Database Systems, CSCI 4380-01
Homework # 7 Answers
Due Monday May 2, 2016 at 11:59:59 PM

Introduction.

In this homework, you are allowed to work in groups of at most 2. If you do, please write the name of your group members in your solution file. Turn in a single submission (PDF or text) per group.

This homework is on the underlying mechanics of query optimization. Read carefully in each question about assumptions you can make.

```
TUPLES(R)=100,000, TUPLES(S)=2,000,000, TUPLES(T)=500,000
PAGES(R)=5,000, PAGES(S)=10,000, PAGES(T)=3,000
```

Attribute	N_DISTINCT
R.A	10,000
R.B	5,000
R.C	20
S.C	500
S.D	2,000
S.E	20
T.F	500,000
T.G	50,000
T.H	200

Question 1. Given the above statistics, find the estimated cardinality (number of tuples) of the following queries. (use 1/3 for range selectivity).

```
Q1. SELECT * FROM R WHERE R.A=10 AND R.B<4;
Q2. SELECT * FROM R,S WHERE R.B=S.C AND (S.C<4 OR S.D=5)
Q3. SELECT * FROM S WHERE S.C=S.D AND S.E=3
```

Answer.

```
Q1: 100,000 * (1/10,000) * (1/3) = 10/3 = 3

Q2: 100,000 * 2,000,000 * (1/max{5,000,500}) * ( (1-((1-1/3)*(1-1/2000)))) =
  13,346,666

Q3: 2,000,000 * (1/max{500,2,000}) * (1/20) = 50
```

Question 2. Estimate the cost of the following join orders (plans 1-3) assuming the following:

- All joins are block-nested loop joins with M=201. Inner and outer joins are fixed as shown.
- All joins are pipelined, i.e. the output of the lower join results is used as the input for the outer relation of the next join.

- Assume that after a join, you can store about 200 tuples per memory block (page).

To accomplish this, you must find the cardinality and size of the lower join in terms of number of pages to see how many times the last relation need to be read.

```
Q. SELECT * FROM R,S,T WHERE R.B=S.C AND S.D=T.F AND R.C=T.H

Plan 1: (R join S) join T
Plan 2: (R join T) join S
Plan 3: (S join T) join R
```

Answer. Let's compute size of the joins being pipelined first.

```
R join S: 100,000 * 2,000,000 * 1/5,000 = 40,000,000
          Fits in: 40,000,000/200 = 200,000 pages

R join T: 100,000 * 500,000 * 1/200 = 250,000,000
          Fits in: 250,000,000/200 = 1,250,000 pages

S join T: 2,000,000 * 500,000 * 1/500,000 = 2,000,000
          Fits in: 2,000,000/200 = 10,000 pages
```

Now, let's compute the cost of each query plan:

```
Plan 1: (R join S) join T

   R join S: 5000 + (5000/200)*10,000
   join of this with T requires that we read T: 200,000/200 times.
   Total cost: 5000 + (5000/200)*10000 + (200000/200)*3000 = 3,255,000 pages

Plan 2: (R join T) join S

    R join T: 5000 + (5000/200)*3,000
    join this with S required that we read S: 1,250,000/200 times
    Total cost: 5000 + (5000/200)*3000 + (1250000/200)*10000 = 62,580,000 pages

Plan 3: (S join T) join R

     S join T: 10000 + (10000/200)*3000
     join this with R requires that we read R: 10000/200 times
     Total cost: 10000 + (10000/200)*3000 + (10000/200)*5000 = 410,000


Of course, not asked in this question, but we could do even better with the
following query plan:

(T join S) join R
  which would cost: 3000 + (3000/200)*10000 + (10000/200)*5000 = 403,000
```

Question 3. What is the cost of sorting S given the following memory: (a) M=200, (b) M=50?

Answer. PAGES(S) = 10,000

```
M=200

   Step 1, cost: 20,000 pages creates 10,000/200 = 50 sorted groups
   Step 2, can merge all the 50 groups (50 <= 200) at once, read and
           merge, with cost 10,000 pages.

   Total cost: 30,000 pages

M=50

   Step 1, cost: 20,000 pages creates 10,000/50 = 200 sorted groups
   Step 2, cannot merge all 200 groups with only 50 memory blocks, merge
          50 groups at once, read/write, resulting in 4 sorted groups.
          Total cost: 20,000 pages
   Step 2, merge 4 groups, read and output. cost: 10,000 pages

   Total cost: 50,000 pages
```

Question 4. What is the cost of sort merge join as given in Plan 1 below for the following query, assuming you have M=201 for each join:

```
Q. SELECT * FROM R,S,T WHERE R.B=S.C AND S.D=T.F AND R.C=T.H

Plan 1: (R join S) join T
```

Note that you can combine the merge and sort steps if all the necessary blocks fit in memory. You can disregard duplicate values (whether they will all fit in memory or not).

You will sort R on R.B and S on S.D. Once you have completed merge sort for (R join S), you need to sort the results again by S.D this time (the more selective attribute). You will also sort T on T.F to match.

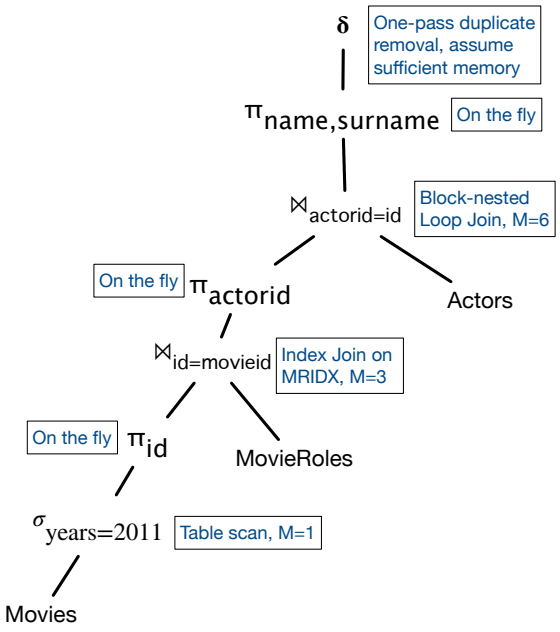
Answer.

```
Q. SELECT * FROM R,S,T WHERE R.B=S.C AND S.D=T.F AND R.C=T.H

Plan 1: (R join S) join T

    R join S: cost
    Sort R:
        Step 1: 10,000 pages (read R once, write R once), creates
        5,000/201=25 sorted groups
    Sort S:
        Step 1: 20,000 pages (read S once, write S once), creates
        10,000/201=50 sorted groups
    Step 2 for both:
        Combine merge step 2 of R and S with join (required 25+50 pages, less
        than available 201 pages), and output to the next operation.

        Cost one more read: 5,000+10,000
    Total cost of R join S: 10,000+20,000+5,000+10,000= 45,000

    We will use the same method in Question 2 to find how many pages are
```

```
    needed to store the results of R join S, i.e. 200,000 pages.

    Let's X = R join S

    Sort X:
       Step 1: No cost to read X as it is being produced in memory. Fill
       201 blocks with X tuples, sort and write.
       Total cost: 200,000 pages. Creates 1000 sorted groups

       Step 2: Read/write once to reduce 1000 sorted groups to 1000/200=5
       groups, cost 400,000 pages.

    Sort T:

       Step 1: Read/write once to create 3,000/200= 15 sorted groups, total
       cost, 6,000 pages.

       Can combine the remaining sort and merge steps (5+15 sorted groups can
       fit in 201 pages). Read each relation one more time: 200,000+3,000

       Total cost= 45,000 (R join S) + 600,000 (sort R join S) +  6,000 (sort T)
                   + 203,000 (read and join)
                 = 854,000 pages

Out of curiosity (not required for hw), let's try the best join
ordering from Question 2 instead with sort merge join only.

(T join S) join R

    T join S:
        Sort T, step 1: 6000 pages creates 15 sorted groups
        Sort S, Step 1: 20,000 pages creates 50 sorted groups
        Merge and Join, Step 2: 13,000 pages

        Total: 39,000 pages
    T join S fits in 10,000 pages

    Sort (T join S):
        Step 1: sort and write, cost: 10,000 pages, creates 50 groups

    Sort R:
        Step 1: raad, sort and write, cost: 10,000 pages, creates 25 groups

    Step 2 for both, read, merge and join (50+25 groups total) with
    additional cost: 15,000 pages.

    Total cost of (T join S) join R = 39,000 + 35,000 = 74,000 pages.
    By far the cheapest option.
```

Question 5. Finally, estimate the full cost of the above query plan and show details of your work.

This question is based on the IMDB database we used in the homeworks. All relevant details are given below. Use the estimates based on the values given below, not the actual values.



- A disk page is 1K bytes. An integer is 4 bytes and a tuple address is 12 bytes.
- TUPLES(Movies)=4,708, PAGES(Movies)=70, N_DISTINCT(Movies.years)=282
- TUPLES(MovieRoles)=265,107, PAGES(MovieRoles)=2,044, N_DISTINCT(MovieRoles.actorid)=149,374, N_DISTINCT(MovieRoles.movieid)=4,340
- Index MRIDX on MovieRoles(movieid,actorid), DEPTH=2 (root, internal and leaf nodes). Assume thought that the root is already in memory.
- TUPLES(Actors)=149,374, PAGES(Actors)=2,044

```
Movies table scan: 70 pages
    number of movies for years = 2011, 4708/282=17 tuples
    17 tuples with a single id (integer) value: 17*4=68 bytes,
    fits in a single memory block.

Join with movieroles:
    Scan index for 17 tuples, root is in memory.
    Each movie id is searched in 2 pages in the worst case. Cost= 34 pages

    Number of tuples expected: (17*265107)/4340 = 1038 tuples
    Keep for each only id values, total size: 1038*4=4,152 bytes
    4,152/1024 = 5 memory blocks

Block-nested loop join:
    All of the results of (movies join movieroles) fits in 5 pages
    Read Actors once using one block and join
    (M=6, 5 pages for outer relation and 1 page for Actors)

    Additional cost: 2,044 pages (read Actors once)

As we assume sufficient memory for duplicate removal, there is no
```

additional cost (it can be done in memory as tuples are found).

Cost of this query = 70 (movies) + 34 (index join) + 2,044 (actors)
= 2,148 pages

(sequential scan of actors is the dominant cost, but likely not too bad because sequential scans can benefit from fewer seeks.)