

# Course ISE 322: Database Systems

## Recitation 12: Triggers and Stored Procedures

Michael J. May

January 6, 2009

The recitation today is based on the following schema:

```
Student(snum:integer, sname:string, major:string, level:string, age:integer)
Class(name:string, meets_at:time, room:string, fid:integer, targilOf:string)
      Enrolled(snum:integer, cname:string)
Faculty (fid:integer, fname:string, deptid:integer)
```

Classes are identified by names. There are two kinds of classes - Lecture classes and Targil classes. They are connected via a simple foreign key constraint and have the following rules:

- All Lecture classes have the “targilOf” field NULL - they are not the Targil session for any course
- All Targil classes have a name in the “targilOf” field - the name of the Lecture they are Targil for.
- Every Lecture may have zero or one Targil sessions associated (not more than 1).
- No Targil session is a targil for more than one Lecture.

## 1 Triggers

In this section you will write a series of triggers to support the following requirement.

### 1.1 A First Trigger

Write a trigger for the following constraint:

When a student is enrolled in a new course (a new row is inserted into Enrolled), the student should be automatically enrolled in the associated Targil class (inserting a new row into Enrolled) if not already enrolled in the Targil.

As a start, you may start assuming that only one row at a time is inserted into Enrolled.

Test it on the following insert statements:

1. `insert into Enrolled (snum, cname) values (51135593, 'Introduction to Math')`
2. `insert into Enrolled (snum, cname)
SELECT snum, 'Introduction to Math'
FROM Student S
WHERE S.sname LIKE 'D%'`

What happens?

## 1.2 Updating the Trigger

Now, attempt to update the trigger to deal with multiple inserts into Enrolled at once (assuming that we are turning the trigger into a Statement level trigger instead of a Row level trigger).

Try running the above inserts again to see what happens.

## 2 Stored Procedures

Now we will write a few simple stored procedures using the above schema.

1. Write a stored procedure which will register all students of a given level to a class. This stored procedure would be useful to register all of the third year ISE students for the 'Database Systems' class, for example. The procedure should have the following signature:

enrollLevel (@level VARCHAR(2), @classname VARCHAR(40))

It should add rows in Enroll for each student in the level. Make sure that you don't attempt to register a student for a course twice or you will get an error when using the function.

2. Another common task for stored procedures is to create **parameterized views**. You will write one now.

Write a stored procedure which takes a Department number as a parameter and shows a table showing information on each class taught by a faculty member in the department. The table should have the name of the faculty member, the name of the class, and the number of students enrolled. For example, for department id 68 we should get:

fname	name	numEnrolled
John Williams	Air Quality Engineering	1
Patricia Jones	Optical Electronics	1

The signature of the stored procedure should be:

departmentClasses (@deptid INT)

**Hint:** For this one you might want to start by writing the SQL code to show the view just for deptid 68 and then using that to write the stored procedure, replacing 68 with @deptid.