Journal of the Technical University of Gabrovo, Vol. 53'2016 (65-68)

ИЗВЕСТИЯ
на Технически университет
Габрово

ТЯСНО МЯСТО ПРИ ВЪНШНИТЕ ИНДЕКСНИ СТРУКТУРИ ОТ ДАННИ

BOTTLENECK FOR EXTERNAL INDEX DATA STRUCTURES

Iliya Tronkov*
STS Soft

Статията е постъпила на 04 април 2016 г.; приета за отпечатване на 12 май 2016 г.

Abstract

One of the most important characteristics of the data base management system (DBMS) is the index maintenance. Thanks to them it is possible to acquire certain knowledge from the information stored in the data base, without the necessity to go round all the data. Therefore the complete performance of the DMBS depends, on a great extent, on the indexing technology, which is used. As a result of the physical limitation of the external storage devices, one of the most important tests for each index data structure is the indexing of the incoming records with random keys. This article shows such comparison test of the speed among the different classes indexing technologies.

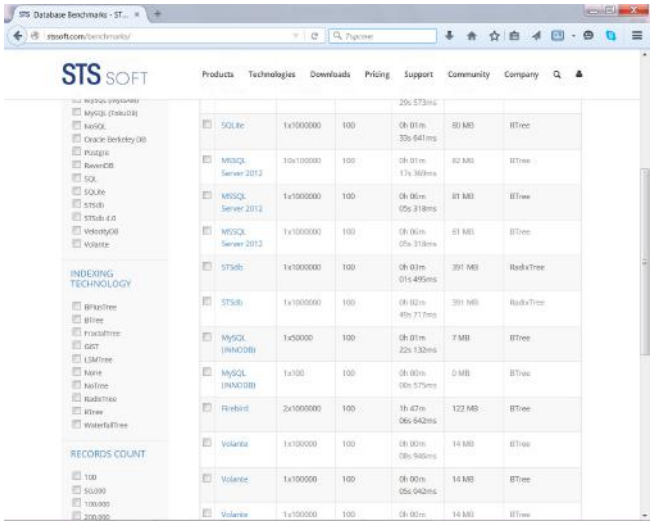
Keywords: Database; Benchmark; Index Data Structure; Indexing; Comparative Analysis.

ВЪВЕДЕНИЕ

От една страна информацията, съдържаща се в данните, има многомерен характер. От друга страна външните запаметяващи устройства (ВЗУ), като HDD [1], SSD [2] и др. имат само едно измерение за данни. Следователно, абстрактно погледнато, индексът е изображение на многомерно пространство в едномерно. Освен това последователността на постъпване на данните, в общия случай, не съвпада с наредбата на индекса. Това налага данните да се разполагат в различни физически блокове на ВЗУ. Проблемът се състои в това, че при ВЗУ произволния достъп до данните е в пъти по-бавен от последователния достъп, т.е. ако данните са физически разпокъсани на части (не са разположени в последователни блокове на ВЗУ), общото време за тяхното четене/запис се увеличава с (времето за произволен достъп [3] на ВЗУ) ✕ (броя на частите). Всичко това създава **тясно място** в цялата СУБД и увеличава трансфера на нерелевантни данни, което допълнително утежнява процесите в СУБД. Следователно, за да има практическа стойност, индексната структура от данни трябва да отговаря на определени изисквания, наложени от физическите ограничения на външните запаметяващи устройства. Всичко това означава, че индексирането на записи с произволни ключове е един от най-тежките тестове за всяка индексна структура от данни. Именно това съображение е водещо при определяне на тестовите данни и видовете тестове в настоящата статия.

СРЕДА ЗА ТЕСТОВЕТЕ

Всички тестове са направени с публично достъпното приложение с отворен код *Database Benchmark 3.0*, проектирано и разработено под ръководството на автора. Приложението може да бъде свалено от http://sourceforge.net/projects/benchmark/. Към него е разработен и сайт за сравнителен анализ на резултатите получени с него (*Фигура 1*).

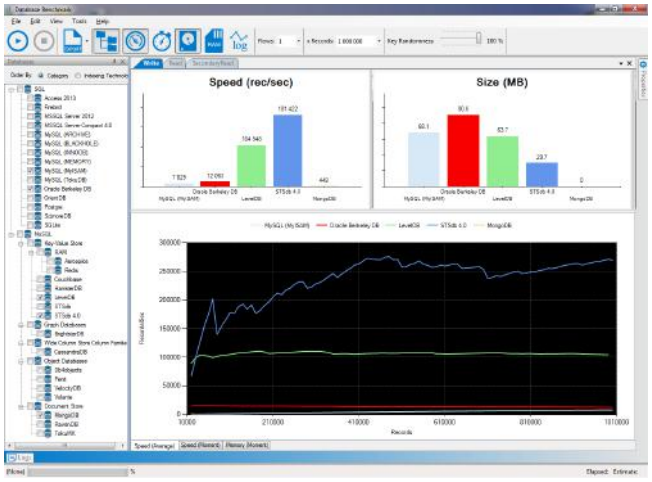


Фиг. 1. Сайт за сравнителен анализ на тестовите резултати получени от Database Benchmark (http://stssoft.com/products/database-benchmark/)

Database Benchmark е едно от най-полулярните приложения за тест и оценка на производителността на индексните структури от данни. То разполага с интуитивен и гъвкав потребителски интерфейс и богат набор от настройки за конфигуриране на тестовете (*Фигура 2*). Към днешна дата Database Benchmark има десетки хиляди сваляния и излиза на първо място в Google търсачката.

Основната задача на Database Benchmark е да предоставя обективни сравнения на бързодействието на различните индексиращи технологии, без значение от техният вид и специфични характеристики.

* *Тел.: 0887308701; e-mail: tronkov@gmail.com*



Фиг. 2. Потребителски интерфейс на Database Benchmark 3.0 с направени тестове

ВИДОВЕ ТЕСТОВЕ

Тестове са така проектирани, че да показват скоростите на индексиране на отделните технологии. За целта всеки от тестовете включва три етапа: 1) добавяне на записи със случайни ключове (Random Insert тест); 2) последователно четене на всички записи във възходящ ред по техния ключ (First Read тест); 3) повторно последователно четене на всички записи (Secondary Read тест).

Първото четене е необходимо, за да се направи приблизителна оценка колко време е необходимо на всяка от тестваните индексни структури от данни да индексира множеството си от записи. Затова непосредствено след Insert теста се изчитат (последователно спрямо ключа) всички налични записи. Това принуждава съответния индекс – дървовиден или недървовиден, да приведе структурата си в такава, в която записите са подредени в нарастващ ред. От повторното четене се вижда какво е забавянето от индексната структура от данни при извличане на данни спрямо линеен файл (който е и на теория и на практика най-бърз при последователно четене).

ТЕСТОВИ ДАННИ

За всички тестове се генерират данни със следната структура: 1) **ключ** - цяло число тип long (8 байта); 2) **запис** - съставен от няколко прости полета (~52 байта). Броят на записите за всеки от тестовете е $N=10^8$. За генериране на случайните ключове е използван псевдослучаен генератор. Генерираните последователни ключове са със стойности от *0* до *N-1*. За генериране на стойностите на записите е използван random-walk алгоритъм.

Тъй като данните за теста не винаги се побират в оперативната памет има два подхода:

1) записите да се генерират предварително и да се запишат във файл и по време на теста да се четат от този файл. Очевидно ще е необходимо да се предвиди и осигури място за този входен файл. При този подход, ако файлът със записите се намира на същото физическо устройство, където е и тестваната индексна структура от данни, четенето от файла ще предизвиква I/O операции, които ще интерферират с I/O операциите на индексната структура от данни. Следователно файлът трябва да се намира на отделно външно запаметяващо устройство. Това от своя страна налага изискването пропускателната способност за четене на това устройство да е по-голяма от пропускателната способност на устройството на което е разположена тестваната индексна структура от данни, в противен случай тестовите данни ще ограничат скоростта на Insert. Всичко това поставя силни ограничения за компютърните конфигурации на които може да се изпълняват тестовете. Дори и предходните изисквания да са изпълнени, поради спецификата на отделните бази от данни и техните индексни структури от данни съществува не толкова очевиден недостатък при този подход. Различните бази от данни и техните индексни структури от данни кешират и буферират (в оперативната памет) на различни нива записите. Кешът обикновено се настройва да използва всичката налична оперативна памет. Но пропускателната способност на оперативната памет е десетки хиляди пъти по-голяма от тази на външните запаметяващи устройства. Следователно този подход на провеждане на тестовете ще създаде сериозно ограничение за тези индексни структури от данни, които успяват да се справят с големи потоци от данни (те няма да могат да покажат пълния си потенциал). При този подход съществува и още едно ограничение свързано с тестването на индексните структури от данни в ситуация при която добавянето на записи се извършва от много нишки. Това ще означава да се създадат толкова входни файла със записи, колкото нишки ще има. От своя страна всеки от тези файлове трябва да бъде на отделно физическо устройство, за да не си пречат при четене (както бе обяснено по-нагоре).

2) записите да се генерират в реално време. При този подход не се минава през междинни файлове следователно ограниченията които налага подход 1) не съществуват. Времето необходимо за генериране на един запис е съизмеримо с това записът да бъде десериализиран от байтов поток, както е в подход 1). Стойностите на ключовете при теста с последователни ключове ще бъдат едни и същи и при двата подхода. Разликата идва при тестовете с произволни ключове. При този подход при всеки тест ще се генерира различна редица от ключове, която ще следва определено вероятностно разпределение (в случая е равномерно разпределение). При тестове с малко на брой записи това би имало някакво значение, но при тестове с над 10^8 записи всякакви флуктации в резултатите предизвикани от различните редици от ключове ще бъдат заличени. При толкова много записи продължителността на теста е часове, през това време компютърът е натоварен до своя предел. Това означава, че операционната система и файловата система нямат възможност да отлагат задачите си за по-късно (както се прави при ненатоварена система). Това променя цялото поведение на операционната система и файловата система. Следователно съществено влияние оказват не разликите в редицата от ключове, а външните фактори. Въпреки това, ако се държи и тези фактори да бъдат отчетени то всеки от тестовете може да бъде извършен по няколко пъти, тогава според закона за големите числа [4] и флуктациите предизвикани от тези фактори ще бъдат заличени. При този подход не съществува проблем в ситуация при която добавянето на записи се извършва от много нишки.

Вижда се, че подход 1) въвежда твърде много ограничения и проблеми, а единственото нещо което дава е, че тестовете с произволни ключове ще се извършват с еднакви редици от ключове. От разглежданията в под-

ход 2) се вижда, че това не е съществено и няма да компрометира по никакъв начин резултатите. Освен това подход 2) ще позволи тестовете да се извършват на практически всякакви компютърни конфигурации. Затова в Database Benchmark е избран и се използва именно подход 2).

ИЗМЕРВАНИ ПАРАМЕТРИ

При провеждане на тестовете се измерват следните параметри: 1) Скорост на добавяне на всички записи (*записи/секунда*). На графиките по-долу е показано как се променя скоростта на добавяне на нови записи с увеличаване на наличните записи в СУБД; 2) Скорост на последователно четене на всички записи, подредени по техния ключ (*записи/секунда*); 3) Размер на базата от данни, която създават упоменатите СУБД (*MB*); 4) Общо време за целия тест на запис, четене и повторно четене (*часове:минути:секунди*).

ВКЛЮЧЕНИ В ТЕСТОВЕТЕ ИНДЕКСИРАЩИ ТЕХНОЛОГИИ И НАСТРОЙКИ НА БАЗИТЕ ОТ ДАННИ

Тестовете са проведени със следните индексиращи технологии B-tree [5-7], LSM-tree [8], FractalIndex Tree [9-11] и NoTree [12]. Тези технологии са представители на различните класове индексни структури от данни. За тестовете са използвани реализации на тези индекси като част от следните продукти:

База от данни	***Индексираща технология***	***Фирма***
STSdb 5.0 alpha (prototype)	NoTree	STS Soft
Oracle Berkeley DB 12c Release 1 Library Version 12.1.6.0	B-tree	Oracle
MySQL с MyISAM 5.7	B-tree	Oracle
MySQL с TokuDB TokuDB 7.5.5 Community Edition за MySQL 5.5.41	FractalIndex Tree	Tokutek/ Percona
MongoDB 3.0 с MMAPv1	B-tree	10gen
MongoDB 3.0 с WiredTiger	LSM-tree	10gen
LevelDB 1.16	LSM-tree	Google

При всички тестове тестовото приложение (*клиентът*) и базата от данни (*сървърът*) работят на една и съща машина (*localhost*). В тестовете участват както сървърни, така и вградени бази от данни. По време на тестовете не работят други приложения. Настройките на отделните бази от данни позволяват използване на цялата налична RAM памет и ресурси.

STSdb, BerkeleyDB, MySQL, MongoDB и LevelDB работят на указаните Windows операционни системи. TokuDB работи на Ubuntu 14.04.

За минимизация на времената за изпращане на данните (при insert тестовете) при сървърните решения е използвано окрупняване на заявките. За SQL-базираните решения са използвани *multi insert queries* в комбинация с *prepared statements*. За MongoDB-решенията са използван *batch insert* заявки с *write concern* = 1 (*acknowledged*).

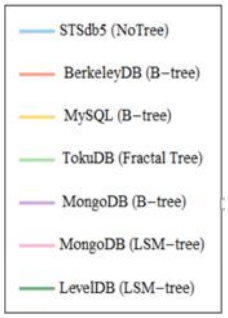
Използваните бази от данни са писани на различни езици, работят на различни платформи и като цяло имат различна философия. И въпреки, че са тествани при един и същ хардуер и с едни и същи типове данни, е ясно, че не могат да бъдат постигнати напълно унифицирани начални условия.

КОМПЮТЪРНА КОНФИГУРАЦИЯ

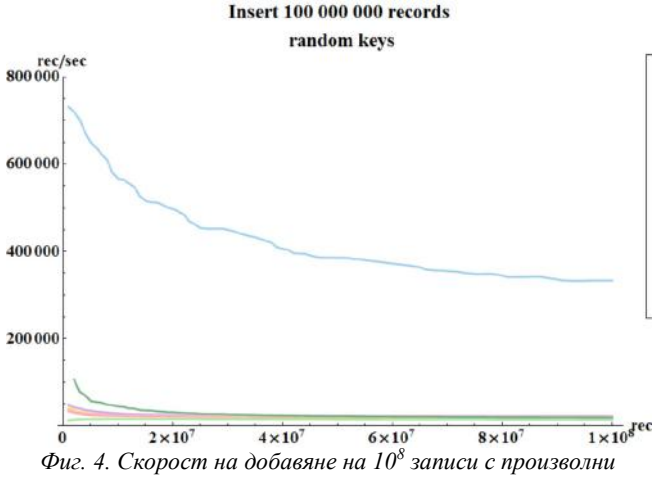
За провеждане на тестовете е използвана следната компютърна конфигурация: 1) CPU - 2 x Intel Xeon E5620 2.4 GHz с 12MB L3 кеш; 2) RAM - 40GB DDR3-1600 MHz; 3) 3 x 256 GB SSD в RAID0; 4) OS - Windows Server 2008 R2 Enterprise Service Pack 1 64 bit с NTFS / Ubuntu 14.04 64 bit с EXT4.

РЕЗУЛТАТИ

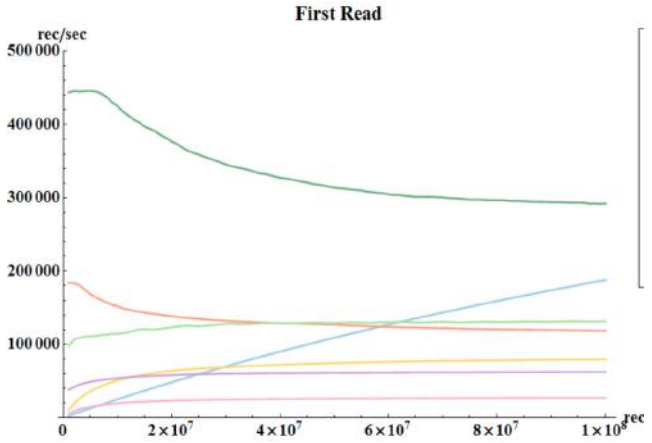
Резултатите от проведените тестове са както следва: 1) скорост на добавяне на 10^8 записа с произволни ключове *Фигура 3*; 2) скорост на първо последователно четене на всички записи *Фигура 4*; 3) скорост на повторно последователно четене на всички записи *Фигура 5*; 4) таблица 1 обобщена информация - размер на базите от данни и сумарно време за изпълнение на пълния тест, за всяка от базите от данни.



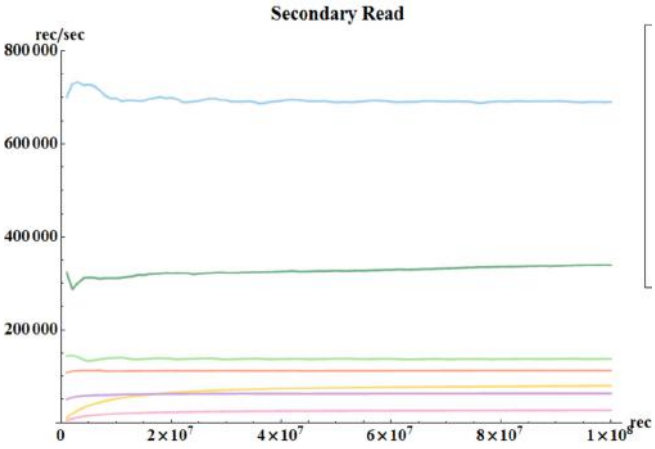
Фиг. 3. Легенда за фигури 4, 5 и 6



Фиг. 4. Скорост на добавяне на 10^8 записи с произволни ключове



Фиг. 5. Скорост на първо последователно четене на всички записи



Фиг. 6. Скорост на повторо последователно четене на всички записи

Таблица 1. Обобщена информация

	Insert (rec/s)	First Read (rec/s)	Secondary Read (rec/s)	Size (MB)	Total Time (h:min:s)
STSdb5	333 204	187 935	689 933	6 219	00:16:17
BerkeleyDB	19 949	118 727	112 478	8 278	01:52:24
MySQL	20 397	79 734	80 019	6 804	02:03:26
TokuDB	14 537	131 464	137 544	4 415	02:19:09
MongoDB (B-tree)	21 869	62 355	63 281	27 273	02:09:16
MongoDB (LSM-tree)	21 556	27 132	28 623	16 241	03:16:58
LevelDB	19 084	292 113	339 183	6 188	01:37:57

Направените тестове нямат за цел да сравняват скоростите на базите от данни, а да направят, до колкото е възможно, сравнителни тестове на скоростта, с която индексират работещите в тях индексиращи технологии.

АНАЛИЗ

В направените тестове общия размер на данните е по-малък от оперативната памет и всички индексиращи технологии се възползват ефективно от кеширането (както собствения кеш от блокове, така и кеша на файловата система) и успяват да се справят със задачата за разумно време.

Тестовите резултати с 10^8 записи показват, че NoTree индексната структура от данни се представя в пъти по-добре спрямо останалите индескни структури от данни. Това се дължи на философията, която стои в основата на NoTree индексирането - всичко в тази стуктура от данни е проектирано и съобразено с тясното място дефинирано във въведението.

ЗАКЛЮЧЕНИЕ

От резултатите направени с 10^8 записи се вижда, че производителността на индексните структури от данни съществено зависи и от конкретната реализация. Тъй като при тези тестове общия размер на данните е по-малък от размера на оперативната памет всяка една от индексиращите технологии успява да завърши тестовете.

Въпреки, че тестовете са извършени на машина с SSD (при SSD seek time е десетки пъти по-малък от seek time на HDD) I/O операциите за произволен достъп до външното запаметяващо устройство отново се явяват тясно място за външните индексни структури от данни.

ЛИТЕРАТУРА

[1] Hard disk drive, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Hard_disk_drive. [Accessed 30 06 2015].

[2] Solid-state drive, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Solid-state_drive. [Accessed 30 06 2015].

[3] Hard disk drive performance characteristics, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics. [Accessed 01 07 2015].

[4] D. P. Bertsekas and J. N. Tsitsiklis, Introduction to Probability, Massachusetts Institute of Technology, 2000.

[5] R. Bayer and E. McCreight, Organization and Maintenance of Large Ordered Indexes, *Acta Informatica,* vol. 1, p. 173–189, 1972.

[6] D. Comer, The Ubiquitous B-Tree, *Computing Surveys,* vol. 11 No 2, June 1979.

[7] D. E. Knuth., Sorting and Searching, volume 3 of The Art of Computer Programming, vol. 3, Massachusetts: Addison-Wesley, 1973.

[8] P. O'Neil, E. Cheng, D. Gawlick and E. O'Neil, The log-structured merge-tree (LSM-tree), *Acta Informatica,* vol. 33, pp. 351-385, June 1996.

[9] M. Farach-Colton, www.tokutek.com, tokutek, 11 2012. [Online]. Available: http://www.tokutek.com/wp-content/uploads/2012/11/Fractal-Tree-Technology-and-the-Art-of-Indexing.pdf. [Accessed 2 9 2014].

[10] M. A. Bender, M. Farach-Colton, J. T. Fineman, B. C. K. Y. Fogel and J. Nelson, Cache-oblivious streaming B-trees., *In Proceedings of the Nineteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA),* pp. 81-92, June 9–11 2007.

[11] B. C. Kuszmaul, A Comparison of Fractal Trees to Log-Structured Merge (LSM) Trees., Tokutek, 22 4 2014. [Online]. Available: http://forms.tokutek.com/acton/attachment/6118/f-0039/1/-/-/-/-/lsm-vs-fractal.pdf. [Accessed 10 3 2015].

[12] I. Tronkov and A. Todorov, METHOD OF DATA INDEXING AND SORTING. International Patent PCT 112008, 5 8 2015.