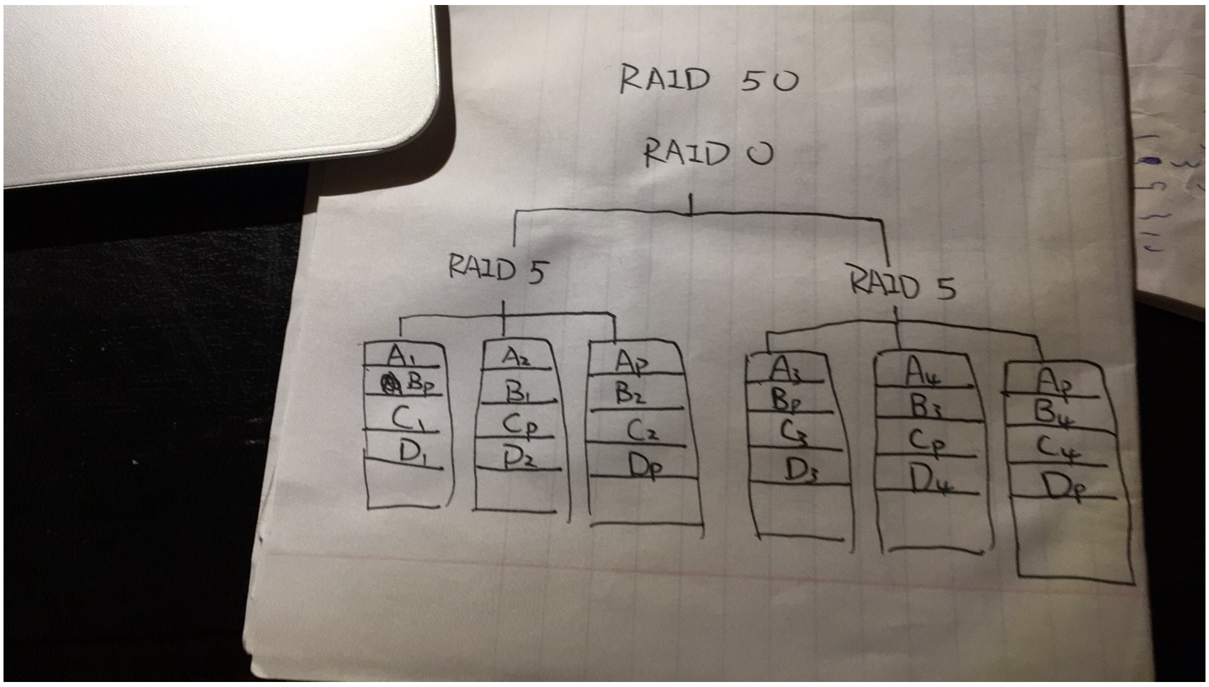
Problem 1



Generally,RAID 50 combines distributed parity (raid 5) with striping (raid 0) and it has better write performance, increased data protection and faster rebuilds than raid 5.

Advantage:

1. Overall better performance than single RAID5 because we add striping property (RAID0 property).
2. Can tolerate more faults if a second fault does not happen in the same set as the previous one.
3. The parity data can protects the data rather than single RAID0.

Disadvantage:

1. For RAID50, we need one disk of parity purpose for each array. Comparing to single RAID5, it has less space.
2. Since RAID50 can be regarded as a combination of RAID0 and RAID5, compare to each single of them, RAID50 is more complicated to construct and build.
3. If more than 2 disks fail within one RAID5 set, the whole RAID50 fails.

Problem 2

The operations like insert, update and delete are easier to manipulated on fixed-length format. Also, the server will need to decompress variable-length format data before processing it. Thus, processing fixed-length format data will also be faster than variable-length format data. Lastly, fixed-length format is more suitable if we know our content is fixed size like zip code or phone number.

Problem 3

LSU:

0	+1
0,1	+1
0,1,2	+1
0,1,2,3	+1
0,1,2,3,4	+1
1,2,3,4,9	+2
1,2,3,9,4	+0
2,3,9,4,8	+2
3,9,4,8,2	+0
9,4,8,2,6	+2
4,8,2,6,1	+2

final status: 4 8 2 6 1

total IOs: 13

MRU:

0	+1
0,1	+1
0,1,2	+1
0,1,2,3	+1
0,1,2,3,4	+1
0,1,2,3,9	+2
0,1,2,3,4	+2
0,1,2,3,8	+2
0,1,3,8,2	+0
0,1,3,8,6	+2
0,3,8,6,1	+0

final status: 0 3 8 6 1

total IOs: 13

They have same number of IOs.

Problem 4

Hash index are like data blocks, it is good for direct search (operators = or !=) because it can directly point to which data block the result belongs to. B-tree has many blocks that lead to different directions according to the key. Therefore B-tree is better for undirect operators (Like) or partial keys. In this specific question, when we have a query with WHERE product_name LIKE ("abc%"), The B-tree may first decide the branch with ‘a’ falling into its range. At the next level(s), it decides the branch corresponds to ‘b’ falling into its branch, etc.

Therefore, I would use B-tree index for product_name.

Problem 5

Nested Loop Join:

Nested loop join joins two sets by using two nested loops. For tuple-based nested loop join, it contains loops ranging over individual tuples of the relations involved. Block-based nested loop join can be regarded as a variation of tuple-based nested loop join. In addition, it organizes access to both argument relations by blocks and using as much main memory as possible for storing tuples of the outer loop relation to reduce the numbers of scans.

Two-Pass Join Algorithm based on sorting:

For two-pass algorithms, data from the operand relations is read into main memory, processed in some way, written out to disk and reread from disk to complete the operation. Here, two just means the number of times data is read into main memory. Based on sorting means that we apply two-pass algorithm on sorted tuples (could be sorted into sublist, set of blocks, etc).

Index-Base Join Algorithm:

As the name suggested, we apply join algorithm by using index. More specifically, we used index to help find tuples in S that join with tuple in R. Index just provides a way for us to extract tuples faster.