A short introduction to ER & SQL

Bipin C. DESAI

Database Languages

- A Database Management System (DBMS) provides two types of languages; they may also be viewed as components of the DBMS language:
  - Data Definition Language (DDL)
    - Language (notation) for *defining* and modifying a database schema
    - It includes syntax for declaring tables, indexes, views, constraints, etc.)
  - Data Manipulation Language (DML)
    - Language for *accessing* and *manipulating* the data (organized/stored according to the appropriate data model)

Query Languages

- Theoretical:
  Relational Algebra, Relational Calculus, Datalog
- Commercial: SQL
- First there were two: SEQUEL (Ingres) and SQL(R)
- SQL developed originally at IBM in 1976
  - First standard: SQL-86
  - Second standard: SQL-92
  - Latest standard: SQL-99, or SQL3,
    SQL3 standard has over 1,000 pages of document
- SQL is the de-facto standard for RDBMS
- The SQL query language components:
  - DDL (e.g., create)
  - DML(e.g., select, insert, update, delete)

Simple SQL Queries

A SQL query has a form**:**

SELECT . . .

FROM . . .

WHERE . . . ;

The **SELECT** clause defines the schema of the result

The **FROM** clause gives the source relation(s) of the query

The **WHERE** clause is one or more predicates to 'select" the tuples of interest.

The query result is a relation and it is **unnamed**.

Example “theoretical” SQL Query

Relation schema:

Course (<u>Cno</u>, Cname, credits)

Query in natural language (English):

Find all the courses stored in the database

Query in SQL:

SELECT ⋆

FROM Course;

Here the “ ⋆ “ in “**SELECT** ⋆” means **all** attributes in the **relation(s)** involved.

More Examples SQL Query

- Relation schema:

  Movie (title, year, length, filmType)

Query in natural language (English):

  Find the titles of all movies stored in the database

Query in SQL:

```
SELECT title
FROM Movie;
```

Relation schema:

  Student (SID, FirstName, LastName, Address, GPA)

Query in natural language (English):

  Find the SID of every student whose GPA is greater than 3

Query in SQL:

```
SELECT SID
FROM Student
WHERE GPA > 3;
```

The "WHERE" clause

The expressions that may follow **WHERE** are conditions

Standard comparison operators **θ** includes { =, <>, <, >, <=, >= }

The values that may be compared include constants and attributes of the relation(s) mentioned in **FROM** clause

Simple expression

A θ Value

A θ B

Where **A**, **B** are attributes and

θ is a comparison operator

We may also apply the usual arithmetic operators, +,-,*,/, etc. to numeric values before comparing them

(year - 1930) * (year - 1930) < 100

The result of a comparison is a Boolean value **TRUE** or **FALSE**

Boolean expressions can be combined by the logical operators **AND**, **OR**, and **NOT**

- Relation schema: **Movie** (<u>title</u>, <u>year</u>, length, filmType)
- Query: Find the titles of all color movies produced in 1950
- Query in SQL:

```
SELECT title
FROM Movie
WHERE filmType = 'color' AND year = 1950;
```

- Query: Find the titles of color movies that are either made after 1970 or are less than 90 minutes long
- Query in SQL:

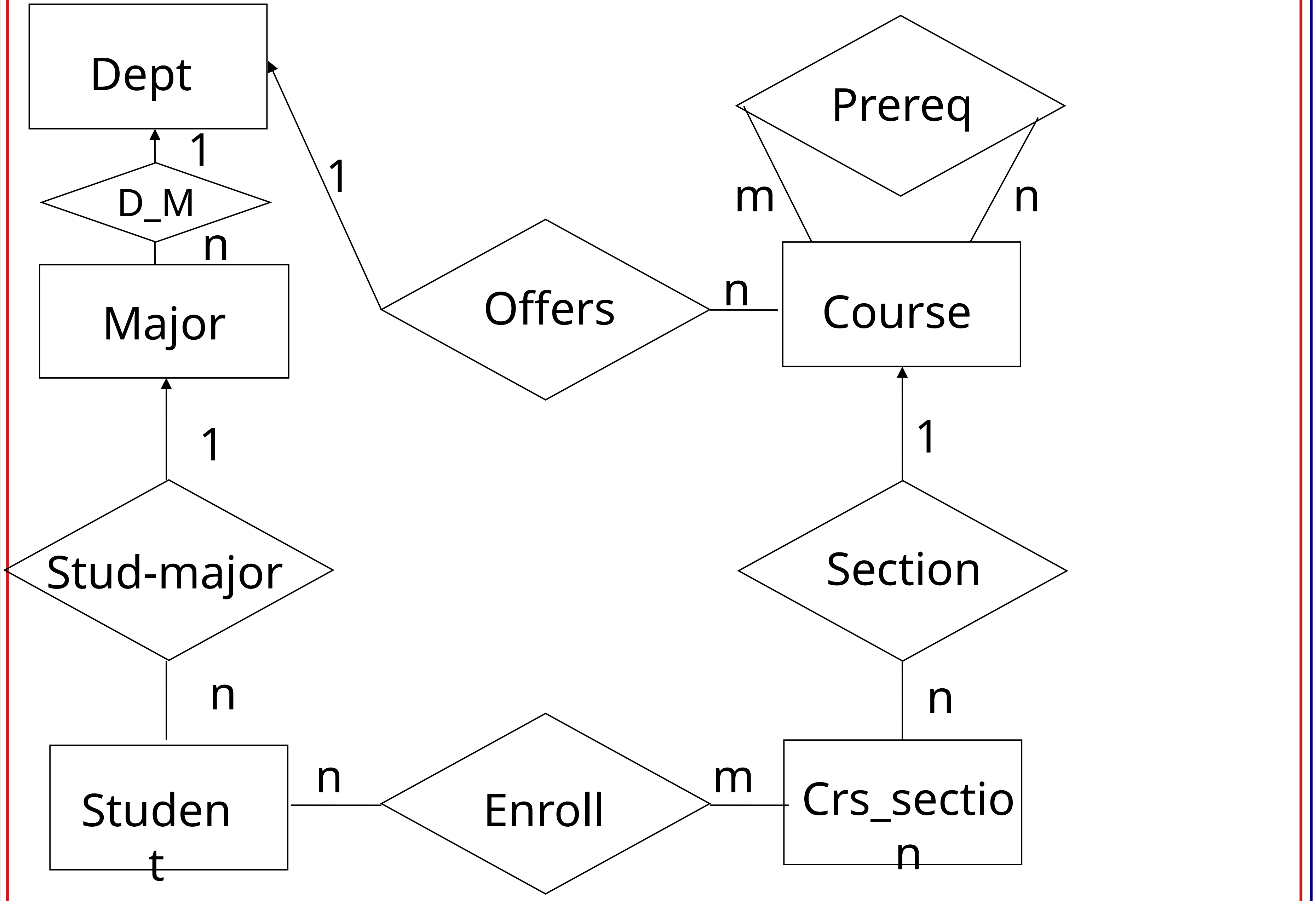
```
SELECT title
FROM Movie
WHERE (year > 1970 OR length < 90) AND filmType =
   'color';
```

Note the precedence rules, when parentheses are absent:

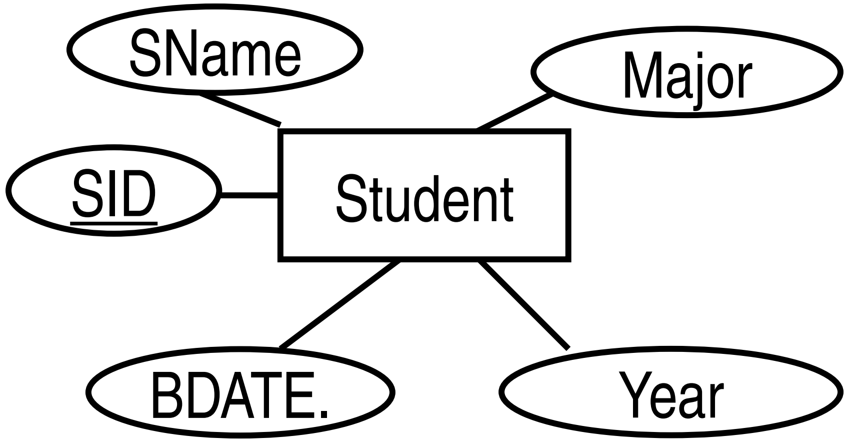
AND takes precedence over **OR**,

and **NOT** takes precedence over both

An example of using SQL from E-R to RDBMS


Dept
1
D_M
n
Major
1
Stud-major
n
Studen
t
1
Offers
n
Prereq
m
n
Course
1
Section
n
Crs_sectio
n
m
Enroll
n

SQL DDL example:



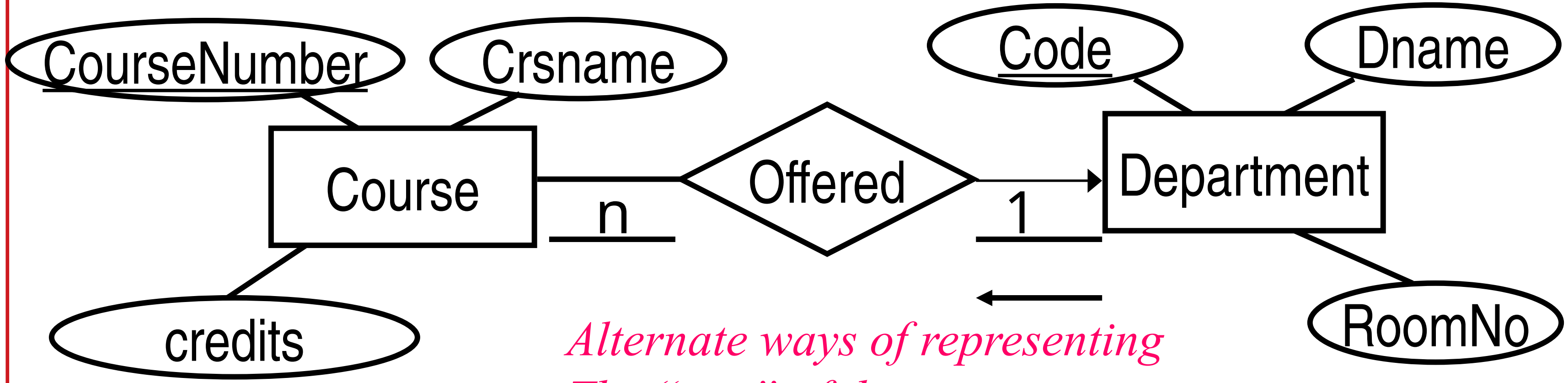
```
create table student
(SID NUMBER(3) primary key not null,
 SNAME VARCHAR2(20),
 MAJOR CHAR(4),
 YEAR NUMBER(1),
 BDATE DATE)
/
```

```
insert into student values
(8,'Brenda','COMP','2','13-AUG-77');
(8,'Brenda','COMP',',‘1977-08-13');
insert into student values
(10,‘Mary','ENGL','1','13-MAY-80');
(10,‘Mary','ENGL','1',‘1980-5-13’);
insert into student values
(13,'Keily','SENG','4','12-AUG-80');
insert into student values
(14,‘Jack','CSAP','1','12-FEB-77');
```

Many to one relationship

A department offers many courses
A given course can be offered by only one department

There is no standard regarding the direction of arrow for the "one" entity.



Alternate ways of representing The "one" of the many-to-one relationship; arrow either pointing to entity on the "one side" or pointing to the relationship

```
create table course
(COURSE_NAME CHAR(20),
 COURSE_NUMBER CHAR(8) primary key NOT
NULL,
 CREDIT_HOURS NUMBER(2),
 OFFERING_DEPT CHAR(4))
 tablespace TUTOR pctfree 5
/
```

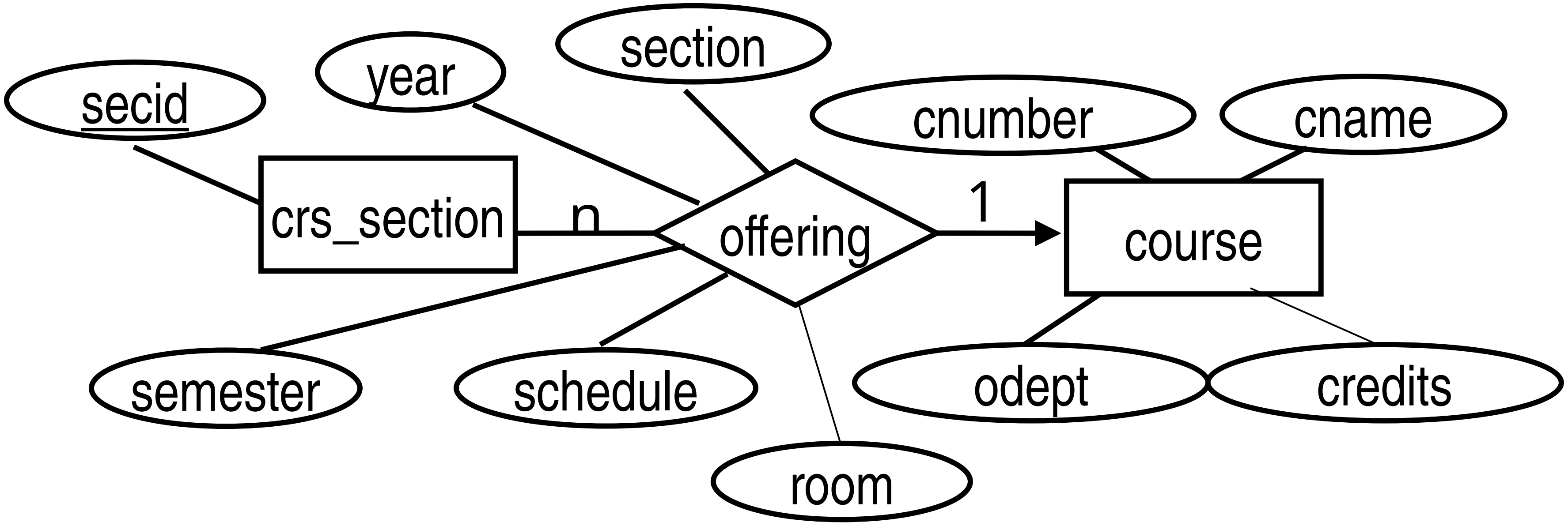
Note how the relationship w/o an attribute is “merged” with one of the entity

```
insert into course values
('C++','COMP248',3,'COMP');

insert into course values
('DATA STRUCTURES ','COMP352',3,'COMP');

insert into course values
('OPERATING SYSTEMS','COMP346',4,'COMP');

insert into course values
('DATABASE','COMP353',4,'COMP');
```



NB: Here course section (crssection) is really a “weak” entity; However, in most cases it is promoted a “strong” entity by introducing an identifying key attribute section ID (secid)

```
create table crs_section
(SECID NUMBER(6) primary key NOT NULL,
 COURSE_NUM CHAR(8),
 SECTION CHAR(2),
 SEMESTER CHAR(4),
 YEAR CHAR(4),
 SCHEDULE CHAR(10),
 ROOM CHAR(7))
 tablespace TUTOR pctfree 2
/
```

Note: We have replaced an entity and the relationship With a single relation

```
insert into crs_section values
(85,'COMP352','A','FALL', '1998','TH16001715','H123');
insert into crs_section values
(90,'COMP353','B','FALL','1999','MW08451000','H631');
```

```
create table enrolment
(STUDENT_NUMBER NUMBER(3) not null,
 SECTION_ID NUMBER(6) not null,
 GRADE CHAR(1),
 primary key(student_number, section_id))
tablespace TUTOR pctfree
/
```

```
insert into enrolment values(8,85,null);
insert into enrolment values(10,90,null);
insert into enrolment values(8,90,null);
insert into enrolment values(14,90,null);
```

Find details of studs. taking a course offered by the “DISC” dept.

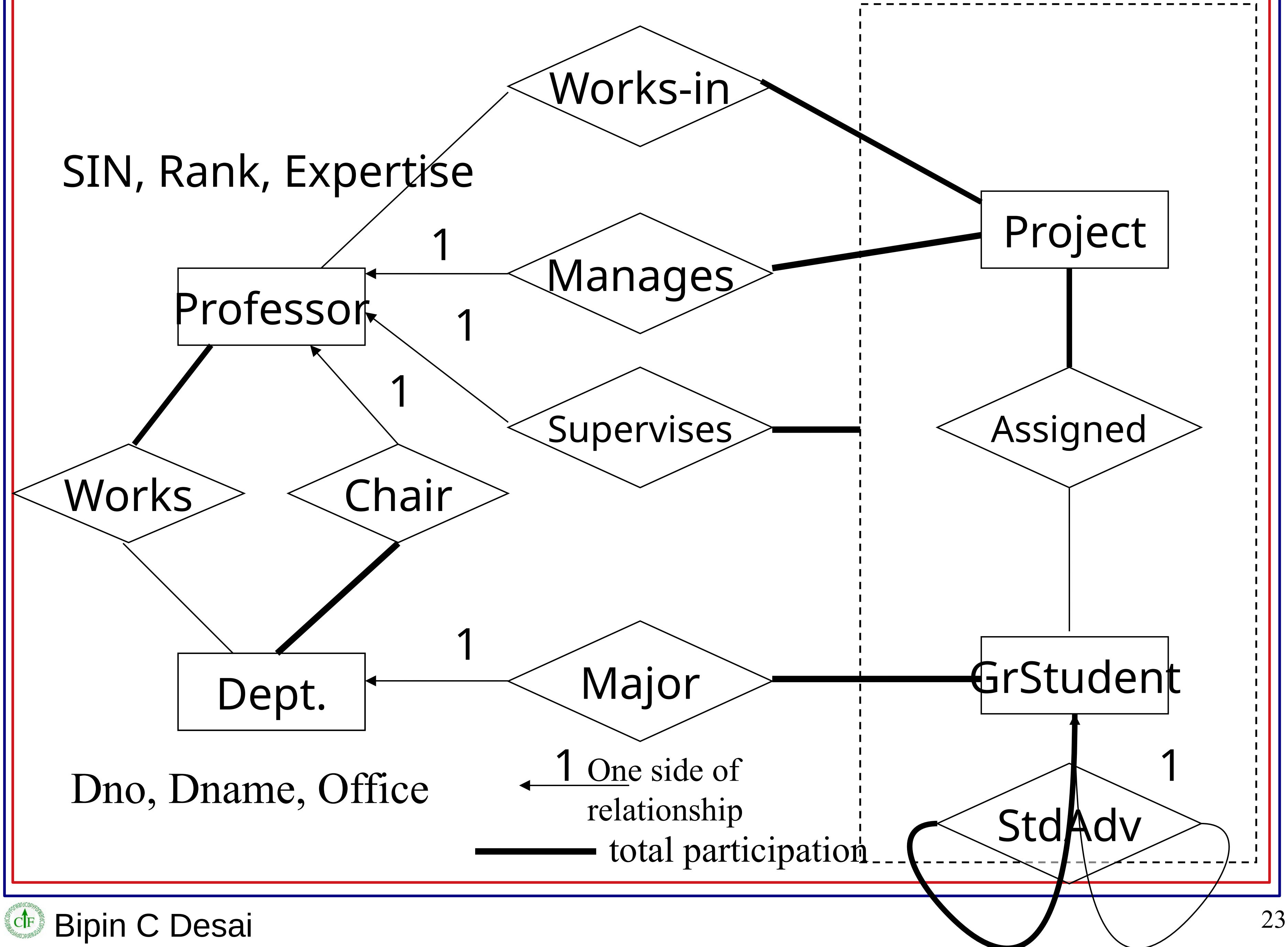
```
select s.SID, s.SNAME, s.MAJOR, s.YEAR, s.BDATE
from student s, dept d, course c, crs_section r, enrolment e
where  c.ODEPT=d.CODE and
       r.COURSE_NUM=c.CNUMBER and
       r.SECID=e.SECTION_ID and
       e.STUDENT_NUMBER = s.SID and
       d.CODE= 'DISC';
```

```
SID  SNAME           MAJOR YEAR BDATE
---- ------------ ----- ---- ------------
 1,4 Jack             CSAP     1 12-FEB-77
```

More examples of using E-R modeling

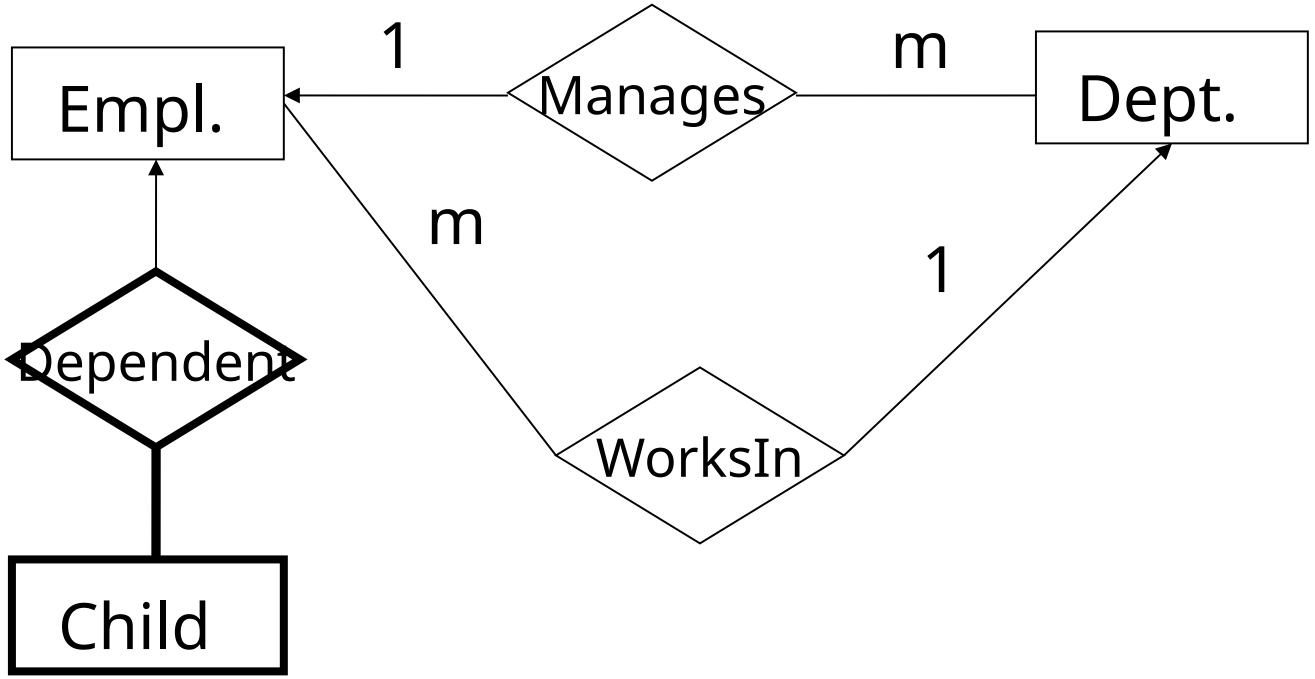
© Bipin C. Desai

Professors have a SIN, a name, an age, a rank, and a research specialty.
Projects have a project number, a sponsor name (e.g., NSERC),
a starting date, an ending date, and a budget.
Graduate students have a SIN, a name, an age, and a degree program
(e.g., M.S. or Ph. D.).
Each project is managed by a professor (principal investigator).
Each project is worked on by one or more professors (co-investigators).
Professors can manage and/or work on multiple projects.
Each project is worked on by one or more graduate students (research assistants).
When a graduate student works on a project, is supervised by a
participating professor.
Graduate students can work on multiple projects.
Departments have a department number, a department name, and a main office.
Departments have a chairman who runs the department.
Professors work in one or more departments(%time)
Graduate students have one major department for their degree.
Each graduate student has another, more senior graduate student
(student advisor)

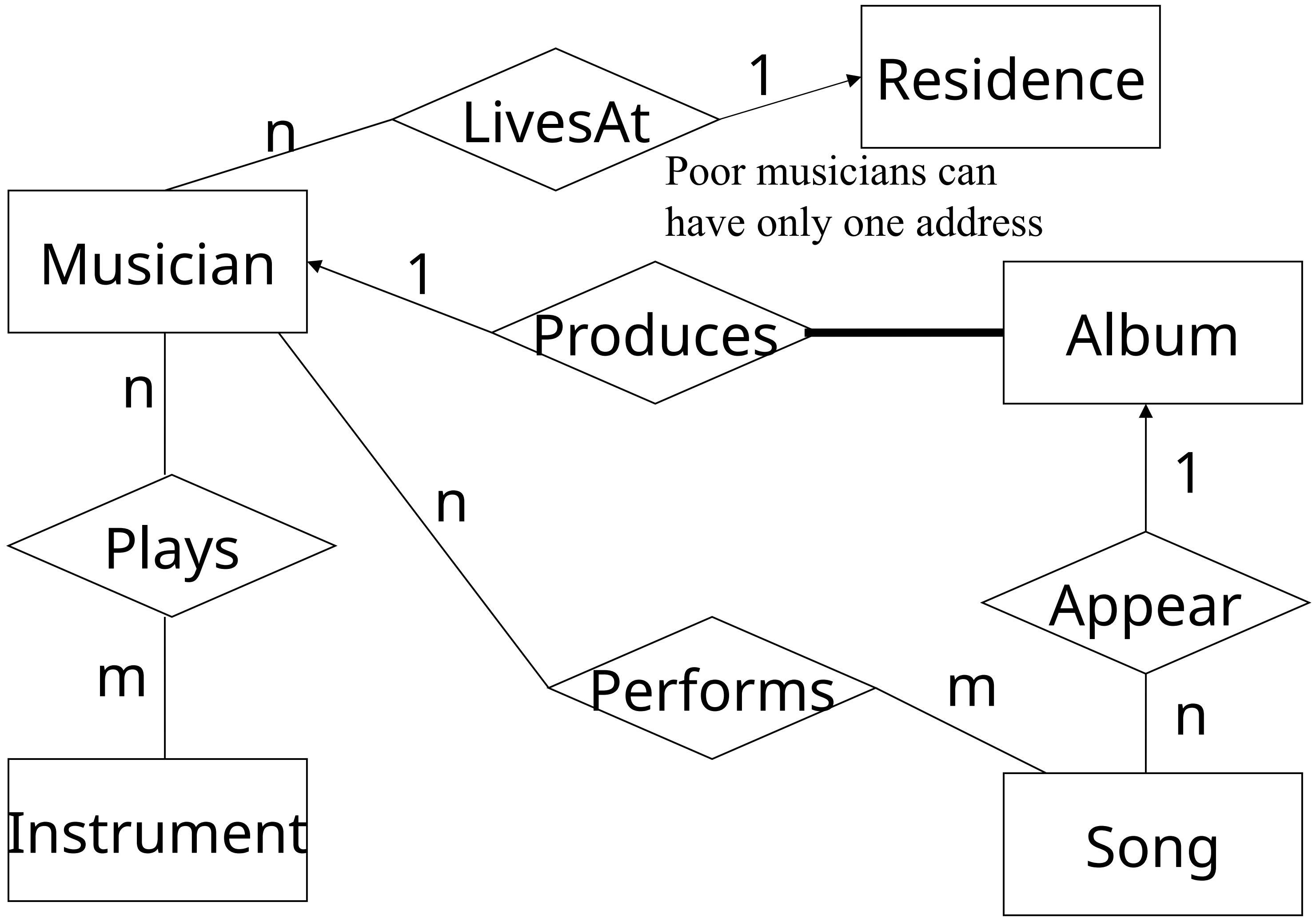

Works-in
SIN, Rank, Expertise
Project
1
Manages
Professor
1
1
Supervises
Assigned
Works
Chair
1
Major
GrStudent
Dept.
1
Dno, Dname, Office
1 One side of relationship
total participation
StdAdv

In a company database, you need to store information about

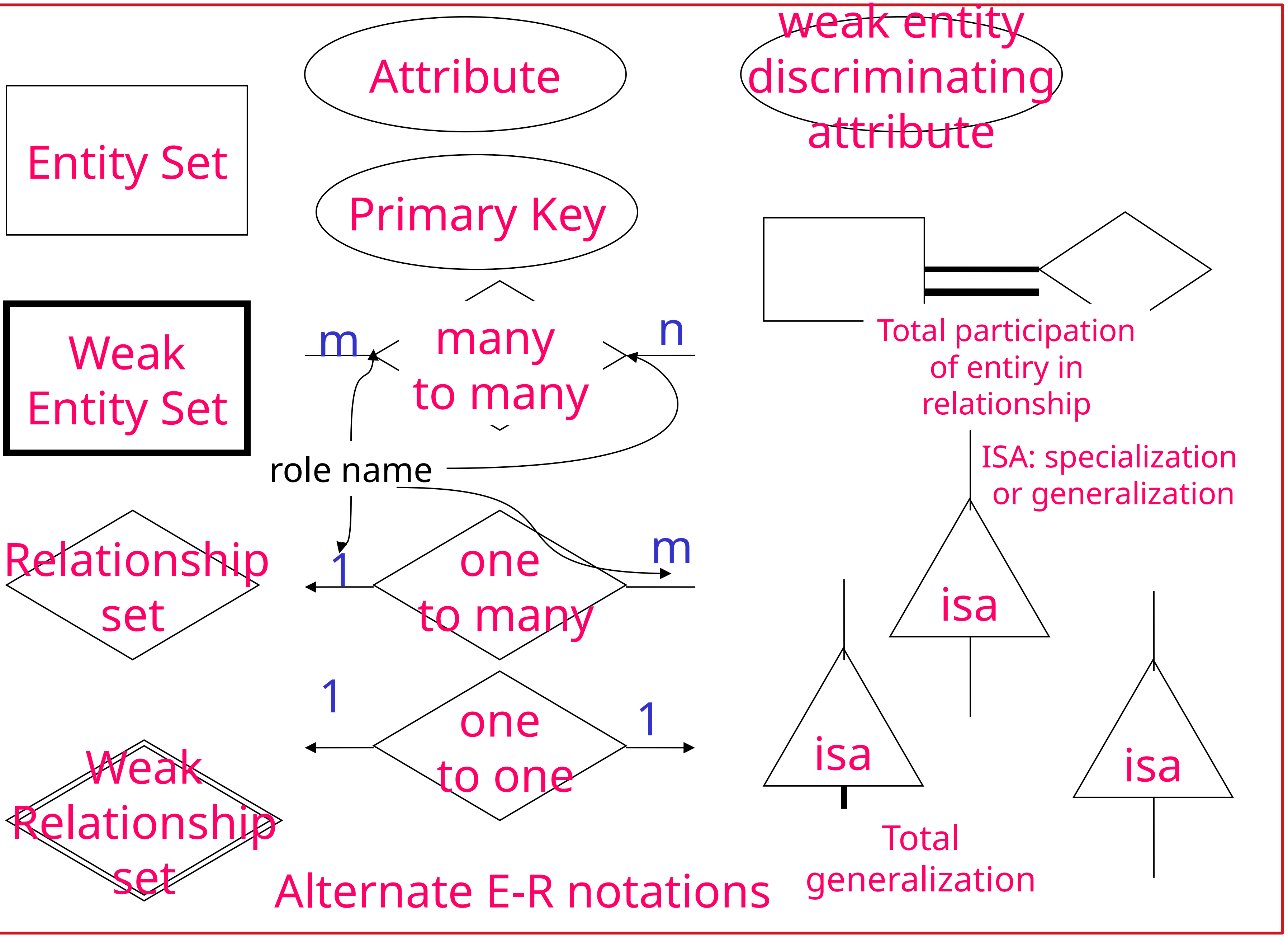
employees (SIN,salary and phone),
departments (dno, dname, budget), and
children of employees (with name and age as attributes).
Employees work in departments;
Each employee works in only one department;
A department could have many employees;
Each department is managed by an employee;
Each department has only one manager(an employee);
A manager could manage many departments
A child can only be identified by name
An employee has only one child with a given name
only one parent can declare a child as a dependent


Empl.
1
Manages
m
Dept.
m
1
Dependent
WorksIn
Child

Each musician that records at Notown has an SIN, a name,
an address, and a phone number.
Musicians often share the same address,
no address has more than one phone.
Each instrument that is used in songs recorded at Notown has a name and
a musical key
Each album that is recorded on has a title, a copyright date, and
an album identifier.
Each song recorded has a title and an author.
Each musician may play several instruments, and
a given instrument may b e played by several musicians.
Each album has a number of songs on it,
but no song may appear on more than one album.
Each song is performed by one or more musicians, and
a musician may perform a number of songs.
For each album, there is exactly one musician that acts as its producer.
A musician may produce several albums.

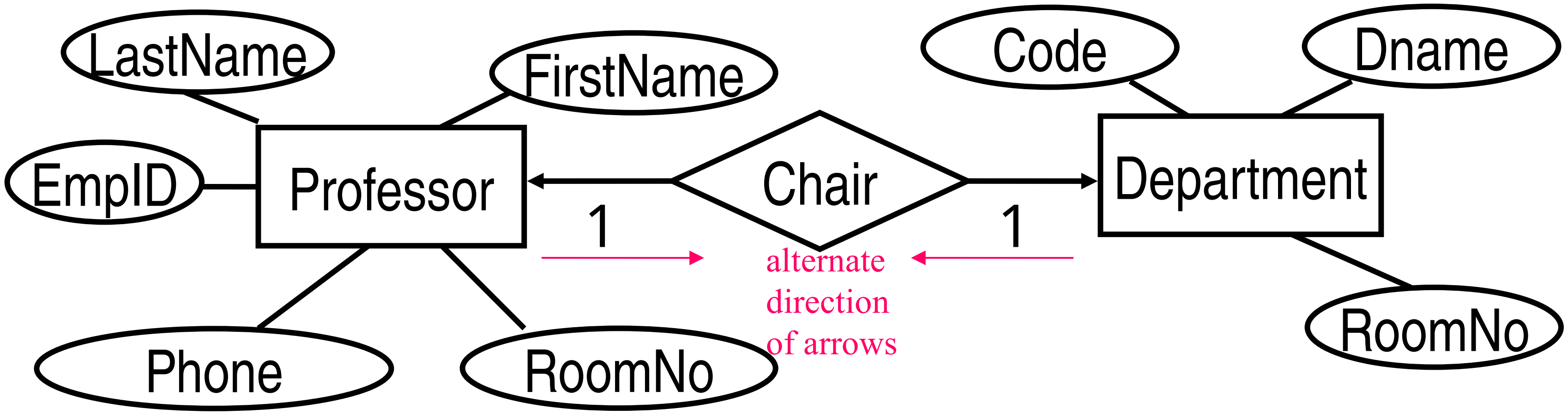


Notes: Since a songs must appear on only one album, Appear is a many to one relationship. Similarly for Produces. Album requires total participation in Produces. Some songs may not be recorded and there may be some instruments that nobody can play!



Alternate E-R notations

- A **one-one** relationship between **Department** and **its** chair (a dept. has one chair and a prof. is the chair of at most one dept) is represented by
  - arrows pointing to both Department and Professor or
  - indicated by a line with the number 1 on it.
  - Sometimes the arrow is in the opposite direction(pointing to the diamond)

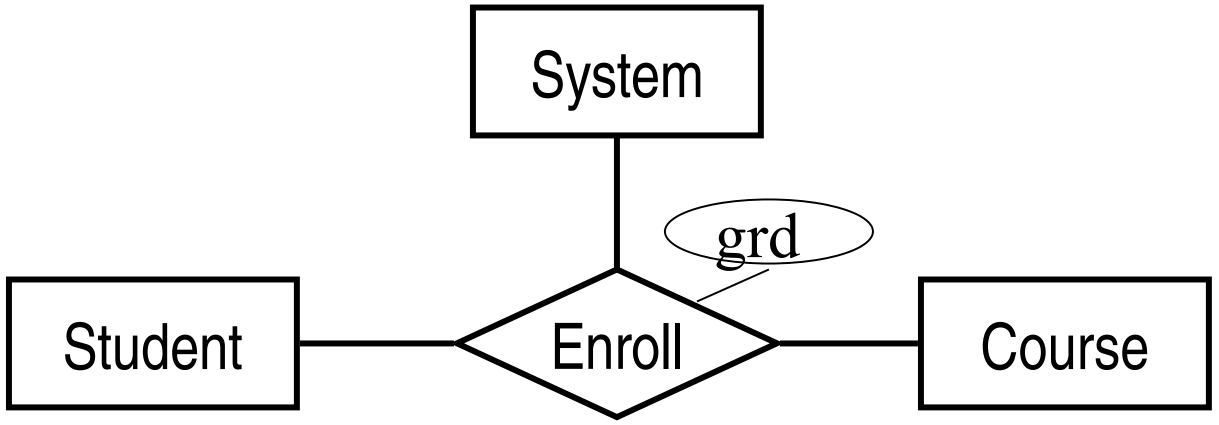


ODL allows only **binary** relationships, i.e., relationships involving two classes.

E/R model makes it convenient to define **(n-ary)** relationships – relationships involving n entity sets

A **n-ary** relationship in an **E/R** diagram is represented by lines from the relationship (diamond) to each of the participating entity sets (rectangles).

N-ary Relationships

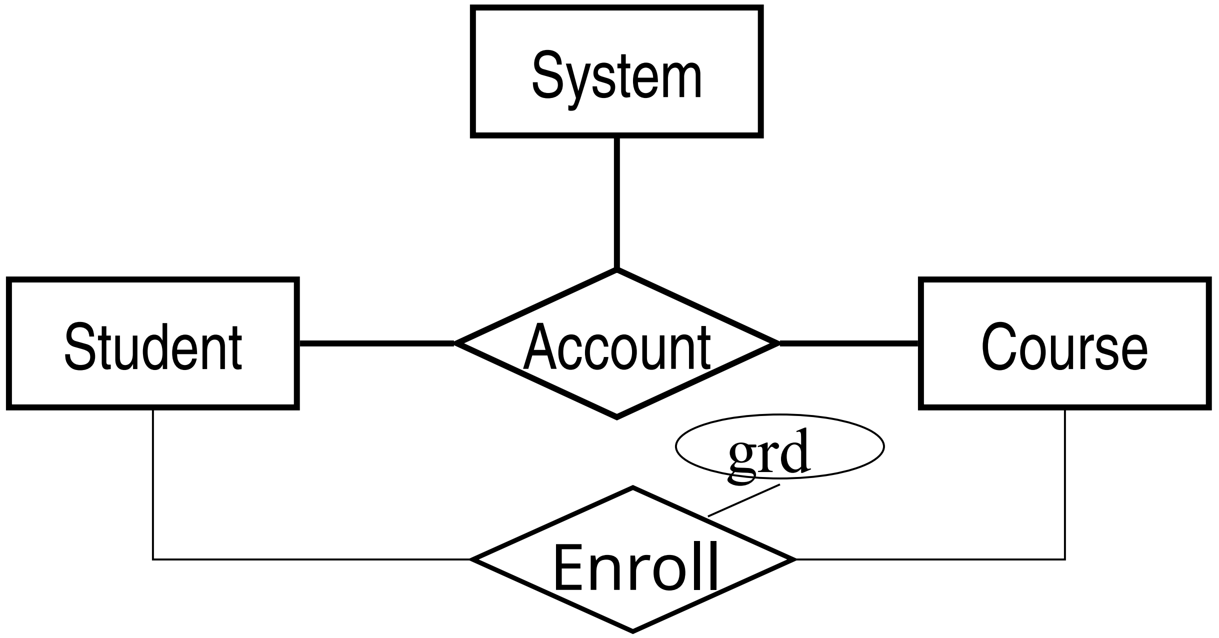


Multiplicity of this ternary relationship: n to n to n

Enroll(sid(fk →Student), cno(fk → Course),, sys(fk → System), grd)

What is the problem here?
What is the schema for Enroll?
How is the n:n:n relationship mapped to a relation(table)?

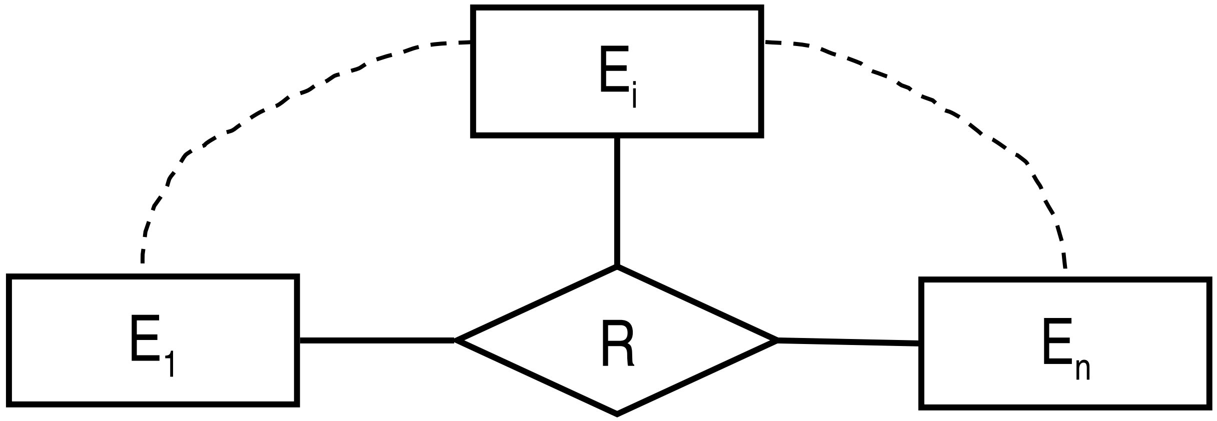
N-ary Relationships



Multiplicity of this ternary relationship: n to n to n

Account(**<u>sid</u>**(fk →Student), **<u>cno</u>**(fk → Course), **<u>sys</u>**(fk → System), **<u>uid</u>**)
Enroll(<u>sid</u>(fk →Student), <u>cno</u>(fk → Course), grd)

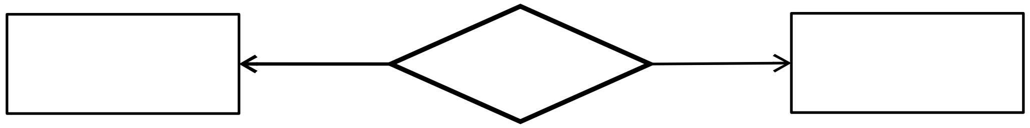
N-ary Relationships



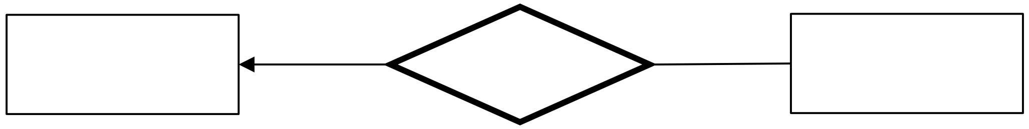
Multiplicity of this N-ary relationship: n_1---- n_i ----- n_n

Any of this n_i could be 1

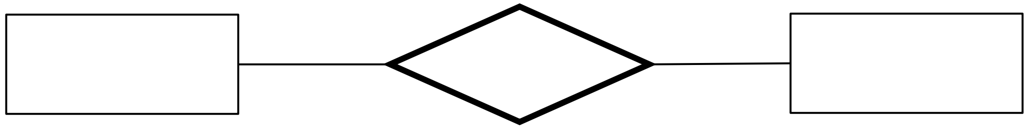
Any of this could be multiple



If a binary relationship between two entity sets is 1-to-1,
- the primary key of the relationship is the key of the entity from either of the '1' side, the other side is a foreign key*(would be unique)*

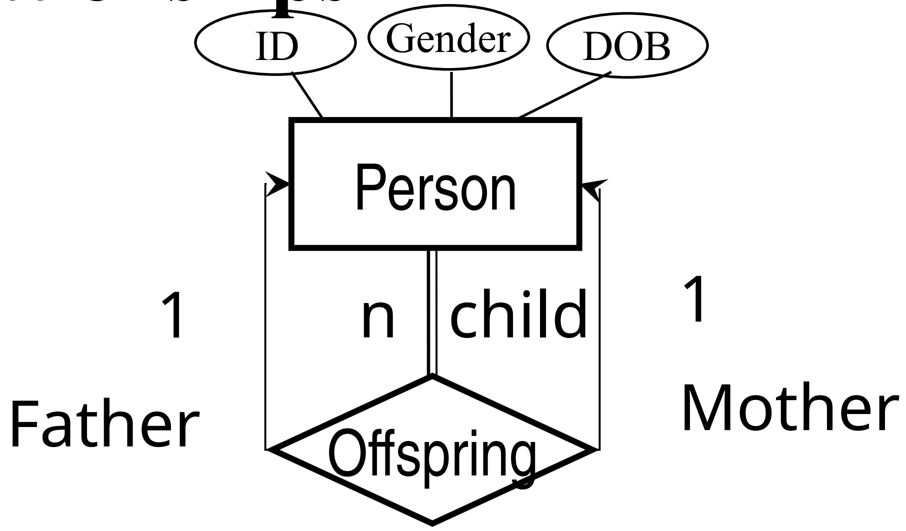


If a binary relationship between two entity sets is 1-to-many,
- the primary key of the relationship is from the 'm' side, the '1' side is the foreign key *(would be unique)*



If a binary relationship between two entity sets is m-to-n,
- the primary key is composite, consisting of the primary key of the entities from each side of the relationship

N-ary Relationships



Multiplicity of this ternary relationship: 1 to 1 to n

Offspring(fid (fatherID → Person(ID)), motherID (fk → Person(ID)), <u>ChildID</u> (fk → Person(ID)))

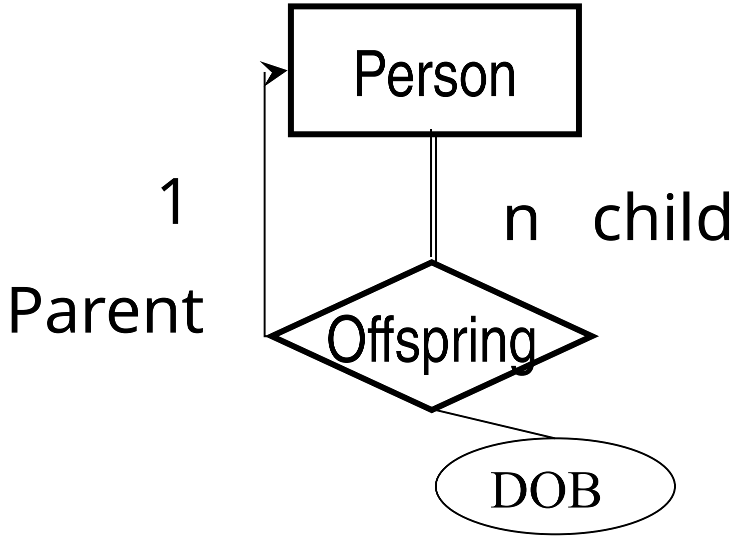
Who here is the father and mother?
What is the key?

```
create table person(
ID number primary key,
gender char(1),
DOB date);
insert into person values(1, 'M', '11-Jan-1900');
insert into person values(2,'F', '11-Jan-1902');
insert into person values(121,'M', '11-Jan-1925');
insert into person values(122,'F', '11-Jan-1927');
insert into person values(3,'M', '11-Jan-1901');
insert into person values(4,'F', '11-Jan-1903');
insert into person values(341,'M', '11-Jan-1926');
insert into person values(342,'F', '11-Jan-1928');
insert into person
          values(1213421,'M', '11-Jan-1948');
insert into person
          values(1213422,'F', '11-Jan-1950');
```

Could two tuples exist in offspring with the same cid???

```
create table offspring(
fid number, mid number,
cid number primary key,
foreign key (fid)
          references person(id),
foreign key (mid)
          references person(id),
foreign key (cid)
          references person(id));
insert into offspring values(1,2,121);
insert into offspring
          values(1,2,122);
insert into offspring
values(3,4,341);
insert into offspring
          values(3,4,342);
insert into offspring
values(121,342, 1213421);
insert into offspring
values(121, 342, 1213422);
```

Replacing a ternary relation by a binary relation



What is the schema for Offspring here?

What is a possible inconsistency problem?

Who is the father, mother??

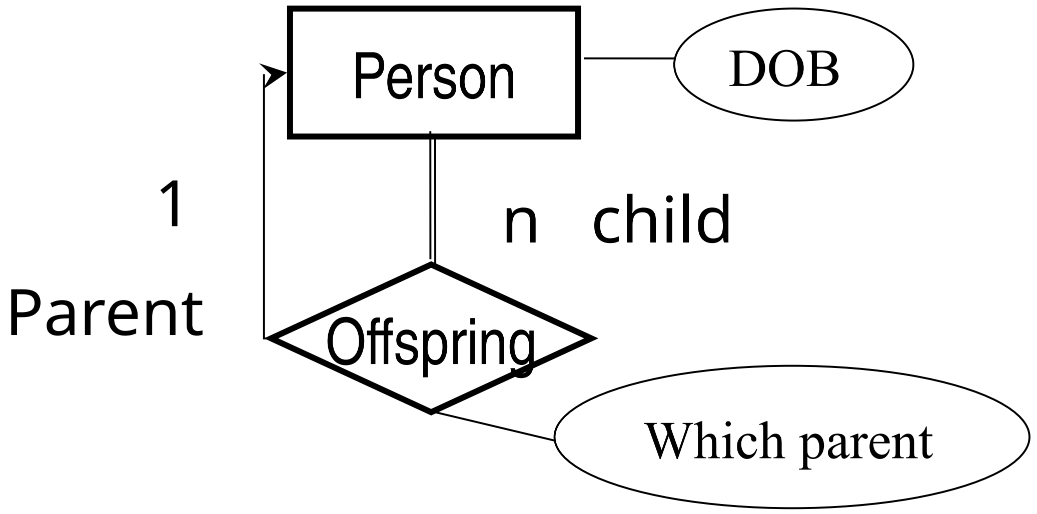
Offspring (<u>IDC</u>, IDF, IDM, DOB)

Father(<u>IDC, IDF</u>, DOB)

Mother(<u>ID1, IDM</u>, DOB)

- ***not the same ER***
- ***duplication of DOB***
- ***Composite key***

Replacing a ternary relation by a binary relation --- an alternate *non-normal form*



What is the schema for Offspring?
Is there a duplication problem?
What is the primary key?

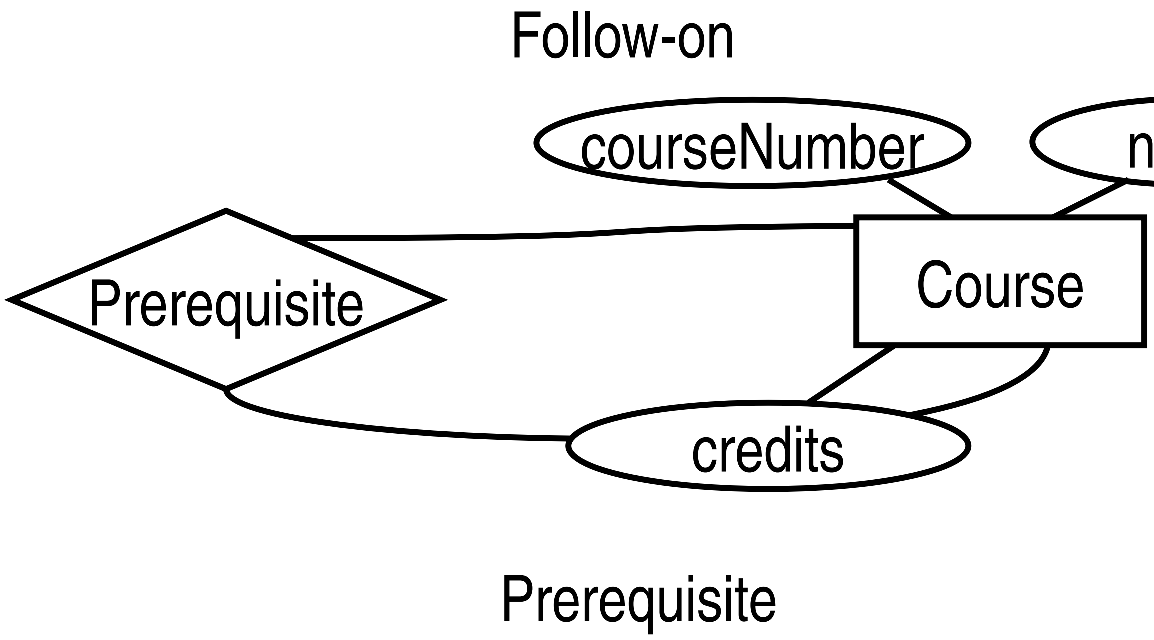
Multivalued attribute

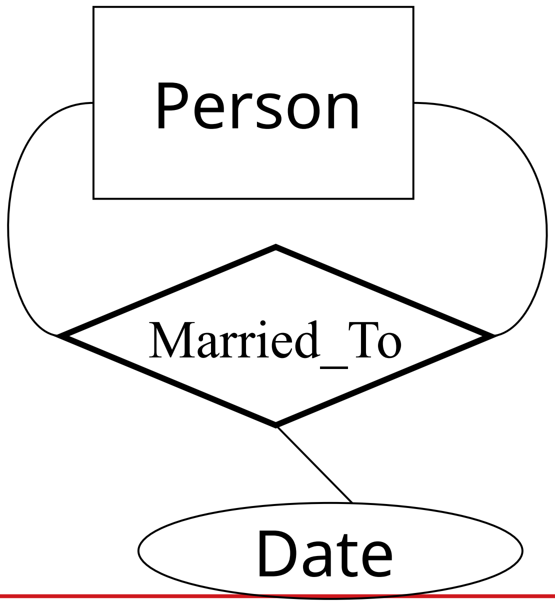
121	1	Father
	2	Mother
122	1	Father
	2	Mother

Roles in Relationships

- It is possible that one entity set appears **two** or **more** times in a relationship
- Suppose, we want to capture relationship between two **courses**, one of which is the **pre-requisite/follow-on** of the other

Each line to the entity set represents a different **role** that the entity set plays in the relationship



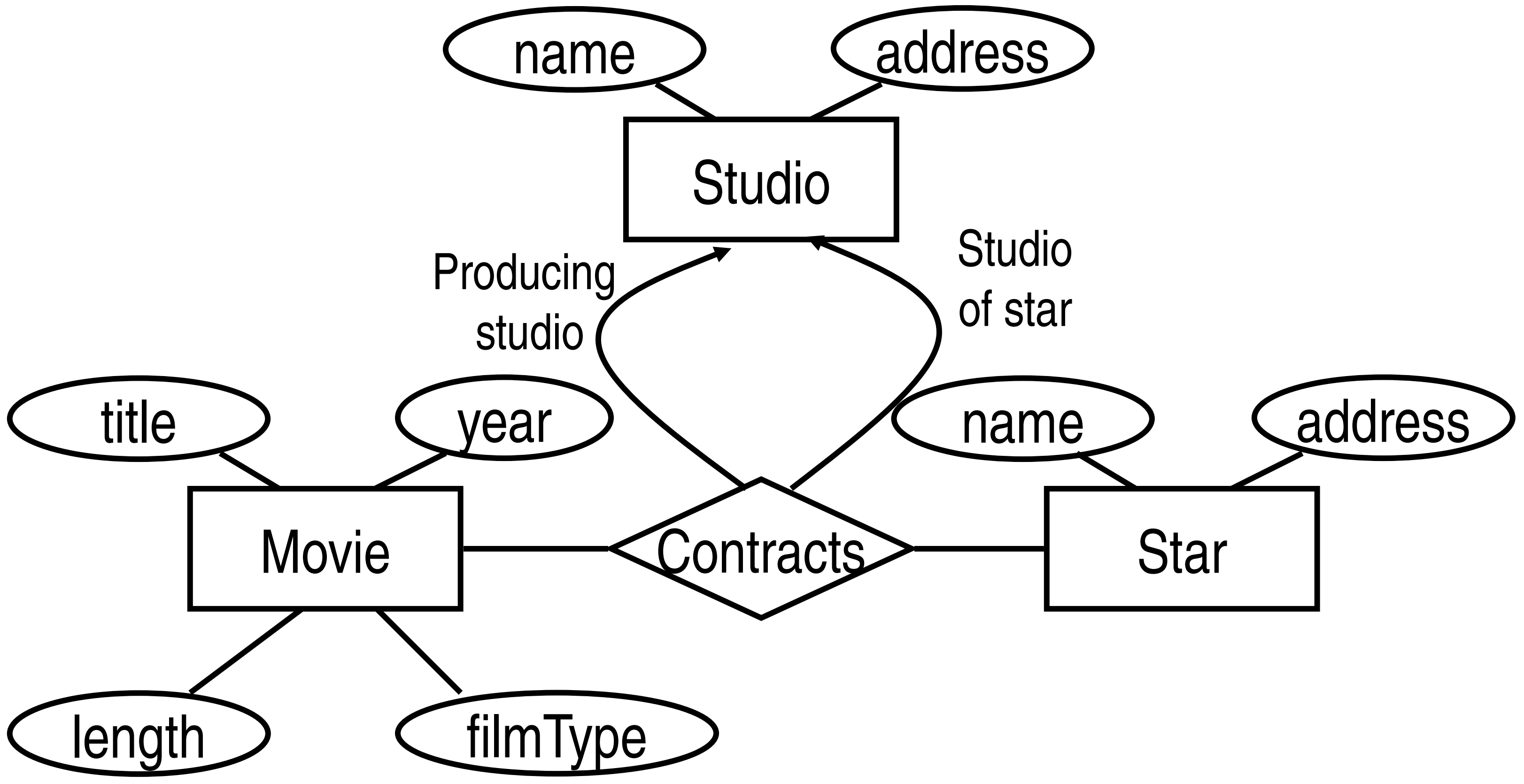


```
create table prereq
(COURSE_Number CHAR(8),
 PREREQ CHAR(8),
 primary key (course_number, prereq))
tablespace TUTOR pctfree 2
/
```

```
insert into prereq values('COMP353','COMP352');
insert into prereq values('COMP353','COMP346');
insert into prereq values('COMP352','COMP248');
```

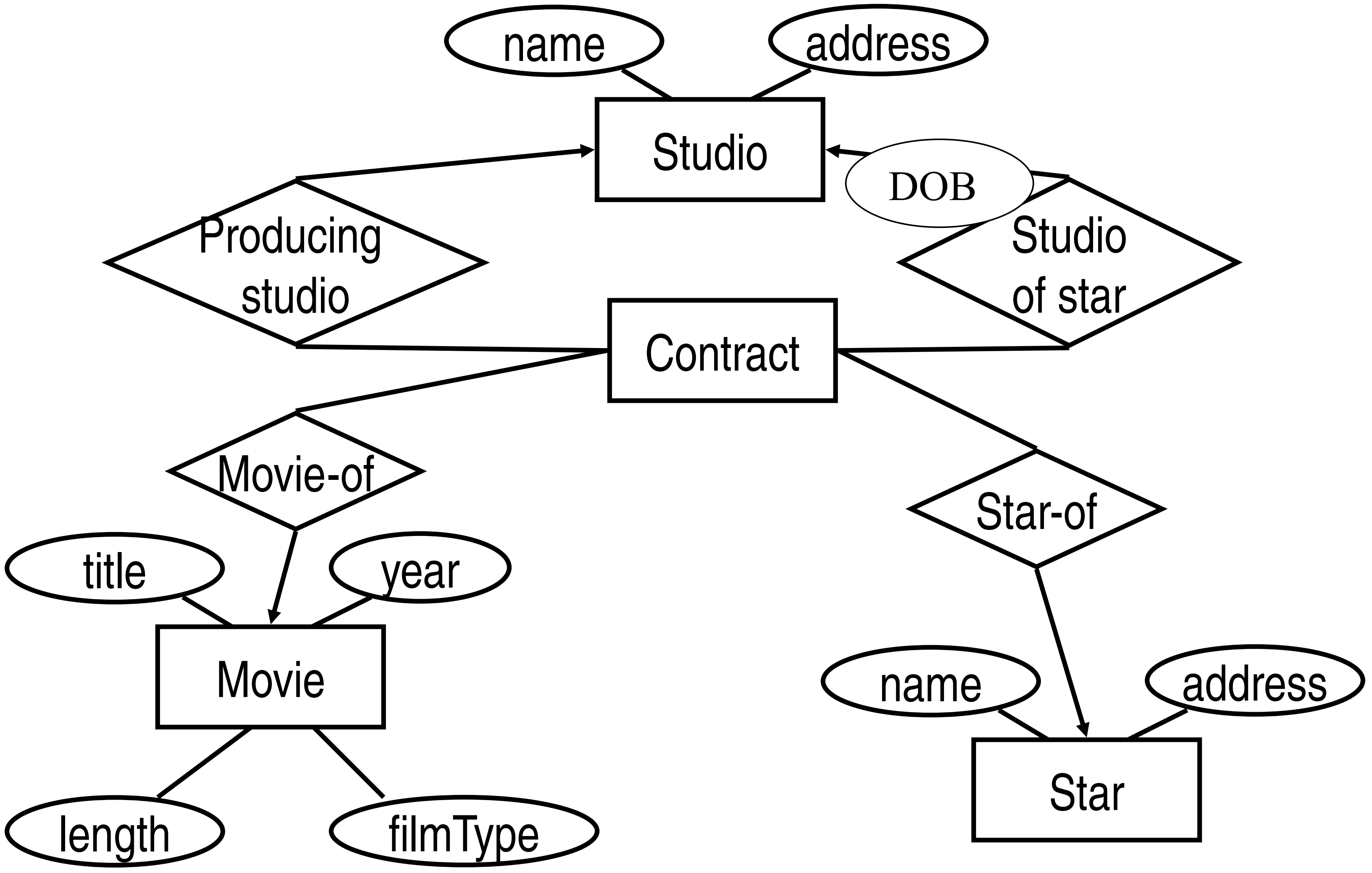
Suppose, each star is under contract with a single studio

The studio of the star may enter into a contract with another studio to allow that star to act in a particular movie

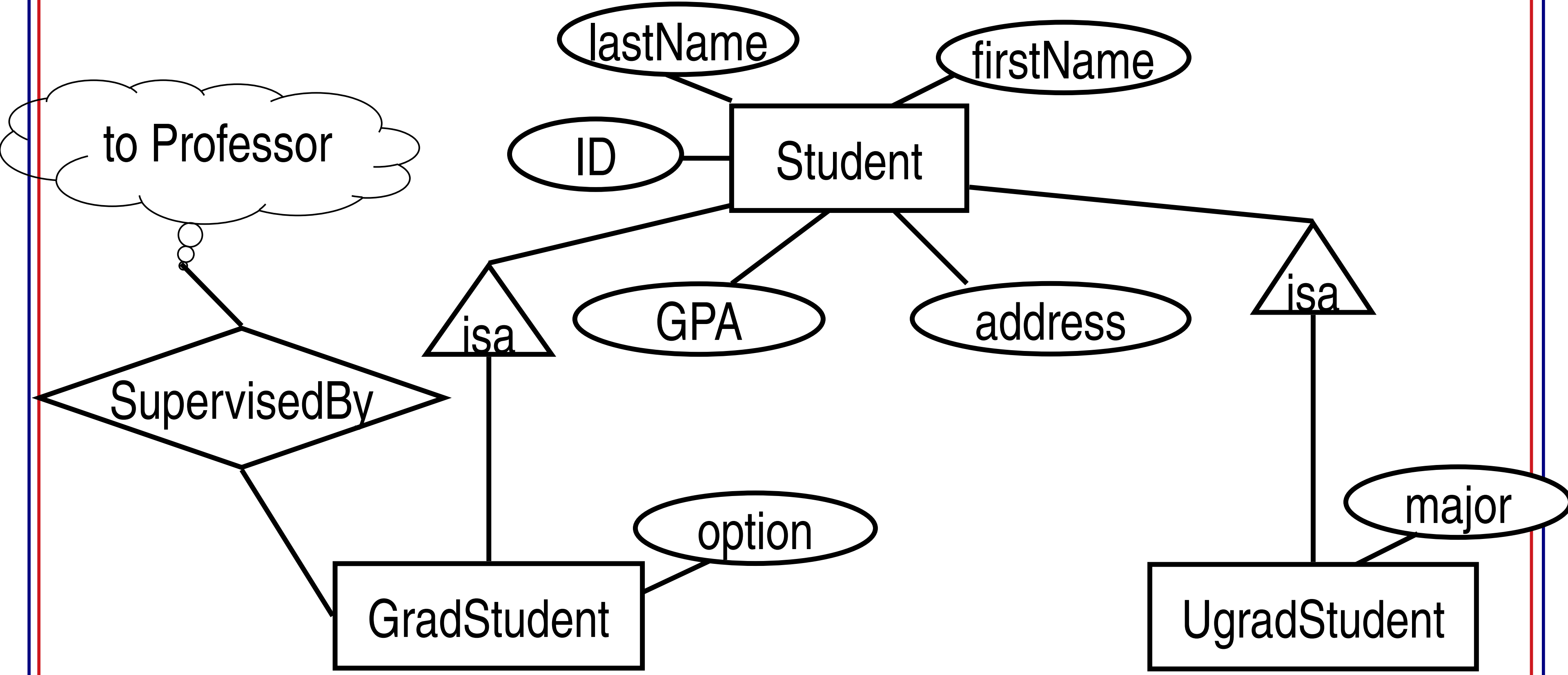


Converting n-ary relationship

- Any n-ary relationship may be converted into a **collection** of **binary** relationships **without loosing** any information???
  - Introduce a **new** entity set – **connecting** *existing* entity set – whose entities might be thought of as tuples of the relationship for the n-ary relationship
  - Introduce **many-to-one** relationships from the **connecting** entity set to each of the entity sets participating in the original n-ary relationship
  - If an entity set plays **more than one** role, then it is the target of one relationship for **each role**
- Usually doesn't convey the same semantics – limitation of modeling


name
address
Studio
DOB
Producing studio
Studio of star
Contract
Movie-of
Star-of
title
year
Movie
name
address
Star
length
filmType

Inheritance in E/R is expressed by ***isa*** relationship



- There is a subtle difference between the concept of inheritance in ODL and in the E/R model
- In **ODL**, an object must be a member of exactly one class
- In the **E/R** model
  - We shall view an entity as having "components" belonging to several entity sets that are "part of" a single **isa**-hierarchy
  - The "components" are connected into a single entity by the **isa** relationships
  - The entity has whatever attributes any of its components has, and it participates in whatever relationships its components participate in

    - We need to represent an entity (e.g., CartoonMurderMystery) in the diagram only if it has attributes and/or relationships of its own

Constraints

- There are some important **aspects** of the **real world** that **cannot** be represented using the **ODL** or **E/R** model introduced so far
- The additional information about these aspects often takes the form of **constraints** on the data
- Sometimes modeling this additional information goes beyond the **structural** and **type** constraints imposed by **classes**, **entity sets**, **attributes,** and **relationships**

Keys are (sets of) attributes that **uniquely** identify an object within its class or an entity within its entity set;

$K \subseteq R$. ***no two entities may agree in all their key values***

Single-value constraints are requirements that the value of an attribute be unique. In addition to key constraints, other attributes must a have a single-value constraints.

Also in an "one" relationship

Referential integrity constraints are requirements that a value referred to by some object must actually exist in the database;

This means, no dangling pointers.

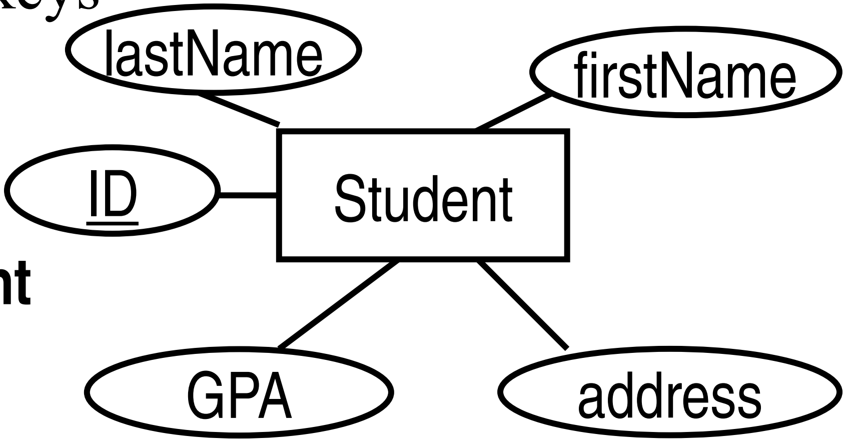
Domain constrains require that the value of an attribute must be drawn from a specific set of values (called attribute domain), or lies within a specific range

General constraints – arbitrary assertions that must hold on the DB

Keys

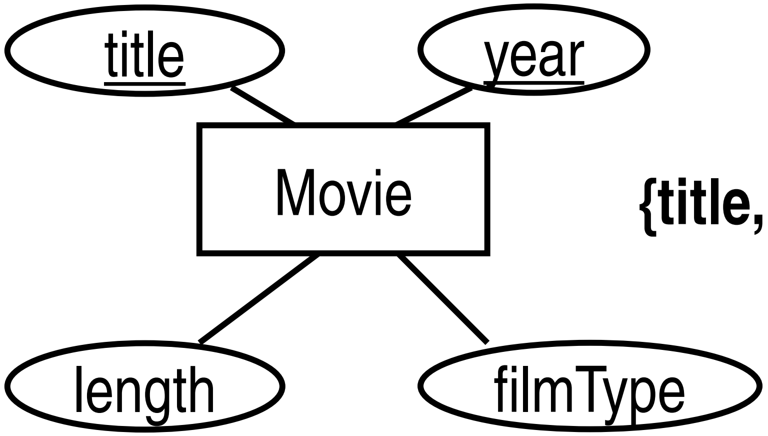
- A *super key* is a set of attributes whose values uniquely identify an entity in the entity set; this set may not be minimal.
- A **minimal super key** is called a **(***candidate***) key.**
- An entity may have more than one **key**. One of them is picked as the primary key**;** others may be called *alternate keys*
- In **E/R,** we <u>underline</u> the key attribute(s) of an entity set (i.e., those attributes that form the primary key of the set)
- No notation in E/R for alternate keys

<u>**ID**</u> is the key for the entity set **Student**



Example

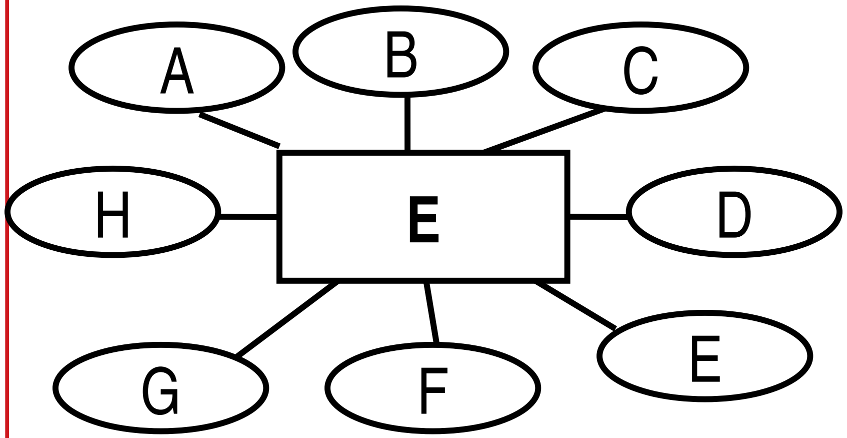
- What should we select as a key for **Movie** ?
- **title**?
  - there could be different movies with the same name
- **{title, year}**?
  - there still could be two movies made in the same year, with the same title, but that's very unlikely



{title, year} is a key for **Movie**

https://en.wikipedia.org/wiki/Lists_of_film_remakes

Selecting A Primary Key



Suppose candidate keys for **E** are :

1. **{A, B}**
2. **{D, E}**
3. **{F, G, H}**

❖ Which of the three should "we" pick as the **primary key**?

Criteria to choose a **primary key** when there are more than one candidate:

Total size

Number of attributes

Convenience

A combination of the above

Single Value Constraint

- In **E/R:**
  - attributes are **atomic**
  - an arrow (→) can be used to express the multiplicity
  - What about multi-valued attributes or relationships?

With the E/R model introduced so far, we cannot express the following options regarding the value of a single-valued **attribute**:

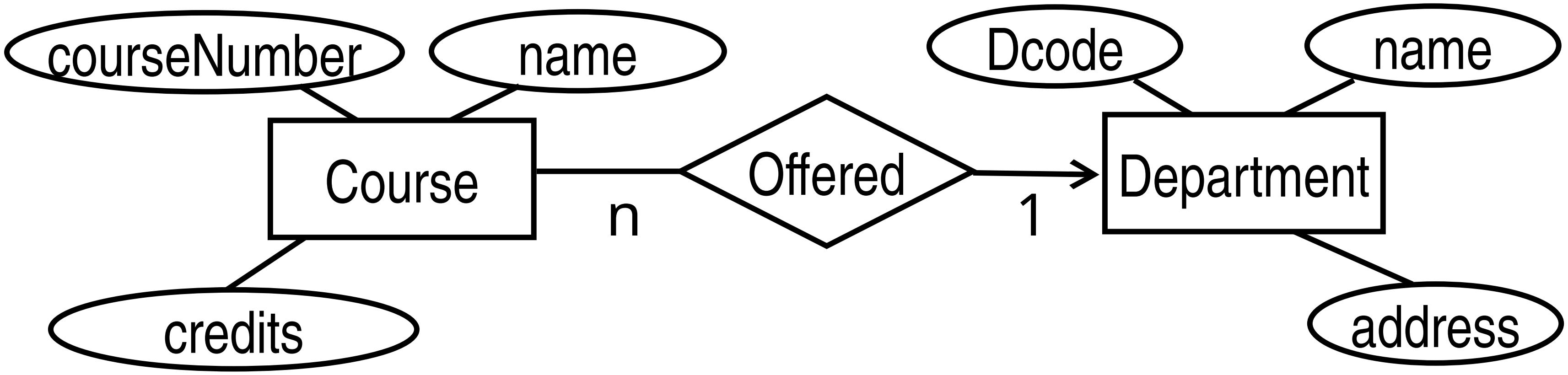
- Require that the value of that attribute to be present(not null)
- Or the presence of the value be optional (null allowed)

If the choice is not explicit, then we may conclude:

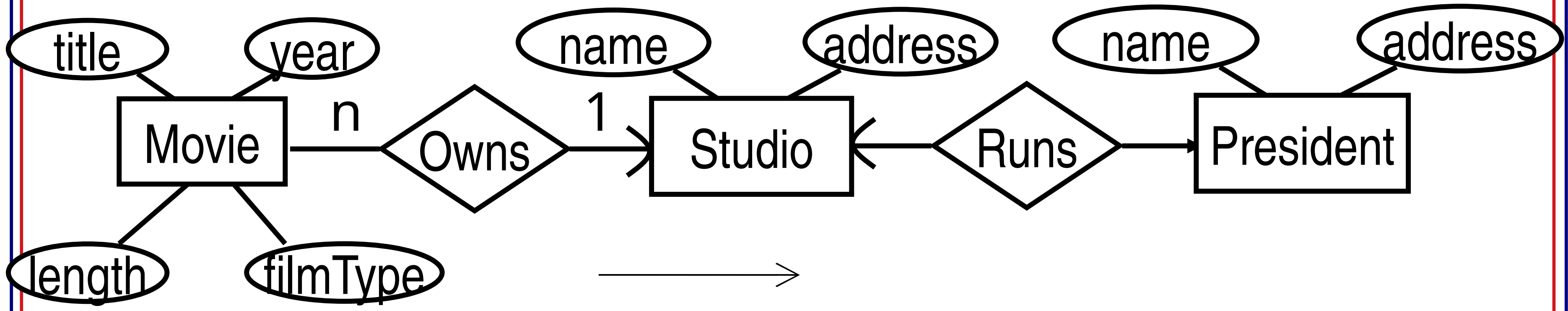
- The value must exist if an attribute is part of the key
- The value is optional, otherwise

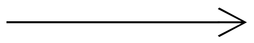
Referential Integrity Constraints

- For relationships:
  - **Single-value + Existence = Referential Integrity Constraint**
- We extend the **arrow** notation to indicate a reference is mandatory (to support referential integrity)
  - The department that gives a course must always exist in the **Department** entity set



Referential Integrity Constraints





open arrow to denote ref. intg.

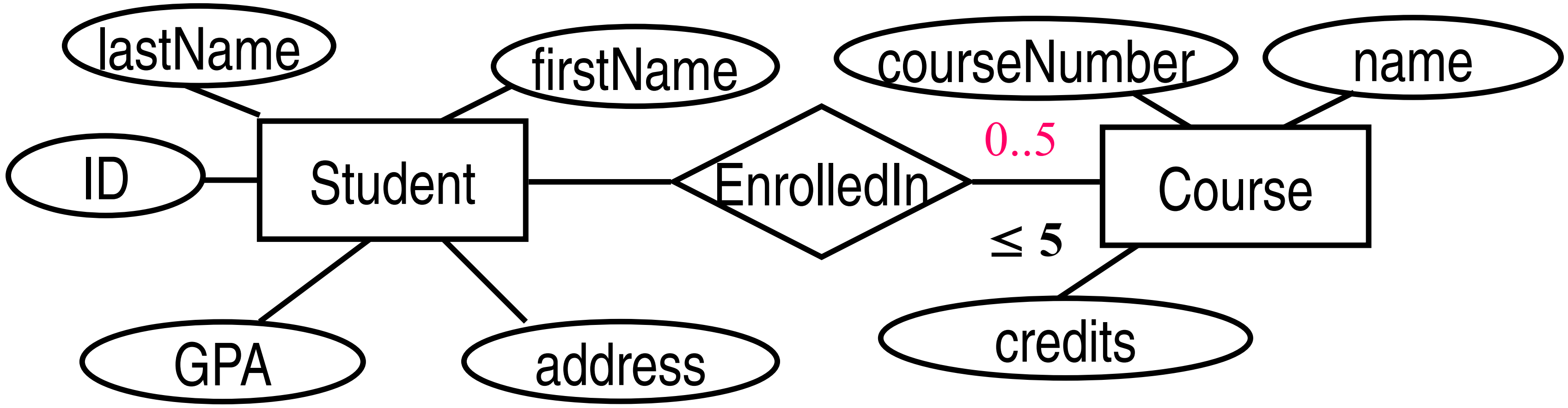
- The studio owning a movie must always be present in the Studios (extent of the Studio entity set)
- If a president runs a studio, then that studio exists in the Studios
- There could be studios without a president (temporarily)

Domain constraints

- **Domain constraints** restrict the values of an attribute to be drawn from a set
  - In ODL, we give a type to the attributes and hence limit their set of values
  - ODL does **not** support other restrictions, such as that the value should be within a certain range
  - E/R, in general, does **not** support imposing domain constraints

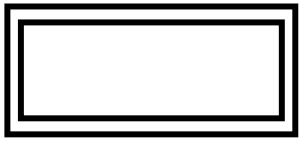
Relationship degree constraints

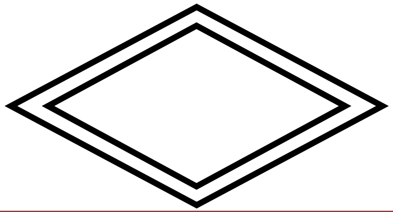
- **Relationship degree constraints** restrict the number that an entity/object can participate in a relationship
  - For example, we can impose a constraint saying that a student cannot be enrolled in more that 5 courses
  - In ODL, we could use, instead of a set of references, an **array** of size 5
  - In E/R, we may attach a bounding number to the corresponding link



Weak Entity / Relationship Sets

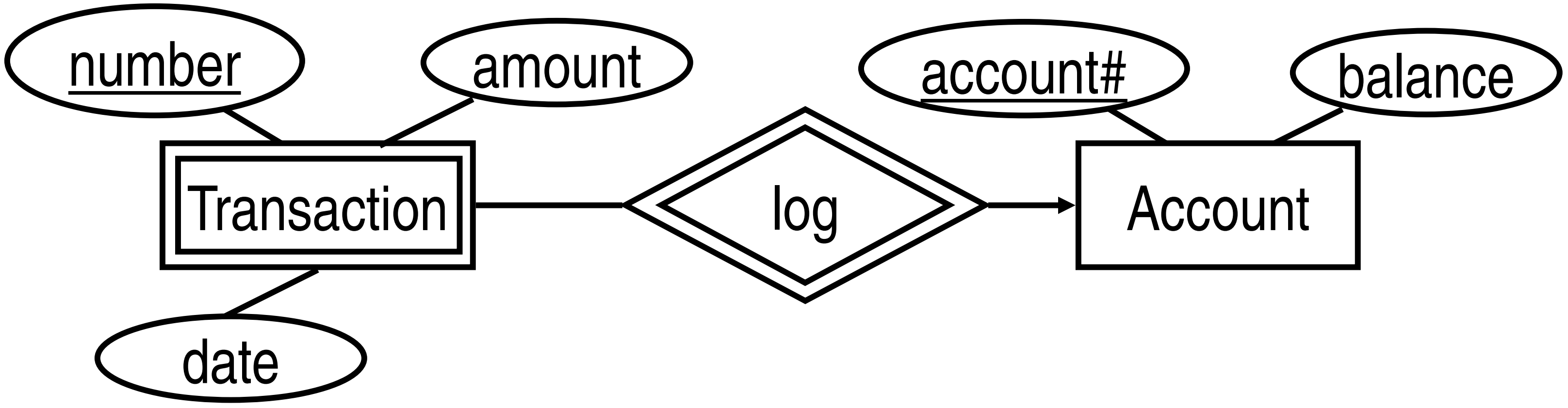
- A **strong** entity set has a primary key
- A **weak** entity set does not have sufficient attributes to form a primary key. It should be part of a one-many relationship (with no descriptive attributes) with a strong entity set
- **Discriminator** of a weak entity set is a set of attributes that distinguishes among the entities corresponding to a strong entity
- **Primary key** of a weak entity set = primary key of the strong entity + discriminator of the weak entity
- Represented in E/R model by

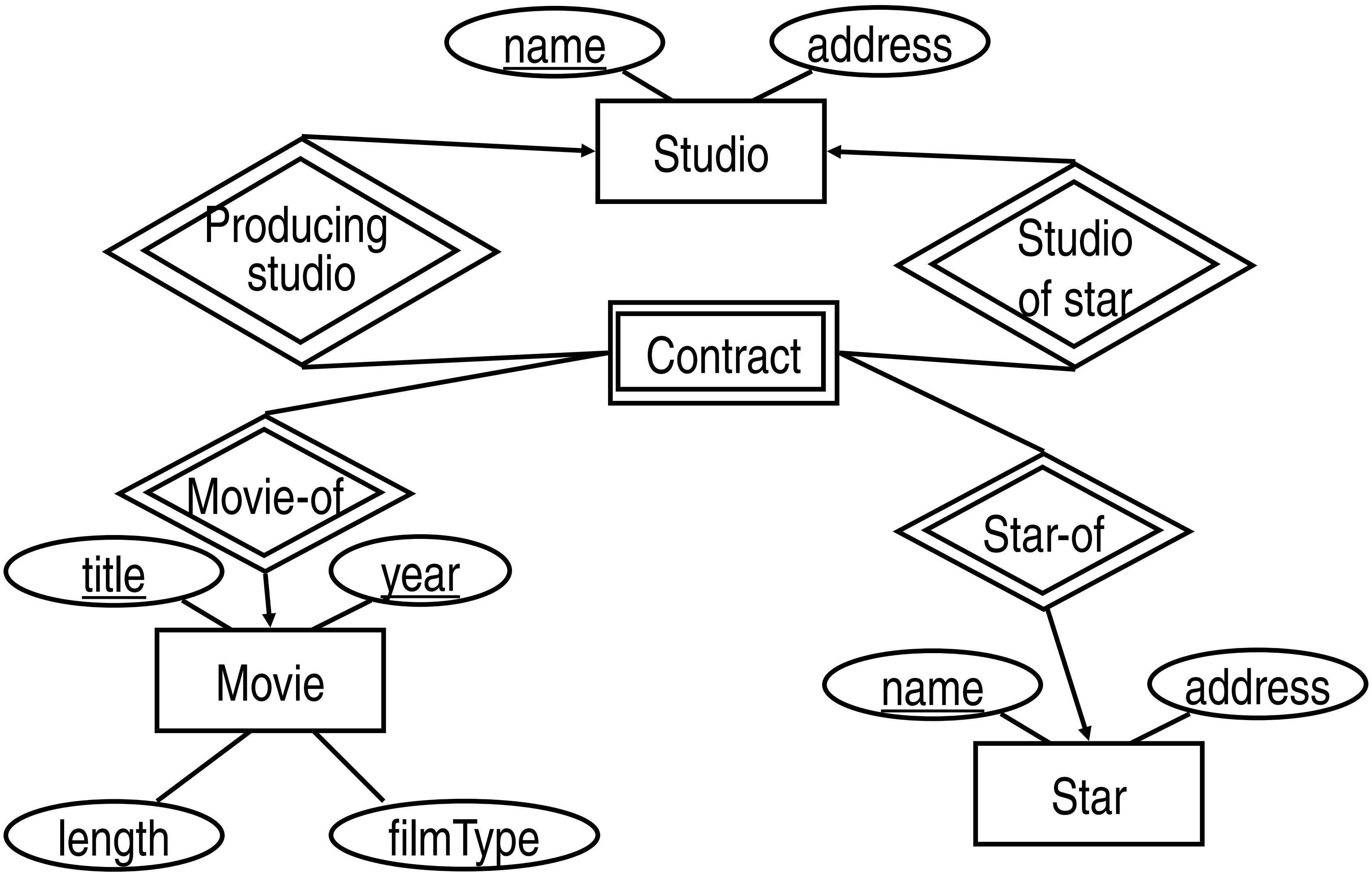




Example

- Log records transactions done by an ATM
- Each transaction has a number, date, and an amount
- Different accounts might have transactions by the same number, on the same date, and for the same amount




name
address
Studio
Producing studio
Studio of star
Contract
Movie-of
Star-of
title
year
Movie
length
filmType
name
address
Star

Design Principles

- Design should
  - Reflect reality
  - Avoid redundancy
    - Redundant information takes space
    - Could cause inconsistency
  - Be as simple as possible
- Be careful when choosing between using attributes and using classes or entity sets. Remember that
  - An attribute is **simpler** to implement than either a class/entity set or a relationship
  - If something has **more information** associated with it **than just its name**, it probably **needs to be an entity set or a class**

Consider the entity set course in a typical university :

It could be involved in many relationships:

one to many relationships with

    the offering department,

    the offerings faculty

    the professor coordinating the course

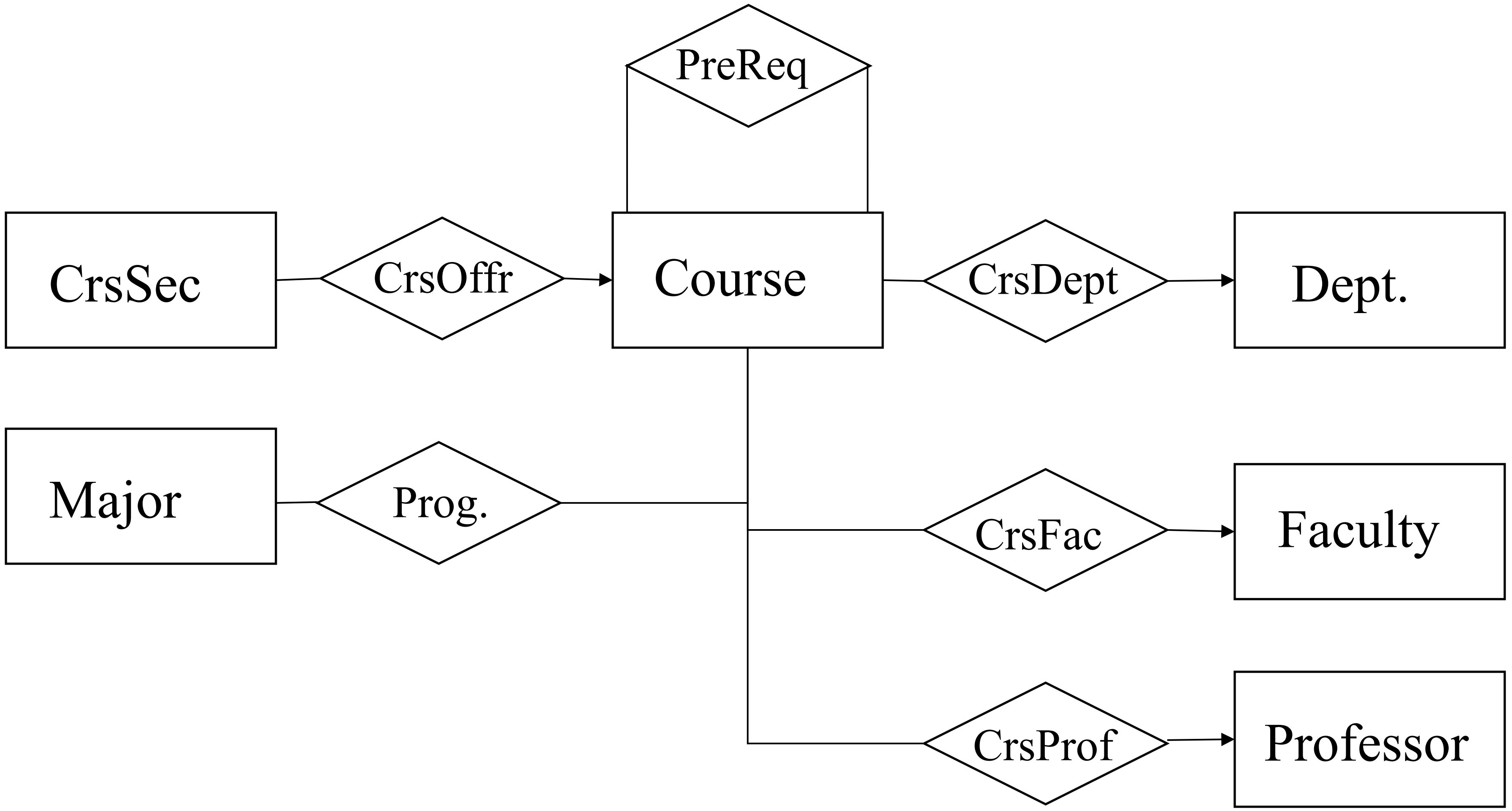
A many to one relationship with

    sections for the course

many to many relationships with

    the major program in which it is required

    the pre-requisites (follow up) courses



How to implement the entity Course and its relationship

Design decision

Merge the one-to-many relationships

CrsDept, CrsFac, CrsProf

in the schema for Course; all attributes of these relationships are also included in the schema for Course

Create a separate relation for each one-to-many relationships

In this case the relation for Course would have a higher arity; but requires one less join to get details for the department, faculty or professor for a given course

Similarly, merge the one-to-many relationship

CrsSec

in the schema for CrsSec

Create a relation for the many to many relationships

Program and PreReq