**13. Fyzická implementace, vykonávání dotazů, indexy**

* Nakreslete binární vyhledávací strom po vložení čísel 20, 14, 5, 10, 8, 20, 3.

## Uveďte asymptotické časové složitosti pro tabulku typu halda a index typu B-strom pro operace vložení záznamu, smazání záznamu a bodový a rozsahový dotaz.

* Tabulka typu halda má složitost vkládání O(1), pro hledání O(n), pro mazání O(n), bodový rozsah O(n)
* Tabulka typu index má pro všechny operace složitost O(log n)

**Popište operace vkládání a vyhledávání v tabulce typu halda, uveďte také složitost operací**

* Operace vkládání je provedena se složitostí O(1), nový záznam je vložen na poslední pozici v haldě, která je volná.
* Vyhledávání má složitost O(n). Halda je procházena sekvenčně

## Co je to složený index? Kdy jej používáme? Popište výhody i nevýhody, uveďte příklad.

* Index obecně umožňuje rychlé vyhledání dle klíče
* Pokud se chceme dotazovat na hodnoty dvou a více atributů v jednom dotazu, můžeme vytvořit tzv. složený index. (Klíčem je více než jeden atribut)
* CREATE INDEX Producer\_name ON Producer(name, address);
* SELECT \* FROM Producer WHERE name=‘mat0354‘ AND address=‘mojeadresa‘;
  + 3 logické přístupy
* Výhody:
  + Rychlejší vyhledávání
  + Můžeme se dotazovat na name nebo (name,address) při stejné složitosti O(log n)
* Nevýhody:
  + Při dotazu na address dojde k sekvenčnímu průchodu se složitostí O(n). Museli bychom vytvořit jednoduchý index pro atribut address, což ale zvýší režii

**Jednoduchý Index**

* Klíčem je jeden atribut
* CREATE INDEX Producer\_name ON Producer(name);
* Dotaz: SELECT \* FROM Producer WHERE name=’prod7452’;
  + Logické přístupy: 3

## Popište plán dotazu SELECT \* FROM tabulka WHERE login = abcde, kde login je index a tabulka je heap table.

* V indexu se nalezne prvek s loginem abcde (průchod B+ stromem, rychlé, ale náhodný přístup k datům na disku/v bufferu) - UNIQUE SCAN
* Podle ROWID z indexu se načte záznam z tabulky a vyberou se všechny atributy - TABLE ACCESS BY ROWID / RID Lookup

## Popsat plán vykonání dotazu "SELECT \* FROM student WHERE login=@login", kdy login je indexem.

* V indexu (obvykle B+ strom) se najde klíč login, pokud je unikátní, jedná se o UNIQUE SCAN.
* Pokud ne, tak se najde nejnižší hodnota a poté se jde dále pomocí zřetězených odkazů v listech (RANGE SCAN).
* Takto se najde rowid záznamu v tabulce typu halda a k tomu se přistoupí a načtou se všechna data

**Kandidáti na index**

* Index se vytváří obvykle pro primární klíče a cizí klíče
* Potencionální kandidáty jsou atributy vyskytující se často v klauzuli WHERE
* Výhody – rychlejší prohledávání tabulek
* Nevýhody – každý index znamená zvýšení počtu operací při změnách v DB

**Tabulka typu Halda**

* Vytvoří se po zadání CREATE TABLE
* Stránkované persistentní pole s velikostí bloku nejčastěji 8kB
* Záznamy v tabulce nejsou nijak uspořádány
* Záznamy nejsou fyzicky mazány, pouze označeny jako smazané
* Při vkládání je záznam umístěn na první volnou pozici nebo konec pole
* Nevýhody: Neefektivní hledání (Složitost O(n))
* Výhody: Insert O(1) a využití místa

**Shlukovaná tabulka**

* Záznamy jsou seřazeny podle zvoleného klíče
* Pro implementaci je využita nějaká varianta B-stromu
* Používá se všude kde potřebujeme získat i hodnoty ostatních atributů kromě klíče
* Výhody: složitost pro základní operace O(log n)
* Nevýhody: Zhoršený výkon při INSERT – data se musí zatřídit

**Indexy**

Index umožňuje rychlé vyhledání dle klíče, ROWID okazuje na kompletní záznam v tabulce typu halda

Nejčastěji implementovány B+ -stromem

**Typy indexů:**

* Jednoduchý index
  + Klíč obsahuje jeden atribut
* Složený index
  + Index, kde je klíčem více atributů
  + Index (name, address) – dotazy jsou vykonány na atribut name nebo (name, address), ale ne na samotný address. Pro address musíme vytvořit samotný index

CREATE INDEX jmeno\_indexu ON Tabulka(klíč); složený: ON Tabulka(klíč1, klíč2)

* Vytvoří se B+ strom, každá položka obsahuje klíč a ROWID (ukazatel na záznam v tabulce typu halda)
* Index primárního klíče je často vytvořen automaticky při vytvoření tabulky

**Kandidáti na index**

* Index se vytváří pro klíče a cizí klíče
* Index je vytvořen pokud:
  + Bude používán k nalezení malého počtu záznamů
  + Pokryje jeden nebo více dotazů
* Potencionální kandidáty jsou atributy vyskytující se často v klauzuli WHERE
* Každý index znamená zvýšení počtu operací při změnách v DB – vytvoření musíme zvážit

**Plán dotazu**

* Převod dotazu do interní formy
  + Eliminujeme syntaxi jazyka dotazu
  + Interní forma je nejčastěji nějaký druh dotazovacího stromu (query tree). Dotazovací strom můžeme chápat jako reprezentaci výrazu relační algebry
  + Nahradíme pohled jeho definicí
* Převod dotazu do kanonické formy
  + Provádí se řada optimalizací
  + Dochází k odstranění rozdílů a k nalezení efektivnějšího tvaru než původní dotaz
  + Optimalizátor aplikuje transformační pravidla, která převádí výraz na ekvivalentní (výraz se stejným výsledkem), např.
    - (A JOIN B) WHERE restrikce na A transformujeme na:
    - (A WHERE restrikce na A) JOIN B
* Vygenerování plánu dotazu a výběr nejlevnějšího plánu
  + Optimalizátor vytváří množinu dotazovacích plánů
  + Každému plánu je přiřazena cena (I/O nebo CPU cost)
  + Cena je závislá na mohutnosti vstupních relací, mezivýsledků…
  + Z této množiny pak vybírá ten nejlevnější

Operace plánu dotazu:

* Table access (full) – Select \* from Tabulka;
* Table access (full) se selekcí – select \* from tabulka where name=‘pepa‘; name není indexová
* Index (unique scan) – select id from Tab Where id=50; pokud je id indexován, cost = 1
* Index (range scan)
* Table access (by index ROWID) – select name from Tab where id = 5; cost = 1
* Selekce
* Projekce
* Join – spojení se provádí zahnízděnými cykly
* Sort