INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

DATABÁZOVÉ SYSTÉMY

FYZICKÁ ORGANIZACE DAT V SOUBORECH

Ing. Lukáš OTTE, Ph.D.

Ostrava 2013

Tento studijní materiál vznikl za finanční podpory Evropského sociálního fondu (ESF) a rozpočtu České republiky v rámci řešení projektu: CZ.1.07/2.2.00/15.0463, MODERNIZACE VÝUKOVÝCH MATERIÁLŮ A DIDAKTICKÝCH METOD

OBSAH

1 FYZICKÁ ORGANIZACE DAT V SOUBORECH

OBSAH KAPITOLY:

Principy fyzické organizace dat v databázi

MOTIVACE:

Po prostudování této přednášky budete schopni vyjmenovat a vysvětlit základní principy organizace dat v databázích. Rovněž budete schopni popsat u jednotlivých typů fyzické organizace dat smysl provádění databázových operací.

CÍL:

Seznámení se základními principy organizace dat v databázích.

1.1 TEXT PŘEDNÁŠKY

Důvodem existence databází je evidovat v nich rozsáhlá data. Ale pouhá evidence není jediným smysluplným využitím databází. O smysluplném využití databáze lze hovořit, pokud dokážeme uložená data zpracovávat na informace a podle okamžitých potřeb tyto informace vyhledávat.

Organizace dat je jisté uspořádání dat, které má za účel umožnit efektivní zpracování těchto dat potřebných pro aplikace (v tomto případě databázové systémy). Zahrnuje postupy a metody, jak data na médiích ukládat a jak je hledat.

Výkon a tím i efektivita používání databáze je velmi závislá na implementaci operací na fyzické úrovni. Komunikaci s databází zajišťují čtyři základní databázové operace, jimiž jsou:

- vyhledávání záznamu podle hodnoty jedné nebo více položek *SELECT*;
- vložení nového záznamu *INSERT*;
- modifikace záznamu *UPDATE*;
- rušení záznamu *DELETE*.

Záznamy mohou být v databázi uloženy fyzicky různým způsobem. Jde nyní o to zvolit při implementaci SŘBD takovou organizaci uložení záznamů v databázi, aby výše uvedené čtyři databázové operace probíhaly co nejrychleji.

V tuto chvíli se nebudeme zabývat nalezením *fyzické adresy uložení záznamu*, které je plně v kompetenci operačního systému, ale budeme se zabývat nalezením logické adresy záznamu. Logická adresa je podle okolností tvořena buď pořadovým číslem záznamu (v případě že se jedná o soubor s pevnou délkou) anebo relativní adresou v rámci souboru.

1.2 SEKVENČNÍ SOUBORY

Sekvenční soubory jsou nejjednodušším způsobem fyzické organizace dat, které vycházejí z přirozeného uspořádání záznamů podle pořadí jejich uložení. Implementace sekvenčních souborů je jednoduchá. Základní databázové operace se provádějí následovně:

Insert

Nový záznam se uloží na konec souboru, k tomu stačí jeden přenos záznamu z paměti na disk.

Select

V případě, že vyhledáváme záznam se zadaným pořadím, určí se adresa záznamu přímo z pořadí a délky záznamu. V případě, že vyhledáváme záznam podle hodnot a ne podle pořadového čísla, pak je nutno prohledat soubor sekvenčně. Tedy je nutno každý záznam postupně načíst a otestovat, zda odpovídá zadané podmínce. Pokračuje se s načítáním do té doby, než je nalezen požadovaný záznam.

Update

U databázové operace modifikace záznamu to znamená provést postupně operace *nalezení, upravení záznamu a jeho opětovný zápis na stejnou adresu.*

Delete

V případě rušení záznamu se u sekvenčních souborů obvykle neprovádí fyzické vymazání, ale pouze označení neplatnosti záznamu při zachování jednotlivých položek. K vyznačení neplatnosti záznamu je obvykle vyhrazeno místo v samotném záznamu o určité velikosti (bit, bajt).

Při dalším prohledávání se pak zrušené záznamy nezpracovávají, ale zabírají v samotném souboru místo. Tento způsob má však řešení. V případě vkládání nových záznamů (celých vět) se tento záznam uloží na první místo zrušeného záznamu. V případě, že takové místo neexistuje, pak se uloží nakonec souboru. Při existenci primárního klíče záznamu je však nutné kontrolovat jedinečnost tohoto klíče v celém souboru.

Příklad: *Evidence dat o zaměstnancích v tabulce. Zaměstnanci jsou zapisováni v pořadí, jak byli do firmy přijati. Při hledání Dudka jde o jeden přenos záznamu, při hledání Nováčkové o n přenosů.*

adresa	delete	**osob**	**jmeno**	…
1		3456	Dudek Jindřich	
2		1243	Kovář František	
3	*	3333	Novák Karel, ing.	
4		5734	Horák Ivo	
5		2578	Sedlák Jiří	
6		9999	Forman Zdeněk	
7		3579	Anděl Gabriel	
…		…	…	
n		7766	Nováčková Iva	
n+1		8765	Gavendová Miriam	

Obrázek 1 - Příklad sekvenčního souboru [1]

1.3 SETŘÍDĚNÉ SEKVENČNÍ SOUBORY

Pokud se v souboru často vyhledává podle některého klíče (zde myslíme vyhledávací klíč, což nemusí být vždy primární klíč) a provádí se relativně málo změn těchto klíčů, je vhodné uchovávat soubor v setříděném tvaru podle vyhledávacího klíče. Znamená to provést setřídění souboru po každé změně, např. vložení nové věty nebo modifikaci klíče, což výrazně zpomaluje tyto operace.

Ovšem vyhledávání podle klíče je pak mnohem rychlejší a jednodušší (např. metodou půlení intervalu). S výhodou se tedy tento způsob uložení používá u souborů s velmi řídkými změnami a v případech, kdy se nepředpokládá častá aktualizace.

1.4 ZŘETĚZENÉ SOUBORY

Místo setříděných sekvenčních souborů je možno použít také techniku zřetězení záznamů. Proti sekvenčnímu souboru obsahuje každý záznam navíc jednu položku. V níž je uložen ukazatel na následující záznam v souboru podle daného uspořádání. V souboru je tak vytvořen seznam či řetěz vzájemně propojených záznamů, kdy řetězy mohou realizovat uspořádání dle libovolného kritéria.

Insert

Při této operaci se záznam fyzicky zapíše kamkoliv, pak se v seznamu vyhledají sousední záznamy dle udržovaného pořadí a přesměrují se ukazatele.

Select

Tato operace se provádí tak, že se seznam prohledává postupně pomocí ukazatelů a testuje, zda záznam vyhovuje vyhledávací podmínce. Tento způsob vyhledávání je

však mnohem náročnější na práci s diskem a pamětí a je vhodný spíše pro použití krátkých seznamů, kdy je možno načíst celý seznam do paměti.

Update

Modifikace záznamu znamená vyhledání záznamu a po modifikaci jeho zápis zpět. Ovšem v případě modifikace položek, které mají vliv na uspořádání seznamu, je nutné provést smazání původního záznamu a uložení záznamu nového.

Delete

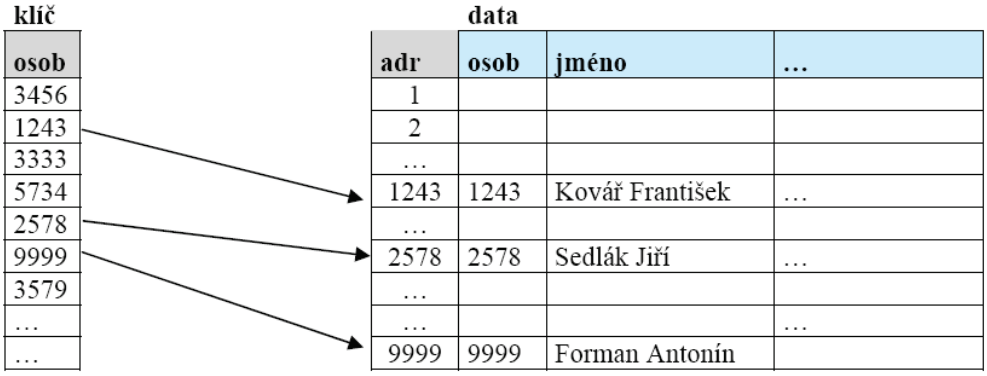
Operace zrušení záznamu se provede tak, že se vyhledá umístění záznamu v seznamu a přesměrují se ukazatele.

1.5 SOUBORY S PŘÍMÝM ADRESOVÁNÍM

Tento způsob fyzické organizace dat zajišťuje velmi rychlý přístup k záznamům prostřednictvím jednoznačných klíčů, ze kterých se určí adresa záznamu v souboru.

Principem metody je, že jednoznačný klíč záznamu se zakóduje do jednoznačného čísla, které pak přímo určuje adresu záznamu v souboru, čímž je pak možné záznam načíst pomocí jediného přístupu na disk a následujícím přístupem na disk pak záznam zapsat po jeho modifikaci.

Příklad: *Čtyřciferné osobní číslo je klíčem záznamu zaměstnance, navíc není nutné ho kódovat číselně. Znamená tedy přímo adresu záznamu. Při hledání podle osobního čísla stačí jediný přenos z diskové adresy totožné s osobním číslem.*



Obrázek 2 - Příklad souboru s přímým adresováním [1]

Řekněme, že tento seznam zaměstnanců obsahuje 100 aktivních záznamů. Pak potřebný prostor pro těchto 100 zaměstnanců zde je 9999 adres.

Interval získaných adres tak může být ohromný a velmi řídce obsazený, čímž dochází k velkému plýtvání kapacitou paměti.

Proto se užívá speciální funkce zvané hašovací (hash function), která transformuje původní interval adres do číselného intervalu požadované velikosti. Velikost výsledného intervalu je zvolena tak, aby zhruba odpovídala skutečnému počtu záznamů.

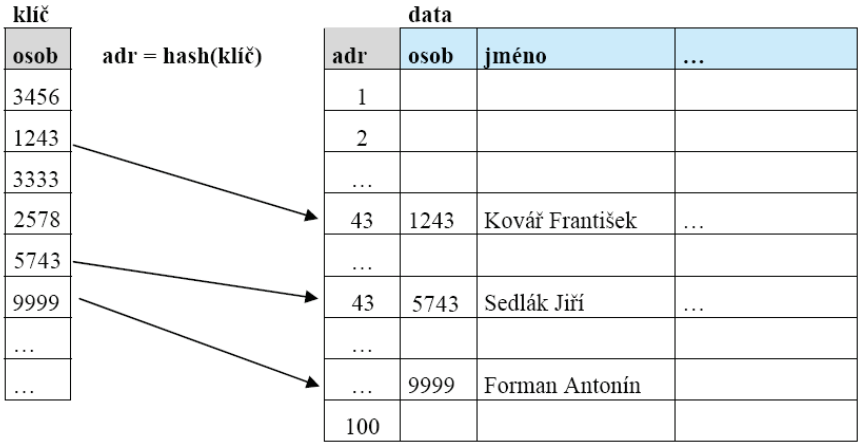
1.5.1 Rovnice 1 - Hašovací funkce

$$h(k) = k \bmod M$$

k je v tomto případě hodnota klíče (osobní číslo), *mod* je funkce MODULO, tedy dělení zbytkem po celočíselném dělení a M je zvolené prvočíslo, v tomto případě nejvyšší počet možných aktivních adres.

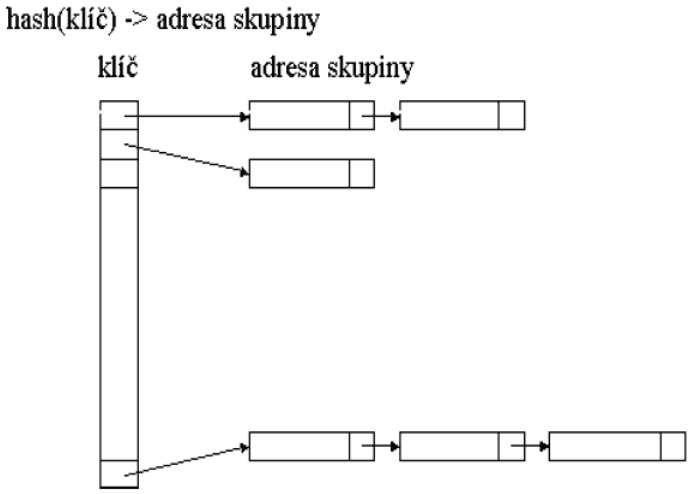
Použitím hašovací funkce však může dojít k nejednoznačnosti výsledné adresy, protože několika původním klíčům může odpovídat stejná adresa. Situace se pak řeší tak, že nejednoznačná adresa znamená adresu skupiny všech těchto záznamů se stejnou hodnotou hašovací funkce, přičemž se záznamy v rámci skupiny ukládají zřetězené. Protože v tuto chvíli již jde o krátké seznamy, ke kterým je rychlý přístup.

Jestliže počet záznamů souboru trvale roste, je potřeba občas upravit hašovací funkci a provést reorganizaci souboru.



Obrázek 3 - Příklad souboru s přímým adresováním a použitím hašovací funkce [1]

Příklad ukazuje, jak s pomocí hašovací funkce transformuje SŘBD čtyřciferné osobní číslo zaměstnance na adresu z intervalu <1,100>. Použitou hašovací funkcí je funkce Modulo 100 a výsledkem je poslední dvojčíslí osobního čísla jako adresa záznamu. [1]



Obrázek 4 - Příklad použití adresování se zřetězeným odkazováním pomocí ukazatelů [1]

Základní databázové operace pak vypadají následovně:

Insert

Pro nový záznam se spočítá adresa skupiny záznamů, v ní se prohledají záznamy (pro kontrolu jednoznačnosti klíče), nový záznam se pak uloží na první volné místo ve skupině.

Select

Vyhledání záznamu podle klíče je velmi rychlé a čas potřebný k výpočtu adresy pomocí hašovací funkce je zanedbatelný: z klíče se vypočte adresa skupiny záznamů a na ní se začne prohledávat zřetězený seznam až do doby nalezení hledaného záznamu.

Vyhledávání podle neklíčové hodnoty se však naopak prodlužuje, protože je potřeba sekvenčně procházet prázdná místa a seznamy.

Update

U operace modifikace to znamená nejprve vyhledání, modifikaci a posléze zpětné uložení. Ovšem při modifikaci klíče je nutné provést nejprve zrušení a pak nový záznam.

Delete

V případě rušení záznamu se provede vyhledání, označení neplatnosti a přesměrování ukazatelů z předchůdce na následníka.

1.6 INDEXOVANÉ A INDEXOVÉ SOUBORY

Jde o nejčastější způsob organizace dat v relačních SŘBD. Základem indexované organizace je sekvenční datový soubor, ke kterému existuje jedna nebo několik dalších pomocných tabulek, pomocí nichž můžeme v datovém souboru rychleji hledat.

Základní datový soubor, který je doplněn takovou pomocnou tabulkou, nazýváme **indexovaným souborem**, pomocnou tabulku nazýváme **indexovým souborem**.

Do pomocné tabulky se umístí hodnota vyhledávacího klíče (**indexu**) a adresa (např. pořadové číslo) záznamu. Tabulka je organizována tak, aby **vyhledání klíčové hodnoty** proběhlo rychle pro jakoukoli část datového souboru.

Například při setřídění pomocné tabulky dle indexu se hledá binárně hodnota klíče. Na vyhledaném řádku v indexovém souboru se zjistí hodnota adresy v datovém souboru a jediným přístupem do tohoto souboru dat se načte hledaný záznam. Často je indexová tabulka dost malá a může tak být při prohledávání umístěna celá v operační paměti, čímž se hledání výrazně zrychlí.

Rozdíl indexování oproti zřetězené organizaci je v tom, že ukazatele nemusí být vždy součástí záznamů (jako u zřetězených organizací), ale mohou být uloženy ve zvláštním (indexovém) souboru. Přičemž indexem nemusí být jen primární klíč, ale kterákoliv položka datového souboru nebo dokonce skupina několika položek. Je však vhodné, aby do této skupiny byly zařazeny ty položky, podle nichž se bude často provádět vyhledávání.

Pokud je indexem primární klíč, pak hovoříme o tzv. **primárním indexování**, v opačném případě o **sekundárním indexování**. Při sekundárním indexování musíme počítat s tím, že stejné hodnoty indexu v datech nabývá více záznamů a tak jedné hodnotě položky odpovídá více adres.

Příklad: *Máme-li tabulku zaměstnanců s poli OSOB (osobní číslo), PLAT a PROC (procento daně z příjmu).*

indexovaný datový soubor

adr	OSOB	PLAT	PROC
1	106	850	20
2	121	870	10
3	124	2400	30
4	100	1230	10
5	133	1340	30
6	110	1900	10
7	120	740	20
8	117	1400	30
9	130	1600	20
10	111	1600	30

primární index

OSOB	Adresa
100	4
106	1
110	6
111	10
117	8
120	7
121	2
124	3
130	9
133	5

sekundární indexy

PROC	Adresy
10	2,4,6
20	1,7,9
30	3,5,8,10

nebo

PROC	Adresa
10	2
10	4
10	6
20	1
…	

Obrázek 5 - Příklad použití indexovaných a indexových souborů [1]

V případě, že volíme indexování pro všechna pole tabulky, pak hovoříme o tzv. **úplně invertovaném souboru** a v tomto případě je pro načtení indexového souboru potřeba dvojnásobek paměti, než pro samotný datový soubor.

Základní databázové operace pak vypadají následovně:

Insert

U vkládání nového záznamu se tento uloží na konec datového souboru, přidá se do všech indexových souborů a ty se setřídí.

Select

Vyhledávání podle indexovaných položek se provádí binárním vyhledáním hodnoty indexu v příslušném indexovém souboru, odkud pak následně získáme adresy záznamu v souboru datovém. Jediným přenosem pak z datového souboru získáme konkrétní záznam.

Vyhledávání podle neindexovaných položek se provádí stejně jako u sekvenčního souboru.

Update

U operace modifikace to znamená nejprve vyhledání, modifikaci a posléze zpětné uložení. Ovšem při úpravě některé indexované položky je nutné upravit indexové soubory a přetřídit je.

Delete

V případě rušení záznamu se provede vyhledání a označení záznamu za neplatný. Nakonec se musí přetřídit indexový soubor.

Výhodou techniky indexování je, že umožňuje rychlejší přístup k tabulkám pomocí indexovaných položek, ale na druhé straně spotřebuje více místa na disku pro indexové soubory a také je zapotřebí vyšší režie na údržbu indexových tabulek při změnách datového souboru.

1.6.1 Hierarchické indexování

Hierarchické indexování se používá v případech, kdy jsou datové soubory obzvláště velké, a pro hledání v indexovém souboru je možno použít další indexový soubor. Tím je pak vytvořena celá hierarchická struktura indexových souborů. Struktura je rozdělena do

jednotlivých úrovní, přičemž na vyšších úrovních indexace se objevují indexy skupin indexů nižší úrovně – jinak řečeno – indexy druhé úrovně odkazují na indexy úrovně první.

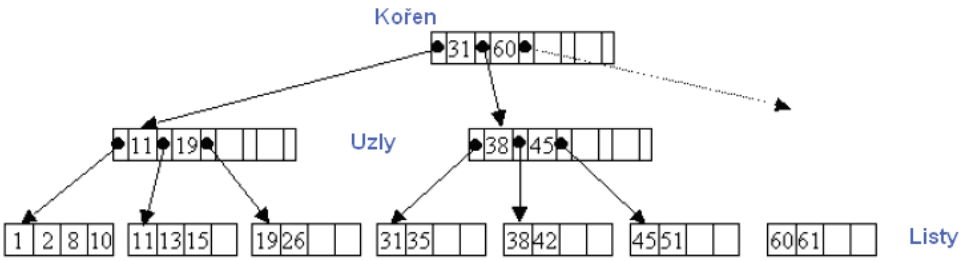
Při samotném vyhledávání se pak postupuje od nejvyšší úrovně, přičemž se nehledá přesná hodnota indexovaného pole, ale interval, do kterého náleží. Až nejnižší úroveň indexu ukazuje přímo do datového souboru.

1.7 B-STROMY (N-STROMY)

B-stromy (Binární stromy) se často využívají jako varianta hierarchického indexování pro uspořádání jednotlivých úrovní indexů. Jde o druh hierarchie, jejíž zvláštností je, že zavádí limity jak na maximální, tak i na minimální počet potomků vrcholu. Díky této vlastnosti je B-strom vyvážený a lze říct, že jednotlivé databázové operace probíhají v logaritmickém čase. Prostřednictvím B-stromů minimalizujeme počet přístupů do paměti.

Rozšířením B-stromů jsou tzv. N stromy (N-ární stromy), které stanoví limit maximálního počtu potomků konstantou n a minimální počet potomků vrcholu jako $^{n}/_{2}$. Zjednodušeně lze říci, že N-strom je v podstatě B-stromem stupně n.

Například, máme-li B-strom hloubky 2 a počet potomků každého uzlu je 1001, můžeme do něj uložit milion klíčů a ke každé položce se dostaneme maximálně po dvou diskových operacích.



Obrázek 6 - Příklad použití B-stromu

B-strom je tvořen kořenem (jeden význačný vrchol), uzly a listy (koncové prvky stromu). Listy stromu vzniknou tak, že indexové soubory všech úrovní (mimo kořen) se rozsekají na takové části (bloky), které svou velikostí nebrání tomu, aby byly načteny jediným přenosem z disku do paměti. V rámci malého a setříděného uzlu se najde nejbližší hodnota hledaného vyhledávacího klíče velmi rychle. Platí, že všechny cesty od kořene stromu do libovolného listu jsou stejně dlouhé – stejným počtem přenosů se dojde k datům.

Základní databázové operace jsou následně prováděny takto:

Insert

U vkládání nového záznamu se tento uloží na konec datového souboru, přidá se do všech indexových souborů a to následovně.
Při vkládání nového záznamu se najde příslušný blok a mohou nastat dvě možnosti. Buď v nalezeném bloku je místo, takže se může přidat vkládaný záznam, nebo nalezený blok je plný, takže se musí vytvořit nový blok. Z původního plného bloku se vytvoří dva bloky, každý obsahuje polovinu záznamů a zbylá prázdná místa.

Do vyšší úrovně musíme nový blok nižší úrovně zaznamenat a opět mohou nastat dva případy. Proces se opakuje až do kořene stromu a případně se musí kořen rozdvojit. Pak se přidá nový kořen a vznikne o úroveň víc v indexové struktuře.

Select

Při hledání najdeme cestu ve stromě od kořene až k listu, v němž by měl být hledaný záznam (pokud v souboru existuje). V každém uzlu najdeme správnou větev porovnáním hledaného klíče s klíči v uzlu. Klíče v uzlu mohou udávat minimální (příp. maximální) hodnotu klíče, která je příslušnou větví dosažitelná.

Update

Záznam se vyhledá, změní a zapíše zpět. Při modifikaci klíče se provede zrušení původního záznamu a zavedení nového záznamu včetně indexu.

Delete

Rušení záznamů se provádí opačně, než vkládání. Při zrušení posledního záznamu bloku se zruší i odkaz na něj, totéž se promítne do vyšších úrovní, případně se v krajním případě může hierarchie indexů o jednu úroveň snížit.

2 ZKUŠEBNÍ OTÁZKY:

1) Co je to fyzická organizace dat?
2) Která aplikace se stará o fyzickou organizaci dat v databázi?
3) Popište princip sekvenčních souborů.
4) Popište princip indexování.

3 DOPLŇUJÍCÍ ZDROJE INFORMACÍ:

[1] ŠARMANOVÁ, J. Teorie zpracování dat [online]. Ostrava, 2003, 76 s. [cit. 2012-08-07]. Dostupné z: < http://www.miroslavkrupa.cz/download/TZD_dist_1.pdf>

4 POUŽITÁ LITERATURA:

[1] ŠARMANOVÁ, J. *Teorie zpracování dat* [online]. Ostrava, 2003, 76 s. [cit. 2012-08-07]. Dostupné z: < http://www.miroslavkrupa.cz/download/TZD_dist_1.pdf>

[2] POKORNÝ, J. a HALAŠKA, I. *Databázové systémy*. Praha, 2003, 145 s.