BAN 610 Problem set 3 – Indexing, Query Optimizer, and Concurrency

Edit your submission in this word document, attaching the screenshots of the codes used for each question. Include narrative descriptions, outputs screenshot, or short answers when requested.

**Task 1**

Suppose we have the following two tables.

TA

|  |  |  |
| --- | --- | --- |
| NetID | Salary | Dept |
| 123 | 100000 | Mgmt |
| 321 | 200000 | Fin |
| 456 | 150000 | Ops |
| 678 | 120000 | Ops |
| 789 | 310000 | Fin |

DeptContact

|  |  |
| --- | --- |
| Dept | Phone |
| Mgmt | 123 |
| Ops | 1232 |
| Fin | 12314 |

Please implement the following relational algebra using SQL code

**Answers:**

Select \* from TA

where salary > 150000

2.

Select NetID, Dept

from TA

Select NetID, Dept

from TA

where salary > 150000

4.

Select \*

from TA

inner join DeptContact

on TA.Dept = DeptContact.Dept

where DeptContact.Dept in (“Fin”, “Ops”)

**Task 2**

Suppose we have 100,000 records in a table, and each record takes up 50 bytes. We use a two-level sparse index to index this table and each index takes up 5 bytes. Suppose one block is capable of storing 500 bytes. Please answer (1) how many indexes are there in each level of index, (2) how many blocks each level of index takes up, and (3) how many blocks we need to search to locate one record for each level index.

|  |  |  |
| --- | --- | --- |
|  | How many indexes are there in the table? | How many blocks does each level of index takes up? |
| First level index | 10,000 | 100 |
| Second level index | 100 | 1 |

**Data:**

Total records = 100,000

Each record takes up 50 bytes. So, 100,000 \* 50 = 50,00,000

Each block is capable of storing 500 bytes.

So, to store 50,00,000, we need 50,00,000 / 500 = 10,000 blocks.

Therefore, **10,000 blocks** are needed to store the data records.

**Index Pointers:**

Each Index takes up 5 bytes and each block is capable of storing 500 bytes.

No of Index pointers in 1 block = 500 /5 = 100

**First Level Index:**

First level Index entries = **10,000**

So, for 10,000 data records, we need 10,000 / 100 = **100 blocks**.

**Second level Index:**

Second level index entries = **100**

So, we need 100 / 100 = **1 block**

**(3) how many blocks we need to search to locate one record for each level index:**

First level Index = No of blocks for First level Index entries + actual data record

= 100 + 1 = 101 blocks

So, we need to search **101 blocks to locate for one record for First level index**.

Second level Index = No of blocks for second level Index entries

+ No of blocks for First level Index entries

+ actual data record

= 1 + 1 + 1 = 3

So, we need to search **3 blocks to locate for one record using second level index**.

**Task 3**

Please store the keys 2,3,5,8,12,18,22,28 using closed addressing. Use the hash function: h(x)=x%10. Compete the following table. If multiple keys are stored with the same hash, use “->” to indicate linked list.

|  |  |
| --- | --- |
| Hash | Key |
| 0 |  |
| 1 |  |
| 2 | 2 -> 12 -> 22 |
| 3 | 3 |
| 4 |  |
| 5 | 5 |
| 6 |  |
| 7 |  |
| 8 | 8 -> 18 -> 28 |
| 9 |  |

**Task 4**

Store the keys 2,3,5,8,12,18,22,28 using linear probing. Use the hash function: h(x)=x%10. Complete the following table.

|  |  |
| --- | --- |
| Hash | Key |
| 0 | 28 |
| 1 |  |
| 2 | 2 |
| 3 | 3 |
| 4 | 12 |
| 5 | 5 |
| 6 | 22 |
| 7 |  |
| 8 | 8 |
| 9 | 18 |

**Task 5**

Store the keys 2,3,5,8,12,18,22,28 in an order 3 B tree. Please draw the tree, you can use any software or draw by hand. The correct answer is not unique.

**Answer:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | 8 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | 3 |  |  |  | 18 |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | 2 |  | 5 |  | 12 |  | 22 | 28 |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**Task 6**

Suppose we have the following two tables.

TA

|  |  |  |
| --- | --- | --- |
| NetID | Salary | Dept |
| 123 | 100000 | Mgmt |
| 321 | 200000 | Fin |
| 456 | 150000 | Ops |
| 678 | 120000 | Ops |
| 789 | 310000 | Fin |

DeptContact

|  |  |
| --- | --- |
| Dept | Phone |
| Mgmt | 123 |
| Ops | 1232 |
| Fin | 12314 |

Please answer if the following two query plans return the same result. And if yes, which query plan runs faster? Please briefly explain your answer.

Plan 1:

Plan 2:

**Answer:**

Plan 1:

Select \* from

TA

inner join DeptContact

on TA.Dept = DeptContact.Dept

where TA.Salary > 150000

Plan 2:

Select \* from

(select \* from TA where salary > 150000) as T

inner join DeptContact

on T.Dept = DeptContact.Dept

* The two query plans return the same result however **query plan 2 runs faster**
* **Query plan 1:** Query plan 1 joins the tables without any condition with all the 5 rows and then apply condition, salary > 150000. so, it takes more time than query plan 2
* **Query plan 2:** Query plan 2 runs faster because, it applies condition salary > 150000 on table TA before join and hence we proceed with only 2 records that satisfies this condition and then join with the table DeptContact
* We discard the unwanted records in query plan 2 before the inner join and hence it runs faster compared to query plan 1.

**Task 7**

Suppose we have the following two tables.

TA

|  |  |  |
| --- | --- | --- |
| NetID | Salary | Dept |
| 123 | 100000 | Mgmt |
| 321 | 200000 | Fin |
| 456 | 150000 | Ops |
| 678 | 120000 | Ops |
| 789 | 310000 | Fin |

DeptContact

|  |  |
| --- | --- |
| Dept | Phone |
| Mgmt | 123 |
| Ops | 1232 |
| Fin | 12314 |

Please answer if the following two query plans return the same result. And if yes, which query plan runs faster? Please briefly explain your answer.

Plan 1:

Plan 2: *(())*

**Answer:**

Plan 1:

select NetID from TA

where Salary > 150000

Plan 2: *(())*

select NetID from

(select NetID, Salary from TA where Salary > 150000) as T

* The two query plans return the same result. however, query plan 2 runs faster.
* **Query plan 1:** we are retrieving the whole NetIDs from TA table which are 5 records and running a query on the whole list to pick the 2 results (NetIDs with salary > 150000)
* **Query plan 2:** we are first finding the 2 records of NetIDs with salary > 150000 and then retrieving only those 2 records as final
* Therefore, Plan 2 runs faster compared to Plan 1 since it discards the unwanted records prior to the retrieval.

**Task 8**

We have the following table.

TA

|  |  |  |
| --- | --- | --- |
| NetID | Salary | Dept |
| 123 | 100000 | Mgmt |
| 321 | 200000 | Fin |
| 456 | 150000 | Ops |
| 678 | 120000 | Ops |
| 789 | 310000 | Fin |

NetID takes 20 bytes, salary takes 6 bytes, Dept takes 10 bytes,

Please calculate the following quantities:

T(TA), S(TA), V(TA,NetID), V(TA, Salary), V(TA, Dept)

**Answer:**

* T(TA) = 5
* S(TA) = 20 + 6 + 10 = 36 bytes
* V(TA,NetID) = 5
* V(TA, Salary) = 5
* V(TA, Dept) = 3

**Task 9**

Please use the hash function h(x)=x%10 to store the keys: 1,12,22,10,8,7,31,42,35 using Robin Hood probing. Please also show the number of searches required to locate each key using the following format: (x,y), where x is the key, and y is the number of searches. Complete the following table.

|  |  |
| --- | --- |
| Hash | Key |
| 0 | (10,0) |
| 1 | (1,0) |
| 2 | (31,1) |
| 3 | (22,1) |
| 4 | (12,2) |
| 5 | (42,3) |
| 6 | (35,1) |
| 7 | (7,0) |
| 8 | (8,0) |
| 9 |  |

**Task 10**

Please draw the precedence plot for the following transaction schedule

S = w1(A)r2(A)r3(A)w4(A)r1(A)

Is this schedule serializable? If yes, what is the equivalent serial schedule?

**Answer: The schedule is not serializable as we observe the circle formation between T1 & T4**



**Task 11**

Is each of the following schedule 2PL? If yes, is it serializable?

**Answer: None of them follows 2PL and all are serializable.**

* In S1, we have shared lock l-s1(A) & l-s2(A) followed by l-x1(A). In this case, when l-x1(A) occurs, there should not be any shared transaction. Hence doesn’t follow 2PL.
* S1 is serializable as there are no conflicts.

* In S2, we have l-x1(A) and hence we cannot have l-s2(A) before unlocking u1(A). Hence doesn’t follow 2PL.
* S2 is serializable as there are no conflicts.

* In S3, we have l-s1(A) and hence we cannot have l-x2(A) before unlocking u1(A). Hence doesn’t follow 2PL.
* S3 is serializable as there are no conflicts.

* In S4, we have l-s1(A) and hence we cannot have l-x2(A) before unlocking u1(A). Hence doesn’t follow 2PL.
* S4 is serializable as there are no conflicts.

**Task 12**

We are doing inner join for three relations R1(A,B), R2(B,C), and R3(C,D).

And we know the following:

* For R1: T(R1) = 2000, V(R1,A)=50, V(R1,B)=100
* For R2: T(R2) = 3000, V(R2,B)=200, V(R2,C)=300
* For R3: T(R3) = 1000, V(R3,C)=90, V(R3,D)=500

Which two relations should we join first to minimize the peak memory consumption (i.e., the maximum total number of records needed to stay in the memory in any time to complete the inner join)?

**Answer: Join R2 and R3 first**

Joining R1 & R2 = (2000 \* 3000) / 200 = 30000

Joining R2 & R3 = (3000 \* 1000) / 300 = 10000

We should join R2 & R3 first to minimize the peak memory consumption as this results in a smaller intermediate result compared to joining R1 and R2 first.