Database Development and Design (CPT201)

Tutorial 1: Database Storage

Dr. Wei Wang
Department of Computing

Q1

- Given a disk with the following characteristics:
  - There are 2^{14}=16384 tracks per surfaces
  - There are 2^{7}=128 sectors per track
  - There are 2^{12}=4096 byte per sector
  - The disk rotates at 7200rpm; i.e., it makes one rotation in 8.33 milliseconds
  - To move the head arm between cylinders (tracks) take one milliseconds to start and stop, plus one additional millisecond for every 1000 cylinders travelled.
- Questions:
  - 1. what is the time to take one track movement?
  - 2. what is the time to move the head from innermost track to outmost track?

Q1 Answer

- 1.001ms: move head arm between tracks needs 1 ms plus 0.001ms for cylinder travelling

- 17.384 millisecond: 16384 tracks on a surface, so total cylinder travelling is 16.384ms, plus 1 ms, which is 17.384ms

Q2

- Given a disk with the following characteristics:
  - There are 2^{14}=16384 tracks per surfaces
  - There are 2^{7}=128 sectors per track
  - There are 2^{12}=4096 byte per sector
  - The disk rotates at 7200rpm; i.e., it makes one rotation in 8.33 milliseconds
  - To move the head arm between cylinders (tracks) takes one milliseconds to start and stop, plus one additional millisecond for every 1000 cylinders travelled.
- Questions:
  - 1. Assume that there is no gap between sectors and each block occupies 4 sectors. what is the minimum time to read a block ?
  - 2. What is the maximum time to read a block?
  - 3. What is the average time?

Q2 Answer

- Best case: seek time = 0; rotational latency = 0
  - Time taken for one complete rotation: 1/(7200/60) = 0.00833s = 8.33ms
  - Reading 4 sectors only needs to rotate 4/128
  - *So the time is 8.33*4/128 = 0.2603 ms*

Q2 Answer cont'd

- Worst case:
  - Worst time: travel from innermost to outermost track; then rotate the whole track (*assume the read/write head passes the first sector a little*); then plus the read time.
  - 17.38 + 8.33 + 0.2603 (rotation latency: 8.33; transfer time: 0.2603)
- Average case:
  - Average time is "½" of the worst time except for the actual read time
  - (17.38+8.33)/2 + 0.2603
    - More precisely, 1+(16.38+8.33)/2+0.2603

Q3

- Suppose that a relation called student holds 25,000 tuples, which are stored as fixed length and fixed format records. The length of each tuple is 350 bytes. The key attribute, student_ID, occupies 10 bytes and another attribute address occupies 50 bytes. The records are sequentially ordered by student_ID and stored in a number of blocks. Each block has the size of 4,096 bytes (i.e., 4 Kilobytes). Assume that a complete record or an index entry must be stored in one block.
  - (1) How many blocks are needed to store the relation student?
  - (2) Consider creating a primary index on the *student_ID* attribute. Each index entry contains a search key and a 10-byte long pointer to the records. Suppose the primary index is sparse (i.e. one index entry for one block), compute the number of blocks needed to store the index.

Q3 Answers

- (1)
  - Each tuple of student is 350 bytes. Each block at most holds $\lfloor 4{,}096 \text{ bytes}/350 \text{ bytes} \rfloor = 11$ tuples (where $\lfloor\ \rfloor$ indicates round down).
  - There are 25,000 tuples, so $\lceil 25000 \text{ tuples}/11 \text{ tuples per block} \rceil = 2{,}273$ blocks required (where $\lceil\ \rceil$ indicates round up).
- (2)
  - Each index entry is 10 bytes for the key plus 10 bytes for the pointer (20 bytes in total).
  - Each block at most can store $\lfloor 4096 \text{ bytes}/20 \text{ bytes} \rfloor = 204$ index entries.
  - There are 2,273 blocks in the *student* relation (answer from question 1), so $\lceil 2{,}273/204 \rceil = 12$ blocks are needed.

NOTE: answers that do not use floor or ceiling functions will be considered as wrong.

Q4

- A relational database contains the following two relations:
  - **Relation *student***:
    - Tuples are stored as fixed-format, fixed-length records, each of 300 bytes.
    - Each tuple contains a key attribute *sID* of length 20 bytes; other fields and the record header make up the rest.
    - There are 1,000 tuples.
  - **Relation *module***:
    - Tuples are stored as fixed-format, fixed-length records, each of 200 bytes.
    - Tuples contain attribute *module.sID* (ID of a student who takes a module), referencing the *student.sID*.
    - There are 5,000 tuples.
- Assume that each student must take exactly 5 modules. The student records are sequentially ordered by *sID* and the "clustered disk organisation" strategy is used. This means that, for each student record, the 5 module records also reside on the same block. Also, assume that a record does not span over more than one block, and the size of each block is 4,096 bytes.

Q4 – cont'd

- (1) Calculate the number of disk blocks needed to store the two relations.
- (2) Suppose that a dense primary index is to be created on *sID* for the *student* relation, i.e. one index entry for each tuple. Each index entry includes a key and a 10-byte pointer to data (an 8 byte block ID and a 2 byte offset). How many disk blocks are needed to store the index?
- (3) Suppose that a sparse primary index is to be created on *sID* for the *student* relation, i.e. one index entry for each disk block. Each index entry includes a key and a 10-byte pointer to data (an 8 byte block ID and a 2 byte offset). How many disk blocks are needed to store the index?

Q4 Answers

- (1) A student record, plus the 5 module records, has the size 300 + 5*200 = 1,300 bytes. Each block can hold $\lfloor 4{,}096/1{,}300 \rfloor$ = 3 of them. There are 1,000 student tuples, so $\lceil 1{,}000 / 3 \rceil$ = 334 blocks are needed to store both relations.
- (2) Clustered organisation, dense primary index: each block can hold up to $\lfloor 4{,}096/(20+10) \rfloor$ = 136 index entries. There are 1,000 tuples in the student relation, so $\lceil 1{,}000/136 \rceil$ = 8 blocks are needed.
- (3) Clustered organisation, sparse primary index: This index requires one index entry for every block. Each block can still hold 136 index entries, but now there are 334 entries needed (answer from question 1), so $\lceil 334/136 \rceil$ = 3 blocks are needed.

NOTE: answers that do not use floor or ceiling functions will be considered as wrong.