Question 1 (5 marks)

Consider two relations A and B. A has 10,000 tuples, and B has 200,000 tuples. Both relations store 100 tuples per a page.
Consider the following SQL statement:
SELECT * FROM A, B WHERE A.a = B.a;
We wish to evaluate a join between A and B, with an equality condition on A.a = B.a. There are 52 buffer pages available in memory for this operation. Both relations are stored as (unsorted) heap files. Neither relation has any indexes built on it. Assume that Sort-Merge Join can be done in 2 passes. Consider alternative join strategies described below and calculate the cost of each alternative. Evaluate the algorithms using the number of disk I/O's (i.e. pages) as the cost. For each strategy, provide the formulae you use to calculate your cost estimates.

a) Page-oriented Nested Loops Join. Consider A as the outer relation. (1 mark)
Cost (PNJL) = NPages(Outer) + NPages(Outer) * NPages(Inner)
A: 10000/100 = 100 pages; B: 200000/100 = 2000 pages
Cost (PNJL) = 100 + 100 * 2000 = 200100 (I/O)

b) Block-oriented Nested Loops Join. Consider A as the outer relation. (1 mark)
Cost (BNJL) = NPages(Outer) + NBlocks(Outer) * NPages(Inner)
NBlocks(Outer) = NPages(Outer)/(Blocksize-2)
NBlocks(A) = 100/(52-2) = 2
Cost (BNJL) = 100 + 2 * 2000 = 4100 (I/O)

c) Sort-Merge Join. (1 mark)
Cost (SMJ) = Sort(Outer) + Sort(Inner) + NPages(Outer) + NPages(Inner)
Sort(R) = External Sort Cost = 2*NumPasses*NPages(R)
Sort(A) = 2*2*100 = 400
Sort(B) = 2*2*2000 = 8000
Cost(SMJ) = 400 + 8000 + 100 + 2000 = 10500 (I/O)

d) Hash Join. (1 mark)
Cost (HJ) = 2 * NPages(Outer) + 2* NPages(Inner) + NPages(Outer) + NPages(Inner)
Cost (HJ) = 3 * 100 + 3 * 2000 = 6300 (I/O)

e) What would be the lowest possible I/O cost for joining A and B using any join algorithm and how much buffer space would be needed to achieve this cost? Explain briefly. (1 mark)
The cost would be lowest if the entire smaller relation is stored in memory. Read the larger relation page by page and compare each tuple in the larger relation with the smaller relation for matching tuples. The buffer space should be enough to have the entire smaller relation, and also with one page for reading every tuple in the larger relation and one page for output buffer.
Total Cost = 2000 + 100 = 2100(I/O)
The minimum number of buffer pages for this cost is 100+1+1 = 102

Question 2 (5 marks)

Consider a relation with the following schema:
Managers (id: integer, name:string, title:string, level: integer)

The Managers relation consists of 500,000 tuples stored in disk pages. The relation is stored as a simple heap file and each page stores 100 tuples. Possible titles in the Managers hierarchy are “CFO”, “CEO”, “CTO”, “Architect”, and “Team Lead”. Manager levels are ranging from 0-20. Suppose that the following SQL query is executed frequently using the given relation:

```
SELECT name
FROM Managers
WHERE title = “Architect” and level > 18;
```

Your job is to analyse the access paths given below and estimate the cost of the best access path utilizing the information given about different indexes in each part.

a) Compute the estimated result size and the reduction factors (selectivity) of this query. (1 mark)

ResultSize = NTuples(R)π i=1..n RFi

RF = 1/NKeys(Col)

RF = (High(Col) – value) / (High(Col) – Low(Col))

NTuples(Managers) = 500000; RF(title) = 1/5; RF(level) = (20-18)/(20-0) = 1/10

ResultSize = 500000 * 1/5 * 1/10 = 10000 tuples

b) Compute the estimated cost of the best access path assuming that a clustered B+ tree index on (title, level) is (the only index) available. Suppose there are 200 index pages. Discuss and calculate alternative access paths. (1 mark)

Cost(clustered B+Tree) = (NPages(I) + NPages(R))* πi=1..n RFi

Cost(clustered B+Tree) = (200 + 500000/100) * 1/5 * 1/10 = 104 (I/O)

Cost(heap scan) = 500000/100 = 5000(I/O)

Using clustered B+tree index on (title, level) would cost less than using a heap scan.

c) Compute the estimated cost of the best access path assuming that an unclustered B+ tree index on (level) is (the only index) available. Suppose there are 200 index pages. Discuss and calculate alternative access paths. (1 mark)

Cost(unclustered B+Tree) = (NPages(I) + NTuples(R))* πi=1..n RFi

Cost(unclustered B+Tree) = (200 + 500000) * 1/10 = 50020 (I/O)

Cost(heap scan) = 5000 (I/O)

Using heap scan would cost less than an unclustered B+ tree index on (level).

d) Compute the estimated cost of the best access path assuming that an unclustered Hash index on (title) is (the only index) available. Discuss and calculate alternative access paths. (1 mark)

Cost(HashIndex) = NTuples(R) * πi=1..n RFi *2.2

Cost(HashIndex) = 500000 * 1/5 * 2.2 = 220000 (I/O)

Cost(Heap Scan) = 5000 (I/O)

Using heap scan would cost less than an unclustered Hash index on (title).

e) Compute the estimated cost of the best access path assuming that an unclustered Hash index on (level) is (the only index) available. Discuss and calculate alternative access paths. (1 mark)
The hash index should not be used over a range, therefore we can only use heap scan for the full table for the query “level > 18”.

Cost(heap scan) = 5000 (I/O)

Question 3 (10 marks)

Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances. The finance sector is organized on a per department basis, i.e. each department has its own financial budget (and thus the corresponding record in the Finance relation).

Emp(eid: integer, did: integer, sal: integer, hobby: char(20))
Dept(did: integer, dname: char(20), floor: integer, phone: char(10))
Finance(did: integer, budget: real, sales: real, expenses: real)

Consider the following query:

```
SELECT D.dname, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.did=D.did AND D.did=F.did
AND E.sal ⩾ 90000 AND E.hobby = ‘bungee jumping;
```

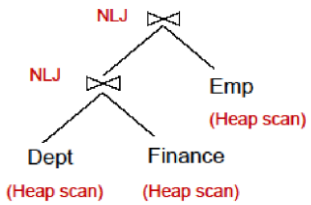
The system's statistics indicate that employee salaries range from 50,000 to 100,000, and employees enjoy 200 different hobbies. There are a total of 10,000 employees and 5,000 departments (each department with a corresponding financial record in the Finance relation) in the database. Each relation fits 100 tuples in a page. Suppose there exists a clustered B+ tree index on (Emp.did) of size 50 pages. Since selection over filtering predicates is not marked in the plans, assume it will happen later on-the-fly after all joins are performed.

a) Compute the estimated result size and the reduction factors (selectivity) of this query (2 marks)

ResultSize = $\pi_{j=1..k}$ NTuples(Rj)$\pi_{i=1..n}$ RFi
RF = 1/NKeys(Col)
RF = (High(Col) – value) / (High(Col) – Low(Col))
RF(sal) = (100000 - 90000)/(100000 - 50000) = 1/5
RF(hobby) = 1/200
RF(E.did = D.did) = 1/5000
RF(D.did = F.did) = 1/5000
ResultSize = 10000 * 5000 * 5000 * 1/5000 * 1/5000 * 1/5 * 1/200 = 10 tuples

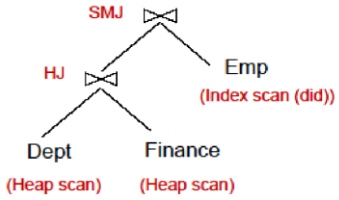
b) Compute the cost of the plans shown below. Assume that sorting of any relation (if required) can be done in 2 passes. NLJ is a Page-oriented Nested Loops Join. Assume that did is the candidate key, and that 100 tuples of a resulting join between Emp and Dept fit in a page. Similarly, 100 tuples of a resulting join between Finance and Dept fit in a page. (8 marks, 2 marks per plan)

1)



1) Number of resulting tuples for DEPT JOIN FINANCE = 1/Keys(I) * NTuples(R) * NTuples(S)
Number of resulting tuples for DEPT JOIN FINANCE = 1/5000 * 5000 * 5000 = 5000 tuples
Number of pages for DEPT JOIN FINANCE = 5000/100 = 50 pages
Cost of scanning Dept = 5000/100 = 50 (I/O)
Cost to join with Finance = 50 * 5000/100 = 2500 (I/O)
Cost to join with Emp = 50 * 10000/100 = 5000(I/O)
Total Cost = 50 + 2500 + 5000 = 7550(I/O)

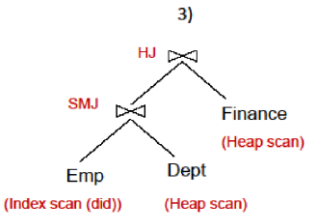
2)



2) Number of resulting tuples for DEPT JOIN FINANCE = 1/Keys(I) * NTuples(R) * NTuples(S)
Number of resulting tuples for DEPT JOIN FINANCE = 1/5000 * 5000 * 5000 = 5000 tuples
Number of pages for DEPT JOIN FINANCE = 5000/100 = 50 pages
Cost(HJ) = 3 * (NPages(Outer) + NPages(Inner))
Cost to join with Finance = 3 * (5000/100 + 5000/100) = 300 (I/O)

Cost (SMJ) = Sort(Outer) + Sort(Inner) + NPages(Outer) + NPages(Inner)
Sort(R) = External Sort Cost = 2*NumPasses*NPages(R)
Sort(DEPT JOIN FINANCE) = 2 * 2 * 50 = 200(I/O)
Cost(JOIN Emp) = Sort(DEPT JOIN FINANCE) + Read(Emp through index) + Read(DEPT JOIN FINANCE)
= Sort(DEPT JOIN FINANCE) + NPages(I(Emp.did)) + NPages(Emp)
= 200 + 50 + 10000/100
= 350(I/O)
Total Cost = 300 + 350 = 650 (I/O)

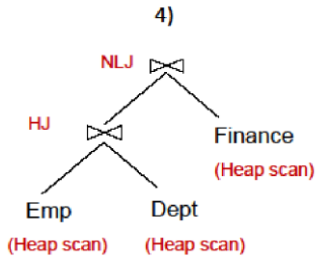
3)



Number of resulting tuples for EMP JOIN DEPT = 1/Keys(I) * NTuples(R) * NTuples(S)
Number of resulting tuples for EMP JOIN DEPT = 1/5000 * 10000 * 5000 = 10000 tuples
Number of pages for DEPT JOIN FINANCE = 10000/100 = 100 pages
Cost(EMP JOIN DEPT) = Sort(Emp) + Sort(Dept) + Read(Dept) + Read(Emp through index)
= 2 * 2 * NPages(Dept) + Sort(Emp) + NPages(I(Emp.did)) +
NPages(Emp)
= 2 * 2 * 5000/100 + 50 + 10000/100 + 50
= 400(I/O)
Cost(HJ) = 3 * (NPages(Outer) + NPages(Inner))
Cost(JOIN FINANCE) = 2 * 100 + 3 * 5000/100 = 350(I/O)

Total Cost = 400 + 350 = 750(I/O)

4)



Number of resulting tuples for EMP JOIN DEPT = 1/Keys(I) * NTuples(R) * NTuples(S)
Number of resulting tuples for EMP JOIN DEPT = 1/5000 * 10000 * 5000 = 10000 tuples
Number of pages for EMP JOIN DEPT = 10000/100 = 100 pages
Cost(HJ) = 3 * (NPages(Outer) + NPages(Inner))
Cost(Join Dept) = 3 * (10000/100 + 5000/100) = 450 (I/O)
Cost(Join Finance) = 100 * 5000/100 = 5000 (I/O)
Total cost = 450 + 5000 = 5450 (I/O)