Name:____

CMSC 661 – Principles of Database Systems Spring 2011, Handout 3

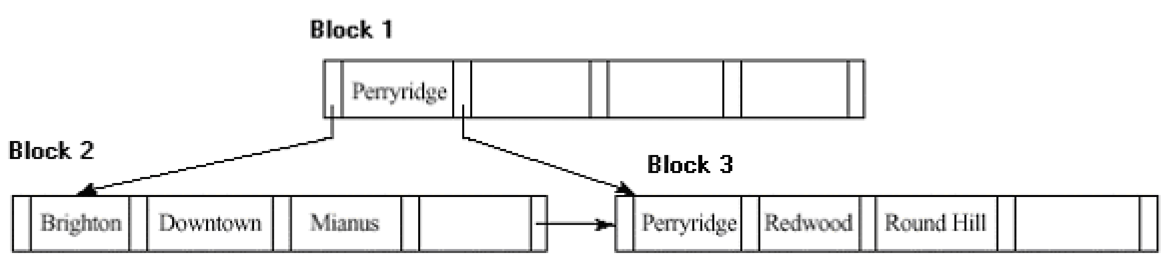
Sections covered since Exam II: 10.1 to 10.9, 11.1 to 11.6, 12.1 to 12.5.3, 12.7, 13.1 to 13.3.3, 14.1 to 14.10.

1. Identify each part (A through F) of the slotted-page structure used to store variable-length records.

A	B	C	D	E	F

A = # of entries and pointer to free space
B, C = pointers to records
D = free space
E, F = records

2. For the B+ tree below,
   a. What is n?
      5
   b. How many values can be stored in non-leaf nodes?
      $\lceil n/2 \rceil$ to n, which is 3 to 5
   c. Would a new value fit into one of the existing leaf nodes?
      Yes, there's room in both leaf nodes, which can hold $\lceil (n-1)/2 \rceil$ to n-1 nodes, which is 2 to 4
   d. In which block would the value "Frederick" be stored?
      In block 2
   e. In which block would the value "Arbutus" be stored?
      In block 2



3. Consider the following schedule.

Time	T1	T2
1	read(A)	
2	write(A)	
3		read(A)
4	read(B)	
5	write(B)	
6		write(A)

Is the schedule conflict serializable as <T1, T2>? Explain why or why not.

Yes. You can swap instructions at times 3 and 4 (they work on different objects). You can then swap instructions at times 4 and 5 (they work on different objects). This results in <T1, T2>.

Although it wasn't part of the question, note that it is NOT conflict serializable at <T2, T1>. To do this, you would need to move the instruction at time 3 up to time 1. However, you can't swap the instructions at time 2 and 3, because they affect the same object and at least one of them is a "write".

4. Consider the following SQL query using R (A, B, C, D) and S (D, E), where r(R) and s(S). Consider the relational algebra expression, derived from the SQL. Using equivalence rules, suggest an equivalent relational algebra expression that is more efficient.

```
select  s.E, r.A
from    r, s
where   r.D = s.D
and     s.D = 'Baltimore'
and     r.B = 'Sales'
```

$\Pi_{s.E,\ r.A}\ (\sigma_{r.D = s.D\ \wedge\ s.D = \text{'Baltimore'}\ \wedge\ r.B = \text{'Sales'}}\ (r \times s))$

There are a number of solutions. Consider our rules of thumb, and perform the select BEFORE the cross product.

$\boldsymbol{\Pi_{s.E,\ r.A}\ (\sigma_{r.D = s.D}\ (_{r.B = \text{'Sales'}}(r) \times {}_{s.D = \text{'Baltimore'}}(s)))}$

5. Given R (A, B, C, D) and S (D, E), where r(R) and s(S), assume that r has 1,000,000 rows with 100 rows stored per block, s has 100,000 rows with 500 rows stored per block, the block seek time is 0.4 microseconds, and the block transfer time is 0.1 microseconds, there is a primary index on A and a secondary index on B. Assume the height of any index used is 5. Assume there are 5 rows where B = 'Baltimore'. How long will it take to execute the following statements?

$\sigma_{B = \text{'Baltimore'}} (r)$

You need algorithm A5 for secondary index, equality on non-key.

(index height + number of rows) * (seek time + transfer time)

$(5 + 5) * (0.4 + 0.1) * 10^{-6}$ seconds

$= 5 * 10^{-6}$ seconds

$r \bowtie s$

The Index-Nested-Loop join won't work, since there's no index on the join attribute (D). This leaves us with Nest-Loop or the Block-Nested-Loop joins.

Here's the answer for the Block-Nested-Loop join.

To retrieve data from r, we need -- $b_r * t_S + b_r * t_T$

To retrieve data from s, we need -- $b_r * t_S + b_r * b_s * t_T$

Note that there are 10,000 blocks in r and 200 blocks in s.

10,000 * 0.4 * 10-6 + 10,000 * 0.1 * 10-6 + 10,000 * 0.4 * 10-6 + 10,000 * 200 * 0.1 * 10-6

= 0.209 seconds

Out of curiosity, is it faster than the Nested-Loop join? Let's see ...

To retrieve data from r, we need -- $b_r * t_S + b_r * t_T$

To retrieve data from s, we need -- $n_r * t_S + n_r * b_s * t_T$

Note that there are 10,000 blocks in r and 200 blocks in s.

Note that the Nested-Loop join uses n_r as a multiplier.

$10{,}000 * 0.4 * 10^{-6} + 10{,}000 * 0.1 * 10^{-6} + 1{,}000{,}000 * 0.4 * 10^{-6} + 1{,}000{,}000 * 200 * 0.1 * 10^{-6}$

= 20.405 seconds