CS145 Final Examination

Monday, December 11, 2000, 8:30–11:30 AM

Directions

The exam is *open book/notes*; any written materials may be used.

For each of the 33 questions, circle the letter (a), (b), (c), or (d) of your chosen answer. Do not circle more than one answer. If you wish to change your answer, please indicate clearly what your "final answer" is.

Score = 3 times number right minus the number wrong, so random guessing nets you nothing on the average. One point is awarded for putting your name on the paper, so the maximum score is 100.

If you wish to explain or demonstrate your solution to a problem for partial credit, you may use page bottoms or the backs of the pages (but warn us on the front). Please use this option sparingly, e.g., if you think the question is flawed or open to multiple interpretations, because we shall only be awarding partial credit in rare situations.

You have approximately 5.4 minutes per question. Use your time wisely, and do not spend too much time on any one question.

Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here (**1 pt.**): ____

In each of the first 12 questions, you are asked to compare two queries Q_1 and Q_2. You must tell whether the queries are:

1. The same [choice (a)], meaning that for every database the answers to the two queries are the same. That is, the same tuples are produced by each query, and a tuple is produced the same number of times by each query. The order in which tuples are produced is not to be considered.
2. Completely different [choice (d)], meaning that there are databases where Q_1 produces more of some particular tuple, and other databases where Q_2 produces more of some particular tuple. Note that the query producing the smaller number of copies of a tuple may produce zero copies of that tuple.
3. One is contained in the other but they are not the same [choice (b) or (c)]. For instance, Q_1 is contained in Q_2 if on every database, Q_2 produces at least as many copies of each tuple as Q_1 does. Note that it is possible Q_2 produces one or more copies of a tuple, while Q_1 produces none of that tuple.

General advice:

- Do not assume a query has a trivial syntactic error and therefore produces nothing.
- Relations mentioned in the queries may have attributes not mentioned, but their existence should not affect the answer.
- Relations may have `NULL`'s.
- Queries should be assumed to be in SQL2 unless stated otherwise.
- The result of SQL or OQL queries is generally a bag, but the result of a query in relational algebra or Datalog is a set.

Question 1:

Q_1:
```
SELECT a
FROM R
WHERE R.b > ALL(SELECT c FROM S);
```

Q_2:
```
SELECT a
FROM R
WHERE R.b > ANY(SELECT c FROM S);
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 2:

Q_1:
```
SELECT r1.a, r3.a
FROM R r1, R r2, R r3
WHERE r1.b = r2.b AND r2.a = r3.a AND r2.b <> r3.b;
```

Q_2:
```
SELECT r1.a, r3.a
FROM R r1, R r2, R r3
WHERE r1.a = r2.a AND r2.b = r3.b AND r1.b <> r2.b;
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 3: In the following Datalog queries, the result of each is the value of the predicate Ans.

Q_1:
```
Ans(x,y) <- R(x,y) AND NOT S(x) AND NOT T(y)
```

Q_2:
```
Temp(u,v) <- S(u) AND T(v)
Ans(x,y) <- R(x,y) AND NOT Temp(x,y)
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 4: In the following, assume R and S are each relations with attributes a and b only.

Q_1: $\pi_a(R) \cap \pi_a(S)$

Q_2: $\pi_a(R \cap S)$

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

The following three questions are based on this ODL schema:

```
interface A (extent Aext) {
    attribute string aName;
    relationship Set<B> myBs inverse B::myA;
}

interface B (extent Bext) {
    attribute string bName;
    relationship A myA inverse A::myBs;
    relationship C myC inverse C::myBs;
}

interface C (extent Cext) {
    attribute string cName;
    relationship Set<B> myBs inverse B::myC;
}
```

Question 5:

Q_1:

```
SELECT bb.myC.cName
FROM Aext aa, aa.myBs bb
WHERE aa.aName = "Sue"
```

Q_2:

```
SELECT bb.myC.cName
FROM Bext bb
WHERE bb.myA.aName = "Sue"
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 6:

Q_1:

```
SELECT cc.cName
FROM Cext cc, cc.myBs bb
WHERE bb.myA.aName = "Sue"
```

Q_2:

```
SELECT cc.cName
FROM Cext cc
WHERE EXISTS bb IN Bext:
    bb.myA.aName = "Sue"
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 7:

Q_1:

```
SELECT x: COUNT(aa.myBs)
FROM Aext aa
```

Q_2:

```
SELECT x: COUNT(partition)
FROM Bext bb
GROUP BY bb.myA
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 8: In the following Datalog queries, the result of each is the value of the predicate *Path*.

Q_1:
```
Path(x,y) <- Arc(x,y)
Path(x,y) <- Path(z,y) AND Arc(x,z)
```

Q_2:
```
Path(x,y) <- Arc(x,y)
Path(x,y) <- Arc(z,y) AND Path(x,z)
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 9: In the following, R is a relation with schema $R(a, b)$. The result of each sequence of modifications is the value of R at the end.

Q_1:
```
UPDATE R SET b = 3 WHERE b = 2;
```

Q_2:
```
INSERT INTO R
    SELECT a, 3 FROM R WHERE b = 2;
DELETE FROM R WHERE b = 2;
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 10:

Q_1: `SELECT * FROM R WHERE a LIKE '%NULL%';`

Q_2: `SELECT * FROM R WHERE a LIKE '%NULL' OR a LIKE 'NULL%';`

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 11: In this question, $R(x)$ is the schema of relation R.

Q_1:
```
SELECT x
FROM R rr
WHERE NOT EXISTS(
    SELECT * FROM R WHERE x > rr.x
);
```

Q_2:
```
SELECT MAX(x) FROM R;
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

Question 12: In the following queries, you may assume $R(a, b)$ has no `NULL`'s but may have duplicates.

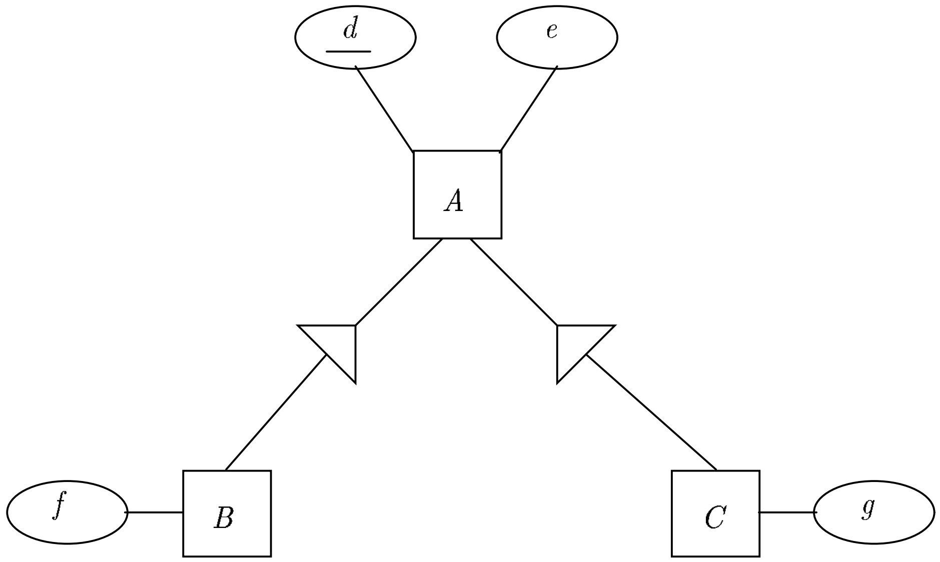
Q_1:
```
SELECT DISTINCT COUNT(*)
FROM R
GROUP BY a;
```

Q_2:
```
SELECT DISTINCT COUNT(b)
FROM R
GROUP BY a;
```

(a) Q_1 and Q_2 produce the same answer.
(b) The answer to Q_1 is always contained in the answer to Q_2.
(c) The answer to Q_2 is always contained in the answer to Q_1.
(d) Q_1 and Q_2 produce different answers.

The remainder of the questions have no particular form of the expected answer.

The next two questions are based on the following E/R diagram:



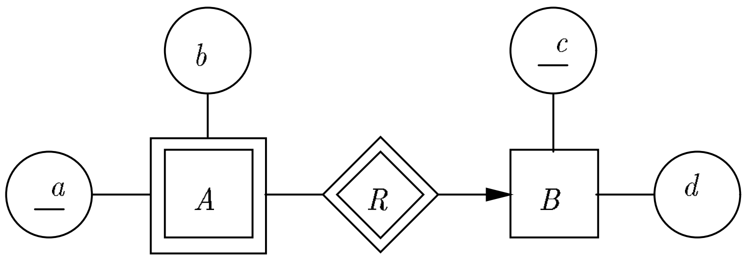
Question 13: If we use the "E/R" approach of converting this diagram to relations, which of the following would be one of the relation schemas we construct?

(a) $B(d, e, f)$ (b) $C(d, e)$ (c) $A(d, f, g)$ (d) $B(d, f)$

Question 14: If we use the "object-oriented" approach to converting this diagram to relations, which set of attributes would *not* be the schema of some relation?

(a) $\{d, e, f\}$ (b) $\{d, f, g\}$ (c) $\{d, e, f, g\}$ (d) $\{d, e\}$

Question 15: If we convert the following E/R diagram to relations using the standard method described in the book and class, we get which of the following database schemas?



(a) $A(a, b)$, $R(a, c)$, and $B(c, d)$
(b) $A(a, b, c)$ and $B(c, d)$
(c) $A(a, b)$ and $B(a, b, d)$
(d) $A(a, b, c)$, $R(a, c)$, and $B(c, d)$

Question 16: Suppose transaction T_1 executes the following modifications to relation $R(a, b)$:

```
INSERT INTO R VALUES (0,1);
DELETE FROM R WHERE a = 2 and b = 3;
```

T_1 runs with isolation level `SERIALIZABLE`, although it doesn't matter. Transaction T_2 runs with isolation level `REPEATABLE READ`, and performs the following two queries.

```
SELECT * FROM R WHERE a >= 0;
SELECT * FROM R WHERE b >= 0;
```

We do not know in which order T_1 and T_2 run, and it is possible that they run at the same time. Which of the choices below is *not* a possible sequence of responses to the two queries?

(a) First $\{(2,3)\}$; second $\{(0,1),\ (2,3)\}$.
(b) First $\emptyset$ (i.e., no tuples); second $\{(0,1)\}$.
(c) First $\{(2,3)\}$; second $\{(2,3)\}$.
(d) First $\{(2,3)\}$; second $\{(0,1)\}$.

Question 17: A is the owner of privilege p. The following sequence of grants and revocations occur:

```
A: GRANT p to B WITH GRANT OPTION
B: GRANT p to C WITH GRANT OPTION
C: GRANT p to B
A: REVOKE p FROM B CASCADE
```

At this time, which statement best describes the situation regarding p?

(a) Only A has the privilege p, and has it with grant option.
(b) A has p with grant option, and B has p without grant option; C does not have p.
(c) A, B and C have p, but only A has it with grant option.
(d) A has p with grant option, C does not have p, and B has the grant option for p but not p itself.

The following two questions concern the relations:

```
Emps(id, name, dept, salary)
Managers(dept, mgr)
```

The first gives the employee ID, their name, department, and salary; the second gives for each department, the manager of that department, which is the employee ID of the person managing the department.

Question 18: We wish to constrain the relations so that in the `mgr` attribute of a `Managers` tuple there must appear the ID of an employee in `Emps`. Which of the following changes, by itself, enforces that constraint?

(a) In the declaration of `Managers`, add for attribute `mgr` the attribute-based check `CHECK(EXISTS(SELECT * FROM Emps WHERE id = mgr))`.

(b) In the declaration of `Emps`, add the constraint `FOREIGN KEY id REFERENCES Managers(mgr)`.

(c) In the declaration of `Managers`, add the constraint `FOREIGN KEY mgr REFERENCES Emps(id)`.

(d) More than one of the above.

Question 19: Suppose we wish to constrain the data so that in no department can the employees have a total salary greater than $1,000,000. The following is a framework for an assertion that will enforce this constraint:

```
CREATE ASSERTION cheap CHECK(
    NOT EXISTS(Q));
```

Which query Q best enforces this constraint?

(a)

```
SELECT * FROM Emps WHERE SUM(salary) > 1000000
```

(b)

```
SELECT dept, SUM(salary) FROM Emps
GROUP BY dept
```

(c)

```
SELECT SUM(salary)
FROM Emps, Managers
WHERE id = mgr
GROUP BY Emps.dept
HAVING SUM(salary) > 1000000
```

(d)

```
SELECT dept FROM Emps
GROUP BY dept HAVING SUM(salary) > 1000000
```

Question 20: In the 3-valued logic of SQL2, the value of `x = y` when `x` has the value `NULL` is:

(a) `UNKNOWN`
(b) `FALSE`
(c) `UNKNOWN` unless `y` also has the value `NULL`, in which case it is `TRUE`
(d) `FALSE` unless `y` also has the value `NULL`, in which case it is `UNKNOWN`

The following two questions concern the PL/SQL program below. It refers to a relation `Teams(name, score)` and a relation `NewTeams`, whose attributes we do not tell you, but which you may assume is initially empty.

```
DECLARE
    i Teams.score%TYPE;
    n Teams.name%TYPE;
BEGIN
    i := 1;
    LOOP
        SELECT name INTO n FROM Teams WHERE score = i;
        INSERT INTO NewTeams(x,y) VALUES(i, n);
        EXIT WHEN i > 10;
        i := i + 1;
    END LOOP;
END;
.
run
```

Question 21: Which of the following best describes the effect on `NewTeams`, assuming that the PL/SQL statement above successfully executes?

(a) There will be at most 10 tuples in `NewTeams`.
(b) `NewTeams` will contain every tuple of `Teams`, with the order of components reversed.
(c) `NewTeams` will contain exactly 11 tuples.
(d) There is no long-term effect on the value of `NewTeams`.

Question 22: Which of the following conditions is *not* necessary for a successful execution of the PL/SQL statement?

(a) `NewTeams` has attributes `x` and `y`.
(b) `Teams.score` is of some type that can be interpreted as an integer.
(c) `score` is a key for `Teams`.
(d) None of the above; i.e., each is essential for the correct execution of the PL/SQL statement.

The following two questions involve the three relations below:

1. $R(a, b) = \{(0, 1),\ (4, 5),\ (8, 9)\}$.
2. $S(b, c) = \{(1, 2),\ (5, 2),\ (5, 6),\ (5, 10), (13, 10)\}$.
3. $T(c, d) = \{(2, 3),\ (6, 7),\ (10, 11),\ (10, 3)\}$.

Question 23: The number of tuples in $R \bowtie S \bowtie T$, where $\bowtie$ is the natural join is:
(a) 5 (b) 8 (c) 10 (d) 13

Question 24: The number of tuples in $(R \overset{\circ}{\bowtie} S) \overset{\circ}{\bowtie} T$, where $\overset{\circ}{\bowtie}$ is the full, natural outer join, is:
(a) 5 (b) 8 (c) 13 (d) 60

Question 25: The following declarations are in Oracle SQL (not SQL2):

```
CREATE TYPE PersonType AS OBJECT(
    name VARCHAR2(50),
    age INT
);
/

CREATE TABLE People OF PersonType;

CREATE TABLE PC(
    child REF PersonType,
    parent PersonType
);
```

Which of the queries below correctly finds the names of children that are older than their parents?

(a) `SELECT REF(child).name FROM PC WHERE REF(child).age > parent.age;`
(b) `SELECT pp.REF(child).name FROM PC pp WHERE pp.REF(child).age > pp.parent.age;`
(c) `SELECT THE(pp.child).name FROM PC pp WHERE THE(pp.child).age > pp.parent.age;`
(d) `SELECT pp.child.name FROM PC pp WHERE pp.child.age > pp.parent.age;`

The next two questions are based on a relation $R(A, B, C, D, E)$ with the following functional dependencies: $AB \rightarrow C$, $BC \rightarrow D$, $CD \rightarrow E$, $DE \rightarrow A$, and $AE \rightarrow B$.

Question 26: The number of superkeys (including keys) of the relation R is
(a) 16 (b) 20 (c) 21 (d) 26

Question 27: If we project R onto the 10 subsets consisting of three of the five attributes, how many of the projected relations are in Boyce-Codd Normal Form?
(a) none (b) all (c) 4 (d) 5

Question 28: Which of the following statements is true if R and S are sets, but not necessarily true if R and S are bags?

(a) $(R - S) \cup (S - R) = R \cap S$
(b) $R \cup (S \cup R) = (R \cup S) \cup R$
(c) $R \cap S = R - (R - S)$
(d) $R \cup (S \cap R) = (R \cup S) \cap R$

The following two questions are based on the relation $R(A, B, C, D)$ below:

A	B	C	D
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Notice that R contains all tuples of 0's and 1's with an even number of 1's.

Question 29: This relation R does not satisfy the multivalued dependency $A \twoheadrightarrow B$. What is the smallest number of tuples we would have to delete from R (without inserting or updating any tuples) to be left with a relation instance that satisfied $A \twoheadrightarrow B$?
(a) 1 (b) 4 (c) 7 (d) 8

Question 30: There are 28 possible nontrivial functional dependencies with singleton right side on attributes A, B, C, and D (to see why, note that there are 12 with singleton left side, 12 more with two attributes on the left, and 4 with three attributes on the left). Of these 28, how many are satisfied by the relation R above?
(a) 0 (b) 4 (c) 12 (d) 28

Question 31: There is a relation `Emps(name, salary)`, and we want to make sure that if an employee is given a raise (i.e., an existing employee's salary is increased), and the old salary was less than \$100,000, then the new salary is also less than \$100,000. Violations of this rule are rejected; i.e., no change to the salary may occur. Here is the outline of an Oracle trigger to enforce this constraint:

```
CREATE TRIGGER SalaryCap
    AFTER UPDATE OF salary ON Emps
    FOR EACH ROW
    WHEN (xxx < 100000)
    BEGIN
        IF yyy >= 100000 THEN
            RAISE_APPLICATION_ERROR(-20000, 'new salary too high');
        ENDIF;
    END;
.
run
```

Two pieces, indicated by **xxx** and **yyy** are omitted. Pick the best combination of choices for thse two missing pieces from the options below.

	xxx	**yyy**
(a)	`:OLD.salary`	`NEW.salary`
(b)	`:OLD.salary`	`:NEW.salary`
(c)	`OLD.salary`	`NEW.salary`
(d)	`OLD.salary`	`:NEW.salary`

Question 32: Suppose unary EDB predicates A and B have the values $A = \{1, 2\}$ and $B = \{2, 3\}$ (i.e., $A(1)$, $A(2)$, $B(2)$, and $B(3)$ are true, but all other A- and B-facts are false). Also let us be given the Datalog program:

```
P(x) <- A(x)
Q(x) <- B(x) AND NOT P(x)
R(x) <- B(x) AND NOT Q(x)
```

Which of the following is a model (i.e, makes each of the rules true no matter what value we choose for x), but is *not* the stratified model?

(a) $P = \{1, 2\}$; $Q = \{2\}$; $R = \{3\}$
(b) $P = \{1\}$; $Q = \{2, 3\}$; $R = \{2\}$
(c) $P = \{1, 2\}$; $Q = \{3\}$; $R = \{2\}$
(d) $P = \{1, 2\}$; $Q = \{2, 3\}$; $R = \emptyset$

Question 33: Suppose we have an SQL2 relation declared by

```
CREATE TABLE Foo(
    name VARCHAR(50) PRIMARY KEY,
    salary INT CHECK(salary <=
        (SELECT AVG(salary) FROM Foo))
);
```

Initially, the Contents of `Foo` is:

name	salary
'Sally'	10000
'Joe'	20000
'Sue'	30000

We now execute the following sequence of modifications:

```
INSERT INTO Foo VALUES('Fred', 12000);
UPDATE Foo SET salary = 20000 WHERE name = 'Sue';
INSERT INTO Foo VALUES('Sally', 13000);
DELETE FROM Foo WHERE name = 'Joe';
```

At the end of these statements, the sum of the salaries over all the tuples then in `Foo` is:
(a) 52,000 (b) 62,000 (c) 65,000 (d) 72,000