

Course ISE 322: Database Systems

Recitation 13: Triggers and Stored Procedures

Michael J. May

January 13, 2009

In the following exercise we will work on the following database schema:

Product (maker STRING, model INT, type STRING)
PC (model INT, speed REAL, ram INT, hd INT, price INT)
Laptop (model INT, speed REAL, ram INT, hd INT, screen REAL, price INT)
Printer (model INT, color STRING, type STRING, price INT)

The SQL to create the tables and sample data for them are provided in an Excel worksheet and text file on the course web site.

1 Stored Procedures

Write three stored procedures to perform INSERTs into the above tables:

1. CREATE PROCEDURE addPC (@model, @maker, @speed, @ram, @hd, @price)
2. CREATE PROCEDURE addLaptop (@model, @maker, @speed, @ram, @hd, @screen, @price)
3. CREATE PROCEDURE addPrinter (@model, @maker, @color, @type, @price)

Next, write a stored procedure to show the maker, model, type, and price of all Products with price > +@price:

- CREATE PROCEDURE showExpensive (@price)

2 Triggers

Write the following as triggers. In each case, disallow or undo the modification if it does not satisfy the stated constraint. The database schema is as above.

1. When updating the price of a PC, check that there is no lower priced PC with the same speed.
2. When inserting a new printer, check that the model number already appears in Product as a Printer
3. When making any modification to the Laptop table, check that the average price of laptops for each maker is at least 1500
4. When updating the RAM or hard disk of any PC, check that the updated PC has at least 100 times as much hard disk as RAM

3 Instead Of Example

Let's consider another example trigger to better understand how "Instead of" triggers work.

Consider the schema:

```
Emp (eid INT, ename STRING, city STRING)
HourlyEmp (eid INT, hoursWorked INT, hourlyRate INT)
ContractEmp (eid INT, monthlySal INT)
```

Consider the following view:

```
CREATE VIEW MonthlyPay (eid, ename, city, sal) AS
  SELECT E1.eid, E1.ename, E1.city, H.hoursWorked * H.hourlyRate
  FROM Emp E1, HourlyEmp H
  WHERE E1.eid = H.eid

UNION

  SELECT E2.eid, E2.ename, E2.city, C.monthlySal
  FROM Emp E2, ContractEmp C
  WHERE E2.eid = C.eid
```

If somebody would want to indicate that employee 107 moved to Tiberias using the following update:

```
UPDATE MonthlyPay SET city = 'Tiberias' WHERE eid = 107
```

We would get an immediate error message. It's not possible to edit a field in a view such as this. So how can we allow updates to the city field using just the view?

3.1 Instead of Trigger

Let's write a trigger which will capture the intended action and actually do it on Emp.

```
CREATE TRIGGER updateCity
ON MonthlyPay INSTEAD OF UPDATE AS
IF UPDATE (city)
BEGIN
  DECLARE @newcit CHAR(20)
  DECLARE @eid INT

  SELECT @eid = (SELECT eid FROM inserted)
  SELECT @newcit = (SELECT city FROM inserted)

  UPDATE Emp SET city = @newcit WHERE eid = @eid
END
```

Now attempting to run the above update command, we find that it works and that the table Emp is updated with the new city information.

3.2 What to do

Now it's your turn to write INSTEAD OF triggers.

1. (Easy) Adapt the above trigger updateCity to also allow the changing of names. Call the new trigger updateName.

2. (Harder) Adapt the above trigger `updateCity` to also allow the updating of salaries. For contract employees, update the `monthlySal` value. For hourly employees, updating the `hourlyRate` value (if the employee worked 10 hours and is paid 100 per hour = 1000 and the raise is to 2000, change the `hourlyRate` to 200). Call the new trigger `updateSal`.

Note: In order to define the above triggers you will need to drop the first one. You can not define more than one INSTEAD OF Update trigger on a single relation. (Why?)