# Kapitola 17

# Implementace databázových systémů

## 17.1 Seznam otázek

Metody indexace relací, hashování, B-stromy, datové struktury na externí paměti. Vícerozměrné dotazy implementované pomocí hashovacích metod, vícerozměrné mřížky, vícerozměrných stromů. Přístupové metody k prostorovým objektům: R-stromy a jejich varianty. Databáze textů: modely (boolský, vektorový), vyhledávání v textech, signatury, metody implementace signatur (vrstvené kódování), uspořádání odpovědi. Komprese dat: predikce a modelování, reprezentace celých čísel, obecné metody komprese, komprese bitových map, řídkých matic, trie, textů. Huffmanovo kódování (statické, dynamické), aritmetické kódování, LZ algoritmy. Uzamykací protokoly, časová razítka. Distribuované transakce.

**Vsechnu latku pokryva text "Zaklady implementace souboru a databazi" od Pokorneho a Zemlicky a slajdy "Dokumentografickych informacnich systemu" od Kopeckeho**

## 17.2 Metody indexace relací

- relacie mozu byt chapane ako subory, zaznamy suboru ako n-tice relacie, samozrejme su tam rozdiely, napr.:
  - schemy vsetkych relacii sa natiahnu naraz s celou DB, subory musime otvarat/zatvarat jednotlivo
- indexsekvenční soubor (indexováno jen podle primárního klíče)
- invertovaný soubor — lze indexovat podle čehokoliv — indexují se přímo záznamy
- bitová mapa
  - pro atributy malých domén (pohlaví, typ karoserie vozu...) se pro každou doménu vytvoří bitový vektor a jednička tam kde hodnota pro daný řádek platí, tedy vždy jen jedna jednička na řádku => hodně nul, lze dobře komprimovat
  - vyhledávání je rychlé používají se operace AND, OR atd.
  - insertem se všechny vektroy prodlouží, přidáním další možnosti (rozšíření domény) se přidá další bitový vektor
  - lze využívat i pro GROUP BY, COUNT atd.

## 17.3 Hashování

- perfektni hasovani Cormacka
- perfektni hasovani Larsona a Kalji
- Deskriptory stranek, Grayovy kody (viz Vícerozměrné dotazy implementované pomocí hashovacích metod)
- hasovani Fagina
- dynamicke hashovani Litwina
- skupinove stepeni stranek

## 17.4 B-stromy

- redundantni, neredundantni B-strom
- redundantni, neredundantni B*-strom
- redundantni B+-stromy
- prefixove stromy
- (a, b)-stromy
- stromy s promenlivou delkou zaznamu
- vicerozmerne B-stromy

## 17.5 Datové struktury na externí paměti

- hromada — nehomogenni záznamy (ruzne delky) jsou ukládány za sebou (bez dalších podmínek) — vyhledání čehokoliv = projití celé hromady (v nejhorším případě)
- sekvenční soubor — podobný jako hromada, jen záznamy jsou pevné délky (homogenni)
  - uspořádaný sekvenční soubor — záznamy uspořádány podle klíče, nově přidávané se ukládají do "souboru aktualizací" a občas je provedena reorganizace, kdy je "soubor aktualizací" vyprázdněn a data z něj zatříděna do uspořádaného sekvenčního souboru. Nalézt záznam lze pak půlením intervalů v case O(log_2(n)).
  - indexsekvenční soubor — obdobně jako předchozí jsou záznamy setříděny podle klíče, navíc existují indexy do jednotlivých bloků (stránek). Indexy jsou tvořeny klíči některých záznamů. Celý index je pak jakýsi B-strom. Při přidávání není index reorganizován, ale je použita oblast přetečení, kam je nový záznam vložen. K záznamu, za který měl být nový záznam vložen se pak uloží číslo bloku i číslo záznamu, kde nový záznam leží. V oblasti přetečení jsou tak vytvářeny lineární seznamy.
  - invertovaný (indexovaný) soubor — indexují se přímo záznamy, ne bloky (jak tomu je v indexsekvenčním souboru), data nejsou v souboru nijak uspořádána, jednotlivé bloky nemusí být na disku za sebou. Indexů může být více různých. Při přidávání není použita další struktura, přímo se upravují indexy. Používá se v DIS (invertovany soubor).
- ukládání pomocí hashování
- b-stromy

## 17.6 Vícerozměrné dotazy implementované pomocí hashovacích metod

- viacrozmerne dotazy — obsahuju viac atributov (n>1)
  - na uplnu zhodu — hodnoty vsetkych n atributov su pevne dane (n=2: MENO='Jiri' AND MESTO='Praha')
  - na ciastocnu zhodu — hodnoty niektorych atributov (<n) su pevne dane
  - na uplnu intervalovu zhodu — pre vsetkych n atributov su dane intervaly hodnot (napr. VEK>22)
  - na ciastocnu intervalovu zhodu — intervaly hodnot su dane len pre niektore atributy
- implementacia
  - **signatury** — d bitove binarne retazce
    - signatura zaznamu (dlzky d) je zretazenim signatur jednotlivych atributov, pre kazdy atribut A_i je samostatna hasovacia funkcia h_i, ktora vytvori signaturu dlzky d_i. Plati, ze d_0+d_1+...+d_n=d; n-pocet atributov. Signatura dotazu sa tvori podobne, nezadane atributy su reprezentovane retazcom nul.
    - signatura dotazu Q sa porovnava so signaturami zaznamov S — na pozicii, kde je v Q jednicka, musi byt jednicka aj v S, inak zaznam do odpovede nepatri
  - **deskriptory stranok**
    - vrstvia sa deskriptory zaznamov v danej stranke pomocou logickeho OR. Deskriptor zaznamu je opat zretazenie deskriptorov atributov A_i, pre kazdy atribut je dana hasovacia funkcia h_i, ktora priradi kazdej hodnote atributu binarny retazec obsahujuci prave 1 jednicku.

    * subor deskriptorov stranok je pripojeny k primarnemu suboru, ak je velky, moze byt dvojurovnovy (prim. subor sa rozdeli do segmentov, kazdy ma svoj subor deskriptorov, tie tvoria 1. uroven; 2. uroven ma deskriptory ukazujuce do 1.)
    * rychlejsie ako signatury, lebo sa musi porovnat menej deskriptorov (ale na ukor pamate); lepsie ako indexovany subor, lebo nerastie cas vyhodnocovania s rastucim poctom atributov v dotaze, pamatove naroky su podobne
  - **grayove kody** — binarne retazce, hodnoty nasledujuce za sebou sa lisia vzdy len v jednom bite
    * adresa stranky dana signaturou je interpretovana ako grayov kod
    * zaznamy vyhovujuce signature dotazu tvoria zhluky, kde zaznamy su fyzicky za sebou v jednej stranke (alebo viacerych po sebe iducich strankach), nikdy nie je viac pristupov na disku ako u binarneho kodovania
    * dobre pre dotazy na ciastocnu zhodu pri malom pocte zadanych atributov a pre dotazy na intervalovu zhodu

## 17.7 Vícerozměrná mřížka

Slajdy Pokorný

## 17.8 Vícerozměrné stromy

### Vícerozměrné B-stromy

slajdy Pokorný

- lze pomocí nich indexovat soubor podle několika atributů současně.
- celý strom má tolik hladin, kolik je atributů (hlavní strom není B-Strom!)
- pro každý atribut jsou vytvořeny stromy — pro první jeden, pro druhý tolik, kolik má první atribut různých hodnot atd. U každé hodnoty atributu (uzlu stromu) existuje index na strom s dalším atributem.
- jednotlivé vrstvy celého stromu jsou indexovány polem Level (vrstva = jeden atribut) — slouží pro přeskočení prvních i atributů, které nebyly v dotazu zadány
- stromy v jedné vrstvě jsou propojeny do spojáku pomocí NEXT v kořeni, poddoména jednoho stromu je určena pomocí LEFT a RIGHT (uloženy v kořeni), které ukazují do další hladiny na první a poslední strom, který se týká daného stromu. Tedy každý prvek z vrstvy i-1 má ve vrstvě i určeny všechny prvky ve vrstvě i+1, LEFT a RIGHT dává vše.
- podobná struktura by pravděpodobně šla definovat i pro obecný, nebo jakýkoliv jiný vyhledávací strom.

## 17.9 Přístupové metody k prostorovým objektům: R-stromy a jejich varianty

viz wen:R-Tree

## 17.10 Databáze textů: modely (boolský, vektorový)

#### Boolský model DIS

Každý dokument je popsán množinou termů, které obsahuje. Termy jsou uloženy v indexovém souboru a odtud potom probíhá vyhledávání.

- určení termů:
  - ručně — každý může určit jiné termy, dokonce i jeden člověk se může 2x rozhodnout jinak
  - automaticky — jednoznačné, ale bez porozumnění textu
  - při určování termů by se měla vynechávat nevýznamová slova (předložky, spojky...), ale také příliš specifická slova
- Indexy lze nad kolekcí dokumentů vytvářet dvojím způsobem. Buď je množina termů pevně daná a pro každý nový dokument se aktualizují jen indexy (řízená indexace), nebo se s každým novým dokumentem přidávají i nové termy (neřízená indexace).

- Vyhledávání:
  - probíhá klasicky přes termy pomocí “AND”, “OR”, “NOT”
  - lze také využít faktografické informace o dokumentu (autor, datum vytvoření...)
  - lze použít zástupné znaly ?, *
- Proximitní omezení
  - při vyhledávání lze také užít proximitních omezení, tedy informaci o tom, v jaké vztahu dva hledané termy jsou (o kolik slov jsou vzdáleny, zda jsou ve stejném odstavci, nebo větě.
  - pro práci s omezeními je třeba buď mít přítomny dokomenty a informaci zjišťovat přímo v nich a nebo rozšířit index -> namísto <term_id, doc_id> musí existovat index <term_id, doc_id, par_nr, sent_nr, word_nr>

Nevýhody boolského DIS:

- při zadávání dotazu jsou všechny termy stejně důležité
- ve výsledku disjunktního dotazu jsou promíchány dokumenty které obsahují všechny požadované termy s těmi, které obsahují pouze jeden
- ve výsledku konjunktivního dotazu nejsou dokumenty, které obsahují neobsahují žádný z termů stejně tak jako ty které neobsahují pouze jeden

### Vektorový model DIS

Vektorový model je mladší než boolský (asi o 20 let) a vznikl aby odstranil chyby boolského modelu — především díky možnosti ohodnotit termy jak v dotazu, tak v indexu.

- Každý dokument je v indexu definován váhami pro jednotlivé termy.
- dotaz se zadává vektorem vah pro požadované termy <w1, w2, .. , wn>, w in <0,1>
- protože dlouhé dokumenty by byly zvýhodněny (obsahují jistě více termů), provádí se normalizace vektorů, to lze provádět:
  - během indexace, což nezvětšuje odezvu, je však zapotřebí občas vše znovu “přenormalizovat”
  - během vyhledávání — zpomaluje odezvu, je v definici podobnostní funkce
- Oproti boolskému modelu lze omezit velikost výstupu — výstup je seřazen dle relevance a tedy lze omezit jak počtem výstupů tak minimální relevancí.
- Podobnost mezi dotazem a dokumentem se určuje pomocí podobnostní funkce. Taková funkce není jednoznačně daná, podobnost může být tedy při různých implementacích různá. Základní podobnostní funkce je skalární součin přes dotaz a dokument (sim(q,d) = q x d), jsou ale i složitější:

Kosinová míra: $sim(q,d) = \frac{q \times d}{\sqrt{(\sum(q_i^2))} * \sqrt{(\sum(d_i^2))}}$
Jaccardova míra: $sim(q,d) = \frac{q \times d}{\sum(q_i) + \sum(d_i) - \sum(q_i * d_i)}$ a další.

- Pokud je po vektorovém modelu požadován i dotaz s negací, lze váhu w uvažovat z <-1, 1> a stejně tak i zadávat dotaz.
- Indexace:
  - frekvence termu v dokumentu TF = #výskytů termu / #všech termů v dokumentu
  - je potřeba vyřadit nevýznamová slova (stop list = {the, a, an, on ... })
  - ignorovat termy s velmi nízkým výskytem
  - normalizovat frekvenci ostatních termů $NTF_i = \frac{1}{2} + \frac{1}{2} * \frac{TF_i}{max(TF_j)}$, j = index jednotlivých termů
- Inverzní frekvence termu v dokumentech
  - Nasel jsem dvě definice. První je od Kopeckého druhá od Pokorného. Definice jsou v podstatě matematicky ekvivalentní.
  - n je počet všech dokumentů.
  - k je počet dokumentů ve kterých se term vyskytuje.

1. $ITF = -log\left(\frac{k}{n}\right)$
2. $IDF = log\left(\frac{n}{k}\right) + 1$

Váha termu i v j-tém dokumentu se urči jako $w_{ij} = NTF_{ij} * ITF_j$

- Dotazování (dotaz má stejnou reprezentaci jako index):
  - Zadáním vektoru dotazu
  - odkazem na zaindexovaný dokument (tedy "najdi mi dokumenty podobné tomuto")
  - odkazem na jiný dokument (není problém pro něj spočítat vektor termů)
  - přímo vložený text (obdoba předchozího)
- Lze uplatnit i zpětnou vazbu, kdy uživatel označuje relevantní dotazy a DIS na to reaguje přehodnocením výpočtu.

### Vyhledávání v textech

### Signatury, metody implementace signatur (vrstvené kódování)

Slajdy pokorny

- signatura bloku textu je d-bitový řetězec, který v sobě nese informaci o tom, které termy v bloku jsou
- **vrstvění signatur**
  - každý term má svou signaturu (vypočtenou pomocí hash funkce) dlouhou d bitů a signatura bloku = OR přes všechny signatury termů v bloku.
- **řetězení signatur**
  - signatury se spojují do řetězce — vhodné pro strukturovaná data (autor, rok vydání, nakladatelství...)
- **transponovaný soubor signatur**
  - namísto N řetězců délky d mám d řetězců délky N (pokud signatury bloků naskládám do matice, koukám se pak na ně místo jako na řádky jako na sloupce)
  - protože počet 1 v dotazu délky d je o mnoho menší než d a můžu porovnávat jen řetězce kde má dotaz jedničku, porovnávám toho daleko méně
  - složitější aktualizace
- **soubor indexovaných deskriptorů**
  - stejně jako z termů vytvářím vrstvením signaturu bloku, můžu pro více bloků vytvořit společnou signaturu
- **dvouúrovňové vrstvené kódování**
  - každý blok má dvě signatury — vnětší (délky ds) a vnitřní (délky dn) kde dn << ds, bloky jsou rozděleny do skupin a každá skupina má vrstvenou signaturu ds — podle ní se rozlišují skupiny a ve skupin se bloky rozlišují podle signatury dn (tedy najdu skupinu bloků a pak v ní blok)
- **rozdělený soubor signatur**
  - signaturu bloku délky d rozdělím na dva kusy (prefix a zbytek) a pak seskupím "zbytky" se stejnými prefixy — pokud testuju, otestuju nejdřív prefix a pokud se neshoduje, do skupiny vůbec nemusím chodit, čímž ušetřím celkem dost času

Metody je možné různě kombinovat tak, aby výsledná struktura byla co nejefektivnější.

### Uspořádání odpovědi

## 17.11 Komprese dat: predikce a modelování

### Reprezentace celých čísel

Slajdy k OZD II (od slajdu 21) (Fibonacciho, Eliasovy kódy, fázování)

### Obecné metody komprese

- ztrátová (nazývá se komprese)
- bezeztrátová (nazývá se kompakce)
- statická
- semiadaptivní
- dynamická

Dvě fáze:

- modelování — přiřazení pravděpodobností jednotkám textu (znaky nebo m-tice znaků pevné délky)
- kódování — transformace do nového kódu

### Komprese bitových map

Slajdy k OZD II (od slajdu 92)

- pomocí Huffmanova kódování
  - kódováním posloupností pevné délky
  - kódování běhů (posloupnost nul ukončených jedničkou apod.)

### Komprese řídkých matic

Něco lze nalézt u Koubka v kapitole o reprezentaci Trie pomocí matic a jejich kompresi — viz materiály k přednášce Datové struktury II

### Trie

### Komprese textů

### Huffmanovo kódování (statické, dynamické)

Slajdy k OZD II (od slajdu 40) (statické, adaptivní kódování)
Dobře popsáno také na stránkách k DIS

### Aritmetické kódování

viz wen:Arithmetic coding

Slajdy k OZD II (od slajdu 47)

Dokumenty k DIS (myslím, že v posledním kroku příkladu je chyba)

### LZ algoritmy

Slajdy k OZD II (od slajdu 56) (LZ77, LZSS, LZ78, LZW a další)

## 17.12 Uzamykací protokoly

viz wen:Concurrency control#Concurrency control mechanism

## 17.13 Časová razítka

viz wen:Timestamp-based concurrency control#Informal

## 17.14 Distribuované transakce

viz wen:Distributed transaction

viz wen:Two-phase commit protocol

Ve stručnosti:

- říká se jim také globální transakce
- musejí splňovat ACID
- jsou to transakce, které pracují s více uzly najednou (s více databázemi v síti apod.)
- celkem jednoduchý algoritmus, který problém řeší je "Dvoufázový commit"

### Dvoufázový commit

**1.fáze (commit-request phase)**

- kordinátor pošle všem uzlům dotaz, který je zde potřeba vykonat
- uzly vykonají vše až do doby, kdy by měly commitovat/abortovat
- každý uzel pošle svůj výsledek koordinátorovi (YES/NO)

**2.fáze (commit phase)**

pokud koordinátor dostal od všech uzlů odpověď YES (success)

- koordinátor pošle commit zprávu všem uzlům
- všechny uzly uvolní své zdroje a zámky
- všechny uzly pošlou koordinátorovi výsledek dotazu
- koordinátor vše skompletuje

pokud koordinátor dostal alespoň jedno NO (failure)

- koordinátor všem uzlům pošle abort zprávu
- každý z uzlů udělá vlastní abort a uvolní zámky a zdroje
- všechny uzly pošlou koordinátorovi výsledek dotazu (??? nechápu ale k čemu to je ???)
- poté co koordinátor obdrží výsledky abortuje celou transakci

### Stromový 2-fázový commit (Tree two-phase commit protocol)

Obdoba předchozího jen s rozdílem, že koordinátor je kořenem stromu a vše se děje na stromové struktuře — commity jsou sbírány do uzlu stromu a až jsou všechny, jsou propagovány víš až ke kořeni. Oproti tomu abort je propagován hned.

Problémem 2PC protokolu je, že každý z uzlů čeká, až dodělá práci poslední z uzlů, čímž dost dlouho blokuje.