ONID:

CS540 Database Management Systems
Winter 2018
School of Electrical Engineering & Computer Science
Oregon State University
Midterm Examination
Time Limit: 75 minutes

• Print your name and ONID below. In addition, print your ONID in the upper right corner of every page.

Name and ONID: ________________________________________________________________

• Including this cover page, this exam booklet contains 7 pages. Check if you have missing pages.

• The exam is closed book and closed notes. You are allowed to use scratch papers. No calculators or other electronic devices are permitted. Any form of cheating on the examination will result in a zero grade.

• Please write your solutions in the spaces provided on the exam. You may use the blank areas and backs of the exam pages for scratch work.

• Please make your answers clear and succinct; you will lose credit for verbose, convoluted, or confusing answers. Simplicity does count!

<table>
<thead>
<tr>
<th>Question:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>9</td>
<td>12</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Indexing

(a) (4 points) Explain the difference between a clustered index and an unclustered index.

**Solution:** The tuples in a relation with a clustered index are sorted based on the order of the search keys in the index. The tuples in a relation with an unclustered index may not be sorted based on the order of the search keys in the index.

(b) (5 points) Consider the relation `Student (ID, Name)` whose tuples have following values for attribute `ID`: 2, 4, 34, 8, 5. Construct a B+ tree index with degree 2, i.e., `d=2`, on attribute `ID` for relation `ID`. You need to show only the final B+ tree.

**Solution:** The root contains search key 5. The root has two leaf child with keys (2,4) and (5,8,34), respectively. Other reasonable solutions are acceptable. The solutions that get part of the B+ tree right may get partial points.
2. Query Processing: Join Algorithms
Consider relations $\text{Student}(ID, \text{Name})$ and $\text{Enrollment}(\text{CourseID}, \text{StudentID})$. The size of relation $\text{Student}$ is 25,000 blocks and the size of $\text{Enrollment}$ is 800 blocks. Assume that the query processor has to choose between the block-based nested loop join, the sort-merge join with two-pass multi-way merge-sort for the sorting phase, and the hash join algorithm to perform the join of $\text{Student} \Join_{\text{ID} = \text{StudentID}} \text{Enrollment}$.

(a) (4 points) Given that there are 30 blocks available in main memory, i.e., $M = 30$, which one of the aforementioned algorithms is the fastest one to perform the join? You should also provide the cost of join using your proposed algorithm.

Solution:
Because the size of the smallest relation is 800, we can use hash join algorithm whose cost is $3 \times (B(\text{Student}) + B(\text{Enrollment})) = \text{I/O accesses}$

(b) (4 points) Given that there are 10 blocks available in main memory, i.e., $M = 10$, which one of the aforementioned algorithms is the fastest one to perform the join? You should also provide the cost of join using your proposed algorithm.

Solution:
We can use the improved block-based nested loop join. The improved block-based nested loop is faster with cost $B(\text{Student}) \times B(\text{Enrollment}) / 10 = \text{I/O accesses}$. The solution $B(\text{Student}) + (B(\text{Student}) / 8) \times B(\text{Enrollment})$ is also acceptable. The solution that uses the original block-based nested loop is also acceptable.

(c) (4 points) Estimate the minimum number of required blocks in main memory, i.e., $M$, to perform the join using the sort-merge join with two-pass multi-way merge-sort for the sorting phase.

Solution:
We can perform the join using at least about 159 blocks in main memory.
3. Query Optimization: Logical Plans

Consider the following relations. Primary keys are underlined.

\[ \text{Student}(\text{StudentId}, \text{Name}, \text{DeptId}) \]
\[ \text{Department}(\text{DeptId}, \text{DeptName}) \]

The following query returns \text{StudentId} of every student whose name is 'John' and department is 'EECS'.

\[
\text{SELECT StudentId, Name, DeptName FROM Student, Department WHERE Student.DeptId = Department.DeptId and Student.Name = 'John' and DeptName = 'EECS'}
\]

(a) (4 points) Suggest an optimized logical query plan for the above query.

**Solution:**

An optimized query plan will select from relation Student the tuples that satisfy the condition:

\[ \text{Student.Name = 'John'} \]

and then apply projection on attributes StudentId and DeptId. It also selects from relation Department the tuples that satisfy the condition:

\[ \text{DeptName = 'EECS'} \]

and then apply projection on attribute StudentId. It then joins the two resulting relations on DeptId.
4. Query Optimization: Cost Estimation

Consider relations $R(A, B)$ and $S(B, C)$. Assume that $R$ contains 20,000 tuples, and that $S$ contains 50,000 tuples. Use Selinger-style, i.e., System-R style, cost estimation formulas to answer the following questions.

(a) (3 points) We want to compute $U = \sigma_{B>10} R$. Assume that we do not have any information about the number of distinct values and the range of values in the attribute $B$ in $R$. What is a reasonable estimate on the size of $U$?

Solution: $20000/3 = 6,667$

(b) (3 points) We want to compute $W = R \bowtie_{R.B=S.B} S$. We know that $V(R, B) = 500$ and $V(S, B) = 50$. (That is, attribute $B$ takes on 500 different values in $R$ and 50 different values in $S$.) What is a reasonable estimate on the size of $W$?

Solution: $20000 \times 50000 / \max(500, 50) = 2,000,000$
5. Concurrency Control: Serializability and 2PL

For the following schedule draw its serialization graph and state whether it is serializable. Is it a 2PL schedule? The actions are listed in the order they are scheduled and prefixed with the transaction name.

(a) (4 points) T1:W(X), T2:W(X), T2:R(Y), T1:W(Y), T1:Commit, T2:Commit.

Solution:
T1 → T2 → T1

Not serializable. Since it is not serializable, it is not 2PL.