Implementation of databases

Kartik vishwakarma
2017csm1001

Problem: 1 solution :
Dataset are created using file named "dataset.cpp". A directory named "dataset" created and all records are stored there.
Data are generate using "rand()" function.
According to convection each file contain 300 records and pointer to next file maintained here.

Query 1: Select SUM(sale amount) From SALES_TABLE Where <condition>

Cost : number of disk block accessed
Time taken to execute query

Experiments 1-a:
Theta = 3000, # ones : 1 lakhs ones in Bf

Indexing type	sum	# disk access	Execution time (in ms)
No indexing	14,96,40,604	10,00,000	2,105
Row id representation	14,96,40,604	6,000	369
Bit array representation	14,96,40,604	96,000	43,214
Bit slice representation	14,96,40,604	512	349

Using no indexing:
Evaluating Query 1 using no indexing on 1 lakhs dataset takes 10,00,000 block access, i.e. for each records it access one block and execution time about 2105 milliseconds.

This is one of native way to access records, but execution cost it high.

Using row id representation:

It tooks 6,000 disk access and 369 execution cost, much better than no indexing in terms of both disk access and execution cost. Here this performance come because took up table, when a row id results according to query, i first check in look table whether it appears first time or not, if it appears first time add into look up table and for all entries correspond to that sale amount check for all row that set in Bf. Increment counter of that sale amount and reset that row id in Bf array.

Using bit array representation:

It has fixed bit string length of 10 lakhs, irrespective of data distribution, since theta is quite high i.e varies from 0 to 3000, there are maximum 300 unique sale amount values. Each value is of length 10 lakh means (10,00,000/32000) 32 file for each sale amount with file size of 32,000 bit string. So total file is 32 x 3,000 = 96,000 files **(around 3 GB).**

This is huge data with remain constant irrespective of Bf and records. Because of such huge data is it took long time 43,214 milli second to process. Also irrespective of need to check all 96,000 file result very high disk access.

Bit slice representation:

Here i am using 16-bit representation each for 10 lakh records i.e. each bit takes 32 file each of size 32,000. Total file size of 16 x 32 = 512 files always same irrespective of data distribution. Because small no of file and bit representation it takes 512 disk access and 349 ms to execute, perform best among all above other techniques.

Experiments 1-b:

Theta = 50, # ones : 1 lakhs ones in Bf

Indexing type	sum	# disk access	Execution time (in ms)
No indexing	25,53,650	10,00,000	2,058
Row id representation	25,53,650	1,073	315

Bit array representation	25,53,650	1,600	9,823
Bit slice representation	25,53,650	512	263

Using no indexing:

Irrespective of theta, since record size remain same, no improvement in this approach since disk access remain same, also execution time remain close to experiment-1-a.

Using row id representation:

Because theta value reduce to 50 only at maximum 50 sale amount value will be each of best case size of 2,00,000 with 1,000 files each size 10,000 records, in my case total 1,600 file are create show sale amount are not uniformly distributed.
But unlike exp-1-a where 3,000 files here i have almost half of file with reduces 1,073 much less than previously 6,000 disk access.
But execution time to 315 remain close to previous, this show that cost of row id heavily depend on theta value (unique records) lesser unique record lesser query cost.

Using bit array representation:

Again with respect to theta number of file goes down to 1,600 cause lower disk access but each records of length 10,00,000 tooks long time, but much less compared to exp-1-a,
Like rowid, bit array also heavily affected because of theta values. (lesser unique value, lesser cost.)

Bit slice representation:

Its fixed and small number of file irrespective of data distribution makes it most stable among all above, as its disk access it remains same and slight difference in execution timing.

Experiments 2-a:

Theta = 3000, # ones : 10,000 ones in Bf

Indexing type	sum	# disk access	Execution time (in ms)
No indexing	1,48,50,691	10,000	1,330
Row id representation	1,48,50,691	5,788	381
Bit array representation	1,48,50,691	96,000	36,732
Bit slice representation	1,48,50,691	512	324

Using no indexing:

Varying Bf set bit cause number of disk / files need to access lesser set bit lesser file required to access. This trade results lower in #disk access and lower execution time also.

Using row id representation:

disk access lower with lower set bit but very small difference hence # disk access close to exp-1 same hold true for execution time also.

Using bit array representation:

Although #disk access remain same, but surprisingly Execution cost slightly higher or lower based on moment it runs. Its run time goes upto 3,74,390 ms results highly unstable technique in such dataset.

Bit slice representation:

LIke previous experiments its have remain same disk access and execution time almost constant, irrespective of data distribution.

Experiments 2-b:

Theta = 50, # ones : 10,000 ones in Bf

Indexing type	sum	# disk access	Execution time (in

			ms)
No indexing	2,57,574	10,000	1,330
Row id representation	2,57,574	1,073	328
Bit array representation	2,57,574	1,600	9,627
Bit slice representation	2,57,574	512	226

Using no indexing:

Irrespective of query it scans all file make it high in disk access and execution cost.

Using row id representation:

Disk access and time remain same as in exp-1.

Using bit array representation:

Disk access and time remain same as in exp-1.

Bit slice representation:

Maintaining it trends, having lowest #disk access and execution time.

Experiments 3-a:

Theta = 3000, # ones: 100 ones in Bf

Indexing type	sum	# disk access	Execution time (in ms)
No indexing	1,42,896	100	4
Row id representation	1,42,896	194	213
Bit array representation	1,42,896	60,000	30,404

Bit slice representation	1,42,896	320	219

Using no indexing:

Unlike previous case here it outperform among all other techniques in both #disk access and execution time. This because of very sparse Bf value very less disk required to access and also very less in execution timing.

Using row id representation:

Preserving no indexing property i.e. very sparse Bf bit string, very few entries would be and hence few number of files result lower number of disk access also lower execution time but much higher compare to no indexing.

Using bit array representation:

It has remain high disk access because large #files and moving over each file cause higher execution timing.

Bit slice representation:

Huge sparse nature of Bf only few relevant file will be there hence its #disk access and
Execution time also reduces.

Experiment 3-b:

Theta - 50, #ones: 100 ones in Bf

Indexing type	sum	# disk access	Execution time (in ms)
No indexing	2,685	100	4
Row id representation	2,685	944	314
Bit array representation	2,685	1,200	9,794
Bit slice representation	2,685	384	180

Using no indexing:

Because very sparse ones in Bf bit string very few records need to be search and resides only few number of blocks required few disk access and execution timing.

Using row id representation:

Here cost depends on uniquely sale amount in Bf bit string because in worse case all 100 one bit access to 100 different file with one entry per file, but unnecessary look up at other row id of same sale amount, since it does know how many bit are set per sale_amount and at what index.

Using bit array representation:

Despite of sparse nature one may randomly distributed to various file, so this may lead to high #disk access and execution time. Its cost depends on 1's bits distribution of Bf.

Bit slice representation:

It have lower cost than bitmap but because of looks up at each 16 bit each bit of 32 file increases its #disk access and execution cost compared to No indexing.

Conclusion:

Over all one large theta and small theta with high set bits in Bf, **bit slice** it preferable because of it almost constant #disk access and execution time.

While one large theta and small theta with less set bits in Bf, **no indexing** is preferable.

While in row id and bit array representation, rowid is always prefered in such queries.

Bit array may prefer in Query range queries.

Select records From SALES_TABLE Where sale_amount = "500"

While in aggregate queries:

Bit slice and no indexing is prefered based on explained above.