|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| final design | **Course:** | **Advance Database Systems** | **Course Code:** | **CS451** |
| **Program:** | **BS(Computer Science)** | **Semester:** | **Spring 2019** |
| **Out Date:** | **27-Mar-2019** | **Total Marks:** |  |
| **Due Date:** | **Thu 04-Apr-2019 *(Start of Class)*** | **Weight:** |  |
| **Section** | **CS** | **Page(s):** | **2** |
| **Assignment:** | **3 (Indexing Structures for Files)** |  |  |
|  |  | | | |

*Instructions:*

* **This assignment is an individual assignment.**
* **Use proper assignment papers for solving your assignment questions. Assignment done on diary pages, register pages, rough pages will not be credited.**
* **Use any valid assumption where needed.**

Consider the following scenario:

A block pointer is P = 6 bytes long and a record pointer is PR = 7 bytes long. A file has above records of fixed length*.* Each record has the following fields: CompanyName (35 bytes), CompanyId (8 bytes), MainWarehouse (9 bytes), Address (41 bytes), Phone (9 bytes), CreationDate (8 bytes), Branches (1 byte), Code (5 bytes), Assets (8 bytes, real number). An additional byte is used as a deletion marker.

**Q1.** Assume Block Size **B = 5041 b**ytes and File Records **r = 10m**illion

**Q2.** AssumeBlock Size **B = 9216 b**ytes and File Records **r = 10b**illion

Attempt all following parts first using **Question1** statistics and then **Question2** statistics.

**Q1 Solution.**

1. Calculate the record size R in bytes.

**Ans:** 35 + 8 + 9 + 41 + 9 + 8 + 1 + 5 + 8 + 1 =125

1. Calculate the blocking factor bfr and the number of file blocks b, assuming an un-spanned organization.

**Ans:**

bfr = floor(B/R) = floor (5041 / 125) = floor (40.3) = **40 (i.e. record per block)**

b = ceiling(r/bfr) = ceiling (10,000,000/40) = **250,000 (i.e. total blocks)**

1. Suppose that the file is *ordered* by the key field CompanyId and we want to construct a *primary* index on CompanyId. Calculate (i) the index blocking factor bfri (which is also the index fan-out, *fo);* (ii) the number of first-level index entries (r1) and the number of first-level index blocks (b1); (iii) the number of levels needed if we make it into a multi-level index (x); (iv) the total number of blocks required by the multi-level index (bi); and (v) the number of block accesses needed to search for and retrieve a record from the file given its CompanyId value, using the primary index.

**Ans:**

i)

Ri = CompanyIdSize + Pb = 8+6 = 14 bytes (index entry size)

bfri = floor(B/Ri) = floor (5041/14) = floor(360.07) = 360 (index entries per block)

ii)

r1 = number of file blocks = b = 250,000 entries.

b1 = ceiling (r1/bfri) = ceiling (250,000/360) = ceiling (694.4) = 695 blocks.

iii)

r2 = number of 1st level index blocks = b1 = 695 entries.

b2 = ceiling (r2/bfri) = ceiling (695/360) = ceiling (1.93) = 2 blocks

r3 = number of 2nd level index blocks = b2 = 2 entries.

b3 = ceiling (r3/bfri) = ceiling (2/360) = ceiling (0.005) = 1 block.

So, x = 3 (total index levels)

iv)

bi = b1+b2+b3 = 695+2+1 =698 (total index blocks)

v)

number of block accesses = x+1 = 3+1=4

1. Suppose that the file is not *ordered* by the key field CompanyId and we want to construct a *secondary* index on CompanyId. Repeat the previous exercise (part c) for the secondary index and compare with the primary index.

**Ans:**

i)

Ri = CompanyIdSize + Pb = 8+6 = 14 bytes (index entry size)

bfri = floor(B/Ri) = floor (5041/14) = floor(360.07) = 360 (index entries per block)

ii)

r1 = number of file records i.e. dense index = r = 10,000,000 entries.

b1 = ceiling (r1/bfri) = ceiling (10,000,000/360) = ceiling (27,777.78) = 27,778 blocks.

iii)

r2 = number of 1st level index blocks = b1 = 27,778 entries.

b2 = ceiling (r2/bfri) = ceiling (27,778/360) = ceiling (77.16) = 78 blocks

r3 = number of 2nd level index blocks = b2 = 78 entries.

b3 = ceiling (r3/bfri) = ceiling (78/360) = ceiling (0.2167) = 1 block.

So, x = 3 (total index levels)

iv)

bi = b1+b2+b3 = 27,778 + 78 + 1 = 27,857 (total index blocks)

v)

number of block accesses = x+1 = 3+1=4

1. Suppose that the file is not *ordered* by the non-key field MainWarehouse and we want to construct a *secondary* index on MainWarehouse, with an extra level of indirection that stores record pointers. Assume there are 20000 distinct values of MainWarehouse and that the Company records are evenly distributed among these values. Calculate (i) the index blocking factor bfri, (which is also the index fan-out, *fo);* (ii) the number of blocks needed by the level of indirection that stores record pointers (bind); (iii) the number of first-level index entries (r1) and the number of first-level index blocks (b1); (iv) the number of levels needed if we make it into a multilevel index (x); (v) the total number of blocks required by the multi-level index (bi) including the blocks used in the extra level of indirection; and (vi) the approximate number of block accesses needed to search for and retrieve all records in the file that have a specific MainWarehouse value, using the index.

**Ans:**

i)

Ri = MainWarehouseSize + Pb = 9+6 = 15 bytes (index entry size)

bfri = floor (B/Ri) = floor (5041/15) = 336 (index entries per block)

ii)

Total distinct values for Main Warehouse = 20,000

Avg number of records per distinct value = r/total distinct values = 10,000,000/20,000 = 500

bytes for indirection for each distinct value = 7 \* 500 = 3500

It fits in one block, Since 3500 < 5041.

so bindirection = 1 \* 20,000 = 20,000

iii)

r1 = number of distinct values of main warehouse = 20,000 entries.

b1 = ceiling (r1/bfri) = ceiling (20,000/336) = 60 blocks.

iv)

r2 = number of 1st level index blocks = b1 = 60 entries.

b2 = ceiling (r2/bfri) = ceiling (60/336) = 1 block

so, x = 2 (total index levels)

v)

bi = b1 + b2 + bindirection = 60 + 1 + 20,000 = 20,061 (total index blocks needed)

vi)

number of block accesses = x + 1 + average record for each distinct value = 2 + 1 + 500 = 503

1. Suppose that the file is *ordered* by the non-key field MainWarehouse and we want to construct a *clustering index* on MainWarehouse that uses block anchors (every new value of MainWarehouse starts at the beginning of a new block). Assume there are 20000 distinct values of MainWarehouse and that the Company records are evenly distributed among these values. Calculate (i) the index blocking factor (bfri), (which is also the index fan-out, *fo);* (ii) the number of first-level index entries (r1) and the number of first-level index blocks (b1); (iii) the number of levels needed if we make it into a multilevel index (x); (iv) the total number of blocks required by the multilevel index (bi); and (v) the number of block accesses needed to search for and retrieve all records in the file that have a specific MainWarehouse value, using the clustering index (assume that multiple blocks in a cluster are contiguous)

**Ans:**

for bi,

bfr = floor(B/R) = floor (40.3) = 40

cluster blocks = ceiling (500/40) = ceiling (12.5) = 13 (i.e. 500 records are clustered in 13 blocks)

total block accesses = x + ClusterBlocks = 2+13 = 15

i)

Ri = MainWarehouseSize + Pb = 9+6 = 15 bytes (index entry size)

bfri = floor(B/Ri) = floor (5041/15) = 336 (index entries per block)

ii)

r1 = Total distinct values for MainWarehouse = 20,000 entries.

b1 = ceiling (r1/bfri) = ceiling (20,000/336) = 60 blocks.

iii)

r2 = number of 1st level index blocks = b1 = 60 entries.

b2 = ceiling (r2/bfri) = ceiling (60/336) = 1 block

So, x = 2 (total index levels)

iv)

bi = b1 + b2 = 60 + 1 = 61 (total index blocks)

v)

bfr = floor(B/R) = floor (40.3) = 40

cluster blocks = ceiling (500/40) = ceiling (12.5) = 13 (i.e. 500 records are clustered in 13 blocks)

Total block accesses = x + ClusterBlocks = 2+13 = 15

1. Suppose the file is not ordered by the key field CompanyId and we want to construct a B+-tree access structure (index) on CompanyId. Calculate (i) the *orders p* and *p leaf* of the B+-tree; (ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience); (iv) the total number of blocks required by the B+-tree; and (v) the number of block accesses needed to search for and retrieve a record from the file--given its SSN value--using the B+-tree.

**Ans:**

i)

For internal and root node:

(p\* P) + ((p-1) \* VCompanyId) < B

(p \* 6) +((p-1) \* 8) < 5041

**Order p = 360**

For leaf node:

pleaf \* (VCompanyId + Pr)) + P < B

pleaf \* (8 + 7)) + 6 < 5041

pleaf < 335.6

**Order pleaf = 335**

ii)

ceiling (0.69 \* pleaf) = ceiling (0.69\*335) = ceiling (231.1) = 232 (keys values per leaf node with 69% full)

b1 = ceiling (10,000,000/232) = ceiling (43,103.448) = 43,104 leaf level blocks

iii)

fo = ceiling (0.69 \* p) = ceiling (0.69 \* 360) = ceiling (248.4) = 249

b2 = ceiling (b1/fo) = ceiling (43,104/249) = ceiling (173.8) = 174

b3 = ceiling(b2/fo) = ceiling (174/249) = 1

Total tree levels = x = 3

iv)

bi = b1 +b2 + b3 = 43,104 + 174 + 1 = 43,279

v)

Total block accesses = x + 1 = 3 + 1 = 4

1. Repeat part (g), but for a B-tree *rather than for a* B+-tree*.* Compare your results for the B-tree and for the B+-tree.

**Ans:**

i)

(p\* P) + ((p-1) \* (VCompanyId + Pr) < B

(p \* 6) +((p-1) \* (8 + 7) < 5041

**Order p = 240**

ii)

ceiling (0.69 \* p) = ceiling (0.69 \* 240) = ceiling (165.6) = 166 (keys values per leaf node with 69% full)

b1 = ceiling (10,000,000/166) = 60,241blocks

iii)

fo = ceiling (0.69 \* p) = ceiling (0.69 \* 240) = ceiling (165.6) = 166 (keys values per internal node with 69% full)

b2 = ceiling (b1/fo) = ceiling (60,241/166) = 363

b3 = ceiling(b2/fo) = ceiling (363/166) = 3

b4 = ceiling(b3/fo) = ceiling (3/166) = 1

Total tree levels = x = 4

iv)

bi = b1 +b2 + b3 + b4 = 60,241 + 363 + 3 + 1 = 60,608

v)

Total block accesses = x + 1 = 4 + 1 = 5

**Q2 Solution. [**Do yourself]

**Q3.** Assume a relation *R (W, Y, Z)* is given; Suppose *W, Y, Z* are integer type values. *R* is stored as an un-ordered file (un-spanned) on key field *W* and contains 500 data blocks. Assume there is B+- tree access structure (index) on *W* of height x=3 (root, 1 intermediate layer, leaf). Moreover, one node of the B+-tree is stored in one block on the disk.

Estimate the number of block fetches needed to compute the following queries:

1. *SELECT W, Y, Z FROM R WHERE W = 12;*
2. *SELECT Y, Z FROM R WHERE W = 23 AND Y = 59;*
3. *SELECT W, Y, Z FROM R WHERE W = 49 OR W = 23;*
4. *SELECT Y, Z FROM R WHERE W > 25;*

**Q3 Solution:**

***a) Block accesses = x + 1 = 4***

***b) Block accesses= x + 1 = 4***

***c) Block accesses = 2(x + 1) = 8***

***d) Index cost (one root node + one internal node + all leaf nodes) + Table access cost (b=500 i.e. full table scan).***

**Q4.** Consider a DBMS that has the following characteristics:

* 4KB fixed-size blocks
* 12-byte pointers (block/record)
* 56-byte block headers

We want to build an index on a search key that is 8 bytes long. Calculate the maximum number of records we can index with

1. a 3-level B+-tree (including the root level)
2. a 3-level B-tree (including the root level)

**Q4 Solution:**

Block Available Space= 4096 – 56 = 4040

a)

Let each node of a B+-tree contain at most p pointers and p-1 keys. Order P is (p \* 12) + ((p - 1) \* 8) < 4040. Therefore, Order p = 202. Order Pleaf is pleaf \* (8 + 12) + 12 < 4040. Therefore, Order Pleaf = 201. The leaf level of a B+-tree can hold at most 202 \* 202 \* 201 record pointers. Therefore, the maximum number of records that can be indexed is **8,201,604**.

b)

Let each node of a B-tree contain at most p index pointers, p-1 keys, and p-1 record pointers. Order P is (p \* 12) + (p - 1) \* (8 + 12) < 4040. Therefore, Order p = 126. The first level of a B-tree can hold at most 125 record pointers. The second level can hold at most 126 \* 125 record pointers. The leaf level can hold at most 126 \* 126 \* 125 record pointers. Therefore, the maximum number of records that can be indexed is 125 + (126 \* 125) + (126 \* 126 \* 125) = 2,000,375.