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 translate the following relational algebra into SQL code

select \*

from TA

where salary > 150000

select NetID, Dept

from TA

select NetID, Dept

from TA

where salary > 150000

select \*

from TA, DeptContact

where TA.Dept = DeptContact.Dept and DeptContact.Dept = ‘Fin’ or DeptContact.Dept = ‘Ops’

**Task 2**

Suppose we have 100,000 records to store, and each record takes up 50 bytes. We use a two-level index to index this database (first level index is sparse), and each index takes up 5 bytes. Suppose one block of storage is capable of storing 500 bytes. Please answer (1) how many indexes are there in each of the two index tables, (2) how many blocks does each index table contain, 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 index table contain? | How many searches are performed on the index to search for one key? |
| First level index | 10,000 | 100 | 1 |
| Second level index | 100 | 1 | 1 |

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

**First level index: 101**

**Second level index: 3**

***Data:***

Total records: 100,000

Each record: 50 bytes

1 block capacity = 500 bytes

No of records in 1 block = 500/50 = 10 records

No of block to store the whole set = 100,000/10 = 10,000 blocks

***First Level Index:***

Size of index pointer = 5 bytes

No of index entries/block = 500/5 = 100

First level indexes = *10,000*

No of blocks for 10,000 records = 10,000/100 = *100 blocks*

No of block searches = No of blocks for index entries + 1 actual data = 100 + 1 = *101*

No of searches on the index table = 1 actual data = *1 search*

***Second Level Index:***

Second level index entries = *100*

No of blocks for 100 records = 100/100 = *1 block*

No of block searches = 1 2nd level index + 1 1st level index + 1 actual data = *3*

No of searches on the index table = 1 actual data = *1 search*

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

**Task 5**

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 runs faster?

Plan 1:

Plan 2:

**Plan 2 runs faster**

If we translate these plans to the SQL queries:

Plan 1:

select \*

from TA, DeptContact

where TA.Dept = DeptContact.Dept and salary > 150000

Plan 2:

select \* from

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

where T.Dept = DeptContact.Dept

Here in Plan 1 we inner join both TA and DeptContact tables first with 5 records and take only the records where salary >150000 which means we are running the query on all the records from the output after the inner joined table.

But in Plan 2 we query the records from where salary >150000 from TA table first with 2 records and inner join only those 2 records with DeptContact table

Hence, **Plan 2 runs faster** because it discards the unwanted records before the inner join, meaning it runs the query on the less records compared to Plan 1

**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 runs faster?

Plan 1:

Plan 2: *(())*

**Plan 2 runs faster**

If we translate these plans to the SQL queries:

Plan 1:

select NetID

from TA

where salary > 150000

Plan 2: *(())*

select NetID

from (select NetID, salary from TA where salary > 150000)

In 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)

In Plan 2 we are first finding the 2 records of NetIDs with salary > 150000 and then retrieved 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:

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

T(TA) = **5**

S(TA) = 20 + 6 + 10 = **36**

V(TA, NetID) = **5**

V(TA, Salary) = **5**

V(TA, Dept) = **3**

**Task 9**

Please draw the precedence plot for the following transaction schedule

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