CS240A MIDTERM EXAM: Open Book, 2 Hours

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

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(32%)</td>
</tr>
<tr>
<td>2</td>
<td>(30%)</td>
</tr>
<tr>
<td>3</td>
<td>(24%)</td>
</tr>
<tr>
<td>4</td>
<td>(14%)</td>
</tr>
<tr>
<td>Total</td>
<td>(100%)</td>
</tr>
</tbody>
</table>

Extra credit: 8%

Adjusted Total:
Problem 1: 32 Points  We have the following tables:

\[
\text{Emp}(\text{Eno INT, Level INT, Sal NUMERIC});
\]

\[
\text{SixMonths}(\text{Level INT, Sal NUMERIC, Hired DATE})
\]

The \text{SixMonths} table contains the salaries of people hired in the previous six months. This concrete view is used to control salaries of new people hired at the same level. Write active DB2 rules to do the following when an employee is hired (i.e., a new tuple is added to \text{Emp}):

1. if the salary for the new employee is more than 5% above the maximum salary of employees hired at the same level during the last six months, hire instead this person at the next level (\text{Level} = \text{Level}+1), and keep the original salary,

   \[
   \text{CREATE TRIGGER FixLevel}
   \text{BEFORE INSERT ON Emp REFERENCING NEW AS Nw}
   \text{FOR EACH ROW}
   \text{WHEN Nw.Sal} > 1.05 \times ( \text{SELECT MAX(Sal) FROM SixMonths}
   \text{WHERE Level= Nw.Level AND}
   \text{Hired} + 180 \text{ Days > Current_Date})
   \text{SET Level= Level+1}
   \]

   \text{MAX(Sal) could be replaced with Sal–existential.}

2. remove all records of employees hired more than six months ago:

   \[
   \text{CREATE TRIGGER CleanOld AFTER INSERT ON Emp FOR EACH STATEMENT}
   \text{delete from SixMonths}
   \text{WHERE Hired} + 180 \text{ Days < Current_Date}
   \]

3. store in \text{SixMonths} the new employee’s salary, level and today’s date,

   \[
   \text{CREATE TRIGGER UpdateView}
   \text{AFTER INSERT ON Emp REFERENCING NEW AS Nw}
   \text{FOR EACH ROW}
   \text{INSERT INTO SixMonths}
   \text{VALUES(Nw.Level, Nw.Sal, Current_Date)}
   \]

4. Explain the order in which your rules will be activated and any recursive firing that your rules can experience.

   \text{Answer: Since it is a BEFORE rule, FixLevel will execute first (thus we need a condition that excludes expired tuples). Then, UpdateView executes next, and CleanOld executes last because of the statement level semantics. Either, row-level or statement-level semantics could be used for UpdateView and CleanOld; but FixLevel should be a BEFORE rule and the other two must be AFTER rules.}
Extra Credit: 8 Points. If SixMonths is only used to compute the top salary for each level over a six month window, how can we make its maintenance more efficient? Rewrite the previous rules for improved performance (hint: do we need to keep all the records of new hires for the last six months?). If Answer Tuple $A$ dominates tuple $B$, if $A.SAL >= B.SAL$ and $A$ is more recent. When $A$ is inserted all the tuples in SixMonths dominated by $A$ can be eliminated. On the average the size of SixMonths is reduced to the log of the previous file.

CREATE TRIGGER CleanOld_revised
AFTER INSERT ON Emp
FOR EACH STATEMENT
REFERENCING NEW AS Nw
delete from SixMonths
WHERE (Hired + 180 Days< Current_Date) OR
(Emp.Sal < Nw.Sal AND Emp.Level=Nw.Level)
Problem 2: 30 Points These two relations store the history of employees and departments they worked for. Thus, From—To is the period of validity for a tuple (granularity days).

\[
\text{empl}(\text{Eno}, \text{Sal}, \text{DeptNo}, \text{From}, \text{To}); \quad \text{dept}(\text{DeptNo}, \text{Manager}, \text{From}, \text{To})
\]

We also know that 'Joe Doe' was the manager of department 201 for several nonconsecutive periods.

1. Which of the next two queries is monotonic (with respect to set containment: relations are sets of tuples) and can thus be written without negation (this observation could simplify the coalescing operations in your queries)?

**Answer:** The first query is monotonic with respect to both \( \text{empl} \) and \( \text{dept} \): the addition of tuples to these two relations only increases (w.r.t. set containment) the original answer. Thus we can write the next query without negation—i.e., without eliminating periods which are contained in other periods. The second query is not monotonic.

2. Write Datalog rules to find employees who worked in department 201 for at least one complete period in which Joe Doe was managing that department.

\[
\begin{align*}
\text{coal}(\text{Eno}, \text{From}, \text{To}) & \leftarrow \text{empl}(\text{Eno}, ., 201, \text{From}, \text{To}). \\
\text{coal}(\text{Eno}, \text{From1}, \text{To2}) & \leftarrow \text{coal}(\text{Eno}, \text{From1}, \text{To1}), \text{coal}(\text{Eno}, \text{From2}, \text{To2}), \\
& \text{From1} <\text{=} \text{From2}, \text{From2} <\text{=} \text{To1}, \text{To1} <\text{=} \text{To2}. \\
\text{someperiod}(\text{Eno}, \text{DFrom}, \text{DTO}) & \leftarrow \text{dept}(201, \text{JoeDoe}, \text{DFrom}, \text{DTO}), \text{coal}(\text{Eno}, \text{From}, \text{To}), \\
& \text{From} <\text{=} \text{DFrom}, \text{DTO} <\text{=} \text{To}.
\end{align*}
\]

This return the employees who worked in dept 201 for a complete period in which Joe Doe was the manager of 201—along with such a period.

3. Write Datalog rules to find employees who worked in department 201 whenever Joe Doe was managing that department (i.e., during all periods in which Joe Doe was the manager of 201, and for the complete durations of those periods).

**Answer:** First find the employees \( E \) that did not cover some of the periods in which Joe Doe was a manager.

\[
\begin{align*}
\text{missing}(\text{Eno}) & \leftarrow \text{dept}(201, \text{JoeDoe}, \text{DFrom}, \text{DTO}), \text{empl}(\text{Eno}, ., 201, .), \\
& \neg\text{someperiod}(\text{Eno}, \text{DFrom}, \text{DTO}). \\
\text{coverall}(\text{Eno}) & \leftarrow \text{empl}(\text{Eno}, ., 201, .), \neg\text{missing}(\text{E}).
\end{align*}
\]
Problem 3: 24 points

1. Write a TSQL2 schema for the example of Problem 2 (where From—To is a valid-time period):

   CREATE TABLE empl(Eno integer, Sal integer, DeptNo integer) AS VALID STATE
   CREATE TABLE dept(DeptNo integer, Manager char(16)) AS VALID STATE

2. Express the first query of Problem 2 in TSQL2:

   SELECT SNAPSHOT E.Eno
   FROM empl(Eno, DeptNo) AS E,
   dept(DeptNo, Manager) AS D
   WHERE E.DeptNo=201 AND D.DeptNo=201 AND D.Manager=’Joe Doe’ AND
   VALID(E) CONTAINS VALID(D)

3. Express the second query of Problem 2 in TSQL2.

   SELECT SNAPSHOT E.Eno
   FROM empl(Eno, DeptNo) AS E,
   dept(DeptNo, Manager) AS D
   WHERE E.DeptNo=201 AND D.DeptNo=201 AND D.Manager=’Joe Doe’ AND
   VALID(D) CONTAINS VALID(E)
Problem 4: 14 Points

1. How many sort operations are needed to support a sort-based computation of \( \text{sum}(\text{Sales}) \) on the following table:

\[
\text{factable}(\text{Time, Location, Product, Retailer, CustomerType, Sales})
\]

\textbf{Answer}: 

How many 4-element subsets of 5? : \( \binom{5}{4} = 5 \)

How many 3-element subsets of 5? : \( \binom{5}{3} = \frac{5 \times 4 \times 3}{3 \times 2 \times 1} = 10 \)

How many 2-element subsets of 5? : \( \binom{5}{2} = \frac{5 \times 4}{2 \times 1} = 10 \)

How many 1-element subsets of 5? : \( \binom{5}{1} = 5 \)

The answer is the max of these, i.e., 10.

2. How many sort operations are needed if you only want \( \text{sum}(\text{Sales}) \) for the three following grouping sets:

1. \( (\text{Time, Location, Product, Retailer}) \)
2. \( (\text{Product, Time, Location}) \)
3. \( (\text{Product, Time, Retailer}) \)

\textbf{Answer}: Obviously you can sort by \( (\text{Product, Time, Location, Retailer}) \), and that takes care of the first two. You need a second sort for the third. Symmetrically if you switch location and retailer.