

UCLA
Computer Science Department
WINTER 2002

Instr: C. Zaniolo

Student Name and ID: _____

CS240A MIDTERM: 2 Hours–Open Book

*Attach extra pages as needed. Write your name and ID on the extra pages.
Please, write neatly.*

Problem	Score	
1	(30%)	
2	(42%)	
3	(28%)	
Total	(100%)	

Extra Credit:

Exam Score:

Problem 1: 30 points

We have transaction-time relation:

EMP(Name,	Salary,	Title,	DoB,	From,	To)
JoeDoe	30K	machinist	1/1/45	1990-1-1	1994-1-1
JoeDoe	40K	machinist	1/1/45	1998-1-1	UC

which only contains two tuples, where UC means until-changed.

1. Write a TSQL2 schema for this transaction-time relation.

```
CREATE TABLE EMP(Name CHAR(30), Salary NUMERIC(12,2),
                  Title CHAR(30), DoB DATE)
AS TRANSACTION
```

2. Say that JoeDoe's salary is increased today by 20% and we update the database immediately to reflect such a change. Write the TSQL2 statement for such an update.

```
UPDATE EMP SET Salary TO 1.2*Salary
WHERE Name='JoeDoe'
```

3. Show the content of the database after this TSQL2 statement is executed.

EMP(Name,	Salary,	Title,	DoB,	From,	To)
JoeDoe	30K	machinist	1/1/45	1990-1-1	1994-1-1
JoeDoe	40K	machinist	1/1/45	1998-1-1	2002-2-25
JoeDoe	40K	machinist	1/1/45	2002-2-26	UC

Problem 2: 42 Points

Consider the employee relation `empl(ENo, Sal, SupNo)`, which describes employees, by their employee number (ENo), their salary, and their supervisor ENo (i.e., SupNo). This is a regular management hierarchy, where a supervisor can also have a supervisor.

You must use active rules to enforce the constraint that no employee can make a higher salary than his/her supervisor. Assume that the constraint holds in the current state. Ignore inserts and deletes, and concentrate on update requesting changes on salaries (which can be raised or lowered). Whenever a request would result in $SE > SS$ (where SE is the salary of the employee and SS that of his/her supervisor) your active rules must assure that both salaries are instead set to SS (i.e., the lower salary). You must consider both the situations of employees who supervise and do not supervise others.

- A. Describe whether BEFORE or AFTER rules can be used to implement this constraint-enforcement policy (explain the reasons for your conclusion).
- B. Write such rules (write both BEFORE and AFTER rules if they can both solve the problem)
- C. Do your rules always terminate? Are they deterministic (even in situations where the update statement involves several records)? Please, Explain!

A. There are two cases:

A1: The salary of employee e is increased above that of e 's supervisor. In this case, we can use BEFORE rules to revise the update on e so that his salary is now that of e .

A2: The salary of a supervisor is decreased to SS : if SS is below that of his employee that salary must be reset to SS too. To do this, we must use AFTER rules (BEFORE rules can modify the old update request, but not issue new update actions).

```
B1: CREATE TRIGGER IncSaL
      BEFORE UPDATE OF Salary ON emp
      REFERENCING NEW AS N
      FOR EACH ROW
      WHEN N.Salary > (SELECT Salary FROM emp
                      WHERE ENo=N.SupNo)
      SET Salary = (SELECT Salary FROM emp
                   WHERE ENo=N.SupNo)
```

```
B2: CREATE TRIGGER DecSaL
      AFTER UPDATE OF Salary ON emp
      REFERENCING NEW AS N
      FOR EACH ROW
      WHEN N.Salary < (SELECT Salary FROM emp AS E
                      WHERE E.ENo=N.SupNo)
      UPDATE emp SET Salary = N.Salary
                 WHERE emp.SupNo= N.ENo)
```

C. Rule IncSal does not propagate any action. Rule DecSal can cascade updates from a supervisor to his/her employees. At worst, the propagation stop with the bottom level employees. Thus the rules always terminate, because there is no cycle in the management hierarchy.

DecSal is Not deterministic when multiple records are updated. For instance, consider a 10% cut to everyone in the company, where a particular employee makes 90k and his supervisor makes 100k. If the supervisor is updated first, then the rule is triggered reducing both salaries to 90. If the employee is updated first, his salary becomes 85.5, and the subsequent update of the supervisor's salary to 90 triggers no rule.

Problem 3: 28 Points

In our database, we store facts such as `dependt(joedoe, [mary, tom, ann])`; this shows that Joe Doe has three children: Mary, Tom and Ann.

1. Write a Datalog program `?morechildren(JoeDoe, X)` to return the names of employee who have more children than Joe Doe (suggestion: use a recursive predicate that determines the length of a list).
2. Explain how this program would be compiled by a deductive database system.

```
morechildren($JD, X)
```

```

morechildren(J, X) ← dependent(J, LJ), dependent(X, LX),
                    length(LJ, 0, FJ), length(LX, 0, FX), FX > FJ.
length([_|T], C, FC) ← C1 = C + 1, length(T, C1, FC).
                    length([], FC, FC).

```

We can compile `length` with the first two arguments bound using the magic set method:

```

m.length($LX, 0).
m.length(T, C1) ← m.length([_|T], C), C1 = C + 1.
length([_|T], C, FC) ← C1 = C + 1, length(T, C1, FC), m.length(T, C).
length([], FC, FC) ← m.length([], FC).

```

Then both `m.length` and `length` are compiled with the differential fixpoint (for linear rules).

Extra Credit Problem—5 points

The switch from the Julian to the Gregorian calendar in 1582 required a jump ahead to make up for a 10 day difference between the two calendars. As today, how many days of difference are there between the two calendars? Justify your conclusions by listing every year after 1582 which had a different number of days according to the two calendars, and explain the reasons for such a difference.

In the Julian calendar, every year divisible by 4 is a leap year. The Gregorian calendar is basically the same, BUT

- years that are divisible by 100 but not by 400 are NOT leap years,
- years that are divisible by 400 are leap years (as in the Julian calendar).

So after 1582, we had 1700, 1800, and 1900 that were divisible by 100 but not by 400. So, the Gregorian calendar did not count them as leap years; Julian calendar did, and fell behind by three more days, for a current lag of $10+3=13$ days. Years 1600 and 2000 are divisible by 400, so they were leap years in both calendars.