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

```
 CREATE TABLE Class(
   dept  CHAR(2),          2 bytes
   cnum  INTEGER,          4 bytes
   sec   INTEGER,          4 bytes
   unit  INTEGER,          4 bytes
   year  INTEGER,          4 bytes
   quarter INTEGER,        4 bytes
   title CHAR(30),         30 bytes
   instructor CHAR(20)     20 bytes
)                          72 bytes
```

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
  => 1/500 rotation/sector = .002 rotation/sector
  => .02 ms/sector
- 1024 bytes per sector
- 6,000 RPM rotational speed
  => 6000 rotation minute
  => 100 rotation/s
  => .01 s/rotation = 10 ms/rotation
- 10ms average seek time

1.What is the average time to read a random sector from the disk?

Access time = seek time + rotational delay + transfer time
= 10ms + .5 * 10 ms + .02 ms
= **15.02 ms**

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

One block = one secotr = 1024 bytes
One tuple = 72 bytes.
1000 tuple = 72000 bytes.
Assume we use spanned method, each block store 1024/72 = 14.22 tuple.
72000 bytes / 1024 bytes = 70.3125 block = **71 block**

3. 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.

Time to sacnning the entire table

= seek time + avg rotational delay + transfer time

= 10 ms + .5 * 10 ms + .02 ms/sector * 71 sector

= **16.42 ms**

4. 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?

71 blocks are splitted into 24 cluster, (the first 23 cluster has 3 blocks, thae last cluste has only 2 blocks). Thus there will be 24 times of random I/O.

First 23 clusters:
Time to scan
    = (seek time + avg rotational delay + transfer time) * 23
    = (10 ms + .5 * 10 ms + .02ms/sector * 3 sector) * 23
    = 346.38 ms

The 24th clusters:
Time to scan
    = seek time + avg rotational delay + transfer time
    = 10 ms + .5 * 10 ms + .02ms/sector * 2 sector
    = 15.04 ms

Total time to scanning the entire table = 346.38 ms + 15.04 ms = **361.42 ms**

5. Now assume that we have a non-clustering index on the year attribute and the index 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 nonclustering index to run this query?

Answer:

It will be helpful to create a nonclustering index to run this query.

Assum we have our tuples sorted by the year attribute.

By using binary search, we still need to jump 6 time around a big chunk of memory in the worse case. Each block has 14 tuples, so we need to do 4 more search in the worst case.

Without the non-clustering index, there are 71 block that we need to jump around to look for our target tuple. By using binary search, we still need to jump 6 times around a big chunk of memory in the worse case. And them, we can binary search within a block. The 6 times search around the 71 block is very expensive.

With the non-clustering index, we can use one block to store all the key-pointer entry. Then we use biary search to search the target tuple location. Jumping with the index table is less expensive then jumping around 71 blocks, since they are in one block.

Expected time
= time to access the index entry + time to access the pointed location
= time to read one random block + time to read one random block
= (seek time + rotational delay + transfer time) * 2
= (10ms + .5 * 10 ms + .02 ms) *2
= **30.04 ms**