CSE 562 Midterm 1 *Solutions*

March 12, 2014

Question	Points Possible	Points Earned
A	8	
B.1	10	
B.2	8	
C	4	
D	10	
Total	40	

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
);
```

Birds	bid	species	leg band
	1	Raven	MORB
	2	Raven	MKYB
	3	Blue Jay	MRRK

Observers	oid	name
	1	Alice
	2	Bob
	3	Carol

Sightings	oid	bid	when	lat	long
	3	2	01/03/14	43.17	-77.96
	2	1	01/03/14	42.59	-78.69
	1	1	01/03/14	42.95	-78.66
	1	3	01/01/14	42.68	-78.65
	1	1	01/01/14	43.15	-79.30
	3	3	01/02/14	43.88	-78.62

Relational Algebra Operator Reference

Select	$\sigma_c(R)$	c : The selection condition
Extended Project	$\pi_{e_1,e_2,\ldots}(R)$	e_i : The column or expression to project
Product	$R_1 \times R_2$	
Join	$R_1 \bowtie_c R_2$	c : the join condition
Distinct	$\delta(R)$	
Group	$\gamma_{gb_1,gb_2,\ldots,\texttt{AGG}(e_1),\ldots}(R)$	gb_i : group by columns, e_i : expression
Set Difference	$R_1 - R_2$	
Union	$R_1 \cup R_2$	
Sort	τ_A	A one or more attributes to sort on

Relational Algebra Equivalences

Rule	Notes
$\sigma_{C_1 \wedge C_2}(R) \equiv \sigma_{C_1}(\sigma_{C_2}(R))$	
$\sigma_{C_1 \vee C_2}(R) \equiv \sigma_{C_1}(R) \cup \sigma_{C_2}(R)$	Note, this is set, not bag union
$\sigma_C(R \times S) \equiv R \bowtie_C S$	
$\sigma_C(R \times S) \equiv \sigma_C(R) \times S$	If C references only R's attributes, also works for joins
$\pi_A(\pi_{A \cup B}(R)) \equiv \pi_A(R)$	
$\sigma_C(\pi_A(R)) \equiv \pi_A(\sigma_C(R))$	If A contains all of the attributes referenced by C
$\pi_{A \cup B}(R \times S) \equiv \pi_A(R) \times \pi_B(S)$	Where A (resp., B) contains attributes in R (resp., S)
$R \times (S \times T) \equiv (R \times S) \times T$	Also works for joins
$R \times S \equiv S \times R$	Also works for joins
$R \cup (S \cup T) \equiv (R \cup S) \cup T$	Also works for intersection and bag-union
$R \cup S \equiv S \cup R$	Also works for intersections and bag-union
$\sigma_C(R \cup S) \equiv \sigma_C(R) \cup \sigma_C(S)$	Also works for intersections and bag-union
$\pi_A(R \cup S) \equiv \pi_A(R) \cup \pi_A(S)$	Also works for intersections and bag-union
$\sigma_C(\gamma_{A,AGG}(R)) \equiv \gamma_{A,AGG}(\sigma_C(R))$	If A contains all of the attributes referenced by C

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.

```
. 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:

```
. 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:

```
. 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 an page **??**, write a **bag relational algebra** expression for the following query:

```
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
```

- $\tau_{o.name,\ seen.species,\ pct\_seen\ \ DESC}\Big($
- $\quad \pi_{name \leftarrow O.name,\ species \leftarrow seen.species,\ pct\_seen \leftarrow (seen.observed / seen.num\_birds)}\Big($
- $\quad\quad \sigma_{seen.oid = cnt.oid \wedge seen.species = cnt.species \wedge seen.oid = o.oid}\Big($
- $\quad\quad\quad O \times$
- $\quad\quad\quad \pi_{seen.oid \leftarrow oid,\ seen.species \leftarrow oid,\ seen.num\_birds \leftarrow num\_birds}\Big($
- $\quad\quad\quad\quad \gamma_{oid,\ species,\ num\_birds \leftarrow COUNT(*)}\Big(\ \gamma_{oid,\ species,\ when}\Big(S \times B\Big)\ \ \Big)$
- $\quad\quad\quad \Big)\times$
- $\quad\quad\quad \pi_{cnt.species \leftarrow species,\ cnt.observed \leftarrow observed}\Big($
- $\quad\quad\quad\quad \gamma_{species,\ observed \leftarrow COUNT(*)}\Big(B\Big)$
- $\quad\quad\quad \Big)$
- $\quad\quad \Big)$
- $\quad \Big)$
- $\Big)$

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., $schema(R) = r \cup b$, $schema(S) = s \cup b$, and $s \cap r = \emptyset$).

. $(R \bowtie S)$
. $\quad \cup\ \pi_{r \leftarrow r, b \leftarrow b, s \leftarrow \texttt{NULL}}\left(\ (\pi_b(R) - \pi_b(S)) \bowtie R\ \right)$
. $\quad \cup\ \pi_{r \leftarrow \texttt{NULL}, b \leftarrow b, s \leftarrow s}\left(\ (\pi_b(S) - \pi_b(R)) \bowtie S\ \right)$

or equivalently

. $(R \bowtie S)$
. $\quad \cup\ \left(\ (R - \pi_r(R \bowtie S)) \times \texttt{NULL}\ \right)$
. $\quad \cup\ \left(\ \texttt{NULL} \times (S - \pi_s(R \bowtie S))\ \right)$

Question C: Query Evaluation
(4 points)

Consider the following *bag*-relational algebra query:

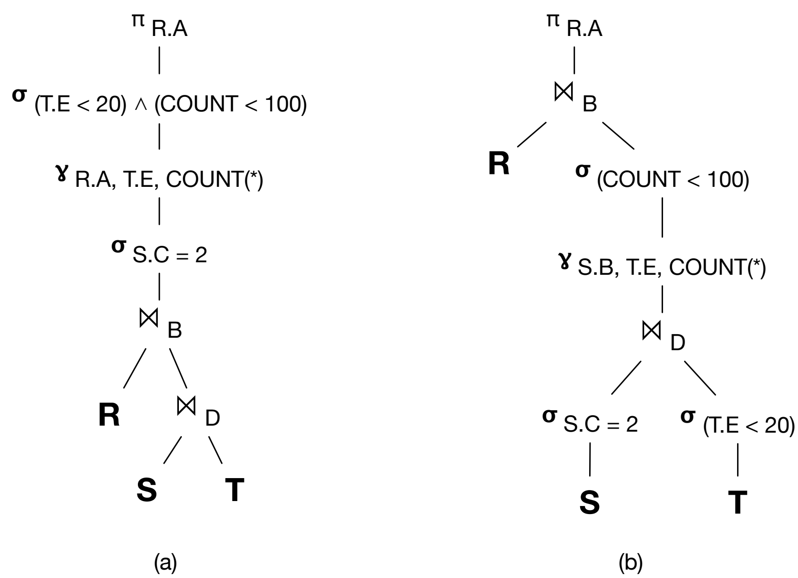
$$\pi_{R.A,T.E}(\sigma_{R.B<S.B}(R \times (S \bowtie_{S.C=T.C} (T_1 \cup \sigma_{T.D=3}T_2))))$$

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

#	**Operator**	**Working Set Size**
1	$\sigma_{T.D=3}(T_2)$ (no index)	$O(1)$
2	$\sigma_{T.D=3}(T_2)$ (B+Tree index on $T_2.D$)	$O(1)$
3	$\bowtie_{S.C=T.C}$ (as Nested Loop Join)	$O(1)$
4	$\bowtie_{S.C=T.C}$ (as Block Nested Loop Join)	$O(\|B\|)$
5	$\bowtie_{S.C=T.C}$ (as Hybrid Hash Join)	$O(\|S\|)$ (or $O(\|T_1 \cup T_2\|)$)
6	$\bowtie_{S.C=T.C}$ (as Index Nested Loop Join)	$O(1)$ + Index
7	$\times$	$O(1)$
8	$\sigma_{R.B<S.B}$	$O(1)$

Question D: Query Rewriting
(10 points)

Consider the following two queries over a database with the schema: $R(A, B)$, $S(B, C, D)$, $T(D, E)$



Prove using **only** the list of common primitive relational algebra equivalencies found on page **??** 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.

R	A	B
	1	1
	1	2
	2	1
	3	2

S	B	C	D
	1	2	1
	2	2	1

T	D	E
	1	1

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):

$$\gamma_{R.A,\,T.E,\,COUNT}(R \bowtie_B \ldots) \equiv R \bowtie_B (\gamma_{S.B,\,T.E,\,COUNT}(\ldots))$$

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.