CMPT506 – Advanced Database System
Homework 2

Due by 4pm Sunday 8/12/2016 – Submit your softcopy to blackboard and your hardcopy at the start of the class.

1. [13 pts] Sort the following values using external merge-sort assuming that each data block stores 2 values and that 3 memory blocks are available:
   44, 55, 6, 4, 3, 7, 2, 11, 16, 8, 21, 12, 9, 5, 7, 6, 33, 22
   Show data for all passes.

Sorting Phase – Pass 0 :

Given M = 3 , each block can store 2 values then:

βr = 18 values / 2 = 9 blocks

n_r (number of initial runs) = $\lceil \frac{Br}{M} \rceil = \lceil \frac{9}{3} \rceil$ = 3 runs → This mean that the original file will be divided into 3 sub files with three blocks each.

3	4	6	7	44	55	2	8	11	12	16	21
5	6	7	9	22	23						

Merging Phase :

Number of passes in Merge phase → $\lceil \log_{M-1} nr \rceil = \lceil \log_2 3 \rceil = \lceil 1.5 \rceil = 2$

Pass 1

2, 3	4,6	7,8	11,12	16,21	44,55
5,6	7,9	22,33			

Pass 2

2, 3	4,5	6,6	7,7	8,9	11,12	16,21	22,33	44,55

2. [7 pts] Consider a file with 20,000 blocks and 5 available buffer blocks. Assume that the external sort-merge algorithm is used to sort the file.
   a. [2 pts] Calculate the initial number of runs produced in the first pass.

$$n_R = \left\lceil \frac{B_r}{M} \right\rceil = \left\lceil \frac{20{,}000}{5} \right\rceil = 4000$$

b. [2 pts] How many passes will it take to sort the file completely?

$$\#merge\ passes = \lceil \log_{23} 4000 \rceil = 6$$

$$Total\ passes = 6 + 1 = 7$$

c. [3 pts] How many buffer blocks are required to sort the file completely in two passes?

2 passes means: Pass 0 in sort and and pass 1 in merge

$$\text{Thus } 1 = \log_{M-1} \frac{B_r}{M}$$

$$M - 1 = M - 1^{\log_{M-1} \frac{B_r}{M}}$$

$$M - 1 = \frac{B_r}{M}$$

$$M^2 - M = 20{,}000$$

$$M = 142$$

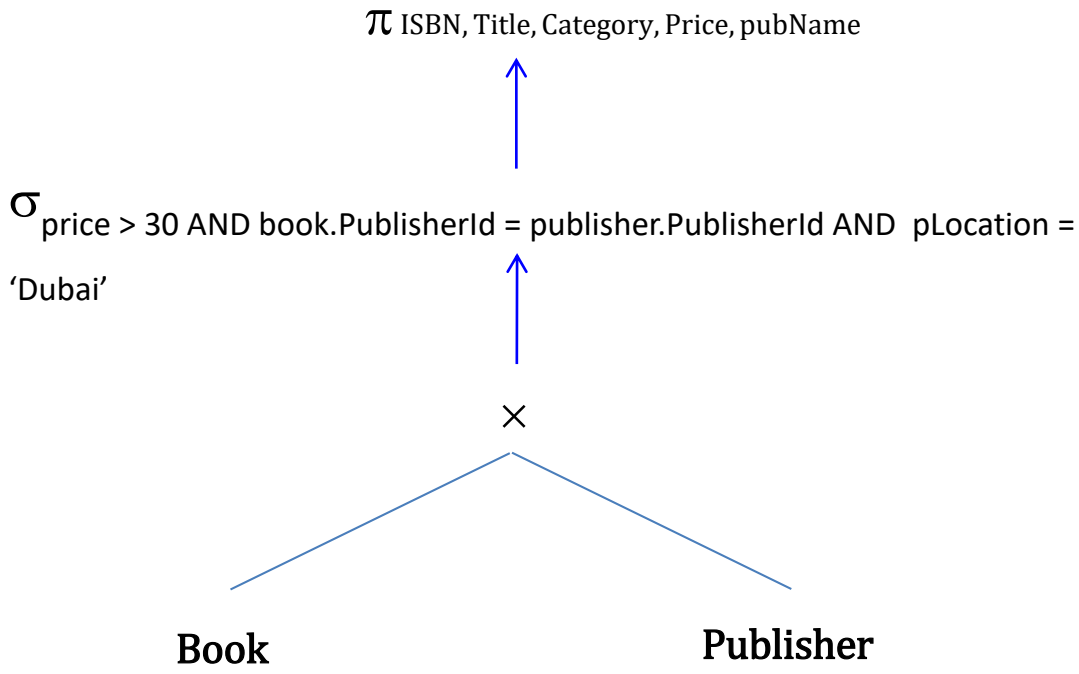
3. [20 pts] Consider the following SQL queries on Online Book Store Application.
   a. [3 pts] Transform these SQL queries into relational algebra expressions.
   b. [3 pts] Draw the initial query trees for each of these expressions.
   c. [4 pts] Derive their optimized query trees after applying heuristics on them. You can just show the last optimal query tree but briefly explain the heuristics applied to arrive to the optimal solution.

```
2.1) SELECT b.ISBN, b.Title, b.Category, b.Price, p.pubName
FROM Book b, Publisher p
WHERE p.PublisherId = b.PublisherId AND p.pLocation ='Dubai'
AND b.Price > 30;
```

Part a:

π ISBN, Title, Category, Price, pubName (σ (price > 30) $\wedge$ (book.PublisherId = publisher.PublisherId) $\wedge$ (pLocation = 'Dubai') (Book $\times$ Publisher))

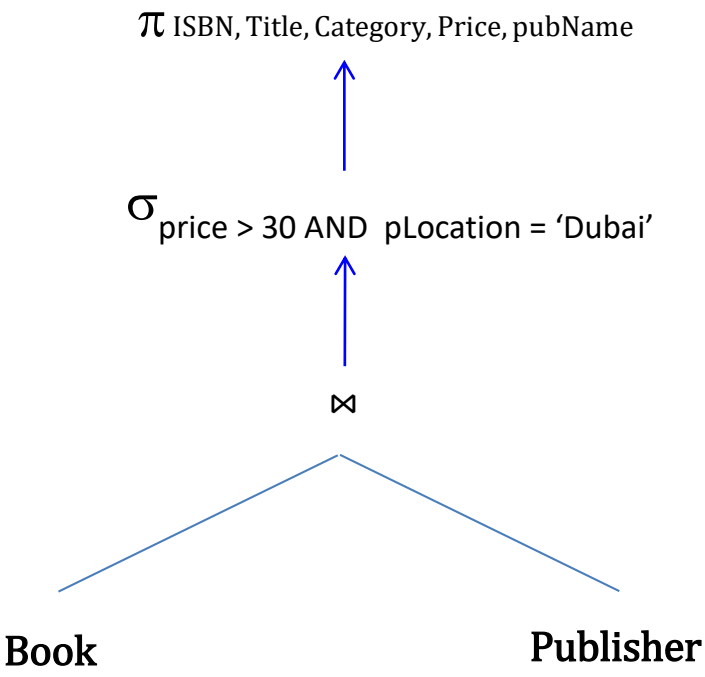
Part b:



Part c:

a. **First heuristics: Convert the Cartesian product into join**

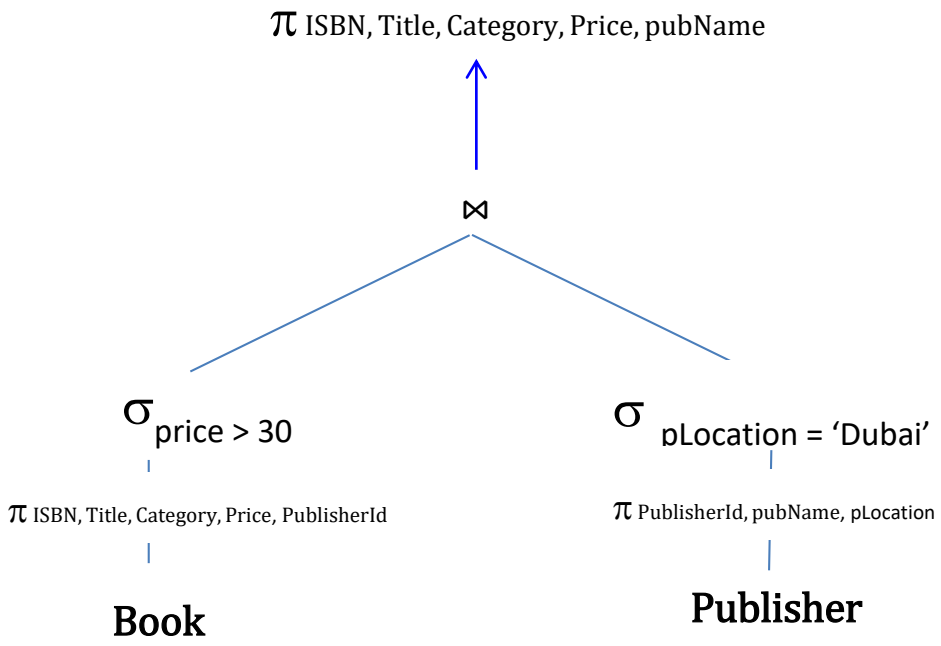
π ISBN, Title, Category, Price, pubName (σ (price > 30) ∧ (pLocation = 'Dubai') (Book ⋈ Publisher))



b. <u>Second heuristics: Push the selection and the projection down the query tree</u>

π **ISBN, Title, Category, Price, pubName**((σ (price > 30) Book) $\bowtie$ (σ (pLocation = 'Dubai' Publisher))

π **ISBN, Title, Category, Price, pubName**((σ (price > 30) π ISBN, Title, Category, Price, PublisherId (Book)) $\bowtie$ (σ (pLocation = 'Dubai' π PublisherId, pubName, pLocation (Publisher)))

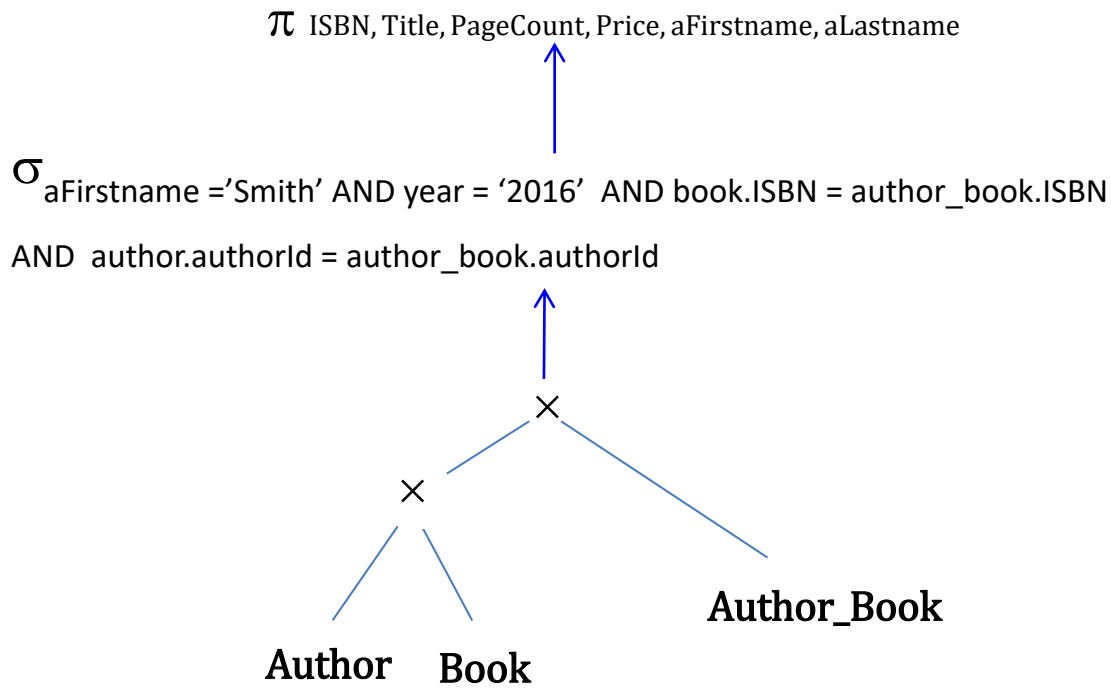



```
2.2) SELECT b.ISBN, b.Title, b.PageCount, b.Price, a.aFirstname,
a.aLastname
FROM Book b, Author a, Author_Book ab
WHERE b.ISBN = ab.ISBN AND ab.AuthorId = a.AuthorId
      AND a.aFirstname = 'Smith' and b.Year = 2016;
```

<u>Part a:</u>

π ISBN, Title, PageCount, Price, aFirstname, aLastname (σ (aFirstname = 'Smith') $\wedge$ (Year= '2016') $\wedge$ (book.ISBN = author_book. ISBN) $\wedge$ (author.AuthorId = author_book. AuthorId) (Author $\times$ Book $\times$ Author_Book))

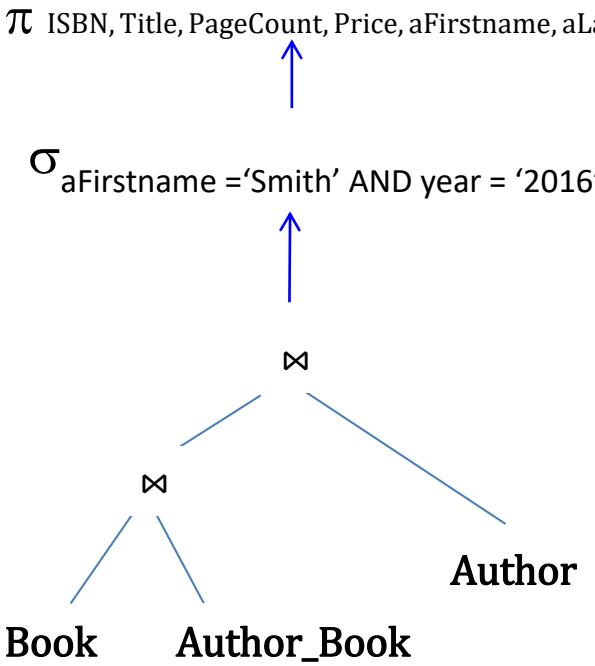
Part b:



Part c:

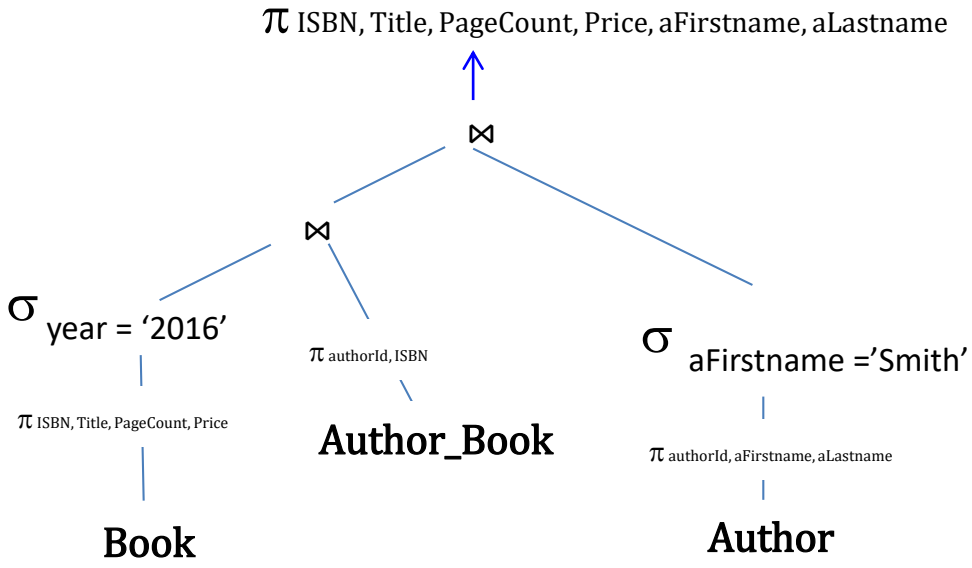
a. **First heuristics**

π ISBN, Title, PageCount, Price, aFirstname, aLastname (σ (aFirstname = 'Smith') ∧ (Year= '2016') (Author ⋈ (Book ⋈ Author_Book)))



b. Second heuristics

π ISBN, Title, PageCount, Price, aFirstname, aLastname ((σ (aFirstname = 'Smith') Author) ⋈ ((σ (Year= '2016') Book) ⋈ Author_Book))



Explanation

There are 3 heuristics used:

- First we convert the Cartesian product into join by pushing the join condition down the query tree. So by this way we reduce the intermediate results. And then apply the remaining filters on the join results.
- Then we push the selection down to reduce the number of rows involved in the join.
- Finally we push the projection down the query tree to reduce the number of columns of the intermediate results.

4. [15 pts] Query Execution

Consider two relations, R(A,B,C) and S(B,D,E). R has 1140 tuples. Each block can hold 15 tuples per block. If it is not specified, assume we have a large enough buffer to perform the operation. Assume that S is the smaller relation and R is the larger relation.

a. Assume we have a memory buffer that can hold 25 blocks (M=25), and the cost of joining R and S using a block nested-loop join is 228. How many blocks are used to store the tuples in S?

$$B_r = \frac{1140}{15} = 76$$

$$228 = B_s + \left(\frac{B_s}{M-2}\right) B_r$$

$$228 = B_s + \frac{76}{23} B_s$$

$$B_s = 53$$

b. What is the cost of joining R and S using a simple sort-merge join, using the B(S) you found in question 1?

$$2b_r\left(\left\lceil \log_{M-1}\left(\frac{B_r}{M}\right)\right\rceil + 1\right) + 2b_s\left(\left\lceil \log_{M-1}\left(\frac{B_s}{M}\right)\right\rceil + 1\right) + B_r + B_s$$

$$2(76)\left(\left\lceil \log_{24}\left(\frac{76}{25}\right)\right\rceil + 1\right) + 2(53)\left(\left\lceil \log_{24}\left(\frac{53}{25}\right)\right\rceil + 1\right) + 76 + 53 = 645$$

c. What is the cost of joining R and S using a hash-based join, using the B(S) you found in question 1?

$$3(B_r + B_s) = 3(76 + 53) = 387$$

5. [15 pts] Cost Estimation

Note that T(R) is the number of tuples in relation R, and V(R,A) is the number of distinct values
of the attribute A in relation R. Consider a database with three relations: R(A,B,C), S(B,D), Z(C,E).
We have the following statistics:
T(R) =500 V(R,A) =10 V(R,B) =100 V(R,C)=50
T(S) =5000 V(S,B) =400 V(S,D) =100
T(Z) =200 V(Z,C) =100 V(Z,E) =25
For each query, estimate the number of tuples returned. You must also write the formula you use
to calculate the number of tuples (in terms of T's and V's):
a. Select Z.E Where Z.E = 10

$$\frac{T(Z)}{V(Z,E)} = \frac{200}{25} = 8$$

b. Select R.B, Z.E From R,Z Where R.C = Z.C

$$\frac{T(R) \times T(Z)}{\max\big(V(R,C), V(Z,C)\big)} = \frac{500 \times 200}{100} = 1000$$

c. Select R.A, S.D From R,S Where R.B = S.B and R.A = 3

$$T(U) = \frac{T(R) \times T(S)}{\max\big(V(R,B), V(S,B)\big)} = \frac{500 \times 5000}{400} = 6250$$

$$\frac{T(U)}{V(U,A)} = \frac{6250}{10} = 625$$

6. [16 pts] Query Execution

Consider a relation R with attributes (a, b) and a relation S with attributes (b, c). Column b in S is a primary key and column b in R is a foreign key. Assume that there are no indexes and no sorted keys available and that there are 25 buffer blocks available. Table R has 1500 blocks with 50 tuples per bock.

Table S has 400 with 100 tuples per block. Compute the I/O costs for the following joins:

a) [4 pts] Block nested loops join with R as the outer relation and S as the inner relation

$$B_r + \left(\frac{B_r}{M-2}\right) B_s = 1500 + \left(\frac{1500}{23}\right) \times 400 = 27{,}587$$

b) [4 pts] Block nested loops join with S as the outer relation and R as the inner relation

$$B_s + \left(\frac{B_s}{M-2}\right) B_r = 400 + \left(\frac{400}{23}\right) \times 1500 = 26{,}487$$

c) [4 pts] Sort merge join R with S

$$2b_r\left(\left\lceil \log_{M-1}\left(\frac{B_r}{M}\right)\right\rceil + 1\right) + 2b_s\left(\left\lceil \log_{M-1}\left(\frac{B_s}{M}\right)\right\rceil + 1\right) + B_r + B_s$$

$$2(1500)\left(\left\lceil \log_{24}\left(\frac{1500}{25}\right)\right\rceil + 1\right) + 2(400)\left(\left\lceil \log_{24}\left(\frac{400}{25}\right)\right\rceil + 1\right) + 1500 + 400 = 12500$$

d) [4 pts] Hash join R with S

$$3(B_r + B_s) = 3(1500 + 400) = 5700$$

7. [15 pts] Join Algorithms

Consider two relations R and S. The tuples in each relation are listed in the following table.

R	S
7	8
2	4
9	2
8	1
3	3
9	2
1	7
3	3
6	

We want to do the natural join of R and S based on different join algorithms. For each algorithm listed as following, give the join results in the order that they would be output by the corresponding join algorithm.

(a) [5 pts] The one tuple at a time nested-loop join algorithm. Suppose R is used for the outer loop and S is used for the inner loop.

R	S
8	8
3	3
3	3
9	9
4	4
4	4
2	2
4	4
4	4

(b) [5 pts] The merge-sort join algorithm.

First we need to sort both relations

R	S
2	2
3	3
4	3
4	4
7	4
8	5
9	8
10	9
10	

R	S
2	2
3	3
3	3
4	4
4	4
4	4
4	4
8	8
9	9

(c) [5 pts] The hash join algorithm. We assume only two hash buckets exist, numbered 0 and 1, respectively. The hash function hashes even values to bucket 0 and odd values to bucket 1. Moreover, we assume that in the join phase of the hash join algorithm, **R** is used as the "load" relation and **S** is used as the "stream" relation (i.e., First load R bucket 0, and for each entry scan through S bucket 0 to find matches. Then do the same for the bucket 1). Furthermore, we assume the content of a bucket are read in the same order as they were written.

First partition tuples in R and S using the hash function

For R →

For even number

8
10
4
10
2
4

For odd numbers

3
9
7

For S →

For even number

2
4
8
4

For odd numbers

9
5
3
3

Join Even number buckets then we join Odd number buckets

8	8
4	4
4	4
2	2
4	4
4	4
3	3
3	3
9	9