# CSI 508. Database Systems I – Fall 2016

## Homework Assignment III

The deadline for this assignment is **11:59 PM, November 7, 2015**. *Submissions after this deadline will not be accepted.* Each student is required to enter the UAlbany Blackboard system and then upload a .pdf file (in the form of [last name]_[first name].pdf) that contains answers to Problems 1-4.

The total grade for this assignment is 100 points. If you find any error or have questions or suggestions, please contact the instructor (`jhh@cs.albany.edu`).

---

**Problem 1.** (60 points) Assume that (1) each of relations $R$ and $S$ contains $10^8$ tuples, (2) the size of each tuple in $R$ and $S$ is 160 bytes, (3) $1GB = 10^3MB = 10^6KB = 10^9B$, (4) the size of each disk block is $4KB$, (5) a hard drive that can read/write $100MB$ per second is used, (6) each disk seek takes 10 milliseconds, and (7) $1GB$ of main memory is used for buffering. Answer each of the following questions:

(a) (10 points) Consider a situation where *external sort-merge* is applied to relation $R$. Assuming that 1GB of main memory is used to create each initial run, explain how many initial runs will be created. Also, describe how many block transfers and disk seeks will be needed to create all of these initial runs.

(*Answer*) By (1), (2), and (3), the size of relation $R$ is $10^8 \times 160B = 16GB$. Therefore, $16(= 16GB/1GB)$ initial runs will be created. By (3) and (4), the number of blocks in relation $R$ is $16GB/4KB = 4 \times 10^6$. To read relation relation $R$ and then write 16 initial runs, $8 \times 10^6 (= 2 \times 4 \times 10^6)$ block transfers will be needed because each of the $4 \times 10^6$ blocks will be read once and write once. Furthermore, $32(= 2 \times 16)$ disk seeks will be needed because each of the 16 initial runs requires one disk seek to read 1GB of data and one more disk seek to write 1GB of data.

(b) (10 points) After the initial runs are created as explained above, *external sort-merge* repeatedly merges a number of runs into a bigger run until only one final run remains. Assume that each merge pass uses $200MB$ of buffer space (out of $1GB$ of main memory) per input/output run. Explain how many merge passes are needed to obtain one final run. Also, describe how many block transfers and disk seeks all of these merge passes require.

(*Answer*) Given $1GB$ of main memory, using $200MB$ of main memory per run leads to four $200MB$ input buffers and one $200MB$ output buffer (i.e., 4-way merge). In this case, two merge passes are sufficient since the first pass can merge 16 runs into 4 runs and then the second pass can merge 4 runs into 1 run. Each merge pass requires $8 \times 10^6 (= 2 \times 4 \times 10^6)$ block transfers (because each of the $4 \times 10^6$ blocks needs to be read once and written once) and $160(= 2 \times 16GB/200MB)$ disk seeks (because one disk seek is needed to read $200MB$ of data and one more disk seek is needed to write $200MB$ of data). Therefore, these two merge passes require $16 \times 10^6$ block transfers and 320 disk seeks.

(c) (10 points) Consider the merge passes described in (b). If each merge pass uses $58MB$ of buffer space (out of $1GB$ of main memory) per input/output run, how many block transfers and disk seeks are necessary? Also, explain whether it is more advantageous to use a $200MB$ buffer per run or a $58MB$ buffer per run. Justify your answer by estimating the overall time for merging all runs into one final run.

(*Answer*) Given $1GB$ of main memory, using $58MB$ of main memory per run leads to sixteen $58MB$ input buffers and one $58MB$ output buffer (i.e., 16-way merge). In this case, one merge pass is sufficient since it merges 16 runs into 1 run. Reading all of the 16 runs requires $4 \times 10^6$ block transfers and $288(= 16 \times \lceil 1GB/58MB \rceil)$ disk seeks. Writing the final run requires $4 \times 10^6$ block transfers and $276(= \lceil 16GB/58MB \rceil)$ disk seeks. Therefore, the merge pass requires $8 \times 10^6$ block transfers and 564 disk seeks.

Since the data transfer rate is $100MB/sec$, $25 \times 10^3 (= 100MB/4KB)$ disk blocks can be transferred per second, meaning that a block transfer takes $4 \times 10^{-5}$ seconds. When a $200MB$ buffer is used for each run, the overall time for merging all runs into one final run is $643.2(= 16 \times 10^6 \times 4 \times 10^{-5} + 320 \times 0.01)$ seconds. On the other hand, when a $58MB$ buffer is used for each run, the overall time for merging all runs into one final run is $325.64(= 8 \times 10^6 \times 4 \times 10^{-5} + 564 \times 0.01)$ seconds. Therefore, it is more advantageous to use a $58MB$ buffer for each run, thereby enabling a 16-way merge.

(d) (10 points) Assume that *merge join* is applied to relations $R$ and $S$ when relations $R$ and $S$ are already sorted by the join column(s). Explain how many block transfers and disk seeks this join requires when it uses a $500MB$ buffer for relation $R$ and another $500MB$ buffer for relation $S$.

(*Answer*) Since relation $R$ has $4 \times 10^6$ blocks and relation $S$ also has $4 \times 10^6$ blocks, merge join requires $8 \times 10^6$ block transfers. It also requires $64(= \lceil 16GB/500MB \rceil + \lceil 16GB/500MB \rceil)$ disk seeks.

(e) (10 points) Consider a situation where *hash join* is applied to relations $R$ and $S$. Explain how many block transfers and disk seeks this join requires. For this problem, partition relation $S$ into 19 buckets (assume that, in this case, each bucket can fit into the $1GB$ memory space). Also assume that relation $R$ is partitioned into 19 buckets. To simplify cost analysis, assume that the last block of each bucket from $R$ and/or $S$ is completely filled. To partition each relation into 19 buckets, use a $50MB(= 1GB/20)$ buffer for the relation and each bucket. To simplify cost analysis, assume that, when the end of each bucket is saved, the $50MB$ buffer for the bucket is completely filled.

(*Answer*)

Partitioning Phase: Relation $R$ occupies $4 \times 10^6$ disk blocks. Therefore, regrouping tuples from $R$ into 19 buckets require $8 \times 10^6 (= 2 \times 4 \times 10^6)$ block transfers and $640(= 2 \times 16GB/50MB)$ disk seeks. Similarly, splitting $S$ into 19 buckets require $8 \times 10^6$ block transfers and 640 disk seeks.

Build and Probe Phases: 19 iterations where one bucket from $R$ and one bucket from $S$ are read require $8 \times 10^6$ block transfers and $38(= 2 \times 19)$ disk seeks.

In conclusion, the overall hash join requires $24 \times 10^6$ block transfers and 1318 disk seeks.

(f) (10 points) Explain whether it is more advantageous to use *hash join* or *merge join* when relations $R$ and $S$ are not sorted by the join column(s). Justify your answer by using your answers to the above questions.

(*Answer*) The overall time for hash join is $643.2 (= 24 \times 10^6 \times 4 \times 10^{-5} + 1318 \times 973.18)$ seconds. On the other hand, the time for sorting relations $R$ and $S$ is $651.28 (= 2 \times 325.64)$ seconds and the time for merge join is $320.64 (= 8 \times 10^6 \times 4 \times 10^{-5} + 64 \times 0.01)$ seconds. Therefore, it is more advantageous to use hash join.

**Problem 2.** (20 points) Determine if each of the following statements is true or false (assume that $R$ and $S$ have the same set of attributes). If a statement is true, prove it. Otherwise, give a relevant example.

(a) (10 points) $\Pi_A(R \cap S) = \Pi_A(R) \cap \Pi_A(S)$.

(*Answer*) False. Consider the following relations:

R:

| A | B |
|---|---|
| 1 | 1 |

S:

| A | B |
|---|---|
| 1 | 2 |

Then, $\Pi_A(R \cap S) = \emptyset$ whereas $\Pi_A(R) \cap \Pi_A(S) = \{(1)\}$.

(b) (10 points) $\sigma_{\theta_1}(\sigma_{\theta_2}(R)) = \sigma_{\theta_1 \wedge \theta_2}(R)$.

(*Answer*) Let $\theta(t)$ denote that tuple $t$ satisfies predicate $\theta$. Then,

$$\begin{aligned} t \in \sigma_{\theta_1}(\sigma_{\theta_2}(R)) &\Leftrightarrow \theta_1(t) \wedge t \in \sigma_{\theta_2}(R) \\ &\Leftrightarrow \theta_1(t) \wedge (\theta_2(t) \wedge t \in R) \\ &\Leftrightarrow (\theta_1(t) \wedge \theta_2(t)) \wedge t \in R \\ &\Leftrightarrow (\theta_1 \wedge \theta_2)(t) \wedge t \in R \\ &\Leftrightarrow t \in \sigma_{\theta_1 \wedge \theta_2}(R) \end{aligned}$$

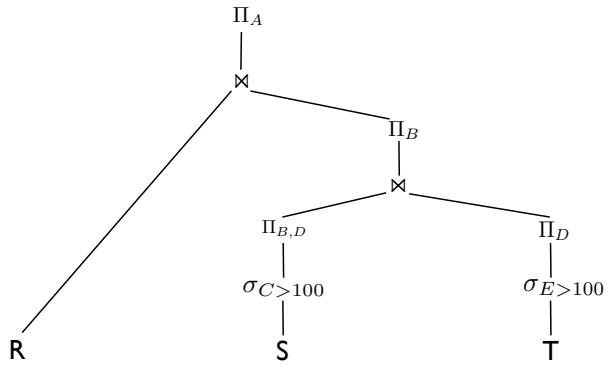
.

**Problem 3.** (10 points) Consider relations $R(A, B)$, $S(B, C, D)$ and $T(D, E)$ and the following query:

```
select A
from R natural join S natural join T
where C > 100 and E > 100;
```

Among many possible relational expressions that are equivalent to the above query, draw the most efficient one (as a tree) that you found. Your relational expression must apply selections and projections as early as possible to reduce the size of intermediate relations that operators would produce. If both a selection and a projection can be applied to a relation, apply the selection earlier than the projection. You also need to determine the order of binary join operations under the assumptions that (1) selections can substantially reduce the size of relations, and (2) it is beneficial to do less expensive join operations (particularly, those with small amounts of input and output data) before other join operations.

(*Answer*)



**Problem 4.** (10 points) Assume that we have four relations $R(A, B)$, $S(A, C)$, $T(A, D)$ and $U(A, E)$ and the size (i.e., the number of rows) of each relation is as follows:

$$r_R = 1000,\ r_S = 2000,\ r_T = 3000,\ r_U = 4000.$$

Assume also that all attributes of the relations are of the same length and we use hash joins, so the cost of joining $X \in \{R, S, T, U\}$ and $Y \in \{R, S, T, U\}$ is approximately:

$$k(r_X \cdot c_X + r_Y \cdot c_Y)$$

where $k$ is a constant, and $r_X$ and $c_X$ ($r_Y$ and $c_Y$) denote the number of rows and the number of columns of $X$ ($Y$), respectively (we ignore the cost of producing the output relation). Finally, assume that the size of a join is always 0.2% of the size of the cross product (i.e., $r_{X \bowtie Y} = 0.002 \cdot r_X \cdot r_Y$).

Under the above assumptions, find the lowest cost plan for computing $R \bowtie S \bowtie T \bowtie U$ using *dynamic programming* and *left-deep* join trees. You need to complete the following table while finding the best plans (e.g., in the form of $((\square \bowtie \square) \bowtie \square) \bowtie \square$ in the last line) and associated costs.

| Subquery | Size | Cost | BestPlan |
|---|---|---|---|
| $R \bowtie S$ | 4000 | $6000k \quad [= (1000 \cdot 2 + 2000 \cdot 2)k]$ | $R \bowtie S$ |
| $R \bowtie T$ | | | |
| $R \bowtie U$ | | | |
| $S \bowtie T$ | | | |
| $S \bowtie U$ | | | |
| $T \bowtie U$ | | | |
| $R \bowtie S \bowtie T$ | | | |
| $R \bowtie S \bowtie U$ | | | |
| $R \bowtie T \bowtie U$ | | | |
| $S \bowtie T \bowtie U$ | | | |
| $R \bowtie S \bowtie T \bowtie U$ | | | |

(*Answer*)

| Subquery | Size | Cost | Best Plan |
|---|---|---|---|
| $R \bowtie S$ | 4000 | $6000k \quad [= (1000 \cdot 2 + 2000 \cdot 2)k]$ | $R \bowtie S$ |
| $R \bowtie T$ | 6000 | $8000k \quad [= (1000 \cdot 2 + 3000 \cdot 2)k]$ | $R \bowtie T$ |
| $R \bowtie U$ | 8000 | $10000k \quad [= (1000 \cdot 2 + 4000 \cdot 2)k]$ | $R \bowtie U$ |
| $S \bowtie T$ | 12000 | $10000k \quad [= (2000 \cdot 2 + 3000 \cdot 2)k]$ | $S \bowtie T$ |
| $S \bowtie U$ | 16000 | $12000k \quad [= (2000 \cdot 2 + 4000 \cdot 2)k]$ | $S \bowtie U$ |
| $T \bowtie U$ | 24000 | $14000k \quad [= (3000 \cdot 2 + 4000 \cdot 2)k]$ | $T \bowtie U$ |
| $R \bowtie S \bowtie T$ | 24000 | $24000k \quad [= cost(R \bowtie S) + (4000 \cdot 3 + 3000 \cdot 2)k]$ | $(R \bowtie S) \bowtie T$ |
| $R \bowtie S \bowtie U$ | 32000 | $26000k \quad [= cost(R \bowtie S) + (4000 \cdot 3 + 4000 \cdot 2)k]$ | $(R \bowtie S) \bowtie U$ |
| $R \bowtie T \bowtie U$ | 48000 | $32000k \quad [= cost(R \bowtie T) + (6000 \cdot 3 + 4000 \cdot 2)k]$ | $(R \bowtie T) \bowtie U$ |
| $S \bowtie T \bowtie U$ | 96000 | $54000k \quad [= cost(S \bowtie T) + (12000 \cdot 3 + 4000 \cdot 2)k]$ | $(S \bowtie T) \bowtie U$ |
| $R \bowtie S \bowtie T \bowtie U$ | 192000 | $128000k \quad [= cost((R \bowtie S) \bowtie T) + (24000 \cdot 4 + 4000 \cdot 2)k]$ | $((R \bowtie S) \bowtie T) \bowtie U$ |

---

After solving the above problems, please state the amount of time spent for this assignment. Feel free to add comments or suggestions if any.