Exam 2

May 10th, 2013

CS525 - Final Exam Solutions
Instructions

- Things that you are not allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes

- The quiz is 120 minutes long

- Multiple choice questions are graded in the following way: You get points for correct answers and points subtracted for wrong answers. The minimum points for each question is 0. For example, assume there is a multiple choice question with 6 answers - each may be correct or incorrect - and each answer gives 1 point. If you answer 3 questions correct and 3 incorrect you get 0 points. If you answer 4 questions correct and 2 incorrect you get 2 points. . . .

- For your convenience the number of points for each part and questions are shown in parenthesis.

- There are 5 parts in this quiz
  1. SQL and Relational Algebra
  2. Index Structures
  3. Sorting
  4. Schedules and Concurrency Control
  5. Physical Optimization
Part 1  SQL and Relational Algebra (Total: 40 Points)

Consider the following database schema and instance:

<table>
<thead>
<tr>
<th>politician</th>
<th>party</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama</td>
<td>Democrats</td>
<td>m</td>
</tr>
<tr>
<td>Bush</td>
<td>Republicans</td>
<td>m</td>
</tr>
<tr>
<td>Paul</td>
<td>Republicans</td>
<td>m</td>
</tr>
<tr>
<td>Clinton</td>
<td>Democrats</td>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>office</th>
<th>title</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>President</td>
<td>500.000</td>
</tr>
<tr>
<td></td>
<td>Governor</td>
<td>100.000</td>
</tr>
<tr>
<td></td>
<td>Mister of Defence</td>
<td>600.000</td>
</tr>
<tr>
<td></td>
<td>Secretary of State</td>
<td>30.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>lobbyist</th>
<th>company</th>
<th>industry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP</td>
<td>gas &amp; oil</td>
</tr>
<tr>
<td></td>
<td>Nestle</td>
<td>food</td>
</tr>
<tr>
<td></td>
<td>Bushmaster</td>
<td>arms</td>
</tr>
<tr>
<td></td>
<td>Boeing</td>
<td>arms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sponsored</th>
<th>lobbyist</th>
<th>politician</th>
<th>year</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP</td>
<td>Obama</td>
<td>1990</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td>BP</td>
<td>Obama</td>
<td>2004</td>
<td>30,000,000</td>
</tr>
<tr>
<td></td>
<td>Bushmaster</td>
<td>Bush</td>
<td>1985</td>
<td>100,000</td>
</tr>
<tr>
<td></td>
<td>Bushmaster</td>
<td>Bush</td>
<td>2013</td>
<td>15,000,000</td>
</tr>
<tr>
<td></td>
<td>Boeing</td>
<td>Bush</td>
<td>2013</td>
<td>5,000,000</td>
</tr>
<tr>
<td></td>
<td>BP</td>
<td>Bush</td>
<td>1985</td>
<td>350,000</td>
</tr>
<tr>
<td></td>
<td>Boeing</td>
<td>Paul</td>
<td>2003</td>
<td>10,000</td>
</tr>
<tr>
<td></td>
<td>Boeing</td>
<td>Paul</td>
<td>2004</td>
<td>400,000</td>
</tr>
<tr>
<td></td>
<td>Nestle</td>
<td>Clinton</td>
<td>2030</td>
<td>40,000</td>
</tr>
</tbody>
</table>

Hints:

- When writing queries do **only** take the schema into account and **not** the example data given here. That is your queries should return correct results for all potential instances of this schema.

- Underlined attribute form the primary key of a relation.

- The attribute politician and office of relation held are foreign keys to the attributes name in relation politician and title in relation office respectively.

- The attribute lobbyist and politician of relation sponsored are foreign key to the attributes company in relation lobbyist and name in relation politician respectively.
**Question 1.1  (8 Points)**

Write an SQL statement that returns the company of all lobbyists that have sponsored at least one democrat politician during a year where this politician became President.

**Solution**

```sql
SELECT DISTINCT l.company
FROM politician p, office o, lobbyist l, held h, sponsored s
WHERE p.name = h.politician
   AND o.title = h.office
   AND l.company = s.company
   AND p.name = s.politician
   AND s.year = h.year
   AND o.title = 'President'
   AND p.party = 'Democrats'
```

The access to relations office and lobbyist can actually be avoided here:

```sql
SELECT DISTINCT s.lobbyist AS company
FROM politician p, held h, sponsored s
WHERE p.name = h.politician
   AND p.name = s.politician
   AND s.year = h.year
   AND h.office = 'President'
   AND p.party = 'Democrats'
```

**Question 1.2  (8 Points)**

Write an SQL statement that returns the total amount of lobbyist money retrieved per political party and company. For example, an example output that may be produced by this query is (Democrats, BP, 10,000) that is the format is (Party, Company, Sum of money). The example was wrong, solutions that group by politician and company are correct too.

**Solution**

```sql
SELECT party, lobbyist, sum(amount) AS total
FROM politician p, sponsored s
WHERE p.name = s.politician
GROUP BY party, lobbyist
```

Alternatively, an redundant additional join with relation lobbyist may be used.
Question 1.3  (10 Points)

Write an SQL statements that returns the name of the politician that retrieved the highest total amount of sponsorship.

Solution

\[
\begin{align*}
\text{SELECT } & \text{name} \\
\text{FROM } & ( \text{SELECT } \text{name}, \text{sum(amount)} \text{ AS total} \\
\text{FROM } & \text{politician p JOIN sponsored s ON (p.name = s.politician)} \\
\text{GROUP BY } & \text{name}) \text{ AS sub} \\
\text{WHERE } & \text{total = ( SELECT max(total) }
\end{align*}
\]

or using limit and order by

\[
\begin{align*}
\text{SELECT } & \text{p.name, sum(amount) AS total} \\
\text{FROM } & \text{politician p JOIN sponsored s ON (p.name = s.politician)} \\
\text{GROUP BY } & \text{name} \\
\text{ORDER BY } & \text{sum(amount) DESC} \\
\text{LIMIT } & 1;
\end{align*}
\]

Question 1.4  SQL → Relational Algebra (7 Points)

Translate the following SQL query into relational algebra (bag semantics).

\[
\begin{align*}
\text{SELECT sum(salary) AS allSal, party} \\
\text{FROM politician p,} \\
\text{office o,} \\
\text{held h,} \\
(\text{SELECT DISTINCT politican} \\
\text{FROM sponsored} \\
\text{WHERE lobbyist = 'BP' OR lobbyist = 'Boeing'}) s \\
\text{WHERE p.name = h.politician AND o.title = h.office AND s.politician = p.name} \\
\text{GROUP BY party}
\end{align*}
\]

Solution

\[
\begin{align*}
q &= \rho_{\text{allSal=sum(salary)(\pi_{\text{sum(salary).party}}(\pi_{\text{salary}}(\sigma_{\text{party}}(\text{join}))))}} \\
\text{join} &= ((\text{politician} \bowtie_{\text{name=politician}} \text{held} \bowtie_{\text{office=title}} \text{office}) \bowtie_{\text{name=politician}} s) \\
s &= \delta_{\text{politician}((\sigma_{\text{lobbyist='BP'}\lor\text{lobbyist='Boeing'}(\text{-sponsored}))}}
\end{align*}
\]
Question 1.5  Equivalences (7 Points)

Consider the following relational schemata:
\[ R(A, B), S(C, D) \].

Prove or disprove the equivalence (set semantics) shown below using the operator definition and algebra equivalence discussed in class. In each step of the proof write down which equivalence you are using (e.g., associativity of union or pushing selections).

\[
\sigma_{(A=5) \land (B>3)}(R \bowtie_{B=C} S) \equiv \sigma_{(A=5) \land (B>3)}(R) \bowtie \sigma_{C>3}(S)
\]

The join condition is missing in the right-hand side: \( B = C \)

Solution

Since the equivalence above is missing a join condition this is actually wrong. Thus, everybody that is disproving this is correct too.

\[
\sigma_{(A=5) \land (B>3)}(R \bowtie_{B=C} S) \equiv \sigma_{A=5}(\sigma_{B>3}(R \bowtie_{B=C} S))
\]

(Splitting of conjunctive selections into several selections)

\[
\sigma_{A=5}(\sigma_{B>3}(R \bowtie_{B=C} S)) \equiv \sigma_{B>3}(\sigma_{A=5}(R \bowtie_{B=C} S))
\]

(Commutativity of selection)

\[
\sigma_{B>3}(\sigma_{A=5}(R \bowtie_{B=C} S)) \equiv \sigma_{B>3}(\sigma_{A=5}(R) \bowtie_{B=C} \sigma_{C>3}(S))
\]

(Direct pushdown of selections through joins)

\[
\sigma_{B>3}(\sigma_{A=5}(R) \bowtie_{B=C} \sigma_{C>3}(S)) \equiv \sigma_{(A=5) \land (B>3)}(R) \bowtie \sigma_{C>3}(S)
\]

(Combining adjacent selections using conjunction and commutativity of boolean AND.)
Part 2  Index Structures (Total: 25 Points)

Question 2.1  B+-tree Operations (15 Points)

Given is the B+-tree shown below (n = 2). When splitting or merging nodes follow the conventions used in the last assignment:

- **Leaf Split**: In case a leaf node need to be split during insertion and n is even, the left node should get the extra key. E.g., if n = 2 and we insert a key 4 into a node [1,5], then the resulting nodes should be [1,4] and [5]. For odd values of n we can always evenly split the keys between the two nodes. In both cases the value inserted into the parent is the smallest value of the right node.

- **Non-Leaf Split**: In case a non-leaf node needs to be split and n is odd, we cannot split the node evenly (one of the new nodes will have one more key). In this case the “middle” value inserted into the parent should be taken from the right node. E.g., if n = 3 and we have to split a non-leaf node [1,3,4,5], the resulting nodes would be [1,3] and [5]. The value inserted into the parent would be 4.

- **Leaf Underflow**: In case of a leaf underflow your implementation should first try to redistribute values from a sibling and only if this fails merge the node with one of its siblings. Both approaches should prefer the left sibling. E.g., if we can borrow values from both the left and right sibling, you should borrow from the left one.

Execute the following operations and write down the resulting B+-tree after each operation.

insert(6),insert(9),insert(8),delete(2),delete(6)

![B+-tree diagram]

Solution
insert(6)

insert(9)

insert(8)

delete(2)

delete(6)
Question 2.2  Extensible Hashing (10 Points)

Consider the extensible Hash index shown below. Each page holds two keys. Execute the following operations insert(6), insert(2), insert(7), delete(2) and write down the resulting index after each operation. Assume the hash function assigns values as follows: \( h(6) = 0011, h(2) = 0000, h(7) = 1111 \). The original version was \( h(6) = 0010 \). Using this variant is correct too!

```
0 1
  0010 0100
0111 1010 1011
```

**Solution**

**insert(6)**
```
00 01 10 11
  0010 0011
0100 1010 1011
```

**insert(2)**
```
00 001 010 011 100 101 110 111
  0000 0010 0011 0100 1010 1011 1111
```

**insert(7)**
```
00 001 010 011 100 101 110 111
  0000 0010 0111 0100 1010 1011 1111
```

**delete(2)**
```
00 01 10 11
  0010 0011 0100 1010 1011 1111
```
Part 3   Sorting (Total: 10 Points)

Question 3.1   External Sorting (10 = 5 + 5 Points)

Assume we have $M = 101$ memory pages available for sorting and have to sort a relation $R$ with $B(R) = 20,000$ pages and $N(R) = 80,000$ records.

1. Compute the number of I/O operations needed to sort $R$ using external merge sort without using a min-heap during run generation.

2. Assume there is a clustered B+-tree index on the sort attribute(s) with $K = 50$ keys in each leaf node and height 4. How many I/Os operations are needed to sort relation $R$ if at the beginning of the sort none of the pages of the B+-tree are in memory.

Solution

1. $2 \cdot B(R) \cdot (1 + \left\lfloor \log_{M-1}(B(R)/M) \right\rfloor) = 2 \cdot 20,000 \cdot (1 + 2) = 120,000$ I/Os

2. clustered B+-tree
   - $N(R)/K + HT(I) + B(R) = 80,000/50 + 4 + 20,000 = 21,604$ I/Os
Part 4 Schedules and Concurrency Control (Total: 15 Points)

Question 4.1 Schedule Classes (15 Points)

Indicate which of the following schedules belong to which class. Recall transaction operations are modelled as follows:

- \( w_1(A) \) transaction 1 wrote item A
- \( r_1(A) \) transaction 1 read item A
- \( c_1 \) transaction 1 commits
- \( a_1 \) transaction 1 aborts

\[
S_1 = r_1(A), w_2(A), r_1(B), c_1, w_3(B), r_3(B), c_3, c_2
\]

\[
S_2 = r_2(B), r_1(B), r_3(A), w_1(B), r_2(C), w_2(C), c_2, w_1(B), c_1, w_3(A), c_3
\]

\[
S_3 = r_1(A), w_2(B), r_1(B), w_1(B), r_2(A), c_1, w_2(A), c_2
\]

\[
S_4 = r_2(B), r_1(B), r_2(A), w_2(A), w_1(B), c_2, c_1
\]

- S1 is recoverable
- S1 is cascade-less
- S1 is conflict serializable
- S1 is 2PL
- S1 is SS2PL

- S2 is recoverable
- S2 is cascade-less
- S2 is conflict serializable
- S2 is 2PL
- S2 is SS2PL

- S3 is recoverable
- S3 is cascade-less
- S3 is conflict serializable
- S3 is 2PL
- S3 is SS2PL

- S4 is recoverable
- S4 is cascade-less
- S4 is conflict serializable
- S4 is 2PL
- S4 is SS2PL
Part 5  Physical Optimization (Total: 30 Points)

Consider the following relations $R(A, B)$, $S(C, D)$, $T(E, F)$ with $S(R) = S(S) = S(T) = \frac{1}{5}$ (5 tuples fit on each page). The sizes and value distributions are shown below. Recall that $N(R)$ is the number of tuples in $R$, $B(R)$ is the number of pages of relation $R$, and $V(R, A)$ is the number of distinct values or relation $R$ in attribute $A$.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Number of Tuples</th>
<th>Number of Distinct Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>$N(R) = 3000$</td>
<td>$V(R, A) = 3000$</td>
</tr>
<tr>
<td>$S$</td>
<td>$N(S) = 1000$</td>
<td>$V(S, C) = 500$</td>
</tr>
<tr>
<td>$T$</td>
<td>$N(T) = 50$</td>
<td>$V(T, E) = 10$</td>
</tr>
</tbody>
</table>

Question 5.1  Dynamic Programming (30 Points)

Use the dynamic programming join enumeration algorithm to find the cheapest left-deep plan for the join $R \bowtie_{B=C} S \bowtie_{D=E} T$. Assume that block nested-loop is the only available join implementation with the left input being the “outer” (for each chunk from the outer we have to scan the whole inner relation). Furthermore, there are no indices defined on any of the relations (that is you have to use sequential scan for each of the relations). As a cost model consider the total number of I/O operations. For example, if you join two relations with 5,000 and 10,000 tuples with $S = \frac{1}{101}$, where the 5,000 tuple relation is the outer and $M = 101$ memory pages are available, then the cost would be 5,500. Assume that the system does not support pipelining at all. That is you need to consider the cost of writing the result of a join to disk and reading it from disk when executing the next level join. Note that you do not have to consider any I/O to write the final result. Ignore plans containing cross-products. Hint: You will have to estimate the size of intermediate results. Use the estimation based on the number of values and not the one based on the size of the domain. Use the assumption that the number of values in a join attribute of a join result is the minimum of the number of values in the join attribute of each input.

Write down the memoized query plans in optPlan after each iteration of the algorithm using the following notation. Write $\{R_1, R_2, \ldots, R_n\} : (R_1, R_2, \ldots, R_n)^{C,S}$ to denote that the plan shown below with I/O cost $C$ and result size $S$ is the bet plan for the set $\{R_1, \ldots, R_n\}$.

Solution
Initialization:

\[
\{R\} = (R)^{600,3000} \quad \{S\} = (S)^{200,1000} \quad \{T\} = (T)^{10,50}
\]

\(i = 2\):

Here we have 4 different options how to join two of the plans from the initialization (the other options contain cross-products and, thus, are not considered):

\[(R,S)^{2400,3000}, (S,R)^{2000,3000}, (S,T)^{240,100}, (T,S)^{230,100}\]

As an example take the join \((R,S)^{2400,3000}\). Here \(R\) is the outer and \(S\) is the inner. Using the formula from class the estimated result size is

\[
\frac{N(R) \cdot N(S)}{\max(V(R,B),V(S,C))} = \frac{3000 \cdot 1000}{1000} = 3000.
\]

The cost is computed as: For each chunk of size \(M - 1\) from \(R\) \((\frac{B(R)}{M - 1})\) we have to scan \(S\) once (200 I/Os) plus we have to read \(R\) once (600 I/Os). Thus, the cost is \(B(R) + \frac{B(R)}{100} \cdot B(S) = 600 + 6 \cdot 200 = 1800\) I/Os. In addition we have to account for the cost of writing the results to disk: 600 I/Os. Thus, the total cost is 2400 I/Os.

\[
\{R,S\} = (S,R)^{2400,3000} \quad \{S,T\} = (T,S)^{230,100}
\]

\(i = 3\):

Now we need to consider all partitions of \(\{R,S,T\}\) that do not contain cross-products. These are \((R),(S,T)\) and \((R,S),T\). For each partition \(S, O\) we consider all join options between \(optPlan(\{S\})\) with \(optPlan(\{O\})\). Since we only allow for left-deep plans the only valid option is to join the result of the two element subset with the singleton subset.

\[(T,S,R)^{980,300}, (S,R,T)^{3050,300}\]

The final state of \(optPlan\) is

\[
\{R,S,T\} = (T,S,R)^{980,300}
\]