**Assignment#3**

**MSIT 630 Database Systems (Summer, 2019)**

**Total: 40 points**

**Due: 7/7/2019 11:59PM**

**1. What is an SQL injection attack? Explain how it works, and what precautions must be taken to prevent SQL injection attacks. (4 points)**

SQL injection is the process of injecting SQL code into entry fields to perform nefarious actions on a data-driven application or database. A good way to mitigate risk of a SQL injection attack is the use of prepared statements (Sometimes called parameterized queries). The Prepared statement allows you to set parameters to your statements. For instance for a form excepting your name, you would set the name variable to be a parameter in your prepared statement. Essentially you can create variables with logic to set limits for your inputs mitigating invalid inputs that could be SQL injection.

**2. What are two advantages of encrypting data stored in the database? (4 points)**

The advantages of encrypting data stored in the database is that it provides security for data at all times whether it is in transit or at rest it will be encrypted and not readable to the naked eye. This is also important because many organizations require databases to be encrypted such as being HIPPA compliant. This keeps data private so that even DBA’s can’t see your personal data in the database. For example passwords, if your passwords weren’t encrypted then anyone with access to the database could view your passwords in clear text. Another benefit of encrypted data is the concept of integrity. Not only does it keep the information confidential but it also protects the integrity of the data. If the data is encrypted it is much more difficult for attackers to alter the data and if they do manage to alter the data there is a good chance the change can be detected. This is why data encryption on a database both helps protects confidentiality and integrity.

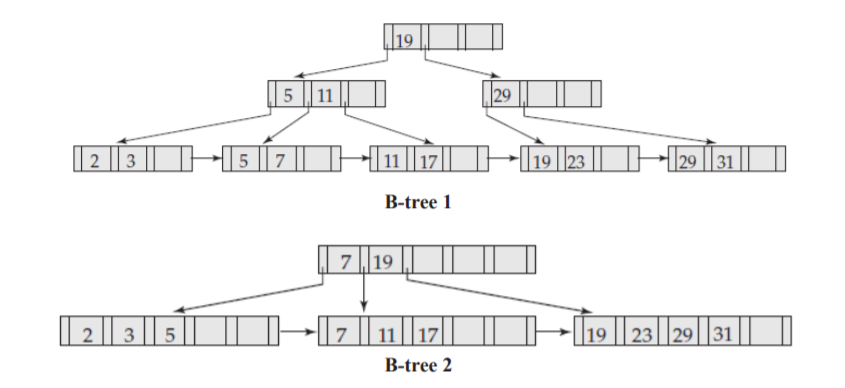
**3. RAID systems typically allow you to replace failed disks without stopping access to the system. Thus, the data in the failed disk must be rebuilt and written to the replacement disk while the system is in operation. Which of the RAID levels yields the least amount of interference between the rebuild and ongoing disk accesses? Explain your answer. (4 points)**

RAID level 1 has the least amount of interference between the rebuild and ongoing disk accesses. This is because RAID 1 uses the concept of Disk Mirroring. Disk Mirroring replicates the data to two or more disks simultaneously. So, if one disk were to fail another exact copy would be available. RAID 1 copies only the data from the failed disk mirror during a rebuild period where the other Raid levels will copy all the content of the other disks. While RAID 1 can offer near instantaneous uptimes it also requires at least 2 separate disks for the mirroring. For instance, if you need to store 1TB of data you will need at least 2TB to fulfill the minimum requirement of 1 additional disk that you can mirror too. Raid 1 yields the least interference but can be costly especially when scaling up.

**4. In the sequential file organization, why is an overflow block used even if there is, at the**

**moment, only one overflow record? (4 points)**

An overflow block is used in sequential file organization because a block is the smallest measurement that is read from the disk. Allowing blocks to contain multiple files in its records would be less cost effective then the saved spaced from allocated disk storage. On top of that the free space in the overflow block can be used for other future insertions. Basically, it is more effective to use the overflow block because it is the smallest usable unit that can be read from the disk then inserting and resequencing your original block.

**5. For each of the following two B+ trees, show the steps involved in the following queries: (10 points) (Note: there are two B+ trees. You are supposed to answer question #a and #b for both trees).** 

**a. Find records with a search-key value of 7.**

**B-Tree 1:**

First level - 7 is not present and is less than 19 so follow the left pointer.

Second level (first leaf node.) – 7 is not present and is between the left and right values of 5 and 11. Follow middle pointer.

Third level (second leaf node.) – Key value 7 is present.

**B-Tree 2:**

First level - key value 7 is present. Follow middle pointer.

Second level (first lead node.) – Key value 7 is present.

**b. Find records with a search-key value between 3 and 17, inclusive.**

**B-Tree 1:**

First level - 3 is not present and is less than 19 so follow left pointer.

Second level (first leaf node.) – 3 is not present and is below 5. Follow left pointer.

Third level (first leaf node.) – Key value 3 is present. Continue for 17. Follow right pointer.

Third level (second leaf node.) – 17 not present. Follow right pointer.

Third level (third leaf node.) – Key value 17 is present.

**B-Tree 2:**

First level - 3 is not present and is less than 7 so follow left pointer.

Second level (first leaf node.) –Key value 3 is present. Continue for 17. Follow right pointer.

Second level (second leaf node.) – Key value 17 is present.

**6. What are the causes of bucket overflow in a hash file organization? What can be done to?**

**Reduce the occurrence of bucket overflows? (4 points)**

Bucket overflow can occur simply from their being an insufficient amount of buckets. Whoever designed the system under estimated the number of buckets that would need to be allocated. Another reason is bucket skew. Some buckets have more records then others. So some buckets might have space while others overflow. This is a bucket skew. They bucket skew can occur from multiple entries having the same search key or by choosing a hashing function that was not truly random or uniform. You can reduce the occurrence of bucket overflow by essentially wasting space and allocating more space in the buckets even if they remain empty most of the time. This reduces the occurrence. Another option is the use of overflow buckets. If the bucket is full it will assign an overflow bucket for that bucket and keep chaining these overflow buckets together in a linked list know as overflow chaining.

**7. Why is a hash structure not the best choice for a search key on which range queries are likely?(4 points)**

A hash structure is not ideal for a search key for range queries because a good hashing algorithm randomly assigns values to buckets. It does not go in sequential order and values are spread around at random through the buckets. The values in the specified range will be spread out across multiple buckets and we will have to read more buckets to find the search keys. So essentially because hashes don’t go to the next bucket in sorted order they are not ideal for range queries because of their randomization.

**8. Suppose you need to sort a relation of 50 gigabytes, with 4 kilobyte blocks, using a memory size of 40 megabytes. Suppose the cost of a seek is 4 milliseconds, while the disk transfer rate is 50 megabytes per second. (6 points)**

**a. Find the cost of sorting the relation, in seconds, with bb = 100.**

Blocks containing records = br.

br = 50GB / 4KB = 12,500,000 blocks

Blocks in memory buffer=M

M = 40MB / 4KB = 10,000 blocks

The initial number of runs = (br / M) = 12,500,000 / 10,000 = 1250

The number of merge passes required = (logm-1(br / M)) = (log99991250) = 0.774

So you will need 1 merge pass.

Block transfers = br(2\*1 + 1) = 12,500,000 \* 3 = 37,500,000 blocks

Seeks = 2 (br / M) + (br / bb) (2 \* 1 - 1) = 2500 + (12,500,000 / 100)(1)

When bb = 100

Seek = 127,500

Total sorting cost in seconds (bb) = (block transfers) \* 4KB/50MB +

(Number of seeks) \* 4/1250

bb = 37,500,000 \* 4KB / 50MB + 127,500 \* 4/1250 = 3000 +

The cost of sorting the relation is 3408sec.‬

**b. How many merge passes are required?**

Just one merge pass.