Term Project:
Database Security in a University

University of Maryland University College

CSMN 668

Project Business Description

This project will illustrate how to implement database security policies in the more visible and most used applications/tables of a University. For example, tables that support registration and enrollment will be presented. In addition, other tables beyond registration will be presented to illustrate other security mechanisms.

Various theoretical tables and users have been developed to demonstrate an oracle and SQL implementation of security. The approach of this project will be to define and implement organizational policies on a per-table basis. This project will present each table and discuss what type off access each type of user has to a particular table. Users will be assigned to roles to make user management simpler and more efficient (an important consideration for Database security as discussed in our Oracle book). These roles will be granted certain types of privileges on the various tables in the project. This will be the primary approach through out the project to illustrate Oracle database security.

Business Problem

The general business problem facing the University is the actual implementation of its security policies. The security implementation has to define how users will have access to a given table and to what extent. For example, will users be able to read and write to every column on a table? Will they have direct access to a table, or be given access through a view. Will users be able to create tables and similar functions, or will they only be allowed to connect to the database? These are the types of questions that need to be analyzed. It is the responsibility of the table developer/DBA to use the written security policies established by the organization's management as a form of guidance to deliver the desired database security.

As mentioned in the project description, this project will present security policies and their associated procedures on a per-table basis. In the proceeding sections this paper will present all the tables, users, security policies and associated procedures for each table.

TABLE CREATION

Enrollment Table

```
create table Enrollment
(Offering# number (4) not null,
 Faculty_ID number (4) not null,
 Instructor varchar (10),
 Course# char(7),
 Student_ID number (4) not null,
 Student varchar(10)
)
;
insert into Enrollment Values (1212, 8888,'SEAVER', 'IS320',100,'ABLE');
insert into Enrollment Values (1313, 1111,'JACKS', 'IS320',500,'WELLS');
insert into Enrollment Values (1414, 8888,'SEAVER','IS460',300,'CHARLES');
insert into Enrollment Values (1515, 9999,'LOONEY', 'IT480',400,'DRAKE');
insert into Enrollment Values (1616, 7777,'MARTIN', 'HI400',400,'DRAKE');
insert into Enrollment Values (1717, 6666,'LEE', 'CS420',300,'CHARLES');
insert into Enrollment Values (1818, 6666,'LEE', 'CS410',600,'SMITH');
insert into Enrollment Values (1919,2222,'CURTIS' , 'ED300',700,'BALCH');
insert into Enrollment Values (2121, 9999,'LOONEY', 'IT400',100,'ABLE');
insert into Enrollment Values (3131, 9999,'LOONEY' , 'IT500',500,'WELLS');

SQL> desc enrollment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 OFFERING#                                 NOT NULL NUMBER(4)
 FACULTY_ID                                NOT NULL NUMBER(4)
 INSTRUCTOR                                         VARCHAR2(10)
 COURSE#                                            CHAR(7)
 STUDENT_ID                                NOT NULL NUMBER(4)
 STUDENT                                            VARCHAR2(10)
```

Description: The primary purpose of this table will be to provide a way for students and faculty to view and/or change enrollment information.

Faculty_Contact Table

```
create table Faculty_Contact
(Faculty_ID number (4) not null,
 Name varchar2(20),
 Department varchar2(10),
 Rank varchar2(10),
 Email varchar2(20),
 Home_Phone varchar2(10)
)
;
insert into Faculty_Contact Values (1111, 'JACKS','IS',
'Lecturer','JACKS@open.edu','3012301111');
insert into Faculty_Contact Values (2222,
'CURTIS','ED','Lecturer','CURTIS@open.edu','3013302222');
insert into Faculty_Contact Values (3333, 'JOHNS', 'PS',
'Lecturer','JOHNS@open.edu','3014303333');
insert into Faculty_Contact Values (4444, 'EDWARDS', 'SA',
'Lecturer','EDWARDS@open.edu','3015304444');
insert into Faculty_Contact Values (5555, 'KIM', 'IM',
'Lecturer','KIM@open.edu','3016305555');
insert into Faculty_Contact Values (6666, 'LEE', 'CS',
'Lecturer','LEE@open.edu','3017306666');
insert into Faculty_Contact Values (7777, 'MARTIN', 'HI',
'Professor','MARTIN@open.edu','3018307777');
insert into Faculty_Contact Values (8888, 'SEAVER', 'IS',
'Professor','SEAVER@open.edu','3019308888');
insert into Faculty_Contact Values (9999, 'LOONEY', 'IT',
'Instructor','LOONEY@open.edu','2023331212');
insert into Faculty_Contact Values (1010, 'MILLS', 'SA',
'Lecturer','MILLS@open.edu','3015007777');

SQL> desc faculty_contact;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 FACULTY_ID                                NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(10)
 RANK                                               VARCHAR2(10)
 EMAIL                                              VARCHAR2(20)
 HOME_PHONE                                         VARCHAR2(10)
```

Description: The primary purpose of this table will be to provide a way for various types of users to be able to locate contact and instructor information. The sensitivity of the columns will dictate who will have access to what information. For example, only the program director and human_resource admin has access to faculty phone numbers.

Offering Table

```
create table Offering
(Offering# number (4) not null,
 Course# varchar2 (5),
 Faculty_ID number (4),
 Term varchar2(8),
 Year varchar2(4),
 TIME varchar2 (5)
)
;
insert into Offering Values (1212, 'IS320', 8888, 'Spring', '2004', '10 AM');
insert into Offering Values (1313, 'IS320', 1111, 'Spring', '2004', '11 AM');
insert into Offering Values (1414, 'IS460', 8888, 'Spring', '2004', '12 PM');
insert into Offering Values (1515, 'IT480', 9999, 'Spring', '2004', '10 AM');
insert into Offering Values (1616, 'HI400', 7777, 'Spring', '2004', '11 AM');
insert into Offering Values (1717, 'CS420', 6666, 'Spring', '2004', '11 AM');
insert into Offering Values (1818, 'CS410', 6666, 'Spring', '2004', '12 PM');
insert into Offering Values (1919, 'ED300', 2222, 'Spring', '2004', '11 AM');
insert into Offering Values (2121, 'IT400', 9999, 'Spring', '2004', '12 PM');
insert into Offering Values (3131, 'IT500', 9999, 'Spring', '2004', '11 AM');
```

```
SQL> desc offering
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 OFFERING#                                 NOT NULL NUMBER(4)
 COURSE#                                            VARCHAR2(5)
 FACULTY_ID                                         NUMBER(4)
 TERM                                               VARCHAR2(8)
 YEAR                                               VARCHAR2(4)
 TIME                                               VARCHAR2(5)
```

Description: The primary purpose of this table is to display course offerings for a semester and associated course information.

Student Major Table

```
create table Student_Major
(Student_ID number (10) not null,
 Name varchar2(8),
 Major varchar2(25),
 Status varchar2(2),
 Age number(2),
 GPA number(4,3)
)
;
insert into Student_Major Values (100, 'ABLE', 'Information Systems', 'SR',
21, 3.00);
insert into Student_Major Values (200, 'BAKER', 'Accounting', 'JR', 21,
2.70);
insert into Student_Major Values (300, 'CHARLES', 'Math', 'SR', 22, 3.50);
insert into Student_Major Values (400, 'DRAKE', 'Computer Science', 'FR', 18,
2.80);
insert into Student_Major Values (500, 'WELLS', 'Computer Science', 'SM', 19,
2.00);
insert into Student_Major Values (600, 'SMITH', 'Computer Science', 'SM', 19,
3.00);
insert into Student_Major Values (700, 'BALCH', 'Computer Science', 'SM', 19,
3.25);
insert into Student_Major Values (800, 'HAVEN', 'Philosophy', 'SM', 20,
3.25);
insert into Student_Major Values (999, 'WOOD', 'History', 'FR', 19, 2.25);
insert into Student_Major Values (900, 'BILL', 'Education', 'FR', 19, 3.00);
```

```
SQL> desc student_major;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 STUDENT_ID                                NOT NULL NUMBER(10)
 NAME                                               VARCHAR2(8)
 MAJOR                                              VARCHAR2(25)
 STATUS                                             VARCHAR2(2)
 AGE                                                NUMBER(2)
 GPA                                                NUMBER(4,3)
```

Description: The primary purpose of this table will be to display academic information for students as noted by the above fields.

Student_Info Table

```
create table Student_Info
(Student_ID number(4) not null,
 Name varchar2(20),
 Residency_Status varchar2(15),
 Age number(2),
 Street varchar2(30),
 City varchar2(15),
 State varchar2(2),
 Zip_Code number(5)
)
;
insert into Student_Info Values (100, 'ABLE', 'In State',21,'11 Red
St','Greenbelt','MD','20770');
insert into Student_Info Values (200, 'BAKER', 'In State',21,'12 Gray
St','Rockville','MD','20770');
insert into Student_Info Values (300, 'CHARLES', 'In State',22, '13 Blue
St','Laurel','MD','21250');
insert into Student_Info Values (400, 'DRAKE', 'Out of State',18, '14 Red
St','Dodge','VA','22380');
insert into Student_Info Values (500, 'WELLS', 'In State',19, '15 Purple
St','Dallas','TX','90145');
insert into Student_Info Values (600, 'SMITH', 'In State',19, '16 Orange
St','New HAVEN','CT','21770');
insert into Student_Info Values (700, 'BALCH', 'In State',19, '17 Violet
St','Baltimore','MD','21252');
insert into Student_Info Values (800, 'HAVEN', 'Out of State',20, '18 Beige
St','Pikesville','MD','21254');
insert into Student_Info Values (999, 'WOOD', 'In State',19, '19 Yellow
St','Dulles','VA','23770');
insert into Student_Info Values (900, 'BILL', 'Out of State',19, '20 Full
St','Germantown','MD','20879');
```

```
SQL> desc student_info;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 STUDENT_ID                                NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 RESIDENCY_STATUS                                   VARCHAR2(15)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 STATE                                              VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)
```

Description: This table will be primarily used for contact information and to determine student residency.

Faculty_Human_Resource_Info Table

```
create table Faculty_Human_Resource_Info
(Faculty_ID number (4) not null,
 Name varchar2(20),
 Marital_Status varchar2(1),
 Age number(2),
 Street varchar2(30),
 City varchar2(15),
 State varchar2(2),
 Zip_Code number(5)
)
;

insert into Faculty_Human_Resource_Info Values (1111, 'JACKS','M','45','10
Bank St','Shady','MD','20880');
insert into Faculty_Human_Resource_Info Values (2222, 'CURTIS', 'S', '31',
'30 House St','Shady','MD','20880');
insert into Faculty_Human_Resource_Info Values (3333, 'JOHNS', 'M', '47', '20
Book St','Shady','MD','20880');
insert into Faculty_Human_Resource_Info Values (4444, 'EDWARDS', 'S', '55',
'20 Kite St','Liberty','MD','20879');
insert into Faculty_Human_Resource_Info Values (5555, 'KIM', 'M', '43', '20
Welcome St','Liberty','MD','20879');
insert into Faculty_Human_Resource_Info Values (6666, 'LEE', 'F', '41', '20
Lane St','Poolsville','MD','22879');
insert into Faculty_Human_Resource_Info Values (7777, 'MARTIN', 'M', '56',
'20 First St','Hope','MD','21234');
insert into Faculty_Human_Resource_Info Values (8888, 'SEAVER', 'M', '41',
'20 Second St','Venice','MD','20371');
insert into Faculty_Human_Resource_Info Values (9999, 'LOONEY', 'F', '45',
'20 Third St','Damuscus','MD','20679');
insert into Faculty_Human_Resource_Info Values (1010, 'MILLS', 'M', '40', '20
Fourth St','Boyds','MD','20453');

SQL> desc Faculty_Human_Resource_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 FACULTY_ID                                NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 MARITAL_STATUS                                     VARCHAR2(1)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 STATE                                              VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)
```

Description: This table will be used and managed primarily by the human resource manager/administrator.

USER CREATION

Create Faculty users

```
create user CURTIS identified by cccccc1#;
create user EDWARDS identified by eeeeee1#;
create user JACKS identified by jjjjjj#1;
create user JOHNS identified by jjjjjj#2;
create user KIM identified by kkkkkk#1;
create user LEE identified by llllll#1;
create user LOONEY identified by llllll#2;
create user MARTIN identified by mmmmmm#1;
create user MILLS identified by mmmmmm#2;
create user SEAVER identified by ssssss#1;
```

Create Student users

```
create user ABLE identified by aaaaaa#1;
create user BALCH identified by bbbbbb#1;
create user BAKER identified by bbbbbb#2;
create user BILL identified by bbbbbb#3;
create user CHARLES identified by cccccc#2;
create user DRAKE identified by dddddd#1;
create user HAVEN identified by hhhhhh#1;
create user SMITH identified by ssssss#1;
create user WELLS identified by wwwwww#1;
create user WOOD identified by wwwwww#2;
```

Create Program_Director users

```
create user PROGRAM_DIRECTOR1 identified by pdpdpd#1;
```

Create Registrar users

```
create user Registrar1 identified by rrrrrr#1;
```

Create Administrator users

```
create user GENERAL_ADMIN identified by adadad#1;
create user HR_ADMIN identified by hrhrhr#1;
```

ROLE CREATION AND SETTING ASSOCIATED SYSTEM PRIVLEGES

```
create role FACULTY_ROLE;

grant connect to FACULTY_ROLE;

create role STUDENT_ROLE;

grant connect to STUDENT_ROLE;

create role PROGRAM_DIRECTOR;

grant connect to PROGRAM_DIRECTOR;

create role REGISTRAR_ROLE;

grant connect to REGISTRAR_ROLE;
```

NOTE: THEORITICALY I WOULD HAVE ONLY GRANTED CREATE SESSION PRIVLIEGE TO STUDENT_ROLE and FACULTY_ROLE, BECAUSE THAT IS ALL THE SYSTEM PRIVLIGES THEY WOULD NEED, HOWEVER, OUR ACCOUNTS DO NOT LET US GRANT INDIVIDUAL PRIVLIEGES. THEREFORE, I GRANTED CONNECT TO BOTH THESE ROLES.

```
create role ADMIN;

grant connect,resource to ADMIN with admin option;
```

ASSIGNING USERS TO A ROLE

```
grant FACULTY_ROLE to CURTIS,EDWARDS,JACKS,JOHNS,KIM,LEE,LOONEY,
MARTIN,MILLS,SEAVER;

grant STUDENT_ROLE to ABLE,BALCH,BAKER,BILL,CHARLES,DRAKE,
HAVEN,SMITH,WELLS,WOOD;

grant PROGRAM_DIRECTOR to PROGRAM_DIRECTOR1;

grant REGISTRAR_ROLE to REGISTRAR1;

grant ADMIN to GENERAL_ADMIN,HR_ADMIN;
```

FACULTY_CONTACT TABLE POLICY AND RULES

FACULTY ACCESS TO FACULTY_CONTACT

Security Policy: Allow faculty members to see other faculty information (except for phone number) in FACULTY_CONTACT:

```
create view PUBLIC_VIEW_OF_FACULTY as select name,department,rank,email from
FACULTY_CONTACT;
```

```
SQL> desc public_view_of_faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 NAME                                               VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(10)
 RANK                                               VARCHAR2(10)
 EMAIL                                              VARCHAR2(20)
```

```
Grant select on PUBLIC_VIEW_OF_FACULTY to FACULTY_ROLE;
```

Output:

```
SQL> show user
USER is "JACKS"
SQL> select * from qqrr.public_view_of_faculty;

NAME                 DEPARTMENT RANK       EMAIL
-------------------- ---------- ---------- --------------------
JACKS                IS         Lecturer   JACKS@open.edu
CURTIS               ED         Lecturer   CURTIS@open.edu
JOHNS                PS         Lecturer   JOHNS@open.edu
EDWARDS              SA         Lecturer   EDWARDS@open.edu
KIM                  IM         Lecturer   KIM@open.edu
LEE                  CS         Lecturer   LEE@open.edu
MARTIN               HI         Professor  MARTIN@open.edu
SEAVER               IS         Professor  SEAVER@open.edu
LOONEY               IT         Instructor LOONEY@open.edu
MILLS                SA         Lecturer   MILLS@open.edu

10 rows selected.
```

Security Policy: Allow each faculty member to view and update their own sensitive information (such as phone number) located in the FACULTY_CONTACT table. Through a view, faculty will be able to see all of their information, but will only be permitted to update their home phone and e-mail.

```
create view MY_FACULTY_CONTACT as
select name,department,rank,email,home_phone
from FACULTY_CONTACT
where FACULTY_CONTACT.name=user;
```

```
SQL> desc qqrr.my_faculty_contact
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 NAME                                               VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(10)
 RANK                                               VARCHAR2(10)
 EMAIL                                              VARCHAR2(20)
 HOME_PHONE                                         VARCHAR2(10)

grant select on MY_FACULTY_CONTACT to FACULTY_ROLE;
```

Output: (Jack can only see his information)

```
SQL> show user
USER is "JACKS"
SQL> select * from qqrr.my_faculty_contact;

NAME                 DEPARTMENT RANK       EMAIL                HOME_PHONE
-------------------- ---------- ---------- -------------------- ----------
JACKS                IS         Lecturer   JACKS@open.edu       3012301111


grant update (EMAIL,HOME_PHONE) on MY_FACULTY_CONTACT to FACULTY_ROLE;
```

Output: (Jack can only update his home phone and e-mail)

```
SQL> show user
USER is "JACKS"
SQL> update qqrr.my_faculty_contact set home_phone=3012301123;

1 row updated.

SQL> select * from qqrr.my_faculty_contact;

NAME                 DEPARTMENT RANK       EMAIL                HOME_PHONE
-------------------- ---------- ---------- -------------------- ----------
JACKS                IS         Lecturer   JACKS@open.edu       3012301123
```

Output: (Jack can not update any other column, only home phone and email)

```
SQL> show user
USER is "JACKS"
SQL> update qqrr.my_faculty_contact
  2  set qqrr.my_faculty_contact.department='CS';
update qqrr.my_faculty_contact
                *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Output: (Even if Jack guessed a valid user name, he can not update any one else's information - only his)

```
SQL> update qqrr.my_faculty_contact
  2  set qqrr.my_faculty_contact.home_phone=3012223344
  3  where name='KIM';

0 rows updated.
```

PROGRAM_DIRECTOR ACCESS TO FACULTY_CONTACT

Security Policy: A program director should be able to contact faculty personally to discuss class and curriculum related issues:

```
Grant select on FACULTY_CONTACT to PROGRAM_DIRECTOR;
```

STUDENT ACCESS TO FACULTY_CONTACT

Security Policy: Allow students to see to all fields in FACULTY_CONTACT (except for phone number):

```
grant select on PUBLIC_VIEW_OF_FACULTY to STUDENT_ROLE;
```

REGISTRAR ACCESS TO FACULTY_CONTACT

Security Policy: The registrar's office may access the view that excludes the phone numbers of faculty:

```
grant select on PUBLIC_VIEW_OF_FACULTY to REGISTRAR_ROLE;
```

ADMIN ACCESS TO FACULTY_CONTACT

Security Policy: The general administrator should have all rights on this table to be able update, trouble-shoot and fix any problems this table may encounter. The general administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on FACULTY_CONTACT to GENERAL_ADMIN with grant option;
```

STUDENT_MAJOR TABLE POLICY AND RULES

STUDENT ACCESS TO STUDENT_MAJOR

Security Policy: Allow students to only see their corresponding records. The registrar's office (REGISTRAR_ROLE) will be responsible for updates to this table:

```
create view MY_STUDENT_MAJOR as
select student_ID,name,major,status,age,gpa
from STUDENT_MAJOR
where STUDENT_MAJOR.name=user;
```

```
SQL> show user
USER is "ABLE"
SQL> desc qqrr.my_student_major;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 STUDENT_ID                                NOT NULL NUMBER(10)
 NAME                                               VARCHAR2(8)
 MAJOR                                              VARCHAR2(25)
 STATUS                                             VARCHAR2(2)
 AGE                                                NUMBER(2)
 GPA                                                NUMBER(4,3)
```

```
grant select on MY_STUDENT_MAJOR to STUDENT_ROLE;
```

Output: (Able can only see his information)

```
SQL> show user
USER is "ABLE"
SQL> select * from qqrr.my_student_major;

STUDENT_ID NAME     MAJOR                     ST        AGE        GPA
---------- -------- ------------------------- -- ---------- ----------
       100 ABLE     Information Systems       SR         21          3
```

FACULTY ACCESS TO STUDENT_MAJOR

Security Policy: Faculty does not need access or need to view this table.

REGISTRAR ACCESS TO STUDENT_MAJOR

Security Policy: As mentioned in the first security policy, the registrar will be responsible for maintaining, deleting, and updating the STUDENT_MAJOR table:

```
grant select,insert,update,delete on STUDENT_MAJOR to REGISTRAR_ROLE;
```

ADMINISTRATOR ACCESS TO STUDENT_MAJOR

Security Policy: The general administrator should have all rights on this table to be able update,trouble-shoot and fix any problems this table may encounter. The general administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on STUDENT_MAJOR to GENERAL_ADMIN with grant option;
```

OFFERING TABLE POLICY AND RULES

STUDENT ACCESS TO OFFERING

Security Policy: Students require read-only access to see class schedules:

```
grant select on OFFERING to STUDENT_ROLE;
```

FACULTY ACCESS TO OFFERING

Security Policy: Faculty require read-only access to see class schedules:

```
grant select on OFFERING to FACULTY_ROLE;
```

REGISTRAR ACCESS TO OFFERING

Security Policy: The registrar's office is responsible for establishing and maintaining class schedules, therefore, they will require complete access to the OFFERING table:

```
grant select,insert,update,delete,alter on OFFERING to REGISTRAR_ROLE;
```

ADMIN_ACCOUNT ACCESS TO OFFERING

Security Policy: The general administrator should have all rights on this table to be able update,trouble-shoot and fix any problems this table may encounter. The general administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on OFFERING to GENERAL_ADMIN with grant option;
```

ENROLLMENT TABLE POLICY AND RULES

STUDENT ACCESS TO ENROLLMENT

Security Policy: Students should be allowed to view and update/change their enrollment. However, they will only be permitted to change and view their own record:

```
create view MY_ENROLLMENT as
select offering#,instructor,course#
from ENROLLMENT
where ENROLLMENT.student=user;

SQL> desc qqrr.my_enrollment;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 OFFERING#                                 NOT NULL NUMBER(4)
 INSTRUCTOR                                         VARCHAR2(10)
 COURSE#                                            CHAR(7)

grant select on MY_ENROLLMENT to STUDENT_ROLE;
```

Output: (Able can only see his enrollment information)

```
SQL> show user
USER is "ABLE"
SQL> select * from qqrr.my_enrollment;

 OFFERING# INSTRUCTOR COURSE#
---------- ---------- -------
      1212 SEAVER     IS320
      2121 LOONEY     IT400
```

Security Policy: Theoretically, given a class is not filled, a student can add a course by being allowed to insert into MY_ENROLLMENT or remove a course by deleting from MY_ENROLLMENT.

```
grant insert,delete on MY_ENROLLMENT to STUDENT_ROLE;
```

Output: (Able is able to drop a course)

```
SQL> show user
USER is "ABLE"
SQL> delete from qqrr.my_enrollment
  2  where qqrr.my_enrollment.offering#=2121;

1 row deleted.

SQL> select * from qqrr.my_enrollment;

 OFFERING# INSTRUCTOR COURSE#
---------- ---------- -------
      1212 SEAVER     IS320
```

FACULTY ACCESS TO ENROLLMENT

Security Policy: Faculty requires read-only access to this table to see what students have registered for classes they are teaching:

```
create view FACULTY_ENROLLMENT_VIEW as
select offering#,course#,student_id,student
from ENROLLMENT
where ENROLLMENT.instructor=user;

USER is "JACKS"
SQL> desc qqrr.faculty_enrollment_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 OFFERING#                                 NOT NULL NUMBER(4)
 COURSE#                                            CHAR(7)
 STUDENT_ID                                NOT NULL NUMBER(4)
 STUDENT                                            VARCHAR2(10)

SQL>

grant select on FACULTY_ENROLLMENT_VIEW to FACULTY_ROLE;
```

Output: (This view shows faculty member Jack all the students enrolled in the course(s) he will be teaching)

```
SQL> show user
USER is "JACKS"
SQL> select * from qqrr.faculty_enrollment_view;

 OFFERING# COURSE# STUDENT_ID STUDENT
---------- ------- ---------- ----------
      1313 IS320          500 WELLS
```

REGISTRAR ACCESS TO ENROLLMENT

Security Policy: The registrar's office will need to have access to this table. For example, if a student isn't able to register successfully through his/her student account, then the registrar should be able to perform this function:

```
grant select,update,insert,delete on ENROLLMENT to REGISTRAR_ROLE;
```

ADMIN_ACCOUNT ACCESS TO ENROLLMENT

Security Policy: The general administrator should have all rights on this table to be able update,trouble-shoot and fix any problems this table may encounter. The general administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on ENROLLMENT to GENERAL_ADMIN with grant option;
```

STUDENT_INFO TABLE POLICY AND RULES

STUDENT ACCESS TO STUDENT_INFO
Security Policy: Students should only be allowed to view and update their own address information:

```
Create view my_sa_info_view as
select substr(student_id,1,3)ID,substr(name,1,6)NAME,
substr(residency_status,1,12)RESIDENCY,substr(age,1,3)AGE,
substr(street,1,13)STREET,substr(city,1,10)CITY,substr(state,1,4)STATE,
substr(zip_code,1,8)ZIP_CODE
from STUDENT_INFO
Where name=user;

SQL> show user
USER is "ABLE"
SQL> desc qqrr.my_sa_info_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 ID                                                 NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 RESIDENCY                                          VARCHAR2(15)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 ST                                                 VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)


Create view my_sa_info_update as
Select student_id,name,residency_status,age,street,city,state,zip_code
from STUDENT_INFO
Where name=user;

SQL> desc qqrr.my_sa_info_update;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
---
 STUDENT_ID                                NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 RESIDENCY_STATUS                                   VARCHAR2(15)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 STATE                                              VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)


grant select on MY_SA_INFO_VIEW to STUDENT_ROLE;
```

Output: (Only shows Able's information)

```
SQL> show user
USER is "ABLE"
SQL> select * from qqrr.my_sa_info_view;
```

```
ID  NAME   RESIDENCY    AGE STREET        CITY       STATE ZIP_CODE
--- ------ ------------ --- ------------- ---------- ----- --------
100 ABLE   In State     21  11 Red St     Greenbelt  MD    20770
```

```
SQL> grant update (street,city,state,zip_code) on MY_SA_INFO_UPDATE to
STUDENT_ROLE;
```

Output: (Able can only update his street,city,state, and zip_code)

```
SQL> show user
USER is "ABLE"
SQL> update qqrr.my_sa_info_update set
  2  street='New Street',city='New City',
  3  state='NS',zip_code='13123';

1 row updated.

SQL> select * from qqrr.my_sa_info_view;

ID  NAME   RESIDENCY    AGE STREET        CITY       STATE ZIP_CODE
--- ------ ------------ --- ------------- ---------- ----- --------
100 ABLE   In State     21  New Street    New City   NS    13123
```

Output: (Able can not update any other column)

```
SQL> show user
USER is "ABLE"
SQL> update qqrr.my_sa_info_update set
  2  residency_status='Out of State';
update qqrr.my_sa_info_update set
                *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Output: (If Able guessed a valid user name, he still can not update anyone else's information)

```
SQL> show user
USER is "ABLE"
SQL> update qqrr.my_sa_info_update set
  2  street='New Street',city='New City',
  3  state='NS',zip_code='13123'
  4  where name='WELLS';

0 rows updated.
```

FACULTY ACCESS TO STUDENT_INFO

Security Policy: Faculty does not need and will not have access to this table.

REGISTRAR ACCESS TO STUDENT_INFO

Security Policy: The registrar's office may access this table to verify student residency and for purposes of knowing where to send transcript requests and similar student paper work.

```
grant select on STUDENT_INFO to REGISTRAR_ROLE;
```

ADMIN_ACCOUNT ACCESS TO STUDENT_INFO

Security Policy: The general administrator should have all rights on this table to be able update, trouble-shoot and fix any problems this table may encounter. The general administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on STUDENT_INFO to GENERAL_ADMIN with grant option;
```

FACULTY_HUMAN_RESOURCE_INFO TABLE POLICY AND RULES

FACULTY ACCESS TO FACULTY_ADDRESS

Security Policy: Faculty should only be allowed to view their own information on this table. In addition they will be permitted to update their street, city, state, and zip-code information:

```
Create view MY_FHR_INFO_VIEW as
select substr(faculty_id,1,4)ID,substr(name,1,6)NAME,
substr(marital_status,1,1)M,substr(age,1,3)AGE,
substr(street,1,13)STREET,substr(city,1,10)CITY,substr(state,1,4)STATE,
substr(zip_code,1,8)ZIP_CODE
from FACULTY_HUMAN_RESOURCE_INFO
where FACULTY_HUMAN_RESOURCE_INFO.name=user;

SQL> show user
USER is "JACKS"
SQL> desc qqrr.my_fhr_info_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 ID                                        NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 M                                                  VARCHAR2(1)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 STATE                                              VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)


create view MY_FHR_INFO_UPDATE as
select faculty_id,name,marital_status,age,
street,city,state,zip_code
from FACULTY_HUMAN_RESOURCE_INFO
where FACULTY_HUMAN_RESOURCE_INFO.name=user;

SQL> desc qqrr.my_fhr_info_update;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
---
 FACULTY_ID                                NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 MARITAL_STATUS                                     VARCHAR2(1)
 AGE                                                NUMBER(2)
 STREET                                             VARCHAR2(30)
 CITY                                               VARCHAR2(15)
 STATE                                              VARCHAR2(2)
 ZIP_CODE                                           NUMBER(5)
grant select on MY_FHR_INFO_VIEW to FACULTY_ROLE;
```

Output: (Only shows Jacks information)

```
SQL> show user
USER is "JACKS"
SQL> select * from qqrr.my_fhr_info_view;
```

```
ID   NAME   M AGE STREET        CITY       ST ZIP_CODE
---- ------ - --- ------------- ---------- -- --------
1111 JACKS  M 45  10 Bank St    Shady      MD 20880
```

```
grant update (street,city,state,zip_code) on MY_FHR_INFO_UPDATE to
FACULTY_ROLE;
```

Output: (Jack can only update his street,city,state,and zip_code)

```
SQL> show user
USER is "JACKS"
SQL> update qqrr.my_fhr_info_update
  2  set street='19 New Ln',city='Newest',state='NE',zip_code=70124;

1 row updated.

SQL> select * from qqrr.my_fhr_info_view;

ID   NAME   M AGE STREET        CITY       STATE ZIP_CODE
---- ------ - --- ------------- ---------- ----- --------
1111 JACKS  M 45  19 New Ln     Newest     NE    70124
```

Output: (Jack can not update any other column)

```
SQL> show user
USER is "JACKS"
SQL> update qqrr.my_fhr_info_update
  2  set name='Jackson';
update qqrr.my_fhr_info_update
                *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Output: (If Jack guessed a valid user name, he still can not update anyone else's information)

```
SQL> show user
USER is "JACKS"
SQL> update qqrr.my_fhr_info_update
  2  set street='19 New Ln',city='Newest',state='NE',zip_code=70124
  3  where name='Lee';

0 rows updated.
```

STUDENT ACCESS TO FACULTY_HUMAN_RESOURCE_INFO

Security Policy: Students do not need and will not have access to this table.

REGISTRAR ACCESS TO FACULTY_HUMAN_RESOURCE_INFO

Security Policy: The registrar's office does not need access to this table.

ADMIN_ACCOUNT ACCESS TO FACULTY_HUMAN_RESOURCE_INFO

Security Policy: The human resource administrator is responsible for maintaining and updating this table, therefore, the human resource administrator should have all rights on this table to be able update,trouble-shoot and fix any problems this table may encounter. The human resource administrator also has the ability to grant object privileges on this table (as noted with the admin option). Therefore, the admin will be responsible for managing and updating privileges on this table for current and future users:

```
grant all on FACULTY_HUMAN_RESOURCE_INFO to HR_ADMIN with grant option;
```