Database Systems

Instructor: Hao-Hua Chu
Fall Semester, 2005

Assignment 6: Query Optimization, Transaction Management, & Concurrency Control

Answers

1. *r1* needs 800 pages, and *r2* needs 1500 pages
   a. Simple Nested Loop Join (20%)
   The total cost
   = Cost of Scanning *r1* + #*r1* tuples* Cost of Scanning *r2*
   = 800 + 20000*1500
   = 30000800 page I/Os

   b. Block Nested Loop Join (20%)
   The total cost
   = Cost of Scanning *r1* + #*r1* blocks* Cost of Scanning *r2*
   = 800 + (800 / (102 – 2))*1500
   = 12800 page I/Os

   c. Hash-Join (20%)
   100 > $\sqrt{1500}$ > $\sqrt{800}$ => No need to do recursive partitioning.

   The total cost
   = Cost in partitioning phase, read&write r1& r2 + Cost in matching phase, read r1& r2
   = 2(800 + 1500) + (800 + 1500)
   = 6900 page I/Os

2. a. (10%) Any non-serializable schedule is OK.
   Ex:

T1	*T2*
read(A)	
	read(B)
	read(A)
read(B)	
if A = 0 then B := B + 1;	
	if B = 0 then A := A + 1;
	write(A)
write(B)	

   b. (10%) There is no parallel execution resulting in a serializable schedule. Since a serializable schedule results in A = 0 or B = 0. Suppose we start with *T1* **read**(A). Then when the schedule ends, no matter when we run the steps of *T2*, B = 1. Now suppose we start executing *T2* prior to completion of *T1*. Then *T2* **read**(B) will give B a value of 0. So when *T2* completes, A = 1. Thus B = 1 and A = 1 $\rightarrow$ not (A = 0 or B = 0). Similarly for starting with *T2* **read**(B).

c. (10%)

```
T1:  S(A);
     read(A);
     X(B);
     read(B);
     if A = 0 then B := B + 1;
     write(B);
     release(A);
     release(B);

T2:  S(B)
     read(B);
     X(A);
     read(A);
     if B = 0 then A := A + 1;
     write(A);
     release(B);
     release(A);
```

d. (10%) Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

T1	*T2*
S(A)	
	S(B)
	read(B)
read(A)	
X(B)	
	X(A)

The transactions are now deadlocked.