CS560 Spring, 2016

Homework #1

Shuo Yang

---

1. Since the block size is 16KB and record size is 32 Bytes, the number of records in each block $R = 16\text{KB}/32\text{Bytes} = 500$. Since there are total number of records 300 million, thus the total number of blocks $B = 300 * 10^6/R = 6 * 10^5$. Assume that $F = 100$, which is the number of index entries in a non-leaf B-tree node.

   (a) *Cost*: $BD_s = 6 * 10^5 * 1ms = 600s$
   Explanation: we must sequentially retrieve each of the B blocks and it takes D_s time to read a block.

   (b) *Cost*: $0.5BD_s = 0.5 * 6 * 10^5 * 1ms = 300s$
   Explanation: On average, we must sequentially scan half the file, assuming that the record exists and the distribution of values in the search fields is uniform.

   (c) *Cost*: $BD_s = 6 * 10^5 * 1ms = 600s$
   Explanation: we must sequentially scan the entire file since qualifying records could appear anywhere in the file.

   (d) *Cost*: $D_r + D_s = 10ms + 1ms = 11ms$
   Explanation: assuming that records are always inserted at the end of the file. It takes one random read to fetch the last block into the memory, insert the record, and write the block back (sequential write).

   (e) *Cost*: cost of search + $D_s = 0.5BD_s + D_s = 300s + 1ms \approx 300s$
   Explanation: we must find the block that contains the record, remove it from the block and write (sequential write) the modified block back, assuming that we don't compact the file to reclaim the free space created by deletion.

   (f) *Cost*: $BD_s = 6 * 10^5 * 1ms = 600s$
   Explanation: we must sequentially retrieve each of the B blocks and it takes D_s time to read a block.

   (g) *Cost*: $D_r * \log_2 B = 10ms * \log_2(6 * 10^5) \approx 190ms$
   Explanation: we can locate the block containing the desired record with a binary search in $\log_2 B$ steps, each step takes a random read.

   (h) Assume that the number of blocks with match records is 10.
   Cost: $D_r * \log_2 B + D_s *$ number of blocks with match records $= 10ms * \log_2(6 * 10^5) + 1ms * 10 \approx 200ms$
   Explanation: the first record that satisfies the selection is located as for search for equality. Subsequently, data blocks are sequentially retrieved until a record is found that does not satisfy range selection.

   (i) *Cost*: cost of equality search + $BD_s = D_r * \log_2 B + BD_s = 190ms + 600s = 600.19s$
   Explanation: we must first find the correct position in the file to insert, insert the record, then fetch and rewrite all the subsequent blocks, because all the old records are shifted by one slot, assuming that the file has no empty slot. On average, we assume that the inserted record belongs the middle of the file. Therefore, we must read the latter half of the file and write it back after adding the new record.

   (j) *Cost*: cost of equality search + $BD_s = D_r * \log_2 B + BD_s = 190ms + 600s = 600.19s$
   Explanation: Same reason as for the insertion.

   (k) *Cost*: $1.5BD_s = 1.5 * 6 * 10^5 * 1ms = 900s$
   Explanation: all data blocks must be retrieved and we assume 67% percent occupancy.

   (l) *Cost*: $D_r \log_F(1.5B) = 10ms * \log_{100}(1.5 * 6 * 10^5) \approx 30ms$
   Explanation: we can locate the block containing the desired record in $log_F(1.5B)$ steps, that is, by fetching all pages from root to the appropriate leaf. Each fetching is a random disk access.

   (m) *Cost*: $D_r \log_F(1.5B) + D_s *$ number of blocks with match records $= 10ms * \log_{100}(1.5 * 6 * 10^5) + 1ms * 10 \approx= 40ms$ *Explanation*: the first record that satisfies the range selection is located as it is for equality search. Subsequently, data pages are fetched sequentially.

   (n) *Cost*: cost of equality search + $D_s \approx 31ms$.
   Explanation: we first use the equality search to locate the leaf page that the new record should be inserted, then add the new record and write it back. We assume that the leaf page has sufficient space for the new record.

(o) *Cost*: cost of equality search + $D_s \approx 31ms$.
Explanation: Same reason as it is for insertion.

(p) *Cost*: $0.15 * BD_s + BD_rR \approx 3 * 10^6 s$ *Explanation*: it cost $0.15BD_s$ to fetch all leaf pages of the tree, then it takes one I/O operation to fetch each record for each data entry in the index because the index is unclustered and each data entry on a leaf page could point to a different page in the file. Each such an operation is a random read.

(q) *Cost*: $D_r(1 + \log_F 0.15B) \approx 35ms$
Explanation: we can locate the page with the desired data entry within $\log_F 0.15B$ steps, each requires a random access, then we need another random disk I/O to fetch the page containing actual desired record.

(r) *Cost*: $D_r(\log_F 0.15B+$ number of matching records $) \approx 1025ms$, assuming that number of matching records is 100
Explanation: the first record that satisfies the selection is located as it is for search with equality, subsequently, data entries are sequentially retrieved, for each matching data entry, it incurs one random disk I/O.

(s) *Cost*: $D_r + D_s+$ cost of equality search $\approx 46ms$ *Explanation*: we first insert the record in the file, at the cost of one random read and one sequential write, and we also need to update the index to insert a new data entry.

(t) *Cost*: $D_r + D_s+$ cost of equality search $\approx 46ms$ *Explanation*: we first locate the data record in the file and the data entry in the index, and then we need to write out the modified pages in the index and the date file.

(u) *Cost*: $0.125*BD_s+BD_rR \approx 3*10^6 s$ *Explanation*: it cost $0.125BD_s$ to fetch all data entries, then it takes one I/O operation to fetch each record for each data entry in the index because the index is unclustered. Each such an operation is a random read.

(v) *Cost*: $2D_r = 20ms$ *Explanation*: it takes one random disk access to fetch the hash bucket and another random disk access to fetch the data record in the file.

(w) *Cost*: $BD_s = 6*10^5*1ms = 600s$ *Explanation*: the hash structure offers no help, so we have to sequentially scan the entire file.

(x) *Cost*: $D_r + D_s+$ cost of equality search $\approx 31ms$ *Explanation*: we first insert the record in the file, at the cost of one random read and one sequential write, and we also need to update the index to insert a new data entry.

(y) *Cost*: $D_r + D_s+$ cost of equality search $\approx 31ms$ *Explanation*: we first locate the data record in the file and the data entry in the index, and then we need to write out the modified pages in the index and the date file.

2. (a) number of sectors: 5,860,533,168
(b) number of platters: 8
(c) number of tracks: 93024336 (calculated as: number of sectors/sectors per track(63))
(d) sector size: 512 Bytes
(e) capacity: 3000GB (3TB)
(f) amount of space for error correction: cannot find
(g) seek time: <12ms typical
(h) rotation delay: 5.1ms
(i) transfer time: it takes 1 second to transfer the maximum of 600MB data
(j) minimum random read time: cannot find
(k) minimum sequential read time: cannot find
(l) maximum random write time: cannot find
(m) maximum sequential write time: cannot find

manufacturer name: Seagate
part number: ST3000DM003
source: http://archive.benchmarkreviews.com/?option=com_content&task=view&id=1071&Itemid=60