CS 240A : Databases and Knowledge Base
Homework #4

Ronak Sumbaly
UID : 604591897

November 2, 2015

Question 1

Was February 30 ever used in some country's calendar? When, where and why did that happen?

Solution.

February 30 was a real date in just one country's calendar. It was added to the 1712 calendar of **Sweden** due to an calendar error that occurred in the previous calendar.
In an attempt to convert from Julian calendar to Gregorian calendar Sweden didn't account for the fact that it was a leap year. However, 1704 and 1708 became leap years due to this error which caused Sweden to be behind in both the Julian and Gregorian calendars.
To overcome this Sweden reverted to the Julian calendar but to compensate for the 2 leap days February 30 was added in the year 1712.

Question 2

With the actual solar year being 365.24219878 days, which of these three calendars is the most accurate:

1. Julian,
2. Gregorian, or
3. Jalaali (8 leaps every 33 years)?

Solution.

Calculating the days present in each of the calendar we get,

1. Julian $= (365 * 4 + 1)/4 = 365.2500$ - 1 leap every 4 years
2. Gregorian $= (400 * 365 + 97)/400 = 365.2425$ - 97 leaps every 400 years
3. Jalaali (8 leaps every 33 years) $= (33 * 365 + 8)/33 = 365.2424$ - 8 leaps every 33 years

Comparing the difference between each calendar and the actual solar days we can see that **Jalaali** is the most accurate calendar.

Question 3

Is "honey moon" a celestial body or a month–and what is origin of the expression?

Solution.

The term "honey moon" refers to the full moon in the month of June. In pagan times, weddings in June were celebrated by drinking a beverage made from Honey. Hence the June moon is called as the "Honey moon".

Question 4

Do exercises 6.1, 6.3 from the ADS textbook. (Errata: for Exercise 6.3,the current tuple contains an additional Dosage attribute with value 100).

Exercise 6.1

Part A.

```
CREATE TABLE Employee (Name CHAR(30), Salary NUMERIC (10), Title CHAR(30),
DateOfBirth DATE) AS VALID DAY AND TRANSACTION
```

Part B.

1. SELECT MAX(Salary) FROM Employee

2. SELECT AVG(Salary) FROM Employee

Exercise 6.3

Effects of the **UPDATE** statement

Name	Drug	Dosage	Valid Time	Transaction Time
Melanie	Proventil	100 mg	[1996-01-01 - 1996-08-31]	[1996-06-01 -1996-09-15]
Melanie	Proventil	100 mg	[1996-01-01 - 1996-02-29]	[1996-09-15 - Until Changed]
Melanie	Proventil	50 mg	[1996-03-01 - 1996-05-30]	[1996-09-15 - Until Changed]
Melanie	Proventil	100 mg	[1996-06-01 - 1996-08-31]	[1996-09-15 - Until Changed]

Question 5

Give a simpler SQL expression for the temporal joins of ADS example 5.9. (Hint: In order to derive a simpler expression for OVERLAP start by negating that period P1 precedes P2 or vice-versa; then apply DeMorgan's rule. Then use the SQL CASE construct to express max and min).

```
SELECT DISTINCT Emp1.Name, Emp1.Salary,
        CASE WHEN (Emp1.Start < Emp2.Start) THEN Emp2.Start
                ELSE Emp1.Start
        END AS StartD,

        CASE WHEN (Emp1.Stop > Emp2.Stop) THEN Emp2.Stop
                ELSE Emp2.Stop
        END AS StopD
FROM Employee Emp1, Employee Emp2
WHERE Emp1.Name = Emp2.Name and Emp1.Stop > Emp2.Start and Emp2.Stop > Emp1.Start and
Emp1.Salary = Emp2.Salary
```

Question 6

Write an test on a Datalog system rules to coalesce the periods after Sal is projected out from EHist(Eno, Sal, Title, From, To)

Coalesce.fac

```
ehist(9711, 60000, 'Assistant Provost', 19950101,19950601)
ehist(9711, 70000, 'Assistant Provost', 19950601,19951001)
ehist(9711, 70000, 'Provost', 19951001,19960201)
ehist(9711, 70000, 'Professor', 19960201,19970101)
```

Coalesce.deal

```
database(ehist(Eno:integer,Salary:integer,Title:string,Start:integer,Stop:integer)).

getNameSalary(Eno,Salary) <- ehist(Eno,Salary,_,_,_).
export getNameSalary(A,B).

getDatesforSalary(Eno,Salary,Start,Stop) <- getNameSalary(Eno,Salary),
ehist(Eno,Salary,_,Start,Stop).

export getDatesforSalary(A,B,C,D).

coalesceSal(Eno,Salary,Start,Stop) <- getDatesforSalary(Eno,Salary,Start,Stop),
∼ getDatesforSalary(Eno,Salary,Stop,Stop2).

coalesceSal(Eno,Salary,Start1,Stop2) <- getDatesforSalary(Eno,Salary,Start1,Stop1),
coalesceSal(Eno,Salary,Stop1,Stop2).

export coalesceSal(A,B,C,D).

removeExtra(Eno,Salary,Start2,Stop) <- coalesceSal(Eno,Salary,Start1,Stop),
coalesceSal(Eno,Salary,Start2,Stop), Start2 > Start1.

export removeExtra(A,B,C,D).

finalOutput(Eno,Salary,Start,Stop) <- coalesceSal(Eno,Salary,Start,Stop),
∼ removeExtra(Eno,Salary,Start,Stop).

export finalOutput(A,B,C,D).
```

Output

```
Query: finalOutput(A,B,C,D)

finalOutput(9711, 60000, 19950101,19950601).
finalOutput(9711, 70000, 19950601,19970101).
```

Question 7

Solve the coalesce problem in DB2. Test your solution.

A. Create ehist table in DB2.

```
db2 CREATE TABLE CS240A.ehist(Eno int, Salary int, from date, to date)
```

B. Insert Values into ehist table

```
db2 INSERT INTO CS240A.ehist VALUES(9711, 60000, ’01/01/1995’, ’06/01/1995’);
db2 INSERT INTO CS240A.ehist VALUES(9711, 70000, ’06/01/1995’, ’10/01/1995’);
db2 INSERT INTO CS240A.ehist VALUES(9711, 70000, ’10/01/1995’, ’02/01/1996’);
db2 INSERT INTO CS240A.ehist VALUES(9711, 70000, ’02/01/1996’, ’01/01/1997’);
```

C. Coalescing Query

```
with coal(Eno, Salary, Start, Stop, CountOcc) As
((select Eno, Salary, from, to, 0 from CS240A.ehist)

union all

(select coal.Eno, coal.Salary, Start, to, CountOcc+1
from coal, CS240A.ehist
where CountOcc<5 and CS240A.ehist.eno = coal.eno and CS240A.ehist.Salary=coal.Salary and
Start <= from and from <= Stop and Stop < to))
select distinct * from coal;

with coal(Eno, Salary, Start, Stop, CountOcc) As
((select Eno, Salary, from, to, 0 from CS240A.ehist)
union all
(select coal.Eno, coal.Salary, Start, to, CountOcc+1
from coal, CS240A.ehist
where CountOcc<5 and CS240A.ehist.eno = coal.eno and CS240A.ehist.Salary=coal.Salary and
Start <= from and from <= Stop and Stop < to))

select distinct * from coal
where not exists (select * from coal as C
where
C.eno=coal.eno and C.Salary=coal.Salary
and C.Start <= Coal.Start and C.Stop>=coal.Stop
and (C.Start <Coal.Start or C.Stop>coal.Stop));
```

D. Output

```
Eno    Salary   Start        Stop         CountOcc
9700   60000    01/01/1995   06/01/1995   0
9700   70000    06/01/1995   01/01/1997   0
```

Question 8

Now EHist(Eno, Sal, Title, From, To)is a concrete view that stores the transaction time history for the relation EMP(Eno, Sal, Title). The concrete view must be maintained by active DB2 rules. Please write those rules (testing optional).

```
CREATE TRIGGER HireEmployee
AFTER INSERT ON EMP
FOR EACH ROW
INSERT INTO EHist VALUES(Eno, Sal, Title, CURRENTDATE, Null)

CREATE TRIGGER FireEmployee
AFTER DELETE ON EMP
FOR EACH ROW
UPDATE EHist SET To = CURRENTDATE
WHERE Ehist.Eno = OLD.Eno AND EHist.To = Null

CREATE TRIGGER ChangeEmployeeUp
AFTER UPDATE ON EMP
FOR EACH ROW
UPDATE EHist SET To = CURRENTDATA
WHERE EHist.Eno = OLD.Eno AND EHist.To = Null

CREATE TRIGGER ChangeEmployeeIn
AFTER UPDATE ON EMP
FOR EACH ROW
INSERT INTO EHist VALUES(Eno, Sal, Title, CURRENTDATE, Null)
```