Elizabeth Deeney

Jon Olson

Documentation

Stored Procedures

Quick View of Procedures (Descriptions below):

1. clrSchedule (sid int, curyear int, curterm int)
2. delStudent (sid int)
3. getAdvisors()
4. getCoursesByCode(ccode varchar(10))
5. getCoursesByType(rtype varchar(10))
6. getGenEds()
7. getGenTypes()
8. getMajors()
9. getSchedBySIDForAdvisor(sid int, aid int, apin VARCHAR(12))
10. getSchedBySIDForStudent(sid int, spin VARCHAR(12))
11. getStudentsByAdvisor(aid int(11))
12. putSchedule(cid int, sid int, schyear int, schterm int)
13. putStudent(sname varchar(45), aid int, mid int, spin varchar(12), stheme varchar(45))
14. **setMajor(sid int, newMid int)**

--------------------------------------------------------------------------------------------------------------------

1. Procedure name = clrSchedule

* This stored procedure is for clearing the schedule based off of the student’s ID, the year, and the term. It looks for the matching student ID along with the given year being greater than the schedule year or if the given year is equal to the scheduled year and the term is greater than the scheduled term
* Variables needed
  + Student ID, the current year, and the current term

clrSchedule (sid int, curyear int, curterm int)

BEGIN

delete from `Schedule` where (stu\_id=sid and (sch\_year>curyear or (sch\_year=curyear and sch\_term>curterm)));

END

1. Procedure name = delStudent

* This procedure is for deleting a student and their schedule out of the DB based on student ID. It searches for the given student id to equal the student ID in both the Schedule table and the Student table.
* Variables needed
  + Student ID

delStudent (sid int)

BEGIN

delete from `Schedule` where stu\_id=sid;

delete from Student where stu\_id=sid;

END

1. Procedure name = getAdvisor

* This procedure will get and return all advisor’s id and names from the advisor table
* Variables needed
  + None

getAdvisors()

BEGIN

select A\_ID, A\_Name from Advisor;

END

1. Procedure name = getCourseByCode

* This procedure gets all of the course information based on the given course code. The course code is compared with all of the courses in the course table, and only the courses that contain the given code in the beginning will be chosen, but as long as the beginning part matches, anything can be after the wildcard (%). Once a match is found, the procedure will return the course id, name, code, and credits.
* Variables needed
  + Course code (ie, IT or IT101 ~‘no space’)

getCoursesByCode(ccode varchar(10))

BEGIN

select c\_id, c\_name, c\_code, c\_credit from Course where c\_code like (ccode+'%');

END

1. Procedure name = getCoursesByType

* This procedure will get information about the course that equals the given requirement type. The procedure joins the Course table with the Req table and selects the course id, name, code, and credit based off of the given req type.
* Variable needed
  + Requirement type (ie, SOCSCI, CSKILL ~ ‘no space’)

getCoursesByType(rtype varchar(10))

BEGIN

select Course.c\_id, c\_name, c\_code, c\_credit from (Course inner join Req) where r\_type=rtype;

END

1. Procedure name = getGenEds

* This procedure will select and return all the course ids, name, code, and credit from a joined table of Course and Req that have the r\_isGen set to true.
* Variable needed
  + None

getGenEds()

BEGIN

select Course.c\_id, c\_name, c\_code, c\_credit from (Course inner join Req) where r\_isGen=True;

END

1. Procedure name = getGenTypes

* This selects and returns the distinct (different) req types from the table Req where the r\_isGen is equal to one.
* Variables needed
  + None

getGenTypes()

BEGIN

select distinct r\_type from Req where r\_isGen=1;

END

1. Procedure name = getMajors

* This procedure selects and returns all major id, name, and type from Major table
* Variable needed
  + None

getMajors()

BEGIN

select m\_id, m\_name, r\_type from Major;

END

1. Procedure name = getSchedBySIDForAdvis

* This procedure selects the pin is in the field that matches the advisors id and then compares it to the given advisor pin. If the pins match then it will select and return the course name, code, credit and schedule term and year from the join Schedule and Course tables that have the student id equal to the given student id. If the pin does not match, then the table will not return anything.
* Variables needed
  + Student id, advisor id, advisor pin

getSchedBySIDForAdvisor(sid int, aid int, apin VARCHAR(12))

BEGIN

if apin= (select a\_PIN from Advisor where a\_id=aid) then

select c\_name, c\_code, c\_credit, sch\_year, sch\_term from (`Schedule` inner join Course) where stu\_id=sid;

else select Null;

end if;

END

1. Procedure name = getSchedBySIDForStudent

* This procedure selects the pin is in the field that matches the students id and then compares it to the given student pin. If the pins match then it will select and return the course name, code, credit and schedule term and year from the join Schedule and Course tables that have the student id equal to the given student id. If the pin does not match, then the table will not return anything.
* Variables needed
  + Student id, student pin

getSchedBySIDForStudent(sid int, spin VARCHAR(12))

BEGIN

if apin= (select stu\_PIN from Student where stu\_id=sid) then

select c\_name, c\_code, c\_credit, sch\_year, sch\_term from (`Schedule` inner join Course) where stu\_id=sid;

else select Null;

end if;

END

1. Procedure name = getStudentsByAdvisor

* This procedure selects and returns the student id, name, major id, name, and req type from a joined Student table and Major table where the advisor id is equal to the given advisor id
* Variables needed
  + Advisors id

getStudentsByAdvisor(aid int(11))

BEGIN

select stu\_id, stu\_name, Student.m\_id, m\_name, r\_type from (Student inner join Major) where a\_id=aid;

END

1. Procedure name = putSchedule

* This procedure inserts a record containing the given values of course id, student id, schedule year, and schedule term into the corresponding columns in the Schedule table
* Variables needed
  + Course id, student id, schedule year, schedule term

putSchedule(cid int, sid int, schyear int, schterm int)

BEGIN

insert into `Schedule` (c\_id, stu\_id, sch\_year, sch\_term) values (cid,sid,schyear,schterm);

END

1. Procedure name = putStudent

* This procedure creates a new student into the database. First it will check to see if the student pin is null, and if it is it will set the pin to a default of 000, and if the student theme is null, it will set it to the default theme of Default. Then it inserts the given variables of student name, advisor id, major id, and student pin to the corresponding columns of the Student table.
* Variables needed
  + Student name, advisor name, major name, student pin, student theme

putStudent(sname varchar(45), aid int, mid int, spin varchar(12), stheme varchar(45))

BEGIN

if spin=null then set spin='000'; end if;

if stheme=null then set stheme='Default'; end if;

insert into Student (stu\_name, a\_id, m\_id, s\_pin) values (sname, aid, mid, spin);

END

1. Procedure name = setAdvisor

* This procedure updates a student’s (based off of the given student id) original advisor to the given advisor id.
* Variables needed
  + Student id, advisor id

setAdvisor(stid int(11), newadv int(11))

BEGIN

update Student set Student.A\_ID = newadv where Student.Stu\_ID = stid;

END

1. Procedure name = setMajor

* This procedure updates a student’s (based off of the given student id) original major to the given major id. It then selects and returns the req type from the Major table were the major id equal the new major id.
* Variables needed
  + Student id, major id

setMajor(sid int, newMid int)

BEGIN

update Student set m\_id=newMid where stu\_id=sid;

select r\_type from Major where m\_id=newMid;

END