COMP 3005: Database Management Systems – Assignment 4 – Andy Chia – 101111058

|  |  |
| --- | --- |
| 1. | In variable-length record representation, the record starts with offset and length pairs of variable-size attributes, followed by fixed-size attributes, then the null bitmap, and finally the variable-size attributes. How can we improve this representation if our application is expected to store tables with large number of attributes, most of which are nulls?  store the null bitmap at the beginning then store the fixed-sized attributes then the two variable-size attributes. this should be efficient in space. |

|  |  |
| --- | --- |
| 2. | Consider the following arrangement for four disks, where Bi is a data block, and Pi is the parity block for the 4 data blocks that precedes it. What problem will this arrangement cause?  A lost of data will occur if by any chance a disk gets damaged since the parity block is not found on B1 to B4 of disk 1 through 4. |

|  |  |  |
| --- | --- | --- |
| 3. |  |  |
|  | (a) | Delete record 9.  After deleting record 9, we will modify record 6 (the previously deleted record on the list) to point to the address of record 9 marking it as empty and to make record 9 point to a null. |
|  | (b) | Insert (20000, Jamie, Physics, 100000).  Find where we can add the entry, if there s a free space put the entry there. if not then we must insert it in another overflow block and update the pointer chain    (I’m sorry) |

|  |  |  |
| --- | --- | --- |
| 4. |  | Construct a B+-tree for the following set of key values: (2, 3, 5, 7, 11, 17, 19, 23, 29, 31). The tree is initially empty and values are added one value at a time in ascending order. Consider the following values of n: |
|  | (a) | n = 4 |
|  | (b) | n = 6 |
|  | (c) | n = 8 |

|  |  |  |
| --- | --- | --- |
| 3. |  |  |
|  | (a) | Delete 23 |
|  | (b) | Delete 19 |

|  |  |
| --- | --- |
| 4. |  |