CHAPTER 10

Storage and File Structure

This chapter presents basic file structure concepts. The chapter really consists of two parts—the first dealing with relational databases, and the second with object-oriented databases. The second part can be omitted without loss of continuity for later chapters.

Many computer science undergraduates have covered some of the material in this chapter in a prior course on data structures or on file structures. Even if students' backgrounds are primarily in data structures, this chapter is still important since it addresses data structure issues as they pertain to disk storage. Buffer management issues, covered in Section 10.8.1 should be familiar to students who have taken an operating systems course. However, there are database-specific aspects of buffer management that make this section worthwhile even for students with an operating system background.

Exercises

10.10 List the physical storage media available on the computers you use routinely. Give the speed with which data can be accessed on each medium.

Answer: Your answer will be based on the computers and storage media that you use. Typical examples would be hard disk, CD and DVD disks, and flash memory in the form of USB keys, memory cards or solid state disks.

The following table shows the typical transfer speeds for the above mentioned storage media, as of early 2010.

Storage Media	Speed (in MB/s)
CD Drive	8
DVD Drive	20
USB Keys	30
Memory Cards	1 - 40
Hard Disk	100
Solid State Disks	> 100

Note that speeds of flash memory cards can vary significantly, with some low end cards giving low transfer speeds, although better ones give much higher transfer speeds.

10.11 How does the remapping of bad sectors by disk controllers affect data-retrieval rates?

Answer: Remapping of bad sectors by disk controllers does reduce data retrieval rates because of the loss of sequentiality amongst the sectors. But that is better than the loss of data in case of no remapping!

10.12 RAID systems typically allow you to replace failed disks without stopping access to the system. Thus, the data in the failed disk must be rebuilt and written to the replacement disk while the system is in operation. Which of the RAID levels yields the least amount of interference between the rebuild and ongoing disk accesses? Explain your answer.

Answer: RAID level 1 (mirroring) is the one which facilitates rebuilding of a failed disk with minimum interference with the on-going disk accesses. This is because rebuilding in this case involves copying data from just the failed disk's mirror. In the other RAID levels, rebuilding involves reading the entire contents of all the other disks.

10.13 What is scrubbing, in the context of RAID systems, and why is scrubbing important?

Answer: Successfully written sectors which are subsequently damaged, but where the damage has not been detected, are referred to as latent sector errors. In RAID systems, latent errors can lead to data loss even on a single disk failure, if the latent error exists on one of the other disks. Disk scrubbing is a background process that reads disk sectors during idle periods, with the goal of detecting latent sector errors. If a sector error is found, the sector can either be rewritten if the media has not been damaged, or remapped to a spare sector in the disk. The data in the sector can be recovered from the other disks in the RAID array.

10.14 In the variable-length record representation, a null bitmap is used to indicate if an attribute has the null value.

a. For variable length fields, if the value is null, what would be stored in the offset and length fields?

b. In some applications, tuples have a very large number of attributes, most of which are null. Can you modify the record representation such that the only overhead for a null attribute is the single bit in the null bitmap.

Answer:

a. It does not matter on what we store in the offset and length fields since we are using a null bitmap to identify null entries. But it would make sense to set the offset and length to zero to avoid having arbitrary values.

b. We should be able to locate the null bitmap and the offset and length values of non-null attributes using the null bitmap. This can be done by storing the null bitmap at the beginning and then for non-null attributes, store the value (for fixed size attributes), or offset and length values (for variable sized attributes) in the same order as in the bitmap, followed by the values for non-null variable sized attributes. This representation is space efficient but needs extra work to retrieve the attributes.

10.15 Explain why the allocation of records to blocks affects database-system performance significantly.

Answer: If we allocate related records to blocks, we can often retrieve most, or all, of the requested records by a query with one disk access. Disk accesses tend to be the bottlenecks in databases; since this allocation strategy reduces the number of disk accesses for a given operation, it significantly improves performance.

10.16 If possible, determine the buffer-management strategy used by the operating system running on your local computer system and what mechanisms it provides to control replacement of pages. Discuss how the control on replacement that it provides would be useful for the implementation of database systems.

Answer: The typical OS uses variants of LRU, which are cheaper to implement than LRU, for buffer replacement. LRU and its variants are often a bad strategy for databases. As explained in Section 10.8.2 of the text, MRU is the best strategy for nested loop join. In general no single strategy handles all scenarios well, and the database system should be able to manage its own buffer cache for which the replacement policy takes into account all the performance related issues.

Many operating systems provide mechanisms to lock pages in memory, which can be used to ensure buffer pages stay in memory. However, operating systems today generally do not allow any other control on page replacement.

10.17 List two advantages and two disadvantages of each of the following strategies for storing a relational database:

a. Store each relation in one file.

b. Store multiple relations (perhaps even the entire database) in one file.

Answer:

a. Advantages of storing a relation as a file include using the file system provided by the OS , thus simplifying the DBMS, but incurs the disadvantage of restricting the ability of the DBMS to increase performance by using more sophisticated storage structures.

b. By using one file for the entire database, these complex structures can be implemented through the DBMS, but this increases the size and complexity of the DBMS.

10.18 In the sequential file organization, why is an overflow *block* used even if there is, at the moment, only one overflow record?

Answer: An overflow block is used in sequential file organization because a block is the smallest space which can be read from a disk. Therefore, using any smaller region would not be useful from a performance standpoint. The space saved by allocating disk storage in record units would be overshadowed by the performance cost of allowing blocks to contain records of multiple files.

10.19 Give a normalized version of the *Index_metadata* relation, and explain why using the normalized version would result in worse performance.

Answer: The *Index_metadata* relation can be normalized as follows

Index_metadata(index_name, relation_name, index_type)
Index_Attrib_metadata (index-name, position, attribute-name)

The normalized version will require extra disk accesses to read *Index_Attrib_metadata* everytime an index has to be accessed. Thus, it will lead to worse performance.

10.20 If you have data that should not be lost on disk failure, and the data are write intensive, how would you store the data?

Answer: A **RAID** array can handle the failure of a single drive (two drives in the case of RAID 6) without data loss, and is relatively inexpensive. There are several RAID alternatives, each with different performance and cost implications. For write intensive data with mostly sequential writes, RAID 1 and RAID 5 will both perform well, but with less storage overhead for RAID 5. If writes are random, RAID 1 is preferred, since a random block write requires multiple reads and writes in RAID 5.

10.21 In earlier generation disks the number of sectors per track was the same across all tracks. Current generation disks have more sectors per track on outer tracks, and fewer sectors per track on inner tracks (since they are shorter in length). What is the effect of such a change on each of the three main indicators of disk speed?

Answer: The main performance effect of storing more sectors on the outer tracks and fewer sectors on the inner tracks is that the disk's **data-transfer** rate will be greater on the outer tracks than the inner tracks. This is because the disk spins at a constant rate, so more sectors pass underneath the drive head in a given amount of time when the arm is positioned on an outer track than when on an inner track.

In fact, some high-performance systems are optimized by storing data only in outer tracks, so that disk arm movement is minimized while maximizing data-transfer rates.

10.22 Standard buffer managers assume each block is of the same size and costs the same to read. Consider a buffer manager that, instead of LRU, uses the rate of reference to objects, that is, how often an object has been accessed in the last n seconds. Suppose we want to store in the buffer objects of varying sizes, and varying read costs (such as Web pages, whose read cost depends on the site from which they are fetched). Suggest how a buffer manager may choose which block to evict from the buffer.

Answer: A good solution would make use of a *priority queue* to evict pages, where the priority (p) is ordered by the *expected cost* of re-reading a page given it's past access frequency (f) in he last n seconds, it's re-read cost (c), and its size s:

$$p = f * c/s$$

The buffer manager should choose to evict pages with the lowest value of p, until there is enough free space to read in a newly referenced object.