1 Postrelační databázový systém

Postrelační databázový systém je relační databázový systém rozšířený o nějakou specializaci na databázové úrovni, jelikož aplikační řešení by bylo nedostačující.

Příklady postrelačních DB systémů:

- *Prostorové databáze* – rozšířeny o práci s prostorovými objekty a vztahy mezi nimi
- *Objektově orientované databáze* – rozšířeny o objektový model dat a vazby
- *Deduktivní databáze* – rozšířeny o funkce pro analýzu dat
- *Temporální databáze* – rozšířeny o temporální logiku
- *Multimediální databáze* – rozšířeny o funkce pro práci s multimediálním obsahem
- *Aktivní databáze* – rozšířeny o aktivní pravidla

V současnosti všechny používané databázové systémy jsou postrelační, jelikož obsahují nějaká rozšíření oproti původnímu relačnímu schématu (např. triggery).

2 Objektově relační databázové systémy

Objektově relační DB systémy nejsou plně objektové (nemodelují objekty a vztahy mezi nimi přímo), ale jsou v nejnižší vrstvě relační a objekty nad nimi jsou simulovány pomocí SŘBD. Čistě objektové systémy nebyly standardizovány, neměly standardní jazyk a nerozšířily se.

Přínosy:

- Standardizace (SQL 1999) a podpora výrobců relačních systémů
- Uživatelské datové typy
- Data reprezentována objekty (které mají OO vlastnosti)
- Jednoznačná identifikace objektu – OID
- OID umožňuje tvorbu trvalých vztahů a zjednodušuje indexaci
- Vztahy (1:1, 1:N, N:N) jsou součástí definice dat a ne v algoritmu
- Vnořené struktury (např. adresa) a tabulky
- Kolekce (řetězec, posloupnost, seznam, soubor) a kurzor

SQL 1999 zavedlo podporu pro objektově orientované databáze:

- *Uživatelské datové typy* – definice struktur a jednoduchá dědičnost, vzácné typy (vycházejí z vestavěných)
- *Typované tabulky* – tabulka je typu UDT (tedy je to třída a řádky jsou objekty)
- *Nové datové typy* – `LOB (BLOB, CLOB)`, `BOOLEAN`, `ARRAY`, `ROW` (složený sloupec)
- *Regulární výrazy* – `WHERE x SIMILAR TO regexp`
- *Databázové triggery* – podpora aktivních prvků databáze
- *OID* – objekty lze reprezentovat jako řádky tabulek (tříd) a mít reference typu `REF`
- Přístup k hodnotám struktur přes operátor `.`

3 Prostorové databáze a reprezentace dat

Prostorové databáze jsou databázové systémy schopné zpracovávat prostorová data v podobě jednoduchých geometrických entit (obrovského množství).

Co nejsou spatDB: GIS (ten spatDB pouze může použít), obrázky prostorů (to jsou multimediální)

Entity spatDB jsou relativně jednoduché objekty s danou identifikací, umístěním a vztahem k okolí.

spatDB modelují body, (lomené) úsečky, polygony, oblasti; např. města, řeky, domy, lesy; a to v *euklidovském prostoru* který je spojitý.

Problém reprezentace souřadnic: spojité souřadnice nejsou k dispozici, chyba při diskretizaci. Problém pak vyvstane při výpočtu průsečíku nebo sousednosti, jelikož všechny body úsečky se pohnou při aproximaci pozice koncového bodu. Řešením je všechny průsečíky (a doteky) vypočítat jen jednou – pomocí simplexů nebo úplných popisů.

Simplex je nejmenší nevyplněný objekt dané dimenze (d-simplex): 0-simplex (bod), 1-simplex (úsečka), 2-simplex (trojúhelník), 3-simplex (čtyřstěn)

d-simplex je tvořen $d+1$ simplexy o rozměru $d-1$.

Úplný popis (deskriptor, realm) je souhrnný popis všech objektů databáze, tedy množina bodů, úseček a vyšších objektů, kde:

- Každý samostatný bod je bodem sítě
- Každý koncový bod úsečky je bodem sítě
- Žádný vnitřní bod úsečky není zaznamenán
- Žádně dvě úsečky nemají průsečík ani se nepřekrývají
- Složitější útvary se skládají z úseček

Opět je zde problém s diskretizací, ne vždy je reprezentace dostatečná.

Základní typy: bod, lomená úsečka (line), region (polygon nebo area)

Základní operace: rovno, nerovno, sousedí s, je uvnitř; průnik, překrytí, voronoi; konvexní obálka, vzdálenost, plocha

Relační operace: projekce (redukce atributů), selekce (redukce objektů), fůze (spojení podle atributu), ořezání, windowing (ořezání se zachováním objektů pod výřezem)

Algebra ROSE vychází z deskriptorů a definuje základní chápání polohy dvou oblastí navzájem:

- plošně vnořená (uvnitř)
- hranově vnořená
- vrcholově vnořená
- plošně disjunktní
- hranově disjunktní
- vrcholově (zcela) disjunktní

Dále definuje R-cyklus a R-plochu a formálně definuje plošné obsažení.

Operace a operátory: pro skládané oblasti bylo nejprve vytvořeno 16 kombinací (dle překryvu ploch a hranic), posléze 52 možných. Lze zjednodušit na 5 operací (dotek, uvnitř, přes, přesah, disjunkce) a 3 operátory na extrakci hranice.

Optimalizace dotazů se provádí např. předpočítáním hodnot (plocha, střed apod.).

4 Prostorové databáze a indexace bodů

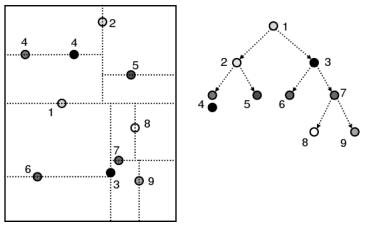
Indexace prostorových dat je možná buď starými metodami (ale je nutné namapovat na jeden rozměr) nebo novými způsoby (musejí se vytvořit).

Architektura indexace je téměř stejná – hash tabulky (lineární, adaptivní) nebo stromy (i když jejich implementace je odlišná).

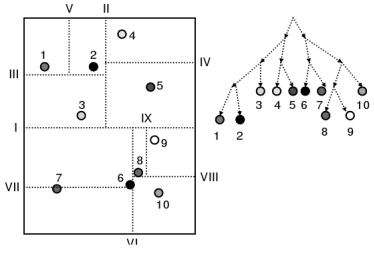
Následník/předchůdce je v 1D jednoduše zjistitelný, ve více rozměrech nelze jednoznačně určit, *mapováním* některé možnosti ztratíme.

Stromy dělící prostor indexují jednotlivé body, ze kterých se objekty skládají – K-D Tree, BSP Tree, Quad-Tree (základ pro další).

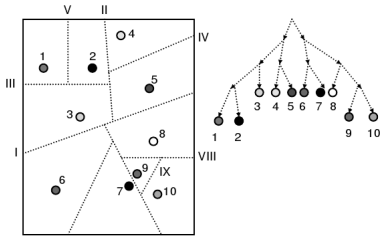
K-D Tree dělí prostor hyperplochami (přímka, plocha, ...), které jsou *rovnoběžné s osami* a musejí *obsahovat bod* (nebo více), který není již obsažen v jiné. K-D Tree je vhodný pro statická data, rychle vyhledává a přidává, pomalu odstraňuje, špatným přidáváním degraduje.



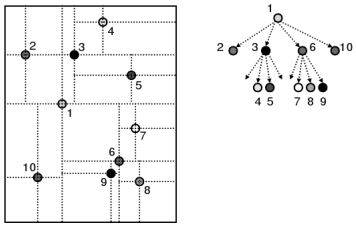
Adaptivní K-D Tree dělí prostor hyperplochami rovnoběžnými s osami tak, aby v obou polovinách byl stejný počet bodů, přičemž body neprochází. Data jsou v listech (mohou se dělit pokud jsou moc plné), vlastnosti lepší pro přidávání (nedegraduje).



BSP Tree (Binary Space Partitioning Tree) pracuje stejně jako Ad-K-D Tree, ale hyperplochy nejsou rovnoběžné s osami a dělení se provádí tak dlouho, dokud počet bodů v podprostorech neklesne pod limit (nemusí být 1 jako pro K-D).

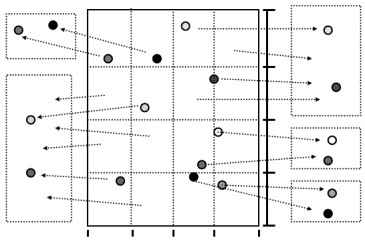


Quad-Tree odpovídá K-D Tree, ale nedělí na dva podprostory, kdežto na 2^n, kde n je dimenze prostoru. Dělí se tak dlouho, dokud počet bodů neklesne pod hranici (může být i 0). Existuje varianta dělící v bodech (point QT) a na stejné části (region QT).

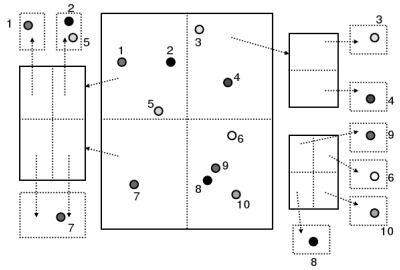


Hashování mapuje klíč na malé číslo, podle kterého se vyhledá záznam.

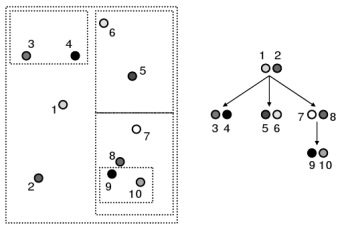
Grid File je adaptivní hashování, které dělí prostor mřížkou (nemusí být pravidelná) a buňky přiřazuje datovým oblastem (bucket). Při mazání/vkládání se musí ověřit přetečení/vyprázdnění bucketu a propagovat změny.



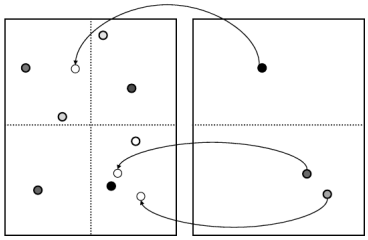
Two-Level Grid File je Grid-File, jehož buckets jsou také Grid-File.



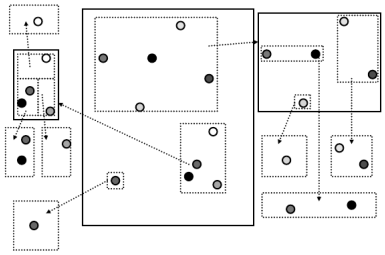
BANG File (Balanced And Nested Grid File) zlepšuje vlastnosti při nerovnoměrném rozložení. Buňky mřížky se překrývají a data jsou uložena ve stromu (buňka má minimum a maximum), problémy se stavbou a úpravou stromu.



Twin Grid File používá sekundární Grid File pro řešení přetečení dat v buňce, nedělí se vždy a prohazuje se primární a sekundární při potřebě. Dobře paralelizovatelné řešení.



Buddy Tree je hybridní řešení, využívá strom obsahující bounding boxy a dělí až do určitého počtu bodů v boxu.



5 Prostorové databáze a indexace vícerozměrných objektů

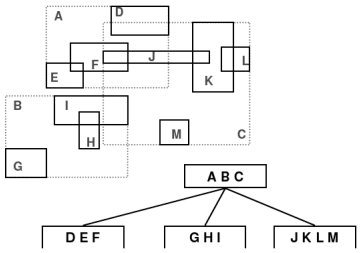
Indexace bodů je sice převážná část indexování, ale nestačí pro aplikační nasazení.

Typy indexace objektů: transformace, překrývání, ořezávání

Transformace mapuje objekty popsané k body v nD prostoru na body v $k \cdot n$D prostoru. Některé dotazy nejsou realizovatelné, mapování složité, někdy nemožné, problém zpětné transformace výsledku.

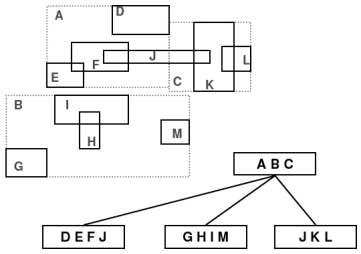
Překrývání: buňky se překrývají a datové jednotky (buckets) většinou také, vzniká více možných cest prohledání, ale neví se kolik (problém při paralelizaci).

R-Tree je zástupce metody překrývání. Obaluje objekty bounding boxy (pokud obsahují moc bodů, rekurzivně dělí). Bounding boxy obalují všechny body daného objektu, takže může docházet k překrývání v rámci několika bodů z objektu.



Ořezávání: buňky se nesmějí překrývat, jediné řešení je pak rozsekat objekty na části podle hranic dotýkajících se bounding boxů. Zpětné shlukování je problém.

R$^+$-Tree je zástupce metody ořezávání. Je podobný R-Tree, ale namísto překrytí dělí objekty na části. Pokud objekt zasahuje do více buněk, které nesousedí, je třeba vložit buňku pro střední část nebo rozšířit jednu z buněk (může nastat problém).



Hashování je používáno i pro vícerozměrná data – PLOP, Multi-Layer Grid File, R-File.

Obojí data (body a vícerozměrná) lze také indexovat, např. P-Tree (konvexní obálky).

6 Temporální databáze

Temporální databáze jsou databáze rozšířené o časovou dimenzi. Používají se v bankovnictví, účetnictví a podobně. Mají dobré vlastnosti pro archivaci a monitorování změn.

Zaznamenávání je diskrétní, čas je spojitý. Lze zaznamenat čas transakce, čas platnosti nebo definovaný čas. Formalizace jako lineárně uspořádaná množina.

Datový typ času (nejčastější) definuje okamžik jako bod na reálné ose a časový úsek jako interval mezi dvěma body. Interval je orientovaný (dopředný a zpětný).

Čas platnosti udává, po který interval jsou data platná (pravdivá v modelu).

Čas transakce udává, kdy jsou data přítomna v DB.

Abstraktní temporální databáze: Byl sestaven formalismus na vyšší úrovni, přenositelnost, převody modelů apod. nejsou problém. Navázány na relační tabulky např. pomocí časů platnosti nebo snímků (snapshots).

Typy tabulek: snímkové, transakce, platného času, obojího času

Snímková tabulka (snapshot) zaznamenává stav dat v jistém okamžiku. Relace uspořádání nad těmito snímky tvoří tok času. Historie databáze je také snímkový model. Problém, pokud se dotazuje na "všechny okamžiky, kdy platilo, že . . . ".

Tabulka s platnými časy obsahuje kromě datových sloupců ještě sloupec pro platnost od/do (možnost udat nekonečno). Pak databáze obsahuje jeden záznam na každou změnu (`Praha 1990 -` se změní na `Praha 1990 1995` a přidá se `Brno 1995 -`).

Tabulka obojího času rozšiřuje tabulku s platnými časy. Nemění údaje od/do, ale přidá nový záznam s opravenými hodnotami. Navíc u každého záznamu obsahuje čas provedení transakce a čas zneplatnění transakce (`Praha 1990 - 1990 -` se změní na `Praha 1990 - 1990 1995` přidá se opravený záznam `Praha 1990 1995 1995 -` a pak se přidá záznam nový `Brno 1995 - 1995 -`).

Transakční tabulka obsahuje pouze časy transakcí (data vypadají stejně jako pro tabulku platného času, ale rozdíl je v tom, že se nejedná o dobu, kdy data platila, ale kdy se vložila do DB).

Dotazovací jazyky nejsou standardizovány. Formálním zápisem je *relační kalkul* rozšířený o časové spojky (until, since, previous, next) a časové proměnné a kvantifikátory, založený na formální logice.

Používané jazyky: TQUEL, TSQL2, vícerozměrné jazyky

Indexování je problém, chceme-li například najít intervaly, které mají neprázdný průnik s jiným. Zvláštnost – většinou se do DB jen zapisuje. Využití R-Tree (viz prostorové DB), AP-Tree, Time Index (na začátku a konci intervalu uloží seznam dalších intervalů, které obsahují tento okamžik).

Shlukování: Někdy je potřeba spojit intervaly kvůli dalšímu zpracování (například při selekci sloupců). Intervaly, které se překrývají (nebo na sebe navazují) a nesoucí stejná data je možné spojit (např. každoroční výkazy při omezení pouze na sloupec adresa).

Problémy: dělení intervalu (jaký bod vybrat), dotazy jsou optimalizovány odhadem, paralelizace není jednoduchá. Problém integrity, když se historie nemůže měnit. Databáze jsou velmi prostorově náročné.

TSQL2:

- Lineární časový model oboustranně omezený.
- Diskrétní časová osa, nejmenší jednotka je chronon (tik hodin).
- Možné nastavit různou granularitu času (a přecházet).
- Typy `DATE, TIME, DATETIME, INTERVAL, PERIOD` (rozdíl).
- Systém obojího času (platnost a transakce), podporuje všechny typy tabulek.
- Časová neurčitost (souvisí se zrnitostí) – asi na konci září.
- Odsávání dat (kvůli velikosti, ale zachovat pro archivaci).

7 Deduktivní databáze

Deduktivní databáze poskytují matematickou logiku (vztahy) spolu s relačními databázemi (znalosti) a využívají znalostí a vztahů k dedukci závěrů.

Na rozdíl od vztahů v aplikaci, jsou zde pravidla také data v DB, lze je pak jednoduše definovat a měnit.

Data jsou trojího typu: fakta, pravidla, odvozená data.

Prolog je inspirací pro DDB, pouze se zpracovává mnohem více dat.

Jazyk jsou množiny dobře definovaných formulí (wffs). Formule jsou predikátové prvního řádu.

Odlišné systémy se neliší v jazyce, ale ve způsobu dedukce (konstrukce důkazu a platnosti formule).

Dokazování se provádí buďto sémanticky nebo syntakticky (inferenční a odvozovací pravidla) – zespoda nahoru nebo shora dolů.

Sémantický důkaz: dosazení konkrétních hodnot do pravidel ($clovek(x) \rightarrow smrtelny(x) \wedge clovek(Jan)$, dedukuje $smrtelny(Jan)$).

Dotazy mohou být otevřené (výsledek je ano/ne) nebo uzavřené (výsledek je množina n-tic).

Predikáty: explicitní (fakta) – $plat(Pepa, 10000)$; implicitní (odvozené) – $vek(x, y), y > 35 : \ plat(x, 20000)$.

Negace na pravé straně formule umožňuje existenci více ekvivalentních formulí (bez negace není možné vytvořit jinou ekvivalentní formuli).

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

Rekurze používá stejné proměnné na pravé i levé straně výrazu. Vyhodnocování je pak mnohem obtížnější.

Model je ohodnocení všech proměnných tak, aby po dosazení do formulí byl výsledek pravdivý.

Bezpečná pravidla mají všechny proměnné omezované, tedy obsahují na pravé straně proměnné z levé strany, proměnné, které jsou porovnány s konstantou nebo porovnány s omezovanou proměnnou.

Pravidla s více pravými stranami se vypočtou paralelně a výsledky se sjednotí.

Operace nad daty jsou typické, DDB přinášejí jen matematický model pro dotazování.

8 Multimediální databáze

Multimediální databáze vznikly v poslední době kvůli potřebě práce s multimediálními daty, které vyžadují speciální funkce pro indexaci, dotazování, extrakci dat a prezentaci. Systém také musí poskytovat možnost heterogenních dotazů (např. najdi všechny adresy osob na této fotografii).

Vyhledávání v obrazových DB:

1. Textový popis (klíčová slova) – popisuje sémantiku, jednodušší zadání, hodně specifické, potřeba rozsáhlé databáze, mylná interpretace slova
2. Podobnost – hodnoty si odvozuje systém, potřeba náčrtku, pracuje pouze s vizuální stránkou a není standardizován postup
   - Metrická metoda – pomocí podobnostní funkce (např. vektor rysů), efektivní, příliš objektivní
   - Transformační metoda – pomocí ceny transformace vzoru na obraz, složitá, subjektivnější

Metrické vyhledávání je výhodné pro výpočty a také dobře indexovatelné, potřebná funkční data jsou vypočítána při vložení dat. Je důležité vybrat správné charakteristiky (můžou se měnit podle typu aplikace) a postup nalezení podobnosti.

Vizuální rysy:

- Nízká úroveň popisu:
  - Obecné – barva, textura, tvar
  - Doménově specifické – markanty otisku prstu, rysy obličeje, ...
- Střední úroveň popisu – statistická data
- Vysoká úroveň popisu – sémantické vyjádření obsahu

Barevné vlastnosti se ukládají ve formě histogramu, podobnost se hledá pomocí průniku. Základní histogram příliš izoluje body a barvy, kumulativní histogram je lepší. Statistické hodnoty pro barvy jsou např. střední hodnota barvy, rozptyl apod.

Texturní vlastnosti se zabývají primitivy textury (části, které se v ní opakují) a její strukturou (hrubost, zrnitost, pravidelnost, ...). Využívají statistických metod (rozložení barev v obraze), zpracování obrazu (filtry hran apod.), geometrické detekce.

Tvarové vlastnosti detekují hrany a oblasti, sestaví seznam takových objektů.

Globální rysy většinou nevedou k výsledku, je potřeba oblast rozdělit a studovat rysy lokální. Rozdělení lze provádět pevně předem a nebo pomocí *segmentace* dynamicky.

Indexace pro podobnostní vyhledávání využívá n-dimenzionální data. Obraz je reprezentován vektorem o n složkách, jedná se tedy o indexaci vícerozměrných dat. Používají se stromové struktury, kde uzly reprezentují jednu oblast obrázku.

- *K-D Tree* – uzly obsahují informaci o hranici oblasti, horší mazání, max 4D
- *Point Quad-tree* – jelikož dělí na 4 části, není potřeba nést informaci o oblasti, jinak stejné problémy
- *R-Tree* – používá nejmenší obálky (obdélník), která udává oblast, problém překrývání, max 5D
- R^+*-Tree* – nemá překrývání, ale dělí oblasti na více místo toho
- R^**-Tree* – má překrytí, ale snaží se ho adaptivně minimalizovat, lepší než R-Tree
- *M-Tree* – používá obalové kružnice (hyperkoule), některé vzdálenosti předpočítány