# CSE 562 Midterm 1 Solutions

March 12, 2014

<table>
<thead>
<tr>
<th>Question</th>
<th>Points Possible</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>B.1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>B.2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>40</strong></td>
<td></td>
</tr>
</tbody>
</table>
The Birdwatcher Schema.

CREATE TABLE Birds (  
  bid integer,  
  species string,  
  legband char(4),  
  PRIMARY KEY (bid),  
  UNIQUE (legband)  
);  

CREATE TABLE Observers (  
  oid integer,  
  name string,  
  PRIMARY KEY (oid)  
);  

CREATE TABLE Sightings (  
  oid integer,  
  bid integer,  
  when date,  
  latitude decimal,  
  longitude decimal,  
  PRIMARY KEY (bid, oid, when),  
  FOREIGN KEY (oid) REFERENCES Observers,  
  FOREIGN KEY (bid) REFERENCES Birds  
);  

<table>
<thead>
<tr>
<th>Birds</th>
<th>bid</th>
<th>species</th>
<th>leg band</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Raven</td>
<td>MORB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Raven</td>
<td>MKYB</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Blue Jay</td>
<td>MRRK</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observers</th>
<th>oid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Carol</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sightings</th>
<th>oid</th>
<th>bid</th>
<th>when</th>
<th>lat</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>01/03/14</td>
<td>43.17</td>
<td>-77.96</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>01/03/14</td>
<td>42.59</td>
<td>-78.69</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01/03/14</td>
<td>42.95</td>
<td>-78.65</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>01/01/14</td>
<td>42.68</td>
<td>-78.65</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>01/01/14</td>
<td>43.15</td>
<td>-79.30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>01/02/14</td>
<td>43.88</td>
<td>-78.62</td>
<td></td>
</tr>
</tbody>
</table>

Relational Algebra Operator Reference

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>$\sigma_c(R)$</td>
<td>$c$ : The selection condition</td>
</tr>
<tr>
<td>Extended Project</td>
<td>$\pi_{e_1,e_2,...}(R)$</td>
<td>$e_i$ : The column or expression to project</td>
</tr>
<tr>
<td>Product</td>
<td>$R_1 \times R_2$</td>
<td></td>
</tr>
<tr>
<td>Join</td>
<td>$R_1 \bowtie_c R_2$</td>
<td>$c$ : the join condition</td>
</tr>
<tr>
<td>Distinct</td>
<td>$\delta(R)$</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td>$\gamma_{g_{b_1},g_{b_2},...,agg(e_1),...}(R)$</td>
<td>$g_{b_i} :$ group by columns, $e_i :$ expression</td>
</tr>
<tr>
<td>Set Difference</td>
<td>$R_1 - R_2$</td>
<td></td>
</tr>
<tr>
<td>Union</td>
<td>$R_1 \cup R_2$</td>
<td></td>
</tr>
<tr>
<td>Sort</td>
<td>$\tau_A$</td>
<td>$A$ one or more attributes to sort on</td>
</tr>
</tbody>
</table>
### Relational Algebra Equivalences

<table>
<thead>
<tr>
<th>Rule</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_{C_1 \land C_2}(R) \equiv \sigma_{C_1}(\sigma_{C_2}(R)) )</td>
<td></td>
</tr>
<tr>
<td>( \sigma_{C_1 \lor C_2}(R) \equiv \sigma_{C_1}(R) \cup \sigma_{C_2}(R) )</td>
<td></td>
</tr>
<tr>
<td>( \sigma_C(R \times S) \equiv R \bowtie_C S )</td>
<td>If ( C ) references only ( R )'s attributes, also works for joins</td>
</tr>
<tr>
<td>( \pi_{A}(\pi_{A \cup B}(R)) \equiv \pi_{A}(R) )</td>
<td>If ( A ) contains all of the attributes referenced by ( C )</td>
</tr>
<tr>
<td>( \sigma_C(\pi_{A}(R)) \equiv \pi_A(\sigma_C(R)) )</td>
<td></td>
</tr>
<tr>
<td>( \pi_{A \cup B}(R \times S) \equiv \pi_A(R) \times \pi_B(S) )</td>
<td>Where ( A ) (resp., ( B )) contains attributes in ( R ) (resp., ( S ))</td>
</tr>
<tr>
<td>( R \times (S \times T) \equiv (R \times S) \times T )</td>
<td>Also works for joins</td>
</tr>
<tr>
<td>( R \times S \equiv S \times R )</td>
<td>Also works for joins</td>
</tr>
<tr>
<td>( R \cup (S \cup T) \equiv (R \cup S) \cup T )</td>
<td>Also works for intersection and bag-union</td>
</tr>
<tr>
<td>( R \cup S \equiv S \cup R )</td>
<td>Also works for intersections and bag-union</td>
</tr>
<tr>
<td>( \sigma_C(R \cup S) \equiv \sigma_C(R) \cup \sigma_C(S) )</td>
<td>Also works for intersections and bag-union</td>
</tr>
<tr>
<td>( \pi_{A}(R \cup S) \equiv \pi_{A}(R) \cup \pi_{A}(S) )</td>
<td>Also works for intersections and bag-union</td>
</tr>
<tr>
<td>( \sigma_C(\gamma_{A,\text{AGG}}(R)) \equiv \gamma_{A,\text{AGG}}(\sigma_C(R)) )</td>
<td>If ( A ) contains all of the attributes referenced by ( C )</td>
</tr>
</tbody>
</table>

---

3 of ??
Question A: SQL
(8 points)

You are hired by a local birdwatching organization, who’s database uses the Birdwatcher Schema on page ???. You are asked to design a leader board for each species of Bird. The leader board ranks Observers by the number of Sightings for Birds of the given species. Write a query that computes the set of names of all Observers who are highest ranked on at least one leader board. Assume that there are no tied rankings.

The most conceptually simple approach follows a pattern similar to Question B.1.

```sql
SELECT DISTINCT o.name
FROM ( SELECT s.oid, b.species, COUNT(*) AS cnt
    FROM Sightings s, Birds b WHERE s.bid = b.bid
    GROUP BY s.oid, b.species
    ) per_o,
    (SELECT species, MAX(cnt) AS max_cnt
    FROM ( SELECT s.oid, b.species, COUNT(*) AS cnt
    FROM Sightings s, Birds b WHERE s.bid = b.bid
    GROUP BY s.oid, b.species
    ) obs_cnt
    GROUP BY species
    ) best_o,
    Observers o
WHERE per_o.oid = best_o.oid AND o.oid = per_o.oid
AND per_o.cnt >= best_o.max_cnt;
```

Several students noted equivalent solutions, most notably using IN or inline computation:

```sql
SELECT DISTINCT o.name
FROM ( SELECT s.oid, b.species
    FROM Sightings s, Birds b WHERE s.bid = b.bid
    GROUP BY s.oid, b.species
    HAVING COUNT(*) = (SELECT COUNT(*) FROM Sightings s2, Birds b2
    WHERE s2.bid = b2.bid AND b2.species = b.species)
    ) best_o,
    Observers o
WHERE o.oid = best_o.oid;
```

A common mistake on the IN formulation was to not properly correlate the nested query, leading to a query result based on the most sightings for one species. The correct query formulation is:

```sql
SELECT DISTINCT o.name
FROM Observers o, (SELECT DISTINCT species FROM bird) species
WHERE o.oid IN (SELECT s.oid FROM Sightings s, Birds b
WHERE s.bid = b.bid and b.species = species.species
GROUP BY s.oid, b.species
ORDER BY COUNT(*) DESC LIMIT 1);
```
Question B.1: Relational Algebra

(10 points)

Using the relational algebra operators shown on page ??, write a bag relational algebra expression for the following query:

```sql
SELECT o.name, seen.species, (observed / num_birds) AS pct_seen
FROM Observers o,
( SELECT oid, species, COUNT(*) as observed FROM ( 
    SELECT DISTINCT s.oid, b.species, s.when 
    FROM Sightings s, Birds b 
  ) sightings_by_species 
) seen,
( SELECT species, COUNT(*) as num_birds FROM Birds 
) cnt
WHERE seen.oid = cnt.oid AND seen.species = cnt.species AND seen.oid = o.oid;
ORDER BY o.name, seen.species, pct_seen DESC
```

```sql
τ o.name, seen.species, pct seen DESC(
  π name←O.name, species←seen.species, pct seen←(seen.observed/seen.num_birds)(
    σ seen.oid=cnt.oid∧seen.species=cnt.species∧seen.oid=o.oid(
      π oid, species, num_birds←oid, species, COUNT(*) S×B
      γ oid, species, num_birds←COUNT(*) (S×B)
      π cnt.species←species, cnt.observed←observed(
      γ species, observed←COUNT(*) (B)
    )
  )
) )
```
UBIT:

Question B.2: Relational Algebra  
(8 points)

Recall that the Full Outer Join operator (×) guarantees that every tuple in either source relation will be represented in the output. Any tuple on the left that is not matched to any tuples on the right will be part of the result set, but with NULL values for all of the attributes derived from the right-hand side. Similarly, every tuple on the right not matched to a tuple on the left will be part of the output with NULL values for the left-hand-side attributes.

Implement Full Outer Join as a bag relational algebra expression using only the operators listed on page ??.

Let $r$ represent the attributes of $R$ not in $S$, $s$ represent the attributes of $S$ not in $R$, and $b$ represent the attributes in both. (i.e., $\text{schema}(R) = r \cup b$, $\text{schema}(S) = s \cup b$, and $s \cap r = \emptyset$).

\[
\begin{align*}
&. (R \bowtie S) \\
&. \cup \quad \pi_{r \leftarrow r, b \leftarrow b, s \leftarrow \text{NULL}} \left( (\pi_b(R) - \pi_b(S)) \bowtie R \right) \\
&. \cup \quad \pi_{r \leftarrow \text{NULL}, b \leftarrow b, s \leftarrow s} \left( (\pi_b(S) - \pi_b(R)) \bowtie S \right)
\end{align*}
\]

or equivalently

\[
\begin{align*}
&. (R \bowtie S) \\
&. \cup \left( (R - \pi_r(R \bowtie S)) \times \text{NULL} \right) \\
&. \cup \left( \text{NULL} \times (S - \pi_s(R \bowtie S)) \right)
\end{align*}
\]
Consider the following *bag*-relational algebra query:

\[
\pi_{R.A,T,E}(\sigma_{R.B < S.B}(R \times (S \bowtie_{S.C = T.C} (T_1 \cup \sigma_{T.D = 3} T_2))))
\]

For each operator appearing in the query, write down the working set size (i.e., memory complexity) of the operator using Big-O notation.

<table>
<thead>
<tr>
<th>#</th>
<th>Operator</th>
<th>Working Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\sigma_{T.D = 3}(T_2)) (no index)</td>
<td>(O(1))</td>
</tr>
<tr>
<td>2</td>
<td>(\sigma_{T.D = 3}(T_2)) (B+Tree index on (T_2.D))</td>
<td>(O(1))</td>
</tr>
<tr>
<td>3</td>
<td>(\bowtie_{S.C = T.C}) (as Nested Loop Join)</td>
<td>(O(1))</td>
</tr>
<tr>
<td>4</td>
<td>(\bowtie_{S.C = T.C}) (as Block Nested Loop Join)</td>
<td>(O(</td>
</tr>
<tr>
<td>5</td>
<td>(\bowtie_{S.C = T.C}) (as Hybrid Hash Join)</td>
<td>(O(</td>
</tr>
<tr>
<td>6</td>
<td>(\bowtie_{S.C = T.C}) (as Index Nested Loop Join)</td>
<td>(O(1) + \text{Index})</td>
</tr>
<tr>
<td>7</td>
<td>(\times)</td>
<td>(O(1))</td>
</tr>
<tr>
<td>8</td>
<td>(\sigma_{R.B &lt; S.B})</td>
<td>(O(1))</td>
</tr>
</tbody>
</table>
Consider the following two queries over a database with the schema: \( R(A, B), S(B, C, D), T(D, E) \)

\[
\begin{align*}
\text{(a)} & \quad \pi_{R.A} \sigma_{(T.E < 20) \land (\text{COUNT} < 100)} \gamma_{R.A, T.E, \text{COUNT}(\cdot)} \\
\text{(b)} & \quad \pi_{R.A} \sigma_{(T.E < 20)} \gamma_{R.A, T.E, \text{COUNT}(\cdot)} \sigma_{S.C = 2} \bowtie B \bowtie D \end{align*}
\]

Prove using only the list of common primitive relational algebra equivalencies found on page 23 of the exam that the above queries are equivalent, or provide a counterexample in the form of input relations \( R, S, \) and \( T \) that demonstrates that they are not.

The two expressions are not equivalent. Given the following data, query (a) produces an output consisting of rows 1, 2, 3, while query (b) produces 2 copies of row 1.

\[
\begin{array}{ccc}
R & A & B \\
1 & 1 & 1 \\
1 & 2 & 2 \\
2 & 1 & 1 \\
3 & 2 & 2 \\
\end{array}
\quad
\begin{array}{ccc}
S & B & C \\
1 & 1 & 1 \\
2 & 1 & 2 \\
\end{array}
\quad
\begin{array}{ccc}
T & D & E \\
1 & 1 & 1 \\
\end{array}
\]

It is almost possible to show that these two expressions are equivalent. Most of the transformations are simple, and eventually, the problem reduces to proving the following equivalence (modulo the schema, which gets projected down to \( R.A \) later on anyway):

\[
\gamma_{R.A, T.E, \text{COUNT}(R \bowtie \cdots)} \equiv R \bowtie (\gamma_{S.B, T.E, \text{COUNT}(\cdot)})
\]

Normally, we could dismiss such a crazy equivalence outright. However, there are some conditions under which the equivalence does hold. If \( R.A \) is a key, then every row of \( R \) produces exactly one group. As a result we can swap the order of the join and grouping operator. To disprove equivalence then, we need to create inputs that produce a result, and where \( R.A \) is not a key. The example above also illustrates the potential benefit of applying this equivalence (when it is correct to do so). If you remove the first row from \( R \) (making \( R.A \) a key), the plans are equivalent and plan (a) creates 3 groups, while plan (b) creates only 2.
Grading Details

Question A (Grader: Dr. Kennedy)

Partial credit was based on proximity to any of these solutions. 4 points were awarded for a solution that included a roughly approximate formulation of the leaderboard query, 2 additional points were awarded for a solution that included an ORDER BY/LIMIT or per.o.cnt > best.o.cnt formulation. 1 point was deducted for substantial SQL errors (e.g., max(count(*))). Up to 1 additional point was awarded at the grader’s discretion based on apparent understanding of the query’s goals.

Question B.1 (Grader: Ning Deng)

There are four SELECT clauses in the question (including their corresponding "WHERE" clauses) and an ORDER BY clause. For every accurately expressed clause, 2 points are given. By "accurately expressed", it means expressing BOTH the OPERATORS and CONDITIONS correctly. If you miss parts of the conditions, 1 point is deducted. If you use wrong operators in the clause, 1 point is deducted.

If you write the answer in a confusing way so that we have to work hard to find or figure out your true answer, 1 or 2 points is deducted.

A common mistake in this question is that students missed the "DESC" condition, or they just didn’t write the ORDER BY clause.

Question B.2 (Grader: Vishrawas Gopalakrishnan)

2 points were awarded for writing $R \bowtie S$. 4 points were awarded for correct usage of difference operator and incorporating NULL. 2 points were awarded for putting all the things together or if the basic structure was right, viz., union of 3 components - $R \bowtie S$, $(R - \pi_r(R \bowtie S))$, and $(S - \pi_s(R \bowtie S))$.

Question C (Grader: Vishrawas Gopalakrishnan)

0.5 pts for each question. No partial grading. So long as the student conveyed basic understanding of the question and working set size with respect to the operator, no points were deducted. Missing constants and incorrect usage of notations were condoned. However, important missing or incorrect presence or absence of terms like missing Index in # 6 or having both $S$ and $T$ in memory for Hybrid Hash Join were penalized.

Question D (Grader: Vishrawas Gopalakrishnan)

10 points were deducted if the student answers the two relational algebra to be equivalent. 8 points were deducted if no reason was provided. 5 points were deducted if the justification was close enough but didn’t show complete understanding of the reason, viz., no counter example was provided nor any justification under the premise that the two tree are equivalent if and only if $A$ is a primary key.