Midterm Exam Csci 5708, Spring 2017

Instructions:
- Time allowed: 75 min
- This is a CLOSED book examination. One 8.5 x 11 sheet with notes is allowed. Notes cannot be shared.
- This examination has 3 questions. Each question has its sub questions.
- Put your name, student id, UMN email address, class group on the cover page.
- Please answer questions in the spaces provided. You may request additional sheets of paper from the instructor or continue your answer on the back of each paper.
- Please state your assumptions clearly.

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student ID</td>
<td></td>
</tr>
<tr>
<td>Email address</td>
<td></td>
</tr>
<tr>
<td>Class group No.</td>
<td></td>
</tr>
<tr>
<td>Course Name</td>
<td>Architecture and Implementation of DBMS</td>
</tr>
<tr>
<td>Course ID</td>
<td>Csci 5708</td>
</tr>
<tr>
<td>Semester</td>
<td>Spring</td>
</tr>
<tr>
<td>Year</td>
<td>2017</td>
</tr>
</tbody>
</table>
[File and index structures]

Q1. Consider the following “EMPLOYEE” table:

Table 1. EMPLOYEE

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Age</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Ahn</td>
<td>098-55-1234</td>
<td>26</td>
<td>Male</td>
</tr>
<tr>
<td>Ann Alam</td>
<td>123-45-6789</td>
<td>25</td>
<td>Female</td>
</tr>
<tr>
<td>Tim Liu</td>
<td>445-21-2842</td>
<td>32</td>
<td>Male</td>
</tr>
<tr>
<td>Lily Miller</td>
<td>111-09-4898</td>
<td>50</td>
<td>Female</td>
</tr>
<tr>
<td>Taylor Potter</td>
<td>931-21-4374</td>
<td>37</td>
<td>Female</td>
</tr>
<tr>
<td>Lily Miller</td>
<td>287-44-3341</td>
<td>29</td>
<td>Male</td>
</tr>
<tr>
<td>Derek Cooper</td>
<td>848-11-2967</td>
<td>35</td>
<td>Male</td>
</tr>
</tbody>
</table>

Q1a) Fill in Table 2 to build a bitmap index on the field “Gender” for the EMPLOYEE table. (The result may not fill all rows.) (5 points)

Table 2. Bitmap index on “Gender”

<table>
<thead>
<tr>
<th>Gender</th>
<th>Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>1010011</td>
</tr>
<tr>
<td>Female</td>
<td>0101100</td>
</tr>
</tbody>
</table>

Q1b) Construct a B+-tree index of order 2 (i.e., each index node can hold 2 keys and 3 pointers) on the field “Age” for the EMPLOYEE table. Show the final state of this B+-tree after inserting records with following values of “Age” field in this order: 26, 25, 32, 50, 37, 29, 35. Assume that when an index node splits, the median key value is stored into the right sibling. (15 points)

If you follow the order I give to insert the values, there is only one correct result. Some students ignored the final sentence in the question and get another result. I gave full marks.

A B+-tree is a balanced tree, which can be used to check your result. More details: textbook p617.
Q1c) Consider a dynamic-hash file structure on field “Age” for the EMPLOYEE table. The hash function family used is $h_0 = K \mod 2^0$, $h_1 = K \mod 2^1$, …, $h_4 = K \mod 2^4$. Assume that each disk block can hold 1 EMPLOYEE record. Records with “Age” field are inserted in this order: 26, 25, 32, 50, 37, 29, 35. Show the final state of the dynamic-hash file structure by drawing a dynamic-hash directory tree as well as contents of hash buckets. (15 points)

Now that in this question, a hash function family is given, the tree’s nodes are generated from the least significant bit. For example, if represented in binary form the last 4 digits of 50 is 0010 and 26 is 1010.

More details can be found in the explanation slides provided on course webpage.

<table>
<thead>
<tr>
<th>K</th>
<th>$K \mod 2^1$</th>
<th>$K \mod 2^2$</th>
<th>$K \mod 2^3$</th>
<th>$K \mod 2^4$</th>
<th>$K \mod 2^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>29</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
[Query processing]

Q2. Consider a natural join of two tables, namely, R(A, B) and S(B, C). The primary key of table R includes columns A and B, while the primary key of table S is column B. There are 20000 records in Table R, and 3000 records in Table S. A disk page can hold either 20 R records, or 10 S records. Main memory provides 102 pages as buffer.

Q2a) Among nested loop join, index-based nested loop join, and hash join algorithms, which one is the cheapest for a natural join on R.B = S.B if there is a clustering B+ tree index on S.B with depth = 2. Justify your answer using I/O cost models. (15 points)

Table R: 20000 records, 1000 pages
Table S: 3000 records, 300 pages

Nested loop join:
1. R is outer: 1000 + \(\frac{20000}{102-2} \times 300\) = 4000
2. S is outer: 300 + \(\frac{3000}{102-2} \times 1000\) = 3300

Index-based nested loop join:
Only S can be inner table. 1000 + 20000 \((2 + 1)\) = 61000

Hash join: 3 \times (1000 + 300) = 3900

Nested loop join with S as the outer table is the cheapest.

For index-based nested loop join, because the only available index is on S.B, only table S can be the inner table. In outer loop, table R is scanned once, resulting in 1000-page cost. For every record in table R, the matched records in table S is searched using the index. More: textbook p714.

Q2b) Among nested loop join, index-based nested loop join, and hash join algorithms, which one is the cheapest for a natural join on R.B = S.B if there is a secondary B+ tree index on R.B with depth = 3. Justify your answer using I/O cost models. (10 points)

The only changed cost is for index-based nested loop join:

Only R can be inner table. 300 + 3000 \(\left(3 + 1 + \frac{20000}{3000}\right)\) = 35000

Nested loop join with S as the outer table is the cheapest.

For index-based nested loop join, because the only available index is on R.B, only table R can be the inner table. In outer loop, table S is scanned once, resulting in 300-page cost. For every record in table S, the matched records in table R is searched using the index. The selectivity is estimated by the ratio of the number of records in two tables. If you did not estimate the selectivity, no point is taken.

More: textbook p714.
[Query optimization]

Q3. Consider three tables, namely, R(A, B), S(B, C), and T(C, D), representing a many-to-many relationship (table S) between entities represented by tables R and T. Assume that all fields are of the same data type, which are all integers.

Q3a) Draw an initial (canonical) query tree for the following query: (10 points)

```
SELECT *
FROM R, S, T
WHERE (R.B = S.B) AND (S.C = T.C) AND (R.A <=20) AND (T.D >=30)
```

\[\sigma(R.B = S.B) \land (S.C = T.C) \land (R.A \leq 20) \land (T.D \geq 30)\]

Q3b) Use tree transforms to heuristically optimize the canonical query tree in Q3a). Justify the correctness of the transformations by mentioning relevant properties of relational algebra such as commutativity of equi-joins. (15 points)

1. Cascade of \(\sigma\);
2. Commuting \(\sigma\) with \(\bowtie\);
3. Commutativity of \(\bowtie\);
4. Commutativity of \(\sigma\);
5. Associativity of \(\bowtie\)

There are two possible answers. Both are acceptable.
Q3c) Suppose that there are 3000 records in R, 6000 records in S, and 2000 records in T. One disk page can store either 20 R records, or 50 S records, or 10 T records. Main memory provides 42 memory pages as buffer. On average in the natural join each R records matches 2 S records, and each T records matches 3 S records. The selectivity of R.A <=20 and T.D >=30 are 0.5 and 0.4 respectively. The intermediate results are saved onto disk.

Among nested loop and hash join algorithms, which one is cheaper for the equi-join nodes in your final query tree in Q3b). Justify your answer with I/O cost model. (15 points)

Table R: 3000 records, 150 pages. After σ_{R.A<=20}, there are 1500 records, 75 pages.
Table S: 6000 records, 120 pages.
Table T: 2000 records, 200 pages. After σ_{T.D>=30}, there are 800 records, 80 pages.
There are two possible final query trees.
Query tree 1:

```
\sigma_{R.A<=20} \ R
\bowtie_{R.B=S.B} \ S
\sigma_{T.D>=30} \ T
\bowtie_{S.C=T.C}
```

For \bowtie_{R.B=S.B}, because the input of this join is the result of \sigma_{R.A<=20} and table S, the cost of nested loop is at least \(75 + \frac{75}{40} \times 120\), and the cost of hash join is \(3 \times (75 + 120)\). Because \(\frac{75}{40} < 3\), we can conclude that nested loop algorithm is cheaper.

For \bowtie_{S.C=T.C}, because the input of this join is the result of \bowtie_{R.B=S.B}, and the result of \sigma_{T.D>=30}, let’s denote \(x\) as the number of pages occupied by the results of \bowtie_{R.B=S.B}, the cost of nested loop is at least \(80 + \frac{80}{40} \times x\), and the cost of hash join is \(3 \times (80 + x)\). Thus, we can conclude that nested loop algorithm is cheaper.

Query tree 2

```
\sigma_{R.A<=20} \ R
\bowtie_{R.B=S.B} \ S
\sigma_{T.D>=30} \ T
\bowtie_{S.C=T.C}
```

The analysis is similar to query tree 1. In both cases, nested loop is cheaper.

5 points for estimating the number of pages of each table.
5 points to analyze \bowtie_{R.B=S.B};
5 points to analyze \bowtie_{S.C=T.C}

You only need to analyze one query tree you choose in Q3b).
Key point: The input of the join operations are not the original tables. You need to estimate the sizes of intermediate results according to the selectivity and the original tables’ sizes.