pádem duplikují
    - Nestačí vyhledat objekt, ale je třeba se vrátit k jeho původní nedělené reprezentaci

- **R-Tree**
    - Metoda překrývání
    - Vždy jistou skupinu obalí MBB. Pokud je počet objektů větší než dané maximum, pokračuje rekurzivně v dělení.
    - MBB se mohou překrývat a objekty zasahovat do více buněk
    - Vyhledávání: Jakmile bod spadá do některého z obdélníků, vyhledává se jen v rámci něj (rekurzivně)
    - Charakterem má algoritmus stejné vlastnosti jako stromové algoritmy, navíc přibývá problém s více cestami a selektivitou.



- **R^+-Tree**
    - Metoda ořezávání
    - V případě, že existuje objekt, který zasahuje do více buněk, tak je nutné objekt rozdělit na hranici na dva (více) kusů. Pokud by hranice buněk nebyly těsně u sebe, tak se musí buďto vložit další buňka (je-li to žádoucí a přínosné), nebo se musí buňka rozšířit — zde je však nebezpečí zamrznutí (deadlock), neboť se může blokovat více buněk v rozšiřování.



Obojí data

- **P-Tree**
  - Ukládá bodové i vícerozměrné objekty
  - Pro vymezení buněk nepoužívá MBB, ale obecně konvexní obálky
  - Data ukládá do listů
  - Datová struktura se podobá té v Buddy Tree (až na hashování)
  - Největší problémem je, podobně jako u jiných algoritmů, zda rozšířit buňku pro nový objekt, nebo zda zavést buňku novou.