Homework Assignment 1

Databases II

**Data Storage**

**Exercise 1.** Explain RAID levels 0, 1, 0+1, and 2. Explain which one uses memory space more efficiently, and which one provides higher reliability. (15 marks)

RAID 0 uses data stripping with no redundant information stored. This allows for low costs but poor reliability. The MFFT, which reflects on the reliability, linearly decreases with the number of disk drives in the array. Although it lacks in reliability, it makes up in being the most efficient RAID in regard to memory space as no redundant information is stored on the disk.

RAID 1 uses mirroring. This means there are 2 identical copies of the information stored on 2 separate disks. This results in very poor uses of memory space and efficiency. Every write on the disk needs to be done first on one disk and then subsequentially on the next. This is to ensure in the case of a system failure we are not left with 2 corrupt disks. Due to the feature, it is a very reliable form of data storage.

RAID 0+1 combines both data stripping and mirroring. All the data is duplicated and before being written on a disk it is also stripped. This provides the same level of reliability as RAID 1 as we have a duplicated set of data in cause of system failure, but it is still very poor on memory space efficiency due to that data redundancy.

RAID 2 makes use of a single bit striping unit in combination with Hamming error correction code. Two disks are needed, one to write the data and one for the error correction codes (ECC). This level is good for work that requires a number of data requests as for each request an aggregated bandwidth of all the disks is used. Overall, this level is reliable but not memory space efficient due to the error checking disks needed.

**Exercise 2.** Consider a disk with p double-sided platters, n tracks per side of its platters, and s sectors per track. If a block is b sectors, how many blocks can be stored on this disk? (10 marks)

First we would need to calculate the number of sectors on each side,

Number of sectors per side = 𝑛 × 𝑠

Then the multiply by 2 because it is a double-sided platter,

Number of sectors per platter = 2 × 𝑛 × 𝑠

Multiplying that by the number of platters gives us,

Number of sectors in one disk = 2 × 𝑛 × 𝑠 × 𝑝

Therefore, the number of blocks that can be stored on this disk are,

**2𝑛𝑠𝑝 blocks**

**Exercise 3.** Consider a disk with n tracks. Assume that moving the disk head between two adjacent tracks takes a fixed time t. Prove that the average seek time for reading a block is t × n 3 if the block has the same probability to be in any track on the disk. (10 marks, bonus)

**Exercise 4.** Consider a disk with n tracks, the average rotational delay of r ms, and the transfer delay of f ms. Answer the following questions assuming that moving the disk head between two adjacent tracks takes t ms.

1. What is the average number of blocks I/Os per second? (10 marks) Hint: Use the average seek time from Ex. 3.

Given,

Average seek time = t x (ms)

Block access time for 1 block is,

t x + 𝑟 + 𝑓 (ms)

Convert from millisecond to second,

1. Explain that using two disks as mirrors can reduce the average seek time by half, but it does not change the transfer delay for a block. (10 marks, bonus)
2. Compute the average number of blocks I/Os per second if we use two disks as mirrors with the same data. (10 marks)

Using 2 disks as mirrors means cutting the average seek time in half,

Average seek time = t x (ms)

Therefore average number of block accesses would be,

**Indexing**

**Exercise 5.** Draw the tree obtained from the B+ tree in Figure 1 after the following operations (for each sequence of operations start from the tree in the figure):

1. Diagram, engineering drawing

   Description automatically generatedInsert 21? and 12? without redistribution. (5 marks)

**7\***



1. Insert 21? and 12? with redistribution. (5 marks)

Diagram, engineering drawing

Description automatically generated

**7\***



Diagram

Description automatically generatedc. Remove 10? and 15? with redistribution. (5 marks)

**7\***



**Exercise 6.** Create an extensible hashing, insert the search keys 3, 8, 10, 20, 5, 8, 6, 1, 9, 15, 14, 13, 2, 17, 18 in order, and report the result. Apply the same hash function used during the lessons. Assume the maximum number of keys in a primary pages is three. (15 marks)

Diagram

Description automatically generatedInsert 3, 8, 10, 20, 5, 6, 1, 9, 15, 14:

Insert 13:

Diagram

Description automatically generated

Insert 2, 17, 18:

Diagram

Description automatically generated

**Exercise 7.** Create a linear hashing, insert the search keys in Ex. 6 in the same order, and report the result. Apply the same family of hash functions used during the lessons. (15 marks)

Insert 3, 8, 10, 20, 5, 8, 6, 1, 9, 15, 14:

A picture containing diagram

Description automatically generated

Insert 13:

Diagram

Description automatically generated

Insert 2, 17:

A picture containing diagram

Description automatically generated

Insert 18:

A picture containing table

Description automatically generated

**PostgreSQL**

**Exercise 8.** Run the following query and report the result (10 marks). Compute the estimation error for the number of rows in the result, i.e. the difference between the actual plan rows and the plan rows using ‘analyze’ (10 marks):

*Output:*

QUERY PLAN

----------------------------------------

[ +

{ +

"Plan": { +

"Node Type": "Seq Scan", +

"Parallel Aware": false, +

"Relation Name": "counties", +

"Alias": "counties", +

"Startup Cost": 0.00, +

"Total Cost": 20041.34, +

"Plan Rows": 240764, +

"Plan Width": 45, +

"Actual Startup Time": 1.177, +

"Actual Total Time": 185.487, +

"Actual Rows": 239233, +

"Actual Loops": 1, +

"Filter": "(cases > 1000)", +

"Rows Removed by Filter": 674794+

}, +

"Planning Time": 0.420, +

"Triggers": [ +

], +

"Execution Time": 199.094 +

} +

]

(1 row)

Estimation Error = | 239233 – 240764 | = 1531 rows.

**Exercise 9.** Use the following command to see the indexes defined on counties and report the result (5 marks):

*Output:*

schemaname | tablename | indexname | tablespace | indexdef

------------+-----------+------------+------------+---------------------------------------------

public | counties | casesindex | | CREATE INDEX casesindex ON public.counties USING btree (cases)

(1 row)

Run the last command in Ex. 8 again, report the result, and explain the difference after adding the index index (10 marks).

*Output:*

QUERY PLAN

---------------------------------------------

[ +

{ +

"Plan": { +

"Node Type": "Bitmap Heap Scan", +

"Parallel Aware": false, +

"Relation Name": "counties", +

"Alias": "counties", +

"Startup Cost": 2771.86, +

"Total Cost": 14386.95, +

"Plan Rows": 239927, +

"Plan Width": 45, +

"Actual Startup Time": 42.135, +

"Actual Total Time": 121.040, +

"Actual Rows": 239233, +

"Actual Loops": 1, +

"Recheck Cond": "(cases > 1000)", +

"Rows Removed by Index Recheck": 0, +

"Exact Heap Blocks": 8276, +

"Lossy Heap Blocks": 0, +

"Plans": [ +

{ +

"Node Type": "Bitmap Index Scan",+

"Parent Relationship": "Outer", +

"Parallel Aware": false, +

"Index Name": "casesindex", +

"Startup Cost": 0.00, +

"Total Cost": 2711.88, +

"Plan Rows": 239927, +

"Plan Width": 0, +

"Actual Startup Time": 40.221, +

"Actual Total Time": 40.222, +

"Actual Rows": 239233, +

"Actual Loops": 1, +

"Index Cond": "(cases > 1000)" +

} +

] +

}, +

"Planning Time": 0.626, +

"Triggers": [ +

], +

"Execution Time": 133.362 +

} +

]

(1 row)

After adding the index, the node type switched from ‘Seq Scan’ to ‘Bitmap Heap Scan’. This reduced the execution time as the query does not have to search through all the data entries and instead only visits the disks it needs to.