CS145 Final Examination

- Please read all instructions (including these) carefully.
- The exam is open book and open notes; any written materials may be used.
- There are 32 questions on the exam, including two extra-credit questions. The exam is divided into two parts: Part I contains 24 multiple-choice questions with 2 points each; Part II contains 8 short-answer questions with a varying number of points for a total of 42 points. You should look through the entire exam before getting started, in order to plan your strategy. You have 180 minutes to complete the exam.
- There is no penalty for guessing multiple-choice questions. For short-answer questions, *simplicity and clarity of solutions will count.* You may get as few as 0 point for a problem if your solution is far more complicated than necessary.
- Unless otherwise specified:
  - Assume that all relations are duplicate-free in questions not related to SQL.
  - Assume that all questions about SQL refer to the SQL2 or SQL3 standard, *not* necessarily to the Oracle implementation of SQL used in programming assignments.

NAME: ______________________________

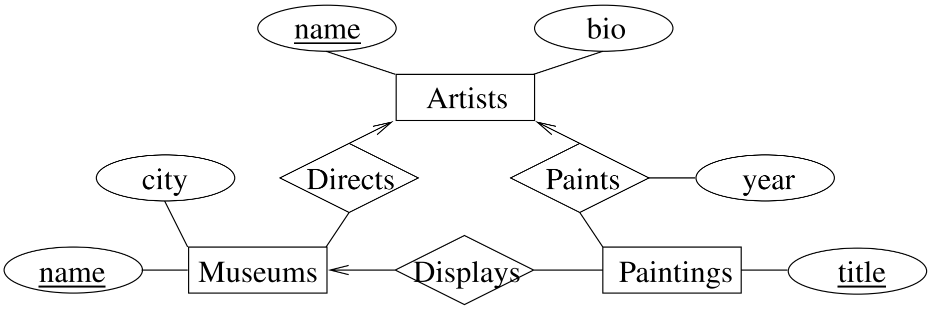
In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: ______________________________

Questions	Max points	Points
1–24	46 (+2)	
25–26	10	
27–29	13	
30	6	
31–32	5 (+8)	
TOTAL	80 (+10)	

Part I. Multiple-Choice Questions (50 points)

Questions 1–4 below refer to a database of museums, paintings, and artists. Museums display paintings; paintings are the work of artists; some artists are also museum directors. More assumptions are encoded in the E/R diagram below, as key and multiplicity specifications.



Question 1: (2 points) Suppose that we want to specify an ODL schema that captures the design of the above E/R diagram. The three entity sets correspond to the following three ODL classes:

```
interface Artist (extent Artists, key name) {
  attribute string name;
  attribute string bio;
  relationship Set<Museum> directs inverse Museum::isDirectedBy;
  relationship Set<Painting> paints inverse Painting::isPaintedBy;
};
interface Painting (extent Paintings, key title) {
  attribute string title;
  attribute integer year;
  relationship Artist isPaintedBy inverse Artist::paints;
  relationship Museum isOnDisplayAt inverse Museum::displays;
};
interface Museum (extent Museums, key name) {
  attribute string name;
  attribute string city;
  relationship __?__ isDirectedBy inverse Artist::directs;
  relationship __?__ displays inverse Painting::isOnDisplayAt;
};
```

The type declarations for `isDirectedBy` and `displays` are missing. What should they be, respectively?

(A) `Artist` and `Painting`
(B) `Artist` and `Set<Painting>`
(C) `Set<Artist>` and `Painting`
(D) `Set<Artist>` and `Set<Painting>`

Answer: ☐

Question 2: (2 points) What is the type of the result of the following OQL query?

```
SELECT a.name, a.paints FROM Artists a;
```

(A) `Struct {string name, Painting paints}`
(B) `Struct {string name, Set<Painting> paints}`
(C) `Bag<Struct {string name, Painting paints}>`
(D) `Bag<Struct {string name, Set<Painting> paints}>`

Answer: ☐

Question 3: (2 points) Which of the following two OQL queries will correctly find all works of Monet on display in Paris?

I.
```
SELECT p
FROM   Paintings p
WHERE  p.isPaintedBy.name = "Monet"
AND    p.isOnDisplayAt.city = "Paris";
```

II.
```
SELECT DISTINCT p
FROM   Paintings p, Artists a, Museums m
WHERE  p = a.paints AND a.name = "Monet"
AND    p = m.displays AND m.city = "Paris";
```

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: ☐

Question 4: (2 points) Which of the following queries require an additional `DISTINCT` (after `SELECT`) to ensure that the output contains no duplicate artists?

I.
```
SELECT a
FROM   Artists a
WHERE  EXISTS p IN a.paints:
        (EXISTS m IN a.directs: m = p.isOnDisplayAt)
ORDER BY a.name;
```

II.
```
SELECT p.isPaintedBy
FROM   Paintings p
WHERE  EXISTS m IN p.isPaintedBy.directs: m = p.isOnDisplayAt
ORDER BY p.isPaintedBy.name;
```

III.
```
SELECT a
FROM   Artists a, a.paints p, a.directs m
WHERE  p.isOnDisplayAt = m
ORDER BY a.name;
```

(A) I (B) I and II (C) II and III (D) III

Answer: ☐

Question 5: (2 points) Consider two relations $R(\underline{A}, B)$ and $S(\underline{B}, C)$, where $R.A$ is the only key of R and $S.B$ is the only key of S. Let relation $T(A, B, C)$ be the natural join of R and S, i.e., $T = R \bowtie S$. Which of the following dependencies must hold in T?

I. $A \rightarrow C$ II. $B \twoheadrightarrow A$

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: $\boxed{}$

Question 6: (2 points) Suppose we have a relation $R(A, B, C, D)$ with the FD's:

$$A \rightarrow B, \quad B \rightarrow C, \quad C \rightarrow D$$

Which one of the following decompositions is *not* lossless (i.e., for some instance of R, the natural join of the decomposed relations is not equal to R)?

(A) $R_1(A, B)$, $R_2(B, C)$, $R_3(C, D)$
(B) $R_1(A, B)$, $R_2(A, C)$, $R_3(A, D)$
(C) $R_1(A, D)$, $R_2(B, D)$, $R_3(C, D)$
(D) None of the above (that is, they are all lossless)

Answer: $\boxed{}$

Question 7: (2 points) Suppose we have a relation $R(A, B, C, D)$ with the MVD $A \twoheadrightarrow CD$. We know that R at least contains the following two tuples:

A	B	C	D
1	1	1	1
1	2	2	2

What is the minimum number of tuples in R (including the two above)?

(A) 2 (B) 4 (C) 6 (D) 8 or more

Answer: $\boxed{}$

Question 8: (2 points) [**extra credit**] Consider a relation $R(A, B, C, D)$ with the following set of dependencies:

$$A \to \spadesuit, \quad CD \to A, \quad A \twoheadrightarrow CD$$

Unfortunately, we don't know what $\spadesuit$ is—it could be any nonempty subset of R's attributes. However, we do know that the given set of dependencies form a minimal basis; that is, every dependency in this set is nontrivial and cannot be inferred from the others in the set. Which one of the following must be true regardless of what is inside $\spadesuit$?

I. R is in BCNF
II. R is not in BCNF
III. R is in 3NF
IV. R is not in 3NF

(A) II and IV (B) II (C) II and III (D) I and III

Answer: $\boxed{}$

Questions 9–10 below refer to a relation $R(A, B, C, D, E)$ with the following FD's:

$$A \to B, \quad A \to C, \quad BD \to A$$

Question 9: (2 points) Which FD's are 4NF violations but *not* 3NF violations?

(A) $A \to C$ only
(B) $BD \to A$ only
(C) $A \to B$ and $A \to C$ only
(D) $A \to B$ and $BD \to A$ only

Answer: $\boxed{}$

Question 10: (2 points) If we decompose R into BCNF starting with $A \to B$, which FD *cannot* be preserved (that is, it cannot be inferred from the FD's that hold in the decomposed relations)?

(A) $A \to B$
(B) $A \to C$
(C) $BD \to A$
(D) None of the above (that is, all FD's in R are preserved in the decomposition)

Answer: $\boxed{}$

Questions 11–12 are based on the following schema:

```
CREATE TABLE T(C INT PRIMARY KEY,
               D INT);

CREATE TABLE S(B INT PRIMARY KEY,
               C INT REFERENCES T(C) ON DELETE CASCADE);

CREATE TABLE R(A INT PRIMARY KEY,
               B INT REFERENCES S(B) ON DELETE SET NULL);
```

Question 11: (2 points) Suppose the current contents of R, S, and T are as follows:

R:

A	B
1	1
2	2

S:

B	C
1	1
2	1

T:

C	D
1	1
2	1

After executing the command

```
DELETE FROM T;
```

what tuples will R contain?

(A) (1, NULL) and (2, 2)
(B) (1, NULL) and (2, NULL)
(C) (2, 2) only
(D) R will contain no tuples

Answer: ☐

Question 12: (2 points) Suppose that tables R, S, and T (*as declared above*) contain r, s, and t tuples respectively. Let n be the number of tuples in the result of the following query:

```
SELECT * FROM R, S, T WHERE R.B = S.B AND S.C = T.C;
```

What is the most restrictive yet correct upper bound of n?

(A) $n \leq r$ (B) $n \leq t$ (C) $n \leq r \times s \times t$ (D) $n \leq min(r, s, t)$

Answer: ☐

Questions 13–14 are based on the following schema:

```
CREATE TABLE R                    CREATE TABLE S
  (A INT UNIQUE NOT NULL,           (A INT NOT NULL,
   B INT UNIQUE NOT NULL,            B INT NOT NULL,
   C INT NOT NULL);                  C INT NOT NULL,
                                     PRIMARY KEY (A, B));
```

Question 13: (2 points) Which of the following modification statements may violate the `PRIMARY KEY` constraint on `S`?

I.
```
INSERT INTO S (SELECT * FROM R
               WHERE NOT EXISTS
                     (SELECT * FROM S s1
                      WHERE s1.A = R.A AND s1.B = R.B));
```

II.
```
INSERT INTO S (SELECT * FROM R
               WHERE R.A NOT IN (SELECT A FROM S));
```

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: ☐

Question 14: (2 points) Which of the following SQL3 triggers, when fired, may violate the `UNIQUE` constraints on `R`? (Note that these `AFTER` triggers will not see any insertions that violate the `PRIMARY KEY` constraint on `S`.)

I.
```
CREATE TRIGGER trigger1
AFTER INSERT ON S
REFERENCING NEW AS newTuple
WHEN (newTuple.A NOT IN (SELECT A FROM R) AND
      newTuple.B NOT IN (SELECT B FROM R))
INSERT INTO R VALUES(newTuple.A, newTuple.B, newTuple.C)
FOR EACH ROW;
```

II.
```
CREATE TRIGGER trigger2
AFTER INSERT ON S
REFERENCING NEW_TABLE AS newTable
INSERT INTO R (SELECT * FROM newTable
               WHERE newTable.A NOT IN (SELECT A FROM R) AND
                     newTable.B NOT IN (SELECT B FROM R));
```

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: ☐

Question 15: (2 points) Consider a relation $R(A, B, C, D)$. Suppose we want to enforce the FD $A \rightarrow BC$ in R using SQL3 triggers. Which of the following triggering events are required?

I. UPDATE OF D ON R II. DELETE ON R

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: $\square$

Question 16: (2 points) Consider a relation $R(A, B, C, D)$. This time, suppose we want to enforce the MVD $A \twoheadrightarrow BC$ using SQL3 triggers. Which of the following triggering events are required?

I. UPDATE OF D ON R II. DELETE ON R

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: $\square$

Questions 17–18 are based on a simple relation TA(SID,CID,stipend) storing TA assignments and stipends. TA's are identified by their student ID's (SID's). Consider the following two transactions, executed by two different clients at approximately the same time:

```
-- T1:
UPDATE TA
SET    stipend = stipend + 200
WHERE  CID = 'CS145';
UPDATE TA
SET    stipend = stipend + 200
WHERE  CID = 'CS145';
COMMIT;
```

```
-- T2:
UPDATE TA
SET    stipend = 2000
WHERE  stipend > 2000;
COMMIT;
```

Before either transaction starts, there is a tuple (987, 'CS145', 1900) in TA.

Question 17: (2 points) Suppose T1 executes with isolation level READ COMMITTED and T2 executes with isolation level READ UNCOMMITTED. Which one of the following figures *cannot* be a possible final stipend value for TA 987?

(A) 2000 (B) 2200 (C) 2300 (D) None of the above (that is, all of them are possible)

Answer: $\square$

Question 18: (2 points) Now, suppose instead that T1 executes with isolation level REPEATABLE READ and T2 executes with isolation level READ COMMITTED. Which one of the following figures *cannot* be a possible final stipend value for TA 987?

(A) 2000 (B) 2200 (C) 2300 (D) None of the above (that is, all of them are possible)

Answer: $\square$

Questions 19–24 are based on a relation `Enroll(SID,CID,term,grade)` which stores the academic records of students in a university database. For example, a tuple (123, `'CS145'`, `'Spring 1999'`, 4.0) represents the fact that student 123 took CS145 in Spring 1999 and received a grade of 4.0 (which is an A).

Question 19: (2 points) Someone in Registrar's Office complained that he obtained different results when repeating the same query inside one transaction:

```
-- T1:
SELECT AVG(grade) FROM Enroll; -- Q1
SELECT AVG(grade) FROM Enroll; -- Q2
COMMIT;
```

Which one of the following actions is sufficient *and* necessary to ensure that `Q1` and `Q2` always return the same result?

(A) Make all transactions, including `T1`, run at `SERIALIZABLE`
(B) Make all transactions, including `T1`, run at `REPEATABLE READ`
(C) Make `T1` run at `SERIALIZABLE`
(D) Make `T1` run at `REPEATABLE READ`

Answer: ☐

Question 20: (2 points) Student 123 ran two queries over his grade history inside a transaction:

```
-- T2:
SELECT MIN(grade) FROM Enroll WHERE SID = 123; -- QMIN
SELECT MAX(grade) FROM Enroll WHERE SID = 123; -- QMAX
COMMIT;
```

He was terrified to find that his highest grade was lower than his lowest grade! Suppose that we know, in the university database, only insertions are allowed on `Enroll`; updates and deletions are disallowed. Under this assumption, which one of the following actions is sufficient *and* necessary to ensure that the result of `QMAX` is always no less than the result of `QMIN`?

(A) Make all transactions, including `T2`, run at `SERIALIZABLE`
(B) Make `T2` run at `SERIALIZABLE`
(C) Make `T2` run at `REPEATABLE READ`
(D) Make `T2` run at `READ COMMITTED`

Answer: ☐

Question 21: (2 points) Suppose that we have defined a view V over the Enroll relation:

```
CREATE VIEW V AS
  SELECT * FROM Enroll e1
  WHERE NOT EXISTS (SELECT * FROM Enroll e2
                    WHERE e2.CID = e1.CID
                    AND e2.grade > e1.grade);
```

Here is a query involving V:

```
SELECT DISTINCT CID FROM V
WHERE grade = 3.0;
```

Which of the following two queries will return the same result as the query above?

I.
```
SELECT DISTINCT e1.CID FROM Enroll e1
WHERE e1.grade = 3.0
AND NOT EXISTS (SELECT * FROM Enroll e2
                WHERE e2.CID = e1.CID
                AND e2.grade > 3.0);
```

II.
```
SELECT DISTINCT CID FROM Enroll
GROUP BY CID
HAVING MAX(grade) = 3.0;
```

(A) I only (B) II only (C) Both I and II (D) Neither I nor II

Answer: ☐

Question 22: (2 points) Suppose we want to find the average grade of the CS145 class in Spring 1999. Assume that there is no other index on Enroll. Which one of the following options will likely provide the biggest performance improvement for our query?

(A) Create an index on Enroll(SID)
(B) Create an index on Enroll(SID) and another on Enroll(grade)
(C) Create an index on Enroll(CID) and another on Enroll(term)
(D) Create an index on Enroll(CID,term)

Answer: ☐

Question 23: (2 points) Which of the following queries are *not* monotone with respect to Enroll?

I. $\sigma_{\texttt{CID} \neq \text{“}\texttt{CS145}\text{”}}(\texttt{Enroll})$
II. $\texttt{Enroll} - \sigma_{\texttt{CID} = \text{“}\texttt{CS145}\text{”}}(\texttt{Enroll})$
III. $\pi_{\texttt{SID}}(\texttt{Enroll}) - \pi_{\texttt{SID}}(\sigma_{\texttt{CID} = \text{“}\texttt{CS145}\text{”}}(\texttt{Enroll}))$

(A) I only (B) I and II only (C) II and III only (D) III only

Answer: ☐

Question 24: (2 points)* Consider the following SQL3 `WITH` statement:

```
WITH RECURSIVE SuperStat(SID, numSuper) AS
       -- for each student,
       -- count the number of super courses taken
       SELECT SID, COUNT(*)
       FROM   Enroll, SuperCourse
       WHERE  Enroll.CID = SuperCourse.CID
       GROUP BY SID,
     RECURSIVE SuperStudent(SID) AS
       -- a super student is a student
       -- who has taken more than 3 super courses
       SELECT SID
       FROM   Enroll, SuperCourse
       WHERE  Enroll.CID = SuperCourse.CID
       GROUP BY SID
       HAVING COUNT(*) > 3,
     RECURSIVE SuperCourse(CID) AS
       -- a super course is a course
       -- that has been taken by more than 7 super students
       SELECT CID
       FROM   Enroll, SuperStudent
       WHERE  Enroll.SID = SuperStudent.SID
       GROUP BY CID
       HAVING COUNT(*) > 7
SELECT * FROM SuperStat;
```

What is the stratum of `SuperStat`?

(A) 0 (B) 1 (C) 2 (D) Infinity

Answer: ☐

*This question is adapted from a problem designed by Yuandong Wang.

Part II. Short-Answer Questions (40 points)

Questions 25–26 are based on the same relation `Enroll(SID,CID,term,grade)` used in Questions 19–24.

Question 25: (4 points) Consider the following SQL query:

```
SELECT SID FROM Enroll GROUP BY SID HAVING MAX(grade) > 4.0;
```

Write an equivalent query in ***Relational Algebra***.

Question 26: (6 points) Consider the following SQL query:

```
SELECT SID FROM Enroll GROUP BY SID HAVING MIN(grade) > 3.0;
```

Write an equivalent query in ***Relational Algebra***.

Questions 27–29 are based on a relation `Rating(univ,prog,score)` containing ratings of doctorate programs at major universities as compiled by *US News & World Report*. An excerpt of the data is shown below:

univ	prog	score
MIT	cs	5.0
Stanford	cs	5.0
Carnegie Mellon	cs	4.9
UC Berkeley	cs	4.9
...	...	...
Caltech	physics	5.0
Stanford	physics	5.0
...	...	...

(Continued from the previous page)

Question 27: (2 points) Suppose we want to specify the following constraint on the `Rating` table: Stanford must always have at least one program with a higher score than its counterpart at UC Berkeley. Can you write a tuple-based `CHECK` on `Rating` to enforce this constraint? Circle **YES** or **NO**. If your answer is no, briefly explain why.

Question 28: (5 points) If your answer to Question 27 is yes, write a tuple-based `CHECK` to enforce the above constraint. If your answer is no, write a general assertion to enforce the constraint.

Question 29: (6 points) Write a SQL query to find the rank of Stanford's math program. When computing the ranking for math programs, you should ignore the scores for other programs. Moreover, make sure you consider the case of ties. Take the CS ranking as an example. In the instance of `Enroll` shown above, MIT and Stanford tie for the first place. Therefore, there is no second place; both CMU and UC Berkeley end up at the third. Please write a standard SQL2 query—no embedded SQL, PL/SQL, or other special Oracle features.

Question 30: (6 points) Suppose a relation `FD(lhs,rhs)` stores a set of FD's of the form `lhs` $\rightarrow$ `rhs`, where `lhs` and `rhs` each contain one single attribute. We are also given a set of attributes stored in a relation `AttrSet(attr)`. Write a SQL3 `WITH` statement to compute `Closure(attr)`, the set of attributes in the closure of `AttrSet` with respect to `FD`.

As a concrete example, your `WITH` statement should be able compute that the closure of $\{A, F\}$ with respect to $\{A \rightarrow B, A \rightarrow C, C \rightarrow D, E \rightarrow C\}$ is $\{A, B, C, D, F\}$:

`FD:`

lhs	rhs
A	B
A	C
C	D
E	C

`AttrSet:`

attr
A
F

`Closure:`

attr
A
F
B
C
D

Questions 31–32 are based on a relation `Knows(person1,person2)`. A tuple (p, q) in `Knows` records the fact that p knows q. It should be noted that `Knows` is not symmetric: p knows q does not necessarily imply q knows p. `Knows` certainly can contain cycles: for example, p knows q, q knows r, and r may know p.

We define the *distance* from p to q as follows. The distance from p to q ($p \neq q$) is n if there is a sequence of $n-1$ persons $p_1, ..., p_{n-1}$ such that p knows p_1, p_1 knows p_2, ..., and p_{n-1} knows q; furthermore, there does not exist any shorter sequence of intermediate persons linking p to q.

Question 31: (5 points) Write a nonrecursive SQL query to find all pairs of persons (p, q) such that $p \neq q$, and the distance from p to q is *exactly* 2.

Question 32: (8 points) [**extra credit**] Write a SQL3 `WITH` statement to compute the distance from Mr. Gates to everybody else. Please make sure that the statement is legal in SQL3 and the fixed-point iteration always converges. For simplicity, you may assume that everybody in the database is within a finite distance from Mr. Gates.