10.3 A power failure that occurs while a disk block is being written could result in the block being only partially written. Assume that partially written blocks can be detected. An atomic block write is one where either the disk block is fully written or nothing is written (i.e., there are no partial writes). Suggest schemes for getting the effect of atomic block writes with the following RAID schemes. Your schemes should involve work on recovery from failure.
a. RAID level 1 (mirroring)
b. RAID level 5 (block interleaved, distributed parity)

Ans:
a. To ensure atomicity, a block write operation is carried out as follows
i. Write the information onto the first physical block.
ii. When the first write completes successfully, write the same information onto the second physical block.
iii. The output is declared completed only after the second write completes successfully. During recovery, each pair of physical blocks is examined. If both are identical and there is no detectable partial write, then no further actions are necessary. If one block has been partially rewritten, then we replace its contents with the contents of the other block. If there has been no partial write, but they differ in content, then we replace the contents of the first block with the contents of the second, or vice versa. This recovery procedure ensures that a write to stable storage either succeeds completely (that is, updates both copies) or results in no change. The requirement of comparing every corresponding pair of blocks during recovery is expensive to meet. We can reduce the cost greatly by keeping track of block writes that are in progress, using a small amount of nonvolatile RAM. On recovery, only blocks for which writes were in progress need to be compared.

b. The idea is similar here. For any block write, the information block is written first followed by the corresponding parity block. At the time of recovery, each set consisting of the nth block of each of the disks are considered. If none of the blocks in the set have been partially written, and the parity block contents are consistent with the contents of the information blocks, then no further action need be taken. If any block has been partially written, it is contents are reconstructed using the other blocks. If no block has been partially written, but the parity block contents do not agree with the information block contents, the parity block's contents are reconstructed.

10.4 Consider the deletion of record 5 from the file of Figure 10.8. Compare the relative merits of the following techniques for implementing the deletion:

- **(a) Move record 6 to the space occupied by record 5 and move record 7 to the space occupied by record 6.**
- **(b) Move record 7 to the space occupied by record 5.**
- **(c) Mark record 5 as deleted and move no records.**

Ans:
(a) It is the most straightforward approach. It preserves ordering and prevents fragmentation of space but requires moving the most records and involves the most accesses.
(b) It moves fewer records and prevents fragmentation of space but destroys any ordering in the file and may require one more disk block access.

(c) It preserves ordering and moves no records but requires additional overhead to keep track of all the free space in the file and records are no longer stored contiguously.

10.9 Give an example of a relational-algebra expression and a query-processing strategy in each of the following situations:
a. MRU is preferable to LRU.
b. LRU is preferable to MRU.

Ans:
a. MRU is preferable to LRU where R1 1 R2 is computed by using a nested loop processing strategy where each tuple in R2 must be compared to each block in R1. After the first tuple of R2 is processed, the next needed block is the first one in R1. However, since it is the least recently used, the LRU buffer management strategy would replace that block if a new block was needed by the system.
b. LRU is preferable to MRU where R1 1 R2 is computed by sorting the relations by join values and then comparing the values by proceeding through the relations. Due to duplicate join values, it may be necessary to "backup" in one of the relations. This "backing-up" could cross a block boundary into the most recently used block, which would have been replaced by a system using MRU buffer management, if a new block was needed.

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?

Ans:
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 nonnull 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.

10.16 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.

Ans: The typical OS uses LRU for buffer replacement. This is often a bad strategy for databases. Also, MRU is the best strategy for nested loop join. In general, no single strategy handles all scenarios well, and ideally the database system should be given its own buffer cache for which the replacement policy takes into account all the performance related issues.

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.

Ans:
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?

Ans: 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

10.21 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?

Ans: 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 asWeb 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.

Ans: 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 the 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.