W4111 – Introduction to Databases Sections 002, V002; spring 2022

Homework 3a – Written Assignment

Instructions

- The homework submission date/time is 2022-APR-17 at 11:59 PM.
- Submission format is a PDF version of this document with your answers. Place your answers in the document after the questions.
- The name of your PDF must be <UNI>_S22_W4111_HW3a_Written.pdf. For example, mine would be dff9_S22_W4111_HW3a_Written.pdf
- You must use the Gradescope functions to mark the location of your questions/answers in the submitted PDF. Failure to mark pages will cause point deductions. **Please, please read the countless Ed posts, TA produced instructions and videos, etc. to prepare your submission.**
- You can use online sources but you must cite your sources. You may not cut and paste text.
- Questions typically require less than five sentences for an answer. You will lose points if your answer runs on and wanders.

  "Verbosity wastes a portion of the reader's or listener's life."

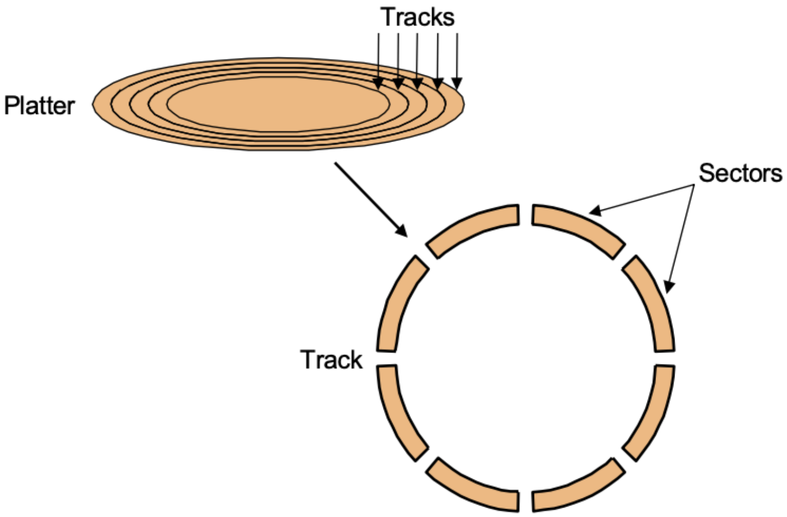
Questions

Question 1: Relative to disk drives, briefly define the following terms:

- Seek time
- Rotational latency time
- Transfer time/data transfer rate

Answer:

1. Seek time is the time taken for the disk head to reach the required track
2. Rotational latency time is the time for the disk head to reach the required sector in a track (by rotating the disk)
3. Transfer time is the time taken for the actual read/write operation



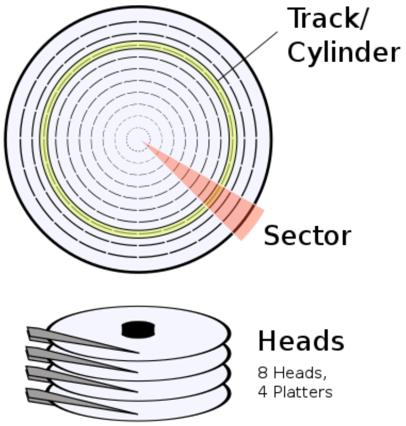
Ref :
https://courses.cs.washington.edu/courses/cse378/10sp/lectures/lec26.pdf

Question 2: Briefly explain logical block addressing and cylinder-sector-head addressing.

Answer:

Logical block addressing is a way to specify addresses for each physical block on a disk. The system needs to only specify a fixed-bit-size address (e.g. 32 bit) which maps to a unique coordinate in the cylinder-sector-head space. The host needs to specify only the 32 bit address without knowing anything about the physical address of the block on the disk.

Cylinder-sector-head addressing is a way to specify addresses for each physical block on a disk. It does this is a 3D coordinate system with a vertical coordinate head (specifying which disk to read) , a horizontal (or radial) coordinate cylinder (specifying the distance of the location from the center of the disk), and an angular coordinate sector (specifying the location of the address on the disk). So when all 3 coordinates are specified, it maps to a specific physical address.



Ref: https://en.wikipedia.org/wiki/Cylinder-head-sector

Question 3: Some databases only map data to/use outer sectors/cylinders of hard disk drive. Why?

Answer:

Since the disks do not have equal number of sectors in every track, databases are mapped to the outer sectors. This allows larger amounts of data to be stored in a continuous space (since more number of sectors for outer tracks) and data can be read faster without additional time overheads of seeking.

Question 4: Briefly explain the elevator algorithm for disk I/O scheduling and how it may improve performance?

Answer:

The elevator algorithm takes a buffer of I/O requests,and similar to an elevator, moved the disk head sequentially over the tracks, fulfilling whatever requests exist along that track.
This boosts performance as the disk head is not continuously moving between tracks for every I/O request, leading to better performance.

Question 5: Relative to database buffer management, briefly explain the following concepts:

- Cache hit/cache miss
- Pinned page/block
- Least recently used replacement algorithm

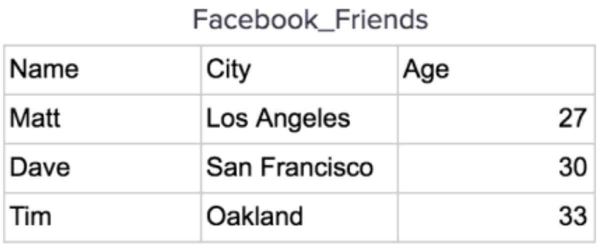
Answer:
The cache is a memory location that has very low I/O overhead costs. So, frequently used data blocks are stored here so that the I/O request time is minimized. When an I/O request is received for a block present in the cache it is called a cache hit, and when an I/O request comes for a block not present in the cache it is called a cache miss.

A page or a block that is not allowed to be written to the hard disk is called a pinned page/block.

Since the size of the the cache is fixed and small, once it is full, there has to be a policy to decide which cache pages get dropped to make space for new pages. LRU algorithm drops the cache block that had the oldest I/O request.

Question 6: Briefly explain the concepts of and differences between row-oriented storage and column-oriented storage. For each model, give an example of row access/query patterns that would benefit from the model.

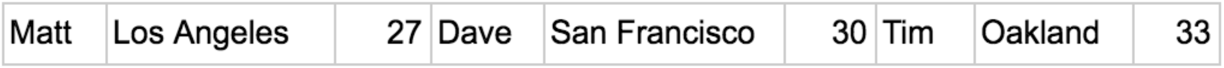
Answer:
Ref:
https://dataschool.com/data-modeling-101/row-vs-column-oriented-databases/
Row-oriented storage stores the database by records, storing them sequentially. In contrast, a column oriented database would store by fields, storing all of the fields together.
For example take the database and it’s corresponding row and column storage on disk.



Facebook_Friends

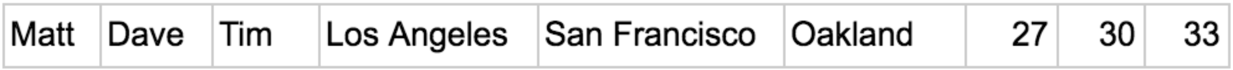
Name	City	Age
Matt	Los Angeles	27
Dave	San Francisco	30
Tim	Oakland	33

Row oriented storage



Matt	Los Angeles	27	Dave	San Francisco	30	Tim	Oakland	33

Column Oriented Storage



Matt	Dave	Tim	Los Angeles	San Francisco	Oakland	27	30	33

If we were to add a new record to the table, the row-oriented storage would just append it at the end, however, the column-oriented storage would have to append each field and shift the remaining data leading to a much larger I/O overhead cost. So, in this scenario row-oriented storage is better.

However, if we want to query all the cities present in this database, the row-oriented would need to traverse the entire database but the column-oriented would just return the sequential data for the field. So in this scenario, column oriented is better.

Question 7: Provide one benefit and one disadvantage of variable length record management/organization relative to fixed length record management/organization.

Answer:
Variable-length records are better than fixed-length records since no space is ever wasted, and only the exact space required by the record/row in the database is used.

However, it becomes much more difficult to locate the record on disk, since in fixed-length records, we know the uniform addressing locations of records.

Question 8: The sample database associated with the book has a *takes* relation/table – *takes(ID, course_id, section_id, semester, year, grade)* that associated students and courses/sections.

1. How might table partitioning (storage) be beneficial for storing the rows in the table?
2. What columns would the database use to partition the data?
3. Why might the database partition the *takes* table but not partition the *student* and *section* tables?

Answer:

1. Since the takes table is very rarely going to change after the semester is complete (students have already taken the course), we can partition this table based on the recent years vs historical data. This would reduce the time taken to add/edit records in the table since we will only be dealing with the newest year/semester.
2. The semester and year would be the columns used to partition this data. Since they are the ones which decide when the student has taken the course, and the older semesters/years are going to be unchanged.
3. The database will not partition student and section tables since their information can always change (e.g. Student changes their email ID even after graduation, course changes their department when the university undergoes a restructuring)

Question 10: Briefly explain and give 2 examples of how the algorithm for mapping records/rows to file/disks blocks can significantly affect performance.

Answer:

The algorithm of mapping records/rows to file/disks blocks will determine the query time and insertion/deletion time for databases.
For example, in Heap File Organization records are placed anywhere in the file that there is free space. However, this requires an efficient way of finding free space in the file. Moreover, records don't move once placed.
In contrast, the hashing method uses a hash function computed on search key; the result specifies in which block of the file the record should be placed. This is able to search for a record without going over all other records sequentially.