Homework 3

Problem 1: Evaluation of Relational Operators

(a) Duplicate eliminator over unsorted relation R

(b) Grouping operator (group by column X) over a sorted relation R on column X

(c) Grouping operator (group by column X) over unsorted relation R

(d) Sorting operator (sort by column X) over unsorted relation R

(e) Sorting operator (sort by column X), and assume the operator can use a B-tree index that exists on R.X to read the tuples.

(f) Join of two relations R and S

(g) Bag Union of relations R and S

1. For each of the items above, report whether the operator is “Blocking” or “Non-Blocking” and describe why.

Operator	**Blocking/Non-Blocking**	**Reason**
(a)	Blocking	Needs to see all tuples to find and remove duplicates
(b)	Non-Blocking	Since R sorted on X, can output a group as soon as completed
(c)	Blocking	Can't group without full knowledge of all tuples
(d)	Blocking	Must see all tuples first before sorting
(e)	Non-Blocking	Can traverse the index and output sorted tuples as they are found
(f)	Depends	Some algorithms like sort-merge join needs full access to sorted data, while nested-loop join can be non-blocking
(g)	Non-Blocking	Can output tuples as they are read from R, then from S, without waiting for all tuples to be available

2. Assume relation R is 1,000 blocks and relation S is 150 blocks, and the available memory buffers are 200. Moreover, for Point (e) above, the R.X index size is 70 blocks. For each of the items above (a to g), discuss:

a. Whether the operator can be done in one pass or not.

b. If it can be done in one pass, what are the size constraints?

c. If it cannot be done in one pass, then how many passes are needed? Describe the algorithm that uses the number of passes you suggest? What will be the I/O cost?

Operator	One pass?	Reason + Algorithm	I/O Cost
(a)	No	Need to sort or hash to find duplicates. External sort: 2 passes needed. - Phase 1: No constraints - Phase 2: B(R) <= M^2	External Sort cost: 3 * B(R) = 3,000
(b)	Yes	Already sorted on X. Just scan once, aggregate on the fly. No constraints.	B(R) = 1,000
(c)	Yes	The groups must fit in M - 1 = 199 buffers.	B(R) = 1,000
(d)	No	Must do external sorting. 2 passes: Phase 1: No constraints. Phase 2: B(R) <= M^2 (1000 <= 200^2)	Same: External Sort cost
(e)	Yes	Use index scan (only 70 blocks) to access tuples ordered by X. 70 < 200 ⇒ fits in memory.	0 if index is in-memory
(f)	No (sort-merge join)	2 passes. No constraints in Phase 1. B(R) + B(S) <= M^2 (1,150 <= 200^2) in Phase 2.	3 * (B(R) + B(S)) = 3 * 1,150 = 3,450
(g)	Yes	Just scan R and S, output tuples. Min(B(R), B(S)) <= M-1 (150 <= 199)	B(R) + B(S) = 1,000 + 150 = 1,150

Problem 2: Estimation of Relation Size

Given the following three relations R1(a, b), R2(b, c), and R3(c, d) and associated statistics shown below in the metadata table. Estimate the number of tuples in the result relation for the different queries listed below, namely, T (Q).

T (R1) = 400;V (R1, a) = 50;

V (R1, b) = 50

T (R2) = 500;V (R2, b) = 40;

V (R2, c) = 100

T (R3) = 1000;V (R3, c) = 50; V (R3, d) = 100

If there are any additional assumptions you need to make to answer any of the questions below, please explicitly state them.

1. Q = σ(a=10) (R1).

Selection on a single equality:

```
T(Q) = T(R1) / V(R1, a) = 400 / 50 = 8
```

2. Q = σ(a>=10) (R1). (Assume that the range of R1.a is [1, 50]).

Range selection (attribute range is [1, 50]):

- Values ≥ 10: 41 values (10 through 50 inclusive).
- Total values = 50.

```
T(Q) = T(R1) * (41 / 50) = 400 * (41 / 50) = 328
```

3. Q = σ(a>=10 AND b=20) (R1). Again assume the range of R1.a is [1, 50].

Conjunction of two selections:

- a >= 10 produces 328 tuples (from above).
- Then applying b = 20 selection:

Apply independent attribute assumption:

```
T(Q) = 328 / V(R1, b) = 328 / 50 = 6.56
```

4. Q = R1 ▷◁ R2, where ▷◁ represents natural join.

Natural join on common column b. Join size formula:

```
T(Q) = T(R1) * T(R2) / Max(V(R1, b), V(R2, b)) = 400 * 500 / Max(50, 40) =
4000
```

5. Q = (R1 ▷◁ R2) ▷◁ R3.

Join results from 4 with R3 on c.

```
T(Q) = 4000 * T(R3) / Max(V(R2, c), V(R3, c)) = 4000 * 1000 / Max(100, 50)
= 40000
```

6. Q = (σ(a>=10) (R1)) ▷◁ R2 ▷◁ R3.

First, results from 2 - 328 tuples - join with R2 on b. Then join with R3 on c.

```
T(Q1) = 328 * T(R2) / Max(V(R1, b), V(R2, b)) = 328 * 500 / Max(50, 40) =
3280
T(Q2) = 3280 * T(R3) / Max(V(R2, c), V(R3, c)) = 3280 * 1000 / Max(100, 50)
= 32800
```