



EUROPEAN UNION
European Regional
Development Fund



Pokročilé databázové systémy

Doc. Dr. Ing. Dušan Kolář

kolar@fit.vutbr.cz

+420-54-114 1238

Konzultace: po dohodě e-mailem



Bodový zisk

• Půlsestestránní zkouška	20
Projekt	20
• <i>Zápočet</i>	<i>20</i>
Závěrečná zkouška	60(25)
• <i>CELKEM</i>	100



Důležitá data

- **Půlsestřední zkouška**
 - 7. týden výuky (31. 10. 2016)
- **Odevzdání projektu (předvedení)**
 - 12. 12. 2016 23:59 (pondělí)
 - 15./16. 12. 2016 cvičení (úterý)
- **Závěrečná zkouška**
 - termín A: bude přidělen
 - termín B: bude přidělen
 - termín C: bude přidělen



Projekt – zadání

- <https://www.fit.vutbr.cz/study/courses/PDB/private/Projekty/projekt2016.pdf>
- **Oracle**
- **MM, Spatial, Temporal**
- **Team**
- **Dr. Rychlý**



Projekt – přihlášení

- **Po sestavení tříčlenného týmu vedoucí odešle email s loginy všech členů**
 - instrukce v zadání
 - E-mail předmět:
 - PDB - registrace týmu
 - datový sklad!
- **Nejpozději 7. října 2016 včetně**



Projekt – průběh, obhajoba

- **23. 10. – plán**
- **6. 11. – schéma**
- **12. 12. – final**
- **15. /16. prosince v N204+N205**
 - **ve stejném čase, jako si studenti zvolí u počítačových cvičení.**
- **Dr. Rychlý**



Projekt – uznání

- *Pro uznání projektů z loňska je nutné do 7.října 2016 poslat žádost a taktéž získat za projekt vypracovaný v loňském roce alespoň 16 bodů.*
- *Kolář*



Projekt – individuální

- **Osobní konzultace**
- **Max. 4**



Demonstrační cvičení

- **Středy – *Viz WIS či stránku PDB***
 - 21. září (úvod a multimediální databáze)
 - 5. října (prostorové a XML databáze)
 - 12. října
(temporální a deduktivní databáze).
- **Tato cvičení jsou pro všechny
(nepřihlašuje se) v D0206**
- **8:00-9:50**
- **Dr. Rychlý**



Počítačová cvičení

- **N204+N203**

- vždy čtvrtek/pátek po demonstračních cvičeních – **viz WIS!**
 - 29./30. září (úvod a multimediální databáze)
 - 6./7. října (prostorové a XML databáze)
 - 13./14. října (temporální a deduktivní databáze)
- Počítačová cvičení jsou od 10:00 do 14:00 (čt) a od 10:00 do 16:00 (pá) a trvají 2 hodiny.
- Přihlašuje se **individuálně** na čas.



Přihlašování na cvičení

- **Od 20:00, dnes, po přednášce...**



Zdroj informací

- **Přednáška!**
- **Informace sdělené na přednášce se neduplikují do e-mailu, ani nikam jinam!**
- **Termíny – viz WIS + privátní stránky kurzu!**
 - (pokud to lze zařídit, často WIS v předstihu)



Úvaha nad studijním zapojením

- **Studium na VŠ**
 - **Práce na plný úvazek**
 - 8h (pro optimisty, spíše 10) => 40h týdně
- **30 kreditů za semestr**
 - 6 předmětů po 5kreditech
 - 6 x 4hodiny (50 minut) = $24 \times 50 = 1200\text{m}$
 - Tedy, 20h, což je polovina



Model

- **Oficiálně**
 - 20h na FIT, a tedy
 - min. 20h doma
- **Doporučeno**
 - Aktivně se účastnit výuky
 - Opakovat si nejasné věci
 - *Nespoléhat* na záznamy (zejména z dřívějších let)
 - Změny v projektu!! (průběžná kontrola)



Zamyšlení na úvod

- **Proč jste tady?**
 - Podívat se, kdo to bude fakticky přednášet...
 - Podívat se, jestli to bude stejné, jako loni...
 - **Chci se dozvědět/konfrontovat věci o pokročilých DB systémech...**
- **Proč vlastně studuji?**
 - Potřebuji titul Ing... Opravdu?
 - Chtějí to rodiče...
 - Chtějí to v práci...
 - **Chci se něco dozvědět, ujasnit si, ...**



Zpětná vazba – anketa

konec semestru

- Předmět mám jako P, PV, V
- Chodil/a jsem na přednášky skoro po celý semestr a dával jsem bedlivý pozor - ano/ne
 - Pokud ano - děláte si poznámky, nebo jen poslech?
 - Pokud ne - z čeho jste se učil/a na zkoušku?
- Čím bych zatraktivnil/a přednášky (ne odpovědi typu legrační oblečení, čokoláda, body, jiný obsah – tento je akreditovaný, ale drobné modifikace jdou).
- Kdyby byly záznamy dostupné – přestal/a bych chodit na přednášku?
- Na projekt je třeba mít aspoň nějaké znalosti z přednášek a z democvik a labin - termín odevzdání projektu byl OK, dal byl se zkrátit, měl by se prodloužit a to i za cenu, že potom nebudu mít zápočet a přitom třeba závěrečnou zkoušku napíši na hodně bodů.
- Jméno
- Příjmení
- Ostatní – v čem je předmět náročný? (časově – proč?, jinak ...)



Literatura

- Gaede, V., Günther, O.: *Multidimensional Access Methods*, Institut für Wirtschaftsinformatik, Humboldt-Universität zu Berlin, Germany
- Kim, W. (edt.): *Modern Database Systems*, ACM Press, 1995, ISBN 0-201-59098-0
- Güting, R.H.: *Spatial Database Systems*, Praktische Informatik IV, Fernuniversität Hagen, Germany
- Zendulka, J.: *Databázové systémy a návrh databází*, přednášky k předmětu
- Chomicki, J., Toman, D.: *A Tutorial on Temporal Databases*, Monmouth & Toronto University



Literatura (pokračování)

- Zurek, T.F.: *Optimisation of Partitioned Temporal Joins*, Ph.D. Thesis, University of Edinburgh, 1997
- Bertino, E., Ooi, B.C., Sacks-Davis, R., Tan, K.T., Zobel, J., Shidlovsky, B., Catania, B.: *Indexing Techniques for Advanced Database Systems*, Kluwer Academic Publishers, 1997, ISBN 0-7923-9985-4
- Bertino, E., Catania, B., Zarri, G.P.: *Intelligent Database Systems*, Addison-Wesley, ACM press, 2001, ISBN 0-201-87736-8
- Jensen C.S., Snodgrass, R.T.: *Temporal Databases, Part 2*, Aalborg University, University of Arizona, 1995-97 (Advanced DB Systems, Morgan Kaufmann)



Přehled přednášek

- 1) **Úvod**
- 2) **OR + multimediální DB (1.10.)**
- 3) **Prostorové DB**
 - 1) *úvod*
 - 2) *modely*
 - 3) *formalismy*
 - 4) *algoritmy (implementace)*
- 4) **XML DB (12.11.)**
- 5) **Temporální DB**
 - 1) *úvod*
 - 2) *časové modely, problematika*
 - 3) *TSQL2*
- 6) **Deduktivní DB**
 - 1) *úvod*
 - 2) *řešení DDB*
 - 3) *DATALOG*
 - 4) *OO a DDB*



1. Úvod



Co jsou postrelační databáze?

- **Obecná definice (pracovní)**
 - *Databázový systém, který už nevystačí se základním relačním schématem a bez přímé podpory na implementační úrovni pro uvažovanou specializaci je zpracování „jiných“ dat velmi neefektivní.*



Co NEjsou postrelační databáze?

- **Obecně**

- Mezi postrelační databáze nelze řadit systémy, které sice z uživatelského hlediska umožňují zpracovávat specializované údaje, ale podpora je pouze na aplikační úrovni.



Které jsou postrelační databáze?

- Objektově orientované - INS
- **Deduktivní (deductive)**
- **Prostorové (spatial)**
- **Časové (temporal)**
- Multimediální
- Aktivní
- *a mnoho dalších ...*



Které další jsou postrelační?

- **Kombinace předchozích**
 - prostorové užívající objektové schéma
- **Všechny, co přijdou**
 - podpora specializace
 - často nevhodné pro *obecné* použití
- **Často i *moderní* databázové systémy založené na relačním schématu**



Kde najdeme postrelační DB?

- **Prostorové**
 - GIS, CAD, 3D modelovací nástroje, VLSI, chemie, molekulární biologie, ...
- **Časové**
 - bankovníctví, pojišťovnictví, účetnictví, medicína, územní správa, ...
- **Deduktivní**
 - burza, molekulární biologie (DNA), ...



Co je nutné pro studium PDB?

- **Chtít**
- **Umět se samostatně vzdělávat**
- **Nebát se**
- **Komunikovat**
- *... a úspěšně složit zkoušku*



Co je třeba pro studium PDB?

- **Dobrá znalost relačního schématu**
- **Dostatečné znalosti v příbuzných oborech**
 - lineární algebra
 - analytická geometrie
 - temporální logika
 - predikátová logika
 - ...



Co je vhodné pro studium PDB?

- **Orientace v OOP**
- **Znalost angličtiny**
- **Rychlá práce a orientace na WWW**
- **Mít vlastní stroj „na hraní“**
- **Umět za-úkolovat svého pedagoga**
- **Účastnit se přednášek**
- **Využít volnosti na cvičeních ve vlastní prospěch**



Jak postupovat při studiu?

- ***Sledovat*** o čem jde řeč
- **Když je něco nejasné**
 - nejprve potrápit pedagoga
 - neví-li/není přítomen, potom se poptat kolegů
 - když i to selže, zbývá už jen WWW
 - ***seznámit ostatní s výsledkem!***



6 důvodů pro studium PDBS

- **Důvod 1.**

- **PDBS jsou více *vidět***

- *Route 66 (99, 2000) – kdysi*
 - *Navigace různých forem a značek*
 - *Zákony*
 - *Burza*

- více tajemství, než by se komu zdálo



6 důvodů pro studium PDBS

- **Důvod 2.**
 - **Podpora mezioborového studia**
 - geometrie
 - počítačová grafika
 - funkcionální & logické programování
 - logika (predikátová, temporální)
 - algoritmy



6 důvodů pro studium PDBS

- **Důvod 3.**

- **Znalosti z PDBS jsou velmi inspirující i pro jiné oblasti - *rozšiřuje obzory***

- vyhledávací algoritmy
 - způsoby ukládání
 - propojení *teorie-algoritmus*



6 důvodů pro studium PDBS

- **Důvod 4.**
 - Řešení v PDBS na *přístupných* datech umožňují později přechod k vyšší úrovni abstrakce.
 - 2D prostor až n-rozměrný prostor
 - časové paralelismy
 - úlohy z dolování dat



6 důvodů pro studium PDBS

- **Důvod 5.**
 - **Demonstrace technologií**
 - vznik *jednobitových* znalostí



6 důvodů pro studium PDBS

- **Důvod 6.**
 - **Nové DB systémy, které i vy budete možná vytvářet, budou, jak doufám, vždy odpovídat datům, která budou obhospodařována.**
 - méně zpracování v aplikaci
 - více úkonů v SŘBD



SQL - stručný přehled

- **Vytvoření tabulky**

- CREATE_TABLE jm_bázové_tabulky
(def_sloupce, ...
[definice_integritních_omezení])

- **Definice sloupce**

- jméno typ [implicitní_hodnota] [seznam_i_o]

- **Integritní omezení**

- PRIMARY KEY (jm_sloupce, ...)
UNIQUE (jm_sloupce, ...)
FOREIGN KEY ... REFERENCES ...



SQL - stručný přehled

- **Změna báze tabulky**

- ALTER TABLE jm_tabulky akce

- **Akce**

- **Sloupce**

- ADD
 - ALTER
 - DROP

- **Integritní omezení tabulky**

- ADD
 - DROP



SQL - stručný přehled

- **Zrušení tabulky**

- DROP TABLE jm_tabulky

- **Indexy a synonyma**

- **nejsou součástí SQL!**

- CREATE UNIQUE INDEX jméno ON jm_tabulky (jm_sloupce (ASC|DESC), ...)
 - CREATE SYNONYM jméno FOR jm_tabulky



SQL - stručný přehled

- **Manipulace s daty - SELECT**
 - SELECT [ALL|DISTINCT] položka [[AS] nové_jméno], ...
FROM tabulka [[AS] nové_jméno], ...
[WHERE podmínka]
[GROUP BY jm_sloupce|číslo, ...]
[HAVING podmínka]
[ORDER BY jm_sloupce|číslo], ...



SQL - stručný přehled

- **Manipulace s daty - INSERT**

- INSERT INTO jm_tabulky [(jm_sloupce, ...)]
zdroj

- **Zdroje**

- DEFAULT_VALUES
 - VALUES (skalární_výraz|NULL|DEFAULT, ...)
 - tabulkový výraz



SQL - stručný přehled

- **Manipulace s daty - DELETE**
 - DELETE FROM jm_tabulky [nové_jméno]
[WHERE podmínka]
- **Manipulace s daty - UPDATE**
 - UPDATE tabulka
SET jm_sloupce = výraz|NULL|DEFAULT, ...
[WHERE podmínka]



SQL - stručný přehled

- **Pohledy**

- CREATE VIEW jm_pohledu [(jm_sloupce, ...)]
AS tabulkový_výraz
[WITH CHECK OPTION]
- DROP VIEW jm_pohledu
[RESTRICT|CASCADE]



Vývoj postrelačných DB systémů

- ***Na zelené louce***
- **Od programovacího jazyka k DB**
- **Od existujícího schématu k novému**



Vývoj na zelené louce

- **Kompletní návrh a vývoj**
 - schématu DB
 - DDL a DML
 - algoritmů
 - ...
- *V současnosti asi nemá smysl, pokud se nejedná o zcela novou oblast*



Od programovacího jazyka ...

- **Základ**
 - existující programovací jazyk - C++
 - algoritmy
- **Přidává se**
 - perzistence datových položek
 - DDL a DML
 - algoritmy



Od existujícího schématu ...

- **Základ**
 - existující DB systém (známé schéma)
 - *pokulhávající* aplikace
- **Přidává se**
 - nové obraty do schématu/SŘBD
 - algoritmy
- **Změna**
 - DDL a DML

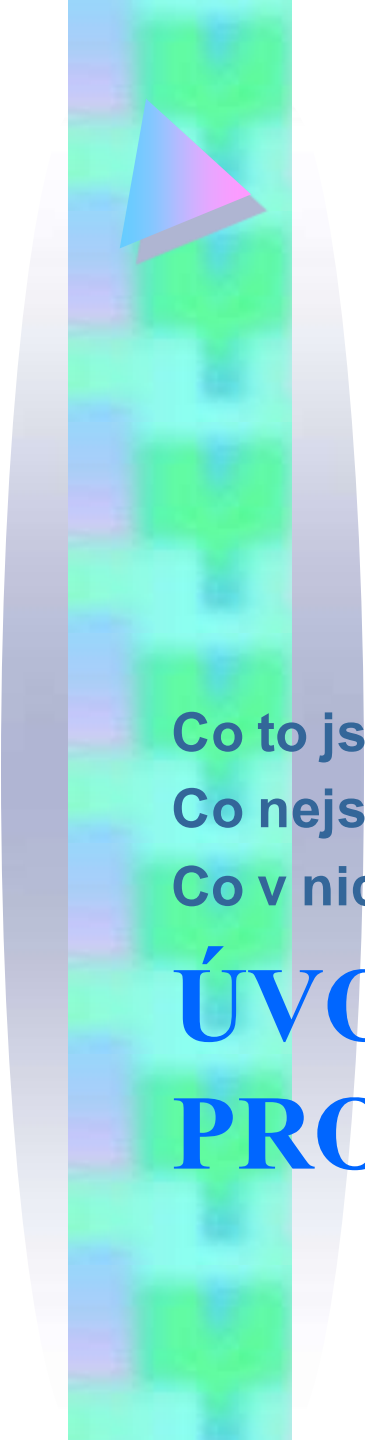


Inspirující *běžné* jazyky

- Imperativní
 - C++/C
 - ...
- Deklarativní
 - Prolog
 - funkcionální jazyky
 - ...



2. Prostorové databáze I



Co to jsou prostorové DB
Co nejsou prostorové DB
Co v nich realizujeme

ÚVOD DO OBLASTI PROSTOROVÝCH DB



Co jsou prostorové databáze

- **Prvotní požadavek**
 - spravovat data vztahující se k určitému prostoru
- **Charakteristika podpůrné technologie**
 - schopnost spravovat velké množství relativně jednoduchých geometrických objektů



Typy *prostorů*

- **2D**
 - geografie
 - GIS, LIS, urbanistika, ...
 - VLSI
- **3D**
 - vesmír
 - astronomie
 - modely mozku
 - medicína
 - molekulární biologie

Co může být v prostorových DB

- množiny *entit* z určitého prostoru

jasná
- identifikace
- umístění
- vztah k okolí

bitmapové obrázky
z určitého prostoru

Prostorové DBS

Obrázkové DBS

analýza, získání
údajů o vztahu,
vlastnostech, ...



Pracovní definice PDBS

- Prostorové DBS jsou databázové systémy
- DDL a DML zahrnují ***prostorové datové typy***
- Podpora prostorových databázových typů na ***implementační úrovni***
 - indexace
 - vyhledávání, join, ...



Modely v PDBS

- **Jaké entity je třeba ukládat**
- **Užité geometrické modely**
- **Prostorové datové typy/algebry**
- **Vztahy mezi entitami v PDBS**
- **Integrace geometrie do SŘBD a jeho datového modelu**



Co chceme uložit

- **Entity v prostoru**
 - města
 - řeky
- **Prostor jako takový**
 - obsahuje údaje o entitách v prostoru
 - tématické mapy
 - rozdělení na státy, provincie, ...



Požadavky na model

- **Modelování oddělených entit (objektů)**
- **Modelování skupin entit (objektů), které spolu *prostorově* souvisejí**



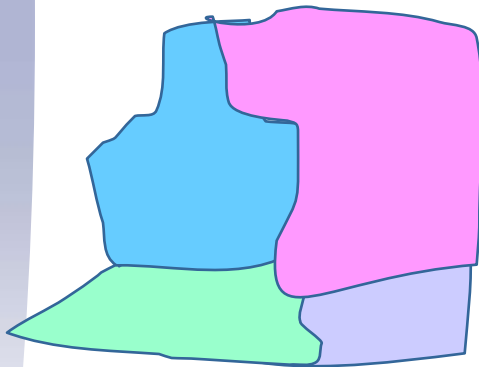
Oddělené entity

- **Základní abstrakce**

- **bod** *město*
 - není třeba ukládat informaci o *okolí*
- **úsečka (lomená)** *řeky, silnice, vedení*
 - pohyb v prostoru, spojení v prostoru
- **oblast** *les, jezero, město*
 - entita plus *okolí*

Prostorově související entity I

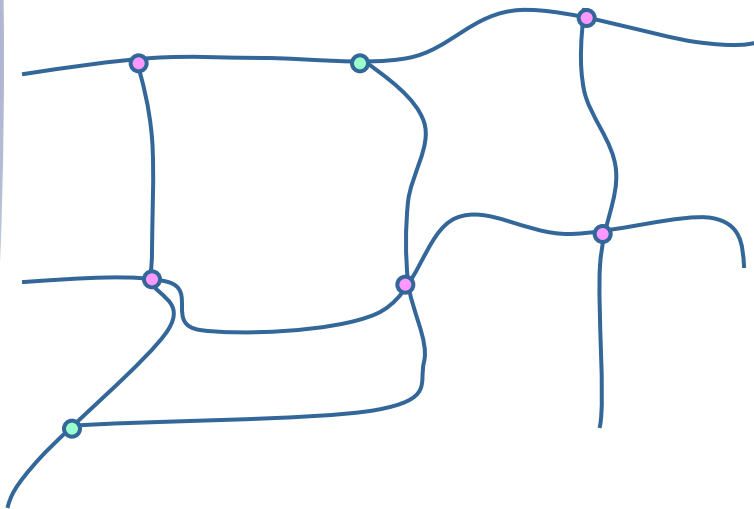
- **Plošné oddíly**
 - územní pokrytí, obvody, katastrální mapy, prostředí kolem význačných bodů (Voronoi)



Prostorově související entity II

- **Prostorové sítě (grafy)**

- silnice, ulice, železnice, řeky, elektrické sítě, telefonní vedení, ...



- **Další typy**

- vnořené plochy
- digitální modely terénu



Matematický model
Ukládání v počítači
Deskriptory, simplex

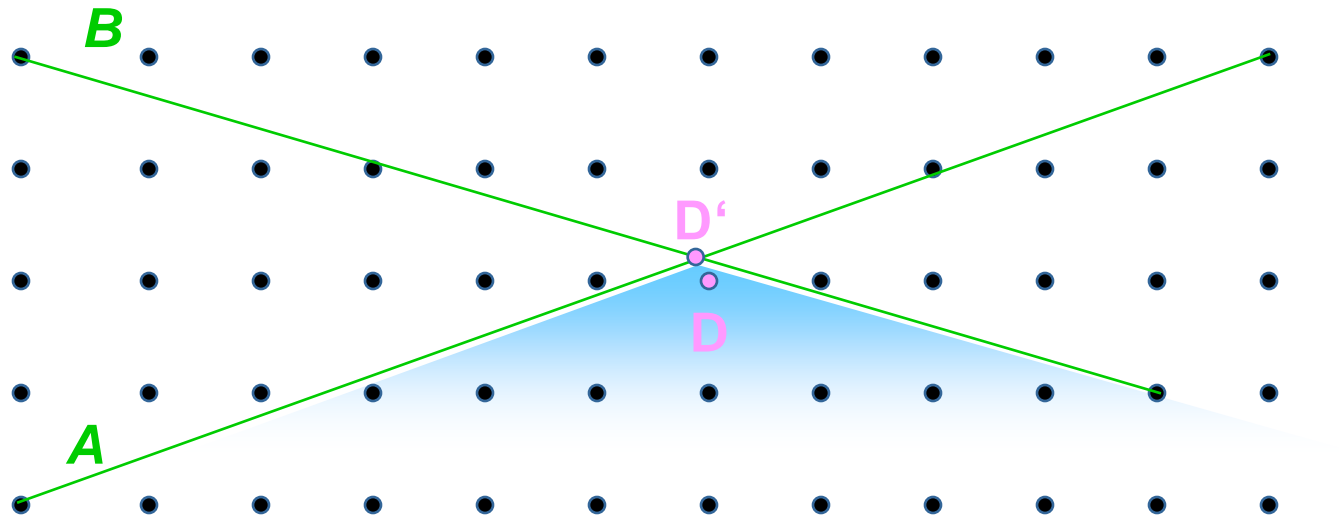
MODEL Y A PROBLEMATIKA REPREZENTACE V POČÍTAČI



Geometrické modely - Euklides

- Euklidovský prostor - **spojitý**
 - $p = (x,y) \in \mathbb{R}^2$
- Čísla v počítači - **diskrétní**
 - $p = (x,y) \in \text{real} \times \text{real}$

Problém v Euklidovském prostoru



- Leží bod D na úsečce A?
- Je bod D zcela obsažen v oblasti pod úsečkami A a B?



Řešení

- **Cíl**

- V průběhu geometrických operací již neprovádět další výpočty průsečíků

- **Oddělení dvou oblastí**

- definice typů a operací

**ošetření číselných problémů
vzhledem ke geom. modelu**

geom. model



Užité přístupy

- **Použití jednoduchých geometrických entit pro skládání složitějších objektů**
 - Frank & Kuhn 86
 - Egenhofer, Frank & Jackson 89
- **Kompletní popis modelované oblasti**
 - Güting & Schneider 93
 - Schneider 97

Jednoduché objekty - simplex

- **d-simplex**

- nejmenší objekt rozměru d

- 0-simplex



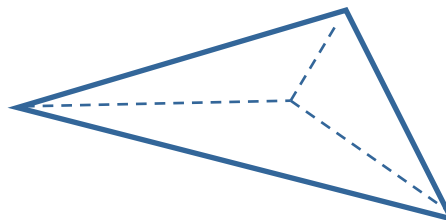
- 1-simplex



- 2-simplex



- 3-simplex



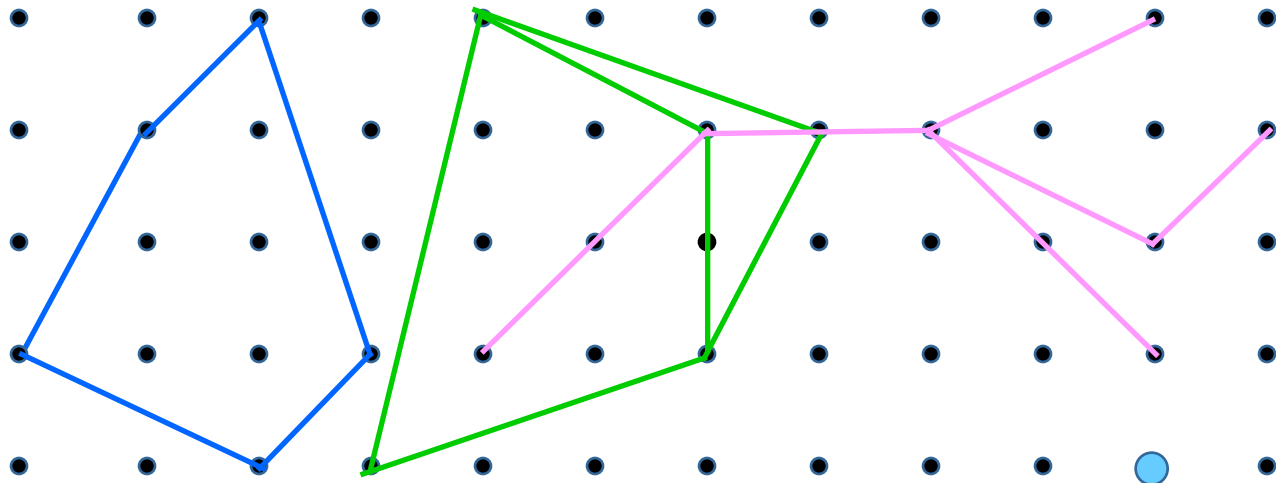
- d -simplex sestává z $d+1$ simplex-ů rozměru $d-1$



Skládání simplex-ů

- **Elementy tvořící jeden simplex se nazývají *styky* (faces)**
- **Komplex simplex-ů**
 - je konečná množina simplex-ů taková, že průnik libovolných dvou simplex-ů je styk

Úplné popisy - deskripty



- **Deskriptor (realm) - intuitivní pohled**
 - úplný popis všech útvarů celé aplikace
- **Formální pohled**
 - konečná množina bodů (úseček) nad sítí bodů daných vlastností (množina)



Deskriptory - vlastnosti

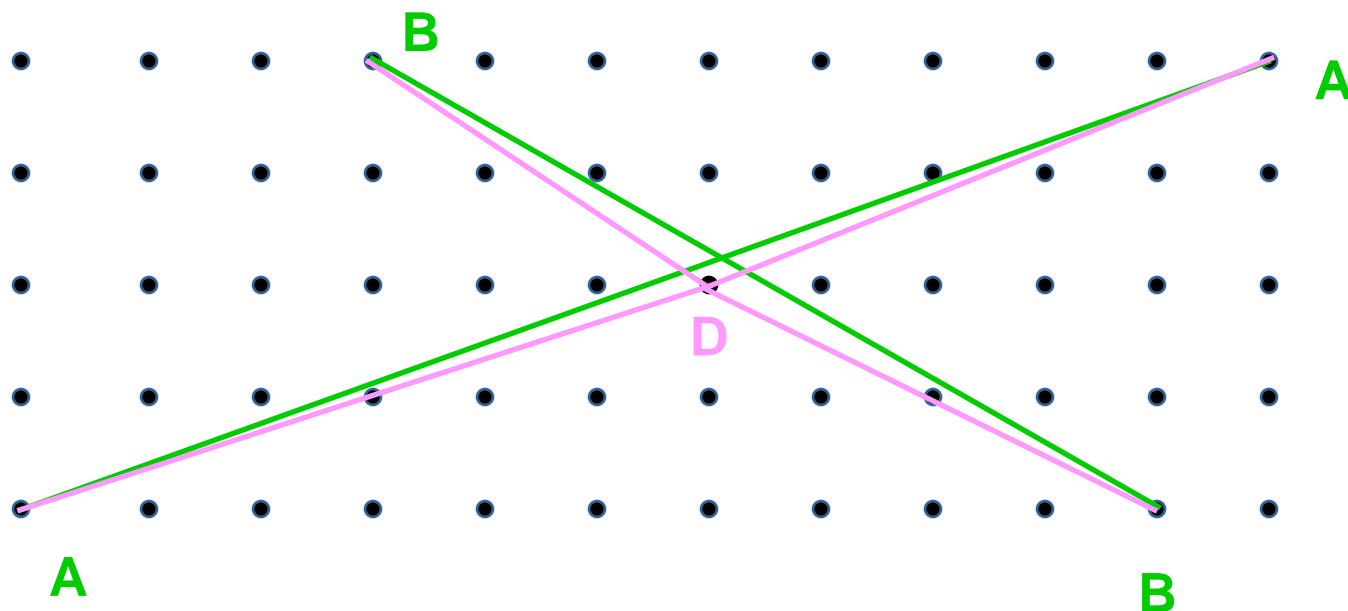
- **Formální vlastnosti - definice**
 - každý (koncový) bod je bodem sítě
 - každý koncový bod úsečky (složitějšího útvaru) je bodem sítě
 - žádný vnitřní bod úsečky (...) není zaznamenán v síti
 - žádné dvě úsečky (...) nemají ani průsečík ani se nepřekrývají



Co deskripty skrývají

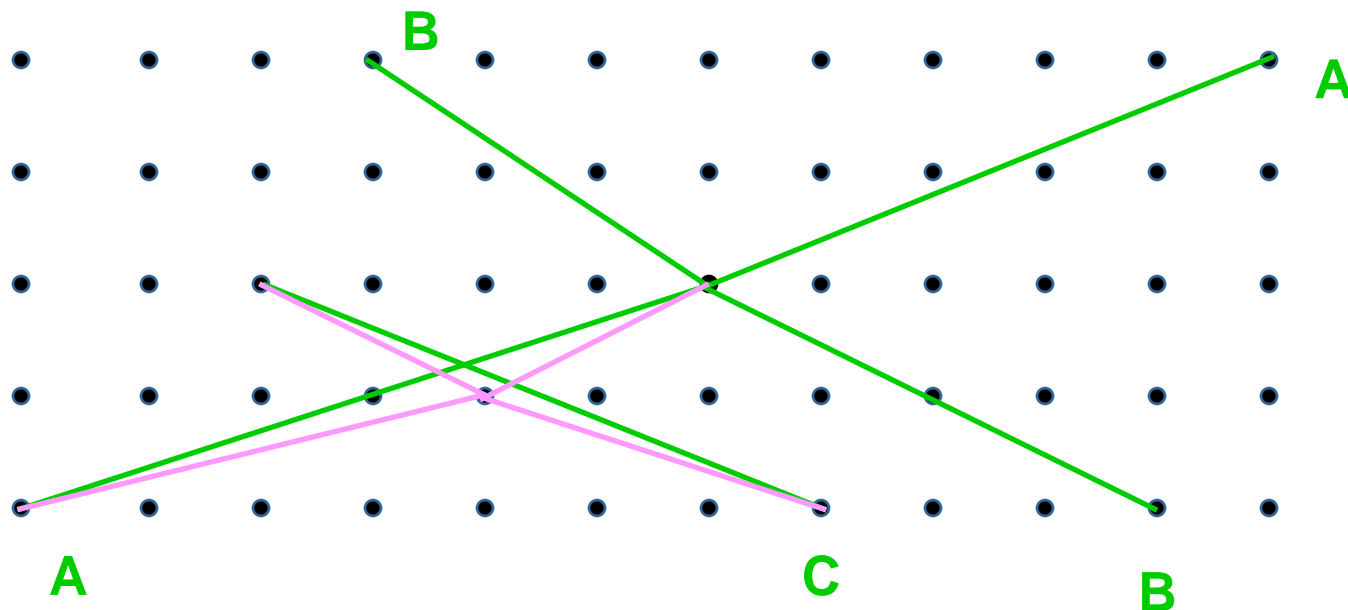
- **Pod vrstvou deskriptorů se skrývají problémy s číselnou reprezentací (zpracováním).**
 - **Aplikační data často obsahují průsečíky vnitřních bodů grafických elementů (úseček) - ty však ale nemusí být v síti.**

Řešení uvedené problematiky



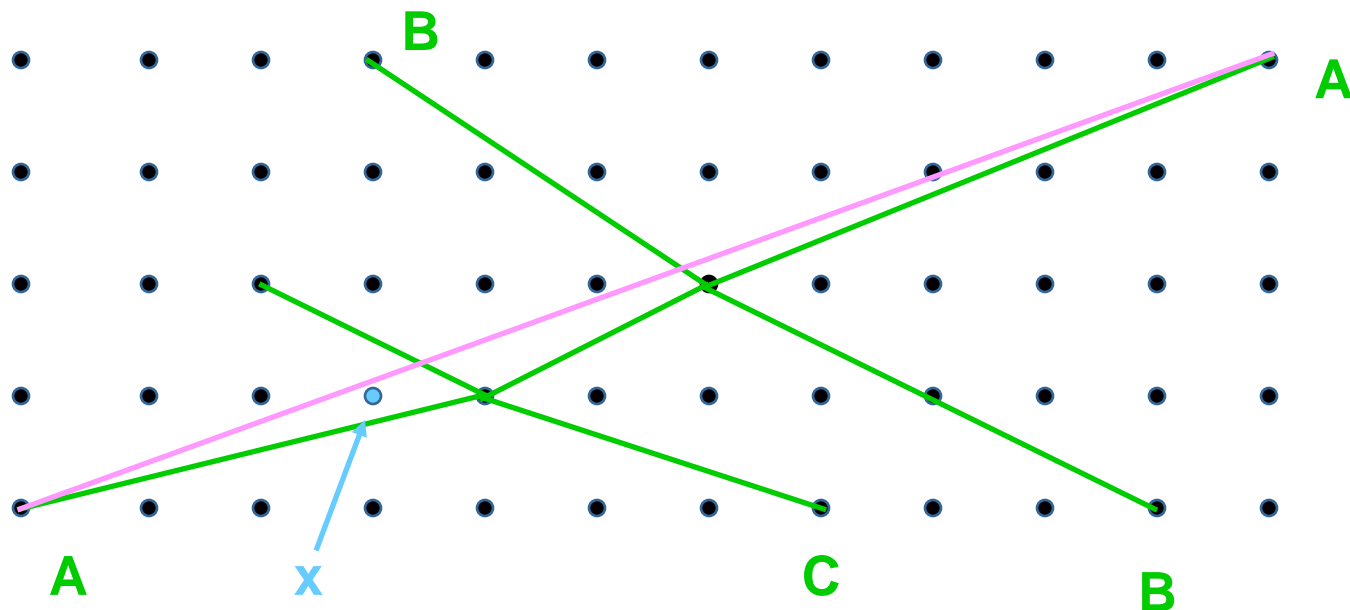
- *Drobné narušení obou segmentů*

Je to opravdu řešení?




- ... a teď si to překreslíme i s původní úsečkou A

Jak se zdá ...



- ... tak nikoliv. Bod x je nyní na opačné straně úsečky. Co teď s tím?



Segmenty „do obálky“

- Segmenty (úsečky) se překreslují pouze v rámci své obálky
 - Greene & Yao 1986
- Segmenty v obálce nikdy *nemohou* přeskočit *přes bod mřížky*.



Jak to formálně popsat
Jak ten popis utvořit

DATOVÉ TYPY A ALGEBRY



Datové typy a algebry

- **Požadavky na prostorové datové typy (PDT)**
 - „vzhled“ datových položek musí být uniformní v rámci množinových operací nad množinami bodů tvořící data
 - formální definice dat a funkcí PDT
 - definice zohledňující aritmetiku s konečnou přesností
 - podpora pro konzistentní popis prostorově souvisejících objektů
 - nezávislé na konkrétním SŘBD, ale s daným úzce spolupracující



Geo-relační algebra

- **Relační algebra, která je použita jako vícedruhová algebra (relace + atomické datové typy)**

- **Druhy**

- **REL**

- NUM**

- STR**

- BOOL**

- POINT**

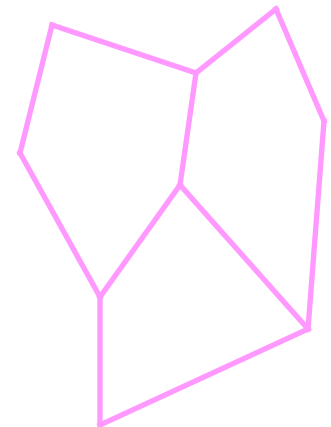
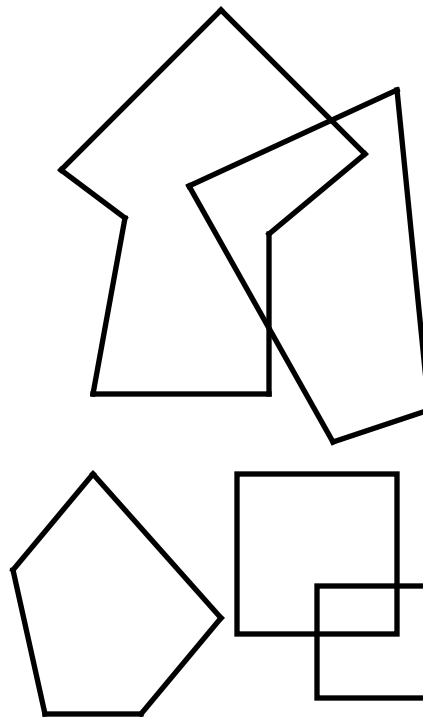
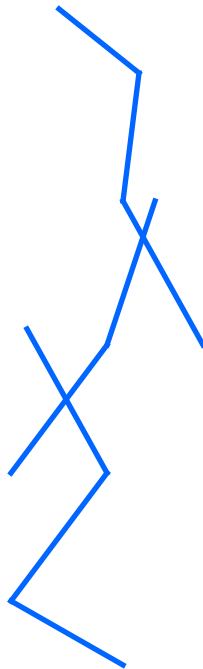
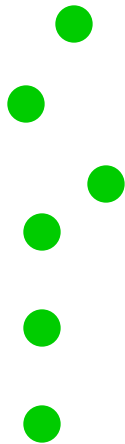
- LINE**

- REG - PGON**

- AREA**

Příklady

- **POINT** **LINE** **PGON** **AREA**



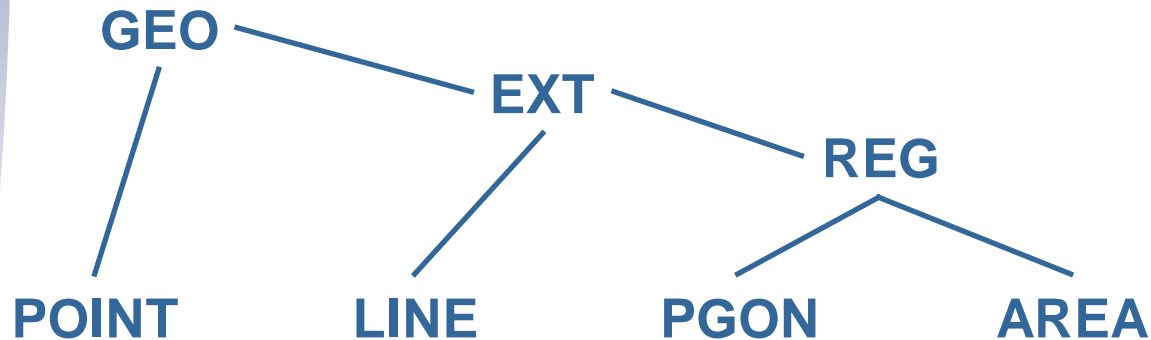


Tabulka Města

- **města**
 - **jmenoM** **STR**
 - **stred** **POINT**
 - **obyvM** **NUM**

Operace - predikáty

- **POINT x POINT** \neq
- **LINE x LINE**
- **REG x REG**
- **GEO x REG** uvnitř
- **EXT x EXT** průnik
- **AREA x AREA** sousedí_s

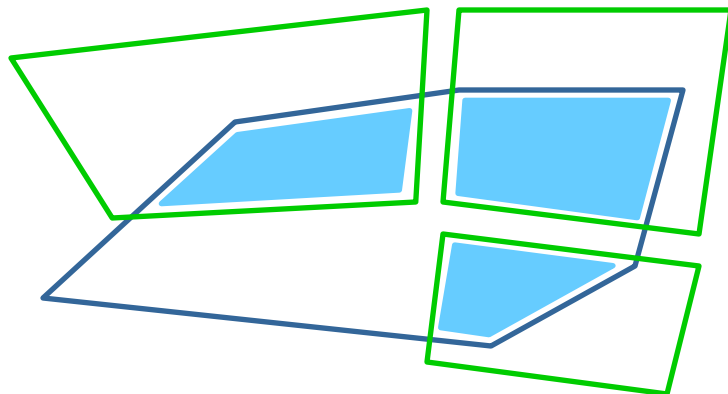




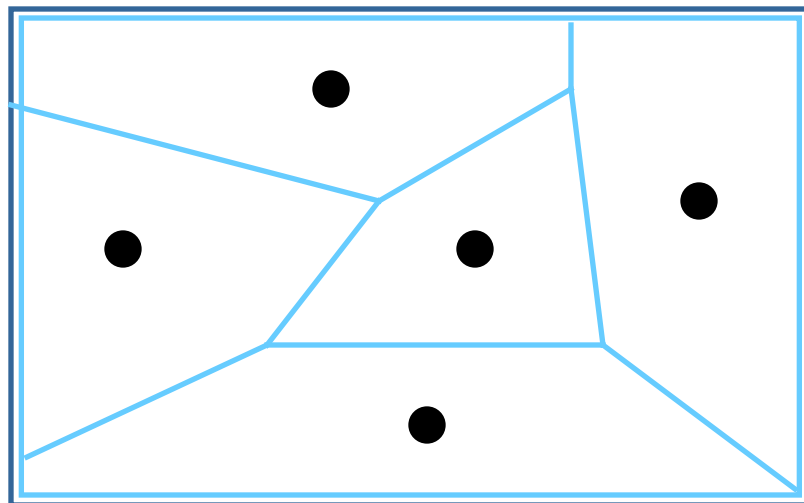
Operace - geometrické relace

- $\text{LINE}^* \times \text{LINE}^* \rightarrow \text{POINT}^*$ průnik
- $\text{LINE}^* \times \text{REG}^* \rightarrow \text{LINE}^*$
- $\text{PGON}^* \times \text{REG}^* \rightarrow \text{PGON}^*$
- $\text{AREA}^* \times \text{AREA}^* \rightarrow \text{AREA}^*$ překrytí
- $\text{EXT}^* \rightarrow \text{POINT}^*$ uzly (grafu)
- $\text{POINT}^* \times \text{REG} \rightarrow \text{AREA}^*$ voronoi
- $\text{POINT}^* \times \text{POINT} \rightarrow \text{REL}$ nejbližší

Příklad



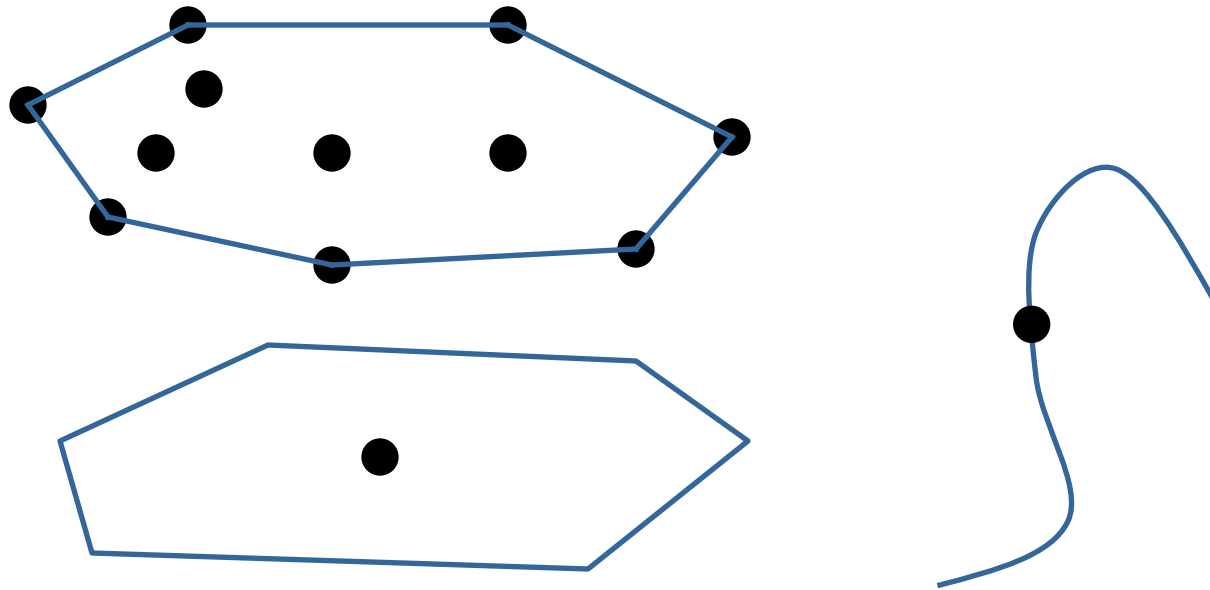
překrytí



voronoi

Operace vracející atomické objekty jako výsledek

- **POINT*** → **PGON** konvexní obálka
- **POINT*** → **POINT** střed
- **EXT** → **POINT**





Operace vracející čísla

- **POINT x POINT** → **NUM** vzdálenost
- **GEO x GEO** → **NUM** min/max vzdálenost
- **POINT*** → **NUM** průměr
- **LINE** → **NUM** délka
- **REG** → **NUM** plocha, obvod



Srovnání s požadavky

- **Uniformita v rámci operací** -
- **Formální definice** +
- **Aproximace reálných čísel** -
- **Podpora „sousednosti“** (-)
- **Nezávislost na SŘBD** -



Shrnutí

- **Plus**

- z konceptuálního pohledu jsou objekty jednoduché
- jednoduchá formální definice
- jednoduché datové struktury a algoritmy

- **Mínus**

- jednoduché obrazce
- vytvoření průniku je složité
- chybí rozdílové operace
- chybí operace, které jsou výpočetně složité



3. Prostorové databáze II

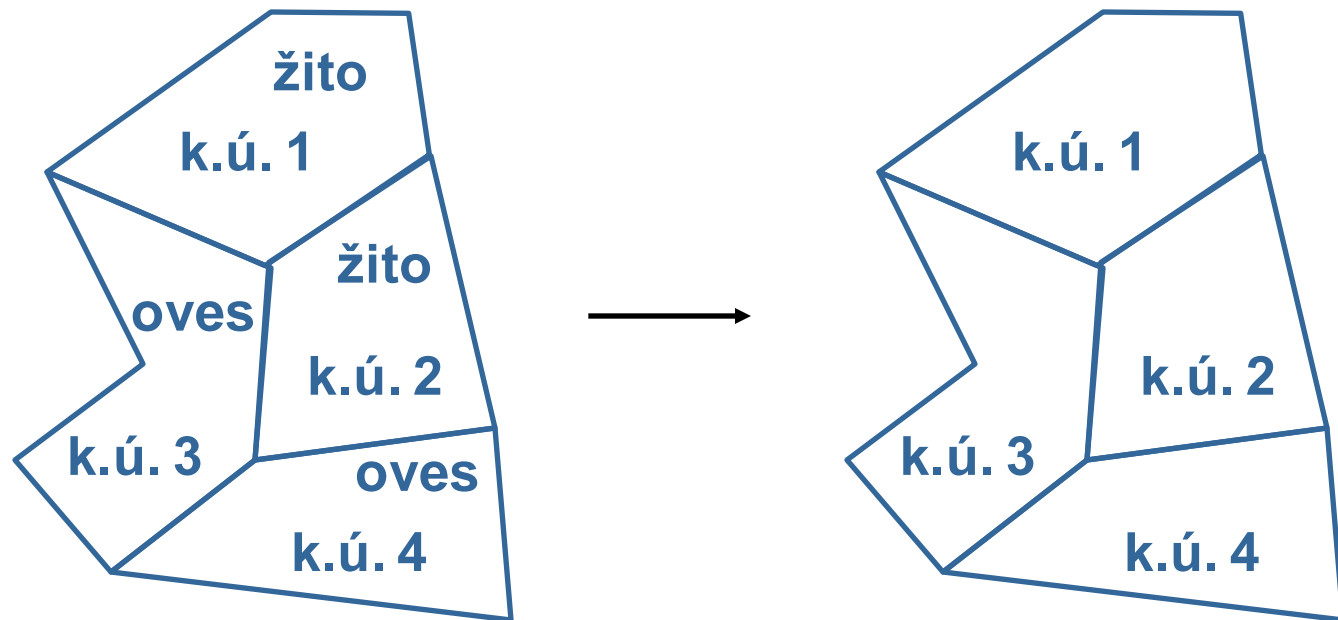


Algebra pro manipulaci oddílů

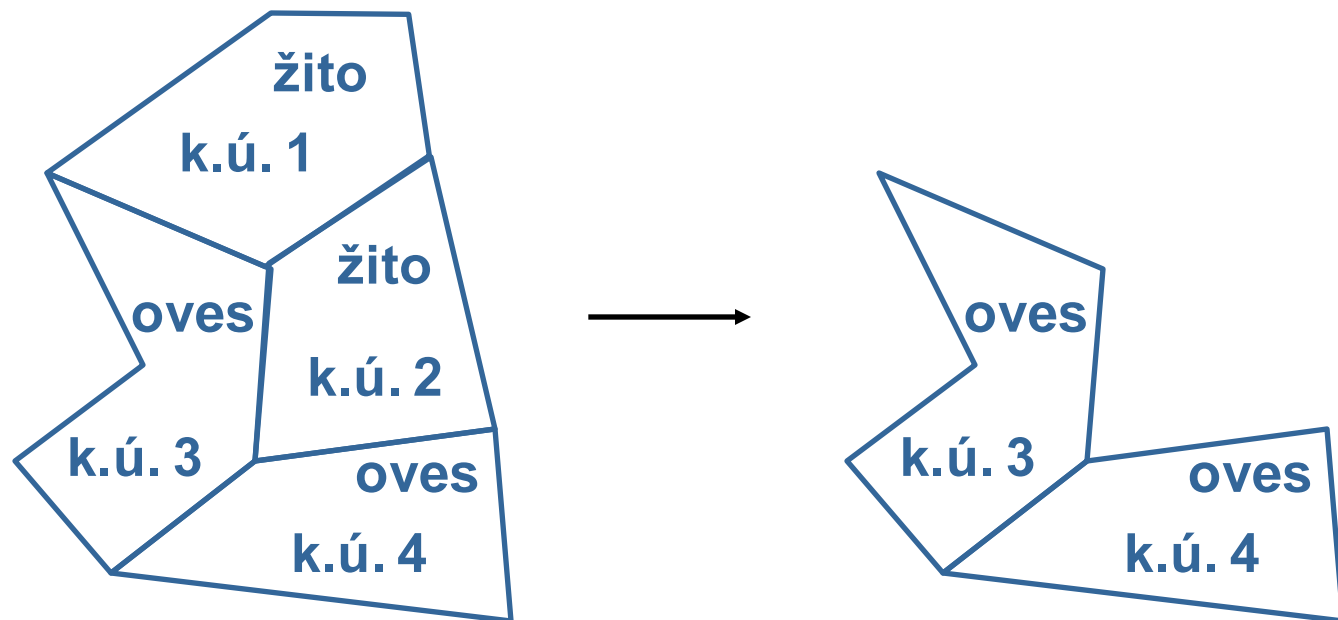
- **Oddíl = mapa, zobrazení**
 - Scholl & Voisard 89
- **Cíl:**
 - formální modelování operací nad oddíly (mapami), včetně těchto
 - projekce
 - selekce
 - fúze (projekce se spojením)
 - “windowing”
 - ořezání



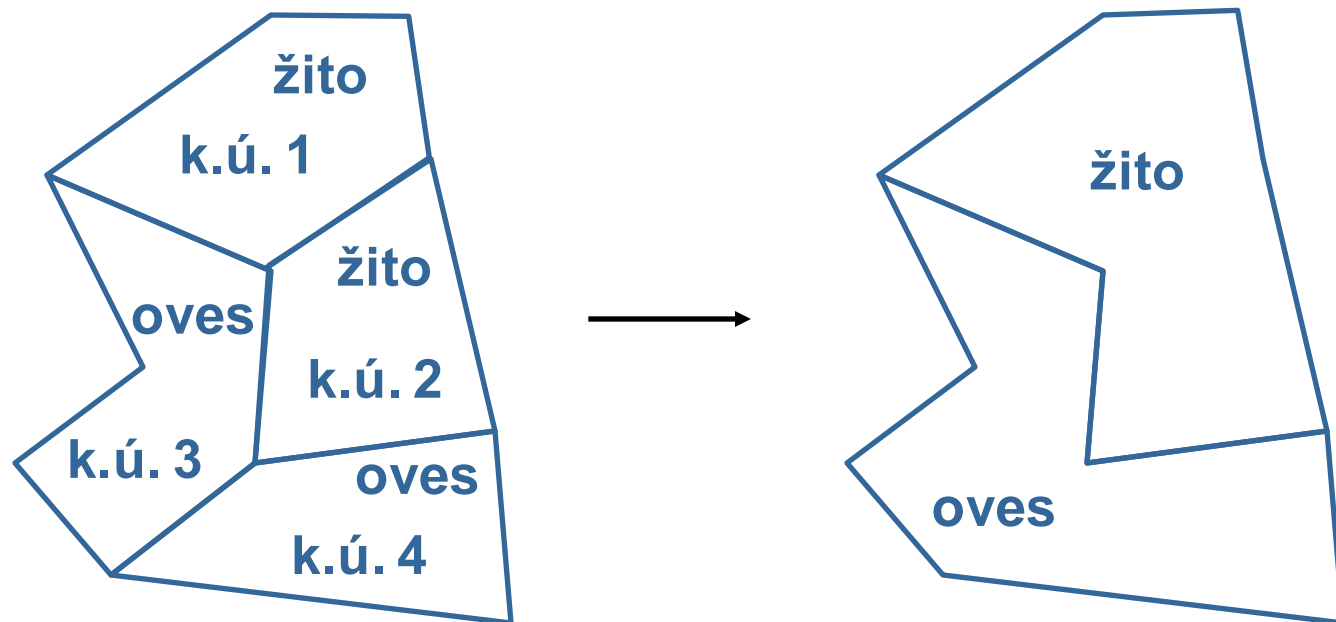
Projekce



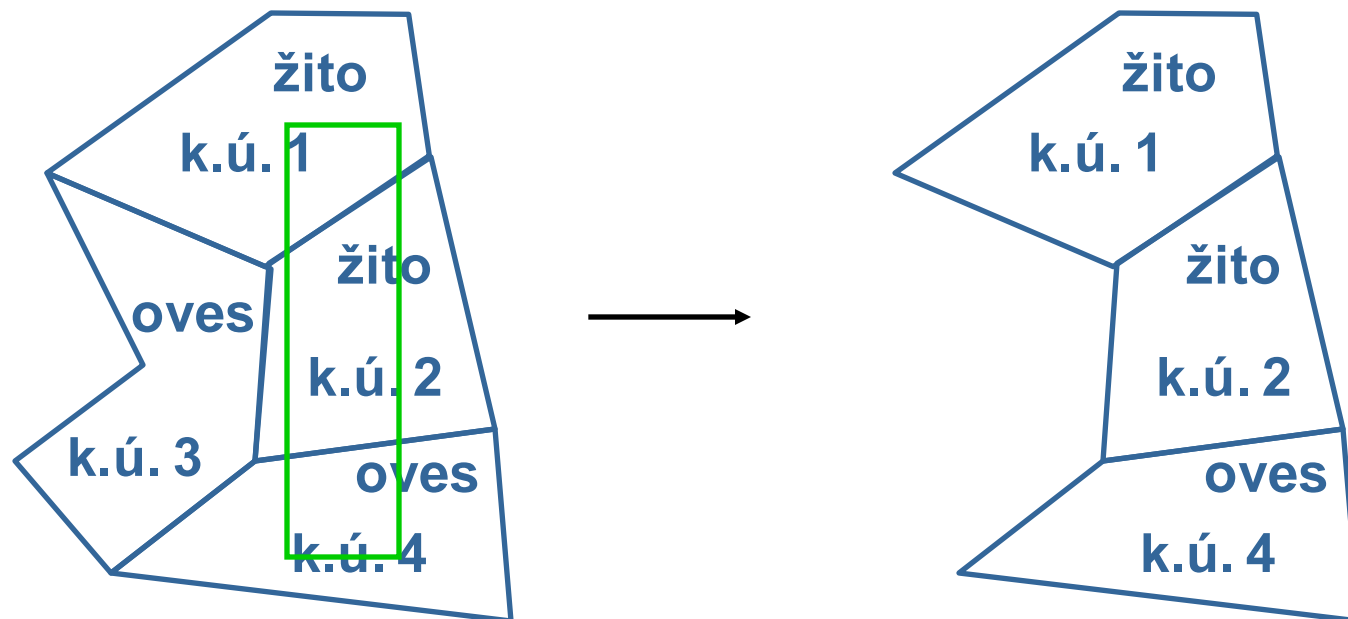
Selekce



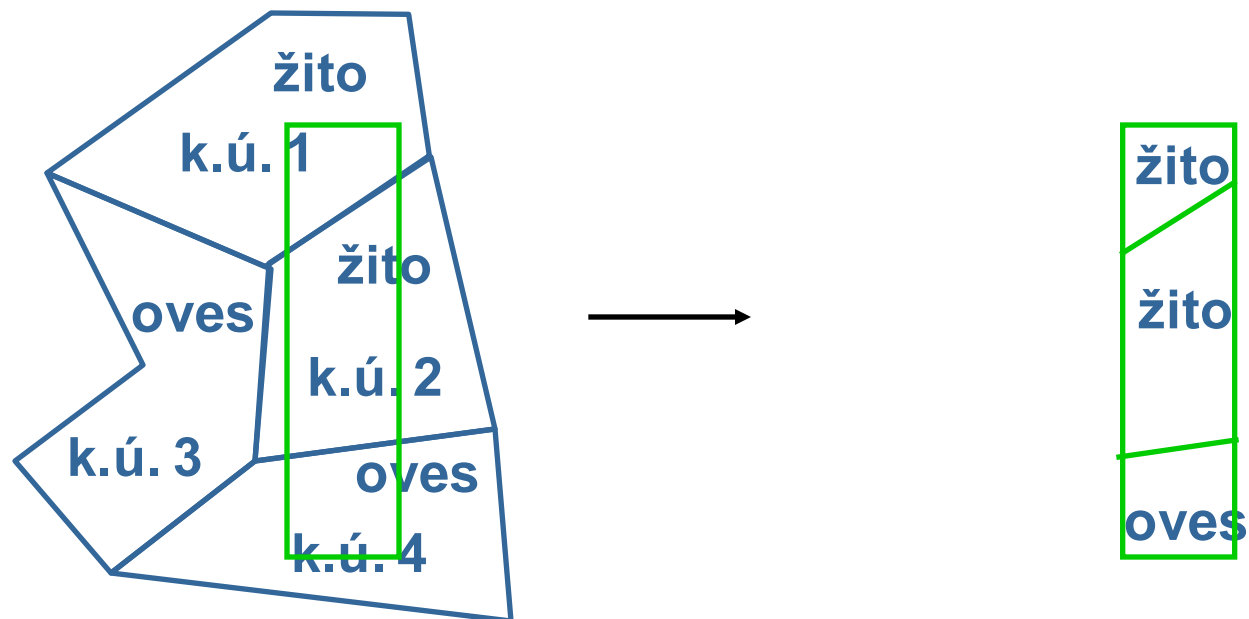
Fúze - projekce a spojení



„Windowing“



Ořezání





Datové typy

- Pro oddíly existuje datový typ *region*
 - označení γ , či $\{\gamma\}$
 - konvexní i konkávní polygon
- Mapa je potom množina n-tic, které mají jako jeden z atributů region



Použitá algebra

- **Algebra pro skládané objekty**
 - operátor vnoření (vložení)
 - primitivní operace nad regiony
 - množinové geometrické operace

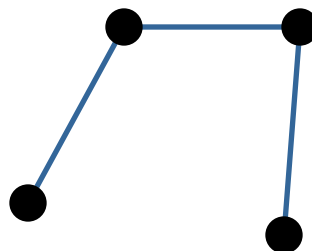
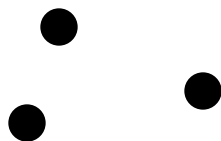


Algebra ROSE

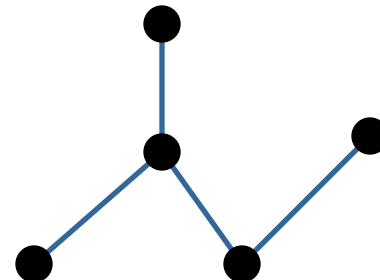
- **RObust Spatial Extension**
 - existuje množina povolených deskriptorů (realms)
 - složitější objekty jejich kompozicí
 - definice, implementace
- **Základní typy**
 - body, úsečky, oblasti

Povolené hodnoty

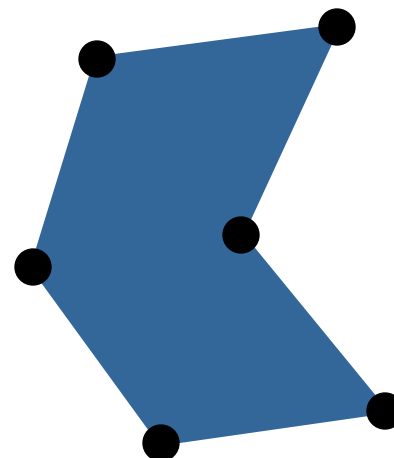
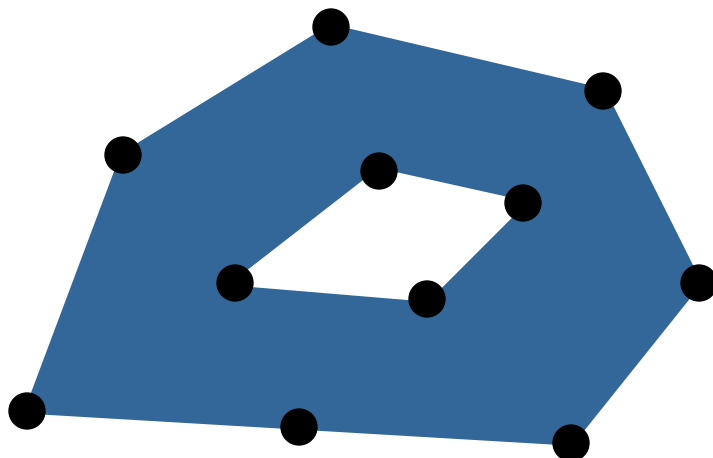
body



úsečky



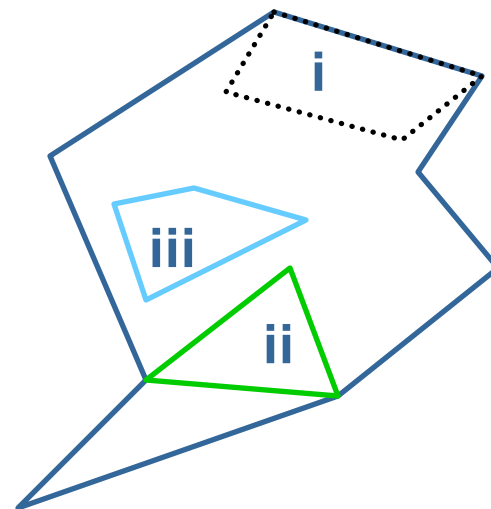
oblasti



Míra „vnoření“

- **Objekt je**

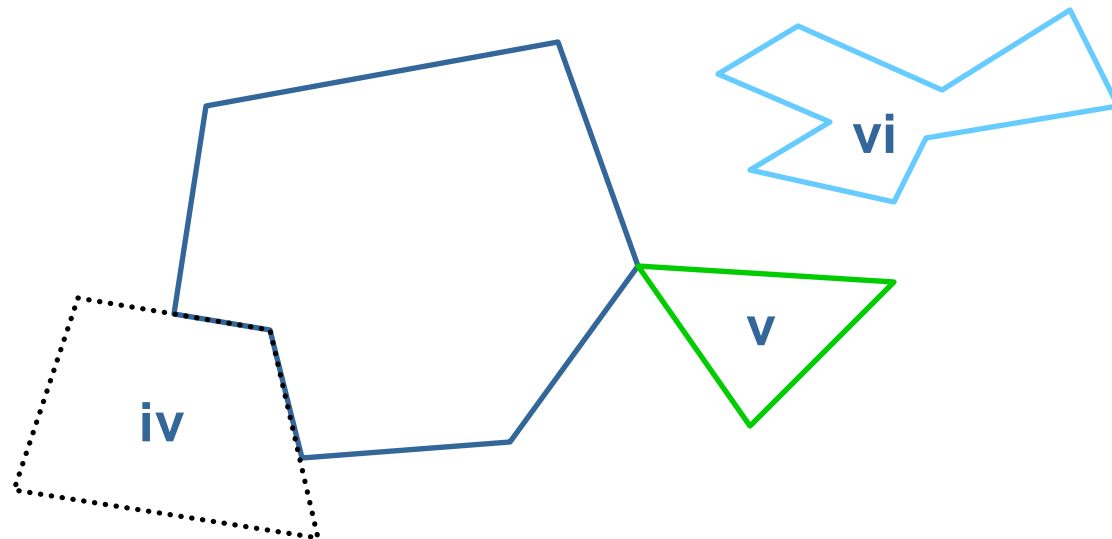
- uvnitř (plošně) - i, ii, iii
- hranově vnořený - ii, iii
- vrcholově vnořený - iii



Míra disjunkce

- **Objekty jsou**

- plošně disjunktní - iv , v , vi
- hranově disjunktní - v , vi
- zcela (vrcholově) disjunktní - vi





R-plocha

- **Definice:**

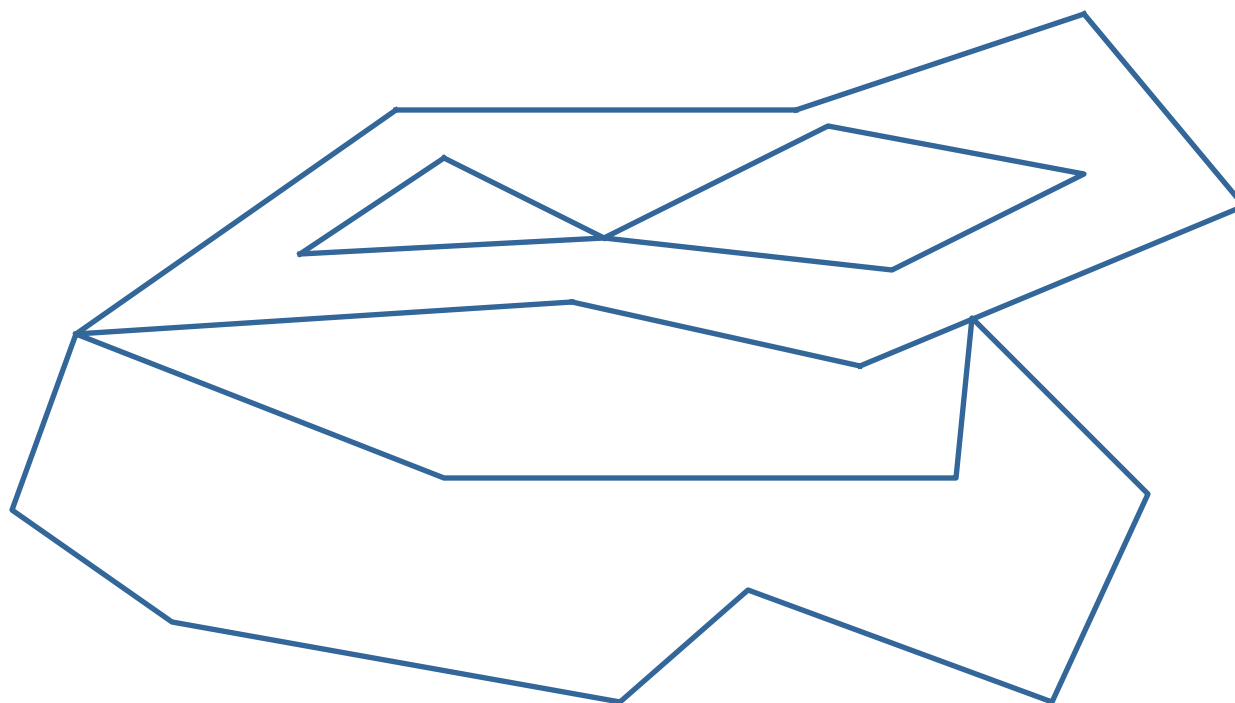
- R-plocha f je dvojice (c, H) taková, že c je R-cyklus, $H = \{h_1, \dots, h_m\}$ je množina R-cyklů a platí:

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

- *poslední podmínka zaručuje jednoznačnost reprezentace*



Příklad

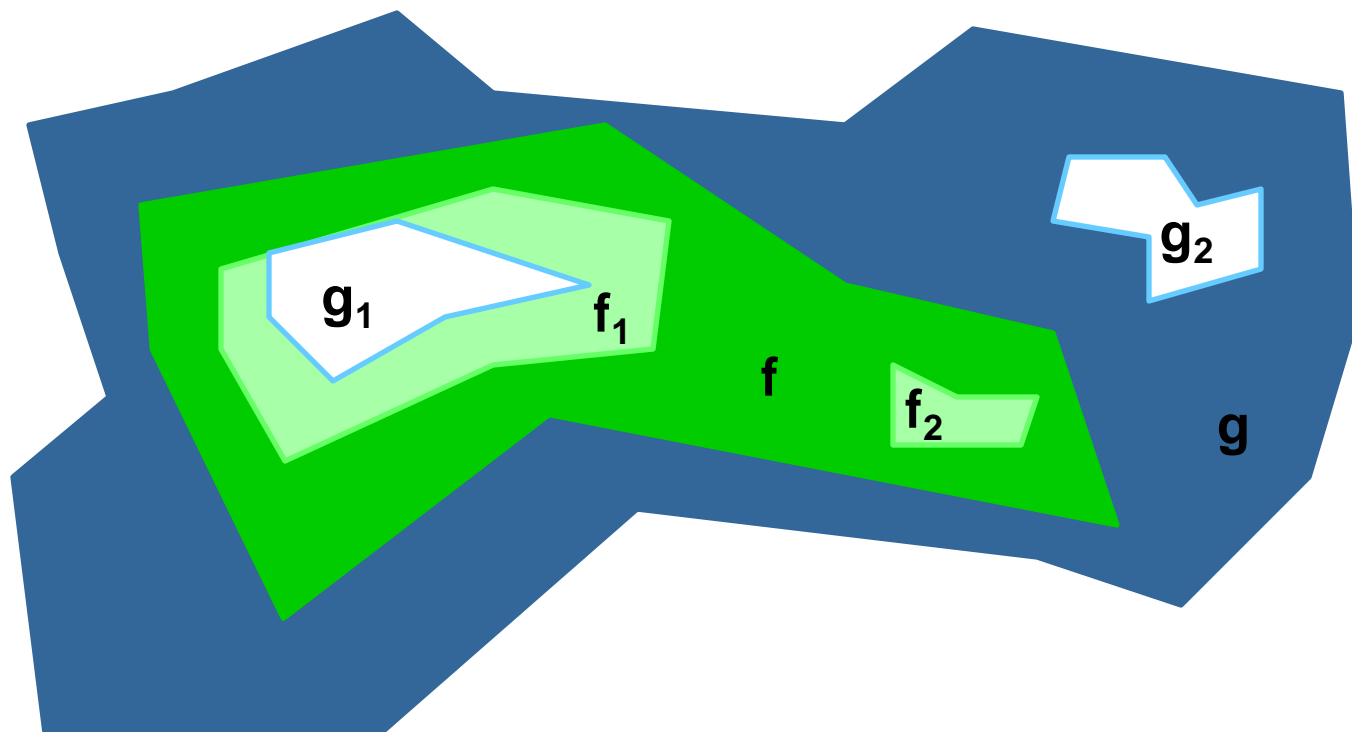




Vnořené plochy

- Necht' $f=(f_0, F)$ a $g=(g_0, G)$ jsou dvě R-plochy. Říkáme, že f je plošně obsažena 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

Příklad





Typ *regions*

- Hodnota F typu *regions* je množina hranově disjunktních R-ploch
- Necht' 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



Algebra ROSE

- **Má přesně definovány operace pro manipulaci hodnot těchto typů:**
 - **GEO**
 - points
 - lines
 - regions
 - **EXT**
 - lines
 - regions



Predikáty v ROSE

- $\forall \textit{geo} \in \textit{GEO}, \forall \textit{ext} \in \textit{EXT}, \text{ platí:}$
 - $\textit{geo} \times \textit{regions} \rightarrow \textit{bool}$
 - uvnitř/vnořen
 - $\textit{regions} \times \textit{regions} \rightarrow \textit{bool}$
 - plošně disjunktní
 - hranově disjunktní
 - hranově vnořený
 - vrcholově vnořený



Výpočetně náročné operace

- ***points* × *ext* → *bool***
– *leží na hranici*
- ***ext* × *regions* → *bool***
– *je hranicí*



Geometrické operace

- Je možné určit průnik, spojení a rozdíl libovolných dvou objektů
 - *points* \times *points* \rightarrow *points*
 - průnik
 - *lines* \times *lines* \rightarrow *points*
 - *regions* \times *regions* \rightarrow *regions*
 - *regions* \times *lines* \rightarrow *lines*
 - *geo* \times *geo* \rightarrow *geo*
 - plus, mínus



Operace nad objekty DB

- Jsou definovány prostřednictvím modelu objektového rozhraní
 - $\forall \text{obj} \in \text{OBJ}, \forall \text{geo}, \text{geo}_1, \text{geo}_2 \in \text{GEO}$:
 - $\text{set}(\text{obj}) \times (\text{obj} \rightarrow \text{geo}) \rightarrow \text{geo}$
 - složení
 - $\text{set}(\text{obj}) \times (\text{obj} \rightarrow \text{geo}_1) \times \text{geo}_2 \rightarrow \text{set}(\text{obj})$
 - nejbližší
- *taktéž definováno překrytí a propojení*



Srovnání s požadavky

- **Uniformita v rámci operací** +
- **Formální definice** +
- **Aproximace reálných čísel** +
- **Podpora „sousednosti“** +
- **Nezávislost na SŘBD** +



Nevýhody

- Chybějí operace pro vytvoření nového geometrického uskupení (voronoi, střed, konvexní obálka)
- Integrace DBS a deskriptorů není jednoduchá. (Při změně atributů deskriptoru je třeba vyhledat příslušný objekt a u něj změnit hodnoty atributů definující deskriptor)



Shrnutí (datové typy/algebry)

- **otevřenost/rozšiřitelnost**
 - extensibility
- **úplnost**
 - completeness
- **1/více typů?**
- **operace nad množinou DB objektů**



VZTAHY V PROSTORU A JEJICH STUDIUM



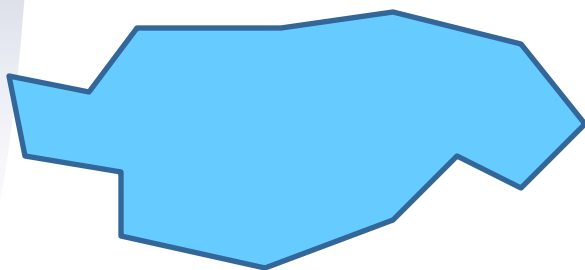
Vztahy v prostoru

- Jsou těmi nejvýznamnějšími operacemi prostorových algeber.
 - Např. nalezení všech objektů s danou vlastností, vymezení dat pro operaci
 - topologické: uvnitř, průnik, vedle, ...
 - invariantní vzhledem k posunutí, rotaci, změně měřítka
 - směrové: nad, pod, severně od, ...
 - metrické: vzdálenost < 100 , ...



Hloubka studia vztahů

- Jaká je dostatečná úroveň?
- Existují kritéria pro definování úplnosti?
- **ANO!**
 - Egenhofer 89 (a další jeho práce)
 - původně jen pro jednoduché objekty
 - bez „děr“, jen propojené



hranice

vnitřní oblast

Možné operace

$\partial A1 \cap \partial A2$	$\partial A1 \cap A2^\circ$	$A1^\circ \cap \partial A2$	$A1^\circ \cap A2^\circ$	operace
\emptyset	\emptyset	\emptyset	\emptyset	disjunktní
\emptyset	\emptyset	$\neq \emptyset$	$\neq \emptyset$	2 uvnitř 1
\emptyset	$\neq \emptyset$	\emptyset	$\neq \emptyset$	1 uvnitř 2
$\neq \emptyset$	\emptyset	\emptyset	\emptyset	1, 2 se dotýkají
$\neq \emptyset$	\emptyset	\emptyset	$\neq \emptyset$	1, 2 shodné
$\neq \emptyset$	\emptyset	$\neq \emptyset$	$\neq \emptyset$	1 pokrývá 2
$\neq \emptyset$	$\neq \emptyset$	\emptyset	$\neq \emptyset$	2 pokrývá 1
$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	1, 2 přesahují

- 4 metody na test průniku ~ 16 kombinací
- *jen některé mají „význam“*
- *nestačí*



Další operace

- **Body a úsečky**
 - Egenhofer & Herring 92
 - de Hoop & van Oosterom 92
- **Průniky doplňků k objektům A^{-1}**
 - Egenhofer 91
- **„Úroveň“ průniku (0D, 1D, 2D)**
 - Clementini a kol. 93
 - 256 kombinací, 52 platných
 - příliš mnoho



Alternativa

- **5 operací**
 - dotek, uvnitř, přes, přesah, disjunkce
- **3 operátory na extrakci hranice**
- **Lze dokázat**
 - 5 vztahů je vzájemně výlučných
 - 5 + 3 umožňují rozlišit všech 52 možností



Další rozšíření

- **Oblasti s „dírami“**
 - Egenhofer, Clementini, Di Felice 94
- **Skládané oblasti**
 - Clementini, Di Felice, Califano 95
- **Vztahy mezi obalující hyper-krychlí a oblastí**
 - Papadias 95
 - 13 možností v 1D, 169 ve 2D



Sítě

- Relativně *nedotčená* oblast (i když grafy jako takové podpořeny jsou)
 - Nevýhoda
 - grafy nejsou viditelné uživateli
 - nejsou *dobře* podpořeny
 - GraphDB
 - Güting 94
 - explicitně integrované grafové elementy v OO modelu



4. Prostorové DB III



Vlastnosti
Požadavky
Řešení

INTEGRACE V SŘBD



Integrace „geometrie“ se SŘBD

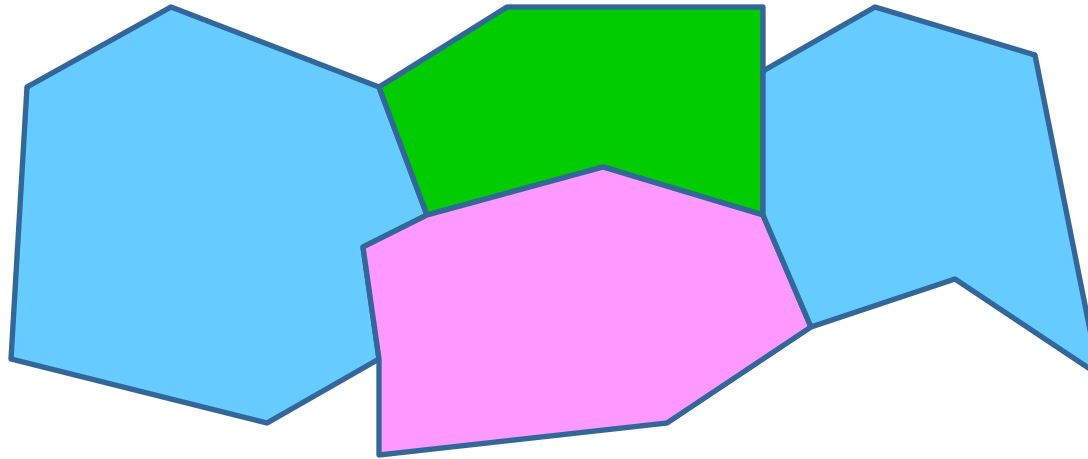
- **Základní myšlenka**
 - reprezentovat „prostorové objekty“ jako *objekty s alespoň jedním atributem definovaným nad PDT*
 - SŘBD musí podporovat tvorbu uživatelsky definovaných typů (ADT)



Příklad - relační model

- **relation** states (sname: STRING; area REGION; spop: INTEGER)
- **relation** cities (cname: STRING; center: POINT; ext: REGION; cpop: INTEGER)
- **relation** rivers (rname: STRING; route: LINE)

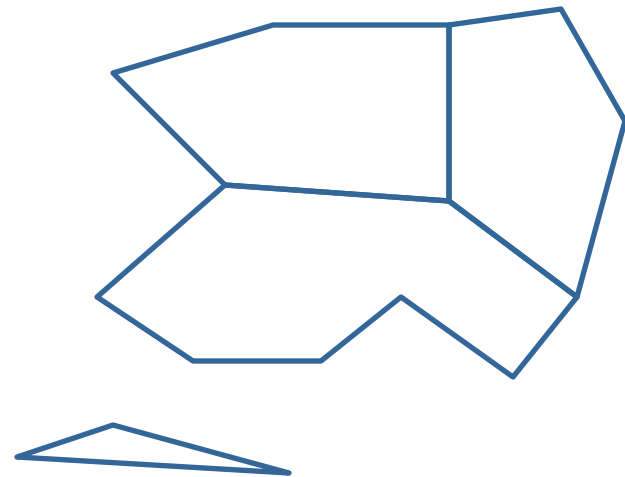
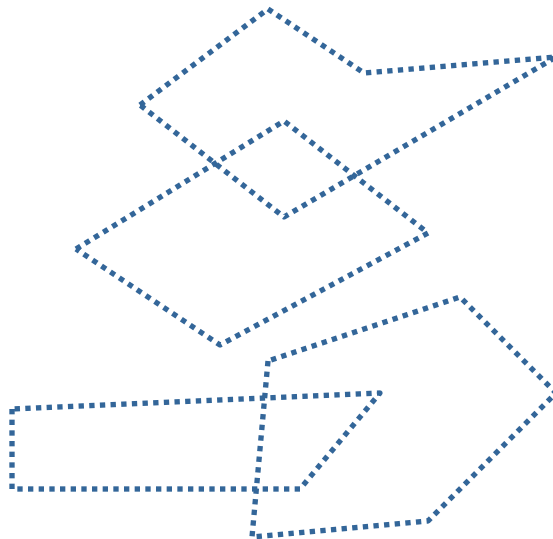
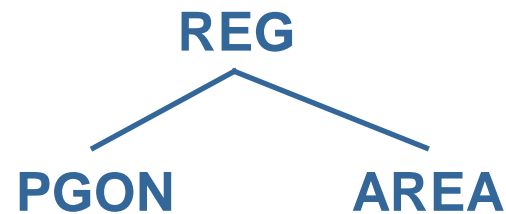
Objekty prostorově vztažené



- **Množina objektů s atributem REGION**
 - **ztráta informace**
 - disjunkce oblastí
 - sousednost oblastí (index)

Geo-relační algebra

Podobná omezení
by měla vycházet
z integritních omezení
nad relacemi





Dotazování

- **Vložení operací prostorové algebry do dotazovacího jazyka SŘBD**
 - základní operace (algebra) pro manipulaci množin DB objektů
 - grafický vstup/výstup
 - rozšíření dotazovacího jazyka



Základní operace

- **prostorová selekce**
- **prostorové spojení (join)**
- **aplikace prostorových funkcí**
- ***a další množinové operace***



Prostorová selekce

- **Selekce, která spočívá na prostorovém predikátu**
 - **Vyhledej všechna města na Moravě**
 - cities select[center inside Morava]
 - **Vyhledej všechny řeky, které protínají dotazovací okno**
 - rivers select[route intersects Window]
 - **Vyhledej všechna větší města do 50km od Olomouce**



Prostorové spojení (join)

- **Spojení založené na predikátu testujícím prostorové atributy**
 - **Ukaž příslušnost měst k jejím státům**
 - cities states join [center inside area]
 - **Najdi všechna města v blízkosti řek**
 - cities rivers join [dist(center, route) < 20]
 - **Ukaž větší města i s jejich okolím**



Aplikace prostorových funkcí

- Kde je možné uplatnit operace prostorové algebry, které vytvářejí nové PDT?
 - Např.
 - regions x lines -> lines (*intersection*)
 - podmínky při selekci
 - „mapování“ operací
 - filtr, náhrada, mapování, rozšíření

- ```
– rivers select [route intersects Morava]
 extend [intersection (route,Morava)
 {part}]
 extend [length(part) {plength}]
 project [rname, part, plength]
```



## Další množinové operace

- Mohou například různými způsoby pracovat s celými množinami prostorových objektů
  - operace je z konceptuálního hlediska jednotka
  - oddělení manipulace s daty v rámci SŘBD a prostorové algebry (většinou) není možné



# Příklady takových operací

- **Překrytí (přenesení)**
- **Spojení (fúze)**
- **Voronoi**
  
- **Napojení SŘBD na prostorovou algebru je v takových případech dosti obtížné**



# Grafický vstup a výstup

- **Požadavek**
  - grafická reprezentace PDT ve výsledcích dotazů
  - vkládání konstant PDT (konstanty v dotazech)
  - Rozumné zobrazení výsledků (překrývání, Z-buffer, ...)



# Požadavky na dotazování v PDT

- Existence PDT
- Grafické znázornění výsledků
- Grafická kombinace několika výsledků
- Zobrazení i s kontextem
- Kontrola stavu displeje
- Dialog
- Různé typy zobrazení
- Legenda, popisky
- Změna měřítka
- Výběr podoblastí

*Egenhofer 94*



## GUI - typicky 3 okna

- **Textové okno pro textovou reprezentaci objektů PDT**
- **Grafické okno pro grafické zobrazení objektů PDT**
- **Textové okno pro vkládání dotazů a zobrazování systémových hlášení**
- *interakce na úrovni textu i grafických entit*





# Skladba dotazu

- **Dotaz může sestávat z těchto částí**
  - popis objektů, které jsou výsledkem dotazu
  - rozdělení těchto objektů na podmnožiny pomocí obrazových/grafických dotazů
  - popis, jak zobrazit tu kterou podmnožinu objektů v závislosti na jejich prostorových attributech
- **GPL (gr. pres. lang.)**



# Rozšíření jazyka

- **Co je třeba vzít v úvahu**
  - hodnoty PDT/grafický vstup
  - čtyři základní operace v rámci SŘBD
    - selekce, spojení, aplikace funkcí, množinové operace
  - **prezentace výsledků**



# Atomické hodnoty PDT

- **DEFINE** Morava  
**ELEMENT SELECT** s.area  
**FROM** s in states  
**WHERE** s.sname = „Morava“
- **Vytváření hodnot PDT**
- **Podpora SŘBD**



# Spojení textu a grafického vstupu

- Druhou možností je vložení speciálního klíčového slova do dotazu, které vyvolá uživatelskou interakci na grafické úrovni
  - **SELECT** sname **FROM** cities **WHERE** center inside **PICK**



# Základní operace

- **Selekce** OK
- **Spojení** OK
- **Aplikace funkcí** OK
- **Množinové operace** *problém*
  - nekonzistentní se **SELECT-FROM-WHERE**



## Příklad 1

- **SELECT \***  
**FROM** rivers  
**WHERE** route intersects Window
- **SELECT** cname, sname  
**FROM** cities, states  
**WHERE** center inside area



## Příklad 2

- **SELECT**

    rname, intersection(route, Morava),  
    length(intersection(route, Morava))

**FROM** rivers

**WHERE** route intersects Morava



# Způsob prezentace výsledků

- **Popis vzhledu výsledku součástí dotazovacího jazyka**
- **Oddělený jazyk**
- **Definován v rámci GUI**
- **Alespoň částečné možnosti pro tvorbu dotazů**





# UKLÁDÁNÍ V SŘBD



# Datové struktury a algoritmy

- **Problém**

- implementace prostorové algebry takovým způsobem, že do dotazovacího systému lze „lehce“ začlenit
  - reprezentaci PDT
  - algoritmy/operace nad PDT
- predikáty v množinově orientovaných DML
  - prostorová selekce, spojení, množinové operace



# Reprezentace hodnot PDT

- **Kompatibilní ze dvou hledisek**
  - **Hledisko SŘBD**
    - PDT stejné jako hodnoty jiných typů
    - různorodá (hodně velká) velikost dat
    - hodnoty na disku (i více stránek)
    - možnost natažení do operační paměti
    - základní operace specializované dle typu



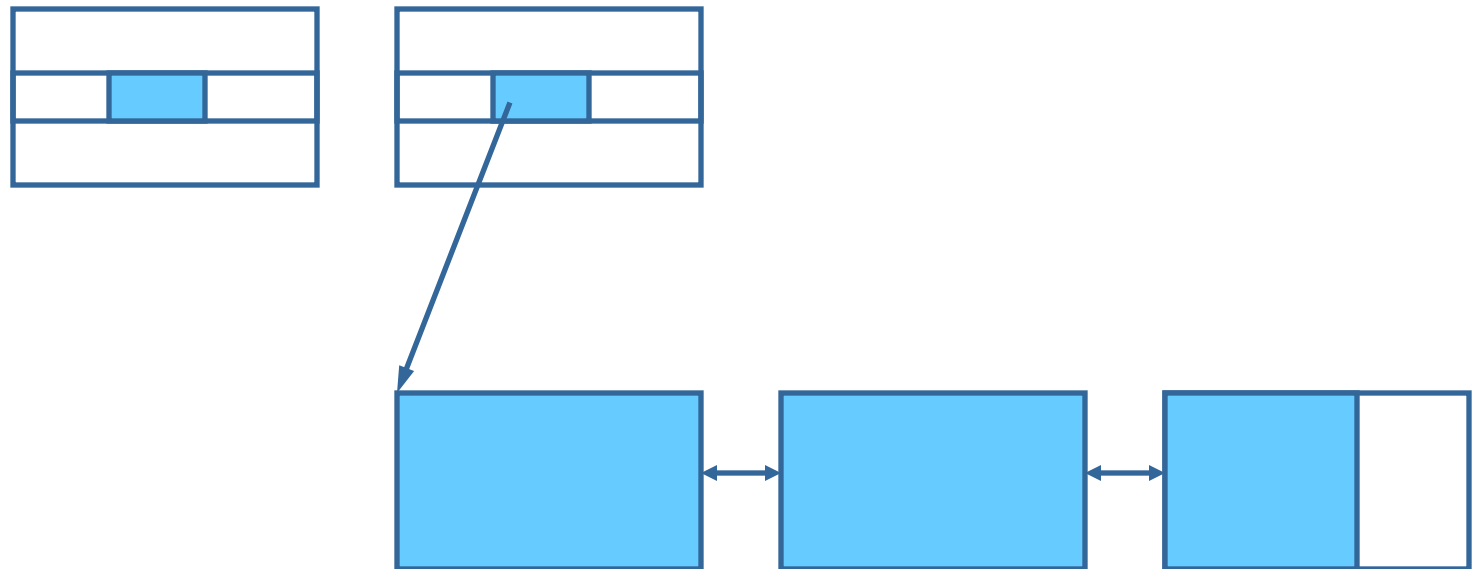
## ... kompatibilita hledisek

### – Hledisko prostorové algebry

- hodnoty odpovídají ADT programovacího jazyka
- jde o nějakou datovou strukturu (typicky složitou)
- podpora algoritmů numerické geometrie
- neoptimalizovány pro jeden algoritmus, ale tak, aby podporovaly stejně všechny

# Podpora SŘBD – velká data


- **stránkované PDT by měly být uloženy podobným způsobem jako běžné velké DT**





# Podpora SŘBD – separace dat

- „info“ a geometrická data odděleny
    - info: konstantní velikost, malá
    - geometrie: různorodá velikost, velká
      - type polygon = pointer to record
        - area: real;
        - perimeter: real;
        - bbox: rectangle;
        - num\_of\_vertices: integer;
        - vertices: array[1..1000000] of point
- info
- geometrie
- end

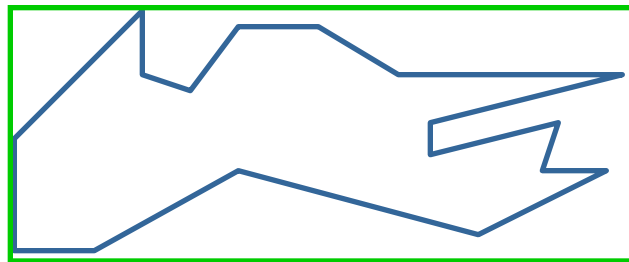


# Podpora SŘBD – znázornění dat

- **Obecné operace**
  - Data  $\leftrightarrow$   
Interní reprezentace  $\leftrightarrow$   
Uživatelské rozhraní (textový, grafický)
- **Prostorové datové typy**
  - Interní reprezentace  $\rightarrow$  Aproximace

# Podpora prostorové algebry

- **Reprezentace obsahuje**
  - **PSS (plane sweep sequence)**
    - statická negeometrická data
  - **aproximace**
  - **uložené hodnoty unárodních funkcí (plocha, průměr, střed, ...)**





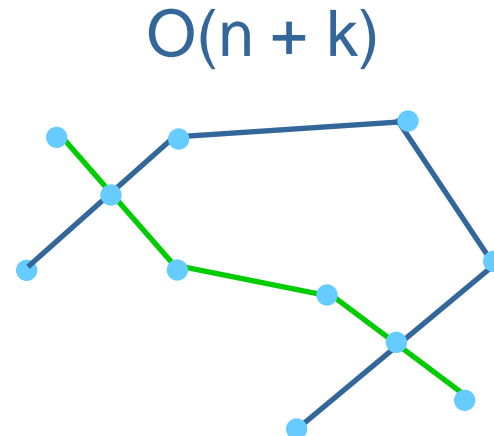
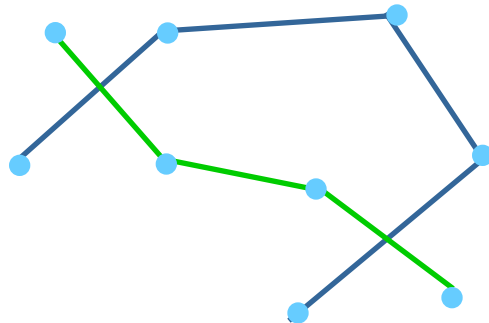


# Implementace operací nad PDT

- prvotní práce s aproximacemi
- využívání uložených předpočítaných konstantních funkčních hodnot
- algoritmy z numerické geometrie
- *Použití některých efektivních algoritmů je silně omezeno typem aplikace*

# PDT pro deskriptory (realms)

- Nikdy se nepočítají místa průniku
  - známy dopředu, zaneseny v obou objektech
- Př.
  - lines x lines → points (průnik)
  - $O(n \log n + k)$





## 5. Prostorové databáze IV



Úvod, principy, problémy

Bodové algoritmy stromové a mřížkové (hash)

Algoritmy pro plošné objekty

# INDEXOVÁNÍ

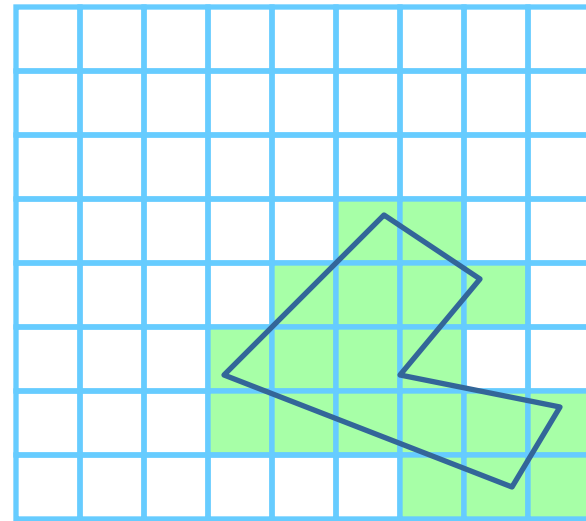
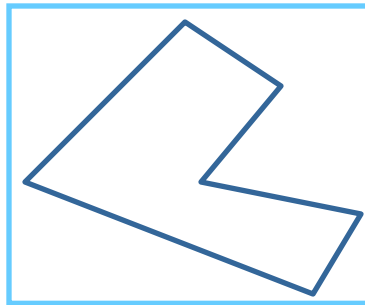


# Indexování u prostorových DB

- **Podpora pro**
  - prostorovou selekci
  - prostorové spojení (join)
  - *i další operace*
- **Organizace „prostoru“ a objektů**
  - zvláštní „externí“ datové struktury
  - mapování objektů do 1D prostoru a použití standardních metod (B-strom)

# Základní metodika indexování

- **Použití vhodných aproximací!**
  - spojitá aproximace
  - aproximace sítí (bodů)





# Zvláštní datové struktury

- Obvykle slouží k uložení bodů, nebo obdélníkových objektů
- Typické operace
  - vkládání
  - mazání
  - dotazování
    - existenční
    - jako takové
- *Vždy záleží na aplikaci*



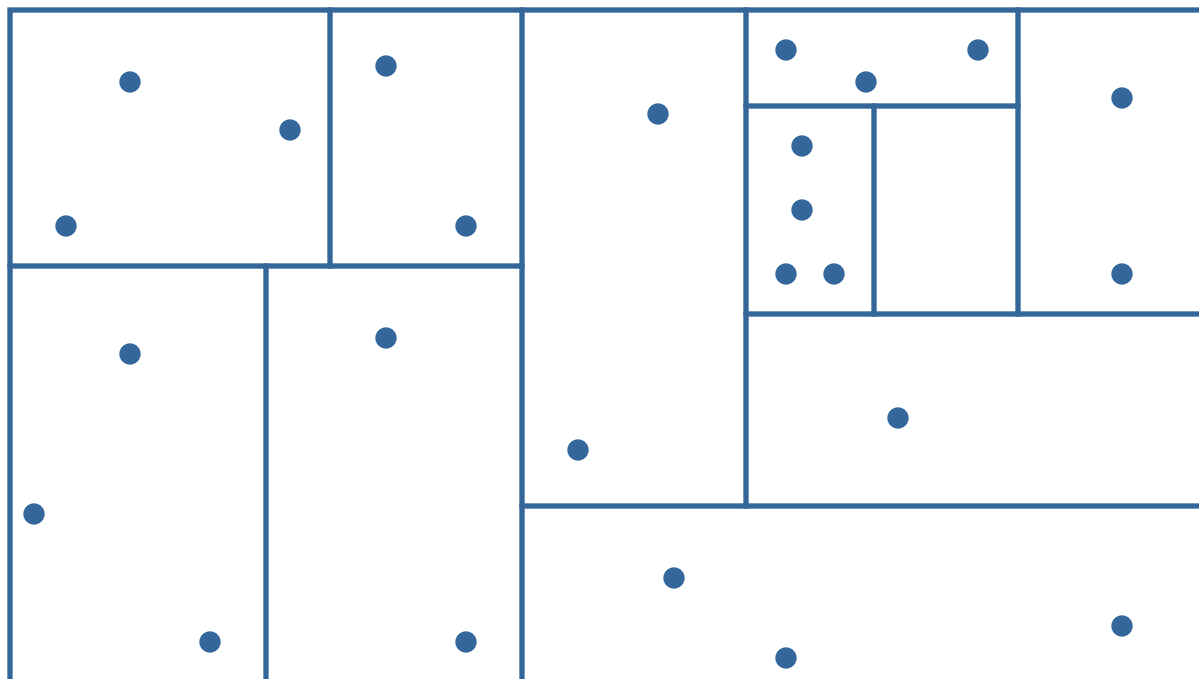
# Dotazování

- velikostní údaje
- nejbližší soused(i)
- určení vzdáleností
- průniky
- obsažení



# Dekompozice prostoru

- Výseky prostoru s danou „kapacitou“





# 1D přístupové metody

- **Lineární hashování**
- **Adaptivní hashování**
- **B-stromy**



# Lineární hashování

- **Prostor  $[A,B)$  dělí na intervaly**
  - $(B-A)/2^k$ , či  $(B-A)/2^{k+1}$ ,  $k \geq 0$
  - odpovídají stránce, oblasti
  - ukazatel  $t$  odděluje už zaplněné intervaly od nezaplněných
  - při vložení může dojít k rozdělení intervalu (jen jednou), i přetečení
  - jakmile  $t$  dosáhne  $B$ , je třeba přerozdělit celý soubor



# Adaptivní hashování

- jako lineární, ale intervaly jsou nazývány *buňkami*
- přetokovou oblast řeší adresář
  - obsahuje index každé buňky
- jestliže výsek přesáhne kapacitu, dělí se všechny buňky 2
  - (na začátku buňka=výsek)
- po dělení více buněk pro výsek
  - datové regiony

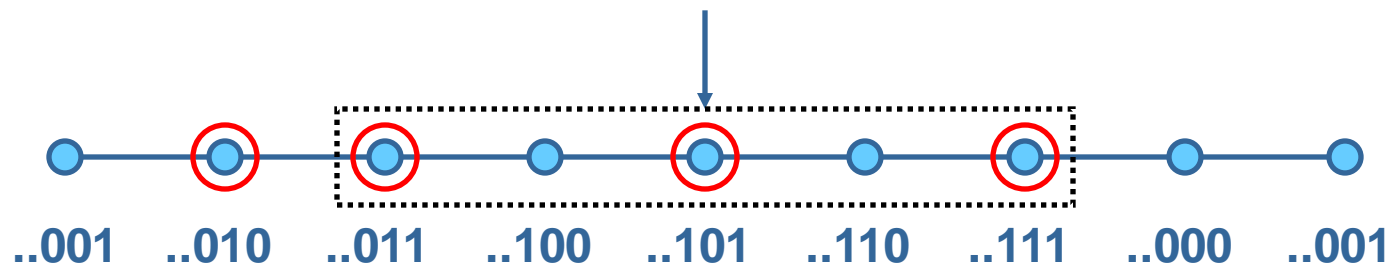


# B-stromy

- hierarchická organizace
- vyvážený (vnořené intervaly)
- uzel ~ stránka ~ interval
- listy - ukazatele na data
- horní/dolní mez na počet následníků
  
- asi nejadaptabilnější
- pro lineární rozložení je hashování lepší

# Vyhledání/vložení dat

- **1D data**
  - **Aproximace posloupností celých čísel**
    - následník/předchůdce
    - okolí
    - vyspělé algoritmy





## 2D, 3D, ... data

- **Problém**

- chybí uspořádání na více rozměrech
  - i když mapování do N, Z, Q existuje
- transformace do 1D
  - nesplňuje vlastnosti sousednosti
  - pokud je model přípustný, složité transformace dotazů



# Základní algoritmy pro n-D

- **K-D-Tree**
- **BSP Tree**
- **Quad-Tree**
  - leží v základu dalších algoritmů
  - neřeší problematiku paměti

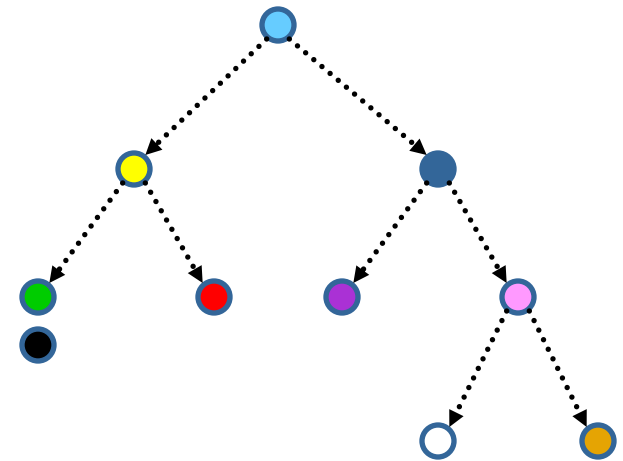
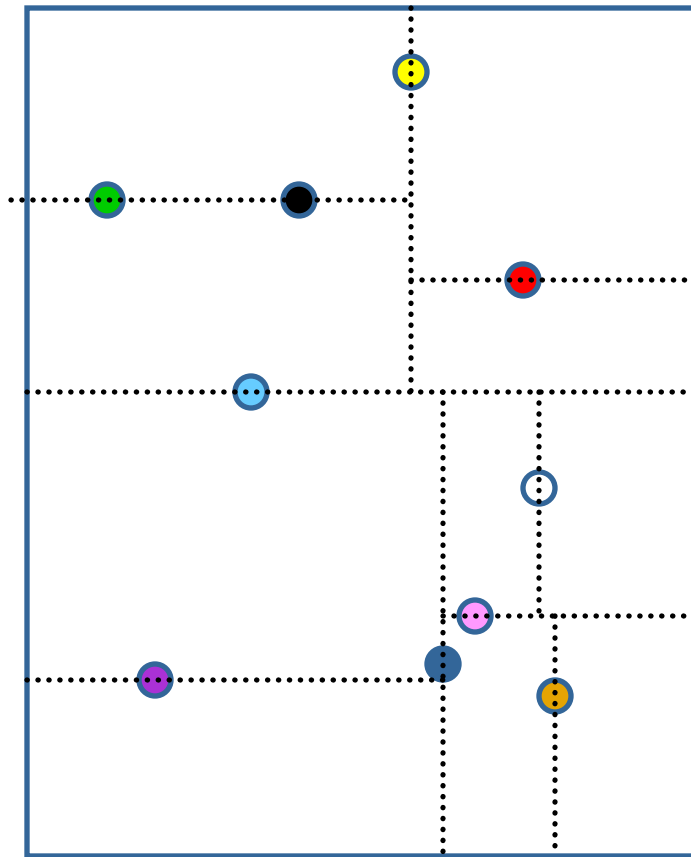




# K-D-Tree

- **Prostor je dělen hyper-plochami na nejvyšší úrovni (rovnoběžné s osami)**
- **Každá plocha musí obsahovat alespoň jeden bod dat**
- **Vkládání, hledání - OK**
- **Mazání - problém**
- **Jen body**

# K-D-Tree ukázka





# Modifikace K-D-Tree

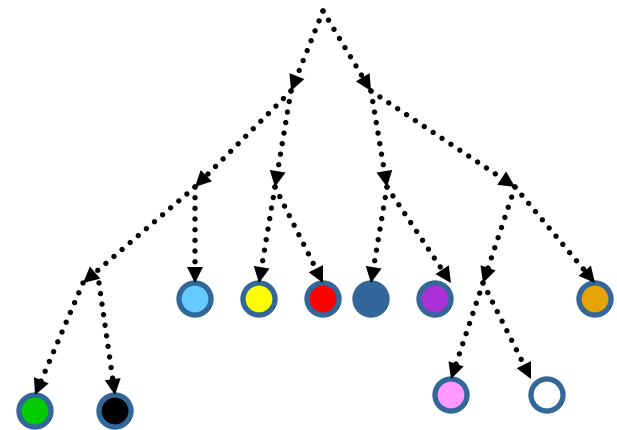
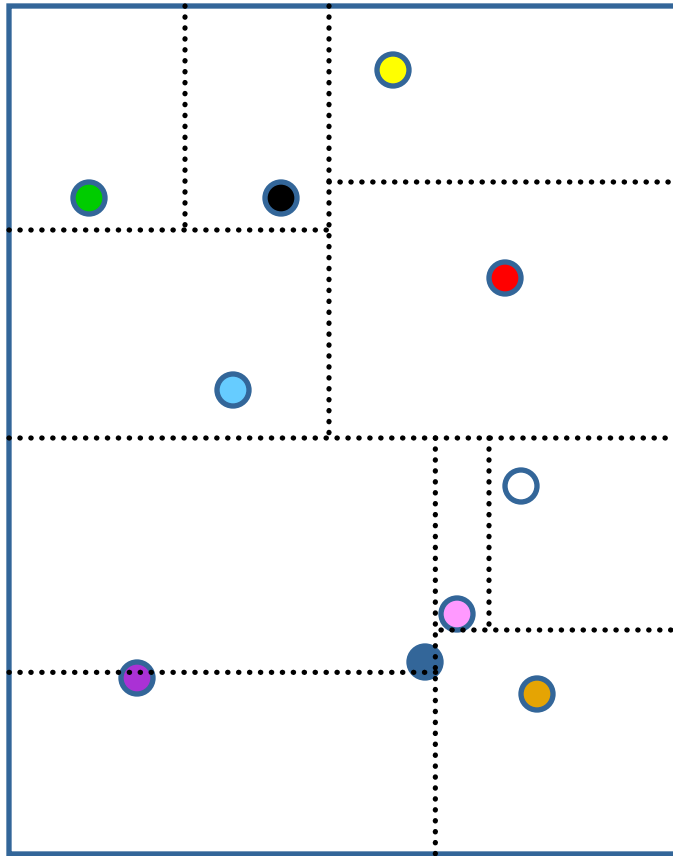
- **Adaptivní K-D-Tree**
  - odstraňuje nevýhodu závislosti na pořadí vkládání
  - data roztroušena po celém stromu
- **Bin-Tree**
  - rekurzivně dělí podprostor na hyperkrychle (shodné velikosti) až každá obsahuje maximálně jeden bod



# Adaptivní K-D-Tree

- Dělení hyper-plochami probíhá tak, aby na každé straně zbývalo zhruba stejně bodů
- I když stále rovnoběžné s osami, hyper-plochy nemusejí obsahovat bod -> data do listů (ve výsledném podprostoru jen jeden bod)
- Vhodný pro statická data

# Adaptivní K-D-Tree - ukázka

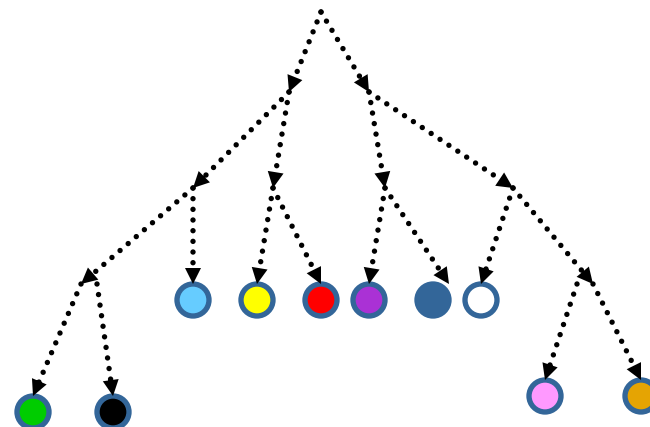
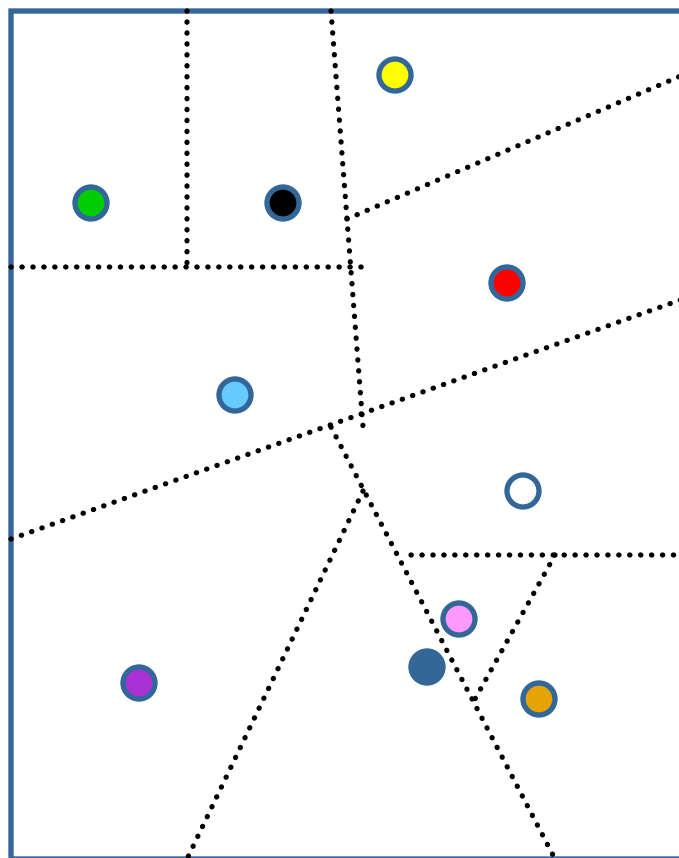




# BSP Tree

- **Binary Space Partitioning**
- **Jako adaptivní K-D Tree**
- **Dělicí hyper-plochy nejsou (nutně) rovnoběžné s osami**
- **Dělení tak dlouho, dokud počet bodů v podprostoru neklesne pod danou hodnotu**
- **Neadaptivní, vyšší nárok na paměť**

# BSP-Tree ukázka



*hloubka!*

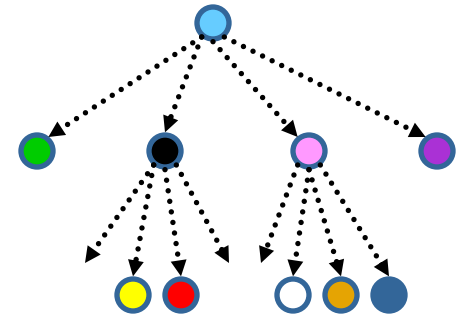
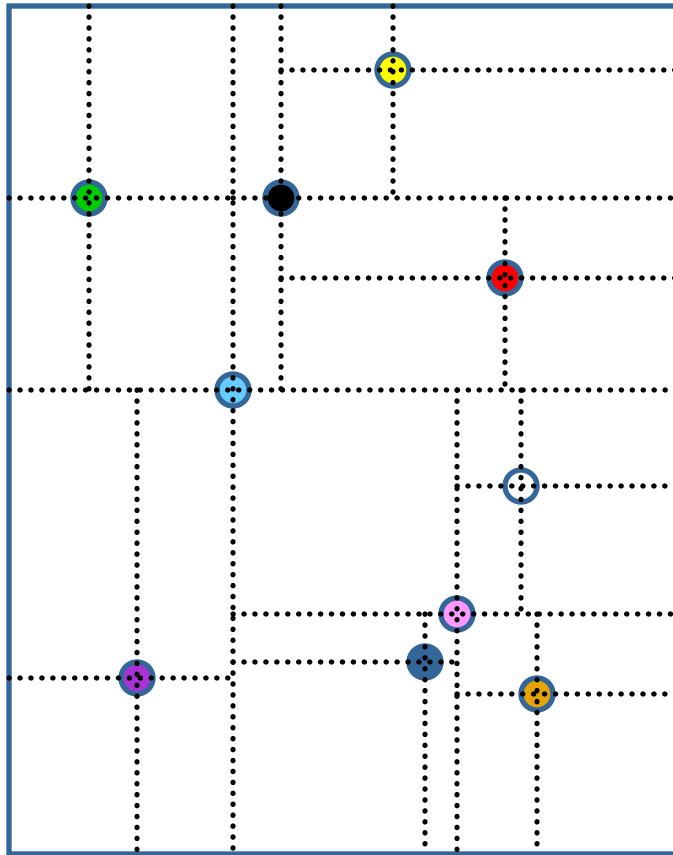


# Quad-Tree

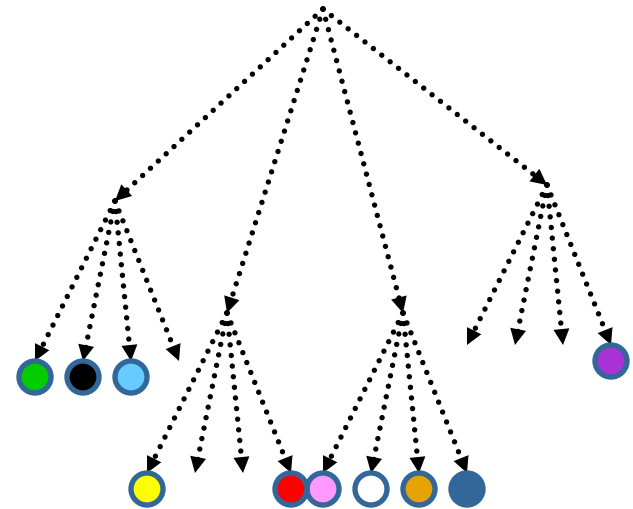
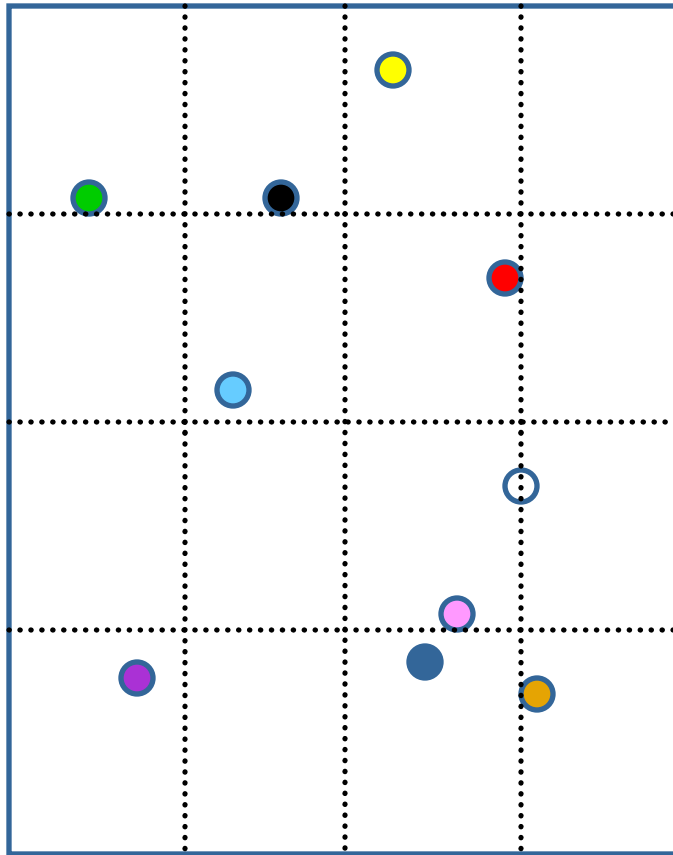
- Shodné s K-D-Tree, ale dělí na  $2^n$  pod-stromů
- Pod-stromy nikoliv nutně shodné
- Dělení do daného počtu elementů na list (i nula)
- Varianta *point a region* quad-tree



# Point Quad Tree - ukázka



# Region Quad Tree



*PM-quad-tree - regiony*



# Přístup k bodům - hashování

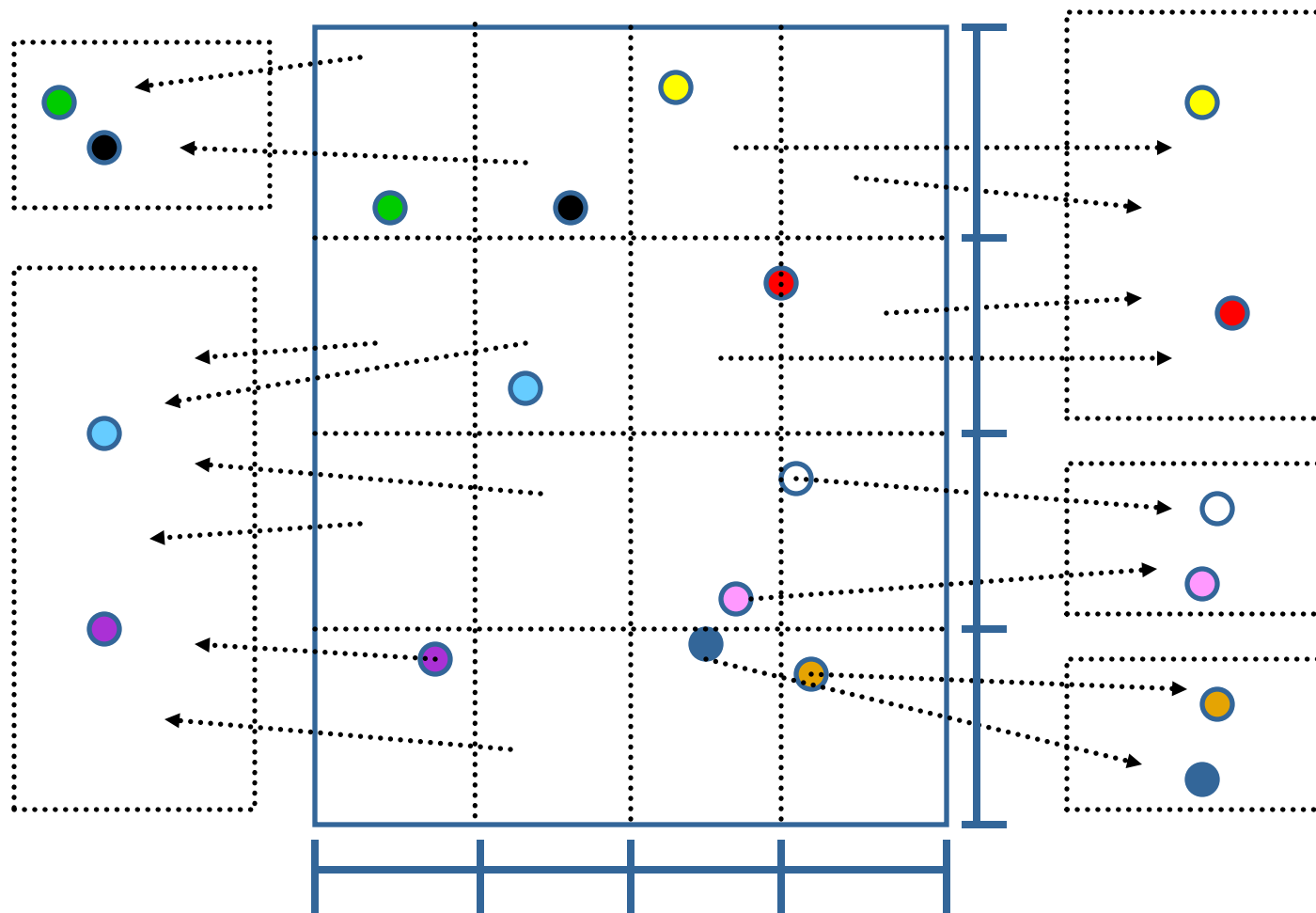
- **Adaptivní hashování**
  - Grid File
  - EXCELL
  - Two-Level Grid File
  - BANG File
  - Twin Grid File
- **Lineární hashování**
- **Hybrid**
  - Buddy Tree



# Grid File

- **Prostor pokryt n-rozměrnou mřížkou**
  - nikoliv nutně pravidelná
- **Výsledné buňky různorodé**
- **Adresář řadí každou buňku (i více) k datové jednotce (bucket)**
- **Adresář velký - na disku**
  - mřížka na disku (jen 2 přístupy na disk)
- **Využití prostoru kolem 69%**

# Grid File - ukázka





# Grid File - vkládání/mazání

- **Vkládání může způsobit přetečení jednotky (není lokální)**
  - rozdělovací hyper-plocha
  - možná distribuce skrze strukturu - nárůst adresáře
- **Mazání (není lokální)**
  - odstranění hyper-plochy je třeba prověřit
  - odstranění datové jednotky



# EXCELL

- **Jako Grid File**
- **Dělí na jednotky stejné velikosti**
- **Dělení je plošné**
  - velký adresář
  - později hierarchie
  - přetokové stránky

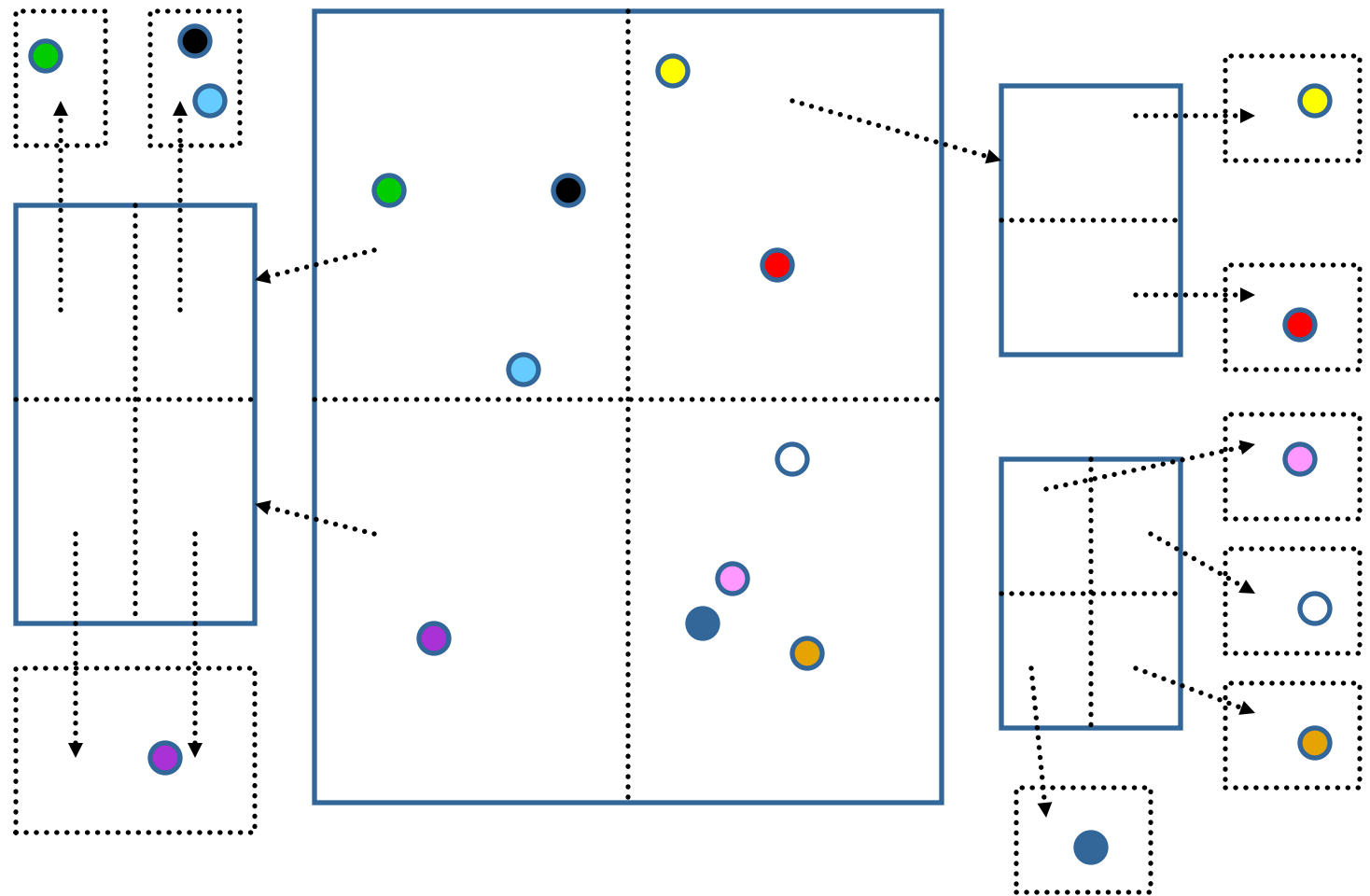


# Two-Level Grid File

- **Mřížka druhé úrovně se používá pro správu adresářů**
  - kořenový adresář
  - pod-adresář
- **Změny jsou často lokální**
  - problematika však přesto není zcela uspokojivě řešena



# Two-Level Grid File - ukázka

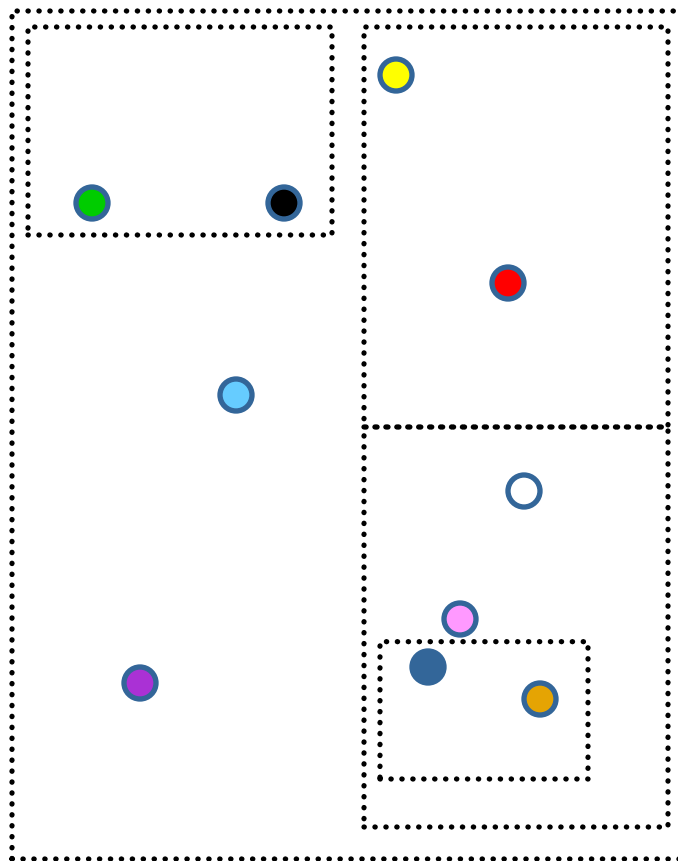




# BANG File

- **Interpolační Grid File je základem**
  - „řeší“ exponenciální růst adresáře
  - buňka = datová jednotka
- **Balanced And Nested Grid File**
  - jako Grid File
  - datové jednotky se překrývají (!)
  - nemusí mít tvar hyper-krychle
  - datové jednotky jsou ve vyváženém stromě

# BANG File - ukázka

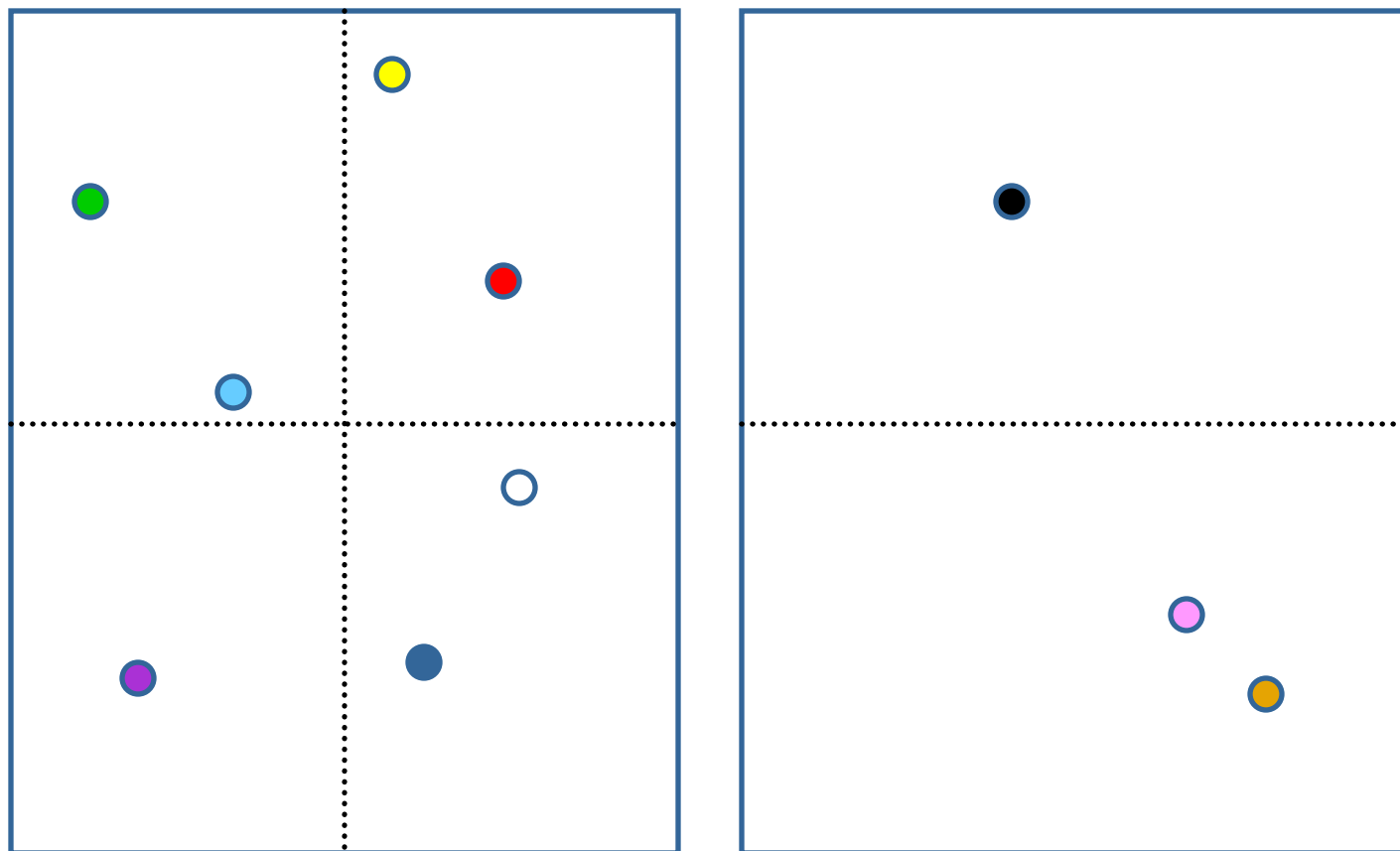




# Twin Grid File

- **Zdvojená struktura Grid File**
  - zlepšení využití prostoru
  - vztah není hierarchický
  - rovnoměrně rozložená data
  - využití prostoru až 90% bez „zpomalení“
  - jeden ze souborů je „primární“
  - druhý je „přetokový“

# Twin Grid File - ukázka





# Vícerozměrné lineární hashování

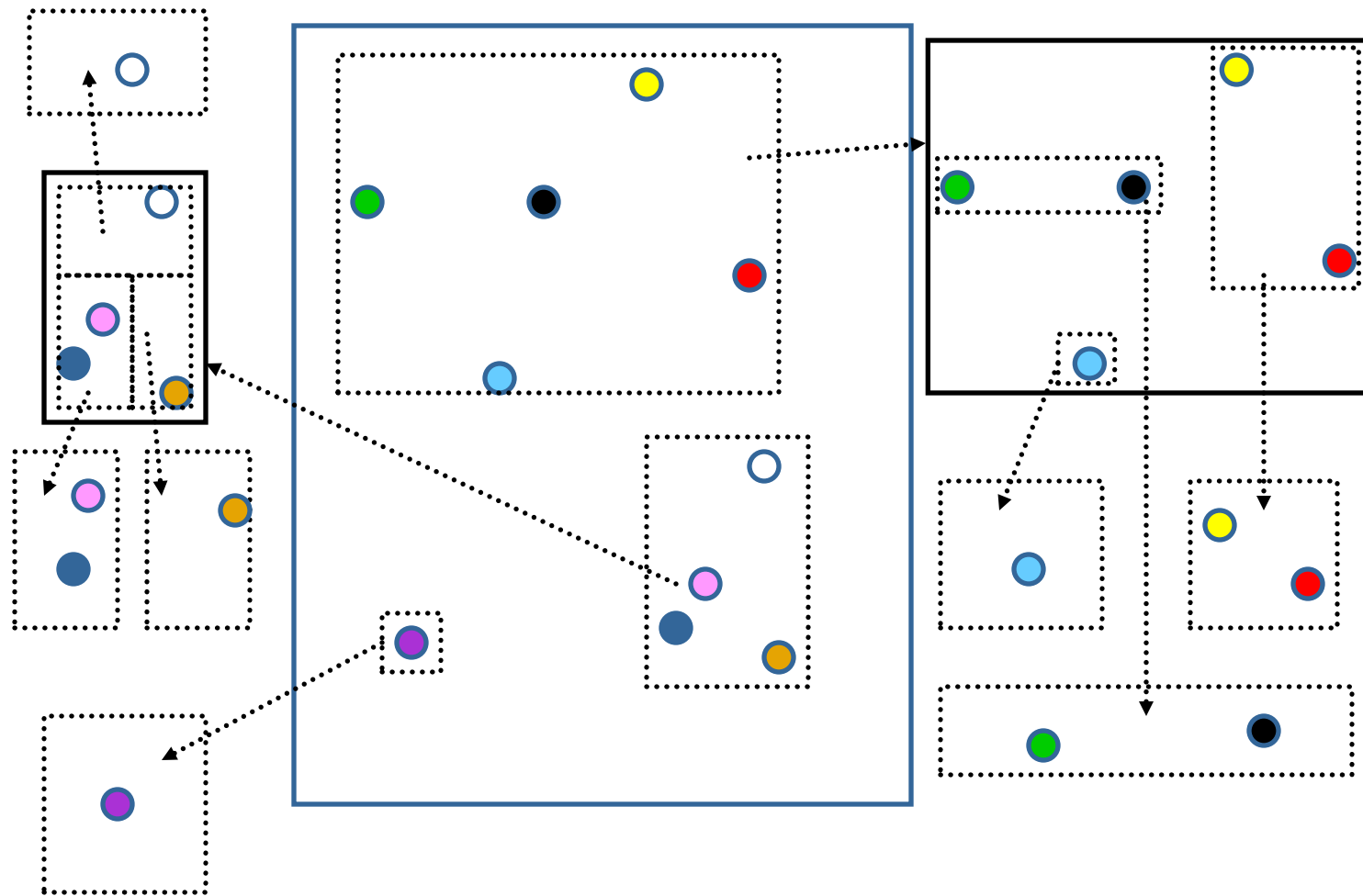
- malé, nebo žádné adresáře (do paměti)
- **MOLHPE**
  - ukazatele pro růst dat
  - datové jednotky stejných rozměrů
  - nevhodné pro nelineární distribuci
- **Quantile hashing**
  - nerovnoměrně distribuovaná data  
„zrovnoměrnňuje“ - jinak MOLHPE
- **Z-hashing**
  - níže



# Buddy Tree

- **Adresář je stromová struktura**
  - minimálně dvě položky (není vyváženo)
  - jeden ukazatel na adresář (stránku); lineární
- **Strom je rozdělován rekurzivně, dělí se hyper-plochami rovnoběžnými s osami**
- **Ve vnitřních uzlech se však prostor omezí na MBB vnitřních bodů - selektivita**

# Buddy Tree - ukázka







# Přístup k bodům - stromy

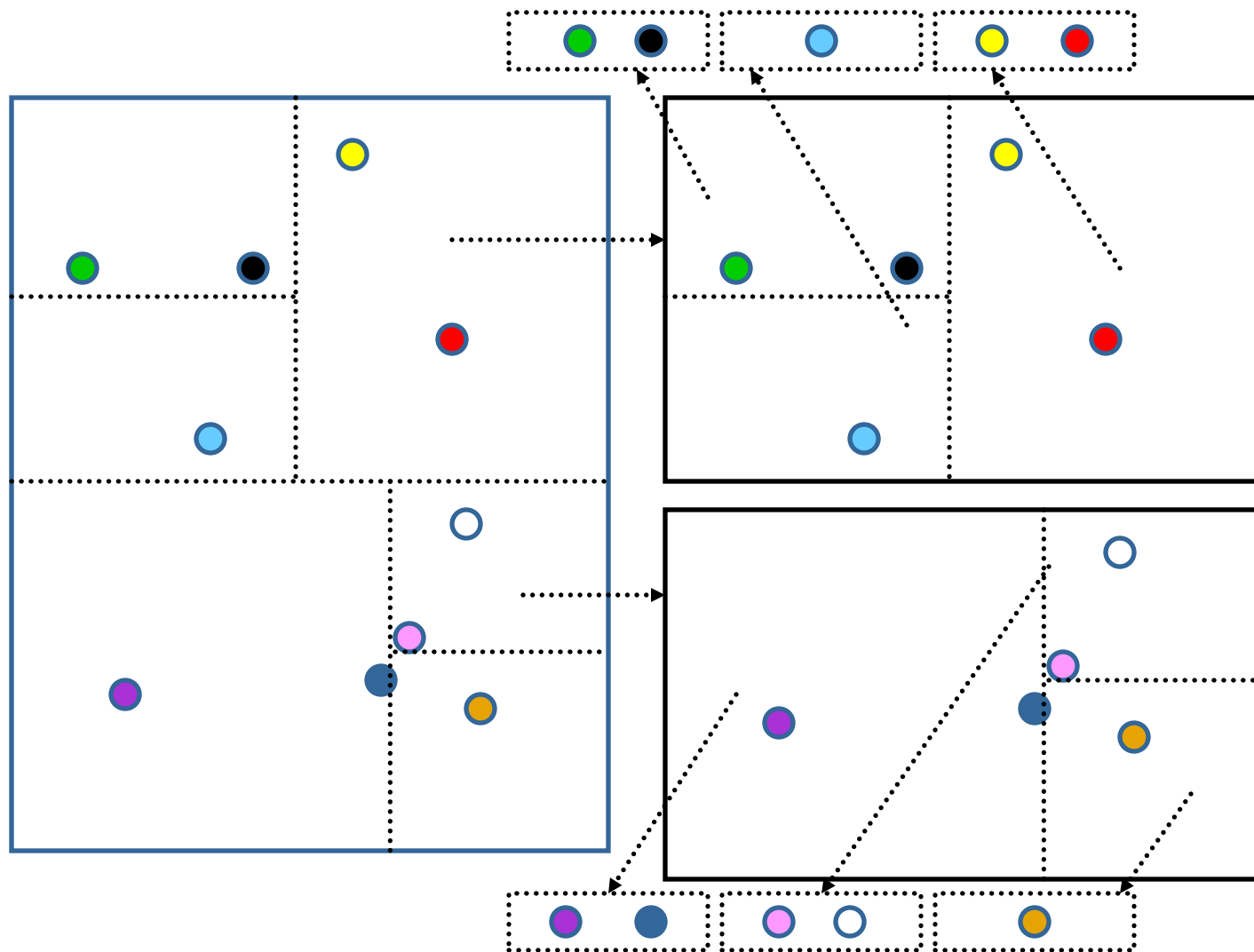
- **K-D-B-Tree**
- **hB-Tree**
- **LSD Tree**



# K-D-B-Tree

- **K-D Tree + B-Tree**
- **Adaptivní K-D Tree**
- **Balancován (B-Tree)**
- **Vkládání může způsobit rozdělení**
  - heuristiky na optimální rozdělení
  - propagace stromem (? využití paměti)
- **Mazání**
  - slučování při podtečení

# K-D-B-Tree - ukázka





# hB-Tree

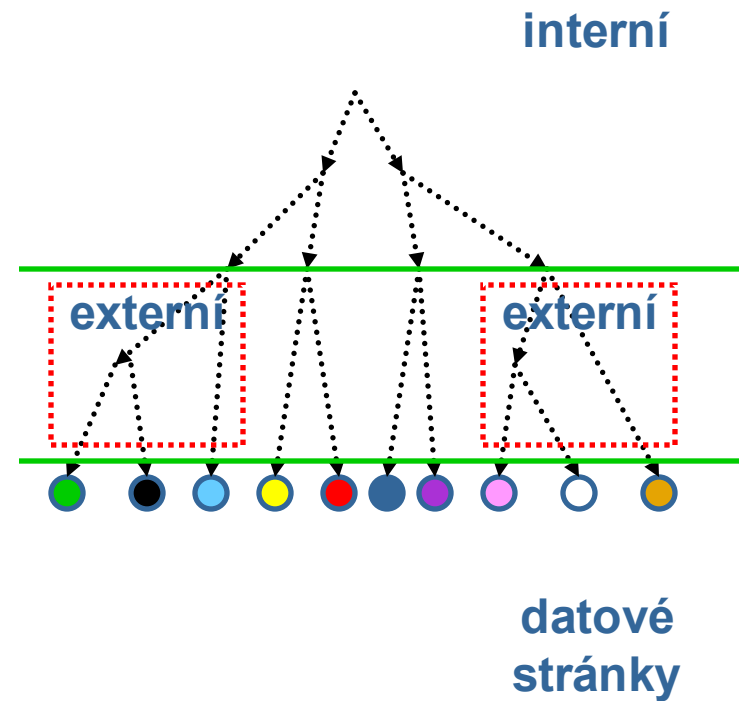
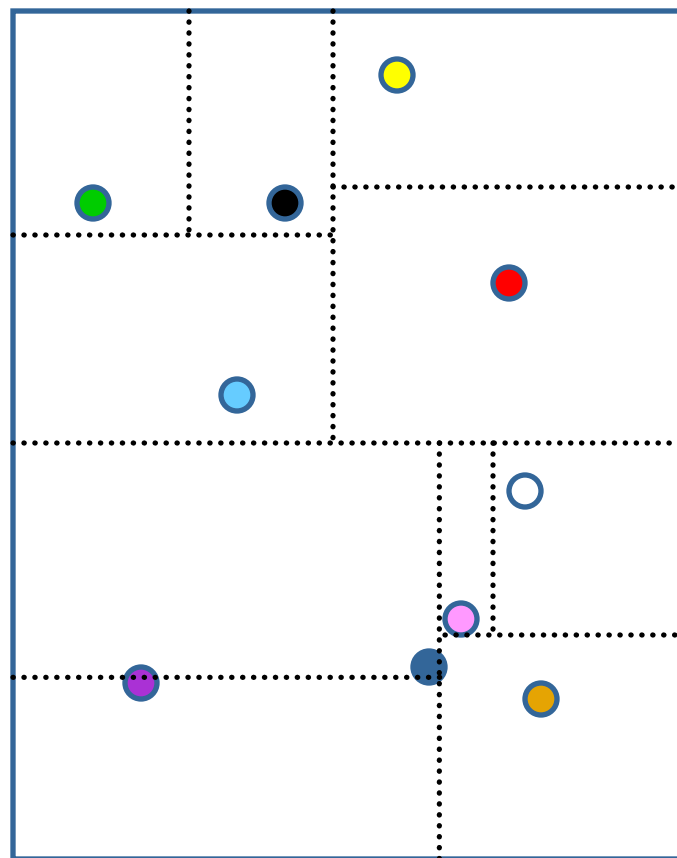
- **Holey brick tree**
- **Dělení uzlu je více-atributové**
  - „Fraktální struktura“
  - BANG File
- **DAG (!)**
- **Paralelní verze**



# LSD Tree

- **Local split decision**
- **Nejen pro prostorová data**
- **Adaptivní K-D Tree**
- **Výškově vyvážený strom**
- **Externí adresářová stránka**

# LSD Tree - ukázka





# Indexování prostorových objektů

- **Hlavní metody**
  - Transformace - mapování
  - Překrývání - vymezení
  - Ořezávání - duplikace objektů
  - *Křivky vyplňující prostor*



# Transformace - mapování

- **Objekty v n-D prostoru jsou body v  $k \cdot n$ -D prostoru**
  - obdélník ve 2D je bod ve 4D (např.)
- **Optimismus  $\sim$  > vystřízlivění**
  - některé dotazy jsou nerealizovatelné
  - mapování je složité (nemožné)
  - interpretace výsledku





# Překrývání

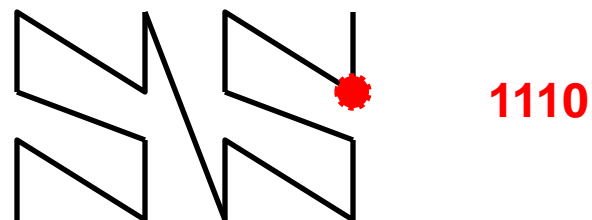
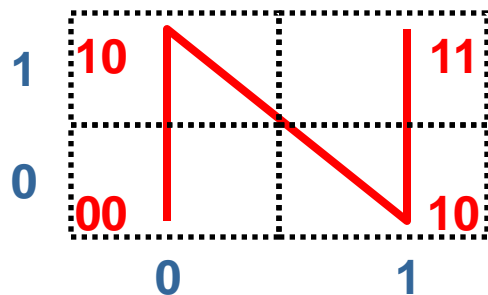
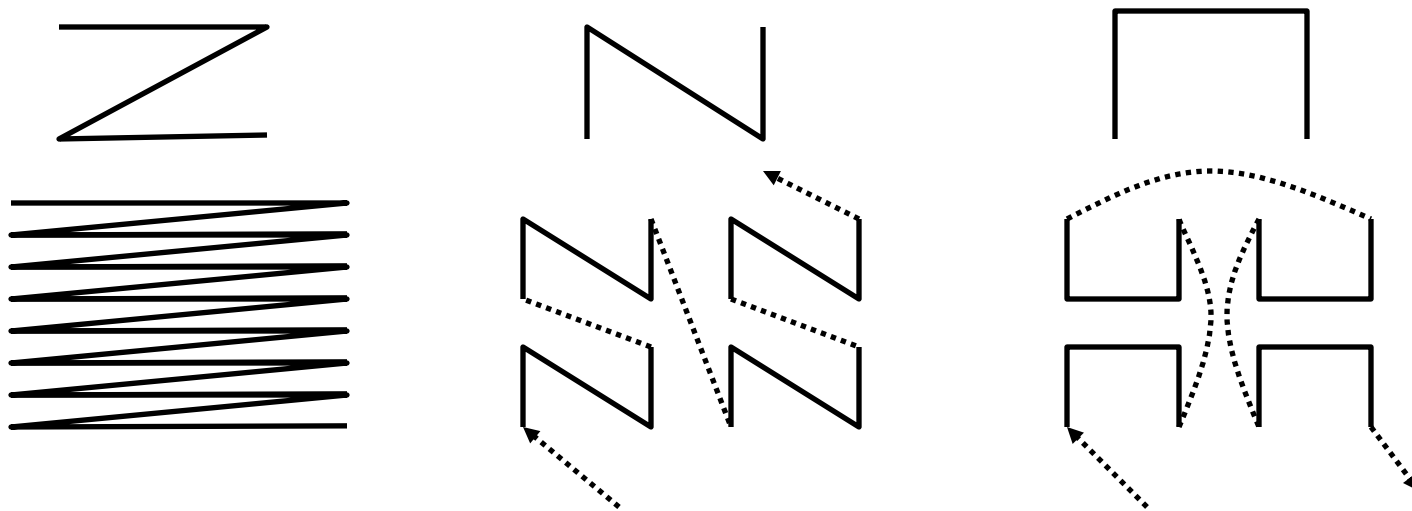
- **Datové jednotky se svými hranicemi překrývají**
- **Často algoritmy zůstávají stejné, jen se počet prohledávaných cest zvětšuje**



# Ořezávání

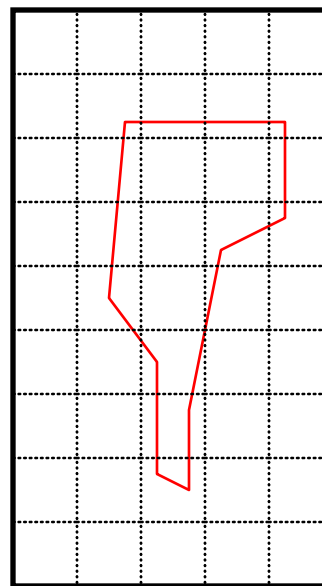
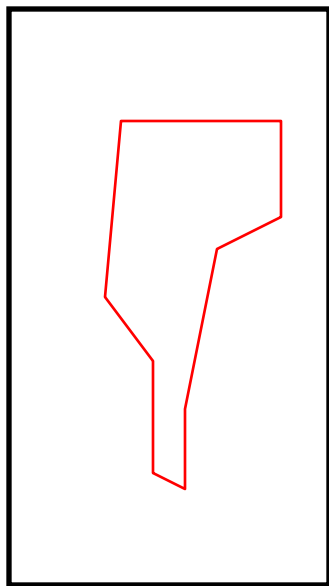
- **Není povoleno překrytí**
- **Dělení objektů**
- **Rozšiřování prostoru datových jednotek**
  - **Deadlock**
- **Dělení datových jednotek**

# Křivky vyplňující prostor

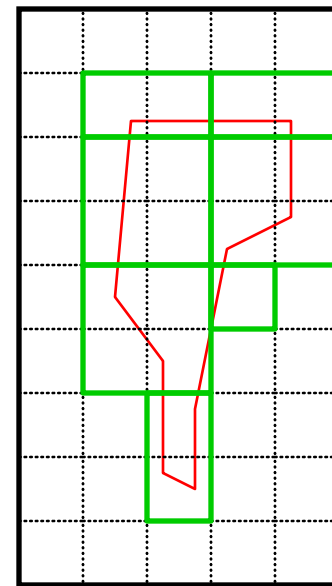


*i Hilbertova křivka*

# Z-ordering



000 001 010 011 100  
00 01  
0

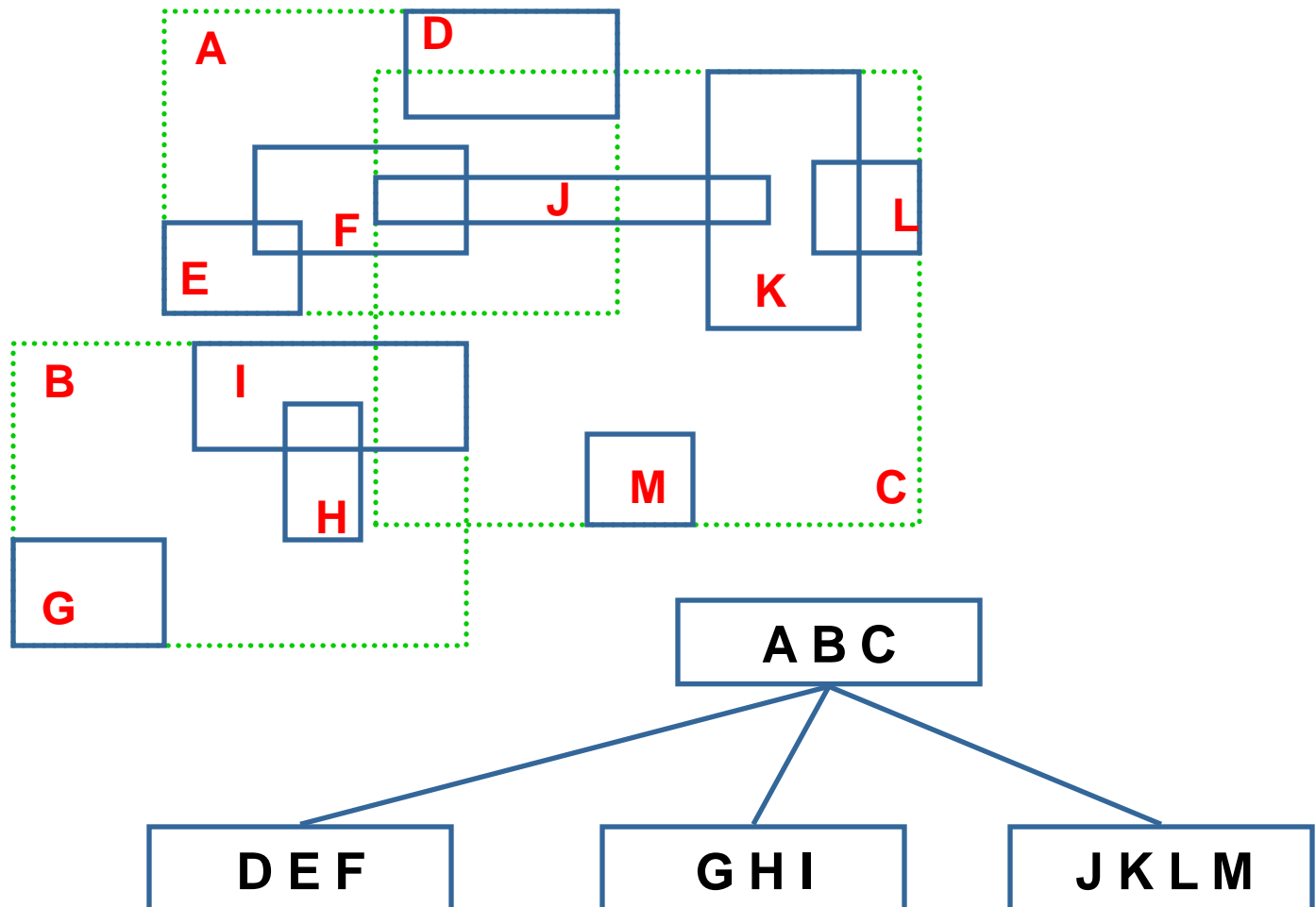




# Algoritmy

- **R-Tree a z něj odvozené**
- **P-Tree a podobné**
- **Další hierarchické metody**
  - Rozšířené K-D Tree
  - SKD Tree
  - GBD Tree
- **Hashovací struktury**
  - PLOP, Multi-Layer Grid File, R-File

# R-Tree





## R-Tree: PointQuery

- **PointQuery(P: point): set of oid**  
**begin**  
    **result = *empty set***  
    **SL = RtreeTraversal(root, P)**  
     **$\forall$  L in SL:**  
         **$\forall$  e in L:**  
            **if (P *in* e.mbb) result += e.oid**  
    **return result**  
**end**



# R-Tree: RtreeTraversal

- **RtreeTraversal(node:Page, P:point):**  
    **set of leaves**

**begin**

**result = empty set**

**N = ReadPage(node)**

**if (N is leaf) return N**

**$\forall$  e in N:**

**if (P in e.dr) then result +=**

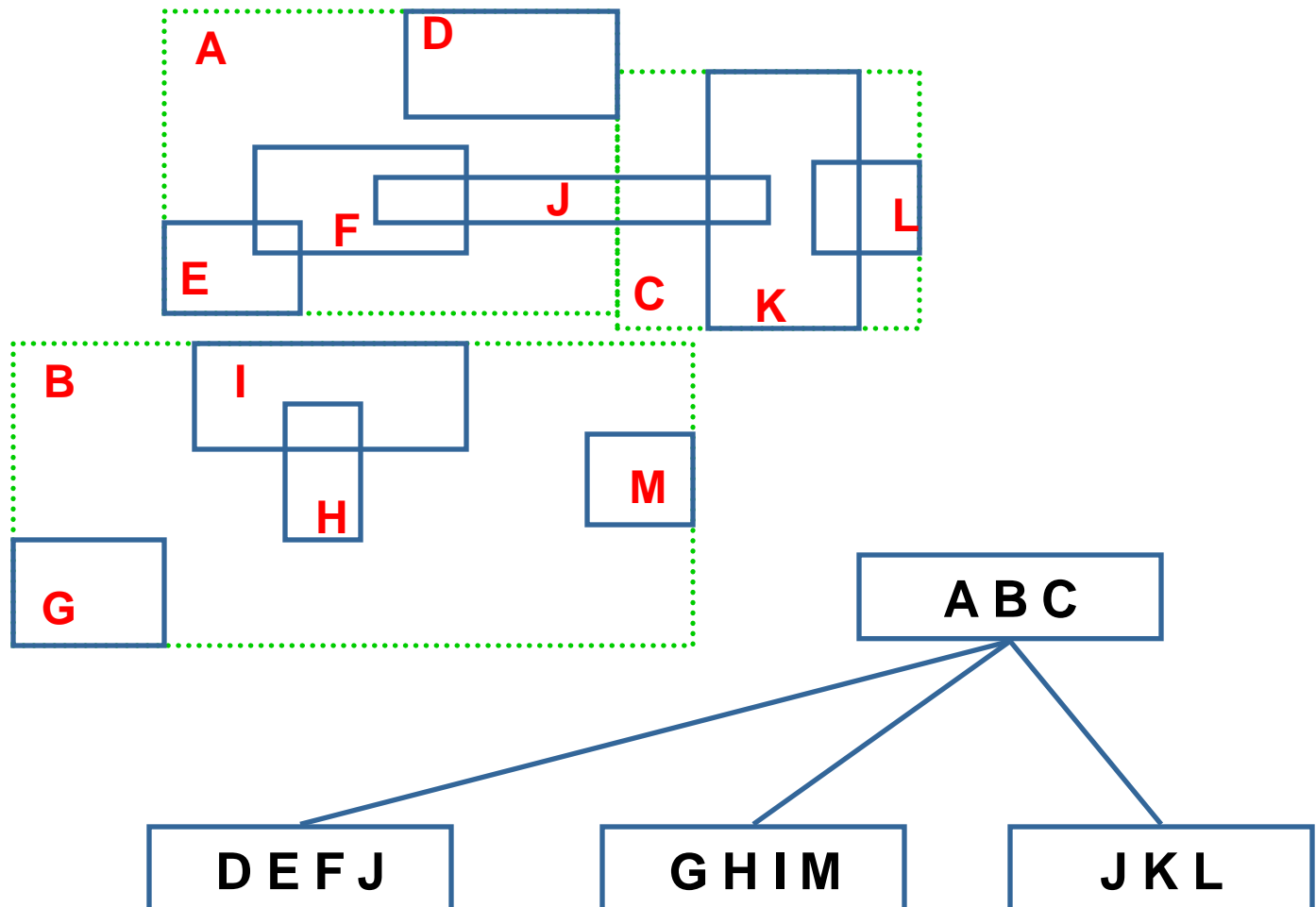
**RtreeTraversal(e.node,P)**

**return result**

**end**

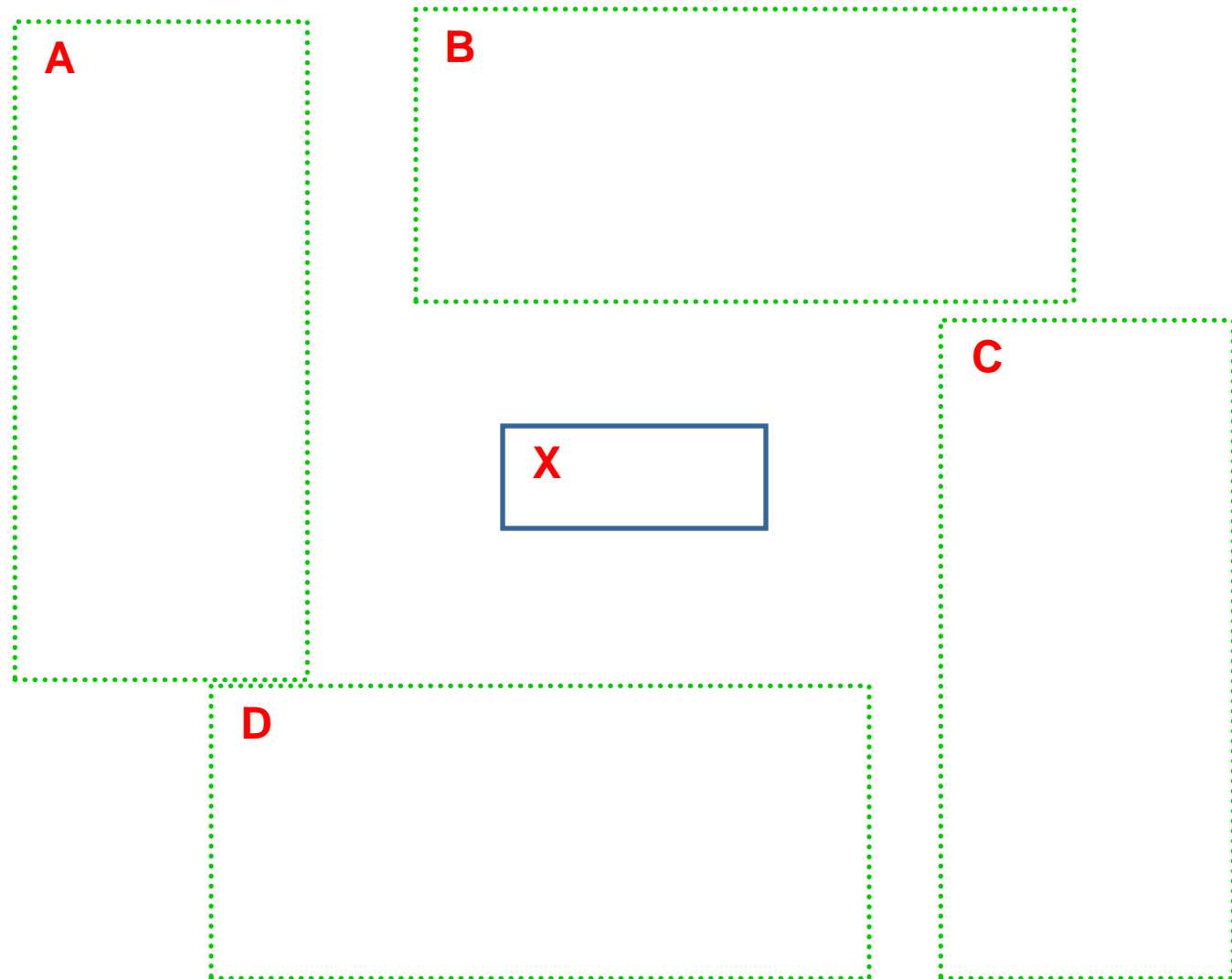


# R<sup>+</sup>-Tree

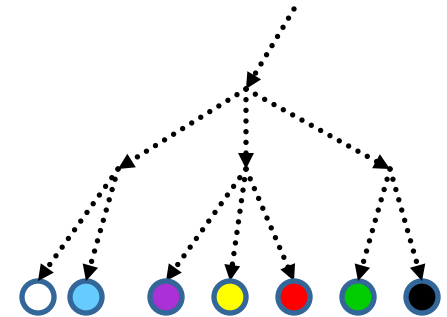
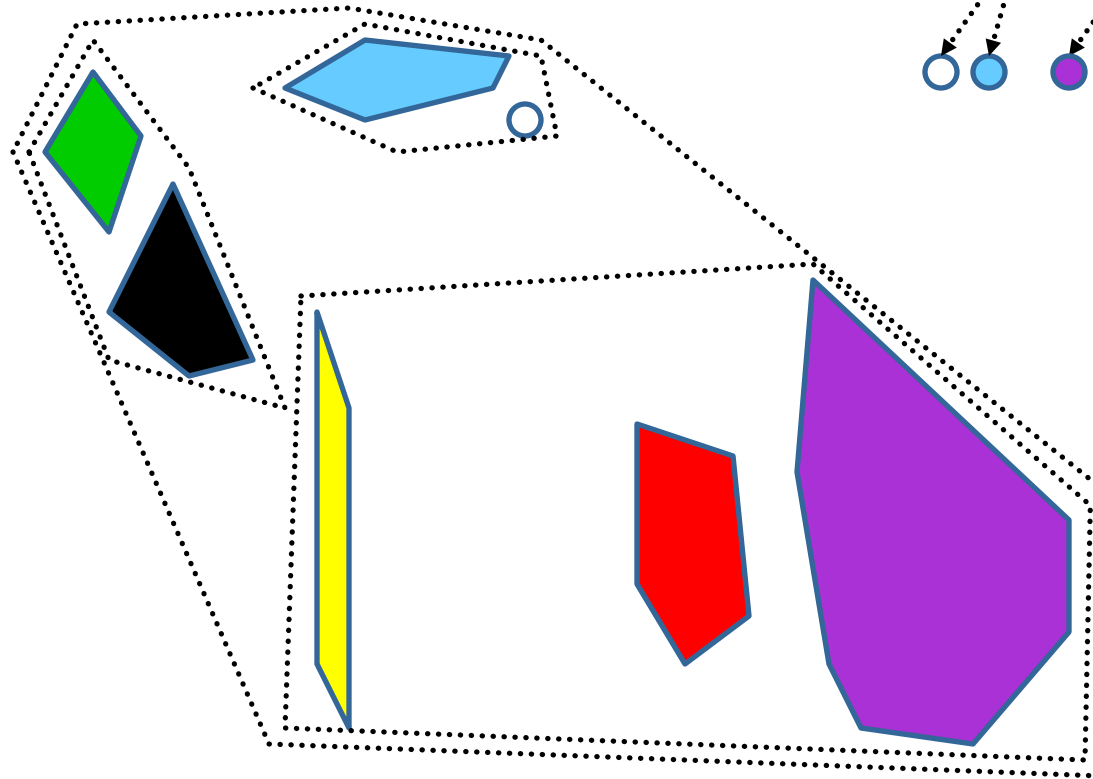


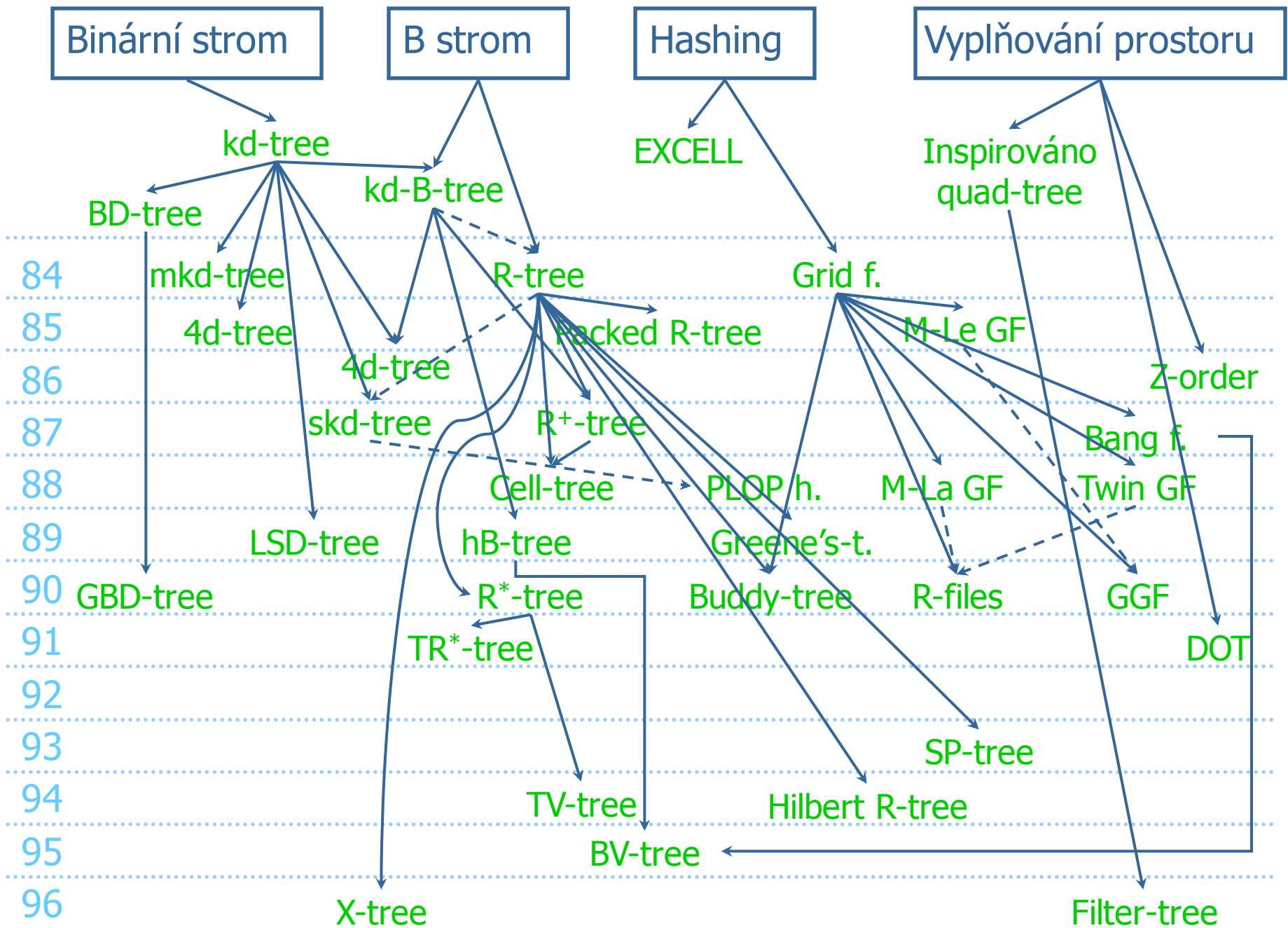


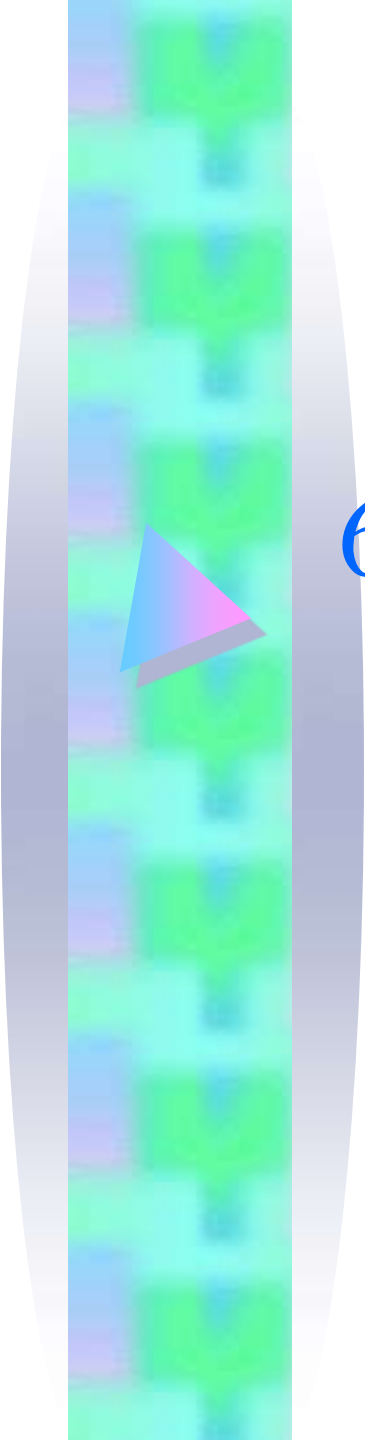
# R<sup>+</sup>-Tree Deadlock



# P-Tree (Schiwietz)







## 6. Temporální databáze - úvod, modely, formalismy



# Temporální databáze

- **Databáze s časovou dimenzí**
  - **Záznamy z řady oblastí**
    - bankovníctví/finančnictví
    - osobní
    - právnické
    - katastrální
    - medicínské
  - **Měření a monitorování**
  - **Historie dat**



# Jak se s časem pracuje

- **uživatelsky definovaný čas**
- **čas platnosti**
- **čas transakce**



# Problematika ukládání času

- **Databáze**
  - body, konečné hodnoty
- **Skutečnost**
  - intervaly
  - periody





# Struktura času - modely

- **Fyzikální hledisko**
  - minulost - omezená ( $14 \pm 4$  miliardy let)
  - budoucnost - několik možností
- **Formální hledisko**
  - časová doména
    - lineárně uspořádaná množina
      - (větvená budoucnost)
    - isomorfní s matematickými doménami



# Matematické domény

- **$N$  (přirozená čísla),  $Z$  (celá čísla)**
  - diskrétní čas
    - (mezi chronony konečný počet jiných)
  - existence následujícího/předchozího okamžiku
- **$Q$  (racionální čísla)**
  - „hustota“ času
- **$R$  (reálná čísla)**
  - spojitý čas



# Některé další domény

- **Nelineární časové domény**
  - větvení času má potenciální databázové aplikace ve správě verzí a systémech řízení toku aplikací
- **Prolínání více časových domén**
  - praktické aplikace
  - řada možných přístupů



# Ontologické časové typy

- **Časový okamžik je bod na reálné časové ose**
  - událost se objevuje v časovém okamžiku
- **Množina časových okamžiků**
- **Časový úsek**
  - doba mezi dvěma okamžiky
  - někdy zvaný interval




# Ontologické časové typy - dok.

- Časový interval je orientované časové trvání. Trvání je časový úsek se známou dobou, ale neznámým začátkem či koncem.
  - Pozitivní (dopředné)
  - Negativní (zpětné)
- Časový (temporální element) je konečné sjednocení časových úseků



# Hodiny

- **Fyzikální proces spojený s měřením sebe sama**
- **Jednotky měřidla jsou chronony hodin**
- **Hodiny dodávají sémantiku časovým razítkům**



# Časy a fakta

- **Čas platnosti** říká, ve kterém období je daný fakt pravdivý v modelovaném světě
- **Čas transakce** faktu je čas, kdy je fakt přítomen v DB a může být získán
- **Tabulky**
  - snímkové, platného času, transakce, obojího času



# Abstraktní temporální databáze

- **Sémantika temporální DB je nezávislá na implementaci**
- **Výhody**
  - deklarativní dotazovací jazyk vysoké úrovně
  - formální báze pro řešení problematiky temporálních databází
    - převoditelnost různých modelů
    - funkční závislosti, normální formy





# Základní stavební kameny

- Časová doména
  - $(T, <)$ , lineárně uspořádaná, neomezená
- Relační databáze
  - $DB(D, \rho)$  nad  $(D, =)$  a schématem  $\rho$
  - $(D, =, r_1, \dots, r_n)$
  - konečná instance  $\rho$  nad  $D$
  - doména neinterpretovaných konstant



# Jak tyto kameny spojit

- **Snímky DB - „Snapshot“**
- **Čas platnosti (razítko)**
- **Pozn.:**
  - Relační databáze popisují světy v každém časovém okamžiku
  - Temporální logika má však zcela opačný záměr (verifikace, specifikace, ...)



# Snímková tabulka

- **SNAPSHOT**
- **Statické dotazy**
- **Možno modifikovat**
- **Analogie**
  - změna jmenovky na dveřích
  - 1990 - asistent
  - 1995 - odborný asistent
  - 2000 - docent

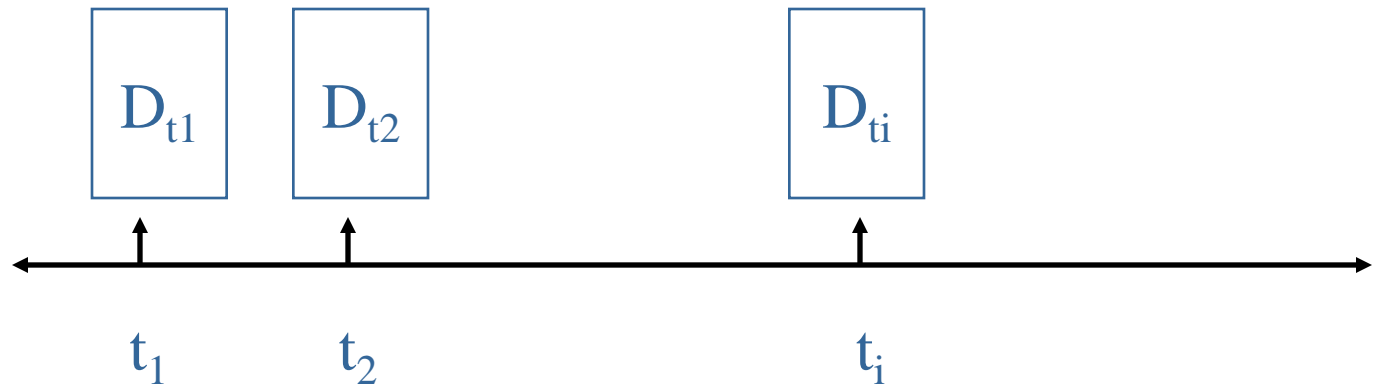


## „Snapshot“

- Je blíže výrokové TL, každá DB popisuje stav světa v konkrétním časovém okamžiku. Relace uspořádání potom definuje tok času.
- Temporální DB je potom funkce typu
  - $T \rightarrow DB(D, \rho)$
  - Datové typy:  $T \rightarrow (D_n \rightarrow \text{bool})$



# Příklad



| Rok  | Snapshot                                    |
|------|---------------------------------------------|
| ...  |                                             |
| 1990 | {Zam(Jan, IBM), Zam(Eva, IBM), Zam(Iva,HP)} |
| 1991 | ...                                         |
| 1992 | ...                                         |
| 1993 | {Zam(Jan, MS), Zam(Eva, IBM), Zam(Iva, HP)} |
| ...  |                                             |



# Co to je historie

- **Při změně údajů v relační databázi se stavy zobrazují na nové stavy**
- **Historie**
  - posloupnost stavů databáze
  - tím pádem takéž „snímkováná“ temporální databáze



## Nevýhoda „snapshot“ modelu

- **Není příliš vhodný pro dotaz typu**
  - $\{t : DB \models \varphi(t)\}$
  - všechny okamžiky, kdy podmínka  $\varphi$  byla v rámci DB platná



# Transakční tabulka

- **Jen připojení dat**
- **Transakce, ROLL-BACK v dotazech**
- **Analogie**
  - výplatní páska





# Tabulka s platnými časy

- **Možno modifikovat**
- **Čas platnosti**
- **Dotazy na historii**
- **Analogie**
  - **přehled zaměstnání na CV**



# Model s časem platnosti - razítka

- Pro každé  $r_i \in \rho$  definujeme relaci  $R_i$  tak, že:
  - $R_i = \{ (t, a_1, \dots, a_k) : (a_1, \dots, a_k) \in r_i \text{ in } D_t \}$
- **Struktura temporální databáze s časovými razítky**
  - $(D, =, T, <, r_1, \dots, r_n)$
  - **konečná instance  $\rho$  nad  $D, T$**
  - **časová doména**
  - **konečná instance  $\rho$  nad  $D$**



## „Výhoda“ modelu s razítky

- **Dotaz na časové okamžiky je realizovatelný**
  - čas je „jen“ jedním z typů
  - ale je součástí daného uspořádání
    - oba modely (razítka, snímky) jsou ekvivalentní
  - datový typ:  $(T \times D^n) \rightarrow \text{bool}$



# Příklad

| • Jméno | Společnost | Rok  |
|---------|------------|------|
| Jan     | IBM        | 1990 |
| Jan     | IBM        | 1991 |
| Jan     | MS         | 1993 |
| Jan     | MS         | ...  |
| Eva     | DEC        | 1984 |
| Eva     | DEC        | 1985 |
| Eva     | IBM        | 1990 |
| Eva     | IBM        | ...  |
| Iva     | HP         | 1990 |
| Iva     | HP         | ...  |



# Tabulka s obojím časem

- **Jen připojuje**
- **Čas platnosti a transakce**
- **ROLL-BACK**
- **Dotazy do historie**
- **Analogy**
  - **CV v šanonu za posledních X let**



# Datový model času platnosti

- **Časová razítka**

- atributy jsou atomické, bez vnitřní struktury

- jeden chronon

- |     |      |    |
|-----|------|----|
| Jan | UIVT | 1  |
| Jan | UMAT | 6  |
| Jan | UIFS | 11 |
| Jan | NULL | 13 |



# Čas platnosti - pokračování

## – časový úsek

- Jan UIVT [1-5]  
Jan UMAT [6-10]  
Jan UIFS [11-12]

## – razítka k datům

- 1 → Jan    1 → UIVT  
...  
12 → Jan    5 → UIVT  
6 → UMAT  
...



# DB s obojím časem

- | Jméno | Ústav | Platný | Transakce |
|-------|-------|--------|-----------|
| Jan   | UIVT  | 1      | 1         |
| Jan   | UIVT  | 2      | 1         |
| ...   |       |        |           |
| Jan   | UIVT  | 1      | 2         |
| Jan   | UIVT  | 1      | 2         |
| ...   |       |        |           |
| Jan   | UMAT  | 6      | 8         |
| Jan   | UMAT  | 7      | 8         |
| ...   |       |        |           |





# DB s obojím časem

- **Možno modelovat i tak, že časová razítka se přiřazují atributům v podobě časových oken**



# Poznámka

- **Konečnost není nutné řešit na této úrovni**
- **Rozdílné modely pro data s časem  
⇒ rozdílná implementace  
(reprezentace)**



# Dotazovací jazyky

- Na formální úrovni
- Relační kalkul
  - $L ::= r_i(x_1, \dots, x_k) \mid x_i = x_j \mid L \wedge L \mid \neg L \mid \exists x. L$
- Příklad: vypiš všechny zaměstnance IBM
  - $\{ x: \exists y. \text{Zam}(x, y) \wedge y = \text{IBM} \}$



# Úprava relačního kalkulu

- **Rozšíření jsou typicky dvě**
  - **implicitní odkazy na časové údaje**
    - časové spojky
  - **explicitní odkazy na časové údaje**
    - proměnné a kvantifikátory
- **Založeno na**
  - ***formální logice***



# Typy formálních logik

- **Výroková (proposiční)**
  - tradiční *Booleovská* algebra,  $\{0,1\}$
- **Logika první řádu (predikátová)**
  - universální a existenční kvantifikátory
- **Logika vyššího řádu (higher-order)**
  - dedukce nad množinami a funkcemi
- **Modální/temporální logika**
  - dedukce o tom, co se stane, může stát, ...



# Výroková logika - syntaxe

- $P, Q, R, \dots, X_i$ 
  - výroky (výrokové symboly)
- $t$ : true;  $f$ : false
- $\neg P$ : negace;  $P \wedge Q$ : součin;  $P \vee Q$ : součet
- $P \rightarrow Q$ : jestli  $P$  potom  $Q$ ;  
 $P \leftrightarrow Q$ :  $P$  právě tehdy když  $Q$



# Výroková logika - sémantika

| • | P | Q | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \rightarrow Q$ | $P \leftrightarrow Q$ |
|---|---|---|----------|--------------|------------|-------------------|-----------------------|
|   | t | t | f        | t            | t          | t                 | t                     |
|   | t | f | f        | f            | t          | f                 | f                     |
|   | f | t | t        | f            | t          | t                 | f                     |
|   | f | f | t        | f            | f          | t                 | t                     |



# Predikátová logika - syntaxe

- **Termy**

- konstanty:  $a, b, c, \dots$
- proměnné:  $u, v, w, \dots$
- funkční symboly:  $f(t_1, \dots, t_n)$

- **Predikáty**

- true (T), false (F)
- predikátové symboly:  $p(t_1, \dots, t_n)$

- **Formule**

- predikáty
- $P, Q$  formule:  $\neg P, P \wedge Q, P \vee Q, P \rightarrow Q, P \leftrightarrow Q$
- $x$  proměnná,  $P$  formule:  $\forall x.P, \exists x.P$





# Predikátová logika - sémantika

- *interpretace* funkcí, konstant, predikátů
- přiřazení hodnot proměnným
- částečná rozhodnutelnost pro formule

# Temporální spojky prvního řádu

- $O ::=$ 
  - $X_i$  predikáty, ...
  - $O \wedge O, \dots$  logické spojky
  - $t_i < t_j$
  - $t_i.O$  temporální prom.
- Definice
  - $n$ -ární temporální spojka je  $O$  formule s přesně jednou volnou proměnnou  $t$  a  $n$  volnými predikáty  $X_1, \dots, X_n$



# Příklady spojek

- **$X_1$  until  $X_2$** 
  - $\exists t_2. t_0 < t_2 \wedge X_2 \wedge \forall t_1 (t_0 < t_1 < t_2 \rightarrow X_1)$
- **$X_1$  since  $X_2$** 
  - $\exists t_2. t_0 > t_2 \wedge X_2 \wedge \forall t_1 (t_0 > t_1 > t_2 \rightarrow X_1)$



# Odvozené spojky

- $\Diamond X_1 = \text{true until } X_1$ 
  - někdy (v budoucnu)
- $\blacklozenge X_1 = \text{true since } X_1$ 
  - někdy (v minulosti)
- $\Box X_1 = \neg \Diamond \neg X_1$ 
  - vždy (v budoucnu)
- $\blacksquare X_1 = \neg \blacklozenge \neg X_1$ 
  - vždy (v minulosti)



# Spojky pro diskrétní čas

- $\bigcirc X_1$ 
  - $\exists t_1. t_1 = t_0 + 1 \wedge X_1$ 
    - příští (v příštím kroku, časovém okamžiku)
- $\bullet X_1$ 
  - $\exists t_1. t_1 + 1 = t_0 \wedge X_1$ 
    - předchozí (v předešlém kroku, časovém okamžiku)
- *Ne všechny spojky lze v rámci prvního řádu definovat*



# Výroková temporální logika

- **Výroková TL(since, until)**
  - shodná s monadickou FOL( $<$ ) nad úplným lineárním uspořádáním
- **Výroková TL(until)**
  - shodná s TL(since, until) nad úplným lineárním uspořádáním omezeným v minulosti
- *Formule v VTL(since, until) mohou být rozděleny na podčásti obsahující pouze dotazy na minulost, současnost a budoucnost*

# Temporální logika prvního řádu - syntaxe

- Necht'  $\Omega$  je konečná množina  
temporálních spojek

–  $F ::= r_i(x_{i1}, \dots, x_{ik})$

$F \wedge F, \dots$

$x_i = x_j$

$\exists x_i. F$

$\omega(F_1, \dots, F_k)$

DB schéma

logické spojky

data (proměnné)

**temp. spojky**

# Temporální logika prvního řádu - sémantika

- $DB, \Theta, t \models r_i(x_{i1}, \dots, x_{ik})$ 
  - když  $r_i \in \rho, (\Theta(x_{i1}), \dots, \Theta(x_{ik})) \in r_i^{DB(t)}$
- $DB, \Theta, t \models x_i = x_j$ 
  - když  $\Theta(x_i) = \Theta(x_j)$
- $DB, \Theta, t \models \varphi = \psi$ 
  - když  $DB, \Theta, t \models \varphi$  a zároveň  $DB, \Theta, t \models \psi$
- $DB, \Theta, t \models \neg\varphi$ 
  - když neplatí, že  $DB, \Theta, t \models \varphi$
- $DB, \Theta, t \models \exists x_i. \varphi$ 
  - když existuje  $a \in D$  a zároveň  $DB, \Theta[x_i \mapsto a], t \models \varphi$
- $DB, \Theta, t \models \omega(F_1, \dots, F_k)$ 
  - když  $T_P, [t_0 \mapsto t] \models \omega^*$ , kde  $T_P, \delta \models X_i$ , iff  $DB, \Theta, \delta(t_i) \models F_i$





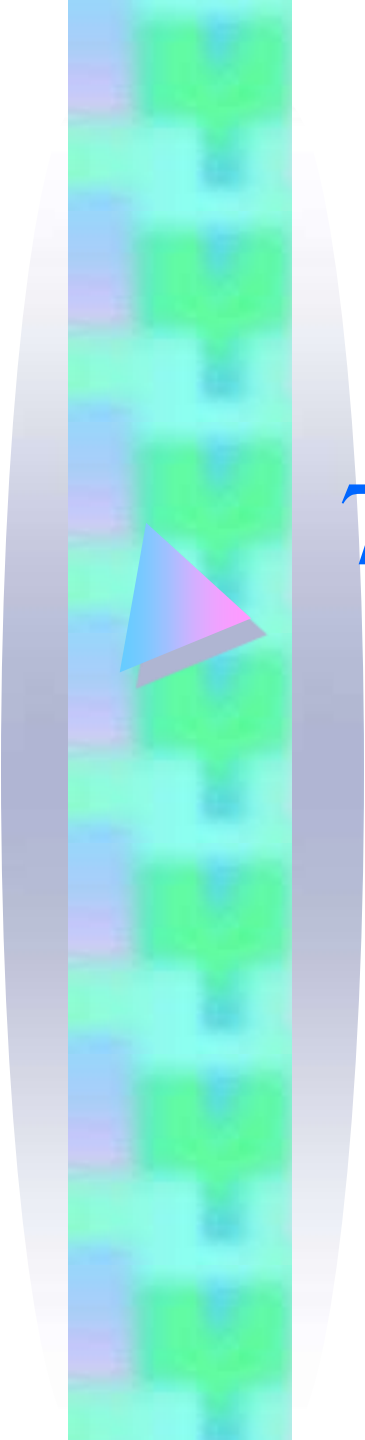
# Příklady

- **Odpověď' obecně**
  - $\varphi(\text{DB}) = \{t, \Theta : \text{DB}, \Theta, t \models \varphi\}$
- **Historie Janova zaměstnání**
  - $\exists x. \text{Zam}(x, y) \wedge x = \text{Jan}$
- **Všichni, kdo byli znovu přijati ke stejnému zaměstnavateli**
  - $\exists y. \text{Zam}(x, y) \wedge \blacklozenge (\neg \text{Zam}(x, y) \wedge \blacklozenge \text{Zam}(x, y))$



# Náměty k zamyšlení

- **Všichni, kdo mezi dvěma zaměstnáními byli nezaměstnaní**
- **Všichni, kdo pracovali u IBM od té doby, co Jan odešel**



## 7. Temporální databáze - temporálně relační kalkul a algebra, jazyky

# Temporálně relační kalkul - syntaxe

- $M ::= R_i(t_j, x_{i1}, \dots, x_{ik})$  rozš. DB schéma
- $M \wedge M,$
- $\neg M$  logické spojky
- $x_i = x_j$
- $\exists x_i. M$  data (proměnné)
- $t_i = t_j$
- $\exists t_i. M$  temp. proměnné

– *jedná se o dvou-druhovou logiku prvního řádu 2-FOL*



# Temporální relační kalkul - sémantika

- $DB, \Theta \models R_j(t_i, x_{i1}, \dots, x_{ik})$ 
  - když  $R_j \in \rho, (\Theta(t_i), \Theta(x_{i1}), \dots, \Theta(x_{ik})) \in R_j^{DB}$
- $DB, \Theta \models x_i = x_j$ 
  - když  $\Theta(x_i) = \Theta(x_j)$
- $DB, \Theta \models \varphi \wedge \psi$ 
  - když  $DB, \Theta \models \varphi$  a zároveň  $DB, \Theta \models \psi$
- $DB, \Theta \models \neg \varphi$ 
  - když neplatí, že  $DB, \Theta \models \varphi$
- $DB, \Theta \models t_i < t_j$ 
  - když  $\Theta(t_i) < \Theta(t_j)$



# Temporální relační kalkul - sémantika (dokončení)

- $DB, \Theta \models \exists t_i. \varphi$ 
  - když existuje  $s \in T_p$  a zároveň  $DB, \Theta[t_i \mapsto s] \models \varphi$
- $DB, \Theta \models \exists x_i. \varphi$ 
  - když existuje  $a \in D$  a zároveň  $DB, \Theta[x_i \mapsto a] \models \varphi$
- **Formát odpovědi na dotaz**
  - $\varphi(DB) = \{ \Theta : DB, \Theta \models \varphi \}$



# Příklady 1

- **Historie Janova zaměstnání**
  - $\exists x. \text{Zam}(t_0, x, y) \wedge x = \text{Jan}$
- **Všichni, kdo byli znovu přijati ke stejnému zaměstnavateli**
  - $\exists y. \text{Zam}(t_0, x, y) \wedge$   
 $\exists t_1 (t_1 < t_0 \wedge \neg \text{Zam}(t_1, x, y) \wedge$   
 $\exists t_2. t_2 < t_1 \wedge \text{Zam}(t_2, x, y))$



## Příklady 2

- **Všichni, kdo mezi dvěma zaměstnáními byli nezaměstnaní**
  - $\exists y. \text{Zam}(t_0, x, y) \wedge$   
 $\exists t_1 (t_1 < t_0 \wedge \neg \exists y. \text{Zam}(t_1, x, y) \wedge$   
 $\exists t_2. t_2 < t_1 \wedge \exists y. \text{Zam}(t_2, x, y))$
- **Všichni, kdo pracovali u IBM od té doby, co Jan odešel**
  - $\exists t_1 < t_0 ( \text{Zam}(t_1, \text{Jan}, \text{IBM}) \wedge$   
 $\forall t_2. t_2 > t_1 \rightarrow \neg \text{Zam}(t_2, \text{Jan}, \text{IBM}) \wedge$   
 $\forall t_2. t_1 < t_2 \leq t_0 \rightarrow \text{Zam}(t_2, x, \text{IBM}) )$





# Vyjadřovací síla ( $E$ )

- $E(r_i(x_1, \dots, x_{v_1})) = R_i(t_0, x_1, \dots, x_{v_1})$
- $E(x_i = x_j) = x_i = x_j$
- $E(F_1 \wedge F_2) = E(F_1) \wedge E(F_2)$
- $E(\neg F) = \neg E(F)$
- $E(\exists x.F) = \exists x.E(F)$
- $E(\omega(F_1, \dots, F_k)) = \omega^*(E(F_1)[t_0/t_1], \dots, E(F_k)[t_0/t_k])$



# Srovnání vyjadřovací síly

- Temporální logika prvního řádu (FOTL) má ostře menší vyjadřovací sílu než dvou-druhov<sup>á</sup> logika prvního řádu (2-FOL) pro libovolnou konečnou množinu logických spojek
- FOTL omezená na budoucí děje má menší vyjadřovací sílu než FOTL s lineárním uspořádáním nad časem s levým ukončením
- $\text{FOTL} = \text{2-FOL}$  jestliže datová doména  $D$  má pevnou velikost



# Temporální relační algebra

- **TRA sestává z:**
  - **Universum** - jednorozměrné temp. relace
  - **Operace** - konečný počet operací typu
$$2^{T \times Dn1} \times \dots \times 2^{T \times Dnk} \rightarrow 2^{T \times Dn}$$
  - **Operace prvního řádu** - formule prvního řádu
- *Všechny operace by měly být uzavřeny vůči universu, což znamená, že nelze vyjádřit všechny dotazy prvního řádu.*



# Proč nelze vše vyjádřit?

- **Problémem nejsou operátory, ale požadavek na *uzavřenost* operací**
  - universum obsahuje pouze jednorozměrné temporální relace
- **Problém i pro implementaci**
  - nelze vytvořit všechny dotazy prvního řádu
- *Řada dotazovacích jazyků to však ignoruje*



# Příklad TRA

- **Universe:**

- temporální relace typu  $T \times D^k$

- **Operace:**

- projekce  $\pi$
  - selekce  $\sigma$
  - spojení  $\bowtie$
  - sjednocení  $\cup$
  - rozdíl  $-$
  - since  $S$
  - until  $U$



# Definice operací

- $\pi_V(R) = \{ t, \Theta|_V : DB, \Theta, t \models R \}$
- $\sigma_F(R) = \{ t, \Theta|_{FV(R)} : DB, \Theta, t \models R \wedge F \}$
- $R \triangleright \triangleleft S = \{ t, \Theta|_{FV(R) \cup FV(S)} : DB, \Theta, t \models R \wedge S \}$
- $R \cup S = \{ t, \Theta|_{FV(R) \cup FV(S)} : DB, \Theta, t \models R \vee S \}$
- $R - S = \{ t, \Theta|_{FV(R) \cup FV(S)} : DB, \Theta, t \models R \wedge \neg S \}$
- $S(R, S) = \{ t, \Theta|_{FV(R) \cup FV(S)} : DB, \Theta, t \models R \text{ since } S \}$
- $U(R, S) = \{ t, \Theta|_{FV(R) \cup FV(S)} : DB, \Theta, t \models R \text{ until } S \}$



# Temporální logika TL(FO)

- **Temporální logika prvního řádu nemá vlastnost oddělení (*separation property*) z výrokové logiky**
  - $\forall x. \blacklozenge p(x) \Leftrightarrow \blacklozenge p(x)$
- **Proto [Gabbay et al., 1994] slabší, dvouvrstvá logika TL(FO)**
  - časové spojky jen mimo dosahu  $\forall, \exists$
  - má možnost oddělení

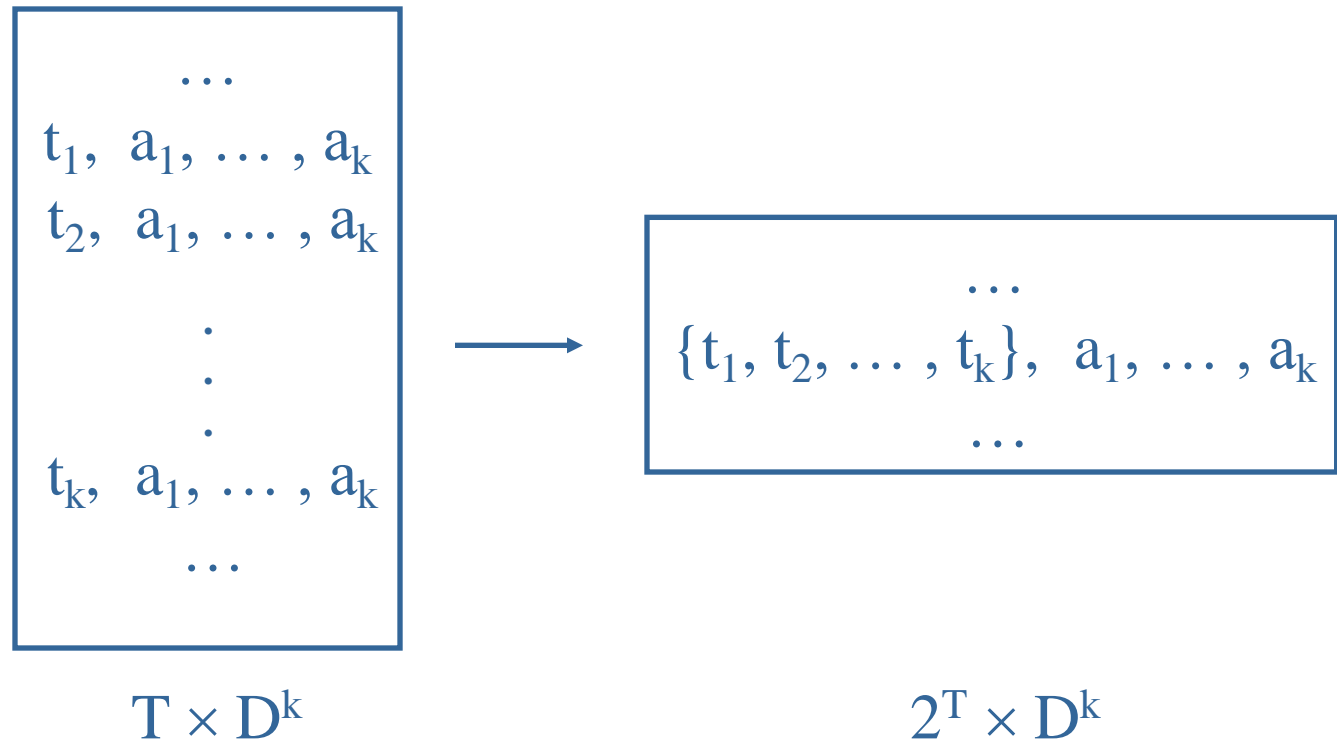


# Konkrétní temporální DB

- **Sdružování přes atributy neobsahující časovou informaci**
- **a konečné kódování výsledných množin časových okamžiků**
- *Poznámka:  
jedna datová  $n$ -tice je často vztažena k několika časovým okamžikům*



# Zobrazení log. DB na impl. DB



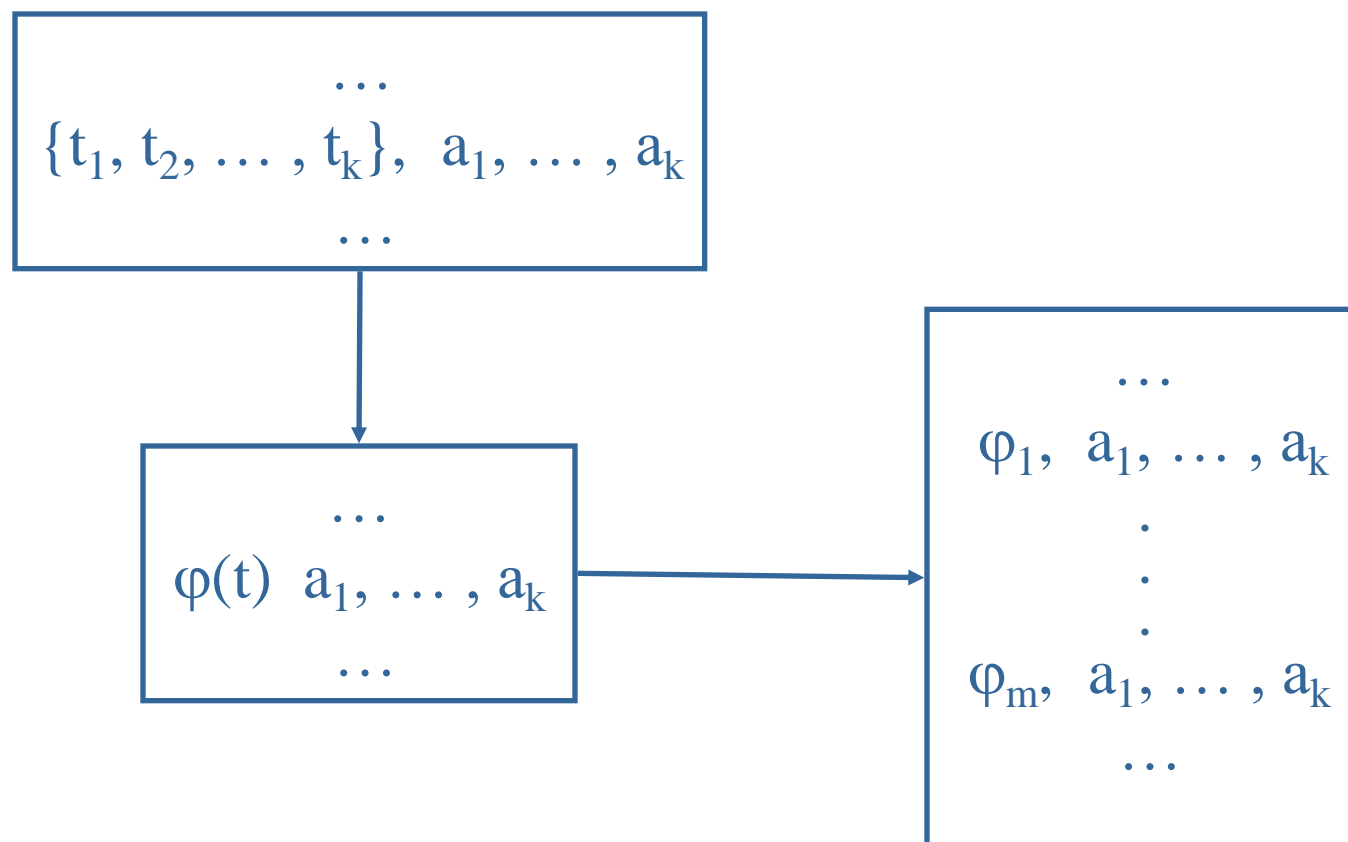
Implicitní funkční závislost:  $A_1 \dots A_k \rightarrow T$  ve  $2^T \times D^k$



# Konečné kódování omezením

- Množiny časových okamžiků jsou definovány svými charakteristickými funkcemi (formulemi)
- Jazyk omezení umožňuje odstranit (eliminovat) kvantifikátory
- $n$ -tice pevné velikosti = formule bez kvantifikátorů  
v CNF

# Způsob kódování





# Kódování intervalů

- Necht'  $T_P$  je časová doména.
- Definujeme množinu
  - $I(T) = \{ (a,b) \mid a \leq b, a \in T \cup \{-\infty\}, b \in T \cup \{\infty\} \}$
- Relace na množině  $I(T)$ 
  - $([a,b] <_- [a',b']) \Leftrightarrow a < a'$
  - $([a,b] <_{-+} [a',b']) \Leftrightarrow a < b'$
  - $([a,b] <_{+-} [a',b']) \Leftrightarrow b < a'$
  - $([a,b] <_{++} [a',b']) \Leftrightarrow b < b'$



# Definice

- Struktura

$$T_I = (I(T), <_{--}, <_{-+}, <_{+-}, <_{++})$$

se nazývá

*intervalová časová doména vzhledem k  $T_P$*

# Kódování intervalů - schéma

Abstraktní temporální databáze

Odrazy konkrétních temporálních databází

D:

$\{(a,1), (a,2), (a,3), \dots\}$

$\varphi = R(x,y) \wedge y < 10$

$\varphi(D)$ :

$\{(a,1), (a,2), \dots, (a,9)\}$

$\|E_1\|$

$\|E_2\|$

$\|. \|$

$\|. \|$

$\{(a,[1,5]), (a,[3,\infty])\}$

$\text{eval}(\varphi)(E_1)$

$\{(a,[1,5]), (a,[3,9])\}$

$\{(a,[1,\infty])\}$

$\text{eval}(\varphi)(E_2)$

$\{(a,[1,9])\}$

Konkrétní temporální databáze



# Příklad

| Zaměstnání |            |                   |
|------------|------------|-------------------|
| Jméno      | Společnost | Rok               |
| Jan        | IBM        | [1990, 1991]      |
| Jan        | HP         | [1993, $\infty$ ] |
| Marie      | DEC        | [1984, 1985]      |
| Marie      | IBM        | [1990, $\infty$ ] |
| Karel      | DELL       | [1990, $\infty$ ] |



# Intervalové dotazy

- $M ::= R_i(I_j, x_{i1}, \dots, x_{ik})$  rozš. DB schéma
  - $M \wedge M,$
  - $\neg M$  logické spojky
  - $x_i = x_j$
  - $\exists x_i. M$  data (proměnné)
  - $I_i^* = I_j^*$
  - $\exists t_i. M$  temp. proměnné
- jazykem  $M$ -formulí je  $L'$






# Generičnost dotazů

- Dotaz  $f$  je generický vzhledem k  $\|\cdot\|$ , pokud platí
  - $\|D_1\| = \|D_2\| \supset \|fD_1\| = \|fD_2\|$
- Příklad
  - $R^{D_1} = \{([0,3],a)\}$
  - $R^{D_2} = \{([0,2],a), ([1,3],a)\}$
  - $\exists i,j. \exists x(R(i,x) \wedge R(j,x) \wedge i \neq j)$  platí v  $D_2$ , ale nikoliv u  $D_1$  - tedy není generický



# Problém

- **Dotaz, který není generický může ,zpochybnit‘ určitá fakta jen podle toho, jak jsou data uložena**
- **Možnosti**
  - **změna sémantiky - velmi složité**
  - **změna struktury DB - možné**



# Shlukování (coalescing)

- **Definice**

- **Jednorozměrná temporální relace obsahuje shluky, pokud je každý fakt spojován s nejvýše konečným počtem nepřekrývajících se intervalů**
  - je třeba jej zaručit, pokud se nad relacemi provádějí ne-logické operace
  - relační operátory shlukování nezaručují (projekce, sjednocení, množinový rozdíl)



# Příklad

- **DB se shluky**
  - $R^{D1} = \{([0,3],a)\}$
- **DB beze shluků**
  - $R^{D2} = \{([0,2],a), ([1,3],a)\}$



# Jazyky temporálních DB

- Na rozdíl od prostorových DB je textová reprezentace možná a vhodná
- Je třeba postihnout
  - klasické potřeby DB jazyků
    - odvození, rozšíření
  - připojit temporální logiku
    - úplné připojení
    - využití klíčových vlastností



# TQUEL [Snodgrass, 1987]

- **Minimální rozšíření jazyka QUEL**
- **Čas platnosti, čas transakce**
- **Kódování intervalů**
- **Shluky**



# TSQL2 [Snodgrass, 1995]

- Podobně jako v předchozím případě rozšiřuje SQL
- Čas platnosti a transakce
  - časové elementy jako časová razítka
  - n-tice nejsou vázány svoji aritou
- **Není formální sémantika**
  - výrazová mocnost není zcela zřejmá



## Příklad 1

- **select**                      **r.name**  
  **from**                      **Works r, Works s**  
  **where**                    **r.name=s.name**  
    **and**                    **r.company=s.company**  
    **and**                    **validtime(r) precedes**  
                                 **validtime(s)**
- **nefunguje obecně, neboť je třeba převést na shluky**



```
– select r.name
 from Works r, Works s
 where r.name=s.name
 and validtime(r) precedes
 validtime(s)
```

- nefunguje, neboť je třeba převést na shluky poté, co je „odstraněno“ jméno zaměstnavatele (pod-dotazy, pohledy)



# Další jazyky

- **HRDM**

- časově závislé atributy, časová razítka

- **IXRM**

- intervaly, vícerozměrný model, EXPSPACE ops

- **TSQL**

- kódování intervalů, temporal relational algebra

- **Následníci TSQL2**

- **ATSQL**

- **SQL/Temporal**



# Další problémy s ukládáním

- **Intervaly a skutečné intervaly**
  - interval jako kódování množin časových okamžiků
  - intervaly jako body ve 2-rozměrném prostoru
- **Shluky**
  - zjednodušují selekci, projekci, spojení
  - nezjednodušují temporální operací



# Selhání shlukování

- **nejednoznačné shlukování pro více rozměrů**
- **FO-úplný dotazovací jazyk se neobejde bez n-rozměrnosti v dotazech**
- **dotazy jsou závislé na reprezentaci**
  - **negenerické dotazy**



# Jiné možnosti v modelech

- **Užití abstraktního dotazovacího jazyka**
- **Vyhodnocení přeložit vhodně pro kódovanou DB**
- **Řešení (např.)**
  - **FOTL + kódování intervalů**
  - **2-FOL + kódování intervalů**
    - **SQL/TP**



# SQL/TP [Toman, 1997]

- **Syntaxe a sémantika**
  - rozšiřuje SQL o nový datový typ
  - podpora konečné duplicity
- **Kódování relací**
  - intervaly (kompaktní)
  - restriktce v syntaxi (bezpečnost)
- **Vyhodnocení dotazů**
  - efektivita je závislá jen na velikosti kódování
  - je možné přeložit na SQL/92



# Jazyky s větší ‚silou‘

- Více časových dimenzí
- Časové spojky vyššího řádu
- Vlastnosti vyšších řádů jsou patrné i v jazyce
- Základem
  - více-dimenzionální temporální logiky



# Více rozměrů

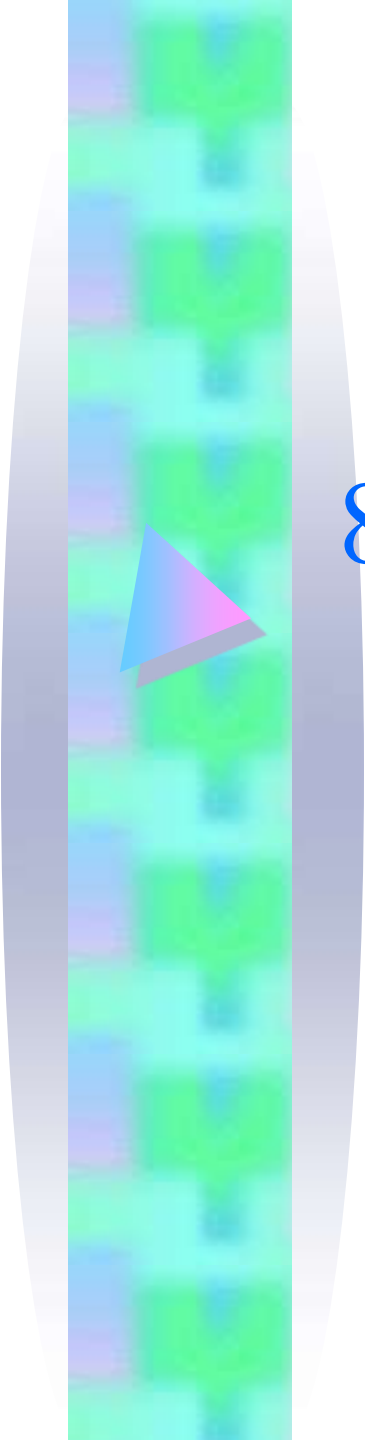
- **V časové rovině**
  - ETL
  - $\mu$ TL
- **V datové rovině**
  - Datalog( $\neg$ )
- **V obou**
  - Datalog<sub>1s</sub>
  - TempLog





# Další rozšíření

- **Neúplné časové informace**
  - **částečná informace**
    - Sylva skončila u IBM a začala u HP před rokem 1992
    - Jan předtím, než pracoval pro IBM byl u HP
  - **různá granularita**
    - Beatles se rozpadli v šedesátých letech
- **Prázdné (NULL) hodnoty**



## 8. Temporální databáze - algoritmy, funkční závislosti, kódování



# Indexování v temporálních DB

- **Problém**

- máme-li interval  $I$ , vyber všechny  $n$ -tice, které mají s tímto intervalem neprázdný průnik
- *výsledkem je nevyvážená struktura pro přístup k datům*



# Co víme dále

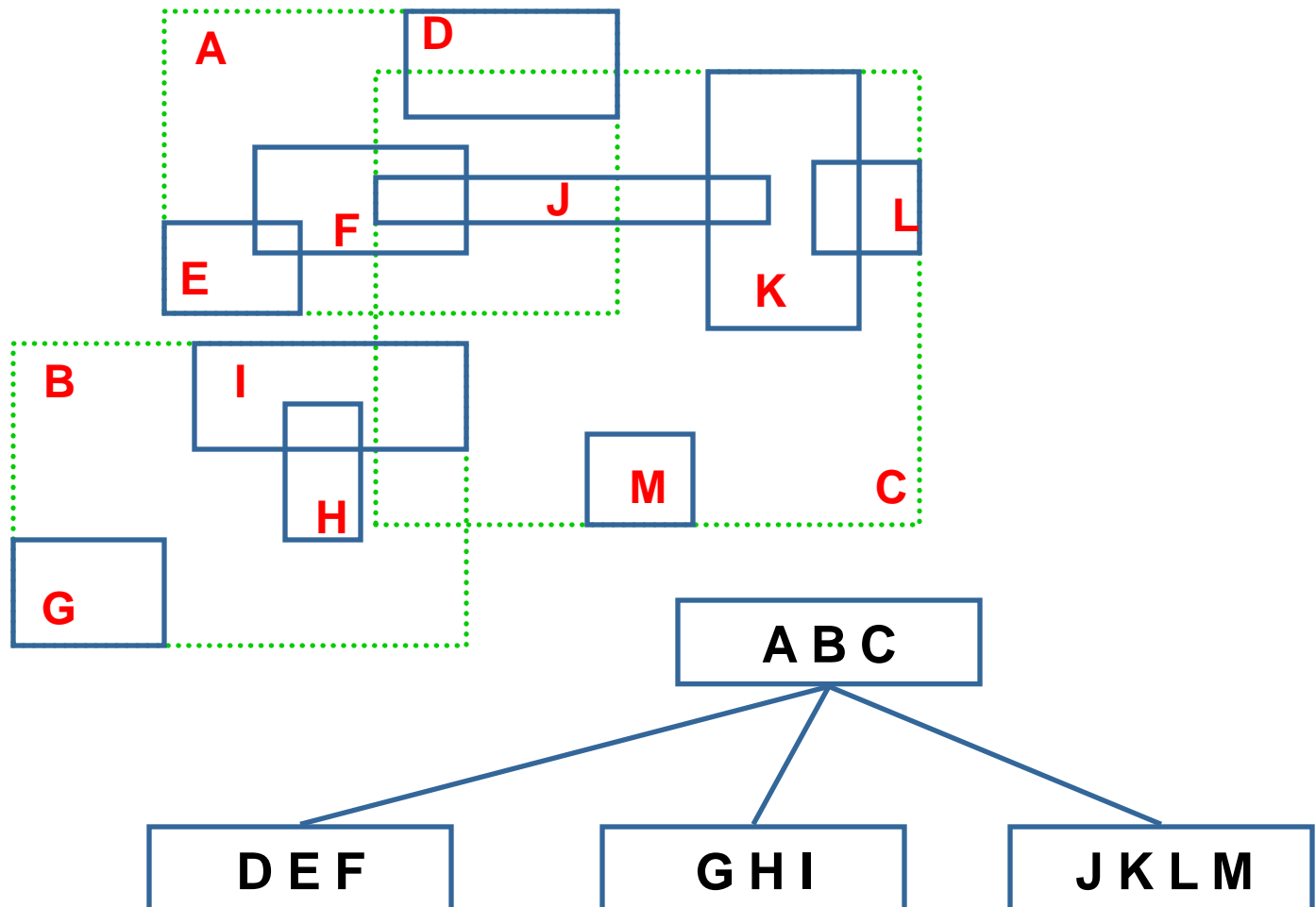
- **intervaly jsou neprázdné**
  - ne každá dvojice časových okamžiků tvoří interval
- **do současného stavu se přistupuje mnohem více dotazy**
- **temporální databáze jsou často ,append only‘ - transakce**



# Metody indexování

- **Time Index [Elmaseri, Ramaswamy]**
- **R-Tree [Guttman]**
  - známe z prostorových DB
- **AP-Tree**
  - index jen pro přidávání dat
  - kombinací algoritmů ISAM a B<sup>+</sup>-Tree

# R-Tree





# Příklad

- **Time Index**
  - na začátku a konci každého intervalu uloží seznam všech intervalů obsahujících tento okamžik
  - nad těmito body postav B-Tree
- **Dotazy**
  - $O(\log n + s/B)$
- **Prostor**
  - $O(n^2 / B)$  - nevýhoda (odstraněn kvadrát)



# Úpravy v DB

- **Množina vs. multi-množina**
  - **vkládání**
    - duplikace razítek a překrývání
  - **mazání**
    - jen část n-tice je mazána
  - **změna dat**
    - jako mazání (část n-tic)
    - problém: úprava na místě





# Specifické problémy

- **Při dělení je složitý výběr dělicích bodů na časové ose**
- **Optimalizace při dotazování jsou často založeny na odhadu velikosti výsledku**
- **Sekvenční a paralelní algoritmy jsou ovlivněny různě**



# Algoritmy

- **Dělení intervalu**
  - rovnoměrné
  - „optimální“
  - další varianty
- **Grafové**
  - klasické
  - „optimální“



# Temporální integritní omezení

- **Relační DB**
  - omezení jsou uzavřené logické formule prvního řádu
- **Temporální DB**
  - omezení jsou uzavřené formule prvního řádu temporálního dotazovacího jazyka



# Použití integritních omezení

- **Zachycení sémantiky DB aplikace**
- **Důvod zavedení**
  - ukládání pouze ‚významných‘ dat
- **Návrh DB**
  - normální formy
    - ‚dobrá‘ schémata (bez anomálií)
    - *dobrá* dekompozice



# Funkční závislosti

- **Klíče DB**
- **Určení závislosti dat na klíči**
- **Boyce-Coddova normální forma**



# Temporální funkční závislosti

- **[Jensen et al. 1996]**
- **neexistují logické formulace**
- **řada technických problémů**



# Závislosti omezení

- Funkční závislosti generují rovnost (ekvivalenci na)
- V časových DB se navíc jedná o závislosti generující omezení
  - *Historii nelze změnit*



# Splnitelnost omezení

- Historie  $H$  splňuje omezení  $O$  jestliže  $O$  je pravdivé v každém stavu  $H$
- Konečná historie  $H$  potenciálně splňuje omezení  $O$  jestliže může být rozšířena do nekonečné historie tak, že splňuje  $O$
- Implementace
  - *po každé změně se kontroluje  $O$  v aktuální historii*





# Omezení dána temporální logikou

- **V zásadě bylo navrženo a studováno mnoho přístupů**
  - rozhodnutelnost
  - použitelnost pro danou aplikaci
  - jednoduchost



# Příklady na rozhodování

- Dříve propuštění zaměstnanci nemohou být znovu přijati
  - $\neg(\exists x)(\text{přijat}(x) \wedge \blacklozenge \text{propuštěn}(x))$
- Dříve propuštění zaměstnanci nemohou být znovu přijati pokud nebyli rehabilitováni
  - $\neg(\exists x)(\text{přijat}(x) \wedge (\neg \text{rehabilitován}(x) \text{ since } \text{propuštěn}(x)))$
- Rehabilitace je možná jen jednou
  - $\neg(\exists x)(\text{rehabilitován}(x) \wedge \blacklozenge \text{rehabilitován}(x))$



# Bi-kvantifikované formule

- **[Lipeck a Saake, 1987]**
  - spojky jen pro budoucnost
  - kvantifikátory (bud'/anebo)
    - externí - mimo rozsah časových spojek
    - interní - bez časové spojky v jejich rozsahu
  - žádné interní kvantifikátory
    - omezení řešitelná v exp. čase
  - jeden interní kvantifikátor
    - nerozhodnutelné



# Formule pro minulost

- **[Chomicki, 1995] - TL reálného času**
  - jen spojky pro operace v minulosti
  - libovolné kvantifikátory
  - omezení potenciálně nerozhodnutelná
  - praktiky na aproximaci rozhodnutelnosti
    - je omezení splněno v současném stavu?



# Kódování historie

- Chomicky, J.: Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. *ACM Transactions on Database Systems*, 20(2):149-186, 1995.
- schéma DB je rozšířeno o *doplňkové* relace
- každý stav DB je rozšířen o instanci doplňkových relací, čímž je vytvořen *rozšířený* stav
- rozšířený stav  $H'_k$  kóduje celou současnou historii ( $H_0, \dots, H_k$ )



# Vlastnosti kódování

- **inkrementální výpočet**
- **prostorově efektivní**
- **ztrátové**



# Rozšířené relace

- **Rozšířené relace pro omezení  $O$ :**
  - jedna rozšířená relace  $r_\alpha$  pro každou temporální sub-formuli  $\alpha$  v  $O$ 
    - např.  $\bullet A$ , nebo  $A$  *since*  $B$
  - arita relace  $r_\alpha$  je rovna počtu volných proměnných v  $\alpha$
  - rozšířená relace omezení  $r_o$  (0-ární)



# Způsob zavedení

- Rozšířené relace jsou definovány indukcí na čase.
- Definice jsou automaticky odvozeny z omezení
- Relace jsou druh pohledů (view)
- Omezení  $O$  je splněno tehdy, pokud, pokud relace  $r_0$  obsahuje prázdnou  $n$ -tici  $()$





# Příklad

- Dříve propuštění zaměstnanci nemohou být znovu přijati pokud nebyli rehabilitováni
  - $\neg(\exists x)(\text{přijat}(x) \wedge (\neg\text{rehabilitován}(x) \text{ since propuštěn}(x)))$
- Rozšířené relace
  - $r_{\alpha}(x) =_{(\text{df})} \neg\text{rehabilitován}(x) \text{ since propuštěn}(x)$
  - $r_0 =_{(\text{df})} \neg(\exists x)(\text{přijat}(x) \wedge r_{\alpha}(x))$
- Induktivní definice  $r_{\alpha}$ 
  - $r_{\alpha}^0(x) =_{(\text{df})} \text{FALSE}$
  - $r_{\alpha}^{k+1}(x) =_{(\text{df})} (r_{\alpha}^k(x) \vee \text{propuštěn}^k(x)) \wedge \neg\text{rehabilitován}^{k+1}(x)$
- *horní index označuje stavy*
- *definice nezávisí na hodnotě indexu (vyjma 0)*



# Prostorové a časové požadavky

- **Předpoklady**

- pevná, konečná množina integritních omezení IC
- pevné DB schéma

- **Aktivní doména  $adom(D)$  historie  $D=(D_0, \dots, D_k)$  je množina doménových hodnot, které se objevují v  $D$**



# Omezení (limitace) historie

- Historie je *omezená* (bounded) když
  - je rozšířené schéma pevné a konečné
  - pro každé  $n \in \mathbb{N}$  existuje  $m \in \mathbb{N}$  takové, že pro každou historii  $H = (H_0, \dots, H_k)$  je splněna podmínka
    - když  $|adom(H)| < n$  pak  $size(H'_k) < m$



# Prostorová složitost (efektivita)

- **Toto kódování je polynomiálně omezené (prostor potřebný k uložení kódu roste polynomiálně vzhledem k počtu vložených stavů - časových okamžiků)**

# (Ne)omezená kódování

- |                            | 0       | 1       | 2       | 3       | ... |
|----------------------------|---------|---------|---------|---------|-----|
| $\{x:p(x)\}$               | $\{a\}$ | $\{\}$  | $\{a\}$ | $\{\}$  | ... |
| $\{x:\blacklozenge p(x)\}$ | $\{\}$  | $\{a\}$ | $\{a\}$ | $\{a\}$ | ... |

– konstantní prostor pro pomocnou relaci

- |                            | 0         | 1         | 2              | 3                   | ... |
|----------------------------|-----------|-----------|----------------|---------------------|-----|
| $\{x:p(x)\}$               | $\{a_0\}$ | $\{a_1\}$ | $\{a_2\}$      | $\{a_3\}$           | ... |
| $\{x:\blacklozenge p(x)\}$ | $\{\}$    | $\{a_0\}$ | $\{a_0, a_1\}$ | $\{a_0, a_1, a_2\}$ | ... |

– pomocná relace vyžaduje neomezený prostor, neboť aktivní doména je též neomezená



## 9. TSQL2



# Časový model

- **Struktura času**
  - lineární časový model
- **Omezenost**
  - omezen na obou koncích
- **Nerozlišuje mezi diskrétními, hustotními a spojitými modely**
  - dotaz zda  $a$  předchází  $b$  je třeba omezit na nějaký časový úsek



# Reprezentace času

- **Omezená diskrétní reprezentace reálné časové osy**
- **Nejmenší časová jednotka je chronon**
- **Následné chronony mohou seskupeny do granulí (různá zrnitost)**
- **Možno přecházet mezi zrnitostmi**





# Temporální datové typy z SQL92

- **Datum**
  - **DATE (YYYY-MM-DD)**
- **Čas**
  - **TIME (HH:MM:SS)**
- **Úplný čas**
  - **DATETIME (YYYY-MM-DD HH:MM:SS)**
- **Interval**
  - **INTERVAL (no default)**



# Temporální datové typy TSQL2

- **Perioda**
  - **PERIOD** (rozdíl (úplných) časů)



# Datový model

- **BCDM**
  - (Bitemporal Conceptual Data Model)
- Každá jednotka informace (fakt) je uložena v přiřazené n-tici
- Rozlišuje čas *platnosti* a *transakce*
- Časová razítka o elementy obojího času
  - množiny chrononů s dvojí časovou inf.



# Modelový případ

- **Plán bezpečnostních prověrek kontrol dodržování předepsaných standardů**
  - Kde se kontrola provádí
  - Kdo ji provádí
  - Počet členů kontrolní komise
  - Typ kontroly
  - Perioda opakování



# Vytvoření tabulky

- **CREATE TABLE** Kontroly  
(Kde, Kdo, Počet, Kontrola,  
Perioda **INTERVAL DAY**)  
**AS VALID MONTH AND**  
**TRANSACTION**



# Vlastnosti tabulky

- Čas platnosti určuje období ve kterých se kontrola provádí
- Čas platnosti má zrnitost na úrovni měsíce
- Čas transakce určuje, kdy byl údaj vložen do DB
  - zrnitost je systémově závislá



# Druhy tabulek

- **Snímek**
  - žádné atributy v definici
- **Platný čas (stavu)**
  - AS VALID [STATE] <zrnitost>
- **Čas platnosti události**
  - AS VALID EVENT <zrnitost>
- **Doba transakce**
  - AS TRANSACTION



## Druhy tabulek (pokr.)

- Oba druhy času, stavová
  - AS VALID [STATE] <zrnitost> AND TRANSACTION
- Oba druhy času, událostní
  - AS VALID EVENT <zrnitost> AND TRANSACTION
- *Typ je možné změnit příkazem ALTER*





## Dotazy 1.1

- **Kde probíhaly/jí kontroly**
  - **SELECT SNAPSHOT Kde**  
**FROM Kontroly**
- **Výsledek:**
  - **Výsledkem je seznam všech míst, kde se kdy konaly/jí nějaké kontroly.**



## Dotazy 1.2

- **Kde byla/je prováděna kontrola dle ISO9001?**
  - **SELECT SNAPSHOT Kde  
FROM Kontroly  
WHERE Kontrola = „ISO9001“**
- **Výsledek:**
  - **Výsledkem je seznam odpovídajících míst.**



## Dotazy 1.3

- **Dotazy spíše do historie**
- **Kde byly prováděny kontroly?**
  - **SELECT Kde**  
**FROM Kontroly**
- **Výsledek:**
  - **Výsledkem je seznam míst a každé je spojeno s jedním či více časovými úseky (maximálními).**



## Dotazy 1.4

- Byly ve spojení s ISO9001 prováděny i jiné kontroly?
  - `SELECT P1.Kde, P2.Kontrola  
FROM Kontroly AS P1, Kontrola AS P2  
WHERE P1.Kontrola = „ISO9001“  
AND P2.Kontrola <> „ISO9001“  
AND P1.Kde = P2.Kde`
- Výsledek:
  - Výsledkem je seznam míst a typů kontrol spolu s odpovídajícím jedním či více časovými úseky (maximálními).



## Dotazy 1.5

- Kde jsou/byly prováděny kontroly v rozpětí alespoň 2 let?
  - `SELECT Kde, Kontrola  
FROM Kontroly(Kde, Kontrola) AS P  
WHERE  
CAST(VALID(P) AS INTERVAL MONTH) >  
INTERVAL „24“ MONTH`
- Výsledek:
  - Výsledkem jsou nejdelší intervaly pro dané místo.



# Lokální pod-dotaz

- **Velmi užitečná vlastnost**
  - *Syntactic sugar*
    - **FROM A(B, C, ...) AS A2**
      - je ekvivalentem
    - **FROM (SELECT B, C, ... FROM A) AS A2**
- **Vnořené projekce automaticky shlukují výsledky**
- **Jiné atributy nejsou v A2 dostupné, A potom není dostupné dále**



## Dotazy 2

- Kde jsou/byly prováděny kontroly na ISO9001?
  - `SELECT SNAPSHOT P1.Kde  
FROM Kontroly(Kde) AS P, P(Kontrola) AS P1  
WHERE P1.Kontrola = „ISO9001“  
AND VALID(P) = VALID(P1)`
- Výsledek:
  - P všechny časy, kdy se konala kontrola.
  - P1 je shlukována podle místa a kontroly. Sloupce s atributem „Kde“ se u P i P1 shodují.



# Další typ pod-dotazu

- Opět je textové synonymum
  - **SELECT ...  
FROM A(B,C,D) AS A1, A1(E,F) AS A2  
WHERE ...**
    - je shodné s
  - **SELECT ...  
FROM (SELECT B,C,D FROM A) AS A1,  
(SELECT B,C,D,E,F FROM A) AS A2  
WHERE ... AND  
A1.B=A2.B AND A1.C=A2.C AND A1.D=A2.D  
AND VALID(A1) OVERLAPS VALID(A2)**





## Jak to funguje

- **A1 obsahuje tu časovou informaci, kdy atributy B, C, D byly konstantní.**
- **A2 obsahuje různé hodnoty atributů D a E, kde časová informace je podmnožinou shluků z A1.**



# Příklad

- **Jméno**

- Marie [1-18)

- Gizela [5-20)

## **Oddělení**

Obuv [1-10)

Papír [10-18)

Obuv [5-15)

Papír [15-20)

## **Vedoucí**

Lenka [1-8)

Jiří [8-18)

Lenka [5-8)

Jiří [8-20)



# Při použití TSQL2

| • | Jméno  | Oddělení | Vedoucí | Čas plat. |
|---|--------|----------|---------|-----------|
| • | Marie  | Obuv     | Lenka   | {[1-8)}   |
| • | Marie  | Obuv     | Jiří    | {[8-10)}  |
| • | Marie  | Papír    | Jiří    | {[10-18)} |
| • | Gizela | Obuv     | Lenka   | {[5-8)}   |
| • | Gizela | Obuv     | Jiří    | {[8-15)}  |
| • | Gizela | Papír    | Jiří    | {[15-20)} |



# Zajímá nás sled oddělení

- **FROM Zam(Jméno) AS E1,  
E1(Oddělení) AS E2**
- E1.Jméno = „Marie“ [1-18)  
E2.Oddělení = „Obuv“ [1-10)  
E2.Oddělení = „Papír“ [10-18)
- E1.Jméno = „Gizela“ [5-20)  
E2.Oddělení = „Obuv“ [5-15)  
E2.Oddělení = „Papír“ [15-20)



# Zajímá nás sled vedoucích

- **FROM Zam(Jméno) AS E1,  
E1(Vedoucí) AS E2**
- E1.Jméno = „Marie“ [1-18)  
E2.Vedoucí = „Lenka“ [1-8)  
E2.Vedoucí = „Jiří“ [8-18)
- E1.Jméno = „Gizela“ [5-20)  
E2.Vedoucí = „Lenka“ [5-8)  
E2.Vedoucí = „Jiří“ [8-20)



# Pokud chceme vše najednou

- **FROM Zam(Jméno) AS E1,  
E1(Oddělení) AS E2  
E1(Vedoucí) AS E3**
- E1.Jméno = „Marie“ [1-18)
  - E2.Oddělení = „Obuv“ [1-10)
  - E2.Oddělení = „Papír“ [10-18)
- E3.Vedoucí = „Lenka“ [1-8)
  - E3.Vedoucí = „Jiří“ [8-18)



# Pokud chceme znát historii oddělení

- **FROM Zam(Oddělení) AS E1,  
E1(Jméno) AS E2**
- E1.Oddělení = „Obuv“ [1-15)  
E2.Jméno = „Marie“ [1-10)  
E2.Jméno = „Gizela“ [5-15)
- E1.Jméno = „Papír“ [10-20)  
E2.Jméno = „Marie“ [10-18)  
E2.Jméno = „Gizela“ [15-20)



# Pokud chceme znát historii vedoucích oddělení

- **FROM Zam(Oddělení) AS E1,  
E1(Vedoucí) AS E2**
- E1.Oddělení = „Obuv“ [1-15)  
E2.Vedoucí = „Lenka“ [1-8)  
E2.Vedoucí = „Jiří“ [8-15)
- E1.Jméno = „Papír“ [10-20)  
E2.Vedoucí = „Jiří“ [10-20)



# Pokud chceme znát kdo byl kým veden

- **FROM Zam(Vedoucí) AS E1,  
E1(Jméno) AS E2**
- E1.Vedoucí = „Lenka“ [1-8)  
E2.Jméno = „Marie“ [1-8)  
E2.Jméno = „Gizela“ [5-8)
- E1. Vedoucí = „Jiří“ [8-20)  
E2.Jméno = „Marie“ [8-18)  
E2.Jméno = „Gizela“ [10-20)



# Pokud chceme znát kterými odděleními vedoucí procházeli

- **FROM Zam(Vedoucí) AS E1,  
E1(Oddělení) AS E2**
- E1.Vedoucí = „Lenka“ [1-8)  
E2.Oddělení = „Obuv“ [1-8)
- E1.Vedoucí = „Jiří“ [8-20)  
E2.Oddělení = „Obuv“ [8-15)  
E2.Oddělení = „Papír“ [10-20)



## Dotazy 3

- Kde byly prováděny tytéž kontroly v rozpětí více jak 2 let po sobě?
  - `SELECT SNAPSHOT Kde, Kontrola, VALID(P)`  
`FROM Kontroly(Kde, Kontrola)(PERIOD) AS P`  
`WHERE`  
`CAST(VALID(P) AS INTERVAL MONTH) >`  
`INTERVAL „24“ MONTH`
- Výsledek:
  - P běží přes dvojice Kde-Kontrola s asociovanými maximálními časovými úseky
  - Několik řádků, kde je shodný pár Kde-Kontrola
  - Jedná se o *snímek*, kde PERIOD je poslední sl.



# Dělení - partitioning

- **Nejedná se textovou zkratku (náhradu delšího zápisu)**
- **Není v souladu s datovým modelem (násobné řádky)**
- **Tento nesoulad je však dočasný a pouze *uvnitř* dotazu**
- **Užitečné pro dotazy, které sledují časovou spojitost událostí**



# Výběr času platnosti

- Časové razítko s časem platnosti pro jméno **A** se zapisuje **VALID(A)**
  - U stavové tabulky s nedělenými jmény se to vyhodnotí na časové elementy
  - U **SELECT** nemůže označovat jméno sloupce
  - U stavových tabulek dělených prostřednictvím **PERIOD** se vyhodnotí jako časový úsek (perioda)



# Práce s časem platnosti

- **Operátory pro elementy času, periody, ... je možné libovolně uplatnit na čas platnosti za SELECT, WHERE a HAVING.**



## Dotazy 4.1

- Jaké kontroly byly prováděny v ČKD v roce 1998?
  - SELECT Kontrola  
VALID INTERSECT(VALID(Kontroly),  
PERIOD „[1994]“ DAY)  
  
FROM Kontroly  
WHERE Kde = „ČKD“
- Výsledek:
  - Seznam typů kontrol, každá se seznamem období, kdy byly tyto kontroly v ČKD prováděny v roce 1998



## Dotazy 4.2

- Čas platnosti lze určit i u vkládané informace
  - INSERT INTO Kontroly  
VALUES („Škoda“, „Novák“, 3, ISO9000,  
INTERVAL „92“ DAY)  
VALID PERIOD „[2001/01/01 – 2002/12/31]“
- Výsledek:
  - Vložené hodnoty budou sdruženy s již existujícími řádky se stejnými hodnotami





## Základní čas platnosti

- Pokud není nic udáno, tak se vkládá tato specifikace:
  - `VALID PERIOD(CURRENT_TIMESTAMP, NOBIND(TIMESTAMP „now“))`
- Příkaz pro vložení založený na poddotazu je zpracován stejně



# Mazání na základě času

- **Odstraňuje se pouze specifikovaná časová informace (čas. sloupec)**
  - **DELETE FROM Kontroly**  
**WHERE Kde = „ČKD“**  
**VALID PERIOD „[1999/01/01 – 1999/12/31]“**
- **Jestliže se touto operací časová složka vyprázdní, tak jsou smazána i ostatní data**



# Sloupce vs. řádky

- **Změna položky ve sloupci podobně jako u SQL**
- **Změna času platnosti i pro jeden řádek (jako SQL + VALID PERIOD)**

# Průběh změn

Původně



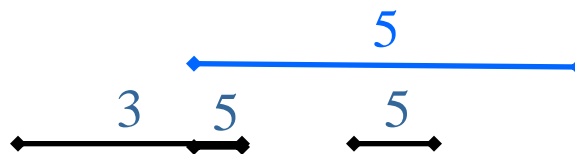
DELETE  
Výsledek



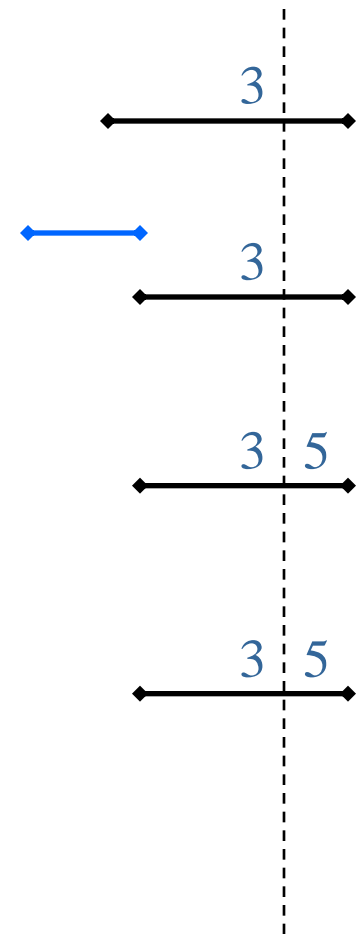
UPDATE  
Výsledek



UPDATE v.t.  
Výsledek



Současnost





# Událostní tabulky

- **Události dostávají časová razítka v podobě množin okamžiků**
- **Každý řádek udává určitou množkovou událost. Časové razítko přiřazené řádku říká, kdy daná událost proběhla**
- **Tyto tabulky mohou být také propojeny s časem transakce**



# Vytvoření událostní tabulky

- **CREATE TABLE**  
Laboratoř(Jméno, Lékař, TestID)  
**AS VALID EVENT HOUR**  
**AND TRANSACTION**
- **Úpravy ve FROM**
  - **FROM Laboratoř VALID(Laboratoř)**
    - množina okamžiků
  - **FROM Laboratoř(INSTANT) VALID(Lab.)**
    - jeden okamžik (den-čas)



## Zástupci (surrogates)

- Je to jedinečná hodnota, která může být testována na rovnost, ale jinak není viditelná
- Vhodné tam, kde je třeba identifikovat objekty a přitom klíč je časově závislý
- Jedná se o datový typ pro sloupec tabulky



# Vytvoření tabulky se zástupcem

- **CREATE TABLE Pacienti  
(Jméno CHAR, ID SURROGATE)  
AS VALID DAY**
- **INSERT INTO Pacienti  
VALUES („Tomáš“, NEW)  
VALID PERIOD „[1990/01/01 - 1994/01/01]“**
  - Klíčové slovo NEW zaručuje vytvoření jedinečné hodnoty pro zástupce





# Agregační funkce

- Jsou aplikovány na každý snímek v mezivýsledku
- Je možné vyhodnocení do určité míry optimalizovat, neboť řada řádků může nést stejné hodnoty



## Dotazy 5.1

- Kolik typů kontrol se provádí v ČKD?
  - `SELECT COUNT(*)`  
`FROM Kontroly`  
`WHERE Kde = „ČKD“`
- Výsledek:
  - Tabulka časů platnosti s časově závislou hodnoty sumy počtu typu kontrol platnou v daném čase



## Dotazy 5.2

- V kolika místech se provádějí jednotlivé kontroly?
  - **SELECT Kde, COUNT(\*)**  
**FROM Kontroly**  
**GROUP BY Kde**
- **Výsledek:**
  - Opět temporální tabulka.



## Dotazy 5.3

- TSQL2 definuje agregační funkci RISING, která určuje nejdelší časový úsek, po který nějaká hodnota rostla.
  - `SELECT SNAPSHOT RISING(Počet)`  
`FROM Kontroly`  
`WHERE Jméno=„Škoda“`  
`AND Kontrola=„IEEE“`
- Výsledek:
  - Úsek, kde bylo třeba ve Škodovce nasazovat na kontrolu IEEE stále více lidí.



# Časová zrnitost

- ***Hrubost*** časového dělení
- **Formuje svaz, který umožňuje přechod mezi úrovněmi**
- **Jsou definovány kalendářem**
  - SET CALENDRIC SYSTÉM SQL-92
- **Zrnitost je dána vloženým údajem**
  - TIMESTAMP „1994-04-19 15:24“ MINUTE



## Časová zrnitost (pokr.)

- Operandy predikátů musejí mít stejnou zrnitost
- Vyhodnocení se provádí na implicitní úrovni zrnitosti
- Operace se mění se zrnitostí operandů
- **CAST(*výraz AS zrnitost*)**



# Časová neurčitost

- **Nepřesné informace (o čase)**
  - mezi 10:00-12:00
  - asi na konci září
- **Souvisí se zrnitostí**
  - první týden v lednu 1995
    - na dny neurčité, na týdny OK
- **Vliv na jazyk**
  - neurčitost v modelu, literálech, predikátech, konstruktorech



# Pojem současnosti

- **SQL 92**
  - **CURRENT\_DATE, CURRENT\_TIME, CURRENT\_TIMESTAMP**
  - nelze uložit do DB, vyhodnoceny v průběhu dotazu
  - Erik je právě nový pacient
    - **INSERT INTO Pacienti  
VALUES („Erik“, NEW)  
VALID PERIOD (CURRENT\_DATE, ??)**
    - **TSQL2 ukládá údaje vztažené k současnosti**





# Doba transakce

- Pokud není explicitně zadáno jinak, je výsledkem to, co se právě považuje za správnou informaci
  - Jaké kontroly kdy byly v ČKD?
    - `SELECT Kontrola`  
`FROM Kontroly`  
`WHERE Kde = „ČKD“`



# Možnost *návratu* v DB

- **Jaké kontroly byly v ČKD provedeny k 1.1.1999?**
  - **SELECT Kontrola  
FROM Kontroly AS P  
WHERE Kde = „ČKD“  
AND TRANSACTION(P) OVERLAPS DATE „1999-01-01“**
- **Je možné ověřovat, kdy byla data měněna i zpětně.**



# Verzování DB schématu

- **SQL-92 neumožňuje (ztráta)**
- **TSQL2 při změně (DB transakční či obojího času) uchová verzi (~CVS)**
  - schéma se stane množinou tabulek s časem transakce
- **Návrat k datu**
  - **SET SCHEMA DATE „1995-04-19“**



# Odsávání

- **Data s časem transakce jsou „append-only“, takže pořád rostou**
  - Mohou přerůst datový prostor
  - Data je třeba uchovávat minimálně po nějakou dobu (zákon)
  - Více dat zpomaluje odezvu počítače
- **Je tedy třeba *odsát* data, která již nejsou potřeba**



## Co odsávání dělá

- **Odstraňuje stará, pravděpodobně zastaralá a nekorektní data**
  - data, jejich čas transakce končí „dnes“  
nebudou nikdy zrušena
  - nechtěné ale stále platné hodnoty  
mohou být odstraněny pomocí **DELETE**
  - probíhá asynchronně po příkazu **ALTER**



## TSQL2 - Shrnutí

- ***Stavové* tabulky jsou označovány razítky s časovými elementy, které jsou tvořeny množinami časových úseků**
- **Konvenční tabulky obdržíme aplikací klíčového slova **SNAPSHOT****



## TSQL2 - Shrnutí

- Časové úseky (periody) jsou určité intervaly v čase – nový datový typ
- V dotazech je možné
  - sdružovat časovou informaci u řádků, které jsou jinak shodné
  - využití spojení času s určitými jmény
  - dělení (partitioning) na základě maximálních časových úseků



## TSQL2 - Shrnutí

- Vybírat řádky na základě času platnosti
- Využití času platnosti i v projekci k tvorbě tabulky (v rámci dotazu)
- Událostní tabulky jsou razítkovány množinou časových okamžiků
- Tabulky s obojím časem obsahují jak čas platnosti, tak čas transakce





## TSQL2 - Shrnutí

- **Zástupci identifikují objekty tam, kde se klíč mění s časem**
- **Zrnitost určuje rozdělení časové osy**
- **Zrnitosti se sdružují do kalendářů**
- **Datový model podporuje časovou neurčitost**
- **Při vyhodnocení se časová informace vztahuje k současnosti**



## TSQL2 - Shrnutí

- **Agregační funkce a seskupování mohou být aplikovány na časové údaje**
- **Volba času transakce umožňuje návrat k předchozí verzi**
- **Je umožněno i verzování schématu**
- **Z tabulek s časem transakce mohou být odsáty staré verze**



# 10. Deduktivní databáze - úvod



# Co vlastně chceme?

- **Systém, který poskytne matematickou logiku ve spojení s *relačním DB modelem***
  - DB na úrovni FOL
  - analýza reprezentace znalostí a vztahu této reprezentace k dedukci



## Jinými slovy

- **DB obsahuje fakta, která odrážejí nějakou skutečnost, ale určitý druh informace je třeba z dat odvodit na základě *daných odvozovacích pravidel*.**



# Možná řešení

- **Vložit odvozovací pravidla do aplikace**
  - špatně čitelná
  - taková aplikace velmi náročná, navíc každá změna je *drahá*
- **Použít deduktivní DB systém**
  - Data jsou
    - odvozovací pravidla
    - samotná data (fakta)



# Použití deduktivního systému

- **Data**
  - explicitní (fakta)
  - odvozovací (deduktivní) pravidla
  - odvozená data
    - vznikají z explicitních použitím odvozovacích pravidel



# Rysy deduktivního DB systému

- **Zpracovává obrovské množství dat**
  - Jako jiné DB systémy
  - Mnohem více v porovnání s Prologem
- **Nad daty umožňuje aplikovat dedukční (logická, odvozovací) pravidla**
  - Jako v Prologu
  - Není běžné v DB systémech, nebo daleko za běžným rámcem





# Různost logických systémů

- Definice *jazyka* jako množiny správně definovaných formulí (wffs)
- Odlišnost ve způsobu dedukce
  - sémantický přístup
    - platnost formule
  - syntaktický přístup
    - konstrukce důkazu formule



# Jaký systém je nejlepší?

- **Možnost a snadnost implementace**
- **Snadnost použití**
- **Řešení typických DB problémů**
  - rekurze, optimalizace dotazů, ...



# Predikátová logika (FOL)

- **Syntaxe - viz temporální DB**
- **Transformace**
  - **Prenexová forma**
    - $u\check{c}í(x,y) \rightarrow (\exists z)diplomka(x,z)$
    - $(\exists z)(diplomka(x,z) \vee \neg u\check{c}í(x,y))$
  - **Skolem forma**
    - $\neg diplomka(x,f(x,y)) \wedge u\check{c}í(x,y)$



# Sémantika FOL

- **Teorie**

- mějme jazyk, potom teorie lze definovat pomocí
  - množiny axiomů
  - inferenčních pravidel

- **FOL**

- wffs + modus ponens + zobecnění
  - jestli  $P \wedge (P \rightarrow Q)$  potom  $Q$
  - z  $P$  odvod'  $(\forall x)P$



# Dokazování

- **Syntaktický přístup**
  - uplatňuje inferenční pravidla na axiomy všemi možnými způsoby
    - zespoda-nahoru: od hypotézy k teorému
    - shora-dolů: od teorému k axiomům
    - ? ukončení
- **Sémantický přístup**
  - vlastnost nesplnitelnosti



# Sémantický přístup

- **Interpretace**
  - pravdivostní hodnoty atomickým formulím
- **(Herbrandův) model**
  - interpretace, která je pravdivá pro každou formuli - model
  - $W = \{ \text{člověk}(\text{lva}), \text{člověk}(x) \rightarrow \text{smrtný}(x) \}$
  - $I_1 = \{ \text{člověk}(\text{lva}), \text{smrtný}(\text{lva}) \}$
  - $I_2 = \{ \text{člověk}(\text{lva}), \text{člověk}(\text{Pepa}), \text{smrtný}(\text{Pepa}) \}$



# Logický důsledek

–  $w$  : wffs

$K$  : množina wffs

$K \models w \Leftrightarrow$

$w$  je pravdivé pro všechny modely  $K$

- **Splnitelnost**

- existence modelu pro formuli

- **Platnost**

- splnitelná ve všech interpretacích



# Automatizovaný důkaz

- **PROLOG**
  - Hornovy klauzule
  - SLD rezoluce
  - viz PRJ, UIN

---

- *Rozšíření FOL*
  - mazání a unifikace
  - faktorizace klauzulí





# Relační model

- **Codd - 1970**
- **Relace**
  - $D_1, \dots, D_n$  - domény  
 $R \subseteq D_1 \times D_2 \times \dots \times D_n$   
 $R = \{ (d_1, \dots, d_n) \mid d_i \in D_i, 1 \leq i \leq n \}$
  - klíč
  - normální formy
  - integritní omezení



# Relační algebra

- **Selekce -  $\sigma$**
- **Projekce -  $\Pi$**
- **Spojení -  $\bowtie$**



# DML

- **SQL**
- **Dotazy**
  - otevřené - odpověď typu ano/ne
  - uzavřené - výsledkem je množina n-tic
- **Pohledy**



# Relační model nad logikou

- **Relace je formule z FOL**
  - osoba(ID, jméno, věk, plat)  
osoba(x, „Karel“, 35, ???)
- **Funkce jsou speciálním případem relací**
  - $y = f(x) \Leftrightarrow F(x, y)$   
 $F(x, y) \wedge F(x, z) \rightarrow (y = z)$



# Různý druh informace

- **explicitně uložené predikáty (EUP)**
  - explicitní data, fakta
    - `osoba(1132, „Pepa“, 45, 34500)`
- **odvoditelně uložené predikáty (OUP)**
  - odvozená data
    - když věk > 34 potom plat = 45000
    - `osoba(x,y,z,45000) :- osoba(x,y,z,w), z>34.`



# Význam pravidel

- **3 interpretace**
  - důkazní (teoretická)
  - modelová (teoretická)
  - výpočetní



# Důkazní interpretace - syntaktická

- **Axiomy**
  - explicitně uložená informace
    - věk(Pepa, 35)
  - implicitní informace, která může být odvozena z EUP či OUP
- **Negace**
  - pozitivní i negativní predikáty



# Důkaz

- Všechny fakty odvozené pomocí OUP jsou odvozeny pomocí modus ponens
  - $\text{osoba}(1234, \text{Pepa}, 44, x)$   
 $\text{osoba}(x,y,z,45000) \text{ :- } \text{osoba}(x,y,z,w), z > 34.$ 

---
  - ...  $\text{osoba}(1234, \text{Pepa}, 44, 45000)$
- uplatňuje se dopředné odvození
  - od axiomů k teorému





# Modelová interpretace - sémantická

- pravidla definují možné modely
- interpretace: predikát přiřazuje pravdu či nepravdu každé možné instanci
- modelem množiny pravidel je interpretace, ve které jsou všechna pravidla pravdivá - nezávisí na přiřazení hodnot proměnným



# Příklad

- **Necht'**

- $p(x) \text{ :- } q(x) \quad (1)$

- $q(x) \text{ :- } r(x) \quad (2)$

- $\text{DB: } \{r1\}$

- **Universum: celá čísla**

- **Interpretace**

- $\{ r(1), q(1), p(1), q(2), p(2), p(3) \}$

- $\{ r(1), q(1), p(1) \}$

- $\{ r(1), q(2), p(2) \}$

- **? modely, ? minimální modely**



# Minimální model

- Necht'  $M_1, \dots, M_n$  jsou modely množiny  $S$  wffs. Minimální model je takový model  $M_k$ , který:
  - $M_k \subseteq M_j, j \in \{1, \dots, k-1, k+1, \dots, n\}$
  - $\neg \exists M, M$  je modelem  $S$  a  $M \subseteq M_k$



# Uspořádání modelů

- Hledáme takovou permutaci , že
  - $M_k = M_{i_1}$ , a
  - $M_{i_1} \subseteq M_{i_2} \subseteq \dots \subseteq M_{i_n}$
- Výhoda
  - z  $M_k$  je možné odvodit ostatní modely
  - $M_k$  obsahuje řešení



# Problém - negace

- $p(x) :- r(x) \wedge \neg q(x)$   
 $q(x) :- r(x) \wedge \neg p(x)$
- DB:  $\{r(1)\}$
- Dva minimální modely
  - $S_1 = \{q(1), r(1)\}$
  - $S_2 = \{p(1), r(1)\}$




# Výpočetní interpretace

- **Algoritmus na „vykonání“ pravdy/nepravdy pravidel**
- **PROLOG**
  - algoritmus na vyhledání důkazu potencionálního faktu
  - to co PROLOG považuje za pravdivé fakty, nemusí být vždy modely
  - v mnoha případech poskytne minimální model



# Způsob integrace DB a Prologu

- **Prolog v základech (upravený)**
- **Pravidla jsou posloupnosti operací relační algebry**
- **Bez negací**
  - výsledkem je jediný minimální model a množina faktů odvoditelných z DB
- **S negacemi (omezené možnosti)**
  - jeden z mnoha minimálních modelů



# Vnitřní model - možná varianta

- **Modelová interpretace**
- **Prologovské příkazy:**
  - atomické formule, predikátové symboly, funkce počítající hodnotu
- **Prologovské konvence**
  - predikáty, funkce a konstanty začínají malými písmeny
  - proměnné začínají velkými písmeny





# Vnitřní model (dokončení)

- **Logické příkazy (výrazy)**
  - Hornovy klausule
  - $B :- A_1, A_2, \dots, A_n$
- **Příklad - boss**
  - $\text{boss}(M,E) :- \text{řídí}(M,E).$
  - $\text{boss}(M,E) :- \text{boss}(N,E), \text{řídí}(M,N).$

- $\text{sum}(X, 0, X).$        $X + 0 = X$
- $\text{sum}(X, s(Y), s(Z)) \text{ :- } \text{sum}(X, Y, Z).$        $X + Y = Z$
- $X + (Y + 1) = (Z + 1)$

# Příklad - DB zaměstnanců

- | jméno | oddělení | plat | adresa |
|-------|----------|------|--------|
| Pepa  | aplikace | 35K  | Brno   |
| Iva   | vývoj    | 45K  | Vyškov |

**DB**  
zaměstnanci

- | jméno | oddělení | adresa |
|-------|----------|--------|
| Pepa  | aplikace | Brno   |
| Iva   | vývoj    | Vyškov |

**DB**  
zaměstnanci  
bezpečná

- bezpečnou DB je možné vypočítat
  - bezpečná(J,O,A) :- zaměstnanec(J,O,P,A), P>30K.



# Co dál lze zjistit...

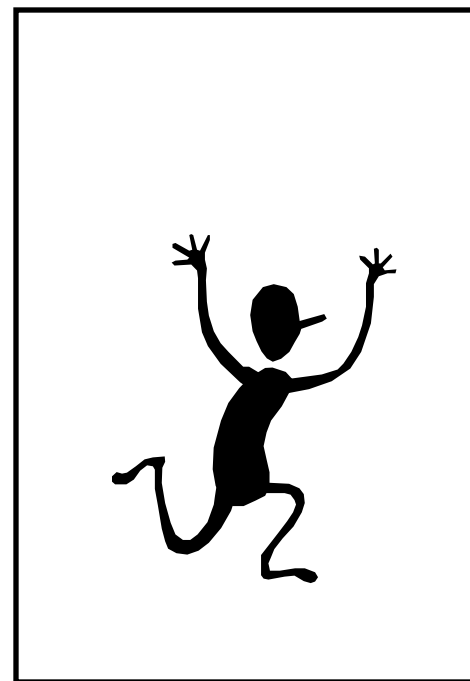
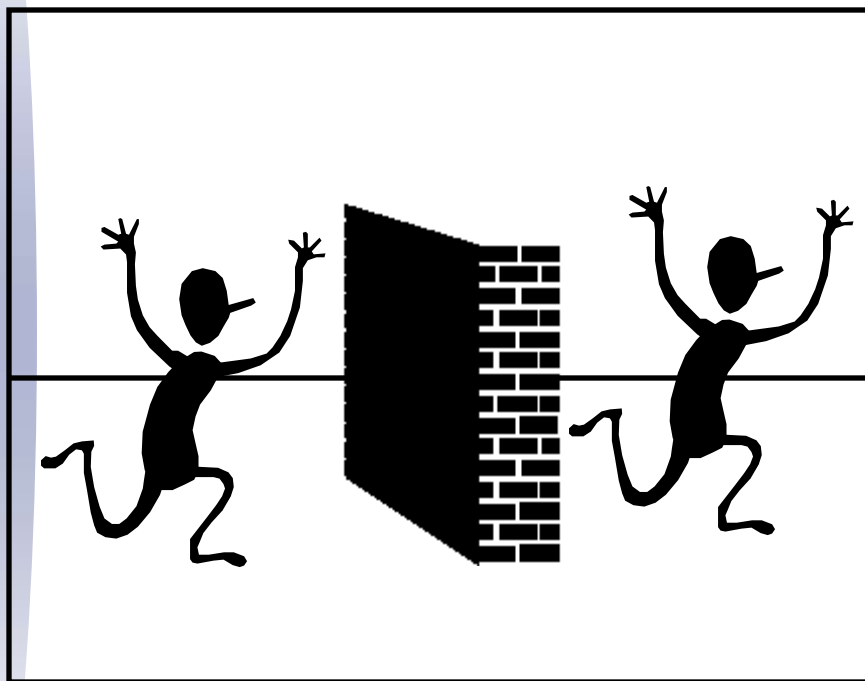
- **řídí(E,M) :- zaměstnanec(E,D), oddělení(D,M).**
- **pohledy - řídí, bezpečná**
- **Dotaz je „druhem“ pohledu**
  - Najdi ředitele (nadřízeného) Pepy Vosáhla
  - řídí(X, 'Pepa Vosáhlo')
- **SQL v roli DML**
  - nedostačující pro transitivní uzávěry - boss



## Příklad - obrázky

- ukládání obrázků tvořeného buňkami
- konstrukce obrázků z buněk
  - buňky jsou čb
  - bitmapové obrázky
  - jen č/b
  - buňky mohou obsahovat jiné buňky, které jsou třeba v hostitelském systému posunuty do „jiného“ místa

# Buňky





# Obrázky atd.

- **Složení obrázků je očividné**
- **Stromová struktura**
- **Figurky mohou být též složeny**
  - **rekurze na každé úrovni**
    - **paměťové problémy**
  - **ukládání obrázků hned po vytvoření**
    - **DML**



# Reprezentace v Prologu

- **nastaven(I,X,Y)** - pixel na pozici X,Y je nastaven (černý/bílý)
- **obsahuje(I,J,X,Y)** - objekt I obsahuje kopii objektu J na pozici X,Y v rámci objektu I
- **svítí(I,X,Y)** - bod X,Y je nastaven přímo, nebo přes další objekty





# Ukázka implementace

- **svítí(I,X,Y) :- nastaven(I,X,Y).**  
**svítí(I,X,Y) :- obsahuje(I,J,U,V),**  
**svítí(J,W,Z),**  
**X=U+W, Y=V+Z.**



# Shrnutí

- **predikáty definují relace**
- **predikáty obsahují navíc položky, které nejsou atributy (ID)**
- **DB jako**
  - **EUP - relace v DB**
  - **vestavěné predikáty**
  - **OUP - Hornovy klauzule**



# 11. Deduktivní databáze - algoritmy



# Tvar atomických formulí

- $p(A_1, \dots, A_n)$ 
  - predikátový symbol, proměnná, konstanta, arita
  - definuje relaci, přičemž ji může zúžit
    - konstanty ~ porovnání na konstantu
    - unifikace proměnných ~ porovnání proměnných
  - jména atributů nejsou známa
  - je třeba si pamatovat pozici a význam



# Příklad

- **zákazník(Pepa, adresa, kredit)**
  - **zákazník(jméno, adresa, kredit)**
  - $\sigma_{\$1=\text{Pepa}}(\mathbf{ZÁKAZNÍK})$
- **stav\_skladu(X, položka, X)**
  - **stav\_skladu(lč, položka, počet)**
  - $\sigma_{\$1=\$3}(\mathbf{stav\_skladu})$



# Hornovy klauzule

- **Fakty**
  - jeden pozitivní literál
    - $p(X,Y)$
- **Integritní omezení**
  - jeden, či více negativních literálů
    - $\text{not}(\text{věk} \geq 17)$
- **Pravidlo**
  - pozitivní a negativní literál(y)
    - $p_1 \wedge \dots \wedge p_n \rightarrow p$



# Bezpečná pravidla

- **Obsahují na pravé straně proměnné z levé strany**
  - $p(X,Y) :- q(X,U), r(U,Y).$
  - je možné je přeložit jako selekce a spojení



# Logický program

- je tvořen souborem Hornových klauzulí
- Pracovní příklad
  - *$X$  a  $Y$  jsou sourozenci, pokud existuje takové  $Z$ , že je rodičem obou individuí a zároveň  $X$  a  $Y$  jsou různá individua*





# Pracovní příklad

- **sourozenci(X,Y) :-**  
    **rodič(Z,X), rodič(Z, Y), X /= Y.**
- **b\_s(X,Y) :-**  
    **rodič(Xp, X), rodič(Yp, Y), sourozenci(Xp, Yp).**
- **b\_s(X,Y) :-**  
    **rodič(Xp, X), rodič(Yp, Y), b\_s(Xp, Yp).**
- **příbuzní(X,Y) :- sourozenci(X,Y).**
- **příbuzní(X,Y) :- příbuzní(X,Z), rodič(Z,Y).**
- **příbuzní(X,Y) :- příbuzní(Z,Y), rodič(Z,X).**



# Konverze logických programů

- **Je třeba zvolit vhodné DB struktury pro reprezentaci různých informací**
  - EUP (OK)
  - OUP
  - závislost mezi predikáty
    - určují algoritmus



# Závislost

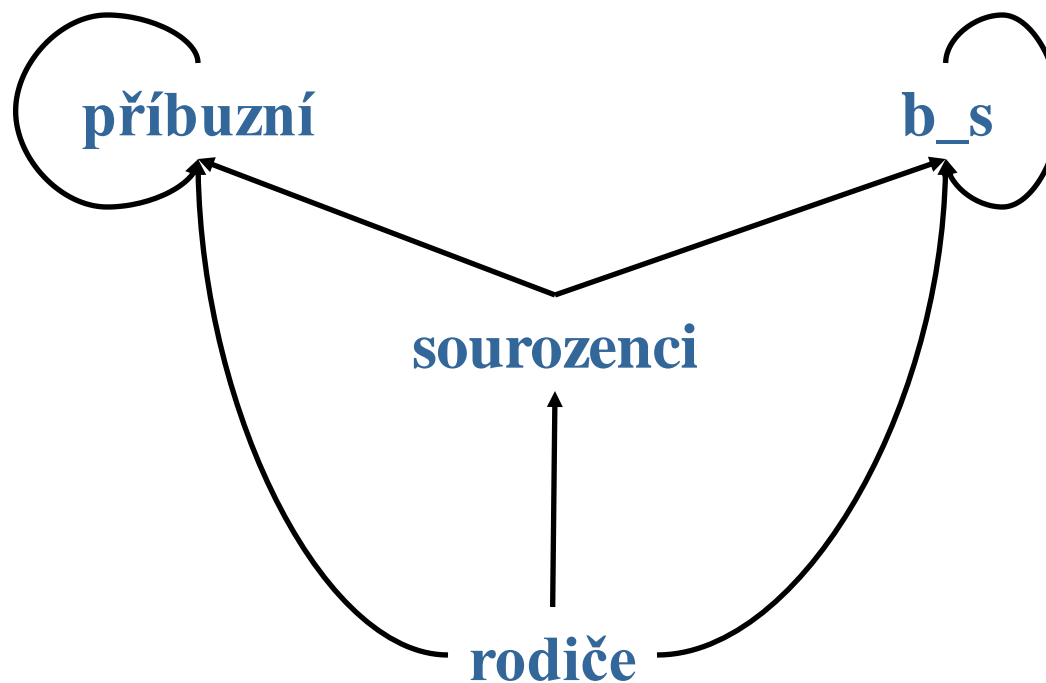
- **Orientovaný graf závislosti  $G$  je dvojice  $G=(V,E)$ , kde**
  - $V$  je množina predikátů (uzly)
  - $E: (p,q) \in E$  jestliže existuje pravidlo  $r$  takové, že  $q$  je hlava pravidla a  $p$  jsou podcíle



# Význam grafu

- **Graf určuje**
  - zda je program rekurzivní (obsahuje alespoň jeden cyklus)
  - které predikáty a jak jsou rekurzivní
  - EUP jsou nerekurzivní

# Pracovní příklad





# Stačí bezpečné predikáty?

- **Odstraňují problém ,volných‘ proměnných**
  - $p(X,Y) \text{ :- } p(Y).$
- **Neřeší problémy s vestavěnými predikáty, které mohou být zdrojem nekonečnosti**
  - $\text{větší\_než}(X,Y) \text{ :- } X \geq Y.$



# Omezované proměnné


- Každá proměnná, která je na pravé straně pravidla je omezovaná
- Každá proměnná porovnávaná s konstantou na rovnost je omezovaná
- Každá proměnná, která je porovnávaná na rovnost s omezovanou proměnnou, je omezovaná



# Redefinice bezpečných pravidel

- Pravidlo je *bezpečné*, pokud jsou všechny jeho proměnné omezované.
- **Příklad**
  - $p(X,Y) :- q(X,Z), W=a, Y=W.$ 
    - ? je bezpečné
    - počítá  $\Pi(q) \times \{a\}$
    - pravidlo je konečné, pokud  $Q$  je konečné





# Vyhodnocení nerekurzivních pravidel

- nesmí obsahovat negaci
- programy
  - přeloženy do relační algebry
  - OUP jsou modelem pro pravidla
- pro nerekurzivní program je možné uspořádat pravidla tak, že pokud v  $p_1, \dots, p_n$  platí  $p_i < p_j$ , potom vede cesta z  $i$  do  $j$



# Vyhodnocení OUP

- Pro každé pravidlo  $r$  s hlavičkou  $p_i$  je spočtena relace těla predikátu. Jde o operaci spojení výsledků všech podcílů.
- Výpočet samotného  $p_i$  je projekcí relace těla predikátu na proměnné korespondující s hlavičkou s tím, že dochází ke sjednocení výsledků pro všechny kombinace.



# Relace definovaná tělem pravidla

- relace  $r$  pro  $q :- p_1, \dots, p_n$ .
- $P_1, \dots, P_n$  jsou relace pro  $p_1, \dots, p_n$
- $P_j = \{(a_1, \dots, a_n) \mid p_j(a_1, \dots, a_n) \text{ je splněno}\}$
- podcíl  $S$  je splněn pokud
  - je-li to běžný podcíl (EUP čí OUP), potom  $S$  je tvaru  $p(b_1, \dots, b_n)$  a  $(b_1, \dots, b_n)$  je  $n$ -tice v relaci  $P$  korespondující k  $p$
  - je-li to vestavěný podcíl potom  $S$  je aritmetická relace, která je splněna



# Příklad 1

- **$b\_s(X,Y) :- \text{rodič}(Xp,X), \text{rodič}(Yp,Y), \text{sourozenci}(Xp,Yp).$** 
  - rodič a sourozenci jsou už vypočtení
  - $R(X,Xp,Y,Yp) = P(Xp,X) \star P(Yp,Y) \star S(Xp,Yp)$
  - Necht' n-tice  $R$  je tvaru  $(a,b,c,d)$ , potom
    - $(a,b)$  je v  $P$
    - $(c,d)$  je v  $P$
    - $(a,c)$  je v  $S$



## Příklad 2

- **sourozenci(X,Y) :- rodič(Z,X), rodič(Z,Y),  
X  $\neq$  Y.**
  - $Q(X,Y,Z) = \sigma_{X \neq Y}(P(Z,X) \star P(Z,Y))$
  - Q obsahuje n-tice (x,y,z), kde
    - (z,x) je v P
    - (z,y) je v P
    - $x \neq y$



## Příklad 3

- $p(X,Y) :- q(a,X), r(X,Z,X), s(Y,Z).$ 
  - relace pro  $q(a,X)$  je  $T(X) = \Pi_2(\sigma_{\$1=a}(Q))$
  - relace pro  $r(X,Z,X)$ :  $U(X,Z) = \Pi_{1,2}(\sigma_{\$1=\$3}(R))$
  - relace pro  $s(Y,Z)$ :  $S(Y,Z)$
  - $T(X) \Join U(X,Z) \Join S(Y,Z)$  potom obsahuje n-tice  $(x,y,z)$  takové, že
    - $(a,x)$  je v  $Q$
    - $(x,z,x)$  je v  $R$
    - $(y,z)$  je v  $S$



# Algoritmus

- **Vstup**

- tělo pravidla  $r$  s podcíli  $S_1, \dots, S_n$ , které generují proměnné  $X_1, \dots, X_m$ . Každé  $S_i = p_i(A_{i1}, \dots, A_{iki})$  má již spočtenou svou relaci  $R_i$  a  $A$  jsou její parametry - proměnné, či konstanty

- **Výstup**

- Výraz relační algebry, který vypočte relaci  $R(X_1, \dots, X_m)$  s  $n$ -ticemi  $(a_1, \dots, a_n)$ , kde při substituci  $a_j$  za  $X_j$  budou všechny podcíle  $S_1, \dots, S_n$  splněny



# Metoda

- Pro každé běžné  $S_i$  je  $Q_i$  ve tvaru  $\prod_{V_i}(\sigma_{F_i}(R))$  kde
  - $V_i$  je množina proměnných z  $S_i$
  - $F_i$  je konjunkce
    - je-li na pozici  $k$  v podcíli  $S_i$  konstanta  $a$ , potom  $F_i$  obsahuje ( $k=a$ )
    - jsou-li na pozicích  $k$  a  $l$  v podcíli  $S_i$  stejné proměnné, potom  $F_i$  obsahuje ( $k=l$ )
- Pro každou proměnnou  $X$ , která se nenachází v běžných podcíchích určí doménu  $D_x$ , která definuje unární relaci obsahující všechny hodnoty, kterých může  $X$  nabývat. Je-li  $r$  bezpečná, potom musí existovat  $Y$  takové, že  $X=Y$  je alespoň jeden z podcílů...





# Metoda (pokr.)

- ... a zároveň  $Y$  je buďto konstanta nebo parametr běžného podcíle
  - $Y = a$ , potom  $D_x = \{a\}$
  - je-li  $Y$   $j$ -tý argument podcíle  $S_i$ , potom  $D_x$  je  $\prod_j(R_i)$
- Necht'  $E$  je normální spojení všech  $Q_i$  z prvního bodu a všech  $D_x$  z druhého bodu. Existence příslušných atributů je dána příslušnými proměnnými z podcílů.
- Vyhodnocení relace  $r$  je potom  $\sigma_F(E)$ , kde  $F$  je konjunkce aritmetických podcílů pro každý podcíl v  $r$  a  $E$  je výraz ze třetího bodu. Nejsou-li vestavěné predikáty, potom zůstává jen  $E$ .



# Příklad

- $p(X,Y) :- q(a,X), r(X,Z,X), s(Y,Z).$
- podcíl  $s_1: q(a,X)$ 
  - $Q_1 = \Pi_2(\sigma_{\$1=a}(Q))$
- podcíl  $s_2: r(X,Z,X)$ 
  - $Q_2 = \Pi_{1,2}(\sigma_{\$1=\$3}(Q)) = U(X,Z)$
- podcíl  $s_3: s(X,Z)$ 
  - $Q_3 = S(X,Z)$



## Věta

- **Uvedený algoritmus je korektní v tom, že výsledná relace  $R$  obsahuje pouze takové  $n$ -tice, kdy substituce za proměnné v podcílích vedou vždy ke splnění těchto podcílů.**
- *Důkaz - samostatně pro zájemce*



# Pravidla s více pravými stranami

- **Vezmi v úvahu všechny predikáty s p v hlavičce**
- **Vypočti jejich těla**
- **Proved' projekci na proměnné z hlavičky**
- **Sjednot' výsledky**



# Problém

- **Konstanty v hlavičce**
  - $q(a,z)$
- **Opakované proměnné**
  - $q(X,Z,A,X)$



# Upravené predikáty

- **Upravený predikát k predikátu p je nový predikát  $p(X_1, \dots, X_k)$ , kde**
  - $X_j$  jsou různé proměnné
  - pokud je třeba, jsou zavedeny nové
  - a na nich jsou vestavěny další podcíle



# Algoritmus úpravy

- $p(Y_1, \dots, Y_k)$  - vstup
- $p(X_1, \dots, X_k)$  - výstup
  - pro konstanty jsou přidány podcíle
    - $X_i = a$
  - pro proměnné  $Y_i = Y_j$ 
    - $X_i = X_j$
  - $X_i$  a  $X_j$  jsou různé pokud  $i$  a  $j$  je různé



## Příklad

- $p(a,X,Y) \text{ :- } r(X,Y).$   
 $p(X,Y,X) \text{ :- } r(Y,X).$
- **Upravená pravidla**
  - $p(U,V,W) \text{ :- } r(V,W), U=a.$   
 $p(U,V,W) \text{ :- } r(V,U), W=U.$





# Lemma

- **Je-li  $r$  bezpečné, potom i upravené  $r$  je bezpečné.**
- **Pravidla  $r$  a upravené pravidlo  $r$  jsou splnitelná pro stejné hodnoty argumentů.**



# Výpočet relací pro nerekurzivní pravidla - predikáty

- **Princip**
  - upravená pravidla
  - postupná aplikace algoritmu
  - sjednocení výsledků
- **Algoritmus**
  - jen pro zájemce



# Další rozšíření

- **Rekurze**
  - není možné uspořádat predikáty - podcíle
- **Princip**
  - vyhodnocení EUP
  - postupné vyhodnocení OUP
    - jde o monotónní funkci a zastavení je provedeno v momentě, kdy už se počet n-tic výsledku nezvyšuje



# Optimalizace rekurze

- **Pevné body**
- **Určení SSK**



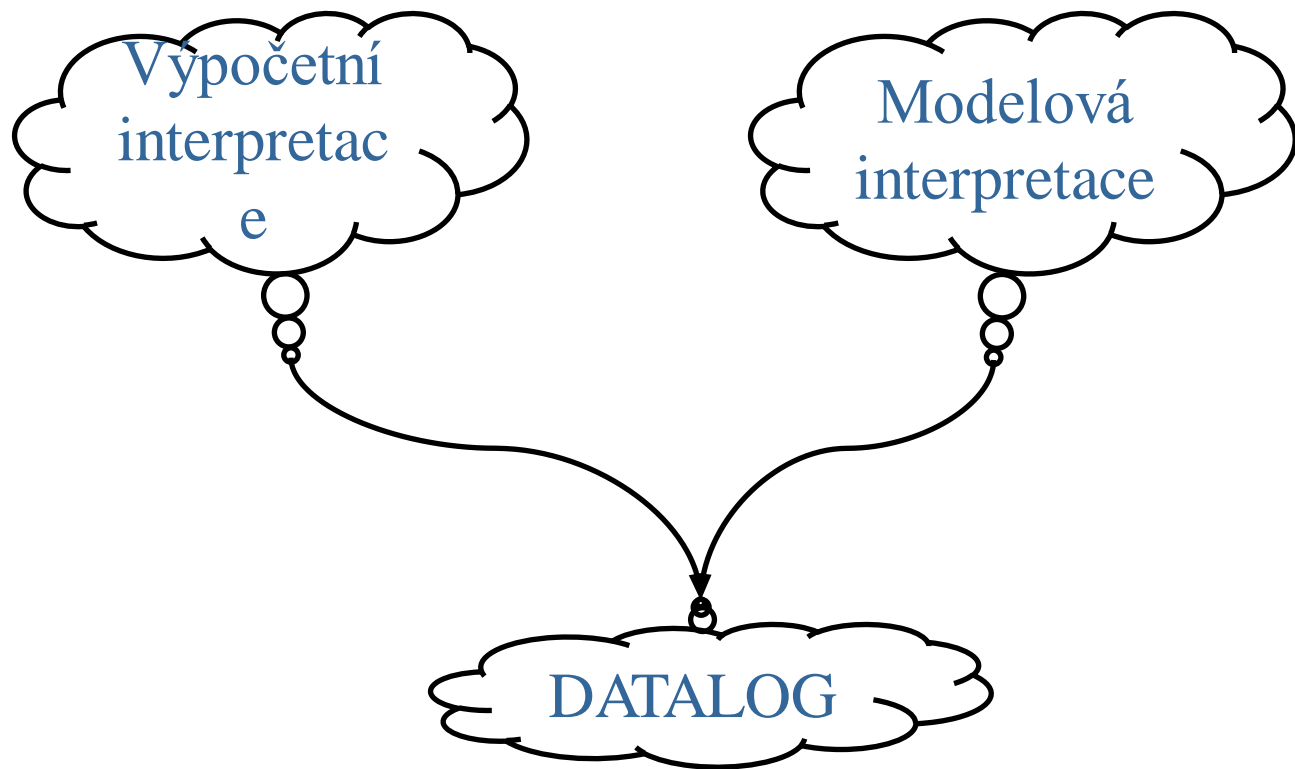
# DDB s negací

- **Problém**
  - více řešení
- **$T = C - S$** 
  - je nahrazeno
- **$T = C \star \text{☠} S$** 
  - kde  $\text{☠} S$
- **$\text{☠} S = U \times U - S$** 
  - U jsou všechny n-tice nadřazeného predikátu



## 12. Jazyk DATALOG

# DATALOG





# Klíčové vlastnosti

- **Relační jazyk bez negace s rekurzí**
- **Často nazýván „univerzitní“ bází, ale**
  - **techniky vyvinuté pro něj použity při optimalizaci SQL dotazů**
  - **některá SQL rozšíření mají limitovanou formu rekurzivních dotazů (DB2)**





# Syntaxe DATALOGu

- **Základem jsou Hornovy klauzule (PROLOG)**
  - řada vlastností (viz PRJ)
  - zejména absence funkčních symbolů
    - jen konstanty a proměnné na úrovni termů
- **Tvar pravidel:**
  - $H(v) \leftarrow B_1(u_1), \dots, B_n(u_n)$ 
    - H,B - predikáty; v,u - n-tice termů



# Omezení pravidel

- $H(v) \leftarrow B_1(u_1), \dots, B_n(u_n)$ 
  - $H$  je (musí být) odvozená informace/pravidlo
  - $B$  mohou být krom toho i fakta (samotná data)
  - všechny proměnné  $z$   $v$  se musí objevit i někde v rámci všech  $u$  (omezení rozsahu)
    - *zaručení konečnosti*



# Příklad

- **Databáze společnosti**
  - vedoucí oddělení členem jiného oddělení (ale nikoliv zase vedoucí)
  - detekce nadřízených - přímých i nepřímých



# Zápis v DATALOGu

- **Datová část**
  - Zaměstnanec(Zam#, Jméno, Zařazení, Plat, Prémie)  
Oddělení(Odd#, Název, Divize, Vedoucí#)  
ZamOdd(Zam#, Odd#)
- **Odvozovaná část**
  - Nadřízený(Vedoucí#, Zam#)
    - ← Zaměstnanec(Zam#, Jm, Zař, Plat, Prémie),  
ZamOdd(Zam#, Odd#),  
Oddělení(Odd#, Náz, Divize, Vedoucí#).
  - Nadřízený(Vedoucí1#, Zam#)
    - ← Zaměstnanec(Zam#, Jm, Zař, Plat, Prémie),  
ZamOdd(Zam#, Odd#),  
Oddělení(Odd#, Náz, Divize, Vedoucí#),  
Nadřízený(Vedoucí1#, Vedoucí#).



# Sémantika v DATALOGu

- **Velmi podobná logickým programům**
- **Jen malé rozdíly**
  - **Odvozovací pravidla se nemění (jako v PROLOGu)**
  - **Fakta (data) se mění - je třeba to podchytit jiným způsobem než v logických programech**



## Více formálně

- Necht' DDB je deduktivní DB, potom sémantika je funkce z množin faktů nad datovými predikáty ( $\text{Pred}^f$ ) do množin faktů nad ( $\text{Pred}^f \cup \text{Pred}^o$ ).
  - Informace získaná z deduktivní DB sestává z faktů explicitně uložených v DB (EDB~EUP) plus všechna odvozená fakta.



# Obvyklý zápis sémantiky

- Jelikož ta část DDB, která obsahuje odvozovací data (IDB~OUP) je považována za pevnou, potom sémantika

$$\text{DDB} = \text{IDB} \cup \text{EDB}$$

se často zapisuje jako

$$S_{\text{IDB}}(\text{EDB})$$



# Důsledek absence funkcí

- **DB v DATALOGu obsahuje konstanty jazyka (pouze)**
- **Jedná se o konečnou množinu => Herbrandovo universum také a Herbrandovy báze též**
- **Tudíž je sémantika DB DATALOGu vždy konečná množina**






# Jak modelovat sémantiku

- **Pro DATALOG v zásadě 3 možnosti**
  - Modelově teoretická
  - Operační - shora dolů
  - Operační - zdola nahoru



# Modelově teoretická

- **Důsledek v rámci DATALOG DB je definován v intencích**
  - interpretací
  - Herbrandových modelů
  - *což je stejné jako u logických programů*
- **Tedy:**
  - Minimální Herbr. Model reprezentuje modelově teoretickou sémantiku DB
    - Takový model je vždy konečný (nejsou fce)



# Operační - shora dolů

- **Použití SLD rezoluce (viz PRJ)**
  - Unifikace je jednodušší, neboť DATALOG nemá funkční symboly
- **Problémové části (tuple-at-a-time)**
  - Náchylné k rekurzivním smyčkám
  - Může provést opakované výpočty nějakých podcílů
  - Často prakticky nedetekovatelné, kdy byla nalezena všechna řešení dotazu



# Operační - zdola nahoru

- **Proces začíná od množiny faktů**
- **Iterace nejmenšího pevného bodu**
  - V každém kroku počítá novou množinu faktů, které je možné odvodit z už získaných faktů pomocí pravidel v IDB
  - V prvním kroku jsou fakta jen data uložená v EDB
  - Použití operátoru *okamžitých důsledků*
  - Model je potom nejmenším pevným bodem tohoto operátoru



## Zdola nahoru - vlastnosti

- **Jelikož sémantika DATALOGu je konečná, lze touto metodou obdržet výsledek v konečném čase**
- **Nemá stejné nedostatky jako metodika shora dolů**
- **Podporuje operace jako relační spojení (set-at-a-time)**
  - **Zvýšení efektivity**



# Dotazování

- Jako u logických programů, cíl:

$$\leftarrow A_1, \dots, A_n$$

- $A$  jsou atomy, nikoliv nutně základní



# Příklad

- **Nadřízení programátorů**
  - Nadřízený(Vedoucí#,Zam#),  
Zaměstnanec(Zam#,Jméno,programátor,Plat,Prémie).
- **Nadřízený Rossiho**
  - Nadřízený(Vedoucí#,Zam#),  
Zaměstnanec(Zam#,rossi,Zařazení,Plat,Prémie).



# Rozdíly PROLOG-DATALOG

- V Prologu nás často zajímá zda existuje *nějaké* řešení
- DATALOG: *všechna* řešení
  - Proto je preferována sémantika zdola nahoru = blíže DB chování (SQL)
  - Může poskytnout více výsledků, než co by stačilo na zodpovězení dotazu
    - vyvinuty optimalizační postupy, které tuto vlastnost do jisté míry eliminují





# Rozšíření DATALOGu

- **Základní verze není příliš použitelná pro tvorbu aplikací - chybí určité, jinak běžné, vlastnosti**
- **Dvě důležitá rozšíření (více později)**
  - **negace**
  - **vestavěné predikáty**



# Negace

- **DATALOG neumožňuje vyjádřit některé jednoduché dotazy jako např. rozdíl dvou relací**
- **Negace řadu těchto problémů překlenuje**
- **Avšak otevírá některé otázky směrem k sémantice a výpočetním modelům**



# Způsoby zavedení negace

- **Negace literálů (atomů) v rámci těla pravidel**
  - **DATALOG**  $\neg$
  - **Problémy řešeny**
    - omezeními na syntaktické úrovni (dekompozice a strukturalizace programu)
    - použitím jiné sémantiky



# Strukturalizace programu

- **Žádný predikát nesmí být negativně závislý sám na sobě**
  - Lze detekovat jednoduchým algoritmem
- **Avšak to znamená zamezení rekurzivní negace => snížení síly jazyka**
  - Hledala se jiná cesta



# Jiné sémantiky

- **WF sémantika:**
  - DB nemusí nutně znát odpověď ano/ne a každý fakt
  - Odpovídá se „nevím“



# Vestavěné predikáty

- **Porovnávání (proměnné s konstantou)**
  - $<$ ,  $>$ ,  $<=$ ,  $>=$
  - Nutné mít implementováno uvnitř
    - sémantika je systému implicitně známa
  - Může vést k nekonečným výsledkům pokud nejsou zavedeny specifické bezpečnostní podmínky



# Vlastnosti DB v DATALOGu

- **Rozhodnutelnost na základě statické analýzy IDB**
  - Obecná výhoda deklarativního programování - lepší „pochopení“ programu => lepší optimalizace a efektivnější vyhodnocení
    - Př. Neuspokojitelný predikát - možno odstranit predikáty, které jej používají, z IDB, nebo ohlášení chyby



# Shodnost/Obsažnost

- $IDB_1$  je obsažená v  $IDB_2$  ( $IDB_2 \subseteq IDB_1$ ) pokud pro nějakou EDB a predikát  $p$  (z  $IDB_1$ ) jsou odpovědi na  $p$  v  $IDB_1 \cup EDB$  obsaženy v odpovědích na  $p$  v rámci  $IDB_2 \cup EDB$ .
- Dvě odvozovací DB jsou shodné, pokud  $IDB_2 \subseteq IDB_1$  a zároveň  $IDB_1 \subseteq IDB_2$ .  
Píšeme:  $IDB_1 \approx IDB_2$





# Splnitelnost

- Predikát  $p$  z IDB je splnitelný jestliže existuje EDB taková, že  $p$  v  $IDB \cup EDB$  generuje neprázdnou množinu odpovědí.



# Omezenost

- **Rekurzivní  $IDB_1$  je omezená, pokud existuje nerekurzivní  $IDB_2$  taková, že  $IDB_1 \approx IDB_2$ .**



# Platnost vlastností

- Čistý DATALOG
- S negací
- S matematickými omezeními
- **Rozhodnutelnost těchto vlastností závisí na specifikách dané mutace DATALOGu**



# Shodnost/Obsažnost

- **Nerozhodnutelná pro**
  - DATALOG
  - DATALOG s IDB predikáty s aritou 2
  - DATALOG s IDB predikáty s aritou 1, negací, nerekurzivními predikáty, nerovností
- **Rozhodnutelná pro**
  - DATALOG se strukturovanou negací (predikáty nesmí negativně záviset sami na sobě) a EDB predikáty s aritou 1



# Splnitelnost

- **Rozhodnutelná pro**
  - **DATALOG**
  - **DATALOG s omezeními nad porovnání s konstantou**
  - **DATALOG s omezeními nad porovnání s konstantou, strukturovaná negace se aplikuje jen na EDB atomy**
  - **DATALOG se strukturovanou negací (predikáty nesmí negativně záviset sami na sobě) a EDB predikáty s aritou 1**



# Omezenost

- **Nerozhodnutelná pro**
  - DATALOG
  - Lineární DATALOG DB - maximálně jeden vzájemně rekurzivní odkaz v těle predikátu
- **Rozhodnutelná pro**
  - DATALOG s IDB predikáty s aritou 1



## Rozdíly mezi DDB a LP (1)

- DDB ukládá značné množství faktů a malinké množství odvozovacích pravidel. U LP tento rozdíl není.
- Predikáty v DDB jsou rozděleny na IDB a EDB. U LP toto dělení není.
- DDB neobsahují typicky funkce, pouze *omezené a vybrané*. U LP toto omezení není.



## Rozdíly mezi DDB a LP (2)

- DDB podporují integritní omezení (typický rys DBS), což ale LP neumí.
- DDB dotaz vyžaduje jako odpověď všechna řešení, LP právě naopak - typicky 1.
  - Toto dramaticky ovlivňuje způsob vyhodnocení dotazu - algoritmus.





# Architektura DDBS

- **Homogenní**
  - Jeden integrovaný systém spravuje IDB i EDB a provádí odvození (dotazy)
- **Heterogenní**
  - Relační DBS je použit pro EDB a logický systém provádí odvození nad IDB

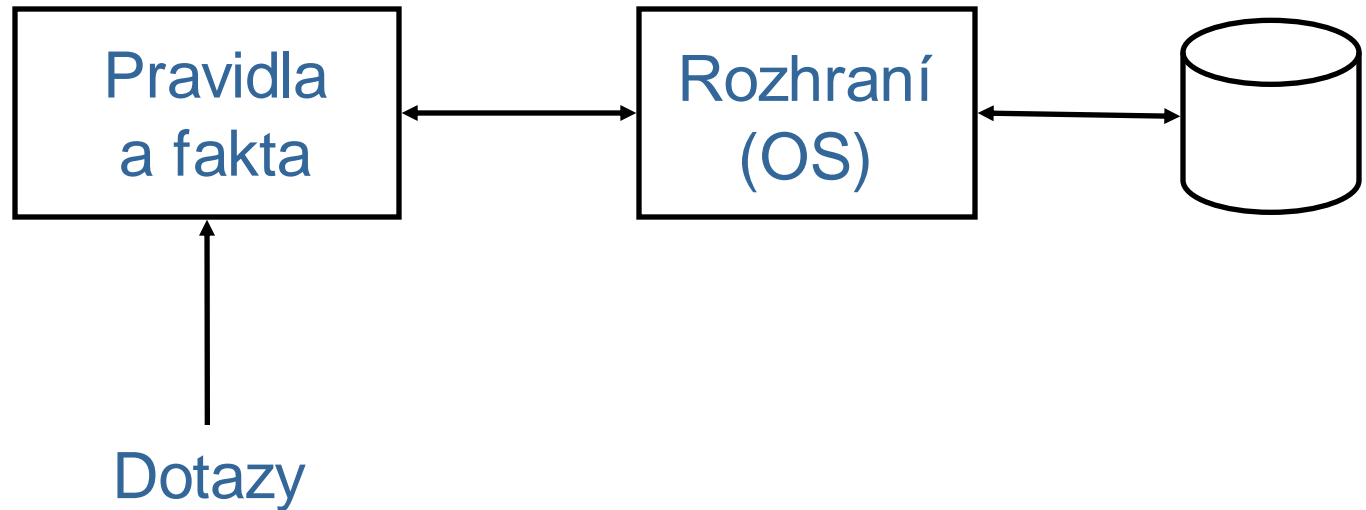



# Homogenní architektury - ČLS

- **Fakta a pravidla na sekundárním paměťovém médiu.**
  - Do hlavní paměti (HP) až „když je čas“
  - Zůstávají tam po dobu vyhodnocení
  - Unifikace jen nad daty v HP
  - Praktický žádný přístup k sekundárním pamětím

# Homogenní architektury - ČLS


Logický systém





# Homogenní a. - nedostatky (1)

- **Reprezentace faktů a pravidel stejným způsobem může být zavádějící.**
  - Různá velikost
  - Různá správa
  - *Pokud použiji různé přístupy, mohu dostat lepší výsledky*



## Homogenní a. - nedostatky (2)

- **Uložení všech pravidel a velkého množství faktů do HP dramaticky snižuje celkovou velikost DB.**
  - Částečně může eliminovat virtualizace paměti.



# Homogenní architektury - PLS

- **DBMS operace (indexace, vyrovnávací paměť, ...) jsou zpracovány na úrovni logického systému (IDB - Prolog). Tak je možné řídit přístup k EDB na sekundárním paměťovém médiu.**

# Homogenní architektury - PLS



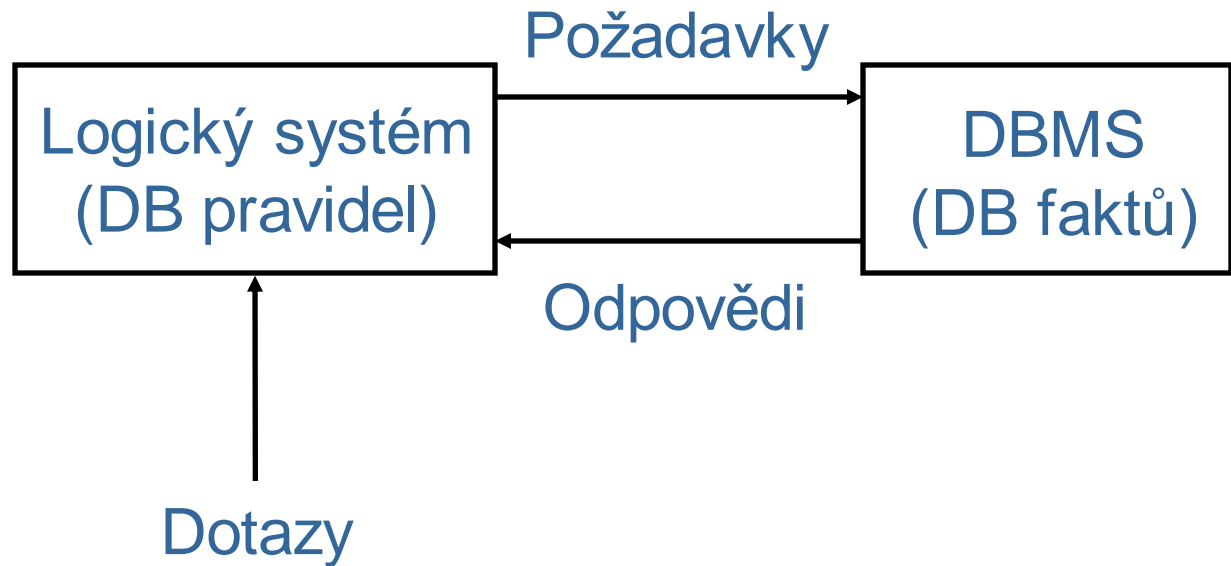


# Heterogenní architektury (1)

- **Logický systém (LS)**
  - „Front end“ - řídí, UR
  - IDB
- **Relační DBMS**
  - „Back end“ - odpovídá na dotazy
  - EDB



# Heterogenní architektury





## Heterogenní architektury (2)

- **Kompilované**
- **Interpretované**
- ***Něco mezi* (hugs, Java, ...)**



# Kompilované HA

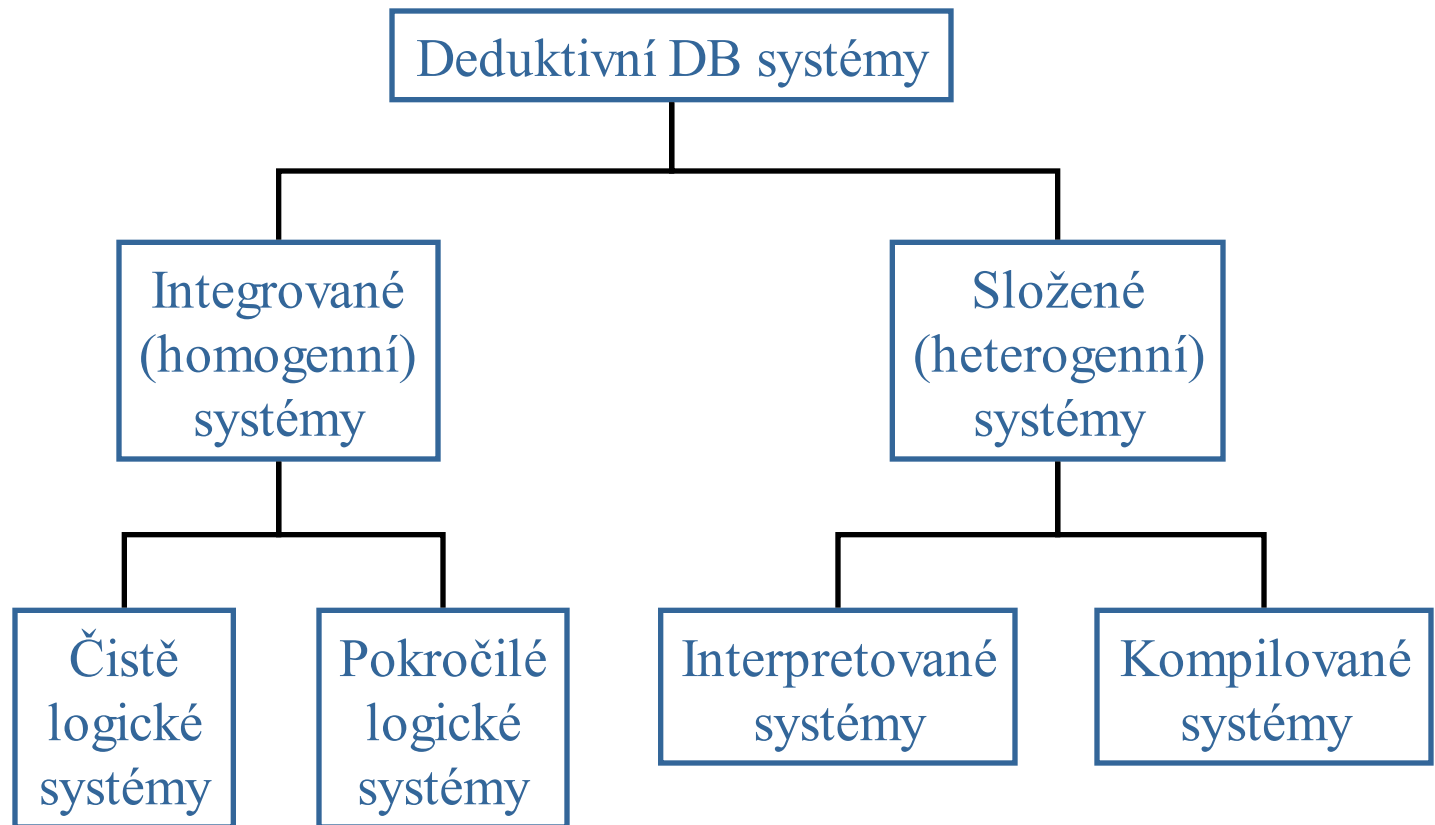
- **LS přeloží dotaz a pravidla do nezávislého DB programu (rek./lin.), který pracuje jen s fakty. Poté DBMS vyhodnotí program (jen nad EDB) a předá výsledky LS.**
- **Komunikace LS a DBMS na vysoké úrovni (mírná/žádná vazba).**
- **Problémy s transformací IDB do práce nad EDB - nekonečně mnoho poddotazů**



# Interpretované HA

- Fakta uložená v DBMS jsou poskytnuta „na požádání“ (úzká vazba).
- Častá interakce, na nízké úrovni.
- Výhoda v řízení vyhledávacího procesu.
- Na druhou stranu častá interakce způsobuje přetížení komunikace a celkově zpomalení.
- Do DBMS posílány jen jednodušší dotazy  
=> neúčinné optimalizace na úrovni DBMS

# Shrnutí architektur





# Přehled systémů (1)

- **PROSQL**
  - heterogenní (Prolog+SQL/Data System)
  - IBM
  - interpretovaný/kompilovaný
  - predikát SQL



## Přehled systémů (2)

- **NAIL!**
  - (Not Another Implementation in Logic!)
  - Stanford University
  - kompilovaný systém
  - **DATALOG+SQLDB**
    - nelineární rekurze v pravidlech
  - **Glue-Nail** - následník
    - G. je imperativní jazyk



## Přehled systémů (3)

- **LDL (logic data language)**
  - **MCC (Micro-Electronic and Computer Corporation)**
  - **zlepšení vazby u heterogenních systémů**
    - vybaven prvky neznámými u log. systémů
    - množiny, negace (vychází z množin)
  - **zdola nahoru**
  - **LDL++ (C/C++ rozhraní)**





## Přehled systémů (4)

- **MegaLog**
  - ECRC (the European Computer Research Center)
  - velký objem dat + vlastnosti Prologu
  - negarantuje ukončení programů
  - původce BANG souborů (prost. DB)
  - EKS (nad MegaLog-em)
    - integritní om., rekurze, pohledy, negace, ...
  - shora dolů + množiny



## Přehled systémů (5)

- **Lola**
  - TU v Mnichově
  - kompilace H. klauzulí do programu Relational LISP (vnořené SQL příkazy)
  - logická část je rezidentní v paměti (jen malé DB)



## Přehled systémů (6)

- **Coral**
  - University of Wisconsin at Madison
  - zdola nahoru + optimalizace
  - jednouživatelský systém, v paměti
  - EDB je v systému Exodus
  - možnost práce s velkými daty je nejasná



# Přehled systémů (7)

- **Aditi**
  - **University of Melbourne**
  - **zdola nahoru, na žádost i opačně**
  - **IDB i EDB mohou být na disku**
    - vhodné i pro velké objemy dat
  - **funkce, negace, agregované funkce**



## 13. Změna dat v DDB



# Změna dat v deduktivních DB

- **V DB běžný požadavek**
- **LS „jen“ dotazovací**
- **Vztah s**
  - množinovými operacemi
  - determinismem



# Množinové operace

- **Možné najednou provést úpravu u více dat, které jsou dány provedením nějaké operace.**
  - výběrový podsystém
  - změnový podsystém
  - Operace SQL (Update a Delete) pracují přesně na tomto principu.



# Role determinismu

- Vyhodnocením dostáváme jediný výsledek, nezávislý na strategii vyhodnocení
- *Spolu s množinovými operacemi a vyhodnocením hraje roli pro SQL*
  - kontroly
  - referenční integrita





# Problémy v DATALOGu

- **Vzájemná interference mezi výběrovým a změnovým podsystémem**
- **Násobné nekonzistentní úpravy jako výsledek množinového přístupu**



# Interference podsystémů

- Pokud se změny provádějí okamžitě, dle toho jak jsou známy výsledky (dílčí), můžeme dostat jiné výsledky, než v případě ukončení operace výběru - není zaručen determinismus.
  - *Viz Prolog a úprava DB v Prologu*



# Nekonzistentní změny

- **Vložení a smazání stejných dat, která „vznikají“ v dotazu.**
- **Nedeterminismus při vícenásobném přístupu.**
- **Dáno množinovým vyhodnocením.**
- **Nutno použít speciální přístupy.**



# Řešení

- Obecně těžké
- Často separace jazyků dotazovacích a obnovovacích (Glue-Nail)
  - Slabá integrace mezi oběma jazyky
- Nebo na syntaktické úrovni
  - uspořádání
    - jazyky nejsou zcela deklarativní
    - neplatí všechny výpočtové modely DATALOGu
  - omezení (LDL)



# Příklady jazyků se syntaktickou úpravou

- **DLP-DL**
  - **Dynamic Logic Programming Declarative Language**
  - **Abitebou, Vianu**
  - **1988**
- **RDL1**
  - **de Maindrevaille, Simon**
  - **1988**



# Řešení u syntaktických úprav

- **Změna atomů**
  - v hlavičce
  - v těle (pravidel)
- **Určuje to vyhodnocovací model**
  - hlavička → zdola nahoru
  - tělo → shora dolů



# Jazyky s omezeními

- **Nepoužívají se**
  - snížení *síly* jazyka
  - nemožnost některých konstrukcí vůbec



# Příklad: DLP-DL (1)

- **DLP - shora dolů**
- **LP - zdola nahoru**
- **DLP:**
  - Najmi(Zam#, Jméno, Zařazení, Plat, Prémie, Odd#) ←  
    <+Zaměstnanec(Zam#, Jméno, Zařazení, Plat, Prémie)>  
    (<+ZamOdd(Zam#, Odd#)>  
    (průmplat(Odd#, prům), prům<30.000)).
  - ?Najmi(1024, Novák, inženýr, 40.000, 0, 10).





## Příklad: DLP-DL (2)

- **DLP:**
  - ?Zaměstnanec(1035, Černá, sekretářka, 12.000, 0),  
<+ZamOdd(1035, 10)> ( Nadřízený(X, 1035) )
- **DL (nemá modelovou sémantiku)**
  - **EDB: s**
  - **IDB: r, t**
    - $r(X) \leftarrow s(X), -t(X).$   
 $t(X) \leftarrow s(X), -r(X).$
    - $[s(1), s(2), s(3)] \sim \rightarrow |z. a. | \sim \rightarrow [s, r, t(1), s, r, t(2), s, r, t(3)]$ 
      - pro shora dolů je to nemyslitelné
      - jde o operační sémantiku



# Deferred Update Semantics

- Řeší problém rozhraní mezi dotazovacím a obnovovacím mechanismem:
  - Změna hodnot není provedena tehdy, když je generována dotazem, ale až poté, co je vyhodnocen celý dotaz.



# Nekonzistentní obnova dat

- **Je nutno dodat pravidla pro řešení konfliktů**
  - jedná se o chybu a výpočet je zastaven
  - definování priorit
  - operace (obnovy/mazání) způsobující nekonzistenci se neprovedou vůbec



# Přístupy k obnovám v DDBS (1)

- **Dotazy/obnovy:**
  - Modelování obnovování údajů ne vždy znamená i modelování dotazů a naopak. Pro DDB je důležité obojí.
- **Sémantika:**
  - Deklarativní sémantika má teoretický model.
- **Vyhodnocení:**
  - Korektní a úplné vzhledem k sémantice.
  - Zdola nahoru a shora dolů - často jen jeden přístup, omezuje optimalizace.



# Přístupy k obnovám v DDBS (2)

- **Místo obnovy:**
  - V LJ v hlavičce nebo těle. Často obnova v těle znamená vyhodnocení shora dolů a naopak. Výjimky: U-DATALOG, Transaction Logic.
- **Provedení obnovy:**
  - Okamžitá/pozdržená sémantika obnovení. U pozdržené sémantiky jsou třeba řídicí operátory mimo jazyk (U-DATALOG).



# Přístupy k obnovám v DDBS (3)

- **Nedeterminismus:**
  - Často umožněn, i když sporná vlastnost.
- **Obnovy založené na množinách**
- **Paralelismus:**
  - Paralelní obnova DB. Není to stejné jako souběžné transakce (jsou serializovány).
- **Řešení konfliktů**



## 14. Integrace DDB a OODB



# Motivace

- **Současné OODBMS ukládají data, nikoliv však znalosti**
- **Zejména spojeny s imperativními jazyky - deklarativní styl chybí**





# Nedostatky z pohledu DDB

- **Modularita chybí**
- **Strukturovaná data**



# Problematika integrace

- **Objekt je**
  - teorém?
  - teorie (množina logických klauzulí)?
  - term?
- **Vývoj stavu OO komponent**
- **Je přítomná úroveň schématu (object blueprint/class)?**



# Možné přístupy (1)

- **Rozšíření logického programování směrem k modularitě a OO vlastnostem**
  - objekt je teorie
  - objekty se nevyvíjejí (žádné schéma)



## Možné přístupy (2)

- **DOODB**
  - objekt je term
  - schéma/instance objektu
  - stav objektu
- *LOGIN: OO jazyk odvozený z Prologu, objekty jsou termy. Ovlivnil další tvůrce.*



## Od LP k MLP a OMLP

- **Kompoziční operátory -  
inkrementální stavba programů  
spojováním nezávislých komponent**
- **Mechanismus na definici abstrakcí a  
rozsahu platnosti**



# MP jako alg. komp. programů

- **Modulární paradigma programování**
- **Kompozice (sub)programů**
- **LP jsou elementy algebry a existují operátory pro kompozici těchto elementů - model**
- **Mechanismy na úrovni meta-jazyka (nikoliv zásah do H. klauzulí, např.)**



# Gödel

- Řada prací vyústila v tento jazyk  
– viz FLP
- Skrývání informací  
(implementace/rozhraní)
- Zapouzdření
- Kompozice modulů



# Modularita jako algebra

- Kompozice je mocným nástrojem pro strukturalizaci programu, která nezasahuje do teorie H. klauzulí
- Podpora znovupoužití modulů a možnost záměny funkčně shodných modulů
- Zapouzdření – pokud existuje mechanismus na definici rozhraní modulu (komponenty)





# Programy otevřené logiky

- **Open logic programs**
- **Forma kompozice:  $\approx_{\Omega}$** 
  - Zobecnění sjednocení programů
  - $\Omega$  je množina predikátů určující které predikáty mohou být sdíleny
- **Parciální/úplná dedukce (přítomnost v  $\Omega$ )**
  - Parciální je možné zúplnit přidáním klauzulí do  $\Omega$



# Operátor sjednocení

- **Implementuje dynamické pravidlo pro určení rozsahu platnosti**
  - odkaz na predikát určuje definici podle toho, z čeho je (a kdy a jak byl) program složen
- **Vhodné pro ,asimilaci‘ znalostí (jakmile je známa nová skutečnost, je okamžitě začleněna)**



# Otevřené programy

- **Otevřené vzhledem k dalším programům**
- **Neúplný popis nějaké znalostní domény, který může být dále doplněn kompozicí s jiným programem**
  - **Něco co platí v jednom programu nemusí platit v jiném a naopak - takto lze dospět k novým k závěrům**



# Kompozicionalita

- Neznámá v LP
- Operátor (syntaktické k.)  $op$
- Platná, když sémantika  $P$   $op$   $Q$  je dána složením sémantik  $P$  a  $Q$
- V logických programech může sjednocení klauzulí způsobit problém (OR-k.)
  - nutné dodat mapování z jedné množiny atomů na jinou



# Diferenční logika

- **Operátor obecné dědičnosti**
  - statická a dynamická dědičnost
  - kompozice sjednocením klauzulí
- **Dědičnost jako prostředek pro diferenční programování**
  - popisujeme, jak se vytvářené programové komponenty liší od stávajících



# Praktiky

- **Využití filtrů**
  - modifikace vnějšího projevu komponenty
- **Modifikovaná verze je nová verze, která má nějaké speciální vlastnosti**
  - pro jejich implementaci může využít existující komponentu (OO dědičnost)



# Diferenční programy

- **Skládají se z komponent**
- **Rozhraní komponenty - tři množiny predikátů**
  - **staticky děděné**
  - **dynamicky děděné**
  - **rozšířitelné**
  - **(interní)**



# Vlastnosti

- **Aplikace ISA sítí**
- **Staticky a dynamicky děděné predikáty se vyhodnocují podobně jako při přetěžování**
- **Rozšířitelné predikáty - ortogonální mechanismus**
  - lokální definice jsou rozšířeny o děděné, nepřetěžují je





# Operační sémantika

- Modifikace SLD rezoluce
- Operátor syntaktické k. je ekvivalentem k operační sémantice ISA sítí
  - transformace struktury do *plochého* ekvivalentu



# MP jako rozšířené H. klauzule

- **Bohatší sémantika**
  - změna konfigurace modulů pro vyhodnocení podcílů v době běhu
- **Vestavěný mechanismus, který ovlivňuje vyhodnocení**
- **Operátory jako logické spojky - rozšíření H. klauzulí (Miller)**



# Nedostatky

- **Nevhodné pro DDB**
  - zánik dat/modulů po vyhodnocení
- **Další řešení**
  - Kontextuální LP
  - Zasílání zpráv a dědičnost



# Kontextuální LP

- **Kontextově závislé predikáty**
- **Proměnný kontext pro důkaz**
- **Základní *kameny***
  - **Jednotky (units)**
  - **Rozšiřující cíle**



# Vyhodnocení

- **Rozšiřující cíl  $u > g$** 
  - $u$  - jednotka
  - $g$  - cíl
  - $g$  je vyhodnocena v kontextu  $u$
  - použití posledně nahraných jednotek
    - do dříve nahraných jednotek se jde jen pro definice aktuálně neznámé
    - různé metodiky vnořování a odkazování se na vnější predikáty



# Zasílání zpráv a dědičnost

- **SelfLog (1992, Bugliesi)**
- **Jazyk H. klauzulí s vestavěnou modularitou**
  - množina klauzulí tvoří pojmenovanou teorii (objekt/jednotku)
  - objekty komunikují při žádostech o vyhodnocení (pod)cíle - model zasílání zpráv



# Notace

- **Zpráva-cíl (model zasílání zpráv)**
  - **$o:g$**
  - cíl  $g$  je vyhodnocen objektem  $o$
  - **symbol** : zaručuje změnu kontextu na objekt  $o$



# Dědičnost

- Rozšířením o dědičnost ~> SelfLog
- Statická i dynamická dědičnost
- Mechanismus rozšíření objektů
- *o:g* - o včetně všech předků





# SelfLog

- **Dědičnost + Zasílání zpráv**
- **Sémantika programu**
  - funkce nad Herbr. množinami (spíše než nad množinou)



# DOODB

- Řada výzkumných směrů
  - „Logika pro objekty“
    - objekt=term, schéma, formální báze
    - neuvažují obnovy a vývoj
  - OO logické jazyky pro dotazování
    - IQL
  - OO rozšíření DATALOGu
    - DB/odvozené relace
    - LGORES, COMPLEX, DATALOG<sup>Meth</sup>



# „Logika pro objekty“

- **Objekt=term, žádná změna objektů**
  - O-logic (Maier, Kifer&Wu)
  - C-logic (Chen & Warren)
  - LOGIN
  - Frame Logic (F-logic)
    - vyšší řády



# Motivace

- **Formální báze pro reprezentaci komplexních objektů**
  - absence formálních OO datových modelů
  - modelově teoretická báze
  - objekt, identita objektu, hierarchie tříd
  - metody, schéma (F-logic)



# LOGIN

- **Třídy a objekty jsou složené termy**
- **Argumenty jsou atributy**
- **Návěští umožňují dědičné vazby**
- **$\Psi$ -termy - záznamy**
- **Unifikace nad typy**
  - **typové rozšíření Prologu**
  - **dědění na úrovni typů (sémantické sítě)**



# LOGIN (pokr.)

- **Množinový model**
- **Neexistuje identita objektu**
- **Nejsou metody**
- **Dědičnost je udána unifikačním algoritmem**
  - sémantika spec. v „equational logic“
- **LIFE - funkcionální rozšíření, vícenásobná dědičnost**



# X-logic

- **Objekty jsou rozšířené termy**
  - Identifikátor objektu
  - Typ
  - Strukturovaná hodnota
    - záznam s návěštími (přístup na hodnoty)
- **Množinové zpracování**
- **Dědičnost jako poset**
- **Typ je zpracováván dynamicky**
  - extent



# F-logic

- **Přidává schéma a metody**
  - schéma a instance není rozlišena
- **Tím, že nerozlišuje  $\Rightarrow$  formální báze**
- **Třídy a objekty organizovány ve svazech**
- **Dědičnost je součástí sémantiky**
  - jakmile je  $p$  před  $v$  ve svazu, tak  $v$  má vlastnosti  $p$





# F-logic

- **Formálně sémantika prvního řádu, ale**
  - množiny
  - pod-třídy
  - schéma
    - návěští jsou formálně objekty (úvahy)
- **Deklarativní spec. metod (také)**
  - metoda je návěští s parametry



# Logika jako dotazovací jazyk

- **Rozšíření DATALOGu o objekty**
  - nepracují v 1NF, ale rozšířeném rel. m.
- **COL**
  - hodnotově orientované modely
  - strukturované objekty
  - datové funkce (EDB i IDB)
    - relace s funkční závislostí
    - manipulace se strukturovanými daty
  - n-tice, množiny, „union“
  - není OLD a dědičnost



# IQL (Identity Q. Lan.)

- **Rozšiřuje COL**
  - **OID (typové ukazatele) a dědičnost**
    - sdílené a cyklické struktury
    - množinové operace
    - zvýšení síly jazyka
  - **OO model (schéma x instance)**
  - **O-REL**
  - **pravidla, statická typová kontrola, úplnost**
  - **n-tice, množiny**
  - **dědičnost (via union)**



# IQL - sémantika

- **Operátor pevného bodu (inflační)**
  - rozšířený o OLD
- **Vytvoření objektu možné**
- **Obnova, smazání nikoliv(!)**



# LLO (Logic Lan. for Objects)

- OO datový model s metodami
- Metoda = klauzule
- Dotaz = zpráva (zpracovává unif. a.)
- Datové abstrakce via metaproměnné
- Objekt - stav, chování, typ, OID
- Přetěžování i předefinování metod je dovoleno



# DTL (Data Type Log)

- **Dotazy k TM jazyku**
  - (TM) je OODB spec. jaz.
    - Množiny, omezení na vícenásob. d.,
    - Možná statická typová kontrola
    - OID
- **Rozlišuje typ a instanci**
- **Množiny (predikáty + dědičnost)**
- **ISA predikát**
- **Dědičnost na úrovni predikátů**



# OO rozšíření DATALOGu

- **LOGRES**
- **COMPLEX**
- **DATALOG<sup>Meth</sup>**



# LOGRES

- **OO datový model+pravidla**
  - Pravidla = dotazy+obnova
- **Sdílení dat, dědičnost**
- **DB struktura je dána**
  - typovými rovnicemi
  - integritními omezeními
- **O-REL**
- **Možná statická typová kontrola**
- **Schéma x instance**





# LOGRES (pokr.)

- **Asociace, složité datové struktury**
- **Datové funkce**
- **OID, negace (hlavička i tělo)**
- **Moduly**
  - zachování deklarativity
  - i když: strategie řízení



# COMPLEX

- **Rozšíření DATALOGu**
  - objekty (složené)
  - OID
  - (vícenásobná) dědičnost
- **Deklarativní**
  - shora dolů
  - zdola nahoru
- **Schéma x instance**



# COMPLEX (pokr.)

- **Datové entity**
  - třídy (+instance)
  - odvozené relace (vztahy instancí)
    - objektová asociace (n-tice)
    - nestabilní (odvozená!)
- **Neumí množiny a n-tice na úrovni datových typů**
- **Objekt je n-tice v relaci**



# DATALOG<sup>Meth</sup>

- **DATALOG plus**
  - metody
  - dědičnost
  - přetěžování
  - pozdní vazba
- **Motivace**
  - dědičnost s předefinováním



# DATALOG<sup>Meth</sup> (pokr.)

- **Třída a její**
  - extent (množina OLD)
  - množina metod
- **Vlastnosti objektů jsou dány**
  - metody a
  - predikáty
- **Metody jsou funkce (ne predikáty!)**
- **Vlastní rezoluční mechanismus**
  - metaprogramování



# Záloha



# Záznamy přednášek

- **Implicitní nastavení**
  - Nevysílá se
  - Neukládá se
- **Důvod**
  - Ztratily původní význam
  - *Nahnat lidi na přednášky* 😊
- **PDB ?**
  - Dle toho, jak to vyjde a obsahu



# Zpětná vazba

- **Smysl?**
  - Stejná věc v PDB je hodnocena negativně oproti jiným předmětům, kde je hodnocena neutrálně, nebo kladně
  - Každý rok se liší
- **Obsah přednášek nelze změnit radikálně**
  - Širší kontext
  - Zajímavosti



# Zpětná vazba

- Cvičení, projekty

- SW

- Je-li 45 týmů, nelze kvůli tomu instalovat 30 různých SW pro referenční stroj

- Volnost v projektu dostatečná

