Otázka 20 – A7B36DBS

Zadání

Relační DB struktury sloužící k optimalizaci dotazů - indexy, clustery, indexem organizované tabulky. (A7B36DBS)

Slovníček pojmů

- **relace** množina prvků, též množina n-tic
- **primární klíč** - sloupec (nebo sloupce), který jednoznačně určuje řádky v tabulce
- **cizí klíč** - slouží pro vyjádření vztahů, relací, mezi databázovými tabulkami. Jedná se o pole či skupinu polí, která nám umožní identifikovat, které záznamy z různých tabulek spolu navzájem souvisí.
- **index** (někdy též označovaný jako klíč - KEY) je databázová konstrukce, sloužící ke zrychlení vyhledávacích a dotazovacích procesů v databázi, definování unikátní hodnoty sloupce tabulky nebo optimalizaci fulltextového vyhledávání

Relační DB struktury sloužící k optimalizaci dotazů – indexy, clustery, indexem organizované tabulky

Indexy
Index je databázová konstrukce, sloužící ke zrychlení vyhledávání a dotazování a definování unikátních hodnot nad databázovými tabulkami. Indexová většinou vypadá jako datová struktura, která obsahuje hodnotu klíče, nad níž je index vytvořen a odkaz na záznam (řádek), ke kterému patří.
Databázové indexy (či indexsekvenční moduly) dělíme do různých druhů podle toho, co chceme při přístupech k primárním datům příslušné databázové tabulky optimalizovat. Označení druhů indexů se může různit, nejčastěji se používají tyto hlavní druhy:
Primary
Je tvořen většinou jedním (ale může být i více) sloupcem, který obsahuje primární klíč. Jedná se o speciální druh indexu, který se může v každé tabulce vyskytovat nanejvýš jednou. Je definován sloupcem (nebo sloupci), které svou hodnotou jednoznačně identifikují záznam v tabulce. Databázový stroj musí zajistit, aby nebylo možné vložit řádek se stejnými daty (ve sloupcích určujících primary index), která už v tabulce existují. Jedná se tedy o speciální případ Unique indexu. Je zažitou konvencí vytvářet primary index nad sloupcem nazvaným Id s celočíselným datovým typem.
Unique
Je velice podobný primary indexu (primary index je vlastně podtypem unique) s tím rozdílem, že se v tabulce může vyskytovat vícekrát.
Index (secondary, vedlejší)

Definicí jednoho či více indexů tohoto typu v tabulce zajišťujeme optimalizaci vyhledávání podle dalších sloupců, mimo primární nebo unikátní indexy. Databázový server vytvoří a nadále udržuje vnitřní konstrukci odkazů na řádky tabulky, jež poskytuje uspořádání podle příslušných hodnot ve sloupci, k němuž je index logicky vázán (podle hodnot sekundárního klíče). Udržování takto uspořádané konstrukce urychluje vyhledávání záznamů v databázi (je možno použít některé matematické interpolační numerické metody), logické či fyzické řazení záznamů jakož i jiné další datové operace s tabulkou, jež se mají provést na podmnožině záznamů z tabulky vymezené podmínkou položenou na hodnoty v sekundárním klíči. Na rozdíl od předchozích indexů PRIMARY a UNIQUE lze do tabulky vkládat záznamy, které nejsou v sekundárním indexu unikátní. U některých databázových systémů se může jednat i o sloupce tzv. fiktivní, tedy sloupce odvozené respektive vypočtené z hodnot sloupců fyzických resp. uložených.

Full-text

Používá se pro optimalizaci full-textového vyhledávání v textových sloupcích. Implementace závisí na zvoleném databázovém stroji, jednou z metod je např. udržování statistiky slov, které se vyskytují ve sloupcích s full-text indexem.

Indexy - implementace

Bitmap index

Tento druh indexu je vhodný pro sloupce, kde se vyskytuje relativně málo hodnot, ale ve velkém množství opakování. Krásným příkladem je např. tabulka uživatelů, o kterých si chceme pamatovat jejich pohlaví. Takový sloupec nabývá buď hodnot muž x žena.

Pro každou z těchto dvou hodnot je vytvořena bitmapa (pole bitů), která obsahuje informace o tom, které řádky obsahují tyto hodnoty a její nad ní možné provádět bitové operace. Výhodou tohoto přístupu je relativně malá datová náročnost na uchovávání indexu, protože každý záznam v tabulce zabere 1bit * počet hodnot, které sloupec nabývá. Ukázkový index pohlaví by tak v tabulce s více než osmi miliony uživatelů zabral 2 MiB. I přes relativně malou náročnost na data je možné index dále komprimovat, typicky pomocí nějaké verze run-length kódování, které jsou velice nenáročné na kompresi/dekompresi. Zde se nejčastěji používá Byte-aligned Bitmap Code (BBC) a Word-aligned Hybrid Code (WAH), které navíc umožňují provádět bitové operace přímo nad komprimovanými daty.

BBC uchovává svoje bitmapy v bytech, zatímco WAH v wordech, které lépe vyhovují dnešním procesorům a dosahují větších výkonu. Dle zveřejněných testů by pak WAH mělo (na reálných i syntetických datech) zabírat zhruba o 50% více datového prostoru, ale dosahovat až 12x rychlejších výsledků než BBC. Obě metody jsou však slině závislé na seřazení dat v tabulce. Jednoduché lexikální seřazení může zmenšit index až 9x a výrazně zvýšit výkon. Čím více dat tabulka obsahuje, tím více roste nutnost správného řazení

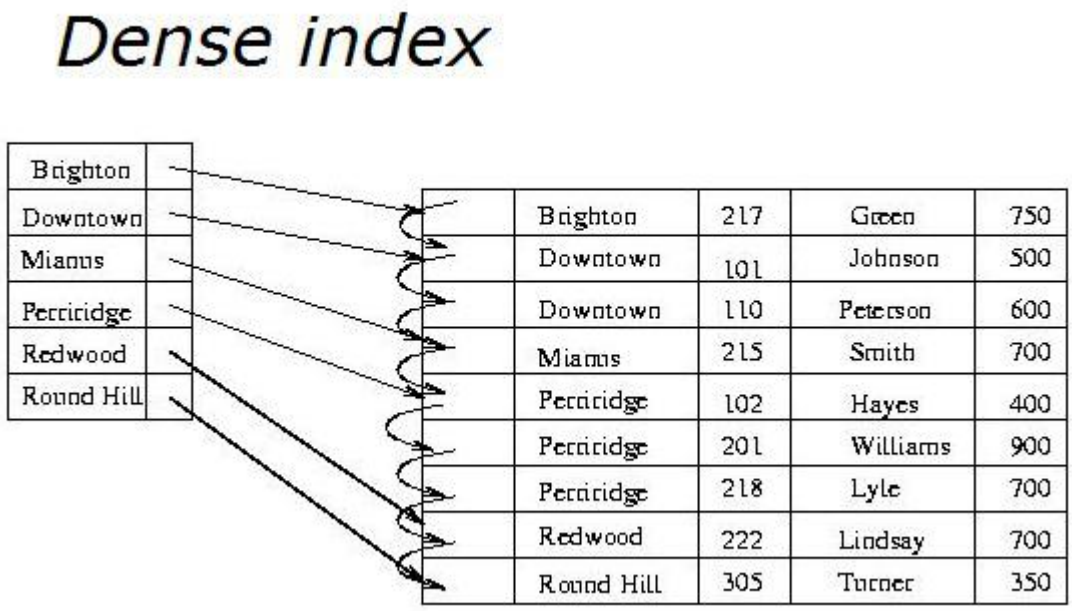
Bitmap index

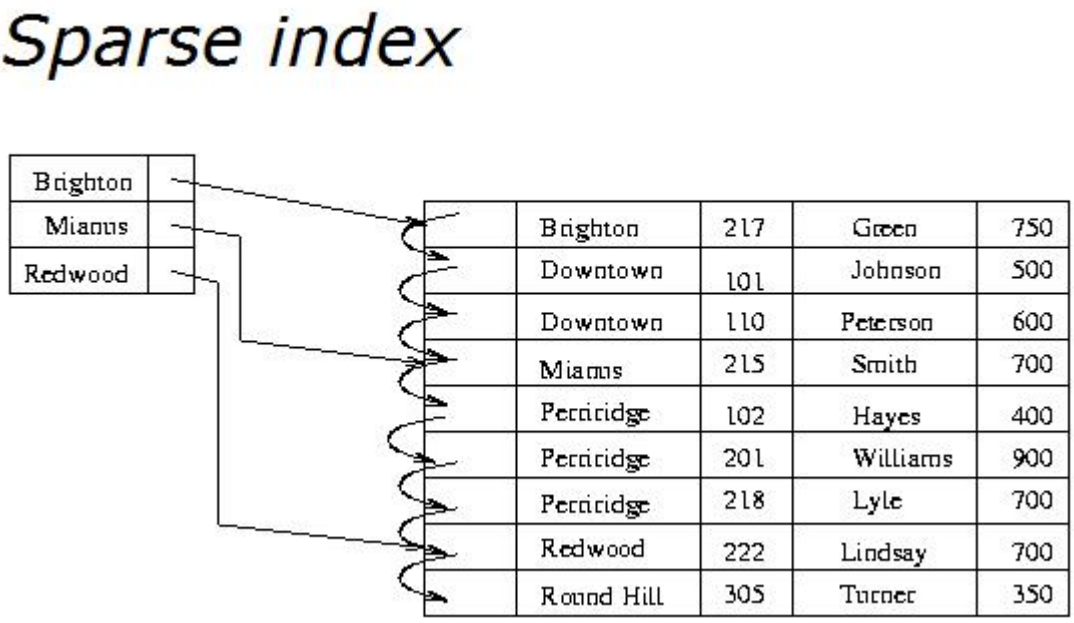
Identifier	Gender	Bitmaps	
		F	M
1	Female	1	0
2	Male	0	1
3	Male	0	1

Dense a Sparse index

Dense index obsahuje všechny hodnoty klíče, které se v tabulce vyskytují spolu s odkazem na relevantní záznamy. Při hledání se pak najde v indexu požadovaný klíč a z něho se přečte umístění požadovaných záznamů.
Sparse index neuchovává záznamy o všech klíčích, ale pouze o určité podmnožině dostupných klíčů. Vyhledávání probíhá tak, že se najde největší klíč, který menší nebo roven tomu hledanému a od tohoto záznamu se sekvenčně prohledává tabulka, než jsou nalezeny požadované záznamy.
Dense indexy jsou obecně o něco rychlejší, ale zabírají více diskového prostoru, než sparse

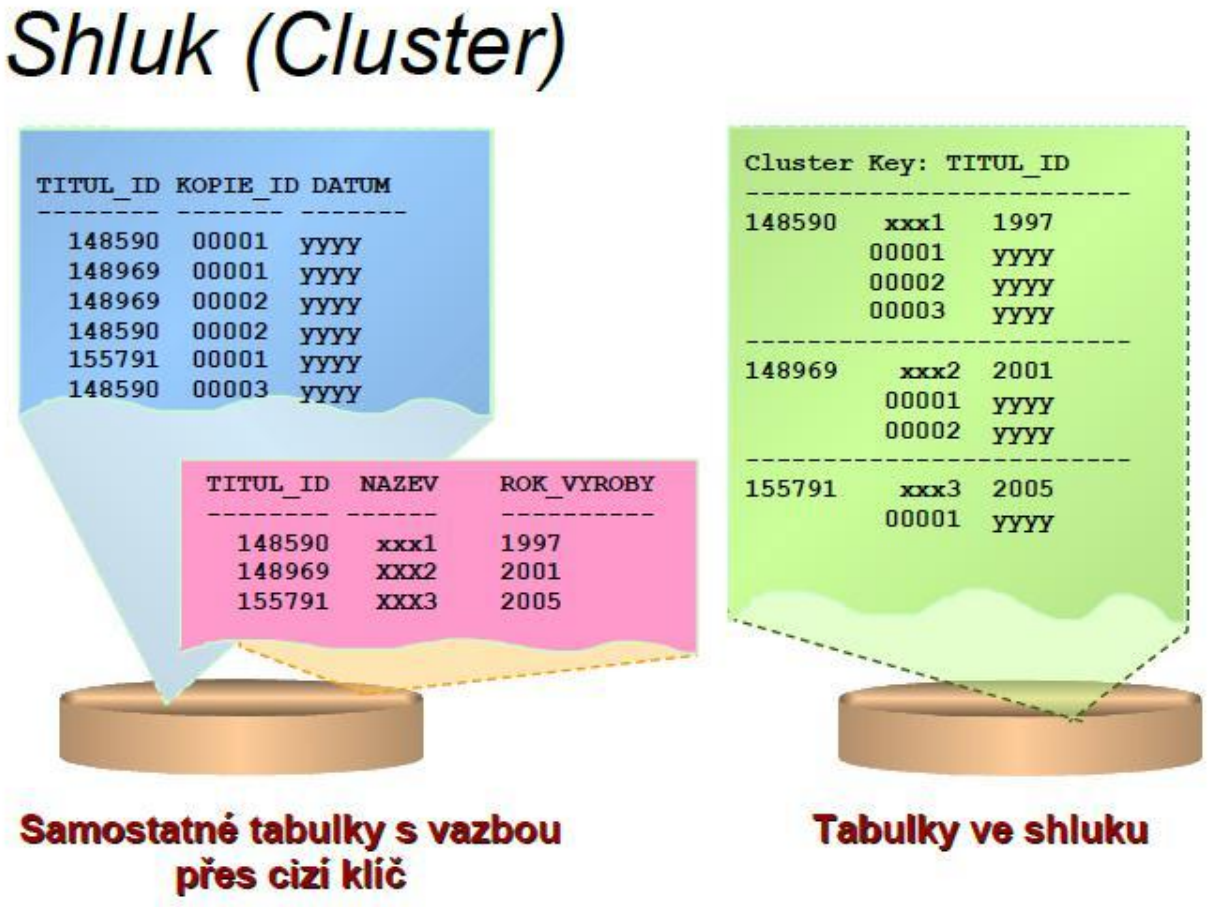
indexy,ale mají větší režii pro přidávání a mazání.





Clustery
V rámci optimalizace rychlosti dotazů umožňují některé databázové enginy ukládat tabulky do tzv. clusterů. Jedná se o formu fyzického uložení na disk, kdy jsou tabulky se souvisejícími daty (např.

spojení přes reference) uloženy u sebe. Tím se zrychlí např. dotazy tyto JOIN na tyto tabulky. Pro uživatele (tedy aplikačního programátora) se však nadále chovají zcela transparentně jako normální tabulky



Indexem organizované tabulky

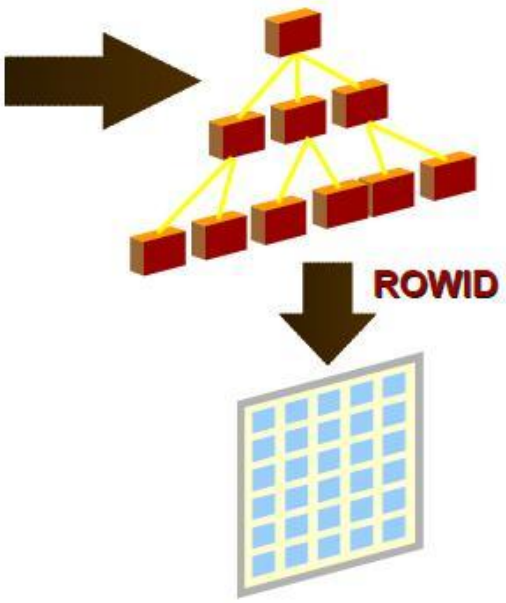
Standardní tabulka ukládají řádky do svých datových bloků přibližně následovně (může se lišit dle implementace): Každý řádek má neměnnou pozici na datovém úložišti. V momentě, kdy je dostane svoji pozici a na té zůstane, dokud nebude smazán a to i v případě, že bude např. přesunut a doplněn o další data - v takovém případě na původním místě alespoň část dat, podle kterých systém dokáže dohledat zbytek. Index v normální tabulce obsahuje data a rowid (interní identifikátor řádku), podle kterého se dohledá správný datový blok.

V indexově organizované tabulce jsou data na úložišti organizována jako B-tree index postavený z primárních klíčů. V takovéto tabulce nemá řádek stejnou pozici a v čase se může přesouvat, aby bylo zachováno správné seřazení B-tree. Výhody

- Rychlý přístup k datům při dotazech na přesnou hodnotu nebo rozsah primárního klíče (např. WHERE id = 5, WHERE BETWEEN 10 AND 20, …)
- Pro 24×7 aplikace: Při reorganizaci dat na datovém úložišti (např. defragmentace) není třeba znovu vytvářet index

Indexově organizovaná tabulka

Heap tabulka s idexem



Indexově organizovaná tabulka