INFO20003 Semester 2 2019 Assignment 3 Solutions

Question 1 (5 marks)

Consider two relations called Parts and Supply. Imagine that relation Parts has 60,000 tuples and Supply has 150,000 tuples. Both relations store 50 tuples per page. Consider the following SQL statement:

```
SELECT *
FROM Parts INNER JOIN Supply
  ON Parts.PartID = Supply.PID;
```

We wish to evaluate an equijoin between Supply and Parts, with an equality condition Parts.PartID = Supply.PID. There are 202 buffer pages available in memory for this operation. Both relations are stored as (unsorted) heap files. Neither relation has any indexes built on it.

Consider the 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 Parts as the outer relation. (1 mark)

NPages(Parts) = NTuples(Parts) / NTuplesPerPage(Parts)
= 60,000 / 50 = 1200 pages
NPages(Supply) = NTuples(Supply) / NTuplesPerPage(Supply)
= 150,000 / 50 = 3000 pages
Cost = NPages(Parts) + NPages(Parts) × NPages(Supply)
= 1200 + 1200 × 3000= **3,601,200** I/O

b) Block-oriented Nested Loops Join. Consider Parts as the outer relation. (1 mark)

NBlocks(Parts) = ceil$\left(\frac{NPages(Parts)}{B-2}\right)$= ceil(1200 / 200) = 6
Cost = NPages(Parts) + NBlocks(Parts) × NPages(Supply)
= 1200 + 6 × 3000 = **19,200** I/O

c) Sort-Merge Join. Assume that Sort-Merge Join can be done in 2 passes. (1 mark)

Cost = Sort(Parts) + Sort(Supply) + Merge
= 2 × NumPasses × NPages(Parts) + 2 × NumPasses × NPages(Supply) +
NPages(Parts) + NPages(Supply)
= 5 × (1200 + 3000) = **21,000** I/O

d) Hash Join. (1 mark)

Cost = 3 × NPages(Parts) + 3 × NPages(Supply)
= 3 × (1200 + 3000) = **12,600** I/O

e) What would be the lowest possible cost to perform this query, assuming that no indexes are built on any of the two relations, and assuming that sufficient buffer space is available? What would be the minimum buffer size required to achieve this cost? Explain briefly. (1 mark)

Block Nested Loops Join (or Hash Join) with *B* chosen so that the smaller table fits into memory as a single block.
Cost = 1200 + 3000 = **4200** I/O
$\left(\frac{NPages(Parts)}{B-2}\right)$= 1 ⇒ *B* = **1202** pages

Question 2 (5 marks)

Consider a relation with the following schema:

Employee (EmpID, firstname, lastname, department, salary)

The Employee relation has 1200 pages and each page stores 120 tuples. The *department* attribute can take one of six values ("Marketing", "Human Resource", "Finance", "Public Relations", "Sales and Distribution", "Operation Management") and *salary* can have values between 100,000 and 500,000 ([100,000, 500,000]).

Suppose that the following SQL query is executed frequently using the given relation:

```
SELECT *
FROM Employee
WHERE salary > 300,000 AND department = 'Marketing';
```

Your job is to analyse the query plans and estimate the cost of the *best plan* utilizing the information given about different indexes in each part.

a) Compute the estimated result size for the query, and the reduction factor of each filter. (1 mark)

$RF_{salary> 300,000} = \frac{High(salary) - Value}{High\ (salary) - Low(salary)} = \frac{500,000 - 300,000}{500,000\ - 100,000} = 0.5$

$RF_{department\ =\ 'Marketing'} = \frac{1}{NKeys(department)} = \frac{1}{6}$

Result size = NTuples(Employee) × ΠRF

= NPages(Employee) × NTuplesPerPage(Employee) × $RF_{salary>250,000}$ × $RF_{department\ =\ 'Marketing'}$

= 1200 × 120 × 1/2 × 1/6= **12,000** tuples

b) Compute the estimated cost of the *best plan* assuming that a *clustered B+ tree* index on *(department, salary)* is the only index available. Suppose there are 300 index pages. Discuss and calculate alternative plans. (1 mark)

Using clustered B+ tree on (department, salary):

Cost = $RF_{salary>300,000}$ × $RF_{department\ =\ 'Marketing'}$ × (NPages(I) + NPages(Employee))

= 1/2 × 1/6 × (300+1200) = **125** I/O

Using full table scan:

Cost = NPages(Employee) = **1200** I/O

The best plan is the **clustered B+ tree** on (department, salary) with a cost of **125** I/O.

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

Using unclustered B+ tree on (salary):
Cost = $RF_{salary>250,000}$ × (NPages(I) + NTuples(Employee))
= 1/2 × (200 + 1200 × 120) = **72,100** I/O
Using full table scan:
Cost = NPages(Employee) = **1200** I/O
The best plan is the **full table scan** with a cost of **1200** I/O.

d) Compute the estimated cost of the *best plan* assuming that an *unclustered Hash* index on *(department)* is the only index available. Discuss and calculate alternative plans. (1 mark)

Using unclustered hash on (faculty):
Cost = $RF_{department = 'Marketing'}$ × 2.2 × NTuples(Employee)
= 1/6 × 2.2 × 1200 × 120 = **52,800** I/O
Using full table scan:
Cost = NPages(Employee) = **1200** I/O
The best plan is the **full table scan** with a cost of **1200** I/O.

e) Compute the estimated cost of the *best plan* assuming that an *unclustered Hash* index on *(salary)* is the only index available. Discuss and calculate alternative plans. (1 mark)

Hash index cannot be used for range queries.
The only available plan is the **full table scan** with a cost of **1200** I/O.

Question 3 (10 marks)

Consider the following relational schema and SQL query. The schema captures information about employees, their departments and the projects they are involved in.

Employee (eid: *integer*, salary: *integer*, name: *char(30)*)

Project (projid: *integer*, code: *char(20)*, start: *date*, end: *date,* eid*: integer*)

Department (did: *integer,* projid: *integer*, budget: *real,* floor: *integer*)

Consider the following query:

```
SELECT e.name, d.projid
FROM Employee e, Project p, Department d
WHERE e.eid = p.eid AND p.projid = d.projid
  AND e.salary < 300,000 AND p.code = ‘alpha 340’;
```

The system's statistics indicate that there are 1000 different *project code* values, and *salary* of the employees range from 100,000 to 500,000 ([100,000, 500,000]). There is a total of 60,000 projects, 5,000 employees and 20,000 departments in the database. Each relation fits 100 tuples in a page. Assume *eid* is a candidate key for Employee, *projid* is a candidate key for Project, and *did* is a candidate key for the Department table. Suppose there exists a *clustered B+ tree* index on *(Project.projid)* of size 200 pages and suppose there is a *clustered B+ tree* index on *(employee.salary)* of size 10 pages.

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

$$RF_{\text{e.eid = p.eid}} = \frac{1}{NKeys\ (eid)} = \frac{1}{NTuples\ (Employee)} = \frac{1}{5000}$$

$$RF_{\text{p.projid = d.projid}} = \frac{1}{NKeys\ (projid)} = \frac{1}{NTuples(Project)} = \frac{1}{60{,}000}$$

$$RF_{\text{salary < 300,000}} = \frac{Value - Low(salary)}{High(Salary) - Low(Salary)} = \frac{300{,}000 - 100{,}000}{500{,}000 - 100{,}000} = \frac{1}{2}$$

$$RF_{\text{code = 'alpha 340'}} = \frac{1}{NKeys\ (code)} = \frac{1}{1000}$$

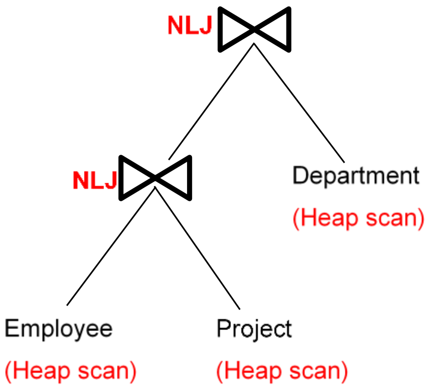
Result size = NTuples(Employee) × NTuples(Project) × NTuples(Department) × RF_{eid} × RF_{projid} × $RF_{\text{salary < 300,000}}$ × $RF_{\text{code = 'alpha 340'}}$

= 5000 × 60,000 × 20,000 × 1/5000 × 1/60,000 × 1/2 × 1/1000

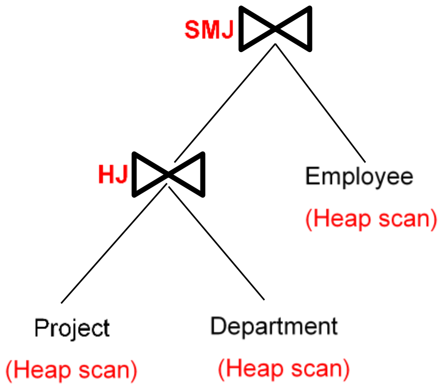
= **10** tuples

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 *eid* is the candidate key of the Employee relation, and *projid* is the candidate key of the Project relation. Assume that 100 tuples of a resulting join between Employee and Project fit in a page. Similarly, 100 tuples of a resulting join between Project and Department fit in a page. If selection over filtering predicates is not marked in the plan, assume it will happen on-the-fly after all joins are performed, as the last operation in the plan. (8 marks, 2 marks per plan)

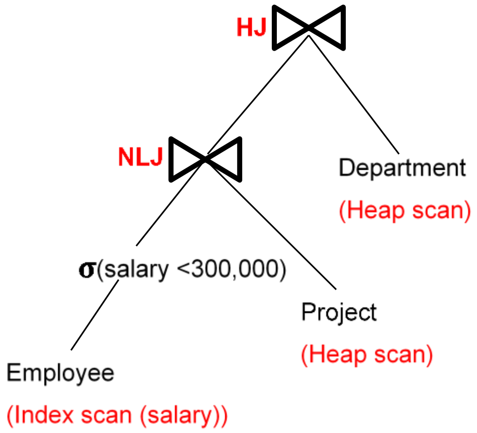
[1]



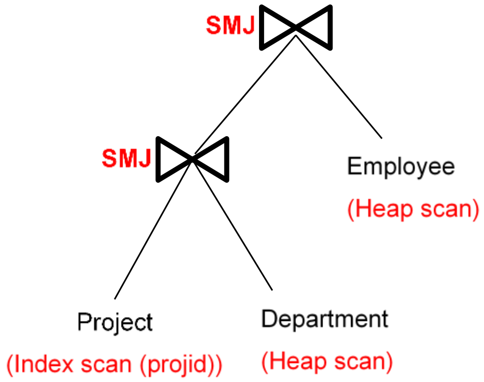
[2]



[3]



[4]



[1]

NPages(Employee) = 5000 / 100 = 50 pages
NPages(Project) = 60,000 / 100 = 600 pages
NPages(Department) = 20,000 / 100 = 200 pages

Cost(NLJ(Employee ⋈ Project) = NPages(Employee) + NPages(Employee) × NPages(Project) = 50 + 50 × 600 = 30,050 I/O

Result size(Employee ⋈ Project) = NTuples(Employee) × NTuples(Project) × 1/NKeys(eid)
= 5000 × 60,000 × 1/5000 = 60,000 tuples

NPages(Employee ⋈ Project) = 60,000 / 100 = 600 pages

Cost(NLJ(⋈Department) = NPages(Employee ⋈ Project) + NPages(Employee ⋈ Project) × NPages(Department) – NPages(Employee ⋈ Project) [due to pipelining]
= 600 × 200 = 120,000 I/O

Overall cost = 30,050 + 120,000 = **150,050** I/O

[2]

NPages(Employee) = 5000 / 100 = 50 pages
NPages(Project) = 60,000 / 100 = 600 pages
NPages(Department) = 20,000 / 100 = 200 pages

Cost(HJ(Project ⋈ Department)) = 3 × NPages(Project) + 3 × NPages(Department)
= 3 × (600 + 200) = 2400 I/O

Result size(Project ⋈ Department) = NTuples(Project) × NTuples(Department) × 1/NKeys(projid) = 60,000 × 20,000 × 1/60,000 = 20,000 tuples
NPages(Project ⋈ Department) = 20,000 / 100 = 200 pages

Cost (SMJ(⋈ Employee)) = 2 × NumPasses × NPages(Project ⋈ Department) +
2 × NumPasses × NPages(Employee) +
NPages(Project ⋈ Department) + NPages(Employee)
– NPages(Project ⋈ Department) [due to pipelining]
= 4 × 200 + 5 × 50 = 1050 I/O

Overall cost = 2400 + 1050 = **3450** I/O

[3]

Cost(I(Employee))
= $RF_{salary < 300,000}$ × (NPages(I(salary)) + NPages(Employee))
= 1/2 × (10 + 50) = 30 I/O

Result size ($\sigma_{salary < 300,000}$(Employee)) = NTuples(Employee) × $RF_{salary < 300,000}$
= 5,000 × 1/2 = 2,500 tuples
NPages ($\sigma_{salary < 300,000}$(Employee))= 2500 / 100 = 25 pages

Cost(NLJ($\sigma_{salary < 300,000}$ (Employee) ⋈ Project) = NPages ($\sigma_{salary < 300,000}$(Employee))+
NPages ($\sigma_{salary < 300,000}$(Employee))× NPages(Project)
– NPages ($\sigma_{salary < 300,000}$(Employee)))) [due to pipelining]
= 25 × 600 = 15,000 I/O

Result size($\sigma_{salary < 300,000}$ (Employee) ⋈ Project)
= NTuples($\sigma_{salary < 300,000}$ (Employee) × NTuples(Project) × 1/NKeys(eid)
= 2,500 × 60,000 × **1/5000** = 30,000 tuples
NPages($\sigma_{salary < 300,000}$ (Employee) ⋈ Project) = 30,000 / 100 = 300 pages

NOTE: 1/5000 is the reduction factor here, since although the left input has only 2500 tuples, the right side still can have possibly up to 5000 distinct EIDs – hence 5000 is the maximum between the two.

Cost (HJ(⋈Department) = 3 × NPages($\sigma_{salary < 300,000}$ (Employee) ⋈ Project) +
3 × NPages(Department)
– NPages($\sigma_{salary < 300,000}$ (Employee) ⋈ Project) [due to pipelining]
= 2 × 300 + 3 × 200 = 1,200 I/O

Overall cost = 30 + 15,000 + 1,200 = **16,230** I/O

[4]

The index on Project. projid is clustered. The data pages of Project are already sorted by projid, so there is no need to sort Project.
The first (and only) read of Project will be done through the index. There is no reduction factor, as all rows are read.

Cost(SMJ(Project ⋈ Department)) = 2 × NumPasses × NPages(Department) +
NPages(Department) + (NPages(I (projid)) + NPages(Project))
= 2 × 2 × 200 + 600 + 200 + 200 = 1,800 I/O

Result size(Project ⋈ Department) = NTuples(Project) × NTuples(Department) ×
1/NKeys(projid) = 60,000 × 20,000 × 1/60,000 = 20,000 tuples
NPages(Project ⋈ Department) = 20,000 / 100 = 200 pages

Cost (SMJ(⋈ Employee) =
2 × NumPasses × NPages(Project ⋈ Department)
+ 2 × NumPasses × NPages(Employee)) + NPages(Project ⋈ Department) +
NPages(Employee) – NPages(Project ⋈ Department) [due to pipelining]

= 2 × 2 × 200 + 2 × 2 × 50 + 200 + 50 – 200 = 4* 200 + 5*50 = 1050 I/O

Overall cost = 1,800 + 1,050 = **2,850** I/O