CS143: Database Systems
Homework #3

1. We want to store the table created by the following SQL statement into a disk.

```
CREATE TABLE Class(
  dept  CHAR(2),
  cnum  INTEGER,
  sec   INTEGER,
  unit  INTEGER,
  year  INTEGER,
  quarter INTEGER,
  title CHAR(30),
  instructor CHAR(20)
)
```

We need to store tuples for 1,000 classes that have been offered so far. 10 classes are offered every year. The tuples are stored in random order (i.e., they are not sequenced by any attribute). A disk of the following parameters is used for storing the table.

- 3 platters (6 surfaces)
- 10,000 cylinders
- 500 sectors per track
- 1024 bytes per sector
- 6,000 RPM rotational speed
- 10ms average seek time

(a) What is the capacity of this disk?

ANSWER:
`6 surfaces/disk * 10,000 tracks/surface * 500 sectors/track * 1KB/sector = 30GB/disk`

(b) What is the average time to read a random sector from the disk?

ANSWER:
`(seek time) + (rotational delay) + (transfer time) = 10ms + 5ms + 0.02ms = 15.02ms`

(c) Assume one disk block corresponds to one disk sector. How many disk blocks are needed to store the above table with 1,000 tuples?

ANSWER:
72 blocks. $\lfloor \frac{\texttt{1024 bytes/block}}{\texttt{1 tuple/72 bytes}} \rfloor = \texttt{14 tuples/block.}$ $\lceil \frac{\texttt{1000 tuples/table}}{\texttt{14 tuples/block}} \rceil = \texttt{72 blocks/table.}$

(d) We want to run the following query by scanning the entire table.

```
SELECT * FROM Class WHERE year = 2005
```

Assuming that all blocks for the table is allocated sequentially, how long will it take to run the query? Assume that the disk head is not on the same track where the first block of the table is stored.

ANSWER:

```
(seek time) + (rotational delay) + (transfer time) = 10ms + 5ms + 72*0.02ms = 16.44ms
```

(e) Now assume that due to frequent updates to the table, disk blocks are allocated such that, on average, sequentiality is broken every three blocks. That is, the table is stored in 24 randomly located "clusters" of 3 consecutive blocks. Assuming that we scan the entire table to execute the above query, how long will it take?

ANSWER:

```
24 * ((seek time) + (rotational delay) + (transfer time)) = 24 * (10ms + 5ms + 3*0.02ms)
= 361.44ms
```

(f) Now assume that we have a B+tree on the year attribute and the tree has already been loaded into main memory. None of the disk blocks containing the Class table has been cached in main memory. What is the expected time to run the above query? Is it helpful to create a B+tree to run this query?

ANSWER:

```
150.2ms.  Since the tuples are not clustered by the search key, we will need to do
10 random IOs to retrieve all 10 tuples.  Therefore, 10*(10ms+5ms+0.02ms) = 150.2ms.
If all blocks are allocated sequentially, using the index may actually slow down the
query execution.
```

Indexes

The table `taken(StNo, CourseID, Year, Quarter, Sec, Grade, Remarks)` contains the grades for the courses completed by UCLA students during the last 20 years. If 10,000 new students enter UCLA every year, we can assume that in `taken` there are 200,000 different students, each identified by a StudentID. Thus will assume there are 40,000 students enrolled each quarter, and that each student takes four classes per quarter (160,000 classes taken by students each quarter) and that there are three quarters in each year (480,000 classes taken every year). Thus we get a total of 9,600,000 tuples recording the grades of all students over the last 20 years. Also assume that the average number of students per class is 100; this implies that 4,800 classes are offered each year.
On this table, we have a sparse index on `StNo` and a dense index on the combination: `(CourseID,Year,Quarter,Sec,StNo)`. Both indexes are implemented as B+ trees where `CourseNo, Year, Quarter, Sec`, and `StNo` take 8 bytes each, and the pointers used in the B+ trees take 10 bytes.

B1 If the file blocks have 4096 bytes and each tuple in `taken` requires 100 bytes, how many blocks will be needed to store the unspanned tuples of this relation ?

$\lfloor 4096/100 \rfloor = 40$ unspanned tuples per block. $\lceil 9,600,000/40 \rceil = 240,000$ blocks total.

B2 Compute the levels and the number of blocks at each level of the B+ tree, assuming a worst-case scenario.

N = $\lfloor (4096 - 10)/50 \rfloor + 1 = 82$ Worst case: why floor not ceiling?

N/2 = 41 pointers: worst case leaf and internal nodes.

Dense index: $\lfloor 9,600,000/41 \rfloor =$ 234,146 at leaf level

$\lfloor 234,146/41 \rfloor =$ 5,710 at first level

$\lfloor 5,710/41 \rfloor =$ 139 at second level

$\lfloor 139/41 \rfloor = 3$ at third level

finally the root level

B3 How many blocks of B+ tree and file will the DBMS retrieve from disk to answer the following query: *Find the average grade in a given class (e.g. find the average grade for: CS143, 2010, Fall, sec. 1).* Assume the **worst-case** scenario, and that all the buffers are initially empty.

We can use the dense index because the search key is a prefix of the dense index key.

5 index blocks to the first leaf, then we need to retrieve leaf pointers for all 100 students. That's up to three additional leaf blocks worst-case. Assuming that 100 students took that particular class, 100 more blocks will be accessed for actual data records, since this is not a a clustered index. Total: 8 B+ tree blocks, 100 file blocks.

B4 We now want to compute the average grade over the (480,000 or so) classes taken in year 2011 (assume that they all have the same credit). Explain how the DBMS will go about searching and retrieving blocks from disk for this query, and estimate the number of blocks the system will have to fetch if those 200,000 students each took 48 classes on the average. Assume the **worst-case** scenario, and that all the buffers are initially empty.

We will have to scan the whole file with the help of the sparse, and thus clustered, index on `StNo`.

$\lfloor(4096 - 10)/18\rfloor = 227$: thus N for this B+ tree is 228. 114 w.c. pointers. So, with 240,000 blocks, we see $\lfloor 240,000/114 \rfloor$ = 2,105 blocks at leaf level. So there are 18 blocks at the next level, and then the root. The complete file is fetched by following the leftmost branch in the B+ tree and then the leaf nodes that are chained together. Thus we have to fetch 2 + 2,105 + 240,000 blocks.

Joins and Optimization

In addition to the table `taken` whose schema and index have been described previously, we also have the table `student(StNo,Level,FirstName,LastName,Major)` describing our 200,000 students. This table has a primary B+ index on `StNo` which is the key for this relation and a foreign key for `taken`.There are five different levels: freshman, softmore, junior, senior, others.

C1 How many blocks will `student` use, if each tuple requires 100 bytes and each block contains 4096 bytes?

Answer: 200,000/40=5,000 blocks

C2 For the query $\pi_{StNo}(\sigma_{Level=\text{``}others\text{''}}(student)) \bowtie taken$ estimate the size of the results (i.e., how many tuples). Also estimate the cost of implementing the query measured by the number if blocks read from disks. You can assume that the join takes advantage of the sparse index on `taken.StNo` (Also, for the sake of simplicity, assume that all indexes used are already in main memory). To compute these estimates use the textbook rules of relational queryoptimizers.

Answer: Because of the FK the complete join has as many tuples as taken 9, 600,000 tuples. Others is one out of five, 9, 600,000/5= 1920,000. Computing the join using the index. 200,000/5=40,000 ”others” students. The courses taken by one student fit in two blocks. Thus we have to access 40000×2 blocks of `taken` plus the 5000 blocks of `student`.