# Database Systems, CSCI 4380-01
# Homework # 8 Answers

**Introduction.**

This homework is worth 5.5% of your total grade. If you choose to skip it, the final exam will be worth 5.5% more. If you complete all the homeworks after Exam #2, there will be a special bonus in terms of your final grade computation. Additionally, doing this work now is essential to doing well in the final exam.

This homework requires no programming, only pen/pencil on paper computations. So you will submit your answers as a PDF file on GRADESCOPE.

**Question 1.** Estimate the cost of following operations with the information given below. All costs should be in number of pages (read or written).

Recall that when the final result is found, it is put in the output buffer. This operation has no additional cost.

```
R(A,B,C,D,E)  TUPLES(R)= 100,000   PAGES(R)= 2,000
S(F,G,A)      TUPLES(S) = 3,000,000 PAGES(S)= 6,000
```

(a) Block-nested loop join for `R` $\bowtie$ `S` with `M=101` blocks. For join ordering, choose the lowest cost one and only show the result of that join.

**Answer.** Allocate 100 pages to R and 1 page to S. R join S, read R once and S 2000/100=20 times. Total cost:

2000+6000*20=122,000 pages ( PAGES(R)+ PAGES(S)* ceiling( PAGES(R)/(M-1))

Note that the order ordering, S join R would have higher cost:

6000 + 2000*(6000/1000)= 126K pages

(b) External sorting of `S` using `M=60` blocks.

**Answer.**

Step 1: 6000/60=100 sorted groups are generated. Cost = 12,000 pages (read S once and write S once)

Step 2: Merge 100 sorted groups to 2 (100/60), write results to disk (sort is not yet completed): Cost = 12,000 pages

Step 2 (repeated): Merge 2 sorted groups into 1, read S once, Cost=6,000 pages.

Total = 30K pages

(c) External sorting of `S` using `M=100` blocks.

**Answer.**

Step 1: 6000/100=60 sorted groups are generated. Cost = 12,000 pages (read S once and write S once)

Step 2: Merge 60 sorted groups and output (60 <= 100, memory available for this operaiton), Read once: Cost = 6,000 pages

Total = 18K pages

(d) Hash join for `R` $\bowtie$ `S` with `M=101` blocks. Describe how join works and any assumptions you make in your computation.

Assume tuples in R and S will uniformly distribute to 101 blocks. If a buckets after hashing will not fit in memory, then you will use block nested loop join.

**Answer.**

Hash R and write to disk, Cost = 4,000 pages (read R once and write R once).

If hashing is perfect, we create 101 buckets, each is 2000/101 $\approx$ 20 pages wide.

Hash S and write to disk, Cost = 12,000 pages (read S once and write R once).

If hashing is perfect, we create 101 buckets, each is 6000/101 $\approx$ 60 pages wide.

Now, we need to join by only joining the same bucket from R and S (join bucket $R_i$ with $S_i$.) To do this, we need 20+60 blocks, given M=101, the join of each bucket can be done in a single pass.

Hence, the join can be accomplished in one read of S and R (PAGES(R)+PAGES(S)=8,000 pages).

Total cost = 4K+12K+8K=24K pages.

(e) Sort merge join for `R` $\bowtie$ `S` with `M=101` blocks. Sort each relation first and then join.

Note that if the number of partially sorted groups is smaller than allocated memory (in this case 101 blocks), they can be sorted and joined together in a single step.

**Answer.**

**Naive solution:**

(i) Sort R completely and write to disk, (ii) Sort S completely and write to disk, (iii) read each one once, join and output.

(i) Step 1: 2000/101=20 sorted groups, cost = 4K

Step 2: Merge 20 groups, cost = 4K

Total = 8K

(ii) Step 1: 6000/101=60 sorted groups, cost = 12K

Step 2: Merge 60 groups, cost = 12K

Total = 24K

(iii) Pages(R)+Pages(S) = 8K

Total:8+24+8 = 40K

**More accurate solution:**

After step 1 of each sort operation, the sort/merge and join steps can be combined.

R has 20 sorted groups, S has 60 sorted groups. We can read one page from each group into memory (need 80 memory blocks, we have 101 blocks), join and output.

Cost: Step 1 for R + Step 1 for S + read R once + read S once

Cost = 4K + 12K + 8K = 24K

We will accept other answers that show it is likely lower than 40K but will give most points for the naive solution.

**Question 2.** Estimate the cost of query `Q1` using different query plans. All costs should be in number of pages. Assume sufficient amount of memory is allocated for each query plan.

```
R(A,B,C,D,E)  TUPLES(R)= 100,000   PAGES(R)= 2,000
Q1: SELECT A,B FROM R WHERE C>10 AND D=25

Index I1 with three levels (root,internal,leaf) on R(C,D,A) with 800
                nodes in the leaf level

Index I2 with three levels (root,internal,leaf) on R(D,A,B,C) with 1,500
                nodes in the leaf level

Index I3 with three levels (root,internal,leaf) on R(C) with 300
                nodes in the leaf level

Index I4 with three levels (root,internal,leaf) on R(D) with 250
                nodes in the leaf level


Expected number of tuples of R for conditions:

Tuples(R.C>10)=20,000
Tuples(R.D=25)=1,000
Tuples(R.C>10 and R.D=25)=50
```

(a) Plan 1: sequential scan over `R`.

**Answer.** Pages(R)

(b) Plan 2: using index `I1`.

(c) Plan 3: using index `I2`.

(d) Plan 4: using index `I3`.

(e) Plan 5: using index `I4`.

(f) Plan 6: using index `I3` and `I4` both.

**Answer for Plans 2-6:**

Note that for each of the plans 2-6, we will look at how we will scan the index, what is the overall index cost it is. If the query is asking attributes that is not in the index, we will read each matching tuples from data pages. We will use the worst case assumption of one page for each matching tuple.

As an example, let's look one plan. The remaining answers are given in the table below without details.

Plan 2: We need to scan the index I1 for `C>10`, find all tuples that satisfy the condition `C>10 AND D=25` and the A values for each of the 50 tuples. Then these tuples need to be read from the data pages because the query is requesting attribute attribute B which is not in the index, resulting in additional 50 pages to be read.

Scanning the index for `C>10` requires the scan for 20,000 tuples. This requires scanning of $\frac{20{,}000}{100{,}000} = \frac{1}{5}$ of the leaf nodes, which is $800/5 = 160$ leaf nodes (stored in 160 pages), plus 2 additional nodes (pages) for root and one from internal level.

Another way to comptue this is as follows: Leaf nodes contain $\frac{100{,}000}{800} = 125$ tuples per page. So, to scan 20,000 tuples, we need to scan $\frac{20{,}000}{125} = 160$ leaf nodes.

| Plan | Scan index for condition | Index Scan cost | Data Page scan cost | Total Cost |
|---|---|---|---|---|
| 2 | `C>10` | 2+160 | 50 | 212 |
| 3 | `D=25` | $2 + \frac{1{,}500*1{,}000}{100{,}000}$ | 0 | 17 |
| 4 | `C>10` | $2 + \frac{300*20{,}000}{100{,}000}$ | 2K-20K(*) | 2K-20K |
| 5 | `D=25` | $2 + \frac{250*1{,}000}{100{,}000}$ | 1,000 | 1,005 |
| 6 | `C>10 and D=25` | 62 for I3, 5 for I4 | 50 | 117 |

(*) Note that for Plan 4, we will also accept 2,000 for an answer. Basically, we will read pages as they are found in the matching 20,000 tuples. This might lead to reading the same page multiple times. However disk controllers have buffers, so they will group requests and reduce this cost. The true cost is unknown, it may be higher or lower than 2,000. We will accept any well-reasoned answer.

**Question 3.** Estimate the size of the following queries:

```
Q1:  select * from games where id = 21;
Q2:  select * from contestants where gameid = 2345;
Q3:  select * from contestants where shortname = 'Gilbert';
Q4:  select * from contestants where gameid = 2345 and shortname = 'Gilbert';
Q5:  select * from games g, contestants c
     where g.gameid=c.gameid and c.shortname='Gilbert';
Q6:  select * from responses where iscorrect = True;
Q7:  select * from responses where shortname = 'Gilbert' and gameid='5881';
Q8:  select * from responses where shortname = 'Gilbert' or gameid='5881';
Q9:  select * from responses r, contestants c
     where c.shortname=r.shortname;
Q10: select * from responses r, contestants c
     where c.shortname=r.shortname and c.gameid=r.gameid;
```

using the following statistics:

```
Tuples(Games) = 10,000
Values(Games.gameid) = 10,000
Values(Games.id) = 40
Values(Games.airdate) = 10,000

Tuples(Contestants)=30,000
Values(Contestants.shortname)=3,000
Values(Contestants.fullname)=18,000
Values(Contestants.gameid)=10,000

Tuples(Responses)=740,000
Values(Responses.gameid)=10,000
Values(Responses.clueid)=520,000
Values(Responses.shortname)=3,000
Values(Responses.iscorrect)=2
```

These are close to the real statistics, but they are not identical. However, the following is a great educational exercise (not required for solving the homework). You can get the statistics for the real data and compare the estimates to the real results (just by running select count(*) queries). When do the estimates are far off from reality and why?

**Answer.**

| Query | Selectivity | Number of Tuples |
|---|---|---|
| Q1 | $\frac{1}{40}$ | 10,000* selectivity = 250 |
| Q2 | $\frac{1}{10,000}$ | 30,000* selectivity = 3 |
| Q3 | $\frac{1}{3,000}$ | 30,000* selectivity = 10 |
| Q4 | $\frac{1}{3,000*10,000}$ | 30,000* selectivity = 0-1 |
| Q5 | $\frac{1}{3,000*10,000}$ | 10,000*30,000* selectivity = 10 |
| Q6 | $\frac{1}{2}$ | 740,000* selectivity = 370,000 |
| Q7 | $\frac{1}{3,000*10,000}$ | 740,000* selectivity = 0-1 |
| Q8 | $1 - ((1 - \frac{1}{3,000}) * (1 - \frac{1}{10,000}))$ | 740,000* selectivity = 320 |
| Q9 | $\frac{1}{3,000}$ | 740,000* 30,000 selectivity = 7,400,000 |
| Q10 | $\frac{1}{3,000*10,000}$ | 740,000* 30,000 selectivity = 740 |

**Question 4.** Estimate the cost of the following query plans for the same query. Which one is the cheapest plan?

The abbreviations used:

| | |
|---|---|
| BNLJ | Block nested loop join |
| SMJ | Sort merge join |
| O-T-F | On the fly (pipelined operation) |
| Sort | External sort |
| I-O-S | Index only scan |
| SS | Sequential scan |

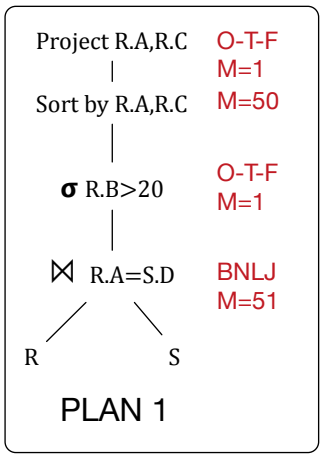
For simplicity, we will give you the size of the result of each operation. You can assume that there are appropriate projections to reduce the size of intermediate relations which are included in the given sizes.

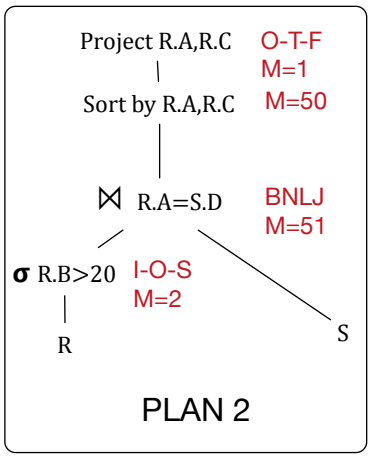
Furthermore, assume that the join operation is over a foreign key (so for each S.D, there is a single R.A value). This simply makes the computation of sort merge join much simpler after a sort.

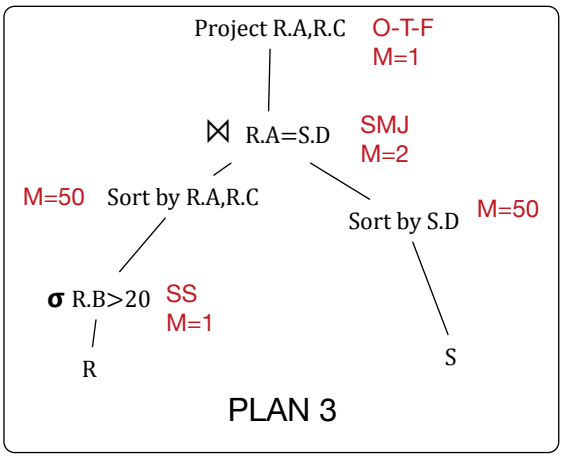
`PAGES(R) = 100`
`PAGES(S) = 800`
`PAGES(`$\sigma_{R.B>20}$ `R) = 25`
`PAGES(R` $\bowtie_{R.A=S.D}$ `S) = 700`
`PAGES((`$\sigma_{R.B>20}$ `R)` $\bowtie_{R.A=S.D}$ `S) = 175`

Index scan for Plan 2 uses an index on `R.B,R.A,R.C`. Assume that index has 2 levels, root and 100 leaf pages. The selectivity of the condition $\sigma_{R.B>20}$ R is 1/4.

Note that operations in each query plan are pipelined from lower operations to upper operations, by placing the output of one operation to the input buffer of the next operation.




**Answer.**

**Plan 1.** Block-nested loop join:

PAGES(R)+ ceiling(PAGES(R)/50)*PAGES(S)=100+2*800=1,700.

On the fly scan: No cost (done in memory)

Sorting the join results (175 pages wide as given in the question):

Step 1: No cost to read the results of the join and selection as they are produced in memory.

Sort in groups of 50 and write to disk in a temporary storage (175/50=4 sorted groups), cost = 175 pages.

Step 2: Read 175 all in 4 sorted groups, sort/merge and output.

Sort cost = 175*2=350

Total cost = 1,700+350=2,050

**Plan 2.**

Index scan to find tuples of R that satisfy `R.B>20`:

1+100/4 (leaf nodes) + 0 (no data page cost)= 26

After the selection, R tuples fit in 25 pages.

Block nested loop join: No cost to read R as tuples are produced from the index scan operation. Read 25 pages of R into memory (given 51 pages for the join) and read S once.

Join cost = 800

Sort cost = Same as in Plan 1: 350

Total cost = 26 + 800 + 350 = 1,176

**Plan 3.**

Sequential scan: 100 pages read, the result of the selection will be 25 pages wide.

Cost of sequential scan = 100

Sort R: All in memory, no additional cost of reading because it is produced by the sequential scan operation below. Sorting does not need to write results to disk because it can hold the results in memory.

Cost of sorting R = 0

Sort S: Step 1: Read and write S once, produce 800/50=16 sorted groups. Cost = 1,600

Step 2: Read S once and sort, and output to the join operation. Cost = 800

Total cost of sorting S = 2,400

Cost of join = 0 as both relations are already in memory due to sort, so the join has no additional cost.

Total cost of the plan = 2,500