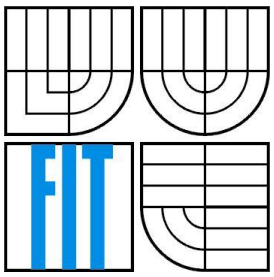
*VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ*

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# PDB

(TEMATICKÉ OKRUHY KE STÁTNICÍM 2009)

Vladimír Veselý – xvesel38

# OBSAH

# 1 DEFINICE POSTRELAČNÍHO DB SYSTÉMU

**Postrelační databázový systém** je DBS, který už nevystačí se základním relačním schématem a bez přímé podpory na implementační úrovni pro uvažovanou specializaci je zpracování „jiných“ (specializovaných, pro systém přirozeně neznámých) dat velmi neefektivní.

Postrelační systémy jsou takové, kde je manipulace s daty zabudována na aplikační úrovni.

Mezi postrelační DBS patří:

- prostorové (spatial);
- objektově orientované;
- časové (temporal);
- multimediální;
- aktivní.

V současné době většina DBS jsou postrelační, neb obsahují mnohá rozšíření vůči svému původnímu relačnímu modelu. Postrelační DBS lze tedy vyvíjet buď na „zelené louce“, nebo právě úpravou již existujících jazyků či DBS.

# 2 OBJEKTOVĚ RELAČNÍ DB SYSTÉMY: ROZDÍL OPROTI RELAČNÍM

**Objektové DB** umožňují modelování a vytváření perzistentních dat jako objektů. Objektově relační DBS nejsou plně objektové (nemodelují objekty a vztahy mezi nimi přímo), ale jsou v nejnižší vrstvě relační a objekty nad nimi jsou simulovány pomocí SŘBD. Čistě objektové systémy nebyly standardizovány, neměly standardní jazyk a nerozšířily se.

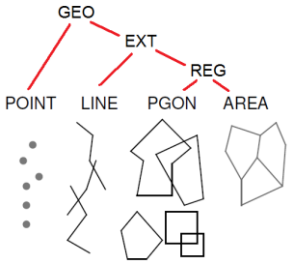
| | Relační DB | Objektová DB |
|---|---|---|
| **Model** | Relační obsahující kolekci tabulek; velmi omezená množina datových typů hodnot; vztahy mezi řádky tabulek vyjádřeny pomocí cizích a kandidátních klíčů, kde pro vztahy M:M nutná vazební tabulka. | Objektový (neexistuje standard ODMG-93 byl pokusem); obsahuje třídy, atributy, operace, jednoznačné OID pro každý perzistentní objekt; podpora ADT; zapouzdření a polymorfismus; atributy objektů mohou být jiné objekty→složité typy; vztahy vytvářeny pomocí reference OID, kde M:M lze vytvářet přímo. |
| **Dotazovací jazyk** | SQL (neprocedurální, deklarativní) | Snaha o vytvoření standardu (OQL), většinou v kombinaci s OO jazykem. |
| **Výpočetní model** | Založen na hodnotách ve sloupcích tabulek, žádné reference či ukazatele, jednoduchá navigace po tabulce pomocí kurzoru. | Významná role OID, pomocí kterého se navigujeme po objektové struktuře. |

Objektové DB tedy nenahrazují, jen doplňují relační model, a proto mluvíme o **objektově-relačním DBS**. U ORDBS je model obohacen o OO, jako dotazovací jazyk se ujal SQL-1999 a výpočetní model specifikuje navigaci jak pomocí kurzoru, tak pomocí referencí.

# 3 SQL-1999: DATOVÉ STRUKTURY A DATOVÉ TYPY, OPERACE NAD NIMI

## 3.1 DATOVÉ TYPY

K tomu, aby bylo možné vkládat prostorová data do databáze a aby bylo možné nad nimi provádět požadované operace, je nutné, aby systém podporoval tvorbu dat daného typu a dále potom nabízel operace, kde takové typy vstupují jako parametry, případně vystupují jako výsledek. Definice dat a operací by tak měla být nezávislá na konkrétním SŘBD, přitom s ním však úzce spolupracovat.




Mezi datové typy, které nám umožní sestavit téměř libovolný 2D objekt, patří pro SQL99 **bod (point)**, **lomená úsečka (line)**, **region** (který lze dělit na nevyplnění **polygon** a vyplněnou **plochu/area**).

Nové datové typy jako LOB, BLOB, CLOB, ARRAY a MULTISET, spolu s novými klíčovými slovy jako SIMILAR, hlavně však REF a OBJECT, které umožnily vytváření strukturovaných uživatelských datových typů. Dědičnost a polymorfismus pak umožňují (NOT) FINAL a (NOT) INSTANTIABLE.

## 3.2 OPERACE

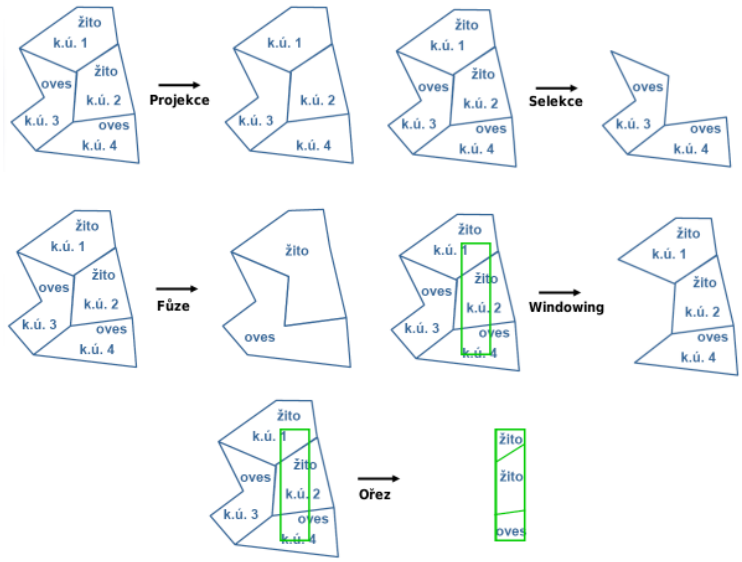
**Kategorie operací** nad uvedenými typy lze rozdělit na:

- **predikáty** – aspoň jedna vstupní hodnota je z oboru GEO, výstupem je pravdivostní hodnota;
- **geometrické relace** – vstupem i výstupem je množina hodnot z GEO;
- **výpočetně náročné operace** – oproti předchozí kategorii jsou výsledkem nově získané hodnoty různých i neGEO typů.

Mezi **základní operace** patří (ne)rovnost, sousednost, obsahování, průnik, sjednocení, překrytí, voronoi, konvexní obálka, vzdálenost.

Základní **relační operace nad prostorovými daty**:

- **projekce** – dochází k redukci atributů, které jsou zobrazovány, často však zachovává oddíly;
- **selekce** – redukuje oddíly, zachovává atributy;
- **fúze** – projekce, kdy oddíly se stejnými atributy jsou spojeny v jeden oddíl (je výpočetně náročná, nejen porovnání, ale i zjištění sousednosti);
- **windowing** – vrací kompletní oddíly, do kterých zasahuje inspekční okénko, je to vlastně průnik nad grafickým obsahem;
- **ořezání** – oproti windowingu provádí navíc i ořezání vybrané oblasti inspekčním okénkem.

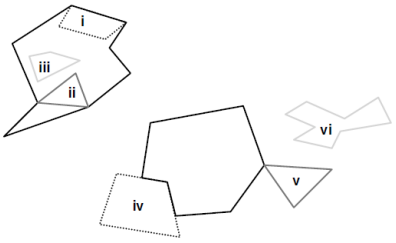



## 3.3 ALGEBRA ROSE

**Algebra ROSE** je algebrou navrženou pro podporu prostorových DBS vycházející částečně z relačních systémů. Využívá deskriptorů a složitější objekty získává jejich skládáním.

Definuje míru vnoření a disjunkce dvou plošných objektů:

- **plošné vnoření** (objekty „i“, „ii“, „iii“);
- **hranové vnoření** (objekty „ii“, „iii“);
- **vrcholové vnoření** (objekt „iii“);
- **plošná disjunkce** (objekty „iv“, „v“, „vi“);
- **hranová disjunkce** (objekty „v“, „vi“);
- **vrcholová disjunkce** (objekt „vi“);




Na základě takového vnímání vnoření a disjunkce, lze definovat **R-cyklus** jako 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.

Jelikož se v modelované realitě moc nevyskytují oddělené plochy, je třeba zadefinova i R-plochu, která by zohledňovala vnoření. **R-plocha** $f$ tedy je dvojice $(c, H)$ taková, že $c$ je R-cyklus a $H = \{h_1, \ldots, h_m\}$ je množina R-cyklů, kde platí:

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

Lze hovořit o **vnoření R-ploch** tehdy, když $f$=($f_0$, $F$) a $g$=($g_0$, $G$) jsou R-plochy, pak říkáme, že $f$ je plošně obsaženo v $g$ právě tehdy, když:

- $(f_0$ je plošně vnořena v $g_0)$
  $\wedge$
- $\forall g \in G: ((g$ je plošně disjunktní s $f_0) \vee (\exists f \in F: g$ je plošně vnořeno v $f))$

# 4 PROSTOROVÉ DATABÁZE: ZÁKLADNÍ PROBLEMATIKA ULOŽENÍ PROSTOROVÝCH ENTIT, PROBLÉMY INDEXACE BODŮ V PROSTORU

## 4.1 ÚVOD PROSTOROVÝCH DB

**Prostorové databázové systémy** jsou databázové systémy, jejichž DDL a DML zahrnují prostorové datové typy, které jsou podpořeny i na implementační úrovni, takže je možné efektivně provádět různé specializované operace (např. indexace, vyhledávání, spojování) – je to tedy typ SŘBD! Prostorové databáze jsou schopné spravovat data, která se váží k určitému prostoru, bez ohledu na to, jak veliký ten prostor je. V prostorových DB, tedy nalezneme množiny entit, u kterých je zřejmá:

- identifikace;
- umístění;
- vztah s okolím.

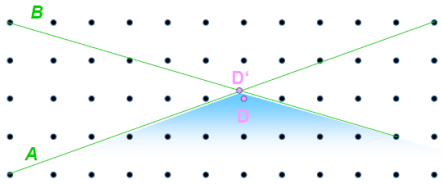
V zásadě tedy ukládáme **geometrické** (základní matematicko-geometrické tvary reprezentující např. města, řeky, domy, atomy, planety) a **topologické** údaje (vzájemné vztahy mezi ukládanými objekty, jako např. vzdálenost). Popisujeme tedy entity v prostoru a prostor jako takový.

V rámci geometrických údajů mluvíme o entitách jako bod, lomená čára, (vyplněný) polygon.

## 4.2 PROBLÉM REPREZENTACE SOUŘADNIC

Obvykle budeme uvažovat dvou nebo tří rozměrný **Euklidovský prostor**, který je spojitý ve svých dimenzích a pozice každého bodu je definován uspořádanou $n$-ticí čísel z množiny $\mathbb{R}$. Pro vzdálenost dvou bodů, platí **Euklidovská metrika**: $d(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$

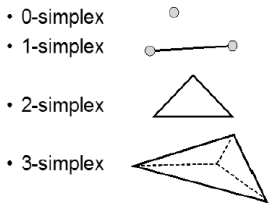
Problémem, který však musí být u prostorových DBS spolehlivě řešen, je nemožnost počítače s nekonečnou přesností reprezentovat reálná čísla – dochází k diskretizaci prostoru. Tento problém vyvstává např. při výpočtu průsečíku, či sousednosti a je ilustrován obrázkem:




Řešením je oddělení typů a operací nad prostorovými daty, tak aby byl korektně ošetřen tento číselný problém vzhledem ke geometrickému systému. Lze použít třeba jednu z těchto metod:

- **simplexy** – použití jednoduchých geometrických entit ke skládání složitějších celků;
- **úplné deskriptory/realms** – kompletní popis modelované oblasti.

**Simplexy** jsou nejmenší nevyplněné objekty dané dimenze: 0-simplex (bod), 1-simplex (úsečka), 2-simplex (trojúhelník), 3-simplex (čtyřstěn),… Lze vypozorovat, že *d*-simplex se sestává z (*d*-1)-simplexů o počtu *d*+1 – takové elementy složené ze simplexů nižšího řádu se nazývají (pohlavní) **styky (faces)**. Kombinace simplexů do složitějších struktur je povolena jen tehdy, pokud průnik libovolných dvou simplexů je styk.

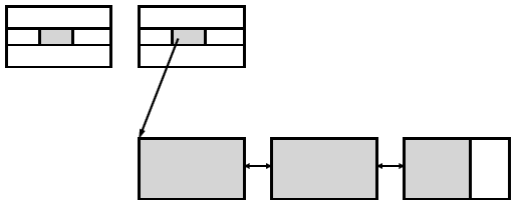



**Úplné popisy (deskriptory** nebo **realms)** jsou vlastně jakýmsi souhrnným popisem všech objektů v DB. Formálněji jsou to množiny bodů, úseček, či vyšších celků, které mají tyto vlastnosti:

- 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.

## 4.3 ULOŽENÍ PROSTOROVÝCH DAT

Prostorový DBS obsahuje prostorové datové typy stejné jako hodnoty jiných typů. Pro SŘBD by bylo ideální, aby nové typy mohl zpracovávat jako stávající, což však není možné. Prostorový DBS si musí poradit i s různorodou (hodně velkou) velikost dat, kde datové položky se mohou pro hodnoty jednoho typu významně lišit a nabývat vysokých hodnot.



Pro prostorová data se volí takový způsob uložení, aby byl konzistentní pro různou velikost dat. **Pro každá prostorová data tak dojde k vyčlenění dat, které se nemění s charakterem vkládaného objektu**. Ostatní data se ukládají mimo, do zvláštních datových stránek, které tak leží mimo a nebrání rychlému zpracování. Pokud prostorová data nepřekročí určitou velikost, tak jsou uložena stejně, jako ostatní. V rámci jedné stránky na disku/v paměti je možné mít více datových záznamů, jakmile však délka přeroste jistou mez, tak je uložen odkaz na souvislou oblast na disku, kde jsou velká data uložena za sebou.

V některých případech se volí i jisté předzpracování prostorových dat – ukládají se např. výsledky agregačních funkcí (průměr, suma, střed) nebo prvotní aproximace.

# 5 ALGORITMY STROMOVÉ/HAŠOVACÍ/KOMBINOVANÉ, ZPŮSOB INDEXACE VÍCEROZMĚRNÝCH OBJEKTŮ, ALGORITMY R A R+ STROM

## 5.1 ÚVOD K INDEXOVÁNÍ V PROSTOROVÝCH DBS

Podobně jako u jiných datových typů, i v případě prostorových datových typů je **tvorba indexu** spojena s podporou, zvýšením efektivity, u takových operací jako je prostorová selekce, prostorové spojení (join) a i další operace.

Způsoby jak indexovat jsou dva:

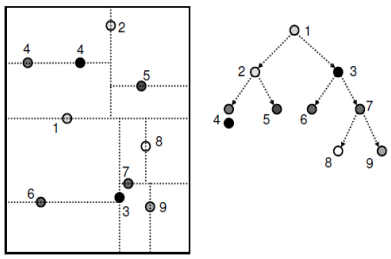
- mapování vícerozměrných údajů do jednoho rozměru a užití známých metod pro tvorbu indexů, což bývá rychlejší a jednodušší;
- vystavění specializovaných indexů pro prostorové datové struktury, což vyžaduje složitější implementaci a přístup k robustnosti.

Indexy jsou používány v klasických databázových operacích, jako vkládání (insert), mazání (delete), dotazování (select).

## 5.2 STROMOVÉ ALGORITMY

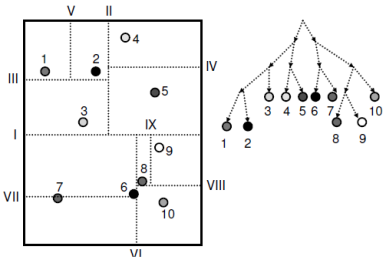
### K-D-TREE

Tento algoritmus je založen na principu dělení prostoru hyperplochami (v rovině to jsou přímky) na nejvyšší možné úrovni, a to vždy na dvě části. Dělící hyperplocha musí být rovnoběžná s osovým (souřadným) systémem. Při dělení, musí ve výsledných plochách být obsažen vždy alespoň jeden bod, který pak už není součástí jakékoli jiné hyperplochy. Vkládání a vyhledávání je bez problému, avšak při mazání je potřeba znovuustavit celý strom, algoritmus je tak výhodný pro statická data.



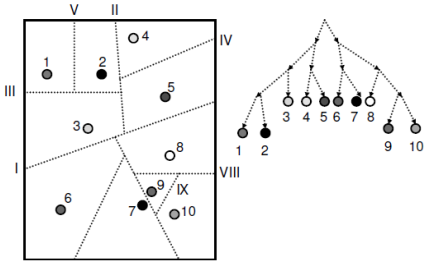
### ADAPTIVNÍ K-D-TREE

Tento algoritmus se snaží eliminovat nevýhodu K-D-Tree v pořadí vkládání hodnot do stromu. Data jsou pak roztroušena po celém stromu. Při dělení hyperplochou se vždy daný podprostor rozdělí tak, aby každá část obsahovala zhruba stejný počet bodů. I když jsou dělicí hyperplochy rovnoběžné s osovým systémem, tak nemusejí obsahovat žádný bod. Data se tak stěhují do listů je stanovena hranice, kolik bodů, i tento algoritmus je však vhodný spíše pro statická data.



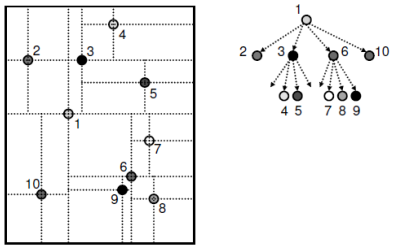
### BSP Tree (binary space partitioning)

Shodný s adaptivním K-D-Tree až na to, že dělící hyperplochy teď nemusí být rovnoběžné se souřadným systémem. Dělí se taktéž tak dlouho, dokud počet bodů v hyperploše neklesne pod určitou úroveň. Má vyšší nároky na paměť, než K-D-Tree (u kterého šlo díky pravidelnému střídání dělení ignorovat jednu souřadnici hyperplochy).




### Quad-Tree

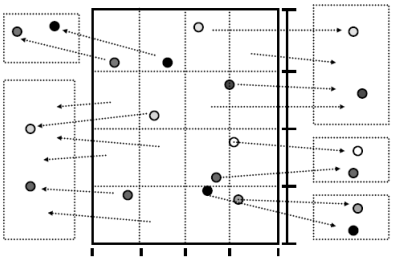
Shoduje se s neadaptivním K-D-Tree, avšak prostor nedělí na poloviny, ale čtvrtiny. Díky tomu nejsou podstromy shodné (některé dokonce neobsahují žádný bod). Dělení opět probíhá tak dlouho, dokud počet bodů neklesne na určitý počet. Algoritmus existuje ve variantě s body, ale i s dělením na regiony shodné velikosti (viz PGR).




## 5.3 Hashovací algoritmy

### Grid file

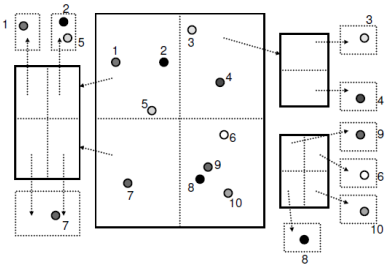
Sledovaný úsek prostoru je pokryt *n*-rozměrnou mřížkou (nikoliv nutně pravidelnou). Výsledné buňky tak obsahují různý počet bodů – různorodé obsazení. K tomuto základnímu rozdělení je dodán adresář, který každou buňku přiřazuje k datové jednotce (bucket). Adresář je poměrně velký a spolu s mřížkou je proto vždy ukládán na disku.



Při vkládání může dojít k přetečení (datová jednotka není schopna pojmout další údaj), které není lokální. Je nutné vložit rozdělující hyperplochu a tak zvětšit adresář. Podobně se chová i mazání – jelikož není lokální, tak odstranění hyperplochy je třeba prověřit. Pokud je dat málo, tak může zaniknout i datová jednotka (data buďto zanikla s ní, nebo jsou přesunuta do jiné datové jednotky).

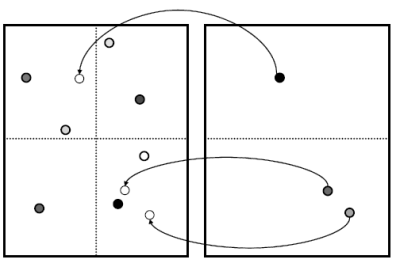
### Two-level grid file

Je dvouúrovňový gridfile, který zavádí mřížku druhé úrovně. V takovém systému jsou změny při vkládání či mazání často lokální, nicméně i tak není problematika přetečení úplně vyřešena. První úroveň tedy slouží pouze jako ukazatel do základní struktury Grid fileu.



### Twin Grid file

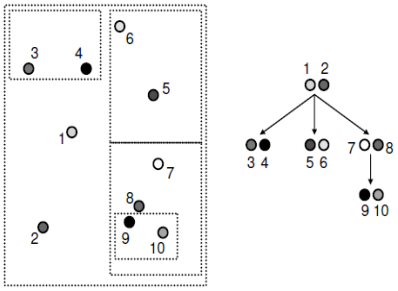
Celá struktura se vyskytuje dvakrát. Vztah však není hierarchický (vertikální), ale horizontální. I když je jedna ze struktur nadřazená, tak je jedno, která to bude. Cílem tohoto algoritmu je maximálně využít prostor pro indexovací strukturu. Proto se data mezi obě části dělí prakticky rovnoměrně. Jedna struktura je však primární a druhá je sekundární, přetoková. Algoritmus má úplnou paralelizaci vyhledávání a částečnou pro vkládání.



## 5.4 HYBRIDNÍ ALGORITMY

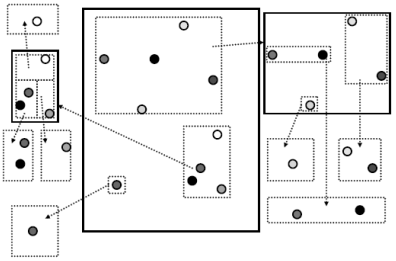
### BANG FILE

Tento algoritmus se snažil odstranit problém s exponenciálním nárůstem adresáře – typická data nejsou rozložena v prostoru rovnoměrně. Základem je opět Grid File. Nicméně buňky se mohou překrývat, dokonce vnořovat. Datová jednotka odpovídá buňce. Buňky ani nemusejí mít tvar hyperkrychle. Jednotlivé datové jednotky jsou potom uloženy ve vyváženém stromě. V každé buňce může být jistý maximální a minimální (nenulový) počet bodů. Při vyšším počtu bodů je nutné vložit nové podprostory a vybudovat vyváženou stromovou strukturu.




### BUDDY TREE

I když je základem hashování, tak adresářem je obecně nevyvážená stromová struktura minimálně se dvěma položkami. Uvnitř stromu jsou ukazatele na nižší úrovně, až v listech jsou ukazatele na datové jednotky. Strom je rozdělován rekurzivně, dělí se hyperplochami rovnoběžnými s osami. Ve vnitřních uzlech se však prostor omezí na **nejmenší obalující obdélník** (minimal bounding box MBB) vnitřních bodů – tím se dosahuje výraznější selektivity.



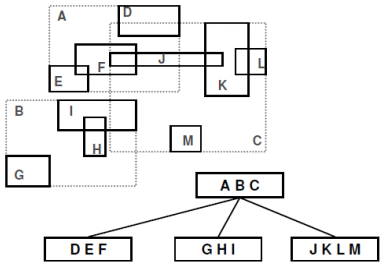
## 5.5 VÍCEROZMĚRNÁ DATA

Uložení bodů je sice primární, ale v obecné realitě s ním nelze vystačit, proto vznikly algoritmy pro indexaci vícerozměrných útvarů, které používají jednu z následujících metod:

- **transformace** – Objekty popsané *k* body v *n*-rozměrném prostoru se mapují na bezrozměrné objekty (body) ve vícedimenzionálním prostoru, přesněji v *k*×*n*-rozměrném prostoru (obdélník ve 2D se tak stává bodem ve 4D);
- **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;
- **ořezávání** - U metod založených na ořezávání není povoleno, aby se MBB, buňky překrývaly. Jediným řešením je rozsekat objekty tak, aby sledovaly hranice dělicích MBB. Jeden objekt tak může být rozdělen na více. Při vyhledávání pak tedy nestačí vyhledat objekt, ale je třeba se vrátit k jeho původní, nedělené reprezentaci.

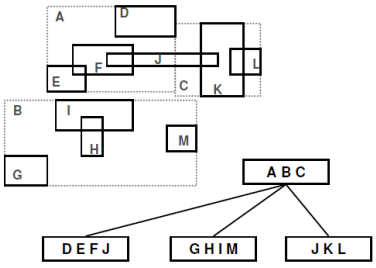
### R-TREE

Je zástupce algoritmů překrývání. Ukládá obdélníkové objekty tak, že vždy jistou skupinu obalí MBB a pokud je počet objektů uvnitř nad jistou mez, tak pokračuje rekurzivně v dělení. Obalující MBB vždy obalují celý objekt, který ukládají. Díky tomu se mohou překrývat a indexované objekty mohou zasahovat do více buněk.



**R+-TREE**

Tento algoritmus je zástupce stromových algoritmů pracující metodou ořezávání. Algoritmus je tedy podobný algoritmu R-Tree s tím, že buňky se nemohou překrývat. 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.




Hašišování je používáno i pro vícerozměrná data – PLOP, Multi-Layer Grid File, R-File. Obojí data (body a vícerozměrná) lze také indexovat, např. P-Tree (princip konvexních obálek).

# 6 TEMPORÁLNÍ DATABÁZE: SNÍMKOVÝ MODEL, MODEL ČASU PLATNOSTI, PROBLÉM INDEXACE, SHLUKOVÁNÍ

**Temporální DB** jsou DB s časovou dimenzí, které nacházejí své uplatnění v řadě odvětví (např. finanční, katastrální, medicínský). Čas můžeme vnímat z několika hledisek, např. fyzikálního (minulost je omezena Velkým Třeskem) nebo formálního (ne-/lineární větvení času).

Logicky již chápeme významy následujících pojmů, že **časový okamžik** je bod na časové ose a **časový úsek** je doba mezi dvěma časovými okamžiky a **časové trvání** je časový úsek se známou dobou, ale neznámým začátkem a koncem. Nejmenší jednotkou měření hodin jsou chronony, přičemž hodiny jako takové dodávají sémantický význam časovým okamžikům (časová razítka).

**Čas platnosti** říká, ve kterém období modelované reality je daný fakt pravdivý. **Čas transakce** faktu, je čas, kdy je fakt přítomen v DB a může být získán.

**Snímkový model DB** je takový, ve kterém je každý stav DB opatřen časovým razítkem, obsahuje snímkovou tabulku (SNAPSHOT). Temporální DB tedy vlastně používá funkcí mapující čas na stav databáze $T \rightarrow DB(D, \rho)$. Snapshot model není příliš vhodný na dotazy typu: *„Zobraz všechny okamžiky, kdy byla nějaká podmínka v rámci DB platná?“*

Při změně údajů v relační databázi se její stav mění na jiný. **Historii** pak chápeme jako posloupnost stavů (temporální) DB.

**Model času platnosti** je založen tabulce s časy platnosti. Takový model umožňuje nejen dotazy nad historií, ale řeší i problém snímkových SNAPSHOT modelu, čas je zde „jedním z datových typů“. Temporální DB s časovými razítky lze formálně definovat jako $TDB = (D, =, T, <, r_1, \dots, r_n)$, kde:

- $D$ jsou data;
- $T$ a $<$ tvoří časovou doménu takové databáze (relace uspořádání v časové ose);
- $r_1, \dots, r_n$ je konečná instance všech relací z ρ nad daty a časem.

**Temporální DB s obojím časem** rozšiřují model času platnosti ještě o atribut času transakce.

Formálním dotazovacím jazykem v rámci temporální logiky je rozšířený relační kalkul o specializované spojky jako (until, since, true until, true since). Pak **temporálně relační kalkul** *M* **logiku prvního řádu** lze definovat jako:

$$M \text{ tvoří } \underbrace{r_i(t_i, x_{i1}, \dots, x_{in})}_{\text{DB schéma}} \Big| \underbrace{M \wedge M, \neg M, \dots}_{\text{logické spojky}} \Big| \underbrace{x_i = x_j}_{\text{data}} \Big| \underbrace{\exists x_i . M}_{\text{proměnné}} \Big| \underbrace{t_i = t_j}_{\substack{\text{temporální} \\ \text{data}}} \Big| \underbrace{\exists t_i . M}_{\substack{\text{temporální} \\ \text{proměnné}}} \Big| \underbrace{\Diamond M, \bullet M, \dots}_{\substack{\text{temporální} \\ \text{spojky}}}$$

Jednorozměrná temporální relace obsahuje **shluky**, pokud je každý fakt spojován s nejvýše konečným počtem nepřekrývajících se intervalů. Někdy je potřeba spojit intervaly kvůli dalšímu zpracování, intervaly, které se překrývají nebo na sebe navazují lze takto spojit.

Mezi jazyky temporálních DB patří TQUEL, TSQL2 či ATSQL. Jazyk TSQL2 pracuje s lineárním časovým modelem, který má diskrétní časovou osu (nejmenší jednotkou je 1 chronon = tik hodin), lze u něj nastavit různou granularitu času a podporuje temporální systémy jak typu snapshot, tak s časem platnosti, či obojího času. Obsahuje standardní datové typy jako DATE, TIME, DATETIME, ale i specializovanější INTERVAL a PERIOD.

**Indexování v temporálních DB** souvisí s problémem, kdy máme nějaký interval a máme vybrat všechny *n*-tice, které mají s tímto intervalem neprázdný průnik, výsledkem bývá nevyvážená struktura pro přístup k datům. Mezi používané metody indexování patří Time index, nebo již známé R-Tree či AP-Tree.

# 7 DEDUKTIVNÍ DATABÁZE: DEFINICE, TYPY PREDIKÁTŮ, SYSTÉMY BEZ NEGACE A BEZ REKURZE, PROBLÉM REKURZE, BEZPEČNÉ PREDIKÁTY

**Deduktivní DB** obsahují fakta odrážející skutečnost a odvozovací pravidla pro práci s touto skutečností, snaží se pak s fakty pracovat a odvozovat z nich nové skutečnosti. Data v deduktivním DBS tak můžeme rozdělit na fakta (**explicitně uložené predikáty**), odvozovací pravidla a odvozená fakta (**odvoditelně uložené predikáty**).

**Jazyk deduktivních DBS** tvoří množina správně definovaných formulí (*wffs*). Predikátová logika **First Order Logic FOL** pracuje na principu *wffs* + modus ponens + zobecnění. Dedukci pak můžeme provádět na základě:

- **sémantiky** (platnosti formule) – na základě vlastnosti nesplnitelnosti, kde pravidla definují možné modely, které jsou pak predikátově ohodnoceny pravdou či nepravdou;
- **syntaktiky** (platné konstrukci formule) – uplatňuje inferencí pravidla na axiomy všemi možnými způsoby („shora dolů“ od teorému k axiomům, „zdola nahoru“ od hypotézy k teorému).

Automatizaci důkazu poprvé představil programovací jazyk PROLOG (Hornovy klauzule, SLD rezoluce). Relační model nad logikou v deduktivních DBS je postaven na relacích (formule z FOL) a funkcích, jakožto speciálním případu relaci.

**(Herbrandovým) Modelem** množiny pravidel je interpretace, ve které jsou všechna pravidla pravdivá – nezávisí na přiřazení hodnot proměnným. Nechť $M_1$, …, $M_n$ jsou modely množiny $S$ *wffs*, **minimální model** je takový model $M_k$, pro který platí:

$$M_k \subseteq M_j, j \in \{1, \dots, k-1, k, k+1, \dots, n\}: \nexists M, M \text{ je modelem } S \wedge M \subset M_k$$

Systémy **bez negace/s negací** se liší v typu výsledku, bez negace získáme jediný minimální model (právě kvůli neexistenci ekvivalentních formulí) a množinu faktů odvoditelných z DB, s negacemi pak jeden z mnoha.

**Atomická formule** je vlastně predikát v deduktivním DBS, který se skládá z predikátového symbolu a uspořádané $n$-tice proměnných a konstant: $p(A_1, \dots, A_n)$. Predikáty tak můžeme dělit na explicitní (přímo vložená fakta) nebo implicitní (odvozená fakta).

Pravidlo (predikát na levé i pravé straně) je **bezpečné,** pokud jsou všechny jeho proměnné **omezované**. Každá proměnná je omezovaná pokud:

- je na pravé straně pravidla;
- je porovnávána na rovnost s konstantou;
- je porovnávána na rovnost s již omezenou proměnnou.

**Rekurzivní** pravidla obsahují stejné proměnné na pravé i levé straně, vyhodnocování je tak mnohem složitější (odpovídá rekurzi v klasickém programování). Problémem rekurze je, že není možné uspořádat predikáty (podcíle) v pravidle.

Jazyk DATALOG je relační jazyk bez negace s rekurzí, kde techniky v něm použité se často používají při optimalizaci SQL dotazů. Mezi implementace deduktivních DBS pak patří např. PROSQL, NAIL!, LDL, MegaLog.

# 8 MULTIMEDIÁLNÍ DATABÁZE: ZÁKLADNÍ CHARAKTERISTIKA, VYHLEDÁVÁNÍ V OBRAZOVÝCH DATABÁZÍCH, INDEXOVÁNÍ PRO PODOBNOSTNÍ VYHLEDÁVÁNÍ

## 8.1 ÚVOD

Současné typy médií jsou text, obraz/fotografie, zvukový záznam, film – od ostatních se audio a video liší svou časovou rovinou. **Multimediálními daty** zpravidla rozumíme nestrukturovaná data, která podle způsobu vnímaní dělíme na vizuální a audiodata.

**Multimediální DB** vznikla s potřeby zajišťovat fyzické uložení, dotazování, indexování, extrakci obsahu a jednotný způsob prezentace multimediálních dat.

## 8.2 VYHLEDÁVÁNÍ V OBRAZOVÝCH DB

Vyhledávání v těchto DB se výrazně liší od vyhledávání nad klasickými (obvykle textovými/číselnými daty).

Rozlišujeme dva základní **typy obrazového hledaní** na základě:

- **textového popisku** – Předpokladem je existující slovní popis obsahu obrazu, který má k obrázku jistý sémantický vztah. Nevýhodou je obvykle nutnost manuálního vkládání takovýchto popisků, které mohou být navíc subjektivně zkresleny osobou vkládajícího;
- **podobnosti** – Dotaz zde má typicky podobu vzorového obrázku nebo náčrtku a z DB je vybráno několik obrázků, které jsou mu nejvíce podobné. Výhodou je, že hodnoty vlastností, jež se porovnávají jsou přímo odvozeny ze vstupního obrázku.

## 8.3 PODOBNOSTNÍ VYHLEDÁVÁNÍ

Rozlišuje dva základní přístupy:

- **metrický** – porovnání dvou obrazů se provádí vyhodnocení podobnostní funkce;
- **transformační** – podobnost dvou obrazů je vyjádřena cenou transformace prvního obrazu na druhý a naopak.

Mezi problémy podobnostního vyhledávání patří tři otázky:

- *„Které vlastnosti obrazu vybrat pro jeho reprezentaci?“*
- *„Jak vyjádřit podobnost dvou obrazů?“*
- *„Jak efektivně indexovat?“*

**Podobnost** můžeme chápat jako matematickou vzdálenostní funkci $d: \Pi \times \Pi \rightarrow \langle 0,1 \rangle$, kde $\Pi$ je množina reprezentací obrazů (typicky nějakým vektorem rysů). Pro libovolné tři reprezentace obrazů $I, J, K \in \Pi$ musí **vzdálenostní funkce *d*** splňovat následující omezení:

- symetrie – $d(I,J) = d(J,I)$;
- nezápornost – $d(I,J) \geq 0$ pro $I \neq J$ a $d(I,I) = 0$;
- trojúhelníková nerovnost – $d(I,K) + d(K,J) \geq d(I,J)$.

**Vizuální rysy obrazu** lze chápat jako jedinečnou charakteristiku obrazu mějící základ ve vizuální stránce. Tyto **rysy** lze pak dělit **podle úrovně popisu** na:

- **nízké**:
  - obecné – barva (kumulativní barevný histogram), textura, tvar (detekce hran);
  - doménově specifické – otisk prstu či držtičky;
- **střední** (statistické, sumarizační hodnoty);
- **vysoké** (sémantické vyjádření obsahu).

## 8.4 INDEXOVÁNÍ OBSAHU

Používají se všechny algoritmy známé z Kapitoly 5 pro vícerozměrná data: K-D-Tree, Point Quad-Tree, R-Tree, $R^+$-Tree, $R^*$-Tree, M-tree.