CS145 Final Examination

Spring 2001, Prof. Widom

- Please read all instructions (including these) carefully.
- There are 13 problems on the exam, with a varying number of points for each problem and subproblem for a total of 120 points. *You should look through the entire exam before getting started, in order to plan your strategy.*
- The exam is closed book and closed notes, but you may refer to your three pages of prepared notes.
- Please write your solutions in the spaces provided on the exam. Make sure your solutions are neat and clearly marked. You may use the blank areas and backs of the exam pages for scratch work. Please do not use any additional scratch paper.
- *Simplicity and clarity of solutions will count.* You may get as few as 0 points for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.

NAME: ____

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

SIGNATURE: ____

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	TOTAL
Max. points	8	9	5	24	6	12	6	10	10	10	6	8	6	120
Points														

1. **Authorization** (8 points)

   Consider relation `Employee(ID,salary,dept)` where `ID` is a key. Suppose you are the owner of this relation, and you would like to give user "Mary" authorization to select records for those employees (and only those employees) who earn less than $80,000 and work in a department with fewer than 10 people. Specify a SQL command or sequence of commands that achieves this goal.

2. **Transactions** (9 points)

   On this problem you will receive 3 points for every correct answer, −1 point for every incorrect answer, and 0 points for no answer.

   Consider again relation `Employee(ID,salary,dept)` where `ID` is a key. Consider the following transaction `T`:

```
T: Q1: SELECT avg(salary) FROM EMPLOYEE
   <read-only activity>
   Q2: SELECT avg(salary) FROM EMPLOYEE
   commit
```

   (a) Suppose all other transactions in the system are declared as `serializable` and `read-only`. What is the weakest isolation level needed for transaction `T` to ensure that queries `Q1` and `Q2` will always get the same result? Circle one:

   `READ UNCOMMITTED`

   `READ COMMITTED`

   `REPEATABLE READ`

   `SERIALIZABLE`

   (Problem continues on next page)

(b) Suppose all other transactions in the system are declared as `serializable`, and they only involve queries, updates, and deletions. What is the weakest isolation level needed for transaction `T` to ensure that queries `Q1` and `Q2` will always get the same result? Circle one:

```
READ UNCOMMITTED
READ COMMITTED
REPEATABLE READ
SERIALIZABLE
```

(c) Suppose all other transactions in the system are declared as `serializable`, and we know nothing else about them. What is the weakest isolation level needed for transaction `T` to ensure that queries `Q1` and `Q2` will always get the same result? Circle one:

```
READ UNCOMMITTED
READ COMMITTED
REPEATABLE READ
SERIALIZABLE
```

3. **Recursion** (5 points)

Consider the following two queries over a relation $R(A, B)$ where A is a key.

```
Query 1:  WITH RECURSIVE S(A,B) AS
                 ((SELECT * FROM R)
                   UNION
                  (SELECT B,A FROM S))
          SELECT * FROM S

Query 2:  (SELECT * FROM R)
           UNION
          (SELECT B,A FROM R)
```

Are these two queries equivalent? That is, do they return the same answer over all possible instances of relation R?

Circle one: YES NO

You will receive 5 points for a correct answer, -2 points for an incorrect answer, and 0 points for no answer.

4. **XML** (24 points)

Consider the following four Document Type Descriptors (DTDs) for XML documents containing data about books and authors.

```
<!DOCTYPE DTD1 [
  <!ELEMENT DB1 (BOOK*)>
  <!ELEMENT BOOK (TITLE, AUTHOR+)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT AUTHOR>
  <!ATTLIST AUTHOR ssn ID name CDATA>
]>

<!DOCTYPE DTD2 [
  <!ELEMENT DB2 (AUTHOR*)>
  <!ELEMENT AUTHOR (BOOK)>
  <!ATTLIST AUTHOR ssn ID name CDATA>
  <!ELEMENT BOOK (TITLE)>
  <!ATTLIST BOOK isbn ID>
  <!ELEMENT TITLE (#PCDATA)>
]>

<!DOCTYPE DTD3 [
  <!ELEMENT DB3 (BOOK*, AUTHOR*)>
  <!ELEMENT BOOK (TITLE)>
  <!ATTLIST BOOK isbn ID author IDREF>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT AUTHOR>
  <!ATTLIST AUTHOR ssn ID name CDATA>
]>

<!DOCTYPE DTD4 [
  <!ELEMENT DB4 (AUTHOR*, BOOK*)>
  <!ELEMENT AUTHOR>
  <!ATTLIST AUTHOR ssn ID book IDREFS>
  <!ELEMENT BOOK (TITLE)>
  <!ATTLIST BOOK isbn ID>
  <!ELEMENT TITLE (#PCDATA)>
]>
```

Remember that attributes of type `ID` must have unique values within a document.

(Problem continues on next page)

For each of the following problem parts, you will receive 1.5 points for every correct circle or X, −.5 points for every incorrect circle or X, and 0 points for each omitted circle or X.

(a) Which of the four DTDs can be used for an XML file that encodes a many-many relationship between books and authors? Circle those that can, and put an X through those that can't: `DTD1 DTD2 DTD3 DTD4`

(b) Which of the four DTDs can be used for an XML file that encodes a many-one relationship from books to authors? Circle those that can, and put an X through those that can't: `DTD1 DTD2 DTD3 DTD4`

(c) Which of the four DTDs can be used for an XML file that encodes a one-many relationship from books to authors? Circle those that can, and put an X through those that can't: `DTD1 DTD2 DTD3 DTD4`

(d) Which of the four DTDs can be used for an XML file that encodes a one-one relationship between books and authors? Circle those that can, and put an X through those that can't: `DTD1 DTD2 DTD3 DTD4`

SCRATCH SPACE (not graded!)

5. **ODL** (6 points)

Consider the following three ODL schemas:

```
Schema1:
  interface Book {
    attribute string title;
    relationship Set<Author> author
      inverse Author::book; }

  interface Author {
    attribute string name;
    relationship Set<Book> book
      inverse Book::author; }
```

```
Schema2:
  interface Book {
    attribute string title;
    relationship Set<Author> author
      inverse Author::book; }

  interface Author {
    attribute string name;
      relationship Book book
      inverse Book::author; }
```

```
Schema3:
  interface Book {
    attribute string title;
    relationship Author author
      inverse Author::book; }

  interface Author {
    attribute string name;
    relationship Book book
      inverse Book::author; }
```

For each of the following problem parts, you will receive 2 points for every correct answer, −1 point for every incorrect answer, and 0 points for no answer.

(a) `Schema1` is most appropriate for data containing (circle one):

- One-one relationships between books and authors
- Many-one relationships from books to authors
- One-many relationships from books to authors
- Many-many relationships between books and authors

(b) `Schema2` is most appropriate for data containing (circle one):

- One-one relationships between books and authors
- Many-one relationships from books to authors
- One-many relationships from books to authors
- Many-many relationships between books and authors

(c) `Schema3` is most appropriate for data containing (circle one):

- One-one relationships between books and authors
- Many-one relationships from books to authors
- One-many relationships from books to authors
- Many-many relationships between books and authors

6. **More ODL** (12 points)

On this problem you will receive 2 points for every correct answer, -1 point for every incorrect answer, and 0 points for no answer.

Consider the following ODL schema:

```
interface X {                          interface Y (key C) {
  unique attribute integer A;            unique attribute integer C;
  attribute integer B;                   attribute integer D;
  relationship Y R1                      relationship Set<X> R2
    inverse Y::R2; }                       inverse X::R1; }
```

For each of the relational schemas in parts (a)–(e), specify whether it is valid or invalid as a relational translation of the above ODL schema. For this problem, a "valid" translation is simply one that can encode the same data and relationships—do not worry about any specific translation algorithm from class or the textbook.

(a) $X(\underline{A}, B, C), Y(\underline{C}, D)$

Circle one: Valid Invalid

(b) $X(\underline{A}, B), Y(\underline{C}, D), R(A, \underline{C})$

Circle one: Valid Invalid

(c) $X(\underline{A}, B), Y(\underline{C}, D), R(\underline{A}, C)$

Circle one: Valid Invalid

(d) $X(\underline{A}, B), Y(\underline{C}, D, \underline{A})$

Circle one: Valid Invalid

(e) $X(\underline{A}, B), Y(\underline{C}, D, A)$

Circle one: Valid Invalid

(f) One of the valid schemas is not in Boyce-Codd Normal Form (BCNF).
Which one?

Circle one: (a) (b) (c) (d) (e)

7. **General Assertions** (6 points)

Consider a SQL relation $R(A, B)$. For the following two general assertions, write in the box provided the constraint that is enforced by the assertion. Please be specific about the actual attributes involved in the constraint. In both cases the correct answer corresponds to a single concept from the course and can be specified in a few words.

(a) Assertion:

```
CREATE ASSERTION WarmUp AS
   (NOT EXISTS (SELECT A FROM R
                GROUP BY A
                HAVING COUNT(*) > 1))
```

Constraint:

(b) Assertion:

```
CREATE ASSERTION Tricky AS
   (NOT EXISTS ((SELECT * FROM R)
                EXCEPT
                (SELECT * FROM R WHERE A = A)))
```

Constraint:

8. **Triggers and Dependencies** (10 points)

On this problem you will receive 1 point for every correct circle or X, −.5 points for every incorrect circle or X, and 0 points for each omitted circle or X.

(a) Consider a relation $R(A, B, C)$ and suppose we want to enforce the functional dependency $A \rightarrow B$ on R using triggers. For each of the following operations, if we need to create a trigger on this operation in order to enforce our functional dependency, circle the operation. If we don't need a trigger on this operation, put an X through it.

```
INSERT ON R
DELETE ON R
UPDATE OF A ON R
UPDATE OF B ON R
UPDATE OF C ON R
```

(Problem continues on next page)

(b) Now suppose we want to enforce the multivalued dependency $A \twoheadrightarrow B$ on R using triggers. For each of the following operations, if we need to create a trigger on this operation in order to enforce our multivalued dependency, circle the operation. If we don't need a trigger on this operation, put an X through it.

```
INSERT ON R
DELETE ON R
UPDATE OF A ON R
UPDATE OF B ON R
UPDATE OF C ON R
```

9. **Triggers and Referential Integrity** (10 points)

Consider relations $R(A, B, C)$ and $S(D, E, F)$ where A is a primary key for R and D is a key for S. Suppose we want to use triggers to enforce a constraint that $S.F$ is a foreign key referencing $R.A$ with the "`on update cascade`" policy. The following skeleton trigger is meant to handle updates to $R.A$. Fill in the blanks, and remember that your solution will be graded on simplicity as well as correctness.

```
CREATE TRIGGER UpdCascade
AFTER UPDATE OF A ON R

REFERENCING [                    ]

WHEN [                    ]

[                    ] (trigger action)

FOR EACH ROW
```

10. **Object-Relational** (10 points)

Consider the following SQL3 relational schema. You may assume that all attributes without a declared type have type `STRING`.

```
CREATE ROW TYPE StudInfo AS (name, majorDept)
CREATE ROW TYPE StudType AS (ID, info StudInfo)
CREATE ROW TYPE CourseType AS (number, dept, title)

CREATE TABLE Student OF TYPE StudType PRIMARY KEY (ID)
CREATE TABLE Course OF TYPE CourseType PRIMARY KEY (number)
CREATE TABLE Took(student REF(StudType),course REF(CourseType),grade)
```

Write a SQL3 query to find the IDs (no duplicates please) of all students who took and got an 'A' in a course offered by a department other than the student's major department. Remember that your solution will be graded on simplicity as well as correctness.

11. **Application Servers** (6 points)

In John Hampton's lecture on Application Servers, he discussed two major topics covered in CS145 that are handled significantly differently in an *n-tier* environment than in a traditional *client-server* database environment. Name those two topics:

(a)

(b)

12. **Data Warehousing and Materialized Views** (8 points)

On this problem you will receive 4 points for every correct answer, -2 points for every incorrect answer, and 0 points for no answer.

Consider a relation `AllData` that is the result of joining the fact table and all of the dimension tables in an OLAP-style *star schema*. Suppose relation `AllData` includes attributes $A_1, A_2, \ldots, A_n$ that are foreign keys to dimension tables, as well as other attributes from the fact table. Let an OLAP *star query* be a query that groups by one or more attributes in $A_1, A_2, \ldots, A_n$ and performs a sum on an `AllData` attribute from the fact table, as discussed in class. Specify whether each of the following two statements is true or false.

(a) Consider a star query Q_1 over `AllData` that groups by attributes $A_1, A_2, \ldots, A_i$, and another star query Q_2 that groups by $A_1, A_2, \ldots, A_j$, where $i < j < n$. The result of query Q_1 will always be no bigger than the result of query Q_2, where size is measured as number of tuples in the result.

Circle one: True False

(b) Consider a materialized view V_1 specified as a star query over `AllData` that groups by attributes $A_1, A_2, \ldots, A_i$, and another materialized view V_2 specified by a star query that groups by $A_1, A_2, \ldots, A_j$, where $i < j < n$. Materialized view V_1 can be used to help answer a larger number of star queries than materialized view V_2.

Circle one: True False

13. **Mystery Topic** (6 points)

Consider the following two SQL table declarations:

```
CREATE TABLE T1 (A integer PRIMARY KEY, B integer)

CREATE TABLE T2 (C integer PRIMARY KEY, D integer)
```

Make the smallest possible modifications to these table declarations you can think of in order to ensure that the full outerjoin of `T1` and `T2` on `A=C` is always equivalent to the join of `T1` and `T2` on `A=C`. On this problem you will *really* be graded on simplicity as well as correctness.