W4111\_003\_2024\_3 –  
Introduction to Databases  
Homework 3A

# Overview

HW3A is primarily questions with short, written answers. Please be concise. Please see Ed for the details on submission date and submission format.

# Storage Devices

S1. What are three important criteria for choosing the type of storage to use for data?

S2. Some databases use magnetic disks in a way that only sectors in outer tracks are used while sectors in inner tracks are left unused. What might be the benefits of doing so?

S3. Assume that the access pattern to a relation is primarily sequential. How might the storage manager place the relation’s data on a cylinder, head, sector storage device?

S4. List two advantages of solid-state storage/drives over magnetic disk drives/storage devices. What is a disadvantage of SSD relative to magnetic disk storage?

S5. What is an advantage of RAID-0 relative to RAID-5? What is an advantage of RAID-5 relative to RAID-0?

S6. Briefly explain logical block addressing and cylinder-head-sector block addressing.

S7. Briefly explain the disk block access concepts of read-ahead and disk-arm scheduling.

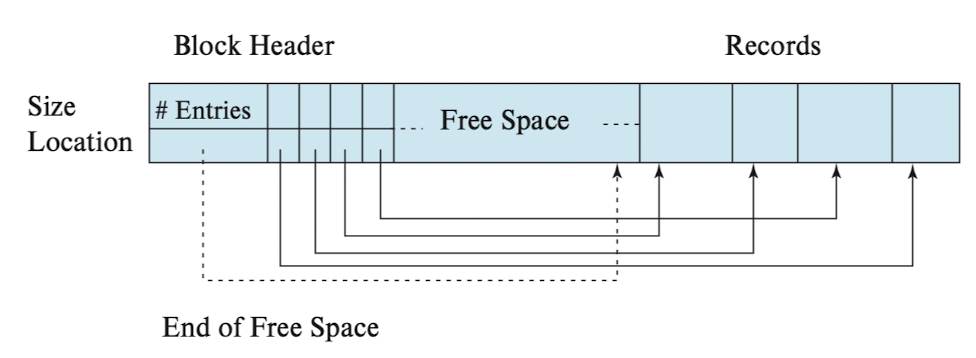
S8. Briefly explain the concepts of network addressable storage and storage area networks.

# Storage Formats

F1. Assume the storage manager is using fixed size/length records. Explain two approaches to implementing addition/deletion of records.

F2. VARCHAR is a common column type and is by definition not a fixed size field. Why might the storage manager choose to use fixed size records and allocate the maximal record size for records with VARCHAR fields?

F3. Consider the image below. Why does the storage manager consolidate free space by moving records when a deletion occurs?



F4. Briefly explain *head file organization, sequential file organization and multi-table clustering organization.* Give a scenario or use case for choosing each of the approaches.

F5. Consider a relation/file that records credit card transactions. Applications periodically retrieve records. 90% of the retrievals are for transactions that occurred in the prior six months. What technique would you recommend for storing the records on disks/files?

# Buffer Management

B1. Operating systems almost exclusively use the least recently used algorithm for the replacement policy. Why do databases sometimes use other algorithms? What is an example of a query for which most recently used might be a good replacement algorithm?

B2. Briefly explain the Clock Algorithm for buffer replacement. What would motivate using the Clock Algorithm instead of least recently used?

B3. What is a pinned block? Does the buffer manager typically use pinned blocks when selecting a block for replacement? Why?

B4. Why would a buffer manager not necessarily write updated blocks to disk in the same order that the updates occurred in the buffer?

# Indexes

I1. How many clustered indexes can exist on a relation/table? Is it possible for a sparse index to be non-clustering/unclustered?

I2. Assume that a very common query on a table with information about people is *select \* from people where last\_name like “XYZ%”* where X, Y and Z are characters. For example, queries of the form *where last\_name like “FER%” or last\_name like “HAW%”* Can you use a hash index to optimize performance for this type of query?

I3. If indexes are so effective at improving query performance, what is the disadvantage of creating very many indexes on a table?

I4. Data Structure courses teach binary search trees. It is common for the degree of a B+ tree to be more than 2. Why do B+ tree use degrees higher than 2? What determines the degree?

# Query Processing

Q1. Briefly explain this sentence, “Each relational algebra operation can be evaluated using one of several different algorithms.” Give three examples for a select operation.

Q2. Consider the following strange query – *select \* from customers join employees on customers.last\_name != employees.last\_name.* Would an index nested loop join work for this type of join?

Q3. What is the primary advantage of a *merge join?* If a merge join requires sorting the relations before performing the join, how is it possible better than a nested-loop join.

Q4. Consider the query *select first\_name, last\_name from person residence\_country=’USA’.* What modification to the query might cause the query engine to create an index on *(first\_name, last\_name)?*

Q5. Briefly explain the concepts of pipelined evaluation and materialized evaluation.