CPS216 Data-Intensive Computing Systems - Fall 2010
Written Assignment 1

- Total points = 100. Due date: Wednesday, Oct. 20, 2010 (5.00 PM).
- Submission: In class, or email solutions in pdf or plain text to gang@cs.duke.edu. You can also drop off the solutions at Gang's office: N303A North Building.
- Do not forget to indicate your name on your submission.
- State all assumptions. For questions where descriptive solutions are required, you will be graded both on the correctness and clarity of your reasoning.
- Email questions to gang@cs.duke.edu and shivnath@cs.duke.edu.

Question 1 **Points 10**

This question is based on the following SQL query over table R(A,B):

```
Select    A, MAX(B)
From      R
Where     B >= 1000 and B < 2000
Group By  A
```

This query drops records in table R that do not satisfy the filter predicate "$B \geq 1000$ *and* $B < 2000$". The remaining records are partitioned into groups based on unique values of attribute A. That is, each distinct value of A (among the remaining records) gives one group of records. The final output will be the value of A and the maximum value of B in each group.

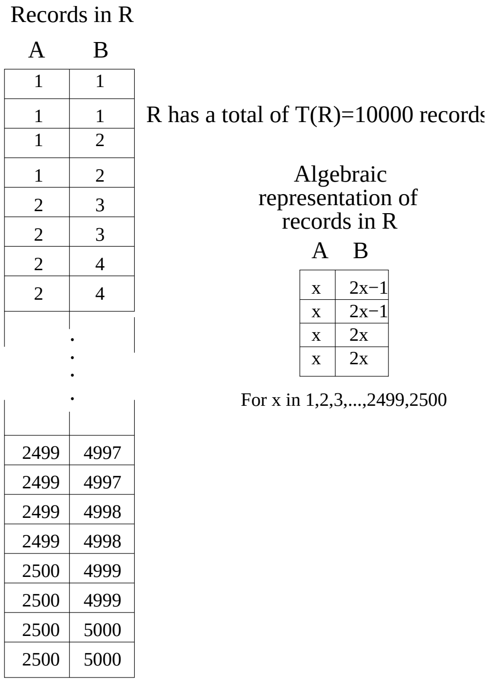
For example, if we have a table $R(A, B)$ with records $\{\{a, 1000\}, \{b, 900\}, \{a, 1500\}, \{c, 1111\}, \{b, 1123\}, \{b, 1234\}, \{d, 10\}\}$, then the result of the query is $\{\{a, 1500\}, \{c, 1111\}, \{b, 1234\}\}$.

(a) Explain how MapReduce can be used **most efficiently** to process this query. That is, explain what the Map phase will do and what the Reduce phase will do. Keep your answer brief and to the point (A couple of sentences are enough to convey the main ideas).

(b) Figure 1 describes the actual contents of the records in table R(A,B). Suppose the records in table R are stored on 10 nodes, M1-M10, as shown in Figure 2. For example, all records with "A ≥ 1 and A ≤ 250" are stored on node M1, all records with "A ≥ 251 and A ≤ 500" are stored on node M2, and so on. The MapReduce computation is done with 10 Mappers and 2 Reducers. The 10 Mappers run respectively on the 10 nodes M1-M10. The 2 Reducers run on 2 separate nodes R1-R2 such that R1 will process all records with "A ≥ 1 and A ≤ 1250", and R2 will process all records with "A ≥ 1251 and A ≤ 2500". In this scenario, how many records will be shuffled from a Mapper node to a Reducer node in the most efficient MapReduce execution?

Question 2 **Points 15**

Consider the following SQL query over tables $R(A)$, $S(A)$, and $T(A)$. Note that "Select Distinct" in SQL represents a duplicate-eliminating projection.



Records in R

A	B
1	1
1	1
1	2
1	2
2	3
2	3
2	4
2	4
.	
.	
.	
.	
2499	4997
2499	4997
2499	4998
2499	4998
2500	4999
2500	4999
2500	5000
2500	5000

R has a total of T(R)=10000 records

Algebraic representation of records in R

A	B
x	2x−1
x	2x−1
x	2x
x	2x

For x in 1,2,3,...,2499,2500

Figure 1: The contents of records in R

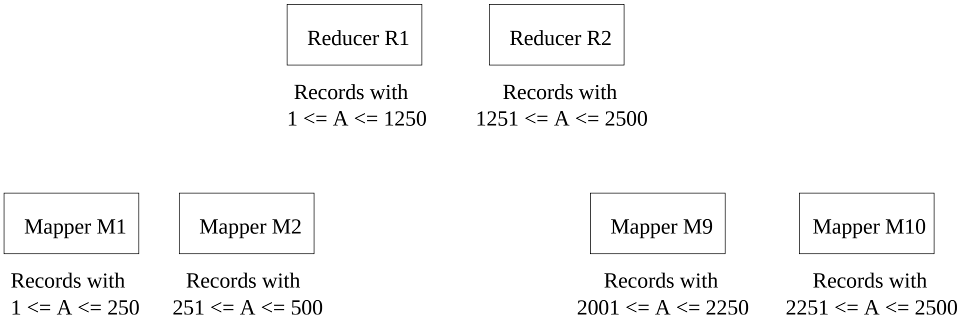


Figure 2: 10 mappers M1-M10 and 2 reducers R1-R2

```
Select Distinct R.A
From   R, S, T
Where R.A = S.A and R.A = T.A
```

If we have a table with $R(A, B)$ tuples $\{\{1, a\}, \{1, b\}, \{2, c\}, \{3, d\}, \{4, e\}, \{4, f\}\}$, then a duplicate-preserving projection on $R.A$ will return $\{1, 1, 2, 3, 4, 4\}$, while a duplicate-eliminating projection on $R.A$ will return $\{1, 2, 3, 4\}$.

Figures 3(a)-(e) show five logical plans. The logical operator π in Figure 3 represents a duplicate-eliminating projection. For example, $\pi_{R.A}$ represents a duplicate-eliminating projection of attribute $R.A$. On the other hand, the logical operator τ represents a duplicate-preserving projection. The logical operator $\bowtie$ represents a natural join. Assume that, in the general case, R, S, and T can contain duplicate tuples.

1. Is the logical plan in Figure 3(a) equivalent to the logical plan in Figure 3(b)? (Equivalent means that they produce the same query result.) If not, what is the minimal set of properties on the tables such that these plans are equivalent?

2. Is the logical plan in Figure 3(a) equivalent to the logical plan in Figure 3(c)? If not, what is the minimal set of properties on the tables such that these plans are equivalent?

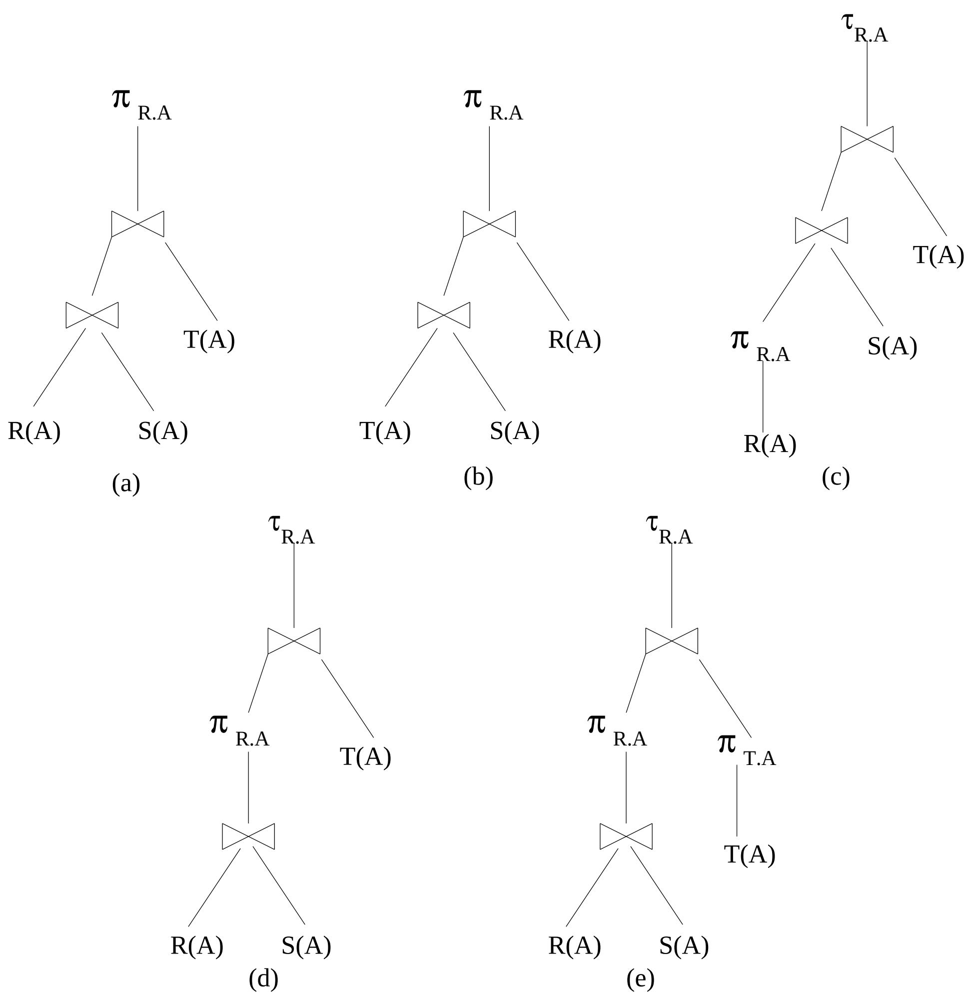


Figure 3: Logical plans

3. Is the logical plan in Figure 3(a) equivalent to the logical plan in Figure 3(d)? If not, what is the minimal set of properties on the tables such that these plans are equivalent?
4. Is the logical plan in Figure 3(a) equivalent to the logical plan in Figure 3(e)? If not, what is the minimal set of properties on the tables such that these plans are equivalent?

Question 3 **Points 15**

In this question we will find the cost of physical plans in terms of the number of getNext() calls. Figure 4 shows the physical plan generated from each logical plan in Figure 3. We make the following assumptions:

- A1 Data from each table is read through a table scan which is represented as *Scan*. We do not consider index scans.
- A2 Each join is implemented using a tuple nested loop join, or *TNLJ*. TNLJ is the only physical join operator available.
- A3 A duplicate-eliminating projection is always implemented using the operator π. Note that π is a physical operator here.
- A4 A duplicate-preserving projection is always implemented using the operator τ. Note that τ is a physical operator here.

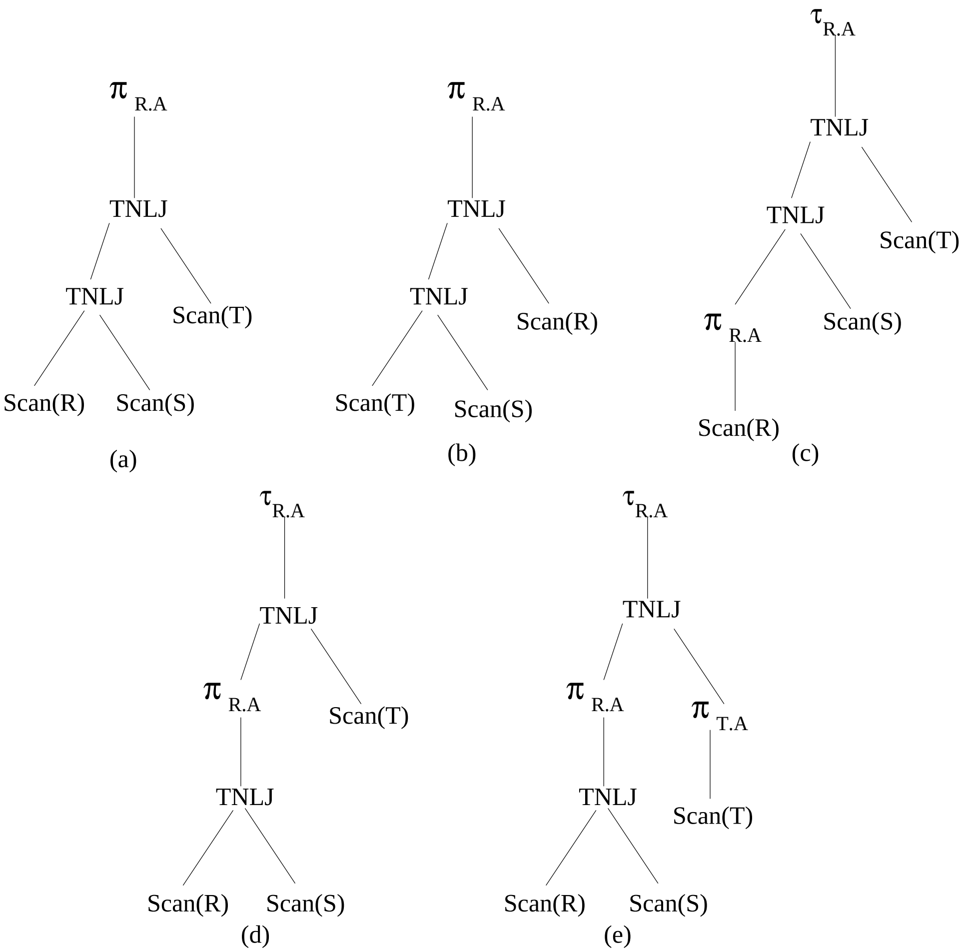


Figure 4: Physical plans

The records in the tables are as follows.

1. $R(A)$ contains $\{1, 1, 2, 2, 3, 3\}$.
2. $S(A)$ contains $\{1, 2\}$.
3. $T(A)$ contains $\{1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4\}$.

Answer the following five questions based on the above information. Give explanations for partial credit.

1. How many getNext() calls are made by the plan in Figure 4(a)?
2. How many getNext() calls are made by the plan in Figure 4(b)?
3. How many getNext() calls are made by the plan in Figure 4(c)?
4. How many getNext() calls are made by the plan in Figure 4(d)?
5. How many getNext() calls are made by the plan in Figure 4(e)?

Question 4 **Points 15**

Given the assumptions A1 to A4 in Question 3 about physical plans, find the number of possible physical plans for the SQL query in Question 2. Assume that R, S, and T contain duplicate tuples. Give explanations for partial credit.

Question 5 **Points 10**

Given the assumptions A1 to A4 in Question 3 about physical plans, find the physical plan that does the minimum number of getNext() calls for the SQL query in Question 2 and the records in tables $R(A)$, $S(A)$, and $T(A)$ given in Question 3. Give explanations for partial credit.

Hint: The physical plan that does the minimum number of getNext() calls may not be one of the plans in Figure 4.

Question 6 **Points 10**

Consider a 3.5 inch disk with 2 magnetic surfaces with 64 tracks per surface, rotating at 3600 rpm. It has a usable capacity of 2 megabytes (2×2^{20} bytes). Assume 20% of each track is used as overhead (gaps). Also, assume that the usable capacity is equally distributed among the tracks.

a. What is the burst bandwidth this disk can support?

b. What is the sustained bandwidth this disk can support?

c. What is the average rotational latency?

d. Assuming the average seek time is 16 ms, what is the average time to fetch a 2-kilobyte (2×2^{10} bytes) sector?

Question 7 **Points 10**

Consider a disk with the following properties:

- There are four platters providing eight surfaces.
- There are $2^{13} = 8192$ tracks per surface.
- There are (on average) $2^8 = 256$ sectors per track.
- There are $2^9 = 512$ bytes per sector.
- The disk rotates at 3840 rpm.
- The block size is $2^{12} = 4096$ bytes.
- Assume 10% of each track is used as overhead.
- The time it takes the head to move n tracks is 1 + n/500 milliseconds.

Suppose that we know that the last I/O request accessed cylinder 3000. (Cylinders are numbered sequentially: $1, 2, \ldots, 8192$.)

a. What is the expected (average) number of cylinders that will be traveled due to the very next I/O request to this disk?

b. What is the expected block access time for the next I/O, again given that the head is on cylinder 3000 initially?

Question 8 **Points 15**

Let $R_1(A, B)$ and $R_2(B, C)$ be two tables of data.

1. Suppose, neither R_1 nor R_2 has duplicate tuples. (That is, there is no pair of distinct tuples $r \in R_1$ and $s \in R_1$ such that $r.A = s.A$ and $r.B = s.B$; similarly for R_2.) What is the necessary and sufficient condition for the following equivalence to hold: $\sigma_{P_1 \vee P_2}(R_1 \bowtie R_2)$ $= (\sigma_{P_1} R_1 \bowtie R_2) \bigcup_B (R_1 \bowtie \sigma_{P_2} R_2)$? Here, P_1 is a predicate that involves attributes in R_1 only, and P_2 is a predicate that involves attributes in R_2 only. $\bigcup_B$ denotes *bag union* (see Figure 5 for an illustration of bag and set unions.) State your condition as an expression in relational algebra. You may use ϕ to denote an empty set of tuples (i.e., a *null set*).

2. How does your answer to (1) change if R_1 and R_2 can have duplicate tuples in them? (That is, now there can be pairs of distinct tuples $r \in R_1$ and $s \in R_1$ such that $r.A = s.A$ and $r.B = s.B$; similarly for R_2.)

3. Does the following condition hold if R_1 and R_2 can have duplicate tuples in them: $\sigma_{P_1 \bigvee P_2}(R_1 \bowtie R_2) = (\sigma_{P_1} R_1 \bowtie R_2) \bigcup_S (R_1 \bowtie \sigma_{P_2} R_2)$? $\bigcup_S$ denotes *set union* (also called duplicate-eliminating union; see Figure 5.) If not, can you suggest a modification to the right hand side of this condition so that the new condition holds? You can express your new condition in relational algebra or describe it in English.

$$\begin{array}{|c|c|}\hline 1 & a \\ \hline 2 & b \\ \hline \end{array} \cup_B \begin{array}{|c|c|}\hline 1 & a \\ \hline 3 & c \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline 1 & a \\ \hline 2 & b \\ \hline \end{array} \cup_S \begin{array}{|c|c|}\hline 1 & a \\ \hline 3 & c \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline \end{array} \cup_B \begin{array}{|c|c|}\hline 1 & a \\ \hline 3 & c \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline 1 & a \\ \hline 3 & c \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline \end{array} \cup_S \begin{array}{|c|c|}\hline 1 & a \\ \hline 3 & c \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & a \\ \hline 2 & b \\ \hline 3 & c \\ \hline \end{array}$$

Figure 5: Examples to illustrate Bag Union ($\bigcup_B$) and Set Union ($\bigcup_S$)