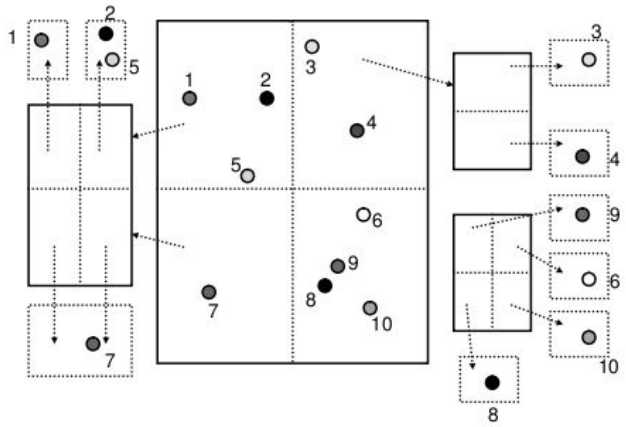
2013/2014 **řádný termin**

Two level grid file

- Grid file na vrchní úrovni (kořenový adresář) adresuje do grid files na druhej úrovni (pod-adresář).
- Zmeny jsou časteji lokální, ale k nelokálním může stále docházet



K-D Tree

- Prostor je delen hyper-plochami na nejvyssi urovni (rovnobezne s osami)
- Kazda plocha musi obsahovat alespon jeden bod dat
- vkladani, hledani - ok
- mazani - problem
- jen body

Relační, objektová a objektově-relační DB (období, základní charakteristika, datový model, dotazovací jazyk a výpočetní model)

Relačná - Od 80.let získaly převahu nad síťovými a hierarchickými.

Vlastnosti:

- standardizovaný - SQL (od 1986)
- relačný model dát, formálne zavedené od Codda
- predstavujú ho tabulky obsahujúce relace
- podpora výrobcov DB produktove

Nevýhody:

- veľmi omedzená množina dátových typov hodnôt
- pre vzťahy M:M je nutná väzobná tabulka

- žiadne referencie či ukazovatele

Model dat:

- relačný
- kolekcia tabuliek
- vzťahy tabuliek vyjadrené pomocou cudzích a kandidátnych klúčov

Dotazovací jazyk:

- SQL
- neprocedurálny, deklaratívny

Výpočetní model:

- založen na hodnotách ve sloupcích tabulek
- žádné reference či ukazatele
- jednoduchá navigace po tabulce pomocí kurzoru

Objektová - Na konci 80.let vliv paradigmatu OO programování

-> vznik OO DB (tj. perzistentního uložení objektů). V objektové databázi by SŘBD neměl povolit:

pokud výsledek jedné z operací je objekt s parametry, který nelze v daném SŘBD uložit (výsledek

operace nelze popsat objekty v DB).

Vlastnosti:

- modelovanie a vytváranie perzistentných dát ako objektov
- nenahrádzajú, doplňujú relačné (kombinácia obidvoch sa nazýva objektovo-relačná DB)

Výhody:

- vzťahy M:M sa dá vytvárať priamo
- navigácia po objektovej štruktúre pomocou referencií prostredníctvom OID
- atribúty objektov môžu byť iné objekty (zložité typy, ADT - abstract data types)

Model dat:

- objektový
- jednoznačný OID(objekt ID) pre každý perzistentný objekt
- podpora ADT, zapúzdrenia, polymorfizmus
- atribúty môžu byť iné objekty
- vzťahy objektov pomocou referencií

Dotazovací jazyk:

- väčšinou bežné objektové jazyky
- snaha o standardizaciu (jazyk OQL od ODMG)

Objektovo-relačná - Cílem je spojit výhody relačního a objektového modelu.

Výhody:

- snaha o obohatenie tabuliek o objektovú orientáciu
- navigácia po tabuľkách pomocou kurzoru aj referencií

Model dat:

- tabulky nemôžu byť normalizované (porušujú prvú normální formu)
- obecnejšie (vnorené) relace (nested relational model)
- data stále v tabulkách, ale hodnoty môžu mať bohatšiu štruktúru - ADT
- ADT zapúzdruje data aj operácie
- OID umožnuje definovať nové typy vzťahov medzi tabulkami
- navigácia pomocou kurzoru a referencií

Dotazovací jazyk:

- v štandarde SQL-1999

Výpočetní model:

- navigace po tabulkách pomocí kurzoru i referencí

Snímková DB + dotaz

- každy snapshot popisuje stav světa v konkrétním časovém okamžiku. Relace uspořádání potom definuje tok času.
- Jinak: Snímkový model DB je takový, ve kterém je každý stav DB opatřen časovým razítkem, obsahuje snímkovou tabulku (SNAPSHOT). Temporální DB tedy vlastně používá funkci mapující čas na stav databáze (viz fce dole...).
- Temporální DB je potom funkce typu
  - $T \rightarrow DB(D, \rho)$ (každý čas se zobrazí na některý snímek databáze ve kterém je stav DB k tomuto času)
    - T… time
    - D… data
    - ró… databázové schéma
  - datové typy $T \rightarrow (D_n \rightarrow bool)$
- zaznamenává stav dat v jistém okamžiku
- Relace uspořádání nad těmito snímky tvoří tok času... historii (posloupnost stavů - snímků).
- Historie databáze je také snímkový model.
- Problém, pokud se dotazuje na "všechny okamžiky, kdy platilo, že ...".

(tj. všechny okamžiky, kdy byla podmínka v rámci DB platná - musel by se kontrolovat každý snímek databáze).

Dotaz:

exists y (isUsed(x, y) && ♦ ((NOT rep(y)) && ♦ (crash(y))))

isUsed(a, b) řidič a používá auto b, rep(y) auto opravováno, crash(y) nehoda auta vyžadující opravu
♦ (crash(y)) - auto malo niekedy v minulosti nehodu vyžadujúcu opravu
(NOT rep(y)) - auto nebylo opravované
♦ ((NOT rep(y)) && ♦ (crash(y)) - pre auto y niekedy v minulosti platilo, že nebolo opravované a že niekedy v minulosti malo nehodu, ktorá si vyžadovala opravu
isUsed(x, y) - vodič x používa auto y
exists y (...) - existuje také auto, že …
jednou vetou teda: Najdi vsetkych soferov, ktori soferuju auto, ktore bolo burane ale nebolo opravene (spravne?) - asi jo

Realms + řeší úplně problém diskretizace?

Realms - **Úplné deskriptory**

- Souhrnný popis všech objektů
- Každý koncový bod je bodem sítě
- Žádný vnitřní bod není zaznamenán v síti
- Žádné dvě úsečky se neprotínají
- Problémy s číselnou reprezentací
- Problémy s daty obsahujícími průsečíky, které ale nemusí být v síti
- Řešení: Segmenty do obálky – segmenty v obálce nikdy nemohou přeskočit přes bod mřížky

Množina bodov, úsečiek, prípadne vyšších celkov, ktoré majú tieto vlastnosti:

1. každý samostatný bod je bodom siete (přesná citace slajdů: každý (koncový) bod je bodem sítě)
2. každý koncový bod úsečky je bodom siete
3. žiadny vnútorný bod úsečky nie je zaznamenaný v sieti
4. žiadne dve úsečky nemajú priesečník ani sa neprekrývajú

Deskriptory tvoria základ pre priestorové dátové typy. Popisujú konečnú množinu bodov a nepretínajúcich sa úsečiek nad diskrétnym oborom. V podstate sa jedná o rovinný graf

//problém diskretizace?

Opět je zde problém s diskretizací, ne vždy je reprezentace dostatečná → **řešení:** *segmenty do obálky* (Pokud by se měly dvě úsečky křížit, je každá rozdělena na dva segmenty a jejich průsečík se stává koncovým bodem těchto segmentů - z pohledu uživatele databázového systému se tam však stále nachází 2 úsečky s průsečíkem.

Protože přidáváním dalších průsečíků se segmenty mohou stále více vzdalovat svému původnímu umístění, některé body mřížky (tudíž body na které se můžeme dotázat) se mohou dostat na opačnou stranu od úsečky a změnit tak výsledek operace. (Např. zda bod leží uvnitř polygonu.) Je proto vytvořena obálka, v rámci které se segmenty úsečky musí pohybovat.)

Temporální integritní omezení

uzavřené formule prvního řádu temporálního dotazovacího jazyka definují integritní omezení

- Použití integritních omezení - zachycení sémantiky DB aplikace, návrh DB - normální formy - dobrá schémata bez anomálií a dobrá dekompozice
- Historie H splňuje omezení O jestliže je O pravdivé v každém stavu H
- Konečná historie H potenciálně splňuje omezení O jestliže může být rozšířena do nekonečné historii tak, že splňuje O
- Důvod zavedení: ukládání pouze "vyznamných" dat
- Implementace tak, že se při každé změně testují integritní omezení nad Historii H

Definovat vnoření dvou R-ploch + netriviální obrázek

R-cyklus je uzavřená lomená úsečka, která je vytvořena podle pravidel ukládání deskriptorů, kde

- lomená úsečka je tvořena posloupností *n* úseček $(s_0, ..., s_{n-1})$
- konec úsečky s_i je shodný se začátkem úsečky $s_{(i+1)\%n}$.
- Přitom se žádné dvě různé úsečky s_i, s_j nikde neprotínají.

R plocha f je dvojice (c, H) taková, že c je R-cyklus, $H = h_1, ...h_m$ je množina R-cyklů a platí

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

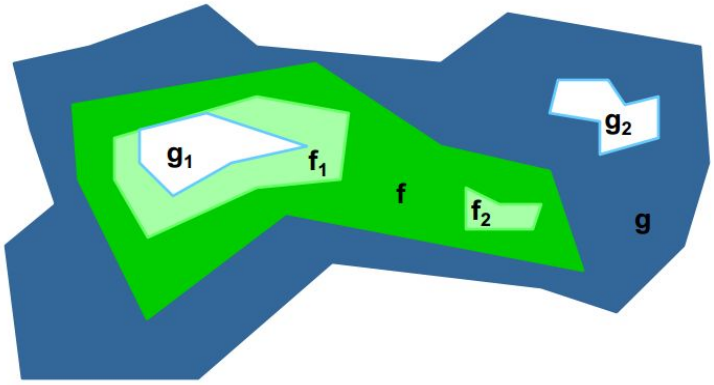
Pozn.: poslední podmínka zaručuje jednoznačnost reprezentace.

Plocha f je plošně obsažena v g

Mějme 2 R-plochy $f = (f_0, F)$ a $g = (g_0, G)$. f je plošně obsažena v g, když platí:

- f_0 je plošně vnořený 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



Jaké jsou pravidla a predikáty deduktivních DB

Pravidla

- Odvozena pravidla - jsou to pouze dotazy/pohledy na data v DB
- Rekurzivni x nerekurzivni
- Pravidla s vice stranami

Predikaty

- explicitne ulozene
- implicitne ulozene (odvozene odvozujicimi pravidly z explicitne ulozenych predikatu)

neni to spis pravidla - rekurzivni/nerekurzivni, predikaty - OUP/EUP?

Důkazní (syntaktická)

- Axiomy: explicitní data, implicitní informace, ktera muze byt odvozena z EUP (explicitně uložené predikáty) ci OUP (odvoditelně uložené predikáty)
- Negace - pozitivní i negativní predikáty
- Důkaz: Vsetky fakty odvodene pomocou OUP su odvodene pomocou modus ponens
- uplatnuje sa dopředná odvození (axiomy -> teorém)

Modelová (sémantická)

- pravidla definují možné modely
- dokazuje se přiřazením hodnot proměnným
- Interpretace: predikát přiřazuje pravdu/nepravdu každé možné instanci

- modelem množiny pravidel je interpretace, kde jsou všechna pravidla pravdivá (tj. nezávisí na hodnotách proměnných)

  Minimální model (M_k)

  - M_k je podmnožinou všech modelů (tj. z M_k je možné odvodit ostatní modely)
  - NEexistuje model $M \subseteq M_k$ (tj. obsahuje řešení)
  - systemy s negací - vic minimalnich modelů

  Uspořádání modelů

  - uspořádáme modely v takovém pořadí, aby $m_i \subseteq M_{i+1}$
  - minimální model je první v pořadí

Výpočetní

- vypočítání pravdy/nepravdy pravidel
- algoritmus na vyhledání důkazu potenciálního faktu
- typicky používá PROLOG
  - to co prolog považuje za pravdivé nemusí být vždy model
  - v mnoha případech poskytne minimální model

DATALOG

- Relační jazyk bez negace s rekurzí
- Integrace DB a PROLOGu
- Pravdila jsou posloupnosti operací relační algebry
- Predikáty definují relace
- DB
  - EUP = relace v DB
  - vestavěné predikáty
  - OUP = Hornovy klauzule
- Modelová interpretace
- Dotaz je druhem pohledu
- Prologovské příkazy: atomické formule, predikáty, funkce
- a) bez negací -> výsledkem je minimální model
- b) s negací -> jeden z mnoha minimálních modelů
- SQL v rolid DML je nedostatečné pro tranzitivní uzávěry

Shlukování

- jednorozměrná temporální relace obsahuje shluky pokud je každý fakt spojovaný s nejvýše konečným počtem nepřekrývajících se intervalů
  - je potřebné zaručit, pokud se nad relacemi vykonávají ne-logické operace
- Někdy je potřeba spojit intervaly kvůli dalšímu zpracování (například při výběru sloupců)
- I=o na sebe navazují bez mezery) a nesoucí stejná data je možné spojit (např. každoroční výkazy při omezení pouze na sloupec adresa).

- zjednodušují selekci, projekci, spojení
- nezjednodušují temporální operace
- Tj místo abychom měli dva řádky:
    - (Hodnota1, [1,5])
    - (Hodnota1, [5,7])
- dostaneme jediný shluknutý řádek:
    - (Hodnota1, [1,7])
- Pozn.: pro více rozměrů času je shlukování nejednoznačné -> problém

Generický dotaz f vzhľadom k ||.|| je dotaz ||D1||=||D2|| ⇒ ||fD 1 ||=||fD 2 ||

Co znamena ten mnozinovy operator? Cekal bych tam spis implikaci.

- Generičnost dotazů = dotaz je generický, pokud jeho výsledek nezávisí na způsobu uložení dat v DB:
    - pokud je v db uložena fakta (a,[0,3]) (a platí od 0 do 3) nebo (a,[0,2]),(a,[1,3]), tak v první případě se jedná jen o shluknutí intervalů pro fakt a
    - ale pro dotaz: ∃ i,j. ∃ x(R(i,x) && R(j,x) && i != j) platí jen v druhém případě, ale v prvním díky shluknutí ne

Bonus - porovnat časy nenalezení položky v kD a adaptivní kD Tree

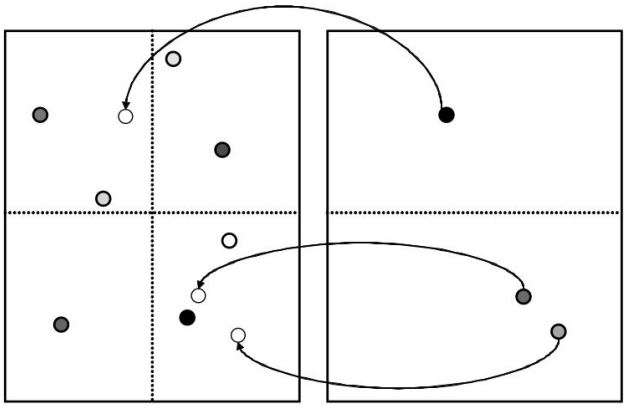
- (ASI) je ten cas rovnaky, lebo sa tak ci tak pri neuspesnom hladani musi pozriet v oboch pripadoch az do listov (ale kD jich ma min, takze je v tomto rychlejsi?)

2012/2013

1. opravný

Popište Twin Grid file a jeho použití v prostorových databázích

- Dva rovnocenné grid files (primární a přetokový) bez hierarchického vztahu
- Body se umisťují do primárního, pokud je plný (muselo by se dělit) dávají se do přetokového
- Rovnoměrné rozložení dat
- Využitie priestoru: až 90% (zlepšení) a bez zpomalení
- Teoreticky mozne paralelni vyhledavani.



Vypište objektové rysy SQL99 a pozdějších a jejich použití

- uživatelem definované typy (UDT)
  - obdoba třídy
  - atributy, metody
  - jednoduchá dědičnost
- typ sloupce tabulky může být UDT

(CREATE TABLE Zamestnanci_fakulty (id INTEGER PRIMARY KEY, zam zamestnanec_t)

  - v tomto případě nemají OID
- odkaz na atributy přes funkce nebo tečkovou notací *(WHERE zam.plat > 20000)*
- typové tabulky
  - tabulka s řádky nesoucími hodnoty strukturovaného UDT
  - každý řádek má OID (hodnota typu REF)
- typ REF *(vedouci REF(zamestnanec_t))*
  - odkazy na řádky v typové tabulce
  - mají vždy rozsah (scope)
- umožňují zpřístupnit atributy hodnoty strukturovaného typu, kterou identifikují *(SELECT vedouci -> jmeno)*

Co je snímková tabulka - definice vlastností, vysvětlete

- snaphot, statické dotazy, možno modifikovat - sorry toto je tak vysvetlenie na presne 0 bodov tpco :-D to je jendo ze to je v slidoch, ale ked to niekto napise do pismky tak vie co ma cakat peknu 0
- podla mna to ma byt ze to je klasicka relacna tabulka nepodporujuca cas platnosti ani cas transakcie, neda sa dotazovat na cas
- 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 -> DB(D, p)
- Datové typy: T -> (Dn -> bool)

- není příliš vhodný pro dotazy typu: “všechny okamžiky, kdy podmínka p byla v rámci DB platná”

∃y.ucastnikNehody(x,y) AND ♦(∃y.(vinikNehody(x,y) OR udelalPrestupek(x,y)))
proč je tam to druhé ∃y? nemělo by to být bez toho? -> to druhe ∃y “vytvara” novy y, teda ten y vo vinikNehody nie je to iste y ako v ucastnikNehody
♦(∃y.(vinikNehody(x,y) OR udelalPrestupek(x,y))) -
~~existuje člověk, který niekedy v minulosti urobil priestupok a bol vinníkom nehody -~~ ~~Proc? a a ne nebo~~

Všetci účastníci nehody, ktorí v minulosti spôsobili nehodu alebo urobili priestupok.

Jak se vyhodnocuje dotaz v deduktivní databázi bez negace - krok po kroku

- nesmí obsahovat negaci
- programy:
  - přeloženy do relační algebry
  - OUP jsou modelem pro pravidla
- pro nerekurzivní program je možné uspořádat pravidla tak, že pokud v p1, .., pn platí pi < pj, potom vede cesta z i do j

Vyhodnocení OUP:

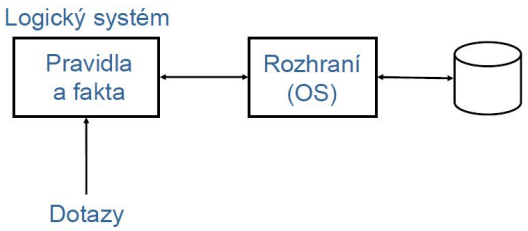
- Pro každé pravidlo **r** s hlavičkou **pi** je spočtena relace těla predikátu. Jde o operaci spojení výsledků všech podcílů.
- Výpočet samotného **pi** je projekcí relace těla predikátu na proměnné korespondující s hlavičkou s tím, že dochází ke sjednocení výsledků pro všechny kombinace.

Relace definovaná tělem pravidla:

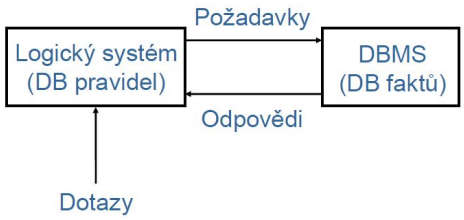
- relace r pro q :- p1, …, pn.
- P1,...,PN jsou relace pro p1,...pn.
- Pj = {(a1,...,an) | pj(a1,...,an) je splněno}
- Podcíl je splněn pokud:
  - je-li to běžný podcíl (EUP či OUP), potom S je tvaru p(b1, … , bn) a (b1, … , bn) je n-tice v relaci P korespondující k p
  - je-li to vestavěný podcíl potom S je aritmetická relace, která je splněna

Jaké existují typy deduktivních Databází a jak se implementují

- integrované (homogenní) - jeden integrovaný systém spravuje EDB i IDB - data, i odvozování
  - čistě logické
  - pokročilé logické
  - fakta a pravidla na sekundárním paměťovém médiu



  - nedostatky: reprezentace faktů a pravidel stejným způsobem může být zavádějící (různá velikost, různá správa)
- složené (heterogenní) - pro data je použit relační DBS (back end, pro EDB) a pro odvozování logický systém (front end, pro IDB)
  - interpretované - interpret vyhodnocuje dotaz a dotazuje se v případě potřeby relačního DBS => častá interakce, na nízké úrovni
  - kompilované - logický systém přeloží dotaz do nezávislého programu, který potom relační DBS vyhodnotí, komunikace mezi oběma systémy je na vysoké úrovni

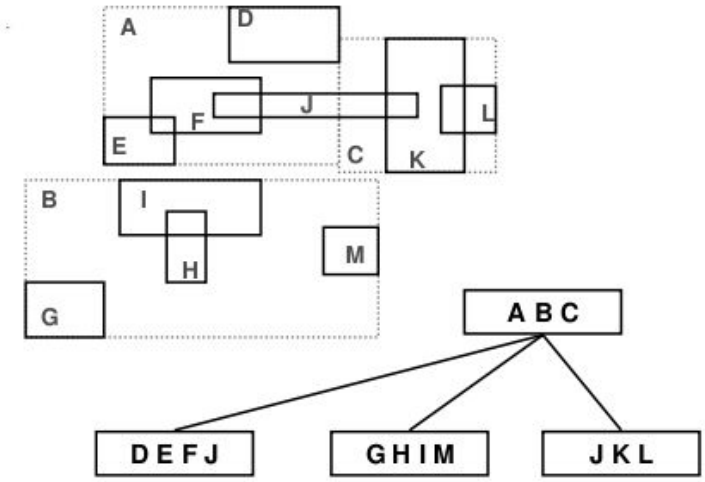


Jaké problémy se řeší při ukládání prostorových dat (ne jak se resi), jaky byste zvolili zpusob ulozeni prostorovych dat a jaky by byl efekt tohoto ulozeni.

- reprezentácia priestorových datových typov
- algoritmy a operácie nad priestorovými datovými typmi
- Problém reprezentace souřadnic: nejsme schopni zachytit reálná čísla (jen diskrétní prostor).
- Problémy s číselnou reprezentací
- Problémy s daty obsahujícími průsečíky, které ale nemusí být v síti

řádný

- ***R+-tree ve vztahu k prostorovym DB -popis, vlastnosti**
  - Zástupce metody ořezávání.
  - Je podobný R-Tree, ale namísto překrytí dělí objekty na části
  - Pokud objekt zasahuje do více buňek, které nesousedí, je třeba vložit buňku pro střední část nebo rozšířit jednu z buněk (může nastat problém).



DB s casem platnosti -popis, vlastnosti a jednou vetou popsat vyznam dotazu:

Ey.(ZK(x,y)) ^ [](not(Zap(x,y) v Exp(x,y))
ZK(a,b) - znamena ze student a ziskal v danem roce zkousku z predmetu b
Zap(a,b) - znamena ze student a ziskal v danem roce zapocet z predmetu b
Exp(a,b) - znamena ze student a byl ze zapoctu pro predmet b v danem roce omluven, nebo predmet zapocet nema
E - existencni kvantifikator
^ - konjunkce ("a")
v - disjunkce ("nebo")
not - logicka negace
[] (vyplneny ctverecek) - spojka temporalni vyrok. logiky "platilo vzdy v minulosti"

všichni studenti, kteří udělali zkoušku z předmětu, aniž by z něj měli zápočet

Čas platnosti říká, ve kterém období je daný fakt pravdivý v modelovaném světě
-v DB s casom platnosti je mozne tento cas modifikovat, dotazovat sa na historiu
Pro každé r_i definujeme relaci R_i tak, že:

• R_i = { (t, a_1, … ,a_k):(a_1, … , a_k) patri do r_i in D_t}

• Struktura temporální databáze s časovými razítky

• (D,=, T,<, r1, …, r_n)

• konečná instance ro nad D, T

• časová doména

• konečná instance ro nad D

-cas je len jeden z typov, ale je sucastou daneho usporiadania

integritni omezeni v temporalnich DB a jak se resi/implementuji (bez formalnich definic...)

uzavřené formule prvního řádu temporálního dotazovacího jazyka

- Použití integritních omezení
  - zachycení sémantiky DB aplikace
  - důvod zavedení je ukládaní jen "významých" dat
  - návrh DB - normální formy - dobrá schémata bez anomálií a dobrá dekompozice
- Historie H splňuje omezení O jestliže je O pravdivé v každém stavu H
- Konečná historie H potenciálně splňuje omezení O jestliže může být rozšířena do nekonečné historie tak, že splňuje O
- Důvod zavedení: ukládání pouze "vyznamných" dat

uplna formalni definice generickeho dotazu (moznost bonusovych bodu za priklad kdy dotaz neni genericky)

- dotaz f je generický vzhladom k |I.II pokiaľ platí – ||D1||=||D2|| ⊃ ||fD1||=||fD2||

Příklad

- $R^{D1} = \{([0,3],a)\}$
- $R^{D2} = \{([0,2],a), ([1,3],a)\}$
- $\exists i,j.\ \exists x(R(i,x) \wedge R(j,x) \wedge i \neq j)$ platí v D_2, ale nikoliv u D_1 - tedy není generický

omezovana promenna v deduktivnich DB -popis a jeji vliv na dotaz (-odstranění zdroje nekonečnosti?)

- každá proměnná na pravé straně pravidla je omezovaná
- každá proměnná porovnávaná s konstanotou na rovnost (..je to tak v slajdoch a je to imho podstatny rozdiel pozn. eurk0) je omezovaná
- každá proměnná porovnávaná na rovnost s omezovanou proměnnou je omezovaná

vliv na pravidlo: pravidlo je bezpečné, pokud jsou všechny jeho proměnné omezované

2011/2012

2. opravný

XML v Oracle 10g

- Typ XMLType - odpovídá standardnímu typu XML
- SQL funkce a operátory pro práci s XML
- Podpora XML Schema

- Dualita XML a SQL - flze používat SQL operace nad XML daty a naopak
- Podpora dotazování SQL/XML, XPath, XQuery
- Organizace dat - hierarchická struktura pomocí XML DB Repository

Indexovani v temporalnich DB

Indexace je problém, chceme-li například najít intervaly, které mají neprázdný průnik s jiným

- R-Tree (viz prostorové DB)
- AP-Tree (index jen pro přidávání dat, ISAM a B+-tree)
- Time Index
    - na začátku a konci intervalu uloží seznam dalších intervalů, které obsahují tento okamžik
    - nad těmito body postav B-Tree

DB obojiho casu + priklad temporalniho dotazu

- ukládá čas platnosti i čas transakce
- pouze připojuje -> Nemění údaje od/do, ale přidá nový záznam s opravenými hodnotami.
- k historii navíc obsahuje historii změn (tabulka platného času + 2x tabulka transakce (čas vytvoření, čas zneplatnění))
- Tabulka s platnými časy
    - Atribut čas platnosti (od/do)
    - Dovoluje modifikace. Obsahuje 1 záznam pro každou změnu.
    - Dotazy do historie
- Transakční tabulka
    - Atribut čas vložení/smazání
    - Dovoluje ROLLBACK v dotazu

Co je upravene pravidlo

Upravené predikáty

- vzniknou z prediktátů tak, že
    - konstanty jsou přepsány na proměnné a jsou pro ně přidány podcíle
    - opakované proměnné A rozdělíme na dvě A1, A2 a přidáme pravidlo A1 = A2
- je-li pravidlo bezpečné potom i upravená verze je bezpečná
- pravidlo i jeho upravená verze jsou splnitelná pro stejné hodnoty

Datove typy a operace v relacni algebre prostorovych DB

- GEO - všechny geometrické objekty
    - POINT - bod
    - EXT - extend (extended object) - objekt který není bezrozměrný

    - LINE - lomená úsečka
    - REG - region = oddíl - polygon bez děr, jehož hranice sebe samu neprotíná
        - AREA - vzájemně se nepřekrývající regiony (jejich průnik je vždy prázdný) (ohraničená plocha?)
        - PGON - obecné regiony, mohou se vzájemně překrývat (pouze hranice bez plochy?)

Predikáty (operace jejichž výstupem je BOOL)

- POINT x POINT -> BOOL (je rovno/není rovno)
- LINE x LINE -> BOOL (je rovno/není rovno)
- REG x REG -> BOOL (je rovno/není rovno)
- GEO x REG -> BOOL (je uvnitř)
- EXT x EXT -> BOOL (mají neprázdný průnik)
- AREA x AREA -> BOOL (sousedí s)

Geometrické relace (operace jejichž výstupem jsou GEO)

- LINE* x LINE* -> POINT* (průnik, výstupem všechny body, kde se úsečky protínají)
- LINE* x REG* -> LINE* (průnik, výstupem výřezy lomených úseček)
- PGON* x REG* -> PGON* (průnik oblastí, jen jedna z nich může být nepřekrývající)
- AREA* x AREA* -> AREA* (překrytí)
- EXT* -> POINT* (uzly grafu)
- POINT* x REG -> AREA* (voronoi - rozdělení oblasti na nepřekrývající se části podle bodů)
- POINT* x POINT -> REL (nejbližší)

Operace vracející atomické objekty (jediný objekt)

- POINT* -> PGON (konvexní obálka)
- POINT* -> POINT (střed)
- EXT -> POINT (střed)

Operace vracející číslo

- POINT x POINT -> NUM (vzdálenost dvou bodů)
- GEO x GEO -> NUM (minimální nebo maximální vzdálenost objektů)
- POINT* -> NUM (průměr)
- LINE -> NUM (délka)
- REG -> NUM (plocha nebo obvod)

<u>Ake typy pravidiel pozname v deduktivnych db + u ktorych hrozi riziko bezpecnosti a kedy</u>

bezpečná pravidla - obsahují na pravé straně proměnné z levé strany

- neodpoveda na otazku V.a.Z.

1. opravný

- **Vysvetlit algoritmus R-Tree**
- **Popsat rozdil relacni, objektova, relacne-objektova + DDL, DML, v jakem roce, ...**
- **Definice R-plochy a vnoreni dvou R-ploch - uz predtim**
- **Popsat deduktivni DB**
- **Cosi ohledne deduktivni nerekurzivni db bez negace .. odvodit**
- **Co je shlukovani - definice, popis, vyznam**
- **Problemy pri implementaci prostorovych DB - jake (napsat v bodech bez reseni) + jake mame moznosti jak to implementovat**
- **Popsat snimkovou DB**

řádný

- **Two level grid file u prostorových DB - uz predtim**
- **Objektové rysy u SQL 1999 (a pozdějších)**
- **Simplexy (+ problémy při ukládaní prostorových dat v DB)**
- **Tabulka platnosti + překlad nějakého temporálního dotazu**
- **Omezovaná proměnná a její využití v deduktivní DB**
- **Omezeni historie v temporálních DB**
- **Minimální model, co způsobuje u DB s negací**
- **Generický dotaz**

Generičnost dotazů = dotaz je generický, pokud jeho výsledek nezávisí na způsobu uložení dat v DB:

pokud je v db uložena fakta (a,[0,3]) (a platí od 0 do 3) nebo (a,[0,2]),(a,[1,3]), tak v první případě se jedná jen o shluknutí intervalů pro fakt a

ale pro dotaz: ∃i,j. ∃x(R(i,x) && R(j,x)) && i != j) platí v druhém případě, ale v prvním jen díky shluknutí ne

- **bonusová, o které všichni věděli jaká bude:) - NoSQL**

2010/2011

1. opravný

- **Co je to shlukování - popsat všechny pojmy**
- **Problematika ukládání prostorových dat v pc + návaznost na algebru a operace**
- **XML v Oracle 10g**
- **LSD tree**
  - **Adaptivní K-D Tree**

- **Výškově vyvážený strom**
- **Local split decision**
- **Externí adresářová stránka**
- **Nejen pro prostorová data**

- **Tabulka s platným časem - vše + slovní popis nějakého zadaného dotazu**
- **Temporální integritní omezení**
- **K čemu je minimální model?**

řádný

- **V prostorovych DB definujte R-Tree**
- **Uplne formalne popiste genericky dotaz v temporalnich databazich**
- **Vysvetlete pojem aproximace v prostorovych databazich**
- **S cim se poji pojem omezeni v temporalnich databazich**
- **Vypiste objektove rysy SQL99 a pozdejsi**
- **Definujte minimalni model v deduktivnich databazich**
- **Snimkova DB, a vysvtleni dotazu ∃y.Hosp(x,y) ∧ □(¬(Hosp(x,y)) ∧ □(Hosp(x,y)))** pozn. Hosp(a,b) - osoba a hospitalizovana v nemocnici
  - Vypise osoby ktore su v nemocnici hospitalizovane a predtym bol cas ze tam hospitalizovane neboli a este predtym bol cas ze hospitalizovane boli **… :D**
  - Vypise osoby, ktere jsou hospitalizovane v nemocnici a jiz jednou v te stejne nemocnici byly hospitalizovany
- **Co je to je upravene pravidlo a jak jej ziskat v deduktivnich databazich**

Upravené predikáty

vzniknou z prediktátů tak, že

pro konstanty jsou přidány podcíle

opakované proměnné A rozdělíme na dvě A1, A2 a přidáme pravidlo A1 = A2

jeli pravidlo bezpečné potom i upravená verze je bezpečná

pravidlo i jeho upravená verze jsou splnitelná pro stejné hodnoty

Příklad

–p(a,X,Y) :- r(X,Y).

p(X,Y,X) :- r(Y,X).

•Upravená pravidla

–p(U,V,W) :- r(V,W), U=a.

p(U,V,W) :- r(V,U), W=U.

XML DB vs XML dokumenty, dotazování v XML.XML v orc10g III

typ XMLType - odpovídá standardnímu typu XML

SQL funkce a operátory pro práci s XML

Podpora XML Schema

Dualita XML a SQL - lze používat SQL operace nad XML daty a naopak

Podpora dotazování SQL/XML, XPath, XQuery

Organizace dat - hierarchická struktura pomocí XML DB Repository

v.z. tu sa pytam predsa na rozdiel medzi xml db a xml dokumentu….

---

simplexy

Jaké problémy se řeší při ukládání prostorových dat (ne jak se resi),

jaky byste zvolili zpusob ulozeni prostorovych dat a jaky by byl efekt tohoto ulozeni, navaznost - algebra a operace IIII

Omezeni historie v temporálních DB

---

integritni omezeni v temporalnich DB a jak se resi/implementuji (bez formalnich definic...) III

- integritni omezeni u temporalnich databazi - obmedzenia su uzavrene formule prveho radu temporalneho dotazovacieho jazyka,

pouzivaju sa pre zachytenie semantiky DB aplikacie a pre vhodny navrh DB v normalovych formach,snaha o dobre schema bez anomalii a dobru dekompoziciu

dovod zavedenia je,aby sa ukladali len "vyznamne" data

Tabulka platnosti + překlad nějakého temporálního dotazu II

---

minimalni a nejmensi model -vysvetlit a vyznam v jazyce s negaci, co zpusobuje u db s negaci IIII

- Model je interpretace (ohodnocení všech proměnných) taková, aby po dosazení formulí byl výsledek pravdivý

- Minimální model:

+ nechť M1,..Mn jsou modely množiny S wffs (Well-Formed Formula Set)

+ Mk je minimální model takový, že:

Mk je podmnožinou Mj, j€ {1,..,n}\{k}

Neexistuje M: M je modelem S a M je podmnožinou Mk

- problém v jazyce s negací:

p(x):- r(x) AND NOT q(x)

q(x):- r(x) AND NOT p(x)

pro DB: {r(1)} dostaneme 2 minimalní modely !

+ S1 = {q(1),r(1)}

+ S2 = {r(1),r(1)}

nejmenší model je model, který je minimální a je jenom jeden

Ake typy pravidiel pozname v deduktivnych db + u ktorych hrozi riziko bezpecnosti a kedy

nerekurzivne pravidla

– je mozne ich usporiadat tak, ze ak v p1..pn plati pi<pj, potom vedie cesta z I do j

-nesmu obsahovat negaciu

-pre kazde pravidlo je spocitana relacia tela predikatu, spoja sa vsetky podciele

-vypocet samotneho pravidla je projekciou relacie tela predikatu na premenne v hlavicke (+ dochadza k zjednoteniu vysledkov pre vsetky kombinacie)

Rekurzivní pravidla

- používá stejné proměnné na pravé i levé straně výrazu. Vyhodnocování je pak mnohem obtížnější.
- není možné uspořádat podcíle
- Nerekurzivní predikát lze převézt na dotaz v relační algebře.
- Rekurzivní predikát je potřeba vyhodnocovat odspodu.

Cosi ohledne deduktivni nerekurzivni db bez negace .. odvodit

---

Co je upravene pravidlo

upraveny predikat k predikatu p je novy predikat p(X1,…,Xk), kde

- Xj su rozne premenne
- ak je treba, su zavedene nove
- a na nich su vystavane nove podciele

mozne upravy:

- pre konstanty su pridane nove podciele
- opakované proměnné A rozdělíme na dvě A1, A2 a přidáme pravidlo A1 = A2

je-li pravidlo bezpečné potom i upravená verze je bezpečná

pravidlo i jeho upravená verze jsou splnitelná pro stejné hodnoty

---

Vyhodnocení nerekurzivních pravidel

- nesmí obsahovat negaci
- pro nerekurzivní program je možné uspořádat pravidla tak, že z každého pavidla existuje v grafu cesta do následujících pravidel

Vyhodnocení OUP (odovoditelne ulozene predikaty)

- pro každé pravidlo je spočtena relace těla predikátu (jde o operaci spojení výsledků všech podcílů)
- výpočet pravidla je pak projekcí relace těla predidkátu na proměnné v hlavičce s tím, že dochází ke sjednocení výsledků pro všechny kombinace
- podcíl je splněn pokud se v jeho relaci nachází n-tice s hodnotami parametrů

==========

DB s casem platnosti -popis, vlastnosti a jednou vetou popsat vyznam dotazu:

- jedná se o tzv. stavovou tabulku

- každý záznam je spjat s množinou všech disjunktních nesousedících intervalů,
nebo okamžiků (tabulka událostí), kdy záznam platil, nebo došlo k události
reprezentované záznamem
- přesnost časové informace je dáno při vytváření tabulky

Ey.(ZK(x,y)) ^ [](not(Zap(x,y) v Exp(x,y))

ZK(a,b) - znamena ze student a ziskal v danem roce zkousku z predmetu b

Zap(a,b) - znamena ze student a ziskal v danem roce zapocet z predmetu b

Exp(a,b) - znamena ze student a byl ze zapoctu pro predmet b v danem roce omluven, nebo predmet zapocet nema

E - existencni kvantifikator

^ - konjunkce ("a")

v - disjunkce ("nebo")

not - logicka negace

[] (vyplneny ctverecek) - spojka temporalni vyrok. logiky "platilo vzdy v minulosti"

==========

BONUS: proc neni vzdy vhodne pouzit ACID (obecne povedomi o ACID nepredpokladejte) [5b]

Např. vyžadování konzistence DB ve všech okamžicích může být zbytečně složité a pomalé, oproti např. BASE modelu. (nosql slajd 12)

Bonus - porovnat časy nenalezení položky v kD a adaptivní kD Tree

Bonus: NoSQL

NoSQL jsou DB podporující nerelační datový model (např. klíč-hodnota, graf, ...), většinou jsou open source.

LSD Tree

Je to stejné jako adaptivní strom ale je vyvážený jen po dané místo - drží si v paměti jenom vyváženou část a pak buď to skočí na disk kde zkoumá sekvenčně a nebo tam má externí stránku, kterou načte, v ní už je nějaký obecný podstrom (nevyvážený), tam provede pár jednoduchý porovnání.