Database Systems, CSCI 4380-01
Homework # 8 Answers
Due Thursday May 3, 2018 at 11 PM

Homework Statement.

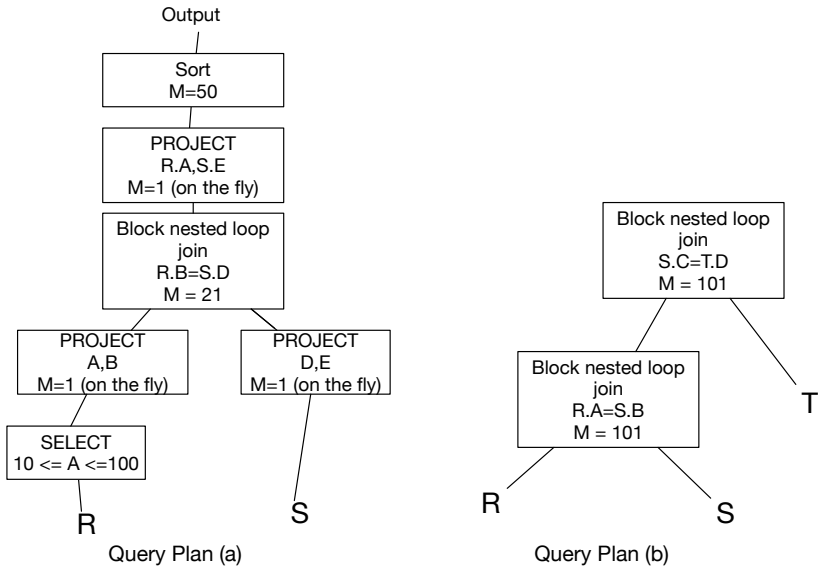

Figure 1: Query trees for Question 1

Question 1 (30 points). Estimate the cost of the query trees in Figure 1 given the statistics below. Show all your work.

A disk page and memory block is 8KB total, use 8,000 bytes for simplicity.

Relation `R(A,B,C,D)`: Pages(R)=20,000, Tuples(R)=400,000. Attribute A takes 4 bytes and B takes 20 bytes to store.

Values(R.A)=2,000 where Minval(R.A)=1 and Maxval(R.A)=2,000.

Values(R.B)=40,000

Relation `S(B,C,D,E,F,G)`: Pages(S)=80,000, Tuples(S)=1,000,000. Attribute D tables 20 bytes to store and E takes 40 bytes to store.

Values(S.D)=30,000, Values(S.C)=100,000, Values(S.B)=10,000.

Relation `T`: Pages(T)=10,000, Tuples(T)=100,000, Values(T.D)=20,000.

Answer 1(a).

Select from R, sequential scan, cost = **20,000 pages**.

Expected output: (90/2000)*400000= **18000 tuples**

Projection in memory, has no additional cost.

After projection, the result takes 24 bytes. 8000/24=333 tuples per page.

So, 18000 tuples fit in 18000/333=55 pages.

Given block nested loop with 21 blocks, we need to **read S 3** times.

S after projection: 60 bytes per tuple, 8000/60=133 tuples per page.

S will then fit in 1,000,000/133=7519 pages after projection.

So, during join, we read S once fully, **80,000 pages** do one pass of join, and also write it out to temporary storage for **7519** pages.

The remaining 2 passes of the join is uses the temporary table, 2*7519= **15,038 pages.**

The size of join: tuples(R)*tuples(S)*1/(values(R.B), values(S.D)). After selection, tuples(R)=18,000.

Given Values(S.D)= 30,000 and tuples(R) is less than 30,000, we will use 30,000 as max value.

Join is expected to have: 18,000*1,000,000/30,000= 600,000 tuples.

Each join tuple after projection is 4+40=44 bytes, hence we can fit 8000/44=181 tuples per page. Projection has no additional cost.

The join output is 600,000/181= 3315 pages.

Sort step 1 creates : 3315/50=67 sorted groups, with an additional cost of writing the **3315 pages**.

Sort step 2 merges 67 groups to 2 groups, cost of reading and writing the output result, **2*3315=6630 pages**

Sort step 2 merges 67 groups to 1 group and otuputs, cost of reading the output result one more time, **3315 pages**

Total cost = 20,000+80,000+7519+15,038+4*3315= 135817 pages.

Answer 1(b).

First join, cost: 20,000 + 20,000*80,000/100= **16,020,000 pages**

Size of output: 400000*1000000/(max(2000,10000))= 40000000 tuples.

R has 400,000/20,000=20 tuples per page, each tuple is 400 bytes long (8000/20)

S has 1,000,000/80,000 = 12 tuples per page, each tuple is 666 bytes long.

The join is 1066 bytes long, a page can store about 7 tuples per page.

The join result will be **5,714,286** pages.

We will read T: **57,142** times (oh boy!), with an additional cost of 10,000*57,142=**571,420,000 pages**.

Total cost: 571,420,000 + 16,020,000 = **587,440,000 pages.**

(Incidentally, if a page read is like 1 ms which is quite fast , this would take 163 hours!)

Question 2 (20 points). Suppose after a crash, you found the following log entries on disk. Furthermore, you are also for each data page on disk, the LSN of the last recorded log change on that page.

Discuss where the analysis step starts from and what its result is.

Then, discuss which operations will be checked for a REDO, and which update operations need to be redone (and in which order).

Finally, show which operations need to be undone (and in which order).

Show enough detail to explain how you arrived at your solutions.

LOG:

LSN	LOG ENTRY	PREVLSN
101	T1 update P1 10 20	-
102	T3 update P3 A B	-
103	commit T3	102
104	T2 update P5 XY ZZ	-
105	T4 update P3 B C	-
106	T2 update P4 5 8	104
107	rollback T2	106
108	checkpoint	
109	TT: T2 107, T4 105, T1 101	
110	DPT: P4 106, P3 105	
111	end checkpoint	
112	T2 undo 106 (P4 8 5)	104
113	T5 update P4 5 12	-
114	T5 update P6 X1 Y2	113
115	T1 update P1 20 30	101
116	commit T5	114
117	T1 update P5 ZZ RT	115
118	T7 update P4 12 15	-
119	T6 update P2 dd aa	-
120	commit T1	117
121	T7 update P1 30 40	118
122	T8 update P6 Y2 Z3	-

DATA PAGES:

PAGE ID	LSN OF LAST UPDATE
P1	115
P2	119
P3	105
P4	113
P5	117
P6	122

Answer.

After analysis, we find the following:

TT. T2 104 (undoing), T4 105, T6 119, T7 121, T8 122

DPT. P4 106, P3 105, P6 114, P1 115, P5 117, P2 T6

We will start the REDO process from LSN=105 (smallest value in DPT)

The following operations will be REDOne:

118 T7 update P4 12 15
121 T7 update P1 30 40

We will then abort and undo T4, T6, T7, T8 and continue to UNDO T2. Here is the order of operations for UNDO:

122
121
119
118
105
104

Question 3 (30 points). You are given the following three different (possible) schedules for the same four transactions. For each schedule, answer the following.

$$
\begin{aligned}
S1 &= r_1(Z)\ w_1(Z)\ r_1(Y)\ r_2(X)\ r_3(Z)\ r_3(X)\ w_1(Y)\ r_2(Y)\ w_3(X)\ r_4(K) \\
&\quad\ w_3(Z)\ r_4(X)\ w_4(K)\ w_2(Y) \\
S2 &= r_1(Z)\ w_1(Z)\ r_1(Y)\ r_4(K)\ r_4(X)\ r_2(X)\ r_3(Z)\ r_3(X)\ w_4(K)\ w_1(Y) \\
&\quad\ r_2(Y)\ w_3(X)\ w_3(Z)\ w_2(Y) \\
S3 &= r_1(Z)\ w_1(Z)\ r_1(Y)\ r_4(K)\ r_4(X)\ w_3(X)\ w_3(Z)\ r_2(X)\ r_3(Z) \\
&\quad\ r_3(X)\ w_4(K)\ w_1(Y)\ r_2(Y)\ w_2(Y)
\end{aligned}
$$

1. List all conflicts and draw the conflict graph.
2. Based on your conflict graph, discuss whether it is serializable or not, and why.
3. Discuss whether it is possible to obtain this schedule if two phase locking is used? Discuss why or why not.

Answers. (S1)

$$
\begin{aligned}
r_1(Z)\ w_3(Z) \\
w_1(Z)\ w_3(Z) \\
w_1(Z)\ r_3(Z) \\
r_2(X)\ w_3(X) \\
w_3(X)\ r_4(X) \\
w_1(Y)\ r_2(Y) \\
r_1(Y)\ w_2(Y)
\end{aligned}
$$

Conflicts: $T1 \rightarrow T2, T1 \rightarrow T3, T2 \rightarrow T3, T3 \rightarrow T4$. There are no cycles, so this schedule is serializable. A potential serial schedule is: T1, T2, T3, T4.

Here is a possible sequence of locks that shows this schedule is possible under 2PL.

$$
\begin{aligned}
&sl1(Z)\, r_1(Z)\, xl1(Z)\, w_1(Z)\, sl1(Y)\, r_1(Y)\, sl2(X)\, r_2(X) \\
&xl_1(Y)\, release_1(Z)\, sl3(Z)\, r_3(Z)\, sl_3(X)\, r_3(X)\, w_1(Y)\, release_1(Y)\, sl_2(Y)\, r_2(Y) \\
&xl_2(Y)\, release_2(X)\, xl_3(X)\, w_3(X)\, sl_4(K)\, r_4(K) \\
&xl_3(Z)\, w_3(Z)\, release_3(X)\, sl_4(X)\, r_4(X) \\
&xl_4(K)\, w_4(K)\ w_2(Y)
\end{aligned}
$$

Answers. (S2) Conflicts

$$r_1(Z)\ w_3(Z)$$
$$w_1(Z)\ w_3(Z)$$
$$w_1(Z)\ r_3(Z)$$
$$r_2(X)\ w_3(X)$$
$$r_4(X)\ w_3(X)$$
$$w_1(Y)\ r_2(Y)$$
$$r_1(Y)\ w_2(Y)$$

Conflicts: $T1 \rightarrow T2, T1 \rightarrow T3, T2 \rightarrow T3, T4 \rightarrow T3$. There are no cycles, so this schedule is serializable. A potential serial schedule is: T1, T2, T4, T3. There are other serials schedule possible.
Here is a potential sequence of locks:

$$sl_1(Z)\, sl_1(Y)\, sl_4(K)\, sl_4(X)\, sl_2(X)$$
$$r_1(Z)$$
$$xl_1(Z)$$
$$w_1(Z)\ r_1(Y)\ r_4(K)\ r_4(X)\ r_2(X)$$
$$xl_1(Y)\, release_1(Z)\, sl_3(Z)\, sl_3(X)\, xl_4(K)$$
$$r_3(Z)\ r_3(X)\ w_4(K)\ w_1(Y)$$
$$release_1(Y)\, sl_2(Y)\, release_4(X)\, xl_3(X)\, release_3(Z)\, xl_3(Z)$$
$$r_2(Y)\ w_3(X)\ w_3(Z)$$
$$xl_2(Y)$$
$$w_2(Y)$$

Answers. (S3) Conflicts

$$r_1(Z)\ w_3(Z)$$
$$w_1(Z)\ w_3(Z)$$
$$w_1(Z)\ r_3(Z)$$
$$r_1(Y)\ w_2(Y)$$
$$w_1(Y)\ w_2(Y)$$
$$r_4(X)\ w_3(X)$$
$$w_3(X)\ r_2(X)$$
$$r_1(Y)\ w_2(Y)$$

Conflicts: $T1 \rightarrow T2, T1 \rightarrow T3, T3 \rightarrow T2, T4 \rightarrow T3$. There are no cycles, so this schedule is serializable. A potential serial schedule is: T4, T1, T3, T2 or T1, T4, T3, T2.
This schedule is not possible under 2PL, due to the following sequence:

$$xl_3(X)\ w_3(X)\, release_3(X)\, sl_2(X)\, r_2(X)\ <\text{T3 cannot lock}>\ r_3(X)$$

Because T3 must lock for the first write, then release for T2 to read X. After releasing a lock, it cannot lock it again.

Question 4 (20 points). For this last question of the homework, your job is to optimize at least one of the queries given in the answer to Homework #5.

To optimize a query, first you will look at its estimated cost. To do this, you can simply add the word `explain` to the beginning of the query. For example:

```
olympics=> explain select * from summer_medals where id = 5 ;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Index Scan using summer_medals_pkey on summer_medals  (cost=0.29..8.30 rows=1 width=22)
   Index Cond: (id = 5)
(2 rows)

olympics=> explain select * from summer_medals where oid = 5 ;
                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on summer_medals  (cost=0.00..588.56 rows=1 width=22)
   Filter: (oid = 5)
(2 rows)


olympics=> explain select * from summer_medals where oid =5 order by medal;
                               QUERY PLAN
-------------------------------------------------------------------------
 Sort  (cost=588.57..588.58 rows=1 width=22)
   Sort Key: medal
   ->  Seq Scan on summer_medals  (cost=0.00..588.56 rows=1 width=22)
         Filter: (oid = 5)
(4 rows)
```

In each query plan, you get two estimated costs (e.g. `0.29 and 0.83`). The first one is the time to the first answer (which is non-zero given the unavoidable cost of scanning the index first) and the second one is the cost of getting all the answers. In the second query, the initial cost is zero because you can start to produce tuples as soon as you start scanning the relation (assuming you find some matching tuples). You can see that the cost of the third query is the same for both first and all results because sort is a blocking query, you cannot return any results until the sort is complete. Then, you can return all the results.

In this question, you are asked to improve the full running time (the second number) of one of the queries by either rewriting the query or by introducing an index. To show the result, you must document the full query plan before your change and the one after in your answer.

To allow you to run this part, you must create the hw#5 data on your computer. I am happy to do this for you on the DB server as well. Expect this to be available by the end of the weekend. In the meantime, you can work on improvements by query rewriting or think of good indices. We will discuss this topic on monday in more detail.

Who will have the best improvement? Technically we only need one. But if you want to show us more than one, feel free.

Answer. We will post examples later.