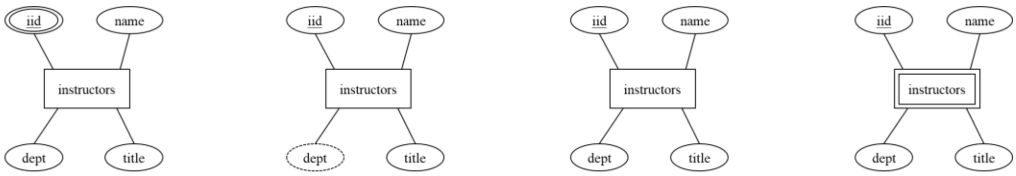
Problem ONE: ER Model (16 points)

Consider an academic office database that maintains data about the following four entities: (a) courses, having unique course number, and other simple attributes: title, credits, syllabus; (b) students, having unique student-id and other simple attributes: name and program; (c) instructors, having unique identification number, and other simple attributes: name, department, and title. (d) course offerings, having simple attributes: course number, year, semester, section number, timings, and classroom;

Based on the above description, please answer the following questions about ER diagram of this database.

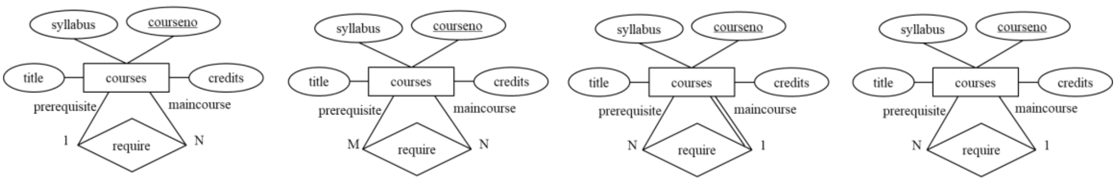
1 Which of the following is the correct notation for the entity type *instructors*. [4 points]



A B C D

Answer: C

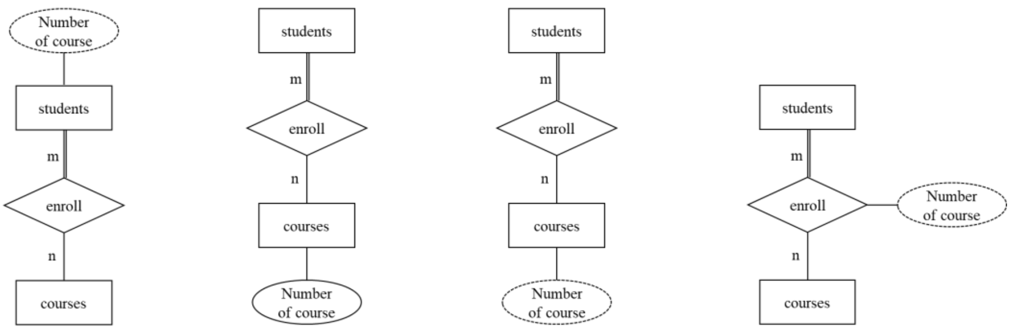
2 Suppose each course may have zero or many prerequisite courses and a course may be a prerequisite of many courses, which of the following is the correct notation for *courses*. [4 points]



A B C D

Answer: B

3 If we want to add the attribute ‘number of courses’ to indicate the number of courses enrolled by a student, which of the following notation is correct? [4 points]



A B C D

Answer: A

4 Please give the definition of the weak entity. Do we need to define a weak entity set in the ER diagram of this database? If YES, please point the weak entity set out and also its identifying relationship.[4 Points]

Answer:

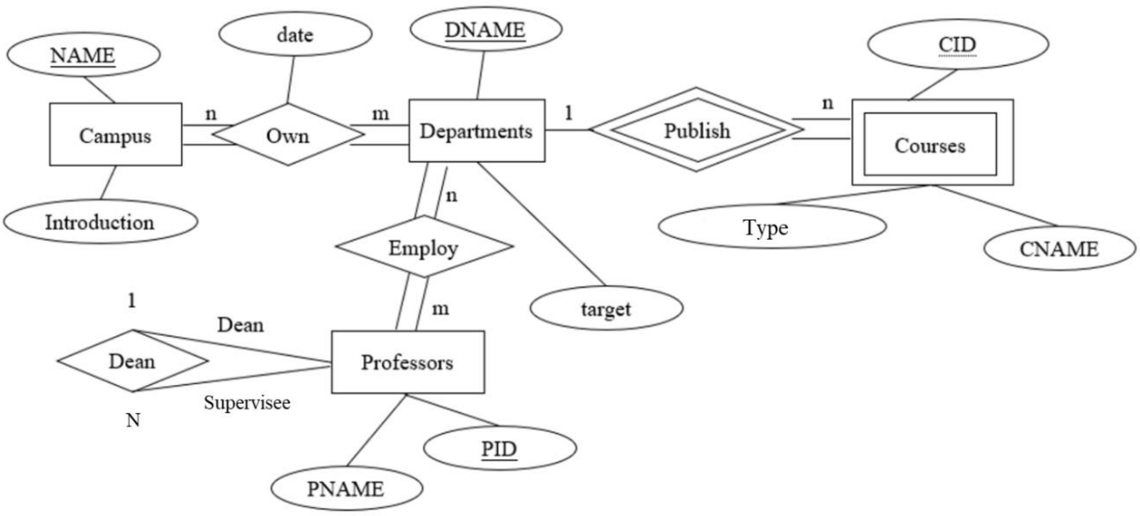
Weak Entity Set is an entity set that does NOT have enough attributes to form a primary/candidate key. [2 pts]

The entity set *course-offering* is a weak entity. [1 pts]

Its identifying relationship is between *course-offering and course.* [1 pts]

Problem TWO: Relational Model (17 points)

1 Please convert the following completed ER diagram into Relation Model.(12 points)



Note: you can define a relation in this format: **Tablename (attribute1, attribute2, attribute3,…)** and indicate the reference of foreign key by an arrow in this way: **TableA.attribute1 → TableB.attribute2**, which means TableA.attribute1 references to TableB.attribute2.

Answer:

//Marking tips: 1 points for each line.

Campus （Name，Introduction）

Own（NAME， DNAME， Date）

Departments (DNAME, Target)

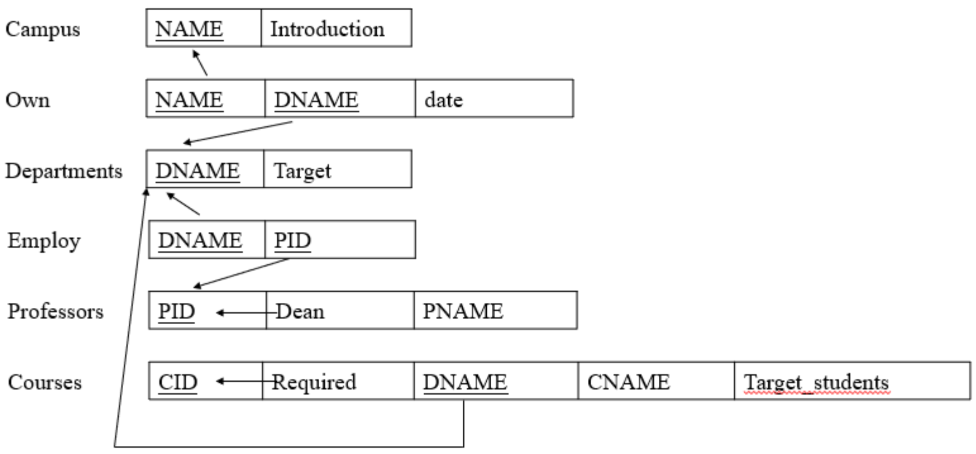
Employ (DNAME, PID)

Professors (PID, Dean, PNAME)

Course (CID, DNAME, CNAME, Type)

Own.NAME→ Campus.NAME

Own.DNAME→ Department.DNAME
Employ.DNAME→ Department.DNAME
Employ.PID→ Professor.PID
Professors.Dean→ Professor.PID
Course.DNAME→ Department.DNAME



2 Explain the distinctions among the terms primary key, candidate key, and superkey. [5 points]

Answer:
A superkey is a set of one or more attributes which, taken together, identify uniquely an entity in an entity set; A superkey for which no proper subset is also a superkey is called a candidate key, which means candidate key is the minimal set of attributes which can identify uniquely an entity in an entity set; The primary key is one of the candidate keys that is chosen by the database designer as the principal means of identifying entities within an entity set.

Problem Three: Integrity Constraints (15 points)

Suppose we have a relational database containing three tables Student(SSN, Name, Email, Sex, DeptName), Department(DeptName, Address, Phone, College_ID, DeptHead, Head_ID) and Employee(Emp_ID, Name, BirthData, Phone). The current state of the database is shown in the following tables.

Student

SSN	Name	Email	Sex	DeptName
004-73-2862	Otis West	wcikizank@fileze.site	M	ME
141-45-2946	Flynn Crane	iremas@caraparcal.com	F	MATH
056-38-9627	Michael Scott	1paradiswinx@bestlibfiles.site	M	MKT

092-99-0173	Dan Anthony	nimed.grait.9u@scaptiean.com	F	MKT
654-58-6672	Kelly Kapoor	mkume@wingkobabat.buzz	F	CS
758-40-0263	Celia Hamer	j963@how1x.site	M	ME
164-20-5327	Josh Porter	mabdeljalil.bouag@oreple.com	F	LT
565-61-4682	Flynn Crane	qanoramozag@ilrlb.com	M	MATH
041-17-4051	Jim Halpert	6dorg7amq@gumaygo.com	M	LT

Department

DeptName	Address	Phone	College_ID	DeptHead	Head_ID
CS	20 Joy Ridge Lane Maumee, OH	718-843-1586	CENG	Alicja Weir	5
ME	13 County Drive Columbia, MD	954-800-4359	CENG	Alicja Weir	2
MKT	33 New Saddle Court Lapeer, MI	816-355-7166	CB	Vernon Robin	3
MATH	944 E. Ocean St. Trenton, NJ	727-906-5989	CSCI	Beauden Lang	1
LT	156 Thorne St. Winter Springs, FL	267-636-4719	CLASS	Khadijah Holman	4

Employee

Emp_ID	Name	BirthDate	Phone
2	Alicja Weir	1969-09-05	517-787-1793
4	Lackawana Country	1958-02-19	857-540-7155
5	Alicja Weir	1964-03-15	405-886-6224
3	Vernon Robin	1978-10-01	912-361-2453
1	Beauden Lang	1973-07-22	440-762-0734
8	Manahil Rodriquez	1967-11-17	812-536-2258

1 List the primary key and the foreign key(s) for each relation. And calculate the number of superkey(s) of each relation. [5 points]

Answer:

For Student table:

Primary key: SSN

Superkey number: 2^4=16 （1 mark）

For Department table:

Primary key: DeptName

Superkey number: 2^5=32 （1 mark）

For Employee table:

Primary key: Emp_ID

Superkey number: 2^3=8 （1 mark）

Foreign key: student.DeptName, department.Head_ID （2 mark）

2 For the questions below, suppose each of the following Update operations is applied directly to the database. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.

a. Insert <’296-67-9428’,’Peter Chan’,’pechan@gmail.com’,100,’EE’ > into Student [5 points]

Answer:

Violates both referential integrity constraint and domain constraint. Violates referential integrity because DeptName is foreign key, and there is no tuple in the Department relation with DeptName='EE'. Violates domain constraint because the domain of Sex should not be integer. (3 points)

We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of DeptName to an existing value in Department, or (iii) inserting a new Department tuple with DeptName='EE'. Meanwhile changing the Sex into char type. (2 points)

b. Update Emp_ID and Phone of Employee tuple with Emp_ID=1 to 3 and null respectively. [5 points]

Answer:

Violates the primary key constraint and the referential integrity constraint. Violates the key constraint because there already exists an Employee tuple with Emp_ID=3. Violates the referential integrity constraint because once the table is updated, no corresponding referenced tuple could be found for the tuple of Department with Head_ID=1. (3 points)

We may enforce the constraint by: (i) rejecting the modifying operation, or (ii) changing the value of Emp_ID to a value that is not null and doesn’t exist in the table Employee. At the same time, we need to change the Head_ID of Department to a value that exists in Employee table. (2 points)

Problem FOUR: Normal forms (16 points)

Let’s consider the following relationship R storing the information about various seminars.
R(SeminarNo, OfferingDeptNo, OfferingDeptName, Year, RoomNo, Address, RoomSize, InstructorID)
It has following functional dependencies:

SeminarNo → OfferingDeptNo
OfferingDeptNo → OfferingDeptName
{SeminarNo, Year} → {RoomNo, Address, InstructorID}
{RoomNo, Address}→ RoomSize

(1) Identify all the candidate keys in this table. [4 Points]

Answer:

{SeminarNo, Year}

(2) Is the relation R in 2NF and why? If not, decompose it into TWO tables which satisfy 2NF. [6 Points]

Answer: Not in 2NF, because there exists partial function dependency on primary keys, SeminarNo → OfferingDeptNo. (2 marks)

R1 (SeminarNo, Year, RoomNo, Address, RoomSize, InstructorID) (2 marks)
R2 (SeminarNo, OfferingDeptNo, OfferingDeptName) (2 marks)

(3) Does your decomposition in (2) satisfy 3NF and why? If not, normalize it into 3NF. [6 Points]

Answer: Not in 3NF, because there exists transitive function dependency on primary keys: OfferingDeptNo → OfferingDeptName, {RoomNo, Address}→ RoomSize (2 marks)

R1A (SeminarNo, Year, RoomNo, Address, InstructorID) (1 marks)
R1B (RoomNo, Address, RoomSize) (1 marks)
R2A (SeminarNo, OfferingDeptNo) (1 marks)
R2B (OfferingDeptNo, OfferingDeptName) (1 marks)

Problem FIVE: SQL I (18 points)

Considering a film database containing following three tables and write **SQL** statement for each question.

User (**user_id** : integer, **user_name**:string)
Film_likes (**user_id** : integer, **film_id** : integer)
Film (**film_id** : integer, **film_name**:string)

Notes: **Film_likes.user_id** is a foreign key referencing **user.user_id**, wile **Film_likes.film_id** is a foreign key referencing **film.film_id**. A tuple in **Film_likes** table represents one user likes one film.

1 Create these three tables and define their integrity constraints with SQL statements. (8 points)

Answer:

CREATE TABLE User(

```
user_id number,
user_name char(32),
PRIMARY KEY (user_id, follower_id)); （2 points）

CREATE TABLE Film (
film_id number,
film_name char(32),
PRIMARY KEY (id));    （2 points）


CREATE TABLE Film_likes (
film_id number,
film_name char(32),
PRIMARY KEY (id));    （2 points）

CREATE TABLE FILM_likes (
user_id number,
music_id number
PRIMARY KEY (user_id,film_id),
FOREIGN KEY (user_id) REFERENCES(User(user_id)) ,
FOREIGN KEY (film_id) REFERENCES(Film(film_id))
); (4 points）
```

2 Retrieve name of users who likes the film Avatar or Titanic. (5 points)

Answer:

```
select user_name
from Film_likes, User
where User.user_id= Film_likes .user_id and (film_name=’Avatar’ OR film_name=’Titanic’);
```

3 Retrieve name of films which are not liked by any user. (5 points)

Answer:

```
select film_name
from Film
where NOT EXIST ( select *
                  from Film_likes
                  where Film.film_id= Film_likes.film_id);
```

Problem SIX: SQL II (18 points)

Given the following relations about the information of courses in a university.

Student (**StudentID**: integer, **Name**: string, **Age**: integer, **Gender**: string)

Course (**CourseID**: integer, **Name**: string, **TeacherID**: integer)

Teacher (**TeacherID**: integer, **Name**: string)

Grade (**<u>StudentID</u>**: integer, **<u>CourseID</u>**: integer, **Score**: integer)
Suppose now we have a valid database state. Write the following queries in SQL.

1 Count the number of courses taught by each teacher. Order the results in the descending order of teacher's ID. [6 points]. (6 points)

Answer:

```
SELECT Teacher.TeacherID, COUNT(*)
FROM Teacher, Course,
WHERE   Teacher.TeacherID= Course.TeacherID
GROUP BY Teacher.TeacherID;
ORDER BY Teacher.TeacherID DESC;
```

2 Find the ID of students who did not enroll in the course with CourseID=101 but enroll in the course with CourseID=102. (6 points)

Answer:

```
SELECT DISTINCT StudentID
FROM Grade
WHERE Grade.StudentID NOT IN (SELECT StudentID FROM Grade WHERE
Grade.CourseID=101)
AND Grade.CourseID =102;
```

OR

```
(SELECT DISTINCT StudentID
FROM Grade
WHERE Grade.CourseID=102)
MINUS
(SELECT DISTINCT StudentID
FROM Grade
WHERE Grade.CourseID=101);
```

3 Query the ID of students who did not enroll in all of the courses. (6 points)

Answer: Problem Two: Relational Model

```
SELECT Student. StudentID
FROM Student
WHERE Student.StudentID NOT IN (SELECT Grade.StudentID
                                 FROM GRADE
                                GROUP BY Grade.StudentID
                                HAVING COUNT(Grade.CourseID) =
                          (SELECT COUNT(CourseID) from Course));
```