CMPT606 – Advanced Database
Fall 2019 – Homework 2

Due Thursday 21/11/2019 midnight – Submit your softcopy to blackboard.

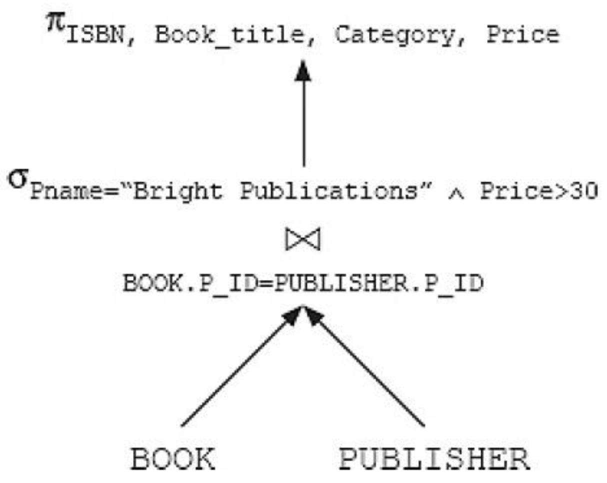
1. [20 pts] Consider the following SQL queries on Online Book Store Application.
   a. [3 pts] Transform these SQL queries into relational algebra expressions.
   b. [3 pts] Draw the initial query trees for each of these expressions.
   c. [4 pts] Derive their optimized query trees after applying heuristics on them. You can just show the last optimal query tree but briefly explain the heuristics applied to arrive to the optimal solution.

Solution

```
1.1) SELECT ISBN, Book_title, Category, Price
FROM BOOK B, PUBLISHER P
WHERE P.P_ID = B.P_ID AND Pname='Bright Publications'
AND Price>30;
```

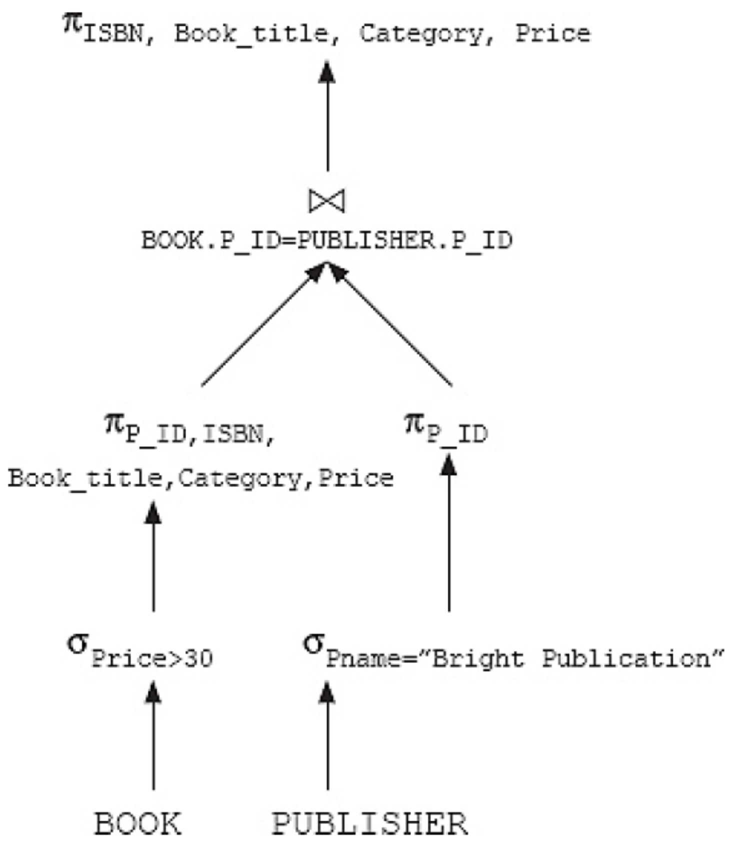
Ans: $\pi_{\text{ISBN, Book\_title, Category, Price}}\ (\sigma_{\text{Pname="Bright Publications"} \wedge \text{Price>30}}\ (\text{BOOK} \bowtie_{\text{B.P\_ID=P.P\_ID}} \text{PUBLISHER}))$

The initial query tree of this expression is given below:



$\pi_{\text{ISBN, Book\_title, Category, Price}}$
$\sigma_{\text{Pname="Bright Publications"} \wedge \text{Price>30}}$
$\bowtie$
BOOK.P_ID=PUBLISHER.P_ID
BOOK
PUBLISHER

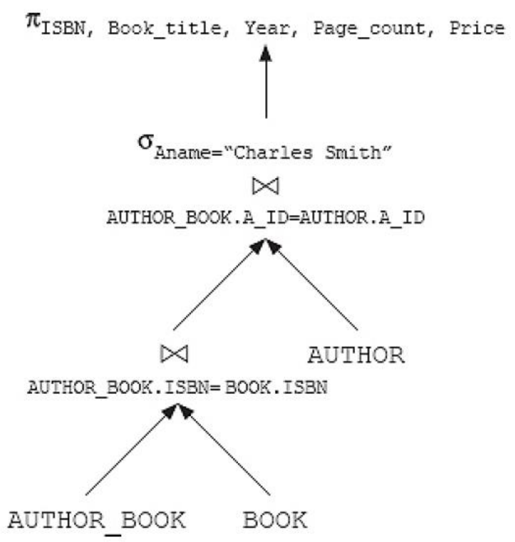
The optimized query tree is given below:


π ISBN, Book_title, Category, Price
⋈ BOOK.P_ID=PUBLISHER.P_ID
π P_ID,ISBN, Book_title,Category,Price
π P_ID
σ Price>30
σ Pname="Bright Publication"
BOOK
PUBLISHER

```
1.2) SELECT ISBN, Book_title, Year, Page_count, Price
FROM BOOK B, AUTHOR A, AUTHOR_BOOK AB
WHERE B.ISBN = AB.ISBN AND AB.A_ID = A.A_ID
AND Aname = ‘Charles Smith’;
```

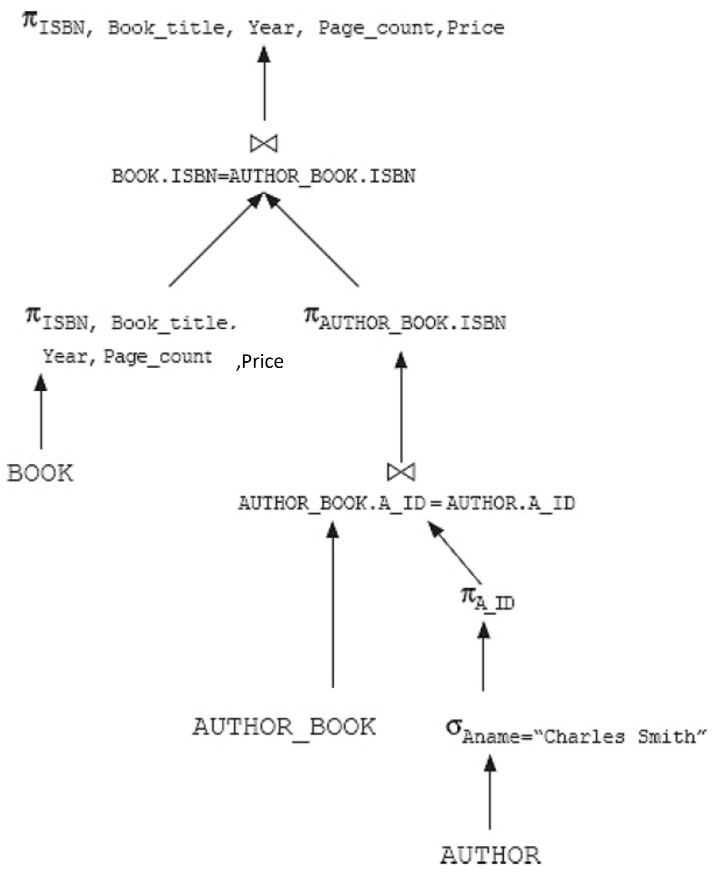
Ans: $\pi_{ISBN,\ Book\_title,\ Year,\ Page\_count,\ Price}\ (\sigma_{Aname=\text{“Charles Smith”}}\ ((BOOK \bowtie_{B.ISBN=AB.ISBN} AUTHOR\_BOOK) \bowtie_{AB.A\_ID=A.A\_ID} AUTHOR))$

The initial query tree of this expression is given below:



π ISBN, Book_title, Year, Page_count, Price
σ Aname="Charles Smith"
⋈ AUTHOR_BOOK.A_ID=AUTHOR.A_ID
⋈ AUTHOR_BOOK.ISBN= BOOK.ISBN
AUTHOR
AUTHOR_BOOK
BOOK

The optimized query tree is given below:



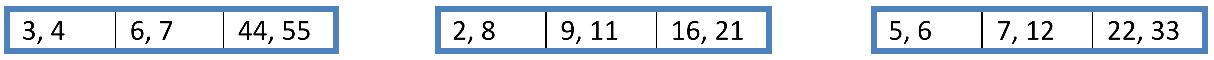
π ISBN, Book_title, Year, Page_count,Price
⋈ BOOK.ISBN=AUTHOR_BOOK.ISBN
π ISBN, Book_title. Year, Page_count ,Price
π AUTHOR_BOOK.ISBN
BOOK
⋈ AUTHOR_BOOK.A_ID = AUTHOR.A_ID
π A_ID
AUTHOR_BOOK
σ Aname="Charles Smith"
AUTHOR

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

Sort phase:

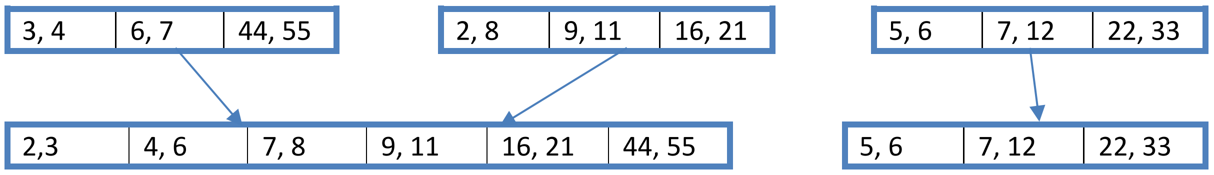
Number of initial runs = $n_r = \left\lceil \frac{B_r}{M} \right\rceil = \left\lceil \frac{9}{3} \right\rceil = 3$

Initial sorted runs:

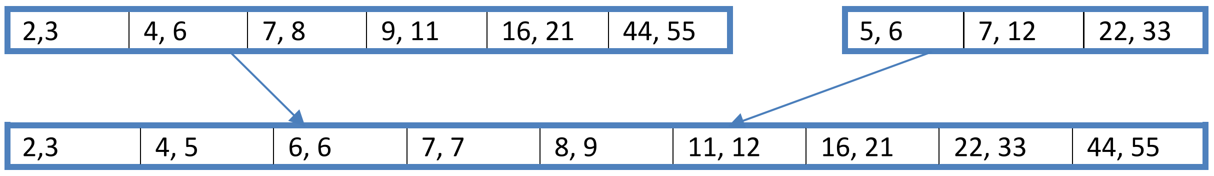


Merge phase: Number of passes = $\lceil \log_{M-1}(n_r) \rceil = \lceil \log_2 3 \rceil = 2$

First pass



Second pass



3. [12 pts] Consider a file with 10,000 blocks and 3 available buffer blocks. Assume that the external sort-merge algorithm is used to sort the file.
   a. [4 pts] Calculate the initial number of runs produced in the first pass.
   b. [4 pts] How many passes will it take to sort the file completely?
   c. [4 pts] How many buffer blocks are required to sort the file completely in two passes?

Solution

a) In the first pass (Pass 0), the number of initial runs (n_R) can be computed as $n_R = \lceil B/M \rceil$, that is, 10000/3 = **3334** initial sorted runs each containing 3 pages except the last that will have only one page.

b) The number of passes required to sort the file completely, including the initial sorting pass, can be calculated as $\lceil \text{Log}_{M-1}(n_R) \rceil + 1$. Here, M−1 is 2 and n_R is 3334. Therefore, $\lceil \text{Log}_2(3334) \rceil + 1$ = 12+1=**13 passes are required**.

c) In Pass 0, $\lceil B/M \rceil$ runs are produced. In Pass 1, we must be able to merge these runs, that is, $M-1 \geq \lceil B/M \rceil$. This implies that M must at least be large enough to satisfy $M * (M-1) \geq B$. Thus, with B = 10000, M = 101.
Thus, we need 101 buffer pages.

4. [15 pts] Query Cost Estimation

Consider two relations, R(A,B,C) and S(B,D,E). R has 1140 tuples. Each block can hold 15 tuples per block. Assume that S is the smaller relation and R is the larger relation.
[5 pts] 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?
[5 pts] 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?
[5 pts] c. What is the cost of joining R and S using a hash-based join, using the B(S) you found in question 1?

Solution
a) B(R) = 1140 / 15 = 76

The formula is: $B(S) + (B(S).B(R))/(M-2) = C$ where S is the smaller relation and R is the larger relation. Any fractional values should be rounded **UP**.

$B(S) + (B(S).B(R))/(M-2) = C$
$B(S) + B(S)*76 / 23 = 228$
$B(S)(1 + 76/23) = 228$
$B(S) = 52.9$ ==> rounded up = **53**

b)

$$\underbrace{2b_R(\lceil \log_{M-1}(b_R/M)\rceil + 1) + 2b_S(\lceil \log_{M-1}(b_S/M)\rceil + 1)}_{\text{Sort Cost}} + \underbrace{(b_R + b_S)}_{\text{Merge Cost}}$$

= 645

c)

$3 * (B(R) + B(S)) = 3 * (76 + 53) =$ **387**

5. [15 pts] Query Cost Estimation

Consider the following two relations:
Customer(cid, name, phone, etc.)
Order(oid, orderDate, cid, etc.)

Customer (C) has 10,000 tuples, with 25 tuples fitting on a block.
Order (O) has 5,000 tuples, with 50 tuples fitting on a block.

There is no index on any attribute of the relations. You may assume the table blocks are contiguous. In your answer always provide the general formula.

(a) Suppose the memory buffer has 101 blocks, compute the cost of using a block-nested loop join to join the above two relations.

(b) Suppose we wanted to join the two relations using a block-nested loop join and limit the cost to 900. What is the smallest value M can be?

(c) What is the cost of joining Customer and Order using a hash-based join?

Solution

(a)

$B(C) = 10{,}000 / 25 = 400$
$B(O) = 5{,}000 / 50 = 100$
Cost $= \lceil B(O) + B(C) \times (B(O) / (M-2)) \rceil = \lceil 100 + 400 \times (100/(101-2)) \rceil = 505$

(b)
$B(C) = 10{,}000 / 25 = 400$
$B(O) = 5{,}000 / 50 = 100$
$B(O) + B(C) \times (B(O) / (M-2)) \leq 900$
$100 + 400*(100/(M-2)) \leq 900$

$100/(M-2) \leq 2$
$50 \leq M - 2$
$52 \leq M$
So M must be at least 52 blocks.

(c) $3B(C) + 3B(O) = 1{,}500$

6. [22 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.

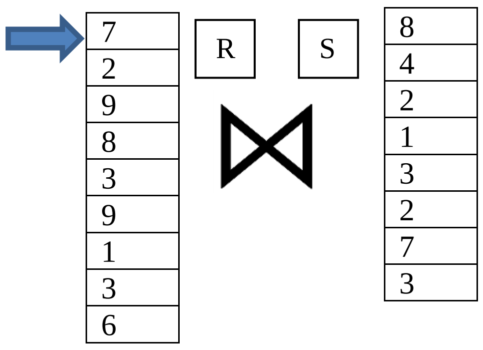
(a) [7 pts] The one tuple at a time nested-loop join algorithm. Suppose R is used for the outer loop and S is used for the inner loop.

(b) [7 pts] The merge-sort join algorithm.

(c) [8 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.

Solution

(a) With R as the outer relation and S as the inner relation.



For every tuple in R (in order), find matches in S (in order). Thus we get:

R	S
7	7
2	2
2	2
8	8
3	3
3	3

1	1
3	3
3	3

(b) **Step 1: Sort R and S**

1
2
3
3
6
7
8
9
9

$\boxed{R}$ $\boxed{S}$

$$\bowtie$$

1
2
2
3
3
4
7
8

Step 2: Merge R and S. Since there are duplicate values in both the tables, **co-grouping** and then **cross-joining** the co-groups are performed. Thus, we get results of the join in sorted order:

R	S
1	1
2	2
2	2
3	3
3	3
3	3
3	3
7	7
8	8

(c) **Step 1: Hashing of R and S into two buckets**
After first phase we obtain:
R bucket$_0$: 2, 8, 6
R bucket$_1$: 7, 9, 3, 9, 1, 3

S bucket$_0$: 8, 4, 2, 2
S bucket$_1$: 1, 3, 7, 3

Step 2: Joining of Buckets with the same index
First load R **bucket$_0$** and then scan through S **bucket$_0$** to find matches: 2, 2, 8
Then load R **bucket$_1$** and then scan through S **bucket$_1$** to find matches: 7, 3, 3, 1, 3, 3

Final result:

R	S
2	2
2	2
8	8
7	7
3	3
3	3
1	1
3	3
3	3