CS145 Final Examination

Autumn 2010, Prof. Widom

- Please read all instructions (including these) carefully.
- There are 10 problems on the exam, with a varying number of points for each problem and subproblem for a total of 120 points to be completed in 120 minutes. *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. The blank areas and backs of the exam pages may be used for *ungraded* scratch work.
- *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	TOTAL
Max. points	12	20	18	5	15	5	10	15	12	8	120
Points											

1. **UML** (12 points)

   Consider the following relational schema describing movies, people, and movie ratings:

```
Movie(title, year, director)     // title is a key
Person(name, age, gender)        // name is a key
Rating(name, title, date, score) // <name,title> is a minimal key
```

   (a) **(6 points)** Draw a UML diagram from which this relational schema could have been produced using one of the translations given in class. Do not draw a diagram with three independent classes—your diagram should be fully connected, and it should be as detailed as possible from the information you have.

   (b) **(2 points)** Now suppose for relation `Rating` we instead have that `name` alone is a key. If your UML diagram changes as a result, state or show the change. (You do not need to redraw the entire diagram, but we must understand exactly what change(s) are in your answer.)

(problem continues on next page)

(c) **(2 points)** Now suppose for relation `Rating` we still have that `name` alone is a key, but we also have that `title` alone is a key. If your UML diagram changes from part (b) as a result, state or show the change. (You do not need to redraw the entire diagram, but we must understand exactly what change(s) are in your answer.)

(d) **(2 points)** Finally suppose for relation `Rating` that `name` alone is a key, `title` alone is a key, and there are two *referential integrity* constraints: one from `Movie.Title` to `Rating.title` (i.e., `Movie.Title` references `Rating.title`), and similarly one from `Person.name` to `Rating.name`. If your UML diagram changes from part (c) as a result of this additional information, state or show the change. (You do not need to redraw the entire diagram, but we must understand exactly what change(s) are in your answer.)

2. **Keys, Referential Integrity, and Triggers** (20 points)

   Consider tables `T1(P,A)` and `T2(F,B)`. This problem explores using triggers to enforce two constraints:

   1) *Key constraint on* `T1.P`
   2) *Referential integrity constraint from* `T2.F` *to* `T1.P`

   To keep things simple, you may assume there are never `null` values for `T1.P`.

   (a) **(4 points)** List all of the data modification operations on `T1` and `T2` that could cause either the key constraint or the referential integrity constraint to become violated. For `update` operations, include the specific columns. You do not need to associate the operations with which constraint(s) they may affect.

   (b) **(8 points)** We hope that "updates to `T1.P`" was part of your answer to part (a). (If it wasn't, you can add it now!) Next you will specify triggers to enforce the two constraints when column `T1.P` is updated. In this part of the problem, you will specify a row-level `before` trigger for the key constraint and a row-level `after` trigger for the referential integrity constraint.

   - You may assume that when a tuple in `T1` is updated, the new value of `P` in that tuple is different from the old one. Make no other assumptions about the updates.
   - For the key constraint, you should execute a special "`raise-error`" command when the constraint is violated. This command will abort the statement that caused the violation.
   - For the referential integrity constraint, please implement the "*On Update Cascade*" policy.

   Fill in the blanks in the following skeletons. Try to make use of trigger features to enforce the constraints, but without getting overly complex. Note that it is fine to leave some boxes empty, as appropriate. *Please use SQL-99 triggers, not those implemented in a specific system.*

   (problem continues on next page)

```
Create Trigger UpdKey
Before Update of P on T1
Referencing
For Each Row
When
```

(action)

```
Create Trigger UpdRI
After Update of P on T1
Referencing
For Each Row
When
```

(action)

(problem continues on next page)

(c) **(8 points)** Repeat part (b), but with the following changes:

- There is no "`For Each Row`", so you are specifying statement-level triggers.
- Both triggers are `After`, but you may assume trigger `UpdKey` executes first.
- For referential integrity, please implement the "*On Update Set Null*" policy.

Once again, try to make use of trigger features to enforce the constraints without getting overly complex, and use SQL-99 triggers rather than those implemented in a specific system.

```
Create Trigger UpdKey
After Update of P on T1
```

`Referencing`

`When`

(action)

```
Create Trigger UpdRI
After Update of P on T1
```

`Referencing`

`When`

(action)

3. **Transactions** (18 points, 3 per part)

Consider table `Giants(player,salary)` where `player` is a key, and the following two transactions:

```
T1: Begin Transaction
    S1: update Giants set salary = 2*salary where player = 'Buster Posey'
    S2: update Giants set salary = 3*salary where player = 'Buster Posey'
    Commit

T2: Begin Transaction
    S3: update Giants set salary = salary-20 where player = 'Buster Posey'
    S4: update Giants set salary = salary-10 where player = 'Buster Posey'
    Commit
```

You may assume that the individual statements `S1`, `S2`, `S3`, and `S4` always execute atomically. Let Buster's salary be 50 before either transaction executes.

(a) Suppose both transactions `T1` and `T2` execute to completion with isolation level `Serializable`. What are Buster's possible final salaries?

(b) Suppose both transactions `T1` and `T2` execute to completion with isolation level `Read-Committed`. What are Buster's possible final salaries?

(c) Suppose transaction `T1` executes with isolation level `Read-Committed`, transaction `T2` executes with isolation level `Read-Uncommitted`, and both transactions execute to completion. What are Buster's possible final salaries?

(problem continues on next page)

(d) Suppose both transactions `T1` and `T2` execute to completion with isolation level `Read-Uncommitted`. What are Buster's possible final salaries?

(e) Suppose both transactions `T1` and `T2` execute with isolation level `Serializable`. Transaction `T1` executes to completion, but transaction `T2` rolls back after statement `S3` and does not re-execute. What are Buster's possible final salaries?

(f) (*this one's a bit tricky*) Suppose both transactions `T1` and `T2` execute with isolation level `Read-Uncommitted`. Transaction `T1` executes to completion, but transaction `T2` rolls back after statement `S3` and does not re-execute. What are Buster's possible final salaries?

4. **Indexes** (5 points)

Consider the following simplified version of the movie-ratings database from Problem 1:

```
Movie(title, director)       // title is a key
Rating(person, title, score) // <person,title> is a minimal key
```

Suppose there are three types of queries commonly asked on this schema:

- Given a movie title, find the director of the movie.
- Match each person with the directors of movies the person has rated.
- Given a person, find the titles of all movies the person has rated.

Here's the actual problem:

(a) **(2 points)** What is the minimum number of indexes needed to speed up all three types of queries? (Do not assume indexes are built automatically on keys.)

(b) **(3 points)** On which attributes should these indexes be created?

5. **Authorization** (15 points, 5 for each part)

Consider the following tables in a database conforming to the SQL standard:

```
Student(ID, name, office, GPA)
Major(ID, dept)
```

Let `ID` be a key for table `Student`, let the pair `<ID,dept>` be a key for table `Major`, and assume no attributes are permitted to be `null`.

The owner (creator) of these tables is a user named *Hennessy*.

(a) Hennessy wants to grant to a user named *Plummer* the ability to read all attributes in the `Student` relation, as well as modify the `office` attribute, for all students with at least one major containing the string “Engineering” (and only those students). Is it possible to specify a command or sequence of commands that achieves this goal? If so, show it. If not, explain why not. Make sure to adhere to the SQL standard.

(problem continues on next page)

(b) Hennessy further wants to grant to a user named *Etchemendy* the ability to read all attributes in the `Student` relation, as well as modify the `office` attribute, for all students whose GPA is the highest among all students in the database. Is it possible to specify a command or sequence of commands that achieves this goal? If so, show it. If not, explain why not. Make sure to adhere to the SQL standard.

(c) Finally, Hennessy wants to grant to a user named *Sahami* the ability read the `ID`s of those students who are majoring in "CS", and to add CS as a student's major (presuming the student is already in the database but not majoring in CS). Is it possible to specify a command or sequence of commands that achieves this goal? If so, show it. If not, explain why not. Make sure to adhere to the SQL standard.

6. **More Authorization** (5 points)

Consider a table `T(A,B,C)` with owner `Amy`, and the following sequence of statements related to privileges on `T`. Each statement is prefaced with the user issuing it.

```
Amy:   Grant Select, Delete On T To Bob With Grant Option
Amy:   Grant Select, Delete On T To Carol With Grant Option
Bob:   Grant Select(A,B), Delete on T to David With Grant Option
Carol: Grant Select(A,C) On T To David With Grant Option
David: Grant Select(A), Delete on T to Eve
Amy:   Revoke Select, Delete on T From Bob Cascade
```

What privileges on table `T` does `Eve` have after this sequence of statements?

7. **Recursion** (10 points, 5 for each part)

(a) Consider a table `T(A,L)` that initially contains a single tuple $\{$`(1,1)`$\}$, and the following query in SQL-99:

```
With Recursive F(A,L) As
 ( Select A,L From T
   Union
   Select A*(L+1), L+1 From F
   Where L < 6 )
Select Max(A) From F
```

What is the result of the query?

(b) Consider a table `T(A)` that initially contains three tuples $\{$`(1)`, `(2)`, `(3)`$\}$, and the following query in SQL-99:

```
With Recursive F(A) As
 ( Select A From T
   Union
   Select A From F
   Union
   Select Sum(A) From F F1
   Where (Select Count(*) From F F2 Where F2.A > F1.A) <= 1
   And (Select Count(*) From F) < 8 )
Select Max(A) From F
```

What is the result of the query?

1-point bonus: What does "F" stands for in each of the above queries?
(-1 point for obscenities!)

(a)

(b)

8. **OLAP** (15 points; 5 per part)

Consider a *fact table* in an OLAP application: `Sales(store, item, color, price)`. Suppose:

- There are two stores, four items, and three colors.
- There are no `null` values in the table.
- Every store has sold every item in every color.

(a) How many tuples are in the result of the following query?

```
Select store, item, color, Sum(price)
From Sales
Group By store, item, color With Cube
```

(b) How many tuples are in the result of the following query?

```
Select store, item, color, Sum(price)
From Sales
Group By store, item, color With Rollup
```

(c) Now suppose we create materialized views from the queries in parts (a) and (b):

```
Create Materialized View VCube as
  Select store, item, color, Sum(price) as p
  From Sales
  Group By store, item, color With Cube

Create Materialized View VRollup as
  Select store, item, color, Sum(price) as p
  From Sales
  Group By store, item, color With Rollup
```

(problem continues on the next page)

Consider the following seven queries, meant to compute the total iPod sales:

```
Q1: Select Sum(price)
    From Sales
    Where item = ’iPod’

Q2: Select Sum(p)
    From VCube
    Where item = ’iPod’

Q3: Select Sum(p)
    From VRollup
    Where item = ’iPod’

Q4: Select Sum(p)
    From VCube
    Where item = ’iPod’ and store is null and color is null

Q5: Select Sum(p)
    From VRollup
    Where item = ’iPod’ and store is null and color is null

Q6: Select Sum(p)
    From VCube
    Where item = ’iPod’ and store is not null and color is not null

Q7: Select Sum(p)
    From VRollup
    Where item = ’iPod’ and store is not null and color is not null
```

Your job is to divide these queries into “equivalence classes”. That is, partition the seven queries into groups such that:

- All queries within each group are equivalent—they return the same answer on every database satisfying the conditions at the beginning of the problem.
- All queries in different groups are not equivalent—there is some database satisfying the conditions at the beginning of the problem such that the two queries return a different answer.

Specify the groups here, being very clear about which queries constitute each group:

9. **Data Mining** (12 points; 4 per part)

Consider the following *market basket* data, represented in a relation as discussed in class.

TransID	**Item**
1	a
1	b
1	c
2	b
2	c
2	d

We are interested in finding *association rules* on this data, restricting ourselves to rules that have exactly one item on the left-hand side, and exactly one different item on the right-hand side. We consider the *support* and *confidence* of association rules as defined in class.

(a) How many such rules are there with *support* > 0.6 and *confidence* > 0.6?

(b) How many such rules are there with *support* > 0.6 and *confidence* < 0.6?

(c) How many such rules are there with *support* < 0.6 and *confidence* > 0.6?

10. **Emerging Trends** (8 points; 2 per part)

Each of the first three questions can be answered correctly in a few words or less.

(a) According to Kevin Weil, if a memory reference is equivalent to walking to one's refrigerator for a snack, then a disk seek is equivalent to what?

(b) During the 2010 soccer world cup, the scoring of some goals produced a per-second tweet rate higher than Twitter had dealt with before. In fact, it uncovered a basic limitation of the way tweets were being stored. What was that limitation?

(c) Name one fundamental difference between the the social graph managed by Twitter versus the one managed by Facebook.

(d) Which of the following systems simultaneously address all of the data management challenges faced by Twitter (and others)—more parallelism, more flexible schemas, more control of memory versus disk, high write and read throughput, and better cluster management? Circle all that apply:

- MySQL with Memcached
- Cassandra
- FlockDB
- Hadoop
- None of the above