IS480 FINAL PROJECT

Phuc Cong Le

Email: **phucsaki@gmail.com**
California State University Long Beach
Fall 2013

1. STUDENT INFO

Name: PHUC CONG LE

E-Mail: phucsaki@gmail.com

Student ID: #009587329

```
SQL> desc WAITLIST;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 SNUM                          NOT NULL VARCHAR2(3)
 CALLNUM                       NOT NULL NUMBER(5)
 REQUESTTIME                            TIMESTAMP(6)
```

```
SQL> desc ENROLLMENTS;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 SNUM                          NOT NULL VARCHAR2(3)
 CALLNUM                       NOT NULL NUMBER(5)
 GRADE                                  VARCHAR2(2)
```

```
SQL> desc STUDENTS;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 SNUM                          NOT NULL VARCHAR2(3)
 SNAME                                  VARCHAR2(10)
 STANDING                               NUMBER(1)
 MAJOR                                  VARCHAR2(3)
 GPA                                    NUMBER(2,1)
 MAJOR_GPA                              NUMBER(2,1)
```

```
SQL> desc MAJORS;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 MAJOR                         NOT NULL VARCHAR2(3)
 MDESC                                  VARCHAR2(30)
```

```
SQL> desc SCHCLASSES;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 CALLNUM                       NOT NULL NUMBER(5)
 YEAR                                   NUMBER(4)
 SEMESTER                               VARCHAR2(3)
 DEPT                                   VARCHAR2(3)
 CNUM                                   VARCHAR2(3)
 SECTION                                NUMBER(2)
 CAPACITY                               NUMBER(3)


SQL> desc COURSES;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 DEPT                          NOT NULL VARCHAR2(3)
 CNUM                          NOT NULL VARCHAR2(3)
 CTITLE                                 VARCHAR2(30)
 CRHR                                   NUMBER(3)
 STANDING                               NUMBER(1)


SQL> desc PREREQ;
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 DEPT                          NOT NULL VARCHAR2(3)
 CNUM                          NOT NULL VARCHAR2(3)
 PDEPT                         NOT NULL VARCHAR2(3)
 PCNUM                         NOT NULL VARCHAR2(3)
```

3. TEST DATA

MAJORS

MAJOR	MDESC
ACC	Accounting
FIN	Finance
IS	Information Systems
MKT	Marketing

WAITLIST

SNUM	CALLNUM	REQUES...
no data		

COURSES

DEPT	CNUM	CTITLE	CRHR	STANDING
IS	300	Intro to MIS	3	2
IS	301	Business Communicatons	3	2
IS	310	Statistics	3	2
IS	340	Business Application	3	3
IS	355	Networks	3	3
IS	380	Database	3	3
IS	385	Systems	3	3
IS	480	Adv Database	3	4
FIN	300	Business Finance	3	3
FIN	350	Investment Principles	3	3
ACC	320	Cost Accounting	3	3
ACC	470	Auditing	4	4
MKT	300	Basic Marketing	3	2

PREREQ

DEPT	CNUM	PDEPT	PCNUM
IS	380	IS	300
IS	380	IS	301
IS	380	IS	310
IS	385	IS	310
IS	355	IS	300
IS	480	IS	380
FIN	350	FIN	300
ACC	470	ACC	320

STUDENTS

SNUM	SNAME	STANDING	MAJOR	GPA	MAJOR_GPA
101	Andy	4	IS	2.8	3.2
102	Betty	2	(null)	3.2	(null)
103	Cindy	3	IS	2.5	3.5
104	David	2	FIN	3.3	3
105	Ellen	1	(null)	2.8	(null)
106	Frank	3	MKT	3.1	2.9
107	Jim	2	MKT	3.7	(null)
108	Susan	3	MKT	3.5	3.4
109	Anna	3	FIN	3.4	3.2
110	Jose	4	MKT	3.2	3.5
111	Mary	4	ACC	3.8	(null)
112	Lisa	2	IS	3.5	3.6
113	Alex	2	(null)	3.2	(null)
114	Fillipe	3	IS	3.3	3.3
115	Joven	3	FIN	3.1	3.4
116	Camila	2	(null)	3.8	3.7
117	Mario	2	(null)	3.4	3.6
118	Ian	3	FIN	2.4	3.2
119	Monica	4	FIN	2.7	3
120	Claudia	3	IS	3.9	3.7
121	Kelly	2	ACC	3.4	3.5
122	Rosa	2	MKT	3.1	(null)
123	Sam	2	(null)	3.7	3.5
124	Jason	3	ACC	3.2	(null)

ENROLLMENTS

SNUM	CALLNUM	GRADE
101	10110	(null)
104	10110	(null)
110	10110	(null)
114	10110	(null)
101	10130	(null)
101	10135	(null)
101	10141	(null)
101	10150	(null)

SCHCLASSES

CALLNUM	YEAR	SEMESTER	DEPT	CNUM	SECTION	CAPACITY
10110	2013	Fa	IS	300	1	4
10115	2013	Fa	IS	300	2	4
10120	2013	Fa	IS	380	1	3
10121	2013	Fa	IS	380	2	3
10125	2013	Fa	IS	300	1	5
10130	2013	Fa	IS	301	1	6
10131	2013	Fa	IS	301	2	6
10135	2013	Fa	IS	340	1	5
10141	2013	Fa	FIN	300	1	4
10142	2013	Fa	FIN	300	2	4
10145	2013	Fa	FIN	350	2	5
10150	2013	Fa	ACC	320	1	3
10160	2013	Fa	MKT	300	1	3

```
SQL> Create or Replace Package Enroll is
  2      Procedure AddMe (
  3        p_snum IN Students.snum%type, p_callnum IN SchClasses.callnum%type,
p_ErrorMsg OUT varchar2);
  4
  5      Procedure DropMe (
  6        p_snum Students.snum%type, p_callnum SchClasses.callnum%type);
  7 End Enroll;
  8 /

   Package created.

SQL>
SQL> Create or replace Package Body Enroll is
  2      -- condition #1.1a check for valid snum
  3      Function Func_validate_snum (
  4        p_snum Students.snum%type) return boolean is
  5        v_count number;
  6      BEGIN
  7        select count(snum) into v_count from students where snum=p_snum;
  8        If v_count > 0 Then
  9          return True;
 10        Else
 11          return False;
 12        End If;
 13      END;
 14
 15      -- condition #1.1b check for valid callnum
 16      Function Func_validate_callnum (
 17        p_callnum SchClasses.callnum%type) return boolean is
 18        v_count number;
 19      BEGIN
 20        select count(callnum) into v_count from SchClasses where
callnum=p_callnum;
 21        If v_count > 0 Then
 22          return True;
 23        Else
 24          return False;
 25        End If;
 26      END;
 27
 28      -- condition #1.2 check for Double Enrollment
 29      Function Func_Check_Double_Enroll (
```

```
30       p_snum Students.snum%type, p_callnum SchClasses.callnum%type) return
varchar2 is
31       v_dept SchClasses.DEPT%Type;
32       v_cnum SchClasses.CNUM%Type;
33       v_count number;
34       begin
35       select DEPT into v_dept from SchClasses where callNum=p_callnum;
36       select CNUM into v_cnum from SchClasses where callNum=p_callnum;
37       select count(snum) into v_count from Enrollments e, Schclasses s Where
e.snum=p_snum AND Dept=v_dept AND cnum=v_cnum and e.callnum=s.callnum;
38       If v_count > 0 Then
39         return 'double enrollment error';
40       End If;
41       return null;
42       end;
43
44     -- condition #1.3 check for Undeclared Major
45     Function Func_Check_Undeclared_Major (
46       p_snum Students.snum%type, p_callnum SchClasses.callnum%type) return
varchar2 is
47       v_major Students.Major%Type;
48       v_cnum SchClasses.CNUM%Type;
49       v_cnumInitial SchClasses.CNUM%Type;
50     begin
51       select NVL(Major,'NA') into v_major from Students where snum=p_snum;
52       select CNUM into v_cnum from SchClasses where callNum=p_callnum;
53       v_cnumInitial := substr(v_cnum,1,1);
54       If v_cnumInitial = '3' or v_cnumInitial = '4' Then
55         if v_major = 'NA' then
56           return 'Undelared major student cannot enroll major course';
57         end if;
58       End If;
59       return null;
60     end;
61
62     -- condition #1.4 check for 15-Hour Rule
63     Function Func_Check_15Hour_Rule (
64       p_snum IN Students.snum%type,
65       p_callnum IN SchClasses.callnum%type) return varchar2 is
66       v_unit number;
67       v_Unit_registered number;
68     begin
69       select CRHR into v_unit from courses c, schclasses s where
callnum=p_callnum AND s.dept=c.dept AND s.cnum=c.cnum;
70       /* Sum credit hours of all null-grade classes only, graded class
aren't counted */
```

```
 71       select sum(CRHR) into v_Unit_registered from courses c, schclasses s,
enrollment s e where e.snum=p_snum
 72       AND e.callnum = s.callnum AND s.dept=c.dept AND s.cnum=c.cnum
 and e.grade is null;
 73       -- make deciscion
 74       If v_unit + v_Unit_registered > 15 Then
 75         return '15 units exceeded';
 76       End If;
 77       return null;
 78     end;
 79
 80     -- condition #1.5 check for Standing Requirement
 81     Function Func_Check_Standing(
 82       p_snum IN Students.snum%type,
 83       p_callnum IN Schclasses.callnum%type) return varchar2 is
 84       v_studentStanding Students.standing%type;
 85       v_courseStanding Courses.standing%type;
 86     Begin
 87       Select Standing into v_studentStanding from students where
 snum=p_snum;
 88       Select c. Standing into v_courseStanding from courses c, schclasses s
where s.ca llnum = p_callnum AND s.dept = c.dept AND s.cnum=c.cnum;
 89       If v_studentStanding < v_courseStanding Then
 90         return 'standing requirement has not met' ;
 91       End If;
 92       return null;
 93     End;
 94
 95     /* condition #1.6 check for Class Capacity, return TRUE if class still
 have room */
 96     Function Func_Check_Capacity(p_callnum IN SchClasses.callnum%type)
return boolean is
 97       v_capacity number;
 98       v_registered number;
 99     begin
100       select capacity into v_capacity from SchClasses where
callNum=p_callnum;
101       Select count(Snum) into v_registered from enrollments where
callnum=p_callnum an d Grade is null;
102       If v_registered < v_capacity Then
103         return TRUE;
104       Else
105         return FALSE;
106       End If;
107     end;
108
```

```
109     -- condition #1.7 check for Prerequisites
110     Function Func_Check_Prerequisites (
111       p_snum Students.snum%type,
112       p_callnum SchClasses.callnum%type) return varchar2 is
113       v_dept SchClasses.DEPT%Type;
114       v_cnum SchClasses.CNUM%Type;
115       v_count number;
116     begin
117       select DEPT into v_dept from SchClasses where callNum=p_callnum;
118       select CNUM into v_cnum from SchClasses where callNum=p_callnum;
119
120       Select count(DEPT) into v_count from (select pDept as DEPT, PCnum as
CNUM from PreReq where dept=v_dept and cnum=v_cnum
121       MINUS Select DEPT, CNUM from schclasses s, enrollments e where
e.snum=p_snum and s.callnum = e.callnum and GRADE in ('A','B','C', 'D'));
122       If v_count > 0 Then
123         return 'Prerequisite has not been met';
124       End If;
125       return null;
126     end;
127
128     -- sub function
129     Function Append_Messages(OldMessage varchar2, NewMessage varchar2)
return varchar2 is
130     Begin
131       If OldMessage is null Then
132         return NewMessage;
133       Else
134         If NewMessage is null Then
135           return OldMessage;
136         Else
137           return OldMessage || ', ' || NewMessage;
138         End If;
139       End If;
140     End;
141
142       -- sub procedure
143     Procedure Pro_Add_to_Waitlist (
144       p_snum Students.snum%type,
145       p_callnum SchClasses.callnum%type) is
146       v_count number;
147     Begin
148       Select count(Snum) into v_count from Waitlist where snum=p_snum and
callnum=p_callnum;
149       If v_count > 0 then
150         dbms_output.put_line ('Student ' || p_snum ||' is still in waitlist
```

```
for ' || p_callnum);
151       Else
152         insert into Waitlist values (p_snum,p_callnum, SysTimeStamp);
153         commit;
154         dbms_output.put_line ('The class was full, and you are added to
waitlist for ' || p_callnum);
155       End If;
156     End;
157
158     -- sub procedure
159     Procedure Pro_Enroll (
160       p_snum Students.snum%type,
161       p_callnum SchClasses.callnum%type) is
162       v_count number;
163     Begin
164       -- to make sure this student have not ever enrolled this class before
165       Select count(Snum) into v_count from Enrollments where snum=p_snum and
callnum=p_callnum;
166       If v_count = 0 then
167         insert into Enrollments values (p_snum, p_callnum, NULL);
168         dbms_output.put_line ('Student ' || p_snum || ' has just
successfully enrolled in ' || p_callnum);
169         commit;
170       Else
171         dbms_output.put_line ('You have already enrolled ' || p_callnum ||
' before');
172       End If;
173
174     End;
175
176     Procedure AddMe (
177       p_snum Students.snum%type, p_callnum SchClasses.callnum%type,
p_ErrorMsg OUT var char2) is
178       v_valid_snum boolean;
179       v_valid_callnum boolean;
180       v_available_room boolean;
181       v_error varchar2(300);
182       v_double_enroll_error varchar2(60);
183       v_undeclared_major_error varchar2(60);
184       v_15hour_rule_error varchar2(60);
185       v_standing_error varchar2(60);
186       v_prerequisite_error varchar2(60);
187
188     BEGIN
189       v_available_room:= false;
190       v_valid_snum := Func_validate_snum(p_snum);
```

```
191       v_valid_callnum := Func_validate_callnum(p_callnum);
192       IF v_valid_snum AND v_valid_callnum THEN
193         v_error := null;
194         v_double_enroll_error := Func_Check_Double_Enroll(p_snum,
p_callnum);
195         v_undeclared_major_error := Func_Check_Undeclared_Major(p_snum,
p_callnum);
196         v_15hour_rule_error := Func_Check_15Hour_Rule(p_snum, p_callnum);
197         v_standing_error := Func_Check_Standing(p_snum, p_callnum);
198         v_prerequisite_error := Func_Check_Prerequisites(p_snum, p_callnum);
199         v_error := Append_Messages(v_error, v_double_enroll_error);
200         v_error := Append_Messages(v_error, v_undeclared_major_error);
201         v_error := Append_Messages(v_error, v_15hour_rule_error);
202         v_error := Append_Messages(v_error, v_standing_error);
203         v_error := Append_Messages(v_error, v_prerequisite_error);
204         If v_error is null Then
205          v_available_room := Func_Check_Capacity(p_callnum);
206          if v_available_room then
207            -- ADD TO ENROLLMENT
208            Pro_Enroll (p_snum, p_callnum);
209          else
210            -- ADD TO WAITLIST
211            Pro_Add_to_Waitlist(p_snum, p_callnum);
212          end if;
213         Else
214          p_ErrorMsg := 'Enrollment error: ' || v_error;
215          dbms_output.put_line (p_ErrorMsg);
216         End If;
217       ELSE
218         If v_valid_snum Then
219          p_ErrorMsg := p_callnum || ' is not valid';
220         Elsif v_valid_callnum Then
221          p_ErrorMsg := p_snum || ' is not valid';
222         Else
223          p_ErrorMsg := p_snum || ' is not valid , and ' || p_callnum ||
' is not vali d, either';
224         End If;
225         dbms_output.put_line ('Enrollment error: ' || p_ErrorMsg);
226       END IF;
227
228     END;
229
230     -- sub procedure for dropme
231     Procedure proceed_waitlist (
232       p_snum Students.snum%type, p_callnum SchClasses.callnum%type) is
233       CURSOR cWaitlist is select snum, callnum from Waitlist where
```

```
callnum=p_callnum o rder by requestTime;
234       v_addClass_error varchar2(300);
235     BEGIN
236       FOR EachWait IN cWaitlist LOOP
237         v_addClass_error := null;
238         AddMe(EachWait.snum, p_callnum, v_addClass_error);
239         If v_addClass_error is Null then
240           delete from waitlist where snum=EachWait.snum and
callnum=p_callnum;
241           commit;
242           exit;
243         End If;
244       END LOOP;
245     END;
246
247     Procedure DropMe (
248       p_snum Students.snum%type, p_callnum SchClasses.callnum%type) is
249       v_ErrorMsg varchar2(60);
250       v_valid_snum boolean;
251       v_valid_callnum boolean;
252       v_count number;
253     BEGIN
254       v_valid_snum := Func_validate_snum(p_snum);
255       v_valid_callnum := Func_validate_callnum(p_callnum);
256
257       IF v_valid_snum AND v_valid_callnum THEN
258         /* 2. a student can only drop if he is enrolled in class that grade
has not been assigned */
259         select count (snum) into v_count from Enrollments    where
snum=p_snum AND calln um=p_callnum AND grade is null;
260         If v_count>0 Then
261           -- 3 withdraws with a 'W'
262           update enrollments set Grade='W' where snum=p_snum AND
callnum=p_callnum;
263           commit;
264           dbms_output.put_line(p_snum || ' has just successfully dropped' ||
p_call num);
265           proceed_waitlist(p_snum, p_callnum);
266         Else
267           dbms_output.put_line ('Class Dropping Error: Grade was assigned or
this student has not ever enrolled in this class!');
268         End If;
269       ELSE
270         If v_valid_snum Then
271           v_ErrorMsg := p_callnum || ' is not valid';
272         Elsif v_valid_callnum Then
```

```
273          v_ErrorMsg := p_snum || ' is not valid';
274        Else
275          v_ErrorMsg := p_snum || ' is not valid , and ' || p_callnum ||
' is not vali d, either';
276        End If;
277        dbms_output.put_line ('Class Dropping Error: ' || v_ErrorMsg);
278       END IF;
279     END;
280 End Enroll;
281 /

Package body created.

SQL> declare
  2      p_ErrorMsg varchar2(250);
  3 begin
  4      Enroll.AddMe('101',10110, p_ErrorMsg);
  5      Enroll.AddMe('104',10110, p_ErrorMsg);
  6      Enroll.AddMe('110',10110, p_ErrorMsg);
  7      Enroll.AddMe('114',10110, p_ErrorMsg);
  8      Enroll.AddMe('101',10130, p_ErrorMsg);
  9      Enroll.AddMe('101',10135, p_ErrorMsg);
 10      Enroll.AddMe('101',10141, p_ErrorMsg);
 11      Enroll.AddMe('101',10150, p_ErrorMsg);
 12 end;
 13 /
Student 101 has just successfully enrolled in 10110
Student 104 has just successfully enrolled in 10110
Student 110 has just successfully enrolled in 10110
Student 114 has just successfully enrolled in 10110
Student 101 has just successfully enrolled in 10130
Student 101 has just successfully enrolled in 10135
Student 101 has just successfully enrolled in 10141
Student 101 has just successfully enrolled in 10150

PL/SQL procedure successfully completed.

SQL> -- select snum, callnum, TO_CHAR (requestTime, 'MM-DD-YYYY HH24:MI:SS.FF')
as Request_Time from Waitlist order by requestTime;
SQL> show error;
No errors.
SQL> pause;

SQL> Spool off;
```

5. PROGRAM EXECUTION

To run AddMe, please enter:

```
Enroll.AddMe( p_snum, p_callnum, p_ErrorMsg);
```

To run DropMe, please enter:

```
Enroll.DropMe(p_snum, p_callnum);
```

Where:

- p_snum is the student number
- p_callnum is the course call number
- p_ErrorMsg is an OUT parameter

Please see appendix for my sample test.

APPENDIX – SAMPLE TEST

```
SQL> truncate table Waitlist;

Table truncated.

SQL> truncate table enrollments;

Table truncated.

SQL> declare
  2   p_ErrorMsg varchar2(250);
  3  begin
  4   Enroll.AddMe('101',10110, p_ErrorMsg);
  5   Enroll.AddMe('102',10110, p_ErrorMsg);
  6   Enroll.AddMe('104',10110, p_ErrorMsg);
  7   Enroll.AddMe('105',10110, p_ErrorMsg);
  8   Enroll.AddMe('110',10110, p_ErrorMsg);
  9   Enroll.AddMe('114',10110, p_ErrorMsg);
 10   Enroll.AddMe('115',10110, p_ErrorMsg);
 11   Enroll.AddMe('101',10115, p_ErrorMsg);
 12   Enroll.AddMe('101',10130, p_ErrorMsg);
 13   Enroll.AddMe('101',10135, p_ErrorMsg);
 14   Enroll.AddMe('122',10110, p_ErrorMsg);
 15   Enroll.AddMe('101',10141, p_ErrorMsg);
 16   Enroll.AddMe('101',10150, p_ErrorMsg);
 17   Enroll.AddMe('101',10160, p_ErrorMsg);
 18   Enroll.AddMe('124',10110, p_ErrorMsg);
 19  end;
 20  /
Student 101 has just successfully enrolled in 10110
Enrollment error: Undelared major student cannot enroll major course
Student 104 has just successfully enrolled in 10110
Enrollment error: Undelared major student cannot enroll major course, standing
requirement has not met
Student 110 has just successfully enrolled in 10110
Student 114 has just successfully enrolled in 10110
The class was full, and you are added to waitlist for 10110
Enrollment error: double enrollment error
Student 101 has just successfully enrolled in 10130
Student 101 has just successfully enrolled in 10135
The class was full, and you are added to waitlist for 10110
Student 101 has just successfully enrolled in 10141
Student 101 has just successfully enrolled in 10150
Enrollment error: 15 units exceeded
The class was full, and you are added to waitlist for 10110

PL/SQL procedure successfully completed.


SQL> select * from enrollments;
```

```
SNU   CALLNUM  GR
--- ---------- --
101     10110
104     10110
110     10110
114     10110
101     10130
101     10135
101     10141
101     10150

8 rows selected.

SQL> select snum, callnum, TO_CHAR (requestTime, 'MM-DD-YYYY HH24:MI:SS.FF' ) as

Request_Time from Waitlist order by requestTime;

SNU   CALLNUM REQUEST_TIME

--- ---------- ------------------------------
115     10110 12-06-2013 22:47:54.860000
122     10110 12-06-2013 22:47:54.876000
124     10110 12-06-2013 22:47:54.881000

SQL> begin
  2   Enroll.DropMe('114',10110);
  3   Enroll.DropMe('101',10150);
  4   Enroll.DropMe('104',10110);
  5  end;
  6  /
114 has just successfully dropped   10110
Student 115 has just successfully enrolled in 10110
101 has just successfully dropped   10150
104 has just successfully dropped   10110
Student 122 has just successfully enrolled in 10110

PL/SQL procedure successfully completed.

SQL> select * from enrollments;

SNU   CALLNUM GR
--- ---------- --
101     10110
104     10110   W
110     10110
114     10110   W
101     10130
101     10135
101     10141
101     10150   W
115     10110
122     10110

10 rows selected.


SQL> select snum, callnum, TO_CHAR (requestTime, 'MM-DD-YYYY HH24:MI:SS.FF' ) as

Request_Time from Waitlist order by requestTime;

SNU   CALLNUM REQUEST_TIME

--- ---------- ------------------------------
124     10110 12-06-2013 22:47:54.881000

SQL> spool off;
```