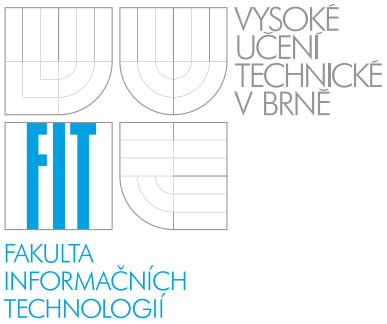


Pokročilé databázové systémy (prostorové databáze)

Studijní opora

PDB – I

Dušan Kolář
Ústav informačních systémů
Fakulta informačních technologií
VUT v Brně

Únor '06 – Říjen '06
Verze 1.0

Tento učební text vznikl za podpory projektu „Zvýšení konkurenceschopnosti IT odborníků – absolventů pro Evropský trh práce“, reg.č. CZ.04.1.03/3.2.15.1/0003. Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Abstrakt

Databázové systémy se často pojí s relačním schématem dat, zejména v jeho původním pojetí. S nástupem objektově orientovaných jazyků se zrodily objektové databáze a zdálo se, že podobně jako v programovacích jazycích začnou hrát rozhodující roli. Ukázalo se však, že tomu tak není. Mezitím systémy relační získaly řadu nových vlastností, takže vznikly databázové systémy objektově-relační, prostorové, temporální, či deduktivní. Buďto vznikly jako přímé rozšíření relačním, nebo vzájemnou kombinací těchto nových přístupů.

Tato publikace se bude blíže zabývat úvodem do prostorových databází. Provedeme základní charakteristiku tohoto typu databázových systémů, budeme specifikovat, v čem spočívá základní problematika těchto systémů a seznámí se s tím, jak se dají implementovat důležité části těchto systémů, zejména ty, které se odlišují od tradičního relačního schématu.

Vřelé díky všem, kdo mě podporovali a povzbuzovali při práci na této publikaci.

Obsah

Notace a konvence použité v publikaci

Každá kapitola a celá tato publikace je uvozena informací o čase, který je potřebný k zvládnutí dané oblasti. Čas uvedený v takové informaci je založen na zkušenostech více odborníků z oblasti a uvažuje čas strávený k pochopení prezentovaného tématu. Tento čas nezahrnuje dobu nutnou pro opakované memorování paměťově náročných statí, neboť tato schopnost je u lidí silně individuální. Příklad takového časového údaje následuje.

Čas potřebný ke studiu: 2 hodiny 15 minut

Podobně jako dobu strávenou studiem můžeme na začátku každé kapitoly či celé publikace nalézt cíle, které si daná pasáž klade za cíl vysvětlit, kam by mělo studium směřovat a čeho by měl na konci studia dané pasáže studující dosáhnout, jak znalostně, tak dovednostně. Cíle budou v kapitole vypadat takto:

Cíle kapitoly

Cíle kapitoly budou poměrně krátké a stručné, v podstatě shrnující obsah kapitoly do několika málo vět, či odrážek.

Poslední, nicméně stejně důležitý, údaj, který najdeme na začátku kapitoly, je průvodce studiem. Jeho posláním je jakýsi poskytnout jakýsi návod, jak postupovat při studiu dané kapitoly, jak pracovat s dalšími zdroji, v jakém sledu budou jednotlivé cíle kapitoly vysvětleny apod. Notace průvodce je taktéž standardní:

Průvodce studiem

Průvodce je často delší než cíle, je více návodný a jde jak do šířky, tak do hloubky, přitom ho nelze považovat za rozšíření cílů, či jakýsi abstrakt dané stati.

Za průvodcem bude vždy uveden obsah kapitoly.

Následující typy zvýrazněných informací se nacházejí uvnitř kapitol, či podkapitol a i když se zpravidla budou vyskytovat v každé kapitole, tak jejich výskyt a pořadí není nijak pevně definováno. Uvedení logické oblasti, kterou by bylo vhodné studovat naráz je označeno slovem „Výklad“ takto:

Výklad

Důležité nebo nové pojmy budou definovány a tyto definice budou číslovány. Důvodem je možnost odkazovat již jednou definované pojmy a tak významně zeštíhlet a zpřehlednit text v této publikaci. Příklad definice je uveden vzápětí:

Definice!

Definice 0.0.1 *Každá definice bude využívat* poznámku na okraji *k tomu, aby upozornila na svou existenci. Jinak je možné zvětšený okraj použít pro vpisování poznámek vlastních. První číslo v číselné identifikaci definice (či algoritmu, viz níže) je číslo kapitoly, kde se nacházela, druhé je číslo podkapitoly a třetí je pořadí samotné entity v rámci podkapitoly.*

Pokud se bude někde vyskytovat určitý postup, či konkrétní algoritmus, tak bude také označen, podobně jako definice. I číslování bude mít stejný charakter a logiku.

Algoritmus!

Algoritmus 0.0.1 *Pokud je čtenář zdatný v oblasti, kterou kapitola, či úsek výkladu prezentuje, potom je možné skočit na další oddíl stejné úrovně.*

Přeskoky v rámci jednoho oddílu však nedoporučujeme.

V průběhu výkladu se navíc budou vyskytovat tzv. řešené příklady. K jejich zadání bude jako jakékoliv jiné, ale krom toho budou obsahovat i řešení s nástinem postupu, jak takové řešení je možné získat. V případě, že řešení by vyžadovalo neúměrnou část prostoru, bude vhodným způsobem zkráceno tak, aby podstata řešení zůstala zachována.

Řešený příklad

Zadání: Vyjmenujte typy rozlišovaných textů, které byly doposud v textu zmíněny.
Řešení: Doposud byly zmíněny tyto rozlišené texty:

- Čas potřebný ke studiu
- Cíle kapitoly
- Průvodce studiem
- Definice
- Algoritmus
- Právě zmiňovaný je potom Řešený příklad

Některé informace mohou být vypíchnuty, či doplněny takto bokem.

V závěru každého výkladové oddílu se potom bude možné setkávat s opětovným zvýrazněním důležitých pojmů které se v dané části vyskytly a případně s úlohou, která slouží pro samostatné prověření schopností a dovedností, které daná část vysvětlovala.

Pojmy k zapamatování

- Rozlišené texty
- Mezi rozlišené texty patří: čas potřebný ke studiu, cíle kapitoly, průvodce studiem, definice, algoritmus, řešený příklad.

Úlohy k procvičení:

Který typ rozlišeného textu se vyskytuje typicky v úvodu kapitoly. Který typ rozlišeného textu se vyskytuje v závěru výkladové části?

Na konci každé kapitoly potom bude určité shrnutí obsahu a krátké resumé.

Závěr

V této úvodní stati publikace byly uvedeny konvence pro zvýraznění rozlišených textů. Zvýraznění textů a pochopení vazeb a umístění zvyšuje rychlost a efektivnost orientace v textu.

Pokud úlohy určené k samostatnému řešení budou vyžadovat nějaký zvláštní postup, který nemusí být okamžitě zřejmý, což lze odhalit tím, že si řešení úlohy vyžaduje enormní množství času, tak je možné nahlédnout k nápovědě, která říká jak, případně kde nalézt podobné řešení, nebo další informace vedoucí k jeho řešení.

Klíč k řešení úloh

Rozlišený text se odlišuje od textu běžného změnou podbarvení, či ohraničením.

Možnosti dalšího studia, či možnosti jak dále rozvíjet danou tématiku jsou shrnuty v poslední nepovinné části kapitoly, která odkazuje, ať přesně, či obecně, na další možné zdroje zabývající se danou problematikou.

Další zdroje

Oblasti, které studují formát textu určeného pro distanční vzdělávání a samostudium, se pojí se samotným termínem distančního či kombinovaného studia (distant learning) či tzv. e-learningu.

Kapitola 1

Koncepce textu

Čas potřebný ke studiu: 21 hodin 30 minut
Tento čas reprezentuje dobu pro studium celého modulu.

Údaj je pochopitelně silně individuální záležitostí a závisí na současných znalostech a schopnostech studujícího. Proto je vhodné jej brát jen orientačně a po nastudování prvních kapitol si provést vlastní revizi, neboť u každé kapitoly je individuálně uveden čas pro její nastudování.

Cíle modulu

Cílem modulu je nastínit problematiku prostorových databázových systémů. Především nastínit a rozebrat hlavní problematiku spojenou s implementací takových systémů. Krom hlavních problémů jsou naznačena i možná řešení a vhodné směry, které se mohou při implementaci uplatnit. Mezi řádky by pak mělo vyplynout, kdy se tyto systémy výhodně uplatňují a kde šetří práci na aplikační úrovni, což je společný rys všech postrelačních databázových systémů.

(Modul je možné považovat za do jisté míry encyklopedický.)

Po ukončení studia modulu:

- budete schopni definovat problematiku prostorových databázových systémů a provést jejich hrubou klasifikaci;
- budete schopni ukázat možná řešení, která se uplatňují v takových systémech;
- bude mít znalosti z principů pro rychlý přístup (indexaci) dat takového typu;

- budete schopni (v případě dalších znalostí z jiných oborů) rozvinout tyto prvotní znalosti na takovou úroveň, že budete schopni implementovat částečně, či úplně systémy řízení báze dat založené a využívající prostorová data.

Průvodce studiem

Modul začíná vymezením pojmu prostorových databázových systémů, kdy specifikujeme obecné vlastnosti postrelačního systému a nadále i specializujeme na systémy prostorové. Provedeme také jasné rozlišení od aplikací. Dále zmíníme hlavní úskalí v implementaci takových systémů a načrtneme možná řešení. V závěru potom zmíníme problematiku indexace a hlavní způsoby indexování prostorových dat.
Pro studium modulu je důležité mít znalosti z oblasti relačních databázových systémů a to jak z pohledu uživatele, tak i z pohledu implementace a teorie stojící za takovými systémy. Práce v postrelačním systému, který navíc nabízí modul s prostorovými rozšířeními je jasnou výhodou, protože je možné znalosti kombinovat z literatury k danému databázovému systému a, kromě toho, řadu nově nabitý znalostí ihned ověřit minimálně z uživatelsko-programátorského pohledu.

Tento modul je jedním z modulů vhodných pro studium pokročilých databázových systémů. Je možné jej studovat zcela nezávisle, nebo, dle doporučení níže, po zvládnutí objektově-relačního schématu. Dalšími doplňujícími moduly, které vyplňují náplň pokročilých databázových systémů jsou moduly zpracovávající problematiku temporálních, multimediálních, či deduktivních databázových systémů.

Návaznost na předchozí znalosti

Pro studium tohoto modulu je tedy nezbytné , aby studující měl vynikající znalosti relačního databázového schématu, jeho implementace a znalost jazyka SQL, ideálně ve variantě SQL/99. Pochopitelně krom znalostí teoretických je nutná i praktická zkušenost s prací v relační databází a s nějakým relačním SŘBD (Oracle, PosgreSQL, ...). Z praktických důvodů, i když to není nezbytné, je vhodná znalost objektově-relačního schématu, jelikož řada systémů (jmenujme např. Oracle), nabízí prostorová data právě přes takováto rozšíření.

1.1 Koncepce modulu

Následující kapitola vymezuje pojem prostorových databázových systémů a provádí jejich zjednodušenou klasifikaci.

Další kapitola se zabývá demonstrací možných problémů a jejich způsobů řešení a možné reprezentace prostorových dat pro vymezené

typy prostorových databázových systémů. Na závěr se budeme zabývat možnostmi jak prostorová data ukládat tak, aby bylo možné je indexovat a zmíníme některé způsoby indexace prostorových dat.

Informace předkládané v kapitolách budou zejména náročné na zapamatování, nicméně u indexačních algoritmů bude zcela nezbytné zvládnutí logiky a pochopení funkce těchto algoritmů jak v době vytváření indexu, tak i v době vyhledávání dat, nebo jejich rušení. Teprve takové zvládnutí umožní další případné rozšíření vlastností z oblasti indexace bez větších nároků, neboť principy dalších algoritmů se liší jen v drobnostech, nebo jsou kombinací těch uváděných.

1.2 Potřebné vybavení

Pro studium a úspěšné zvládnutí tohoto modulu není třeba žádné speciální vybavení. Je však *vhodné* doplnit text osobní zkušeností s nějakým databázovým systémem nabízejícím prostorová data k zpracování. Potom je nutné mít přístup k odpovídající výpočetní technice (běžně dostupné PC) a ke zdrojům (Internet) nabízející volně použitelné verze takových databázových systémů (Oracle pro osobní užití [Express Edition], PosgresSQL, a další).

Další zdroje

Prostorové databáze v sobě kombinují znalosti z řady oborů a proto je možné, že právě dostatečné znalosti v takovýchto oborech, či jejich doplnění bude dostatečné pro celkové zvládnutí (databázové systémy, analytická geometrie, indexační techniky pro vícedimenzionální data, problematika konečné reprezentace čísel v počítači, apod.).
Z literatury, která se specializuje na dané téma, je potom možné doporučit následující, i když je jistě možné nají i jiné zástupce na dané téma, kteří nabídnou dostatečně hlubokou informaci: Kim, W. (ed.): Modern Database Systems, ACM Press, 1995, ISBN 0-201-59098-0; Shekhar, S., Chawla, S.: Spatial Databases: A Tour, Prentice Hall, 2002/2003, p. 262, ISBN 0-13-017480-7.
V textu jsou také u vybraných klíčových termínů uváděny i odpovídající anglické termíny, které by měly umožnit rychlé vyhledání relevantních odkazů na Internetu, který je v tomto směru bohatou studnicí znalostí, jenž mohou vhodně doplnit a rozšířit studovanou problematiku.

Kapitola 2

Úvod

Tato kapitola prezentuje a vymezuje pojem prostorových databázových systémů, specifikuje základní problematiku a vybrané pojmy.

Čas potřebný ke studiu: 3 hodiny 10 minut.

Cíle kapitoly

Cílem kapitoly je naučit čtenáře definovat a vymezit charakteristické vlastnosti prostorových databázových systémů. Zejména potom určit jejich aplikační doménu a odlišit je od aplikací nad nimi budovanými. Kapitola proto definuje základní pojmy a problematiku spojenou s prostorovými databázemi. Dojde také k vymezení vlastností databázových systémů, které musejí splňovat, aby bylo možné je takto charakterizovat, ukážeme také úskalí „intuitivních“ definic rozšíření. Jedná se o úvodní zavedení problematiky, na kterém se bude dále stavět. Je tedy vhodné kapitolu nejdříve zvládnout, než přejdete dál.

Průvodce studiem

Pro studium této kapitoly je nutná dobrá orientace v relačních databázových systémech, způsobech reprezentace čísel v počítači a zvládnutí základních pojmů z geometrie. Pojmy z těchto oblastí kombinuje a dále rozvíjí a vymezuje pojem prostorových databázových systémů. Přitom definuje i některé klíčové problémy z jejich implementace (problém indexace je studován v poslední kapitole).
Ke studiu této kapitoly není třeba žádného speciálního vybavení. Jako u většiny ostatních se jedná o to si zapamatovat danou terminologii a pochopit klíčové principy předkládaných konceptů. Pro hlubší studium je vhodné využít informací na Internetu, nebo praktické zkušenosti s databázovým systémem podporujícím prostorová data — potom tedy ale budete potřebovat správně vybavené PC, případně přístup k Internetu.

Obsah

Výklad

2.1 Úvod k postrelačním databázím

Prostorové (spatial) databázové systémy, přesněji databázové systémy s podporou ukládání, zpracování a manipulace s prostorovými daty, jsou případem pokročilého databázového systému, jinými slovy tedy postrelačního databázového systému.

Definice 2.1.1 *Obecná definice (pracovní) postrelačních databázových systémů: Databázový systém, který už nevystačí se základním relačním schématem a bez přímé podpory na implementační úrovni pro uvažovanou specializaci je zpracování „jiných“ (specializovaných, pro systém přirozeně neznámých) dat velmi neefektivní.¨*

Definice!

Často se může jako jednodušší jevit popis toho, co *nejsou* postrelační databázové systémy — nejsou to systémy, které zapojené do určité aplikace, na základě vlastností této aplikace pouze budí dojem, že podporují manipulaci nějakých dat, ale tato podpora je zabudována na *aplikační úrovni* a samotný systém slouží jen jako úložiště hrubých, nestrukturovaných dat.

co nejsou postrelační DB

Je tedy nutné rozlišit, k čemu přistupujeme a podle toho kategorizovat jak systém, tak případnou aplikaci. Kromě prostorových patří mezi postrelační například i tyto databázové systémy:

- Objektově orientované
- Deduktivní (deductive)
- Časové (temporal)
- Multimediální
- Aktivní

Jejich uplatnění najdeme v řadě odvětví a i když data jsou na relační úrovni, tak užití spouštěcích mechanismů (triggers) a dalších vlastností už dávno nespadá jen do čistě relačních databází, ale posouváme se k aktivním, takže v současnosti jsou postrelační, tedy pokročilé databázové systémy prakticky všude.

Vývoj takového systému může probíhat obecně dvojím způsobem:

vývoj

1. modifikací existujícího jazyka či databázového systému;
2. vývojem „na zelené louce“.

Druhý způsob přitom dnes asi nepřipadá v úvahu.

Pokud bychom pokročilý databázový systém odvozovali z nějakého existujícího programovacího jazyka, tak zřejmě proto, že by disponoval typem a manipulací dat, která jsou pro nás zajímavá. Potom by bylo třeba zavést úpravy v jazyku pro jasné rozlišení práce s persistencí dat a samotnou persistenci dat zabezpečit. Zejména druhý úkol ve spojení s vyhledáváním a práci nad daty je poměrně náročný.

Kdybychom však modifikovali existující databázový systém a chtěli ho rozšířit o nová data, tak situace je jen zdánlivě jednodušší, neboť tvorbě indexačních algoritmů a úpravě jazyka pro definici a manipulaci dat se také nevyhneme a navíc budeme muset implementovat i všechny funkce a operace pracující nad novými daty.

2.2 Prostorové databáze — úvod

Základní charakteristika

charakteristika

Prostorové databáze jsou především databázové systémy schopné spravovat data, která se váží k určitému prostoru, bez ohledu na to, jak veliký ten prostor je. Z toho je možné odvodit i charakteristiku podpůrné technologie, kterou však na uživatelské úrovni vidět být nemusí (pozn.: viz např. rozdíl mezi konceptuální a implementační úrovní): je to schopnost spravovat velké množství relativně jednoduchých geometrických objektů.

prostor

O prostoru přitom mluvíme až od dvou dimenzí, i když, přísně vzato, jsou i prostory méněrozměrné. Příklady uplatnění a stupeň dimenze prostoru jistě najdete sami. Mezi největší chyby přitom patří, když si dáte rovná se mezi prostorové databázové systémy a geografické informační systémy (GIS). Chyba je už v tom, že první je typ SŘBD (DBMS), zatímco druhé je aplikace, která může s výhodou užít prostorové SŘBD.

prostorové databázové systémy a GIS

prostorové databázové systémy a obrázkové databáze

Stejně tak chyba z druhé strany, kdy obrázky prostoru uložené v databázi někdo považuje za prostorovou databázi, je častým jevem. Přitom se jedná spíše o aplikaci z oblasti multimediálních databází. Pokud však vezmeme takové obrázky jako základ, analyzujeme je, získáme údaje o objektech v nich zachycených, o vztazích mezi těmito objekty, tak potom, pokud tyto uložíme do databáze, získáme prostorovou databázi (pokud to tedy SŘBD podporuje).

V prostorových databázích tedy nalezneme *množiny entit z určitého prostoru*, u kterých je zřejmá:

- identifikace,
- umístění,

- vztah k okolí.

Pracovně si tedy můžeme prostorové databázové systémy definovat takto:

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

Definice!

Pozn.: V GIS je na prvním místě model prostoru, ten může být rastrový*, nebo* vektorový*. Vektorový model se potom může využít při užití prostorového databázového systému pro vytvoření příslušné aplikace. Vektorový model lze dále rozdělit na* špagetový *a* topologický*. U prvně jmenovaného ukládáme pouze entity „drátového modelu“ (body, linie, ...) bez dodaných topologických vztahů. Další text se bude zabývat popisem prostorových databázových systémů, které jsou zejména využitelné při vektorovém modelování v GIS. I když se terminologie může prolínat, tak je však třeba mít na paměti, že GIS je aplikace a prostorový DB systém je typ SŘBD.*

Výklad

Modely v prostorových DBS

Při tvorbě prostorového DBS je třeba nejdříve zodpovědět na několik otázek. Zejména to jsou tyto:

1. Jaké entity je třeba ukládat?
2. Jaké je možné užít geometrické modely?
3. Jaké jsou k dispozici prostorové typy/algebry?
4. Jaké vztahy mezi entitami chceme/potřebujeme zachytit?
5. Jak je geometrie integrována do SŘBD a jeho datového modelu?

Pokud si uvědomíme, co můžeme chtít ukládat, tak jsou to v zásadě dvě věci:

- entity v prostoru — města, řeky, domy, cesty, atomy, planety, ...
- prostor jako takový — vazby mezi entitami, vztahy v prostoru, často se jedná o entity vyšších dimenzí (plochy, objemy, ...), např. tématické mapy (typy rostlin, státy, povodí, ...)

Co se týká geometrických modelů, tak musejí být schopny zachytit to, co chceme uložit. Jednak tedy oddělené entity a potom také skupiny entit, které spolu nějak prostorově souvisejí. Pro oddělené entity jsou geometrické modely typicky tyto:

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

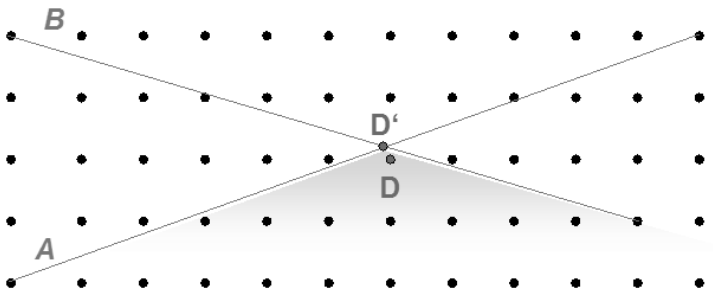
Pro popis prostoru potom můžeme uvážit např.:

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

O tom, jak jednotlivé abstraktní entity, užité pro modelování prostoru, zachytíme, rozhoduje zvolený model prostoru. Často budeme uvažovat Euklidovský prostor, který je spojitý a ve dvou dimenzích je každý bod p definován souřadnicí dvou bodů: $p = (x, y) \in \mathcal{R}^2$, kde $\mathcal{R}$ označuje množinu reálných čísel.

problém reprezentace souřadnic

Zde vystupuje první vážný problém, který musí být na implementační úrovni u prostorových databázových systémů spolehlivě řešen. V počítači totiž nejsme schopni reálná čísla zachytit, pouze čísla racionální a velmi často jen čísla desetinná, s různou míry přesnosti. Zachycujeme tedy diskrétní prostor.



Obrázek 2.1: Diskrétní zachycení spojitého prostoru

Obrázek 2.1 zachycuje situaci, kdy koncové body úseček A a B jsou zachyceny v síti, ale jejich průsečík nikoliv — tzn. nelze jeho hodnotu reprezentovat, není v oboru povolených hodnot. Jak třeba potom můžeme odpovědět otázku, zda bod D leží na úsečce A? Nebo to, zda bod D je zcela obsažen v oblasti pod úsečkami A a B (naznačena lehkým podstíněním)? Při dané reprezentaci desetinných čísel a dané diskretizaci prostoru tyto otázky spolehlivě odpovědět nelze.

Řešení problematiky je v tom, že v průběhu geometrických operací se již nadále neprovádí další výpočty průsečíků. Dochází tak k oddělení typů a operací nad prostorovými daty a ošetření číselných problémů korektně ke geometrickému modelu. A právě geometrický model je onou hranicí, která obě části odděluje a značným způsobem i vymezuje.

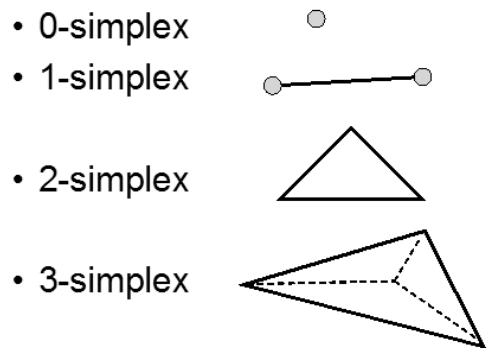
oddělení číselných problémů

Mezi hlavní přístupy, jak toto řešit je možné uvést:

- Použití jednoduchých geometrických entit pro skládání složitějších objektů (Frank & Kuhn 1986; Egenhofer, Frank & Jackson 1989) — simplexy.
- Kompletní popis modelované oblasti (Güting & Schneider 93; Schneider 97) — úplné deskriptory/realms.

Simplexy jsou nejmenší nevyplněné objekty dané dimenze. Často se tedy označují jako d-simplexy. 0-simplex je potom bod, 1-simplex je úsečka, 2-simplex je trojúhelník, 3-simplex je čtyřstěn atd. Lze jednoduše vypozorovat, že d-simplex sestává z d+1 simplexů rozměru d-1. Takové tvořící elementy se potom nazývají styky (faces). Kombinace simplexů do složitějších struktur je povolena jen tehdy, pokud průnik libovolných dvou simplexů je styk.

simplexy



Obrázek 2.2: Simplexy

Úplné popisy, či deskriptory (realms) jsou vlastně jakýmsi souhrnným popisem všech objektů v databázi. Formálněji je to potom

deskriptory

množina bodů, úseček, případně vyšších celků, které mají tyto vlastnosti:

- každý (koncový) bod je bodem sítě
- každý koncový bod úsečky (složitějšího útvaru) je bodem sítě
- žádný vnitřní bod úsečky (složitějšího útvaru) není zaznamenán v síti
- žádné dvě úsečky (složitější útvary) nemají ani průsečík ani se nepřekrývají

Deskriptory však musejí na implementační úrovni řešit problémy s číselnou reprezentací a to se ne vždy daří zcela uspokojivě — aplikační data často obsahují průsečíky vnitřních bodů grafických elementů (úseček), ty však ale nemusí být v síti. Jednoduchá řešení v tomto případě neexistují, často není problém najít protipříklad, kdy navrhované řešení selže. Možné řešení však existuje, nicméně jeho složitost je nad rámec tohoto textu (zájemci mohou hledat práce Greene & Yao z roku 1986).

Pojmy k zapamatování

Mezi klíčové pojmy této kapitoly, kterým je nutné porozumět a je nutné si je zapamatovat, patří:

- postrelační databázové systémy, prostorové databázové systémy
- hlavní entity ukládané v prostorových databázových systémech
- geometrické modely — problematika reprezentace prostorových objektů
- simplexy
- deskriptory

Závěr

Cílem kapitoly bylo definovat a vymezit pojem postrelačních a potom zejména prostorových databázových systémů. Definovat typické objekty ukládané v takových databázích a s tím spojenou problematiku — co se bude ukládat, jaký se užije geometrický model a jaká problematika je s ukládáním entit spojena, jaké jsou způsoby řešení vzniklých problémů na úrovni základních ukládaných entit. Pochopení těchto základních definic by mělo umožnit správně zpracovat a pochopit následující kapitoly.

Úlohy k procvičení:

1. Jaké jsou typické rysy postrelačních databázových systémů? Jaký je rozdíl mezi prostorovým databázovým systémem a GIS?
2. Jaká problematika volby geometrického modelu se pojí s ukládáním prostorových entit do databáze? Vysvětlete!
3. Co to je simplex? K čemu slouží, proč vznikl?

Klíč k řešení úloh

Odpovědi na všechny otázky najdete v textu.

Další zdroje

Další informace k této kapitole najdete na internetu či v literatuře zabývající se prostorovými databázemi (uvedena v úvodní kapitole) nebo ve Güting, R.H.: Spatial Database Systems, Praktische Informatik IV, Fernuniversität Hagen, Germany. Pokud se setkáte v jistých směrech i s jinými možnými řešeními, nebo jinou klasifikací, tak to neznamená, že jedna, nebo druhá je špatně. Zkuste najít společné rysy a rozdíly v přístupu, zapátrejte po zdrojích, abyste si ujasnili genealogii jednotlivých řešení.

Kapitola 3

Typy, operace, podpora implementace

V této kapitole se seznámíte s problematikou definice typů a operací nad nimi, nahlédnete i na vybrané implementační mechanismy.

Čas potřebný ke studiu: 10 hodin 30 minut.

Cíle kapitoly

Cílem kapitoly je upozornit a přednést problematiku spojenou s rozšiřováním stávajících formalismů pro podporu práce s databázovým systémem, jako jsou relační kalkul či algebra. Dále se budeme zabývat změnami na úrovni DML a DDL a ukážeme některé nutné postupy na implementační úrovni.
Kapitola definuje vybrané pojmy a obraty spojené s rozšiřováním relační (či jiné) algebry/kalkulu. Poté demonstruje možnosti implementace rozhraní prostorového databázového systému i úpravy nutné na úrovni DML/DDL a penetraci úprav do implementace. Kapitola je završením předchozí a obě jsou nutné pro postup do další části.

Průvodce studiem

Pro studium této kapitoly je nutná dobrá orientace v relačních databázových systémech, relačním kalkulu a pojmech předchozí kapitoly. Pojmy z těchto oblastí kombinuje a dále rozvíjí a obohacuje.
Ke studiu této kapitoly není třeba žádného speciálního vybavení. Jako u většiny ostatních se jedná o to si zapamatovat danou terminologii a pochopit klíčové principy předkládaných konceptů. Pro hlubší studium je vhodné využít informací na Internetu, nebo praktické zkušenosti s databázovým systémem podporujícím prostorová data — potom tedy ale budete potřebovat správně vybavené PC, případně přístup k Internetu.

Obsah

Výklad

3.1 Typy a operace nad nimi

K tomu, aby bylo možné vkládat prostorová data do databáze a aby bylo možné nad nimi provádět požadované operace, je nutné, aby systém podporoval tvorbu dat daného typu a dále potom nabízel operace, kde takové typy vstupují jako parametry, případně vystupují jako výsledek.

Definice typů a operací však není jen intuitivní záležitostí. Dají se stanovit kritéria, která potom mohou nějaký návrh či systém ohodnotit z hlediska kvality a úplnosti podpory práce s prostorovými daty. Kritéria jsou:

kritéria pro systém

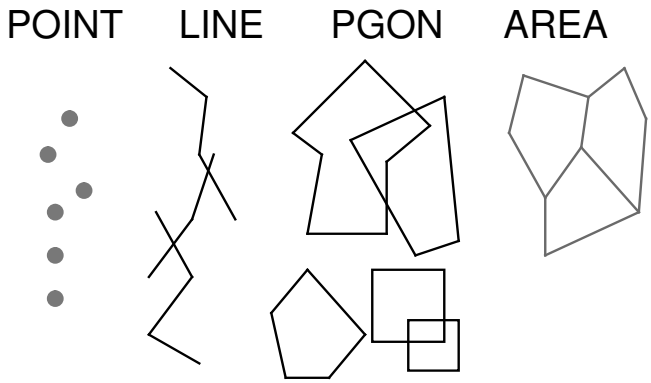
- „vzhled“ datových položek musí být uniformní v rámci množinových operací nad množinami objektů tvořící data i případný výsledek;
- systém musí obsahovat formální definice dat a funkcí nad prostorovými datovými typy;
- v předchozím bodě zmíněné definice musejí zohledňovat aritmetiku s konečnou přesností;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

Uvažme za základ, pro jednoduchost, relační databázový systém. Ten jako svůj formální základ využívá relační algebru, což je vícedruhová algebra (relace, atomické typy). Takovou algebru musím rozšířit o další typy tak, aby kromě atomických typů bylo možné zaznamenat i prostorová data. Vzhledem k předchozímu zvolíme jistou minimální množinu konstrukcí, které umožní sestavit prakticky libovolný objekt ve 2D:

ukázka prostorových typů

- bod (point)
- lomená úsečka (line)
- region, který se dále dělí na
  - uzavřená lomená úsečka (polygon, pgon) — význam hranice, obdoba kružnice

– ohraničená plocha (area) — plošně vyplněná oblast, obdoba kruhu



Obrázek 3.1: Ukázky prostorových typů

Pro systematizaci typů je možné definovat i společné pojmenování pro skupiny typů stejného charakteru. My zavedeme pro typy *PGON* a *AREA* společný nadtyp *REG* (z definice výše). Pro úsečky a objekty typu *REG* zavedeme společné označení *EXT* (obsahuje objekty, které nejsou bezrozměrné). No a pro všechny objekty zavedeme označení *GEO*.

kategorie operací

Pro definici operací potom je možné postupovat poměrně systematicky. V zásadě máme 3 kategorie operací, které je možné aplikovat:

1. **predikáty** — vstupem operací jsou různé hodnoty, z nichž alespoň jedna je nějakého z typů *GEO* a výsledkem je pravdivostní hodnota;

2. **geometrické relace** — vstupem je hodnota, či množina hodnot jednoho či více typů ze skupiny *GEO* a výsledkem je jedna či více hodnot z typu *GEO*, nebo relace, které takové typy obsahuje alespoň v jednom atributu;

3. **výpočetně náročné operace** — i operace z předchozí kategorie mohou být velmi náročné, ale typově se jedná o zjišťovaní vztahů, v této kategorii jsou výsledkem nově získané hodnoty různých i negeometrických typů, jedná se i o operace množinové.

Řešený příklad

Zadání: Pro každou kategorii operací nad prostorovými typy definovanými výše uveďte příklady operací.
Řešení: Pro každou kategorii ukážeme několik operací:

- Predikáty:
  - rovno, nerovno :: $POINT \times POINT \rightarrow Bool$
  - sousedí_s :: $AREA \times AREA \rightarrow Bool$
  - je_uvnitř :: $GEO \times REG \rightarrow Bool$
- Relace:
  - průnik :: $LINE^* \times LINE^* \rightarrow POINT^*$
  - překrytí :: $AREA^* \times AREA^* \rightarrow AREA^*$
  - voronoi :: $POINT^* \times REG \rightarrow AREA^*$
- Ostatní:
  - konvexní_obálka :: $POINT^* \rightarrow PGON$
  - vzdálenost :: $POINT \times POINT \rightarrow NUM$
  - plocha :: $REG \rightarrow NUM$

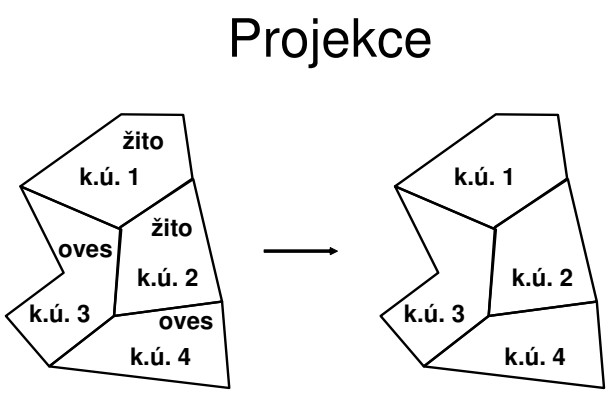
Výklad

Kromě včlenění nových operací do jazyků pro definici a manipulaci dat, je také nutné zajistit funkci již existujících operací nad novými daty, minimálně tam, kde to je možné.

Jedním případem je manipulace s plochami, přesněji oddíly (Scholl & Voisard 1989) v intenci základních operací relační algebry. Jsou formálně definovány následující operace plus další operace, jako výše:

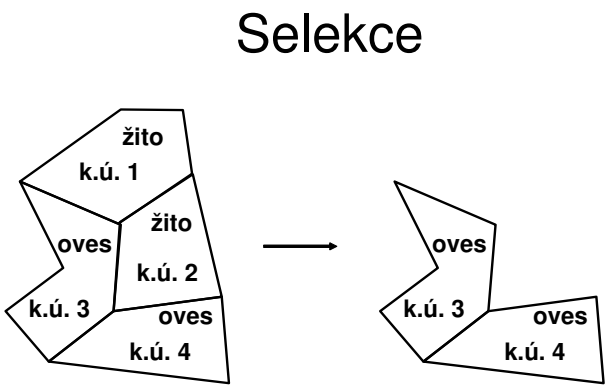
- projekce
- selekce
- fúze (projekce se spojením)
- „windowing“
- ořezání

To, jakým způsobem operace pracují, je ukázáno na obrázcích 3.2 – 3.6. Ač se u některých operací může zdát, že jsou triviální, tak je třeba vzít do úvahy, že se vždy jedná o operace nad grafickými objekty a s grafickým výstupem.



Obrázek 3.2: Manipulace s oddíly — projekce

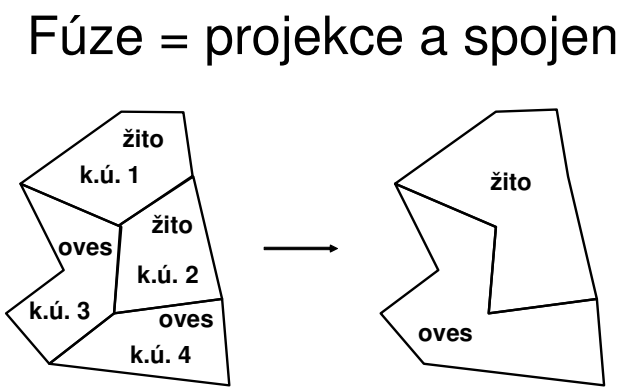
U projekce dochází k redukci atributů, které jsou zobrazovány a případně manipulovány dalším dotazem. Dochází však často k zachování původních oddílů/ploch. Naproti tomu selekce typicky povede k redukci zobrazovaného počtu oddílů.



Obrázek 3.3: Manipulace s oddíly — selekce

Operace kombinující dvě operace do jedné tak, aby poskytla optimální výsledek, je fúze. Jedná se o projekci, kdy oddíly, u kterých zbylé atributy jsou shodné hodnoty, jsou spojeny v jeden grafický objekt (je-li to možné). Tato operace klade na grafickou část výpočtu nesmírné nároky, neboť je nutné zjistit shodnost atributů a sousednost příslušných oddílů, což není časově nenáročná operace.

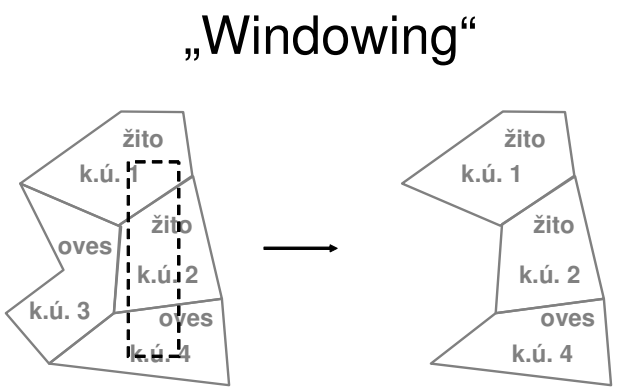
Operace „windowing“ vrací ty oddíly (kompletní), do kterých zasahuje určité okno. Tato operace vlastně dělá průnik na obsah s oknem na grafické úrovni, ostatní atributy nejsou dotčeny. Přímé užití bude asi malé, ale v návaznosti na další operace může být tato operace



Obrázek 3.4: Manipulace s oddíly — fúze

významná.

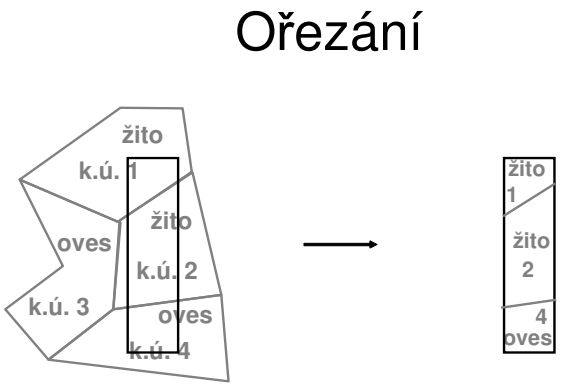
Jakýmsi doplňkem, či opakem operace „windowing“ je operace ořezání. Ta vybere do výstupu všechny atributy, které zasahuje okno, ale plošně objekty/oddíly upraví tak, aby výsledkem byla pouze ta jejich část, která se nachází uvnitř okna.



Obrázek 3.5: Manipulace s oddíly — „windowing“

Pro každý oddíl, jelikož je obdobou typu plocha (*AREA*), je možné definovat datový typ, říkejme mu, jako v jiných případech, region (v tomto případě označován jako γ nebo $\{\gamma\}$). Mapa, což je vlastně tabulka s prostorovými daty v relační databázi s podporou prostorových dat, je potom množina n-tic, kde jeden z atributů je právě tohoto typu region.

Algebra nad takovýmito objekty potom obsahuje (z hlediska prostorových dat) operátory vložení, primitivní operace nad regiony a množinové geometrické operace.

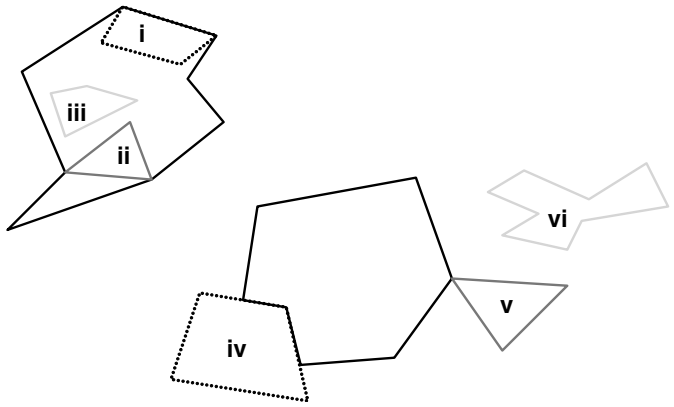


Obrázek 3.6: Manipulace s oddíly — ořezání

Výklad

Poslední zajímavou algebrou navrženou pro podporu prostorových databázových systémů vycházejících (do jisté míry) z relačních systémů je algebra ROSE (RObust Spatial Extension — Güting & Schneider 1995). Tato algebra vychází z deskriptorů (realms) a složitější objekty získává jejich skládáním. Základem jsou tedy body, úsečky a oblasti (plochy, které mohou mít i „díry“).

Pro definici vztahů, které jsou jedním z klíčových předmětů zájmu prostorových databázových systémů, je nutné zejména definovat způsoby chápání polohy 2 oblastí. A to jak při vnoření, tak pokud leží mimo sebe.



Obrázek 3.7: Míra vnoření a disjunkce

Obrázek 3.7 ukazuje jak míru vnoření, tak míru disjunkce. U vno-

ření hovoříme o, že objekt je:

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

Podobně, u míry disjunkce hovoříme o tom, že objekt je:

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

Na základě tohoto vnímání vnoření a disjunkce ploch potom můžeme učinit tuto neformální definici R-cyklu (uvědomte si, že pracujeme s deskriptory [realms], takže uzavřená lomená úsečka je složena z úseček splňující jistá pravidla — viz výše):

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

Definice!

Jelikož se často v modelované realitě nevyskytují oddělené plochy, tak budeme na základě předchozí definice specifikovat R-plochu, která již pojímá vnořené plochy.

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

Definice!

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

Pozn.: poslední podmínka zaručuje jednoznačnost reprezentace.

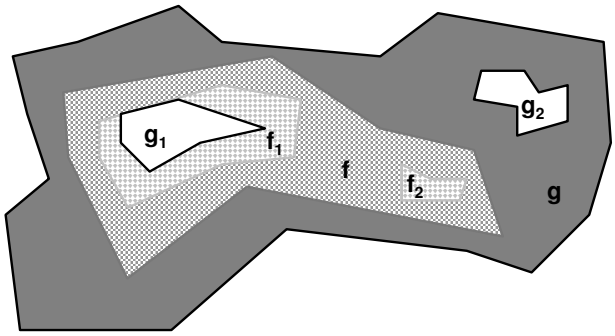
I takové plochy, které vyhovují definici R-ploch lze různě kombinovat, tedy vnořovat. O tom, kdy hovoříme o vnoření u dvou R-ploch, více následující definice. Jiná možnost, než definovaná potom není možná. U plošných útvarů založených na deskriptorech to však přináší řadu výhod ve zpracování.

Definice!

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

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

Obrázek 3.8 ukazuje dvě R-plochy f a g, přičemž přesně podle definice je R-plocha f plošně obsažena v R-ploše g.



Obrázek 3.8: Vnořené R-plochy

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

Definice!

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

Algebra ROSE specifikuje krom typů pro reprezentaci prostorových objektů i všechny operace s hodnotami těchto typů. I když byla navržena se zvláštní obezřetností, tak přesto má určité 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).

Výklad

3.2 Návrh prostředí, jazyka, implementace

Hloubka vztahů

Jak již naznačily předchozí výkladové části, tak jedním z důležitých aspektů je korektní práce s a určení vztahů mezi objekty v databázi. Jelikož možnosti v tomto směru jsou poměrně široké, tak pochopitelně se objevily myšlenky a práce, které zkoumaly, zda existuje nějaká horní mez pro definici vztahů dvou objektů v prostorech různé dimenze.

Průkopníkem v této oblasti byl Egenhofer, který nejprve studoval plošné objekty bez děr a vnořených objektů. Každý objekt rozčlenil na hranici a jeho vnitřní oblast. Čistě kombinatorickým způsobem sestavil tabulku, kde zapsal, zda se vnitřní oblasti dvou objektů vzájemně překrývají, zda se překrývají vnitřky s hranicemi a i samotné hranice. Takto dostaneme 16 kombinací, z nichž některé jsou nesmyslné. Nakonec se však ukázalo, že takovýto přístup je nedostatečný.

Postupně byl tento koncept dále studován a rozvíjen. Vrcholem bylo studium úrovně průniku pro bezrozměrné, jednorozměrné a dvourozměrné prostory a objekty (Clementini a kol. 1993). Ti nakonec stanovili 256 kombinací, z nichž bylo 52 platných. I když jejich výčet byl dostatečný, tak počet 52 byl příliš vysoký. Naštěstí se podařilo najít alternativu:

- 5 operací: dotek, uvnitř, přes, přesah, disjunkce;
- 3 operátory na extrakci hranice.

Lze dokázat, že tyto operace jsou vzájemně výlučné a spolu s operátory pro extrakci hranice umožňují rozlišit a realizovat všech 52 operací.

Další rozšíření následovala pro oblasti s dírami, skládané oblasti a další. Pro další studium je možné doporučit společné práce Egenhofera, Clementinihp, Di Felice, Califana, případně Papadiase z let 1994 a 1995.

Integrace do SŘBD a jazyka

Na to, abychom byli schopni vložit prostorová data do databáze, tak SŘBD musí podporovat tvorbu takových dat, kdy aspoň jeden z atributů je definován nad prostorovým typem. V relačních databázích to často znamená, že právě jeden z atributů nějaké relace je prostorového typu.

Integrace prostorových dat do SŘBD po sestavení algebry pro manipulaci s objekty, či množinami všech typů, tj. i prostorových, pokračuje implementací vstupních a výstupních operací pro nové, tedy prostorové datové typy a rozšířením jazyků pro definici a manipulaci dat.

Operace, které se musejí nyní do jazyka dostat, jsou již dobře známé:

- prostorová selekce — selekce spočívající na prostorovém predikátu;
- prostorové spojení (join) — spojení založené na predikátu testujícím prostorové atributy;
- aplikace prostorových funkcí — mapování operací a podmínky při selekci;
- a další množinové operace.

Řešený příklad

Zadání: Uveďte příklady operací prostorové selekce, spojení, aplikace prostorové funkce. Pro jejich složitost v napojení na algebru, pouze uveďte některé množinové operace. Užijte vhodně rozšířenou relační algebru.
Řešení: Nejdříve budeme definovat relace, nad kterými budeme operace demonstrovat:

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

Prostorová selekce: Vyhledej všechny řeky, které protínají dotazovací okno — rivers **select**[route **intersects** Window]
Prostorové spojení: Najdi všechny města v blízkosti řek —
cities rivers **join**[**dist**(center, route) < 20]

Aplikace prostorové funkce: Ke každé řece protékající Moravou vypiš jméno, kudy protéká a délku příslušné části —
rivers **select** [route **intersects** Morava]
  extend [**intersection** (route,Morava) {part}]
  extend [**length**(part) {plength}]
  project [rname, part, plength]
Množinové operace: překrytí (přenesení), spojení (fúze), voronoi.

Pro plně funkční implementaci však nestačí jen výše jmenované úkony a začlenění operací. Často je třeba si uvědomit, že součástí je i grafická prezentace, interaktivita, tvorba dotazů na základě okamžitých výsledků apod. Požadavky na práci s prostorovými datovými typy stanovil Egenhofer už v roce 1994 takto:

- Existence prostorových datových typů — zcela základní a nutný požadavek.

- Grafické znázornění výsledků — je třeba si uvědomit, že i když je možné hodnoty prostorových typů reprezentovat textově, tak především grafická reprezentace je žádána a očekávána.

- Grafická kombinace několika výsledků — jednotlivé dotazy často poskytují jen dílčí výsledky, kombinace více dotazů může tak poskytnout mnohem lepší obrázek; např. zobrazím část území a na něm jen cesty a sídla, v dalším kroku však chci doplnit řeky apod.

- Zobrazení i s kontextem — výběrem jednoho města jistě nemyslíme zobrazení jen údajů po jeho hranice, ale i přilehlé okolí, protože další náš zájem může být právě tímto směrem.

- Kontrola stavu displeje — díky různým operacím na displeji (výběr objektů do dotazů apod.) je možné, že dojde k poškození některých grafických objektů, případně je možné, že poslední dotaz nedodal informaci, co bychom potřebovali, potom je nutné obnovit původní stav displeje, případně se vrátit o několik kroků zpět apod.

- Dialog — systém by měl vést s uživatelem/programátorem dialog, neboť výsledek často není získán jediným dotazem.

- Různé typy zobrazení — způsob zobrazení jednotlivých typů údajů by měl být volitelný a i okamžitý způsob by se měl dát měnit (např. drátový model v 3D, stínovaný, realistický).

- Legenda, popisky — u řady údajů od místopisu až po třeba modely chemických sloučenin, např. DNA, jsou často doplněny textovými údaji. Jejich vhodné a správné zobrazení, stejně tak jako třeba údaje o měřítku by mělo být samozřejmostí.

- Změna měřítka — možnost změny měřítka je známou funkcí u řady programů s grafickým uživatelským rozhraním, tedy přirozeně i v prostorových databázových systémech.

- Výběr podoblastí — podobně možnost práce jen s určitou částí celé scény je již standardem a mělo by být dostupné ve všech systémech.

Výklad

Grafické uživatelské rozhraní

systém se 3 okny

Pro plně vybavený systém pro podporu práce s prostorovými databázemi je nutné mít i komfortní systém pro zadávání dotazů, zobrazování jejich výsledků apod. Za jistý standard by se dal považovat systém, který má typicky 3 okna.

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

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

nastavení zobrazení

To, jakým způsobem se zobrazí prostorová data ve výstupním okně, závisí na řadě skutečností. Ideální však je, když si to může nastavit uživatel. V podstatě má k dispozici tyto možnosti, nebo jejich kombinaci:

- popis vzhledu výsledku součástí dotazovacího jazyka;
- oddělený jazyk — GPL, graphical presentation language;
- definován v rámci GUI — součástí GUI je i dialog, který umožňuje nastavení způsobu zobrazení.

Datové struktury a algoritmy

Na úrovni implementace se krom řady dílčích vyskytují 2 principiální problémy:

1. implementace prostorové algebry takovým způsobem, že do dotazovacího systému lze „lehce“ začlenit
   - reprezentaci prostorových datových typů
   - algoritmy/operace nad prostorovými datovými typy
2. predikáty v množinově orientovaných DML — zejména se jedná o prostorovou selekci, spojení a množinové operace.

Důvodem je to, že typy pro prostorová data mají nestandardní a proměnlivou velikost a operace jsou silně závislé a konkrétní hodnotě typu (např. vzdálenost se jinak počítá pro 2 body a jinak pro polygon a kružnici).

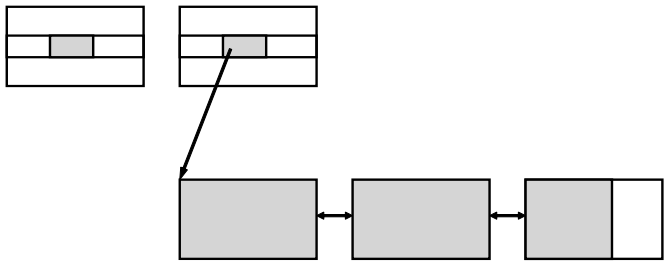
Co se týká reprezentace hodnot datových typů je navíc nutné udržet kompatibilitu mezi dvěma hledisky:

kompatibilita datových typů

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

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

Pro prostorová data se volí takový způsob uložení, aby byl konzistentní pro různou velikost dat.



Obrázek 3.9: Uložení prostorových dat

Pro každá prostorová data dojde k vyčlenění té částí dat, které se nemění s charakterem vkládaného objektu. Ostatní data se ukládají mimo, do zvláštních datových stránek, které tak leží mimo a nebrání rychlému zpracování (viz obrázek 3.9 — pokud data nepřekročí určitou velikost, tak jsou uložena stejně, jak ostatní a v rámci jedné stránky na disku/v paměti je možné mít více datových záznamů, jakmile však délka přeroste jistou mez, tak je uložen odkaz na souvislou oblast na disku, kde jsou velká data uložena za sebou).

Implementačně by se separace dat mohla např. realizovat tak, že pro každý prostorový datový typ, kde není zaručena konstantní velikost, oddělíme konstantní část od části proměnlivé a v části implementace persistence datové položky zajistíme různé uložení.

Řešený příklad

Zadání: Definujte typ pro uzavřenou lomenou úsečku (polygon) tak, aby se dal užít v implementaci prostorových datových typů.
Řešení: Pro definici typu užijeme hypotetický programovací jazyk:

```
type polygon = pointer to record
  area: real;
  perimeter: real;
  bbox: rectangle;
  num_of_vertices: integer;
  vertices: array[1..1000000] of point
end
```

Všechny položky až na poslední (vertices) jsou uloženy v základní části dat, neboť jejich velikost se nemění s velikostí uloženého polygonu.

Aby bylo možné dostát podpoře datových typů na co nejlepší úrovni, tak je nutné reprezentaci datových typů často dále vylepšit tak, že krom základních informací (viz řešený příklad) bude obsahovat i tyto informace:

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

práce s aproximacemi

Jelikož se aproximace zavádějí do datových typů, tak je nutné, aby s těmito aproximacemi počítaly i všechny operace. Pokud, navíc, mohou pomoci nějaké hodnoty, které jsou konstantní v závislosti na vložených datech, v dalších výpočtech, tak je žádoucí jejich maximální využití.

využití způsobu uložení dat

Podobně se dá těžit i z charakteru modelu pro ukládání dat, například pro deskriptory (realms) platí, že se nesmí žádné průsečíky v uložených datech vyskytovat. Předpočítavají se tedy v době uložení

dat do databáze. Potom např. vyhledání množiny průsečíků dvou lomených úseček můžeme zpracovat s časovou složitostí řádu $O(n+k)$, zatímco normálně bychom tutéž informaci získali s časovou složitostí $O(n.log(n)+k)$.

Pojmy k zapamatování

Tato kapitola je poměrně rozsáhlá a mezi klíčové patří zejména *pojmy spojené s problematikou*:

- definice operací a typů pro prostorové datové typy, zejména:
  - kategorie datových typů
  - kategorie operací nad novými typy
  - realizace standardních operací nad novými typy
- formalizace popisu vztahu dvou prostorových, často plošných útvarů, zejména algebra ROSE
- nezbytná hloubka studia vztahů objektů
- integrace operací do SŘBD, zejména tedy standardních
- grafická prezentace výsledků dotazů
  - požadavky
  - GUI
- začlenění prostorových algoritmů do SŘBD
- začlenění prostorových dat do SŘBD

Závěr

Tato poměrně rozsáhlá kapitola prezentovala jisté mezikroky, které jsou nutné pro začlenění nových datových typů, přesněji prostorových, do stávajícího SŘBD. Zejména se jedná o kroky, kdy se definují typy a nové operace nad nimi, dále se definují standardní operace pro nové typy, dává se tak operacím nová sémantika pro prostorové datové typy. Pro topologické operace je nutné další studium vztahů objektů a další definice pro korektní funkci systému. No a ve finále potom o implementaci zobrazení a vstupu prostorových dat a vlastní začlenění dat a algoritmů do implementace SŘBD. Celá tato kapitola tak naznačuje cestu od záměru k realizaci a je cestou k poslední kapitole, kde se budeme bavit o indexačních algoritmech, které jsou nutné pro efektivní realizaci řady operací zavedených v kapitole této.

Úlohy k procvičení:

1. Jaká jsou kritéria pro stanovení typů pro uložení prostorových dat v DB?
2. Jaké kategorie operací přidáváme do jazyka, abychom umožnili manipulaci s prostorovými daty?
3. Jak lze u ploch definovat jinak standardní operace z relační algebry jako projekce a selekce? Je možné definovat i další operace? Jaké? Vysvětlete jejich činnost.
4. Co to je R-cyklus, R-plocha a kdy jsou dvě R-plochy vnořené?
5. Jak precizně je nutné zkoumat vzájemnou polohu dvou objektů, abychom přesně určili, v jakém jsou vztahu?
6. Navrhněte možnost, jak definovat způsob zobrazení prostorových dat? Uveďte na příkladech.

Klíč k řešení úloh

Odpovědi na všechny otázky najdete v textu.

Další zdroje

Další informace k této kapitole najdete na internetu či v literatuře zabývající se prostorovými databázemi (uvedena v úvodní kapitole) nebo ve Güting, R.H.: Spatial Database Systems, Tutorial Notes, Fernuniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany, případně v Schneider, M.: Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS, FernUniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany. Pokud se setkáte v jistých směrech i s jinými možnými řešeními, nebo jinou klasifikací, tak to neznamená, že jedna, nebo druhá je špatně. Možnosti, jak daný problém řešit, je celá řada.

Kapitola 4

Indexační algoritmy

Tato kapitola prezentuje vybrané algoritmy pro indexaci v prostorových databázích.

Čas potřebný ke studiu: 7 hodin 50 minut.

Cíle kapitoly

Mezi hlavní cíle této kapitoly patří zejména prezentace problematiky tvorby indexu pro prostorové databázové systémy a potom prezentace samotných algoritmů, které se pro indexaci využívají.
V kapitole se tedy setkáme s možnými přístupy k řešení základního a principiálního problému tvorby indexu nad prostorovými daty. Tato kapitola je vyvrcholením předcházejících kapitol a je jejich logickým zakončením. V případě zájmu je však možné ji studovat odděleně, pouze s limitovanými znalostmi, ale nelze tento postup doporučit.

Průvodce studiem

Pro studium této kapitoly je nutná dobrá orientace v relačních databázových systémech a zejména indexačních technikách v nich standardně využívaných. Taktéž znalosti z teorie grafů je možné s úspěchem využít.
Ke studiu této kapitoly není třeba žádného speciálního vybavení. Jedná se zejména o pochopení principů a způsoby práce algoritmů, případně o zapamatování si definovaných pojmů. Pro hlubší studium je vhodné využít informací na Internetu, nebo praktické zkušenosti s databázovým systémem podporujícím prostorová data — potom tedy ale budete potřebovat správně vybavené PC, případně přístup k Internetu.

Obsah

Výklad

4.1 Úvod

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

Možnosti, jak indexaci zvládnout, jsou v zásadě 2:

- mapování vícerozměrných údajů do jednoho rozměru a užití známých metod pro tvorbu indexů;
- vystavění specializovaných indexů pro prostorové datové struktury.

Je jasné, že asi každý přístup má svá pro i proti. Ten dříve jmenovaný zcela jistě vede k rychlejší a jednodušší implementaci, nicméně je otázka, jaké bude poskytovat výsledky. Druhý přístup může nabídnout zajímavé výsledky, ale implementace takových algoritmů a jejich robustnost zase nemusí být vždy uspokojivá.

Jak uvidíme níže, tak ve velkém množství případů bude rozhodovat to, jaký charakter má aplikace a jaká data budeme ukládat. Přičemž typické operace snad ani není třeba připomínat, jsou to:

- vkládání
- mazání
- dotazování
  - existenční
  - jako takové

Pro indexování se pak využívají vždy již známé obecné přístupy: tabulka s rozptýlenými položkami (hash table) v lineární i adaptivní formě, stromy. Přičemž implementace pro prostorová data se od těch standardních budou velmi zásadně lišit, takže např. dobře známý B-strom tu nikde neuvidíme, i když stromových algoritmů je v této oblasti celá řada.

4.2 Problém

Pro jednorozměrná data platí, že je možné, třeba i po jednoduchém zakódování, definovat nad těmito daty jednoznačné uspořádání. Takže

máme-li hodnotu x, tak bez většího úsilí a zcela jednoznačně určíme hodnotu předchůdce či následníka. Pro číselné hodnoty, na které často kódujeme, je to prostě $x - 1$ a $x + 1$. Díky této vlastnosti lze stanovit okolí každého bodu libovolně široké, pro vyhledávání a indexování obecně byly vytvořeny pokročilé algoritmy.

Pro data ležící ve 2, 3, ... rozměrech však něco takového neexistuje! A to bez ohledu na to, že lze často namapovat, nebo modelovat aplikace s daty ve více rozměrech s pomocí množiny celých, přirozených, nebo racionálních čísel.

4.3 Řešení

Mapování do 1D

Nejjednodušší je využití možnosti, že lze namapovat data do jednorozměrného prostoru. Touto transformací však zcela jistě ztratíme sousednost. Krom toho, taková transformace nemusí být vůbec realizovatelná. A pokud i je, tak výsledek dotazů nemusí být transformovatelný zpět, takže lze výsledek jen těžko interpretovat.

Stromy dělící prostor

Úplně základní algoritmy indexují v prostoru především bodové údaje. Minimálně lze totiž každý objekt rozdělit na body, kterými lze takový objekt plně popsat. Potom stačí využít body pro indexaci a uložit někde informaci o tom, které body tvoří jaký objekt a jak.

Mezi základní algoritmy založené na stromových strukturách, které lze využít i pro indexování ve vícerozměrném prostoru patří tyto (uvedeny anglicky v původním znění):

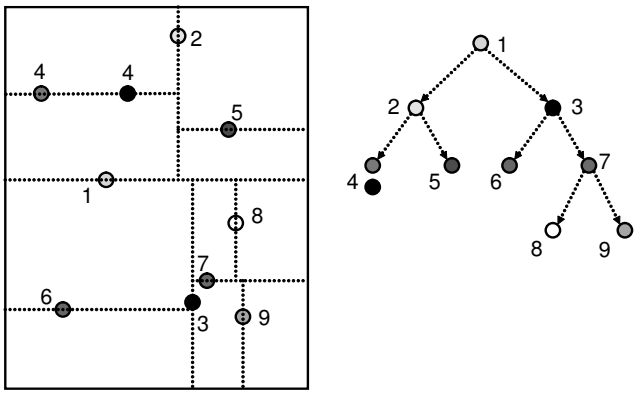
- K-D-Tree
- BSP Tree
- Quad-Tree

Posledně jmenovaný algoritmus leží v základu dalších algoritmů a neřeší problematiku paměti, nicméně je velmi inspirativní.

K-D-Tree

Algoritmus K-D-Tree, nebo též k-D-Tree, či kD-Tree je založen na principu, kdy prostor je dělen hyperplochami (v rovině to znamená přímkami) na nejvyšší možné úrovni vždy na dvě části. Dělicí hyperplocha musí být rovnoběžná s osovým systémem.

Při dělení musí v každé dělicí hyperploše být obsažen uvnitř této plochy nějaký bod, přičemž ten bod nesmí již být obsažen v jiné hyperploše, ale jedna hyperplocha může obsahovat více bodů. Vkládání i vyhledávání je bez problémů, i když nevhodné pořadí vkládání může vést k degradaci stromové struktury. Mazání je však velmi problematické a musí dojít k znovuvložení celého podstromu.



Obrázek 4.1: K-D-Tree

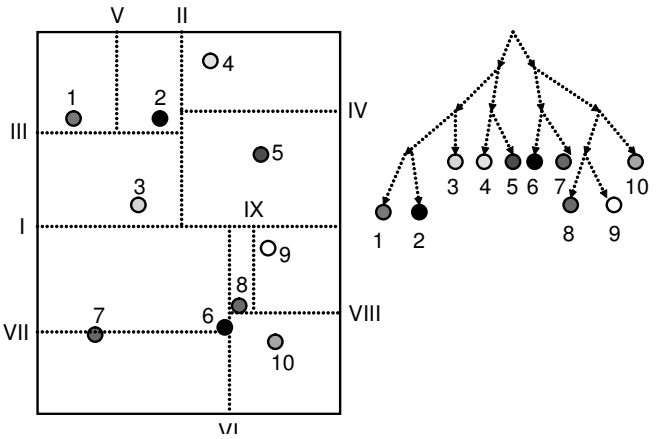
Jak již bylo řečeno, tento algoritmus je vhodný pouze pro body. Na obrázku 4.1 je naznačeno, jak se buduje indexační strom. V levé části ohraničuje obdélník uvažovaný prostor s body. Čísla u bodů označují pořadí, v jakém je prostor dělen hyperplochami (tečkované čáry, vždy jen v daném podprostoru) a vkládány údaje do stromu. Rovnoběžnost s osovým systémem umožňuje eliminovat jednu hodnotu souřadnice při testu v době vkládání a zejména vyhledávání. Algoritmus je tak výhodný pro statická data.

Adaptivní K-D-Tree

Tento algoritmus se snaží eliminovat nevýhodu K-D-Tree v pořadí vkládání hodnot do stromu. Data jsou pak roztroušena po celém stromu. Při dělení hyperplochou se vždy daný podprostor rozdělit tak, aby každá část obsahovala zhruba stejný počet bodů. I když jsou dělicí hyperplochy rovnoběžné s osovým systémem, tak nemusejí obsahovat žádný bod.

Data se tak stěhují do listů — je stanovena hranice, kolik bodů může být v listových datech, např. 1. I tento algoritmus je však vhodný pouze pro statická data.

Na obrázku 4.2 je znázorněna tvorba takové stromové struktury. Římská čísla označují pořadí, v jakém dochází k dělení podprostoru na poloviny vždy příslušnou dělicí hyperplochou. Arabská čísla slouží



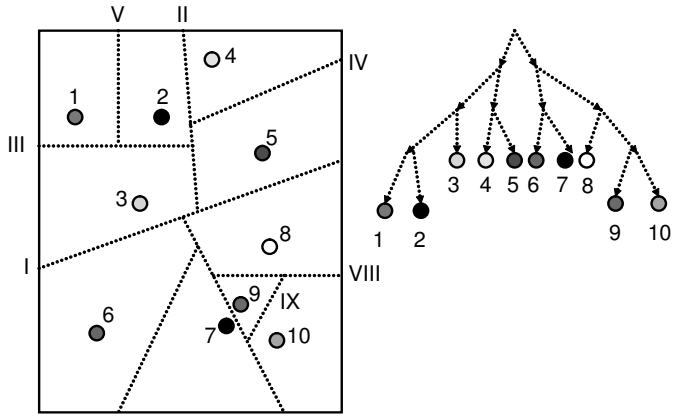
Obrázek 4.2: Adaptivní K-D-Tree

pro identifikaci bodů ve výsledném stromě.

BSP Tree

BSP v případě tohoto algoritmu stojí za *Binary Space Partitioning*. Tento algoritmus je shodný s algoritmem adaptivní K-D-Tree až na jednu drobnost — dělicí hyperplochy nemusejí být rovnoběžné se souřadným systémem.

Dělení taktéž probíhá tak dlouho, dokud počet bodů, které zbývají, neklesne pod určitou hodnotu (pro naše ukázky to je 1, ale jinak to není typický počet). Tento algoritmus však rozhodně není nikterak adaptivní (odolný proti změně dat) a navíc má vyšší nárok na paměť. Nicméně svým konceptem, kdy nedělí podprostor vždy rovnoběžně s nějakou osou, je nový.



Obrázek 4.3: BSP Tree

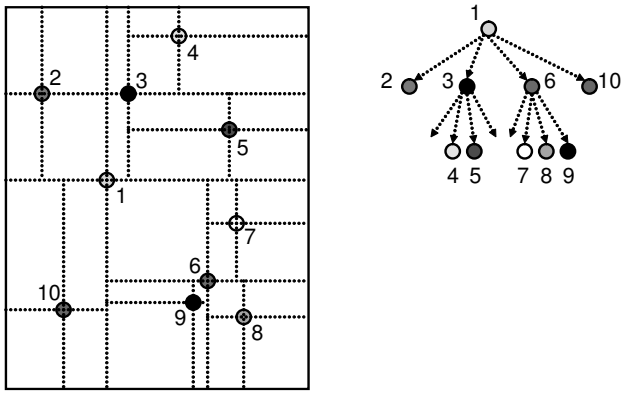
Na obrázku 4.3 je jako v minulém případě znázorněna tvorba ta-

kové stromové struktury. Shodně označují římská čísla taktéž pořadí, v jakém dochází k dělení podprostoru na poloviny vždy příslušnou dělicí hyperplochou. Arabská čísla slouží pro identifikaci bodů ve výsledném stromě.

Quad-Tree
Tento algoritmus se sice úplně shoduje s algoritmem K-D-Tree, avšak jeho novinkou je to, že podprostor neděli na polovinu, ale na 2^n podstromů (n je stupeň dimenze prostoru). Díky tomu nemusejí být podstromy shodné, neboť některé listy nemusejí vést do míst, kde je obsažen nějaký bod.

Dělení tak probíhá tak dlouho dokud počet bodů neklesne pod nějaký počet, který v tomto případě může být tedy i nula.

Algoritmus se vyskytuje ve variantě dělicí podprostor v bodech (point quad-tree) a také dělicí prostor na prakticky shodné části (region quad tree).



Obrázek 4.4: (Point) Quad-Tree

Na obrázku 4.4 je naznačeno, jak se buduje indexační strom. Čísla u bodů označují pořadí, v jakém je prostor dělen hyperplochami na $2^n = 4$ podprostory a jak jsou vkládány údaje do stromu. Povšimněte si, že některé větve nemusejí obsahovat data, i když sousední je obsahují.

Výklad

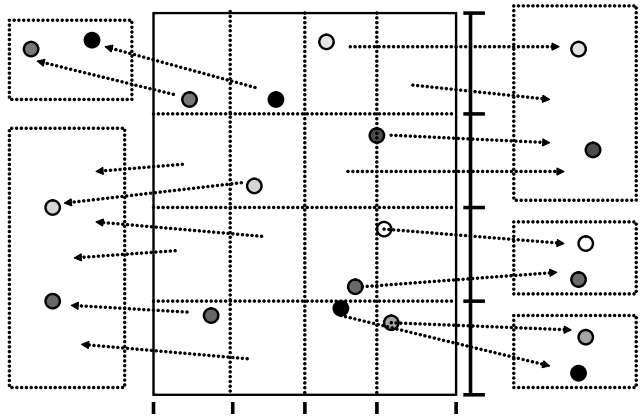
Hashování

V metodách využívající hashování se uplatňují principy jak adaptivního, tak lineárního hashování a dokonce najdeme i hybridní algoritmy, které kombinují hashování a stromy.

Ukážeme si algoritmus Grid File a jeho varianty jako zástupce adaptivního hashování. V závěru této části to bude algoritmus Buddy Tree, který je představitelem hybridních algoritmů.

Grid File
Sledovaný úsek prostoru je pokryt n-rozměrnou mřížkou (nikoliv nutně pravidelnou). Výsledné buňky tak obsahují různý počet bodů — různorodé obsazení. K tomuto základnímu rozdělení je dodán adresář, který každou buňku přiřazuje k datové jednotce (bucket).

Adresář je poměrně velký a je proto vždy ukládán na disku. Mřížka je také uložena na disku, nicméně pro vyhledání datové jednotky stačí pouze 2 přístupy na disk, což je snesitelné. Algoritmus se může pyšnit až 69% využitím prostoru.



Obrázek 4.5: Grid File

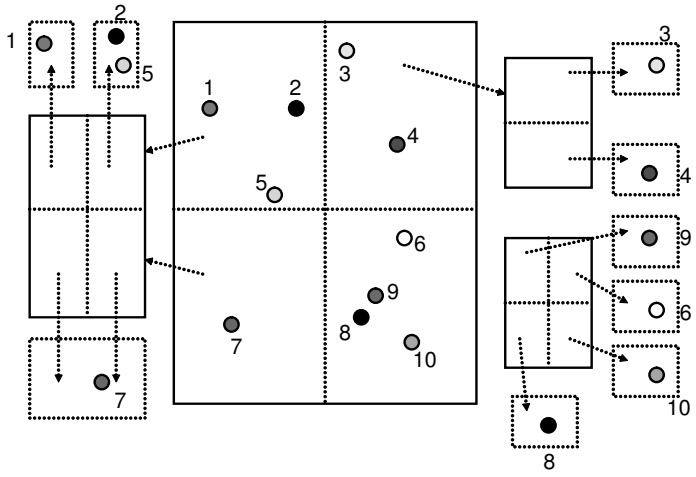
Obrázek 4.5 naznačuje takovou strukturu. prostor je rozdělen mřížkou, šipky naznačují příslušnost buněk (bodů) k datovým jednotkám. Při vkládání může dojít k přetečení (datová jednotka není schopna pojmout další údaj), které není lokální. je nutné vložit rozdělující hyperplochu a tak zvětšit adresář, jak je tento údaj distribuován existující datovou strukturou.

Podobně se chová i mazání. Jelikož není lokální, tak odstranění hyperplochy je třeba prověřit. Pokud je dat málo, tak může zaniknout i datová jednotka (data buďto zanikla s ní, nebo jsou přesunuta do jiné datové jednotky).

Two-Level Grid File
Dvouúrovňový Grid File je algoritmus, který vlastně staví algoritmus Grid File dvakrát za sebe. Mřížka druhé úrovně (ne ta, co je nejvýše)

se používá pro správu adresářů (objevuje se vztah kořenový adresář, podadresář).

V takovém systému jsou změny při vkládání či mazání často lokální, nicméně i tak není problematika přetečení úspěšně řešena.



Obrázek 4.6: Two-Level Grid File

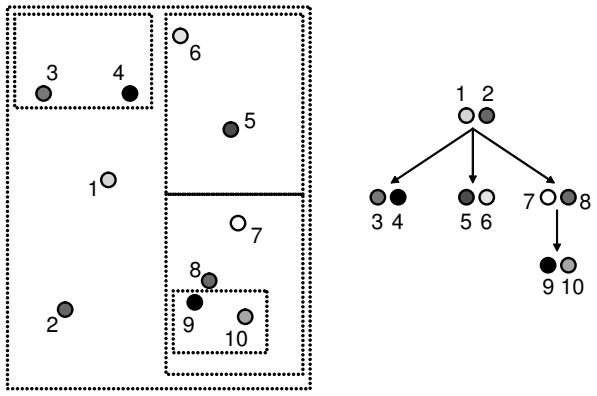
Obrázek 4.6 naznačuje vznik takové struktury. Na nejvyšší úrovni dělí mřížka oblast na poměrně velké celky. Ty jsou potom předmětem dělení autonomní struktury Grid File. Jedná se tedy o dvě úrovně, kde ta první pouze odkazuje na základní strukturu Grid File. (Čísla na obrázku pouze provazují shodné body.)

BANG File

Označení BANG skrývá *Balanced And Nested Grid*. Svým způsobem by tento algoritmus šel zařadit mezi hybridní. Tento algoritmus se snažil odstranit problém s exponenciálním nárůstem adresáře — **typická data nejsou rozložena v prostoru rovnoměrně**(!).

Základem je opět Grid File. Nicméně buňky se mohou překrývat, dokonce vnořovat. Datová jednotka odpovídá buňce. Buňky ani nemusejí mít tvar hyperkrychle. Jednotlivé datové jednotky jsou potom uloženy ve vyváženém stromě.

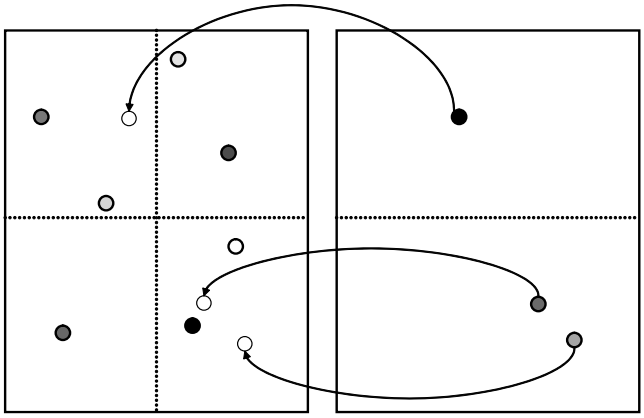
Na obrázku 4.7 je načrtnut vznik kombinované struktury. Čísla užitá u bodů slouží pouze pro jejich jednoznačnou identifikaci. V každé buňce může být jistý maximální a minimální (nenulový) počet bodů. Při vyšším počtu bodů je nutné vložit nové podprostory a vybudovat vyváženou stromovou strukturu. Právě stromová struktura může být největším problémem tohoto algoritmu.



Obrázek 4.7: BANG File

Twin Grid File
Volně přeloženo, jedná se o dvojčata algoritmů Grid File. Opravdu tomu tak je i ve skutečnosti. Celá struktura se vyskytuje dvakrát. Vztah však není hierarchický (vertikální), ale horizontální. I když je jedna ze struktur nadřazená, tak je jedno, která to bude.

Cílem tohoto algoritmu je maximálně využít prostor pro indexovací strukturu. Proto se data mezi obě části dělí prakticky rovnoměrně. Jedna struktura je však primární a druhá je sekundární, přetoková. Takto může využít až 90% prostoru bez výrazného zpomalení.



Obrázek 4.8: Twin Grid File

Obrázek 4.8 naznačuje, jak jsou do přetokové struktury odkládány body tak, aby nedošlo k přetečení ve struktuře primární. Šipky naznačují, kde jsou body umístěny v prostoru. Do sekundární struktury se body odsouvají tehdy, pokud by mělo dojít k rozdělení primární struktury a přitom úroveň dělení struktury sekundární je menší, nebo

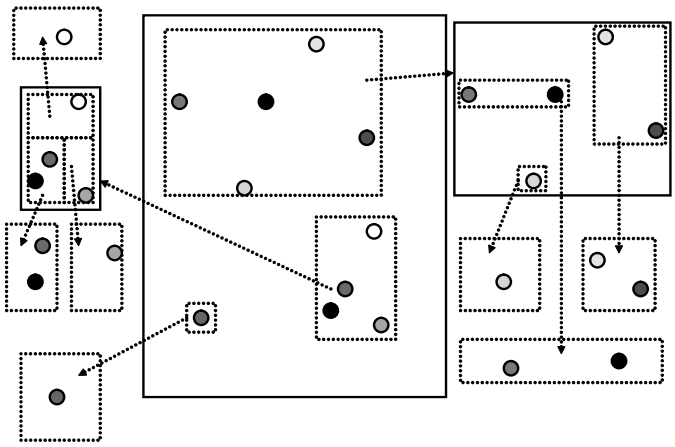
je schopna bod pojmout bez dělení.

Tento algoritmus umožňuje částečnou paralelizaci pro vkládání a plnou pro vyhledávání. Tato vlastnost se dostává do popředí zejména s vícejadrovými procesory (od roku 2004).

Buddy Tree

Jediná plně hybridní struktura. I když je základem hashování, tak adresářem je obecně nevyvážená stromová struktura minimálně se dvěma položkami. Uvnitř stromu jsou ukazatele na nižší úrovně, až v listech jsou ukazatele na datové jednotky.

Strom je rozdělován rekurzivně, dělí se hyperplochami rovnoběžnými s osami. Ve vnitřních uzlech se však prostor omezí na nejmenší obalující obdélník (MBB — minimal bounding box) vnitřních bodů — tím se dosahuje výraznější selektivity.



Obrázek 4.9: Buddy Tree

Na obrázku 4.9 je znázorněno, jak je primární prostor rozdělen pomocí MBB na podprostory a ty jsou uloženy do stromové struktury (naznačeno šipkami). Dělení probíhá tak dlouho, až počet bodů neklesne pod určitou mez. Nemůže však být nulový. Díky začlenění stromu se však nevyhneme problémům, které spolu stromy přinášejí.

Vícerozměrná data

I když uložení bodů je primární, tak s ním jistě nemůžeme, minimálně aplikačně, vystačit. Proto vznikly algoritmy pro indexaci vícerozměrných útvarů. Hlavní metody přístupu lze rozdělit do tří skupin:

1. transformace — mapování objektů na bezrozměrné objekty vícedimenzionálních prostorů;

2. překrývání — indexační struktury se překrývají, takže vzniká více vyhledávacích cest;
3. ořezávání — aby nedocházelo k překrytí, tak se objekty rozdělí, ale tím pádem duplikují.

Transformace
Objekty popsané k body v n-rozměrném prostoru se mapují na bezrozměrné objekty (body) ve vícedimenzionálním prostoru, přesněji v $k*n$-rozměrném prostoru. Obdélník ve 2D se tak stává bodem ve 4D.

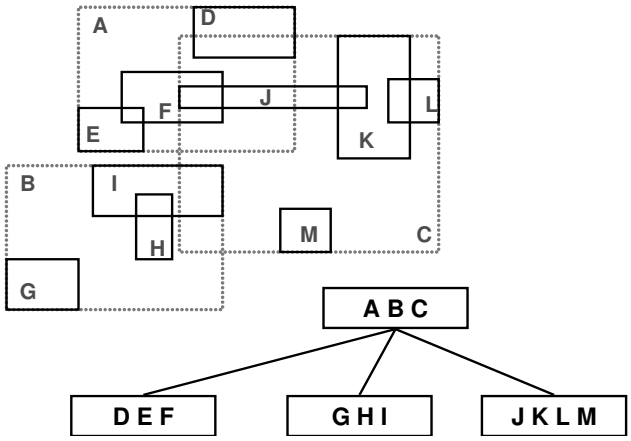
I když tento se přístup nejdříve jevil jako slibný a vlil řadě lidí do krve optimismus, tak záhy přišlo vystřízlivění (podobně jako u mapování vícedimenzionálních bodů do jedné dimenze). Některé dotazy jsou nerealizovatelné. Mapování je často složité, ne-li nemožné. A v neposlední řadě je problém s interpretací výsledku — problém zpětné transformace.

Překrývání
Buňky se překrývají svými hranicemi a jelikož se často (minimálně na nejnižší úrovni) shodují s datovými jednotkami, tak i ty se překrývají (svým dosahem). V praxi tak dochází k tomu, že algoritmus je stejný, jen počet prohledávacích cest se zvětšuje, protože dopředu není jasné, ve které buňce je nakonec objekt uložen. I když by se mohlo zdát, že to je vhodné pro paralelizaci, tak opak je pravdou, neboť počet prohledávaných cest není dopředu znám.

R-Tree
Jedním ze zástupců překrývání je R-Tree, nebo R-strom. Ukládá obdélníkové objekty tak, že vždy jistou skupinu obalí MBB a pokud je počet objektů uvnitř nad jistou mez, tak pokračuje rekurzivně v dělení. Obalující MBB vždy obalují celý objekt, který ukládají. Díky tomu se mohou překrývat a indexované objekty mohou zasahovat do více buněk.

Obrázek 4.10 naznačuje jednoduchý případ. Plnou čarou jsou kresleny indexované objekty, zatímco přerušovanou čarou jsou vykresleny MBB pro buňky užité pro indexaci. Data jsou až v listech. Charakterem má algoritmus stejné vlastnosti jako stromové algoritmy, nevíc přibývá problém s více cestami a selektivitou.

Ořezávání
U metod založených na ořezávání není povoleno, aby se MBB, buňky překrývaly. Abychom neomezili uživatele v možnostech vkládaných objektů, tak jediným řešením je rozsekat objekty tak, aby sledovaly



Obrázek 4.10: R-strom/R-Tree

hranice dělicích MBB. Jeden objekt je tak může být rozdělen na více. Při vyhledávání pak tedy nestačí vyhledat objekt, ale je třeba se vrátit k jeho původní, nedělené reprezentaci.

Při nedostatku místa u datových jednotek je možné je nějak rozšiřovat, nicméně to často vede k zamrznutí (deadlock). Proto se musejí datové jednotky dělit. Zpětné slučování opět není jednoduché.

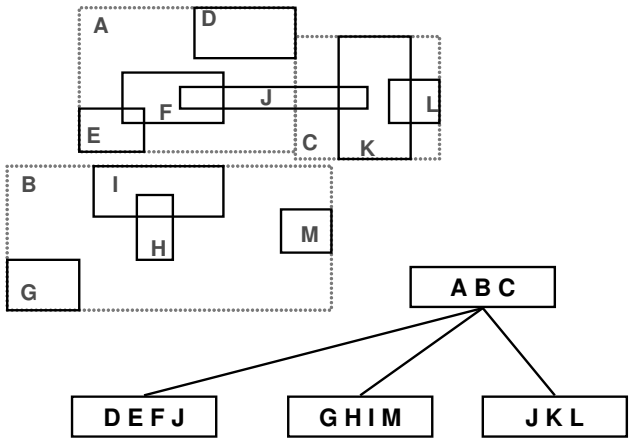
R$^+$-Tree

R$^+$-Tree, nebo R+ strom je zástupce stromových algoritmů pracující metodou ořezávání. Algoritmus je tedy podobný algoritmu R-Tree s tím, že buňky se nemohou překrývat. V případě, že existuje objekt, který zasahuje do více buněk, tak je nutné objekt rozdělit na hranici na dva (více) kusů. Pokud by hranice buněk nebyly těsně u sebe, tak se musí buďto vložit další buňka (je-li to žádoucí a přínosné), nebo se musí buňka rozšířit — zde je však nebezpečí zamrznutí (deadlock), neboť se může blokovat více buněk v rozšiřování.

Obrázek 4.11 demonstruje strukturu R+ stromu na stejných datech jako pro R-Tree. Vidíme, že v několika listech se vyskytuje stejná datová položka. Význam čar je shodný jako pro R-Tree.

Hashování

I pro vícerozměrné objekty byly vyvinuty metody založené na hashování. Za základ si stejně jako metody stromové berou algoritmy vyvinuté pro body. Pro ilustraci uveďme alespoň některé jejich zástupce: PLOP, Multi-Layer Grid File, R-File.



Obrázek 4.11: R+ strom/R^+-Tree

Obojí data

Existují i algoritmy, které dokáží ukládat jak bodové, tak vícerozměrné datové objekty do indexačních struktur. Za všechny jmenujme alespoň P-Tree (autor Schiwietz). Tento algoritmus nepoužívá pro vymezení buněk MBB, ale obecně konvexní obálky. Data ukládá do listů. Pokud pomineme hashování, které zde přirozeně není zastoupeno, tak datová struktura se nejvíce podobá té uplatněné v Buddy Tree.

Největší problémem je, podobně jako u jiných algoritmů, zda rozšířit buňku pro nový objekt, nebo zda zavést buňku novou.

Pojmy k zapamatování

Tato kapitola se především zaměřila na indexační techniky a algoritmy užívané pro indexaci prostorových dat. Je tedy nutné zvládnout pojmy a algoritmy spojené zejména, ne však výhradně s:

- problémem indexace prostorových dat
- stromovými algoritmy pro bodová data
- hashovací algoritmy pro bodová data
- hybridní algoritmy pro bodová data
- algoritmy pro vícerozměrná data
  - obecné postupy řešení
  - případy řešení
  - algoritmy pro body i vícerozměrná data

Závěr

Cílem kapitoly bylo ukázat základní problém s indexací vícedimenzionálních dat a ukázat přístupy pro jejich indexaci jak pro bezrozměrné, tak vícerozměrné údaje. Zvládnutí těchto algoritmů a technik by mělo najít uplatnění nejen při práci s prostorovými databázemi a aplikacemi, ale i v dalších oborech, které pracují s vícedimenzionálními daty. I když popis algoritmů není vyčerpávající z implementačního hlediska, tak by měl dostačovat pro pochopení idee a minimální implementaci, nebo pro detailní nastudování z jiných zdrojů.

Úlohy k procvičení:

1. Jaký je principiální problém indexace vícedimenzionálních dat? Vysvětlete.
2. Jak pracuje algoritmus adaptivní K-D-Tree?
3. Jak pracuje algoritmus Two-Level Grid File?
4. Jaké jsou principy indexace vícerozměrných dat?
5. Jak pracuje algoritmus R+ strom.

Klíč k řešení úloh

Odpovědi na všechny otázky najdete v textu.

Další zdroje

Další informace k této kapitole najdete na internetu či v literatuře zabývající se prostorovými databázemi (uvedena v úvodní kapitole) nebo ve Güting, R.H.: Spatial Database Systems, Tutorial Notes, Fernuniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany, případně v Schneider, M.: Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS, FernUniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany. Pro další práci je možné nastudovat i další algoritmy.

Kapitola 5

Závěr

Publikace představuje v několika kapitolách typické rysy a problematiku implementace prostorových databázových jazyků. Soustřeďuje se na definici klíčových problémů, jejich možná řešení i prezentaci praktického přístupu pro vybrané úlohy spojené s prostorovými databázemi. Cílem bylo zejména uvést do problematiky a nastolit otázky a motivovat k jejich zodpovězení na základě hlubšího studia s využitím dalších zdrojů.

Po absolvování by student měl být schopen definovat problematiku spojenou s prostorovými databázemi, nastínit hlavní výhody užití takových systémů, popsat klíčové momenty, které musejí být splněny, aby takový databázový systém byl plněn funkční i sdělit jaká se uplatňují řešení.

Modul je prvním modulem pro předmět Pokročilé databázové systémy a na něj navazují moduly o objektově relačních a multimediálních databázích a modul o temporálních a deduktivních databázích. Pro zúplnění znalostí je vhodné problematiku těchto navazujících modulů také obsáhnout, aby byl celkově dotvořen obraz postrelačních databází.

Modul obsahuje pouze nejnutnější informace a nástiny problematiky. Pro úplné zvládnutí problematiky je vhodné rozšířit si vědomosti nějakou vhodnou doporučenou literaturou jak z oblasti prostorových databází, tak třeba z okrajových témat (indexační algoritmy, analytická geometrie, atd.).

Literatura

[1] Abadi, M., Cardelli, L.: *A Theory of Objects*, Springer, New York, 1996, ISBN 0-387-94775-2.

[2] Aho, A.V., Hopcroft, J.E., Ullman, J.D.: *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.

[3] Aho, A.V., Sethi, R., Ullman, J.D.: *Compilers: Principles, Techniques, and Tools*, Addison Wesley, Reading MA, 1986.

[4] Aho, A.V., Ullman, J.D.: *The Theory of Parsing, Translation, and Compiling, Volume I: Parsing*, Prentice-Hall, Inc., 1972, ISBN 0-13-914556-7.

[5] Aho, A.V., Ullman, J.D.: *The Theory of Parsing, Translation, and Compiling, Volume II: Compiling*, Prentice-Hall, Inc., 1972, ISBN 0-13-914564-8.

[6] Appel, A.W.: Garbage Collection Can Be Faster Than Stack Allocation, *Information Processing Letters 25 (1987)*, North Holland, pages 275–279.

[7] Appel, A.W.: *Compiling with Continuations*, Cambridge University Press, 1992.

[8] Augustsson, L., Johnsson, T.: *Parallel Graph Reduction with the* $< \nu, G >$*-Machine*, Functional Programming Languages and Computer Architecture 1989, pages 202–213.

[9] Barendregt, H.P., Kennaway, R., Klop, J.W., Sleep, M.R.: *Needed Reduction and Spine Strategies for the Lambda Calculus*, Information and Computation 75(3): 191-231 (1987).

[10] Beneš, M.: *Object-Oriented Model of a Programming Language*, Proceedings of MOSIS'96 Conference, 1996, Krnov, Czech Republic, pp. 33–38, MARQ Ostrava, VSB - TU Ostrava.

[11] Beneš, M.: *Type Systems in Object-Oriented Model*, Proceedings of MOSIS'97 Conference Volume 1, April 28–30, 1997, Hradec nad Moravicí, Czech Republic, pp. 104–109, ISBN 80-85988-16-X.

[12] Beneš, M., Češka, M., Hruška, T.: *Překladače*, Technical University of Brno, 1992.

[13] Beneš, M., Hruška, T.: *Modelling Objects with Changing Roles*, Proceedings of 23rd Conference of ASU, 1997, Stara Lesna, High Tatras, Slovakia, pp. 188–195, MARQ Ostrava, VSZ Informatika s r.o., Kosice.

[14] Beneš, M., Hruška, T.: *Layout of Object in Object-Oriented Database System*, Proceedings of 17th Conference DATASEM 97, 1997, Brno, Czech Republic, pp. 89–96, CS-COMPEX, a.s., ISBN 80-238-1176-2.

[15] 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.

[16] Bertino, E., Catania, B., Zarri, G.P.: *Intelligent Database Systems*, Addison-Wesley, ACM press, 2001, ISBN 0-201-87736-8.

[17] Bieliková, M., Návrat, P.: *Funkcionálne a logické programovanie*, Vydavatelstvo STU, Vazovova 5, Bratislava, 2000.

[18] Brodský, J., Staudek, J., Pokorný, J.: *Operační a databázové systémy*, Technical University of Brno, 1992.

[19] Bruce, K.B.: A Paradigmatic Object-Oriented Programming Language: Design, Static Typing and Semantics, *J. of Functional Programming*, January 1993, Cambridge University Press.

[20] Cattell, G.G.: *The Object Database Standard ODMG-93*, Release 1.1, Morgan Kaufmann Publishers, 1994.

[21] Češka, M., Hruška, T., Motyčková, L.: *Vyčíslitelnost a složitost*, Technical University of Brno, 1992.

[22] Češka, M., Rábová, Z.: *Gramatiky a jazyky*, Technical University of Brno, 1988.

[23] Damas, L., Milner, R.: *Principal Type Schemes for Functional Programs*, Ninth Annual ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 1982, pages 207–212.

[24] Dassow, J., Paun, G.: *Regulated Rewriting in Formal Language Theory*, Springer, New York, 1989.

[25] Douence, R., Fradet, P.: *A taxonomy of functional language implementations. Part II: Call-by-Name, Call-by-Need and Graph Reduction*, INRIA, technical report No 3050, Nov. 1996.

[26] Ellis, M.A., Stroustrup, B.: *The Annotated C++ Reference Manual*, AT&T Bell Laboratories, 1990, ISBN 0-201-51459-1.

[27] Finne, S., Burn, G.: *Assessing the Evaluation Transformer Model of Reduction on the Spineless G-Machine*, Functional Programming Languages and Computer Architecture 1993, pages 331-339.

[28] Fradet, P.: *Compilation of Head and Strong Reduction*, In Porc. of the 5th European Symposium on Programming, LNCS, vol. 788, pp. 211-224. Springer-Verlag, Edinburg, UK, April 1994.

[29] Gaede, V., Günther, O.: *Multidimensional Access Methods*, Institut für Wirtschaftsinformatik, Humboldt-Universität zu Berlin, Germany.

[30] Georgeff M.: *Transformations and Reduction Strategies for Typed Lambda Expressions*, ACM Transactions on Programming Languages and Systems, Vol. 6, No. 4, October 1984, pages 603–631.

[31] Gordon, M.J.C.: *Programming Language Theory and its Implementation*, Prentice Hall, 1988, ISBN 0-13-730417-X, ISBN 0-13-730409-9 Pbk.

[32] Gray, P.M.D., Kulkarni, K.G., Paton, N.W.: *Object-Oriented Databases*, Prentice Hall, 1992.

[33] Greibach, S., Hopcroft, J.: Scattered Context Grammars, *Journal of Computer and System Sciences*, Vol: 3, pp. 233–247, Academia Press, Inc., 1969.

[34] Güting, R.H.: *Spatial Database Systems*, Tutorial Notes, Fernuniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany.

[35] Harrison, M.: *Introduction to Formal Language Theory*, Addison Wesley, Reading, 1978.

[36] Hruška, T., Beneš, M.: *Jazyk pro popis údajů objektově orientovaného databázového modelu*, In: Sborník konference Některé nové přístupy při tvorbě informačních systémů, ÚIVT FEI VUT Brno 1995, pp. 28–32.

[37] Hruška, T., Beneš, M., Máčel, M.: *Database System G2*, In: Proceeding of COFAX Conference of Database Systems, House of Technology Bratislava 1995, pp. 13–19.

[38] Issarny, V.: *Configuration-Based Programming Systems*, In: Proc. of SOFSEM'97: Theory and Practise of Informatics, Milovy, Czech Republic, Novemeber 22-29, 1997, ISBN 0302-9743, pp. 183-200.

[39] Jensen, K.: *Coloured Petri Nets*, Springer-Verlag Berlin Heidelberg, 1992.

[40] Jeuring, J., Meijer, E.: *Advanced Functional Programming*, Springer-Verlag, 1995.

[41] Jones, M.P.: *A system of constructor classes: overloading and implicit higher-order polymorphism*, In FPCA '93: Conference on Functional Programming Languages and Computer Architecture, Copenhagen, Denmark, June 1993.

[42] Jones, M.P.: *Dictionary-free Overloading by Partial Evaluation*, ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation, Orlando, Florida, June 1994.

[43] Jones, M.P.: *Functional Programming with Overloading and Higher-Order Polymorphism*, First International Spring School on Advanced Functional Programming Techniques, Båstad, Sweden, Springer-Verlag Lecture Notes in Computer Science 925, May 1995.

[44] Jones, M.P.: *GOFER, Functional programming environment, Version 2.20*, mpj@prg.ox.ac.uk, 1991.

[45] Jones, M.P.: *ML typing, explicit polymorphism and qualified types*, In TACS '94: Conference on theoretical aspects of computer software, Sendai, Japan, Springer-Verlag Lecture Notes in Computer Science, 789, April, 1994.

[46] Jones, M.P.: *A theory of qualified types*, In proc. of ESOP'92, 4th European Symposium on Programming, Rennes, France, February 1992, Springer-Verlag, pp. 287-306.

[47] Jones, S.L.P.: *The Implementation of Functional Programming Languages*, Prentice-Hall, 1987.

[48] Jones, S.L.P., Lester, D.: *Implementing Functional Languages.*, Prentice-Hall, 1992.

[49] Kim, W. (edt.): *Modern Database Systems*, ACM Press, 1995, ISBN 0-201-59098-0.

[50] Kleijn, H.C.M., Rozenberg, G.: On the Generative Power of Regular Pattern Grammars, *Acta Informatica*, Vol. 20, pp. 391–411, 1983.

[51] Khoshafian, S., Abnous, R.: *Object Orientation. Concepts, Languages, Databases, User Interfaces*, John Wiley & Sons, 1990, ISBN 0-471-51802-6.

[52] Kolář, D.: *Compilation of Functional Languages To Efficient Sequential Code*, Diploma Thesis, TU Brno, 1994.

[53] Kolář, D.: *Overloading in Object-Oriented Data Models*, Proceedings of MOSIS'97 Conference Volume 1, April 28–30, 1997, Hradec nad Moravicí, Czech Republic, pp. 86–91, ISBN 80-85988-16-X.

[54] Kolář, D.: *Functional Technology for Object-Oriented Modeling and Databases*, PhD Thesis, TU Brno, 1998.

[55] Kolář, D.: *Simulation of LL_k Parsers with Wide Context by Automaton with One-Symbol Reading Head*, Proceedings of 38th International Conference MOSIS '04—Modelling and Simulation of Systems, April 19–21, 2004, Rožnov pod Radhoštěm, Czech Republic, pp. 347–354, ISBN 80-85988-98-4.

[56] Latteux, M., Leguy, B., Ratoandromanana, B.: The family of one-counter languages is closed under quotient, *Acta Informatica*, 22 (1985), 579–588.

[57] Leroy, X.: *The Objective Caml system, documentation and user's guide*, 1997, Institut National de Recherche en Informatique et Automatique, Francie, Release 1.05, http://pauillac.inria.fr/ocaml/htmlman/.

[58] Martin, J.C.: *Introduction To Languages and The Theory of Computation*, McGraw-Hill, Inc., USA, 1991, ISBN 0-07-040659-6.

[59] Meduna, A.: *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[60] Meduna, A., Kolář, D.: Regulated Pushdown Automata, *Acta Cybernetica*, Vol. 14, pp. 653–664, 2000.

[61] Meduna, A., Kolář, D.: One-Turn Regulated Pushdown Automata and Their Reduction, In: *Fundamenta Informaticae*, 2002, Vol. 16, Amsterdam, NL, pp. 399–405, ISSN 0169-2968.

[62] Mens, T., Mens, K., Steyaert, P.: *OPUS: a Formal Approach to Object-Orientation*, Published in FME '94 Proceedingss, LNCS 873, Springer-Verlag, 1994, pp. 326-345.

[63] Mens, T., Mens, K., Steyaert, P.: *OPUS: a Calculus for Modelling Object-Oriented Concepts*, Published in OOIS '94 Proceedings, Springer-Verlag, 1994, pp. 152-165.

[64] Milner, R.: A Theory of Type Polymorphism In Programming, *Journal of Computer and System Sciences*, 17, 3, 1978.

[65] Mycroft, A.: *Abstract Interpretation and Optimising Transformations for Applicative Programs*, PhD Thesis, Department of computer Science, University of Edinburgh, Scotland, 1981. 180 pages. Also report CST-15-81.

[66] Nilsson, U., Maluszynski, J.: *Logic, Programming and Prolog (2ed)*, John Wiley & Sons Ltd., 1995.

[67] Odersky, M., Wadler, P.: *Pizza into Java: Translating theory into practice*, Proc. 24th ACM Symposium on Principles of Programming Languages, January 1997.

[68] Odersky, M., Wadler, P., Wehr, M.: *A Second Look at Overloading*, Proc. of FPCA'95 Conf. on Functional Programming Languages and Computer Architecture, 1995.

[69] Okawa, S., Hirose, S.: Homomorphic characterizations of recursively enumerable languages with very small language classes, *Theor. Computer Sci.*, 250, 1 (2001), 55–69.

[70] Păun, Gh., Rozenberg, G., Salomaa, A.: *DNA Computing*, Springer-Verlag, Berlin, 1998.

[71] Reisig, W.: *A Primer in Petri Net Design*, Springer-Verlag Berlin Heidelberg, 1992.

[72] Robinson, J. A.: A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12, 23–41, 1965.

[73] Rozenberg, G., Salomaa, A. — eds.: *Handbook of Formal Languages; Volumes 1 through 3*, Springer, Berlin/Heidelberg, 1997.

[74] Salomaa, A.: *Formal Languages*, Academic Press, New York, 1973.

[75] Schmidt, D.A.: *The Structure of Typed Programming Languages*, MIT Press, 1994.

[76] Schneider, M.: *Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS*, FernUniversität Hagen, Praktische Informatik IV, D-58084 Hagen, Germany.

[77] Tofte, M., Talpin, J.-P.: *Implementation of the Typed Call-by-Value λ-calculus using a Stack of Regions*, POPL '94: 21st ACM Symposium on Principles of Programming Languages, January 17–21, 1994, Portland, OR USA, pages 188–201.

[78] Traub, K.R.: *Implementation of Non-Strict Functional Programming Languages*, Pitman, 1991.

[79] Volpano, D.M., Smith, G.S.: *On the Complexity of ML Typability with Overloading*, Proc. of FPCA'91 Conf. on Functional Programming Languages and Computer Architecture, 1991.

[80] Wikström, Å.: *Functional Programming Using Standard ML*, Prentice Hall, 1987.

[81] Williams, M.H., Paton, N.W.: *From OO Through Deduction to Active Databases - ROCK, ROLL & RAP*, In: Proc. of SOFSEM'97: Theory and Practise of Informatics, Milovy, Czech Republic, Novemeber 22-29, 1997, ISBN 0302-9743, pp. 313-330.

[82] Lindholm, T., Yellin, F.: *The Java Virtual Machine Specification*, Addison-Wesley, 1996, ISBN 0-201-63452-X.