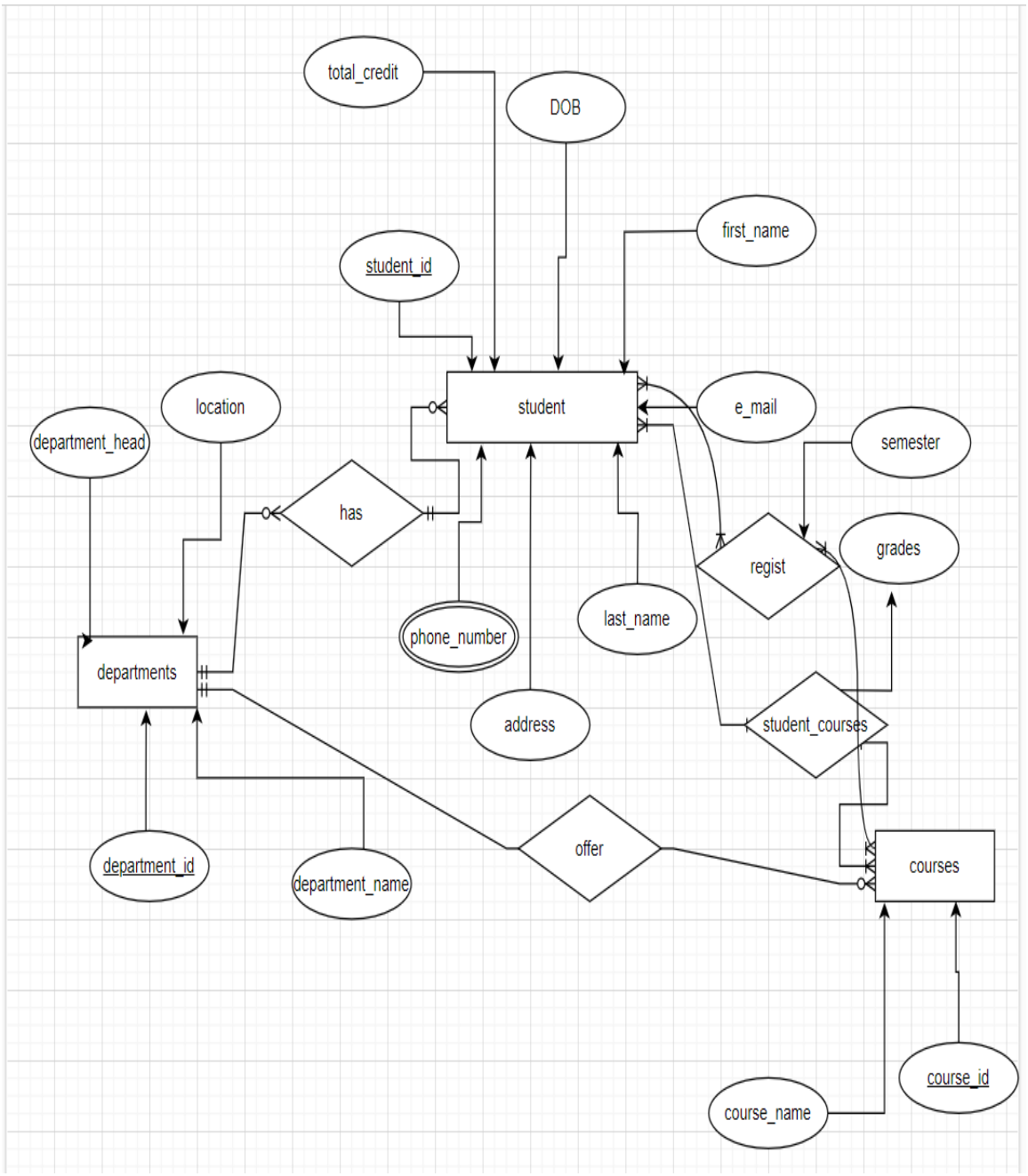
University DBMS

ENG: Mohamed hossam

Table of content:

1-ER-Diagram:


total_credit
DOB
first_name
student_id
location
student
e_mail
department_head
semester
has
regist
grades
last_name
phone_number
departments
address
student_courses
offer
department_id
department_name
courses
course_id
course_name

2-Mapping:

Zero normal form:

Student [#student_id,first_name,last_name,e_mail,city,street,DOB,
total_credit,phone_number,department_id(fk)];

Department [#department_id,location,department_name,department_head];

Courses [#courses_id,course_name,credit,department_id(fk)];

Student_grades [student_id(fk),course_id(fk),grades];

Enrollment[student_id(fk),course_id(fk),semester];

First normal form:

I have multivalued attribute (phone_number) so we but it in a new table (student_phone)

Student [#student_id,first_name,last_name,e_mail,
total_credit,city,street,DOB,department_id(fk)];

Student_phone [student_id(fk),phone_number];

Department [#department_id,location,department_name,department_head];

Courses [#courses_id,course_name,credit,department_id(fk)];

Student_grades [student_id(fk),course_id(fk),grades];

Enrollment[student_id(fk),course_id(fk),semester];

Second normal form:

I don’t have any partial dependency so I don’t have second normal form.

Third normal form:

I don’t have any transitive dependency so I don’t have third normal form.

3-SQL implementation:

1. Creating a new user and give the privileges

SQL> conn sys/123 as sysdba

Connected.

SQL> create user university identified by uni;

User created.

SQL> grant dba to university;

Grant succeeded.

2. Creating tables

```
CREATE TABLE UNIVERSITY.COURSES
(
  COURSE_ID     INTEGER,
  COURSE_NAME   VARCHAR2(255 BYTE),
  DEPARTMENT_ID INTEGER,
  CREDIT        INTEGER
)
```

```
CREATE TABLE UNIVERSITY.DEPARTMENT
(
  DEPARTMENT_ID   INTEGER,
  LOCATION        VARCHAR2(255 BYTE),
  DEPARTMENT_NAME VARCHAR2(255 BYTE),
  DEPARTMENT_HEAD VARCHAR2(255 BYTE)
)
```

```
CREATE TABLE UNIVERSITY.ENROLLMENT
(
  STUDENT_ID INTEGER,
  COURSE_ID  INTEGER,
  SEMESTER   INTEGER
)
```

```
CREATE TABLE UNIVERSITY.STUDENT
(
  STUDENT_ID    INTEGER,
  FIRST_NAME    VARCHAR2(255 BYTE),
  LAST_NAME     VARCHAR2(255 BYTE),
  E_MAIL        VARCHAR2(255 BYTE),
  DOB           DATE,
  DEPARTMENT_ID INTEGER,
  CITY          VARCHAR2(50 BYTE),
  STREET        VARCHAR2(50 BYTE),
  GPA           NUMBER,
  TOTAL_CREDIT  NUMBER
)
```

```
CREATE TABLE UNIVERSITY.STUDENT_GRADES
(
  STUDENT_ID  INTEGER,
  COURSE_ID   INTEGER,
  GRADES      VARCHAR2(2 BYTE)
)
```

```
CREATE TABLE UNIVERSITY.STUDENT_PHONE
(
  STUDENT_ID    INTEGER,
  PHONE_NUMBER  VARCHAR2(15 BYTE)
)
```

3-check constraint

```
ALTER TABLE Student
ADD CONSTRAINT gpa_range_check
CHECK (gpa >= 0 AND gpa <= 4.0);
```

Check constraint if GPA between (0,4)

```
ALTER TABLE Student
ADD CONSTRAINT email_format_check
CHECK (REGEXP_LIKE(E_MAIL, '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'));
```

Check if email wrote in the valid format

4-PL/SQL implementation

Calculate GPA:

```
CREATE OR REPLACE PROCEDURE UNIVERSITY.calculate_student_gpa(student_id IN Student_grades.student_id%TYPE) IS
   v_total_credits NUMBER := 0;
   v_total_grade_points NUMBER := 0;
   v_gpa NUMBER;

   CURSOR grade_cursor IS
      SELECT * FROM S_GRADE WHERE student_id = calculate_student_gpa.student_id;

BEGIN
   FOR gra_rec IN grade_cursor LOOP
      CASE gra_rec.GRADES
         WHEN 'A' THEN
            v_total_grade_points := v_total_grade_points + (4.0 * gra_rec.CREDIT);
         WHEN 'B' THEN
            v_total_grade_points := v_total_grade_points + (3.0 * gra_rec.CREDIT);
         WHEN 'C' THEN
            v_total_grade_points := v_total_grade_points + (2.0 * gra_rec.CREDIT);
         WHEN 'D' THEN
            v_total_grade_points := v_total_grade_points + (1.0 * gra_rec.CREDIT);
         WHEN 'F' THEN
            v_total_grade_points := v_total_grade_points + (0.0 * gra_rec.CREDIT);
      END CASE;

      v_total_credits := v_total_credits + gra_rec.CREDIT;
   END LOOP;

   IF v_total_credits > 0 THEN
      v_gpa := v_total_grade_points / v_total_credits;


      UPDATE Student
      SET GPA = v_gpa,total_credit = v_total_credits
      WHERE student_id = calculate_student_gpa.student_id;
```

```
   ELSE

      UPDATE Student
      SET GPA = 0,total_credit = 0
      WHERE student_id = calculate_student_gpa.student_id;

   END IF;
END ;
```

Update grade:

```
CREATE OR REPLACE procedure UNIVERSITY.update_grade(v_std_id in number , v_course_id in number , new_grade in varchar2)
is
v_grade varchar2(2 byte) ;
begin
select GRADES
into v_grade
from STUDENT_GRADES
where COURSE_ID = v_course_id and STUDENT_ID = v_std_id;

if v_grade in ( 'A' , 'B' ) then
      dbms_output.put_line('Grade is '|| v_grade || ', You cannot improve grade higher than C') ;
else
   if new_grade in ('A' ) then
        update STUDENT_GRADES
      set GRADES = 'B'
      where COURSE_ID = v_course_id and STUDENT_ID = v_std_id ;
   else
      update STUDENT_GRADES
      set GRADES = new_grade
      where COURSE_ID = v_course_id and STUDENT_ID = v_std_id ;
   end if ;
end if;
end ;
```

Trigger :

```
CREATE OR REPLACE TRIGGER UNIVERSITY.S_G_T
FOR INSERT OR UPDATE  ON UNIVERSITY.STUDENT_GRADES
COMPOUND TRIGGER
    student_ids number;
     after each row is
     begin
       student_ids := :new.student_id;
     end after each row;

     after statement is
     begin
      CALCULATE_STUDENT_GPA(student_ids);
     end after statement;
END;
```

5-bash scripting

Back-up script

```
#!/bin/bash

DB_USER="UNIVERSITY"
DB_PASSWORD="uni"
DB_NAME="XE"



TIMESTAMP=$(date "+%Y%m%d%H%M%S")
name="$DB_USER$TIMESTAMP"
BACKUP_FILE="$name.dmp"
log_name=$TIMESTAMP"_log"
log_file="$log_name.log"



expdp $DB_USER/$DB_PASSWORD@$DB_NAME DIRECTORY=my_data_pump_dir DUMPFILE=$BACKUP_FILE LOGFILE=$log_file SCHEMAS=$DB_USER


if [ $? -eq 0 ]; then
    echo "Database backup completed successfully: $BACKUP_FILE"
else
    echo "Error: Database backup failed! See expdp.log for details."
fi
```

Check the current use of the dick

```
#!/bin/bash


THRESHOLD=90
TIMESTAMP=$(date "+%Y%m%d%H%M%S")

DISK_USAGE=$(df -h / | awk 'NR==2 {print $5}' | cut -d'%' -f1)


if [ $DISK_USAGE -gt $THRESHOLD ]; then

    ALERT_MESSAGE="Disk space alert: Current usage is $DISK_USAGE%"

    echo "$(date): $ALERT_MESSAGE" > C:/Users/in/Desktop/disk_space/$TIMESTAMP"_logs".log

fi
```

Monitor the available space in the dick

```
#!/bin/bash
threshold=10
TIMESTAMP=$(date "+%Y%m%d%H%M%S")

disk_space=$(df / | awk 'NR==2 {print $4/$2 * 100.0}')

if [[ $disk_space < $threshold ]]; then
    touch C:/Users/in/Desktop/anmulies_check/$TIMESTAMP"_disk_available_logs".log
    echo "$(date): disk_space=$disk_space" > C:/Users/in/Desktop/anmulies_check/$TIMESTAMP"_disk_available_logs".log

fi
```

Check sudden increase in database

```
#!/bin/bash
TIMESTAMP=$(date "+%Y%m%d%H%M%S")

log_file="C:/Users/in/Desktop/anmulies_check/$TIMESTAMP _sudden_increace.log"


threshold_increase=20


echo "$(date): Starting memory monitoring" >> "$log_file"


initial_memory=$(df -h / | awk 'NR==2 {print $5}' | cut -d'%' -f1)


sleep 5


current_memory=$(df -h / | awk 'NR==2 {print $5}' | cut -d'%' -f1)


percentage_increase=$(( ((current_memory - initial_memory) * 100) / initial_memory ))


echo "$(date): Memory usage increase: $percentage_increase %" >> "$log_file"


if [ $percentage_increase -gt $threshold_increase ]; then

    echo "$(date): Abnormal increase in memory consumption. Change: $percentage_increase %" >> "$log_file"

fi


echo "$(date): Memory monitoring complete" >> "$log_file"
```