Midterm 2 Exam

⊕ 75 minutes

⊕ Avoid asking questions - If something is unclear, document your assumptions and move on.

⊕ Write your name on the first page.

⊕ We will not grade anything on the backs of test sheets.

⊕ Be neat and precise. We will not grade answers we cannot read.

⊕ Draw simple figures if it will make your answers clearer.

⊕ Good luck, and remember that brevity is the soul of wit.
1. (10 pts) **Definitions**

(a) **dirty read** -
   **answer:** reading value written by uncommitted trans

(b) **rigorous 2PL** -
   **answer:** holding both shared and exclusive locks until commit

(c) **wound-wait** -
   **answer:** Method of breaking deadlock by having older transaction kill the younger. Younger transactions always wait for older.

(d) **lock point** -
   **answer:** Time of last lock acquire by a transaction in 2PL.

(e) **snapshot isolation** (how reads/writes satisfied, how conflicts with other transactions handled...)
   **answer:** must say all reads from same snapshot, conflicting write sets
2. (10 pts) **Precedence graphs**

Assume the following schedule:

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>write(A)</td>
<td>write(B)</td>
<td>read(A)</td>
<td>read(B)</td>
<td>read(D)</td>
<td>write(D)</td>
<td></td>
</tr>
<tr>
<td>read(A)</td>
<td>read(B)</td>
<td>read(D)</td>
<td>read(D)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read(C)</td>
<td>read(B)</td>
<td>read(D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) (5 pts) Draw the precedence graph for this schedule.

![Precedence Graph](image)

**answer:**

(b) (5 pts) Is the schedule conflict serializable? Explain.

**answer:** No cycle, so YES it is conflict serializable
3. (15 pts) $B^+$-trees
Assume that the trees in this exercise have $n=4$. For full credit you will need to follow the exact algorithms described in class. It is not sufficient to draw a new tree that satisfies the B+ tree criteria.

(a) (8 pts) Starting with the tree drawn above, insert "EF" in this tree and redraw the structure.

answer: split leaf and parent, no change in height
(b) (7 pts) Starting with the same original tree (repeated here), delete ’CC’.

answer: merge, then redistribute pointers between two middle nodes and appropriately update parent.
4. RAID (10 pts)

(a) (5 pts) For each of RAID 0, 1, and 5, how much data can we store on a RAID array? Assume the RAID array has four 1-TB disks.

answer: 4, 2, 3

(b) (5 pts) Explain the process of writing a block to a RAID 5 array where a single disk has failed, and that disk holds the block in question.

answer: 1) Read all disks, including parity 2) calc OLD as XOR of all of them, 3) calc Par’=Par XOR old XOR new, 3) write P’
5. **Query Processing** (15 pts)
   Assume a natural join between two relations $R$ and $S$, with $n_r = 1000$, $n_s = 600$, $b_r = 250$, and $b_s = 200$.

   (a) (5 pts) Assuming room for only three blocks in memory, **how many block reads** are required for nested loop join? No indexes, and make the most efficient choice of inner and outer relations.

   **answer:** $n_r \times b_s + b_r = 1000 \times 200 + 250 = 200,250$ (r outer relation), or $= 600 \times 250 + 200$ (s outer). Clearly S outer is better $= 150,200$

   (b) (5 pts) Similarly, how many for block nested-loop join under similar conditions?

   **answer:** $b_r \times b_s + b_r = 50,250$ for r outer, or $= 200 \times 250 + 200 = 50,200$ w/ s outer.
(c) (5 pts) Assume 5 blocks of memory, 80 tuples in a relation, 4 tuples per page. Give the total number of blocks read or written for a merge-sort. Use standard assumptions.

answer: There are 20 pages of data. Sort them into four runs of 20 tuples (5 pages) each. We can merge all four runs in one pass (1 page for each of 4 runs, 1 page for output). Cost is then a total of 4 reads and writes of all pages (4 * 20), but we don’t count last write, so \( blocks = 3 \times 20 = 60 \).
6. **Serializability** (15 pts)

*Assume* $A = 20, B = 10$ to start each of the following.

(a) (5 pts) Circle the most specific term (i.e. if a schedule is “conflict serializable”, circle that instead of “serializable”).

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>read(A)</td>
<td>read(A)</td>
</tr>
<tr>
<td>$A = A - 60$</td>
<td>$A = A - 10$</td>
</tr>
<tr>
<td>write(A)</td>
<td>write(A)</td>
</tr>
<tr>
<td></td>
<td>read(B)</td>
</tr>
<tr>
<td></td>
<td>$B = B - 60$</td>
</tr>
<tr>
<td></td>
<td>write(B)</td>
</tr>
<tr>
<td></td>
<td>read(B)</td>
</tr>
<tr>
<td></td>
<td>$B = B - 10$</td>
</tr>
<tr>
<td></td>
<td>write(B)</td>
</tr>
</tbody>
</table>

**Answer:** conflict serializable, can make T2,T1 by moving T2’s B operations up past T1’s A operations. Conflict serial implies view and serializable.

(b) (5 pts) Circle the most specific term (i.e. if a schedule is “conflict serializable”, circle that instead of “serializable”).

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>read(A)</td>
<td>read(A)</td>
</tr>
<tr>
<td>tmp = .1 * A</td>
<td>$A = A - 20$</td>
</tr>
<tr>
<td>$A = A - tmp$</td>
<td></td>
</tr>
<tr>
<td>write(A)</td>
<td></td>
</tr>
<tr>
<td>read(B)</td>
<td>read(B)</td>
</tr>
<tr>
<td>$B = B + tmp$</td>
<td>$B = B + 20$</td>
</tr>
<tr>
<td>write(B)</td>
<td>write(B)</td>
</tr>
</tbody>
</table>

**Answer:** not serializable; results not equiv to either serial schedule. Not serial implies not conflict or view.
(c) (5 pts) Circle the most specific term (i.e. if a schedule is “conflict serializable”, circle that instead of “serializable”).

\begin{align*}
T1 & \quad T2 \\
\text{read}(A) & \quad \text{write}(A) \\
A &= A - 60 & \text{CONFLICT SERIALIZABLE} \\
\text{read}(B) & \quad \text{write}(B) \\
B &= B - 10 & \text{VIEW SERIALIZABLE} \\
\text{write}(B) & \quad \text{read}(B) \\
B &= B - 60 & \text{SERIALIZABLE} \\
\text{read}(A) & \quad \text{write}(B) \\
A &= A - 10 & \text{NOT SERIALIZABLE} \\
\text{write}(A) &
\end{align*}

**answer:** Serializable.

7. **Time-stamp Concurrency Control** (10 pts)
Assume starting timestamps are ordered as follows: $TS(T_1) < TS(T_2) < TS(T_3) < TS(T_4) < TS(T_5)$. Put a line through each access where a transaction aborts.

\begin{align*}
T1 & \quad T2 & \quad T3 & \quad T4 & \quad T5 \\
\text{read}(x) & \quad \text{write}(y) & \quad \text{read}(y) & \quad \text{write}(t) & \quad \text{write}(p) \\
\text{write}(y) & \quad \text{read}(z) & \quad \text{read}(t) & \quad \text{read}(p) & \quad \text{read}(t) \\
\text{write}(p) & \quad \text{write}(t) & \quad \text{write}(t) & \quad \text{write}(z)
\end{align*}

**answer:** Aborts:
T4 aborts at read(t) because of T5’s prior write.
T3 aborts at write(t) because of prior write(t) by T5 (not because of T4’s read).
T1 aborts at write(p) because of prior read(p) by T5.
8. 3NF (15 pts)

(a) (10 pts) For relation $R=(ABCDE)$, compute the canonical cover of:
$A \rightarrow B, B \rightarrow D, AC \rightarrow B, E \rightarrow D$

**Answer:**
C extraneous in $AC \rightarrow B$, merge two $A \rightarrow B$, left with:
$A \rightarrow B, B \rightarrow D, E \rightarrow D$.

(b) (5 pts) Decompose this canonical cover into 3NF:

**Answer:** Candidate key is $ACE$.

$R_1 = \{AB\}, R_2 = \{BD\}, R_3 = \{ED\}, R_4 = \{ACE\}$