Information Technology Institute (ITI)

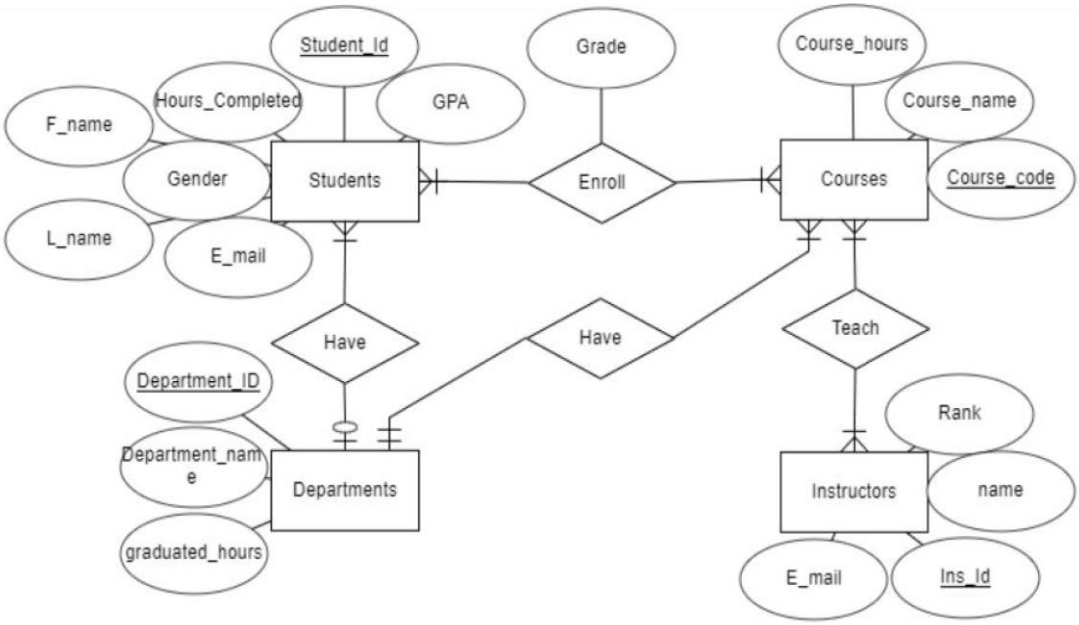
UNIVERSITY DBMS

Adham Ayman ELsayed

Contents

1. Database Design:

- **Entity Relationship Diagram.**



- **Mapping**

Students (#std_id ,F_name , L_name , Gender ,
hours_completed,e_mail , Dep_id(fk))

Departments (#Dep_id , Dep_name, graduated_hours)

Courses (#courses_code , courses_name, course_hours , Dep_id(fk))

Grades (#std_id(fk) , #courses_id(fk) , grade)

Instructors (#inst_ID , inst_name , E_mail)

Instructor_courses (#inst_id(fk) , #course_id(fk))

2. SQL Implementation

- **Create University User:**

1 – Connect to sysdba.

```
SQL> conn sys/123 as sysdba
```

2 – Create university Schema

```
SQL> create user university identified by 123;
```

3 - Grant DBA to university user

```
SQL> grant dba to university ;
```

- **Check Constraints**

```
ALTER TABLE students
ADD CONSTRAINT check_gpa CHECK (gpa > 0 and gpa <= 4.00);

ALTER TABLE students
ADD CONSTRAINT check_email
CHECK (REGEXP_LIKE(e_mail,'^[A-Za-z0-9._]+@example.com'));

ALTER TABLE grades
ADD CONSTRAINT grades_result CHECK (grade in ('A+', 'A' , 'A-' , 'B+' , 'B', 'B-','C+', 'C' , 'C-',
'D+','D','D-' , 'F'));
```

3. PLSQL Implementation:

- **Procedure to calculate and update GPA.**

```
CREATE OR REPLACE procedure UNIVERSITY2.update_gpa
is

total_hours number :=0;
total_grades number :=0;
v_gpa number(3,2) := 0;

begin

for i in (SELECT distinct std_id FROM grades )

loop
declare

v_student_id number := i.std_id;


cursor enrollment_cursor is
select e.std_id, e.course_code , e.grade, c.course_hours
from grades e , courses c
where e.course_code=c.course_code and e.std_id=v_student_id;
begin

for enrollment_record in enrollment_cursor
loop
total_hours := total_hours + enrollment_record.course_hours;

if enrollment_record.grade = 'A' or enrollment_record.grade = 'A+'  then total_grades := total_grades + (4 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'B+' then total_grades := total_grades + (3.7 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'B' then total_grades := total_grades + (3.3 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'B-' then total_grades := total_grades + (3 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'C+' then total_grades := total_grades + (2.7 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'C' then total_grades := total_grades + (2.3 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'C-' then total_grades := total_grades + (2 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'D+' then total_grades := total_grades + (1.7 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'D' then total_grades := total_grades + (1.3 * enrollment_record.course_hours);
elsif enrollment_record.grade = 'D-' then total_grades := total_grades + (1 * enrollment_record.course_hours);
else  total_grades := total_grades + (0 * enrollment_record.course_hours);
end if;
end loop ;
v_gpa := total_grades / total_hours;



update Students
SET gpa = v_gpa
WHERE student_id = v_student_id;
end;




end loop;
end;
/
```

- **Function to calculate average GPA for each department.**

```
CREATE OR REPLACE function UNIVERSITY2.avg_gpa
return number
is
total_gpa number;
total_students number ;

begin
for i in (select department_id  from departments)
loop
declare

v_dep_id number := i.department_id;

cursor dep_cursor is
select GPA
from students
where dept_id = v_dep_id ;


begin

total_gpa := 0;
total_students := 0 ;
for dep_record in dep_cursor
loop
total_gpa := total_gpa + dep_record.GPA;
total_students := total_students + 1;

end loop ;

dbms_output.put_line( 'Department id:'|| v_dep_id ||  ', AVG GPA: ' ||  total_gpa /total_students );
end;
-- return total_gpa / (select count(*) from students group by dept_id) ;
end loop ;


return null ;
end ;
/
```

- **Procedure to update grade with trigger to update GPA After updating grade.**

  - Procedure to update Grade.

```
CREATE OR REPLACE procedure UNIVERSITY2.update_grade(v_std_id in number , v_course_id in number , new_grade in varchar2)
is

v_grade varchar2(2 byte) ;

begin

select grade
into v_grade
from grades
where course_code = v_course_id and std_id = v_std_id;

if v_grade in ('A+' , 'A' , 'A-' , 'B+' , 'B' , 'B-') then
      dbms_output.put_line('Grade is '|| v_grade || ', You cannot improve grade higher than C+') ;

else
   if new_grade in ('A+' , 'A' , 'A-') then

      update grades
      set grade = 'B+'
      where course_code = v_course_id and std_id = v_std_id ;
      dbms_output.put_line(' The maximum improvement grade is B+') ;
   else
      update grades
      set grade = new_grade
      where course_code = v_course_id and std_id = v_std_id ;
   end if ;
end if;
```

  - Trigger to update GPA after updating in student.

```
DROP TRIGGER UNIVERSITY2.UPDATEGPATRIGGER;

CREATE OR REPLACE TRIGGER UNIVERSITY2.UpdategpaTrigger
FOR INSERT OR UPDATE  ON UNIVERSITY2.GRADES
COMPOUND TRIGGER
    student_ids number;
     after each row is
     begin
      student_ids := :new.std_id;
     end after each row;

     after statement is
     begin
      update_one_std(student_ids);
     end after statement;
END;
/
```

- **Procedure to Update department with trigger to insert old data in table student_history.**

  - Procedure to Update Department

```
CREATE OR REPLACE procedure UNIVERSITY2.update_dept(v_std_id in number , new_dep in number)
is

begin
update students
set dept_id = new_dep
where student_id = v_std_id ;

end ;
/
```

  - Trigger to insert old data in table student_history and delete data from table grades.

```
DROP TRIGGER UNIVERSITY2.STD_HISTORY;

CREATE OR REPLACE TRIGGER UNIVERSITY2.std_history
before update ON UNIVERSITY2.STUDENTS for each row
declare
olddept number ;
begin
olddept := :old.dept_id ;
   INSERT INTO student_history (STUDENT_ID, OLD_DEPARTMENT, COURSE_CODE, GRADE)
   SELECT :NEW.STUDENT_ID, :OLD.dept_id, g.course_code, g.grade
   FROM grades g
   JOIN courses c ON g.course_code = c.course_code AND c.dept_id = olddept
   WHERE g.std_id = :NEW.STUDENT_ID;

   DELETE FROM grades
   WHERE std_id = :NEW.STUDENT_ID AND course_code IN (SELECT course_code FROM courses WHERE dept_id = olddept);

end ;
/
```

4. Automation Scripts:

- **Bash Script for backup.**

```bash
#!/bin/bash

SOURCE_DIR="/C/oraclexe/app/oracle"
BACKUP_DIR="/C/Users/lenovo/Desktop/Case_Study/Bash/backup_folder"

TIMESTAMP=$(date +%Y%m%d_%H%M%S)

tar cvf "${BACKUP_DIR}/backup_${TIMESTAMP}.tar" -C "${SOURCE_DIR}" .

echo "Backup completed. Archive stored in ${BACKUP_DIR}/backup_${TIMESTAMP}.tar"
```

- **Bash Script for disk space.**

```bash
#!/bin/bash

# Set the threshold for disk usage (in percentage)
THRESHOLD=80

# Get current disk usage


DISK_USAGE=$(df -h | grep "C:"| awk '{ print $6 }' | tr '%' ' ')
#echo $DISK_USAGE


if [[ $DISK_USAGE -gt $THRESHOLD ]]; then
    # Send alert (replace with your notification method)
    echo "ALERT: Disk usage exceeded $THRESHOLD% - Current usage: $DISK_USAGE%"
else
    echo "Disk usage is within the threshold: $DISK_USAGE%"
fi

```

- **Bash Script for memory usage.**

```bash
TotalSpace=$(systeminfo | grep 'Total Physical Memory' | awk '{ print $4 }' | tr -d ',')
AvailableSpace=$(systeminfo | grep 'Available Physical Memory' | awk '{ print $4 }' | tr -d ',')
UsedSpace=$((TotalSpace - AvailableSpace))
echo "Used Space = $UsedSpace MB"

Percentt=$(((AvailableSpace*100)/UsedSpace))
THRESHOLDD=60
if [[ $Percentt -gt $THRESHOLDD ]]; then
   echo "ALERT !! Memory consumption has exceeded $THRESHOLDD% , Current Consumption : $Percentt%"
else
   echo "Memory consumption is safe and within the threshold : $Percentt%"
fi
```