COMP 3005B
Assignment #4
Due: Nov. 13 @11:59PM

Instruction

1. You should do the assignment independently. If copying is found, the case will be reported to the office of the Dean of Science immediately.
2. The assignment must be typed, completed on an individual basis, and submitted as a single Word/PDF file with your name as the filename to **brightspace**.
3. Last in the Supplier table is your last name. If your information is not shown correctly in the result, you will get a 0 mark for the assignment.
4. You should directly do your assignment on this document and name the document with your last name followed by your first name so that it is easy for TAs to mark.
5. For Part 2, you need to use Oracle VM and SQLPLUS interface to Oracle DBMS, test each program carefully, and submit the final version of the program together with several representative screenshots of the execution of the program. If there is no screenshot, you will get 0 for the question.

Part 1 Concepts (20 marks)

Explain the following concepts based on the information in the lecture notes.

1. View: abstract way to access data in a database. Users create views and query the view as if it were the real table.
2. Impedance Mismatch: PL/SQL is a procedural language designed to write procedure code with loops and conditions whereas SQL is a tuple-based language. SQL results are sets of tuples whereas PL/SQL can only process one record at a time. This means we need a way to bridge together the impedance mismatch between PL/SQL and SQL. (One way is using cursors)
3. Execute immediate in PLSQL: used to execute SQL statements dynamically. You can choose to redirect the result of the SQL query onto a variable or record using the INTO clause.
4. Cursor in PLSQL: a pointer to the result of a query. This allows us to work with individual rows of the result tuples returned by SQL statements. There are two types: explicit (user-defined) and implicit (automatically created).
5. Update Cursor: A type of cursor that allows you to update data in a table, usually based on certain conditions. It allows you to fetch and update one or more rows of data in a table. Useful when you want to make updates and conditional updates to a selective row(s)
6. Parameterized Cursor in PLSQL: A type of cursor that has parameters. This allows us to pass values as parameters to the cursor when it's opened, rather than hardcoding specific values into the query. This gives us some flexibility when using SQL queries.
7. Physical Model: At the lowest level, a database system is designed using a set of storage and structural rules. This is known as the physical design. This includes things like how the data will be stored (data types), retrieved, etc.

8. ER Model: At the highest level, a database system is conceptually designed using an ER, EER, or UML model that outlines the schema for the mini-world. ER represents real-world entities and their relationships.
9. EER Model: extended entity-relationship modeling includes all modeling concepts of basic ER with addition concepts such as subtypes, specialization, categories and attribute and relationship inheritance. This more accurately conceptualizing applications.
10. Aggregation: Used to show hierarchy relationships between entities, where one entity is composed of or includes other entities. This makes it easier to understand and represent real world scenarios since entities are organized in certain ways.

Part 2 PL/SQL (40 Marks)

This part is based on the Suppliers-Parts database which has three tables shown below.

1. Delete all three tables if you created before and then use **execute immediate** statement to write a PL/SQL program to create and populate the three tables (10 marks)

Creating the tables:

```
begin
execute immediate
'create table Suppliers (
S# char(4) primary key,
Name varchar2(10),
Status integer,
City varchar2(10))';

execute immediate
'create table Parts (
P# char(4) primary key,
Name varchar2(10),
Color varchar2(10),
Weight number,
City varchar2(10))';

execute immediate
'create table SP (
S# char(4) references Supplier(S#),
P# char(4) references Parts(P#),
QTY integer,
primary key (S#,P#))';
end;
/
```

```
SQL> begin
execute immediate
'create table Supplier (
S# char(4) primary key,
Name varchar2(10),
Status integer,
City varchar2(10))';

execute immediate
'create table Parts (
P# char(4) primary key,
Name varchar2(10),
Color varchar2(10),  2    3    4    5    6    7    8    9   10   11   12   13
Weight number,
Ci 14   15  ty varchar2(10))';

execute immediate
'create tabl 16   17   18  e SP (
S# char(4) references Supplier(S#),
P# char 19   20  (4) references Parts(P#),
QTY integer,
primary key 21   22   (S#,P#))';
 23  end;
 24  /

PL/SQL procedure successfully completed.

SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
SP
PARTS
SUPPLIER
GRADE
COURSE
STUDENT

6 rows selected.

SQL> alter table Supplier rename to Suppliers;

Table altered.
```

Populating Suppliers:

```
begin
execute immediate
'insert into Suppliers values("S1", "Smith", 20, "London")';
execute immediate
```

'insert into Suppliers values("S2", "Jones", 30, "Paris")';
execute immediate
'insert into Suppliers values("S3", "Blake", 30, "Paris")';
execute immediate
'insert into Suppliers values("S4", "Rodelo", 20, "London")';
execute immediate
'insert into Suppliers values("S5", "Adams", 30, "Athens")';
end;

```
PL/SQL procedure successfully completed.

SQL> select * from Suppliers;

S#   NAME           STATUS CITY
---- ---------- ---------- ----------
S1   Smith              20 London
S2   Jones              30 Paris
S3   Blake              30 Paris
S4   Rodelo             20 London
S5   Adams              30 Athens
```

Populating Parts:
begin
execute immediate
'insert into Parts values("P1", "Nut", "Red", 12.0, "London")';
begin
execute immediate
'insert into Parts values("P2", "Bolt", "Green", 17.0, "Paris")';
execute immediate
'insert into Parts values("P3", "Screw", "Blue", 17.0, "Oslo")';
execute immediate
'insert into Parts values("P4", "Screw", "Red ", 14.0, "London")';
execute immediate
'insert into Parts values("P5", "Cam", "Blue", 12.0, "Paris")';
execute immediate
'insert into Parts values("P6", "Cog", "Red", 19.0, "London")';
end;
/

```
PL/SQL procedure successfully completed.

SQL> select * from parts;

P#   NAME       COLOR          WEIGHT CITY
---- ---------- ---------- ---------- ----------
P1   Nut        Red                12 London
P2   Bolt       Green              17 Paris
P3   Screw      Blue               17 Oslo
P4   Screw      Red                14 London
P5   Cam        Blue               12 Paris
P6   Cog        Red                19 London
```

Populating SP:

```
begin
execute immediate
'insert into SP values("S1", "P1", 300)';
execute immediate
'insert into SP values("S1", "P2", 200)';
execute immediate
'insert into SP values("S1", "P3", 400)';
execute immediate
'insert into SP values("S1", "P4", 200)';
execute immediate
'insert into SP values("S1", "P5", 100)';
execute immediate
'insert into SP values("S1", "P6", 100)';
execute immediate
'insert into SP values("S2", "P1", 300)';
execute immediate
'insert into SP values("S2", "P2", 400)';
execute immediate
'insert into SP values("S3", "P2", 200)';
execute immediate
'insert into SP values("S4", "P2", 200)';
execute immediate
'insert into SP values("S4", "P3", 300)';
execute immediate
'insert into SP values("S4", "P4", 400)';
execute immediate
'insert into SP values("S4", "P5", 500)';
execute immediate
'insert into SP values("S4", "P6", 600)';
end;
/
```

```
PL/SQL procedure successfully completed.

SQL> select * from SP
  2  ;

S#   P#          QTY
---- ---- ----------
S1   P1          300
S1   P2          200
S1   P3          400
S1   P4          200
S1   P5          100
S1   P6          100
S2   P1          300
S2   P2          400
S3   P2          200
S4   P2          200
S4   P3          300

S#   P#          QTY
---- ---- ----------
S4   P4          400
S4   P5          500
S4   P6          600
```

2. Write a PL/SQL program to list all supplier rows, in supplier number order so that each supplier row is immediately followed in the listing by all bank rows for parts that the supplier supplies in part number order. Supplies that do not supply parts should still be listed. (15 marks)

```
begin
 for S in (select * from Suppliers order by S# ASC) loop
  dbms_output.put_line(S.S#||' '||S.name||' '||S.status||' '||S.city);
  for P in (select P.P#, P.name, P.color, P.weight, P.City from Parts P, SP A
        where S.S# = A.S# and A.P# = P.P#) loop
   dbms_output.put_line(P.P#||' '||P.name||' '||P.color||' '||P.weight||' '||P.city);
```

```
    end loop;
  end loop;
end;
/
```

```
SQL> SET SERVEROUTPUT ON;
SQL> begin
  for S in (select * from Suppliers order by S# ASC) loop
    dbms_output.put_line(S.S#||' '||S.name||' '||S.status||' '||S.city);
    for P in (select P.P#, P.name, P.color, P.weight, P.City from Parts P, SP A
              where S.S# = A  2    3    4    5  .S# and A.P# = P.P#) loop
      dbms_output.put_line(P  6  .P#||' '||P.name||' '||P.color||' '||P.weight||' '||P.city);
    end loop;
  end loop;
end;
  7    8    9   10  /
S1   Smith 20 London
P1   Nut Red 12 London
P2   Bolt Green 17 Paris
P3   Screw Blue 17 Oslo
P4   Screw Red  14 London
P5   Cam Blue 12 Paris
P6   Cog Red 19 London
S2   Jones 30 Paris
P1   Nut Red 12 London
P2   Bolt Green 17 Paris
S3   Blake 30 Paris
P2   Bolt Green 17 Paris
S4   Rodelo 20 London
P2   Bolt Green 17 Paris
P3   Screw Blue 17 Oslo
P4   Screw Red  14 London
P5   Cam Blue 12 Paris
P6   Cog Red 19 London
S5   Adams 30 Athens

PL/SQL procedure successfully completed.
```

3. Redo question 2 using a parameterized cursor that takes a supplier name. It should first prompt the user to enter a supplier name and then display the same information as in 2 just for the given customer. Use your own last name in the table to test this program. (15 marks)

```
declare
  cursor S (sname Suppliers.name%type) is select * from Suppliers
  where sname = Suppliers.name;
  sn Suppliers.S#%type;
  name Suppliers.name%type;
  status Suppliers.status%type;
  city Suppliers.city%type;
begin
  open S('&name');
  loop
    fetch S into sn, name, status, city;
```

```
    exit when S%NOTFOUND;
    dbms_output.put_line(sn ||' '||name||' '||status||' '||city);
    for P in (select P.P#, P.Name, P.Color, P.Weight, P.City from Parts P, SP A
            where A.S# = sn and P.P# = A.P# order by P.P#) loop
      dbms_output.put_line(P.P#||' '||P.name||' '||P.color||' '||P.weight||' '||P.city);
    end loop;
  end loop;
end;
/
```

```
SQL> declare
  cursor S (sname Suppliers.name%type) is   2  select * from Suppliers
  where sname = Suppliers.  3  name;
  sn Suppliers.S#%type;
  name Suppliers.nam  4    5  e%type;
  status Suppliers.status%type;
  city Suppliers.city%type;
begin
  open S('&name');
  loop
    fetch S into sn, name, status, city;
    exit when S%NOTFOUND;
    dbms_output.put_line(sn ||' '||name||' '||status||' '||city);
    for P in (select P.P#, P.Name, P.Color, P.Weight, P.City from P  6    7    8    9   10   11   12   13   14  arts P, SP A
              where A.S# = sn and P. 15  P# = A.P# order by P.P#) loop
      dbms_output.pu 16  t_line(P.P#||' '||P.name||' '||P.color||' '||P.weight||' '||P.city);
    end loop;
  end loop;
end;
 17   18   19   20  /
Enter value for name: Rodelo
old   9:   open S('&name');
new   9:   open S('Rodelo');
S4   Rodelo 20 London
P2   Bolt Green 17 Paris
P3   Screw Blue 17 Oslo
P4   Screw Red  14 London
P5   Cam Blue 12 Paris
P6   Cog Red 19 London

PL/SQL procedure successfully completed.
```

Suppliers

S#	NAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	30	Paris
S3	Blake	30	Paris
S4	Rodelo	20	London
S5	Adams	30	Athens

Parts

P#	NAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

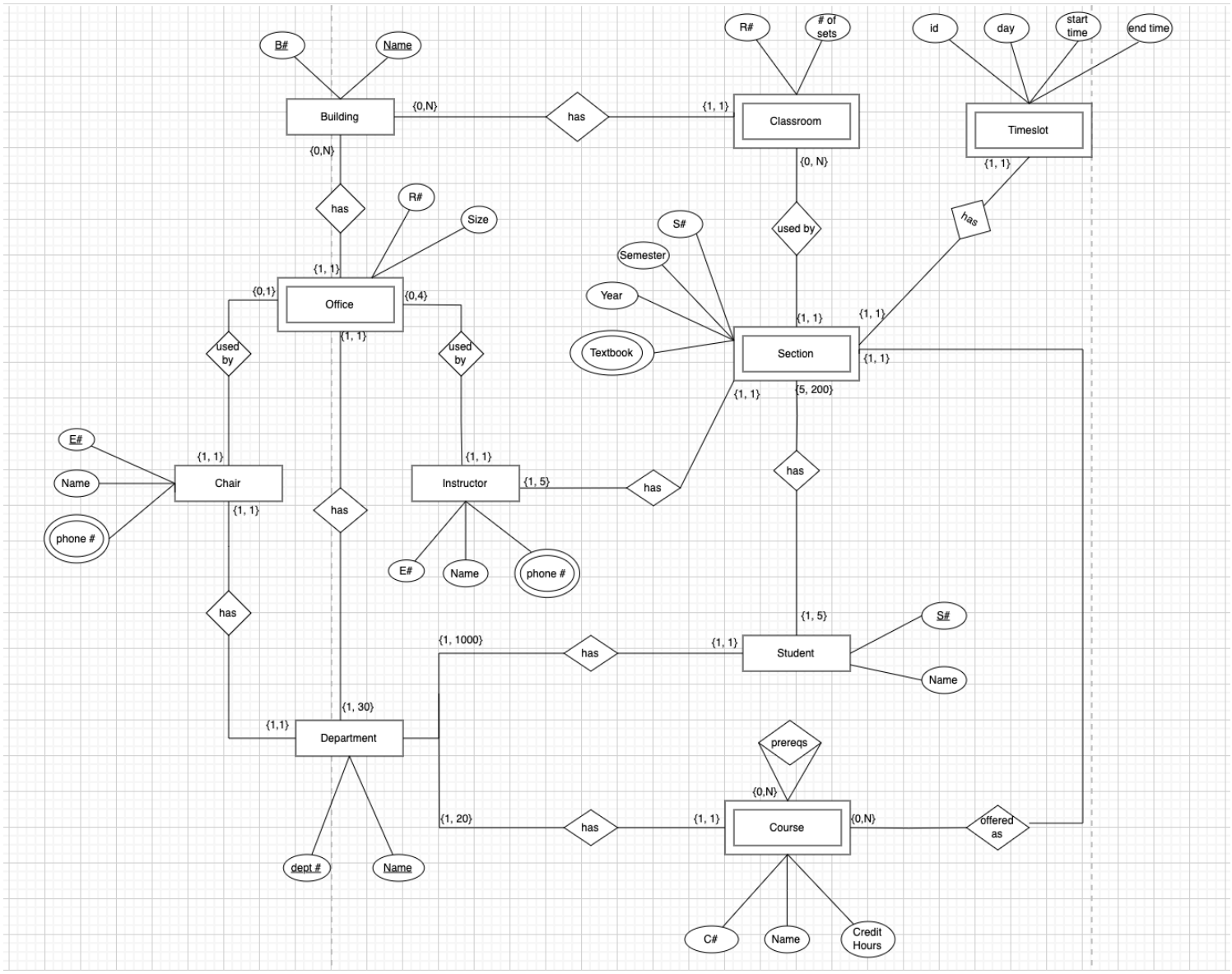
SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P3	300
S4	P4	400
S4	P5	500
S4	P6	600

Part 3 ER Model (40 marks)

A university information system involves buildings, classrooms, offices, departments, courses, sections, timeslot, chairs, instructors, and students.

a) A building has a unique building number such as HP, a unique name, and a number of classrooms and offices.
b) A classroom has a room number such as 5125 that is unique in the building, the number of seats, and is either empty or used by a number of sections at different days and times.
c) An office has a room number that is unique in the building, the size in square feet, and is either empty or occupied by a chair or up to 4 instructors.
d) A department has a unique dept code such as COMP, a unique name, 1 chair, 1 to 40 instructors, 1 to 1000 students, 1 to 20 courses, 1 to 30 offices in the same or different buildings, and no offices are shared by different departments.
e) A course has a unique course number such as 3005 and a name such as Databases that are unique in the department that offers the course, credit hours, and a number of prerequisite courses. Courses are offered as sections and not all courses are offered.
f) A section has a unique section code A, B, C, D, or E *within* the course, semester, year, classroom, timeslot, and textbooks, and is related to one course, one instructor, and 5 to 200 students. Just consider current sections only.
g) A timeslot has a timeslot id, day, start time, and end time.
h) A chair or an instructor has a unique employee number, a name, an office, 0 to 2 phone numbers, and can only work in one department. Note that a chair is not an instructor, and vice versa. An instructor teaches 1-5 sections.
i) A student has a unique student number, a name, majors in one department, takes 1 to 5 sections, and has a grade for each section.
j) Draw the ER diagram for this information system that can represent the constraints specified above. You can use free **draw.io** to do this part.


B#
Name
Building
{0,N}
has
{1, 1}
Classroom
R#
# of sets
id
day
start time
end time
Timeslot
{1, 1}
{0, N}
used by
has
R#
Size
{1, 1}
Office
{0,1}
{0,4}
{1, 1}
used by
used by
S#
Semester
Year
Textbook
Section
{1, 1}
{1, 1}
{1, 1}
{5, 200}
{1, 1}
E#
Name
phone #
{1, 1}
Chair
{1, 1}
Instructor
{1, 5}
has
has
has
E#
Name
phone #
has
{1, 5}
S#
Name
{1, 1000}
has
{1, 1}
Student
{1, 30}
{1,1}
Department
prereqs
{0,N}
{1, 20}
has
{1, 1}
Course
{0,N}
offered as
dept #
Name
C#
Name
Credit Hours