CS2102 Lecture 4
Entity-Relationship Model

Database Design Process

1. **Requirement Analysis** - find out the data/application/performance requirements of the enterprise
2. **Conceptual Database Design** - capture data requirements using a conceptual schema
3. **Logical Database Design** - map conceptual schema to logical schema supported by DBMS
4. **Schema Refinement** - improve logical schema design using data constraints
5. **Physical Database Design** - use performance requirements to design physical schema
6. **Application & Security Design** - specify access control policies

Requirement Analysis: Example

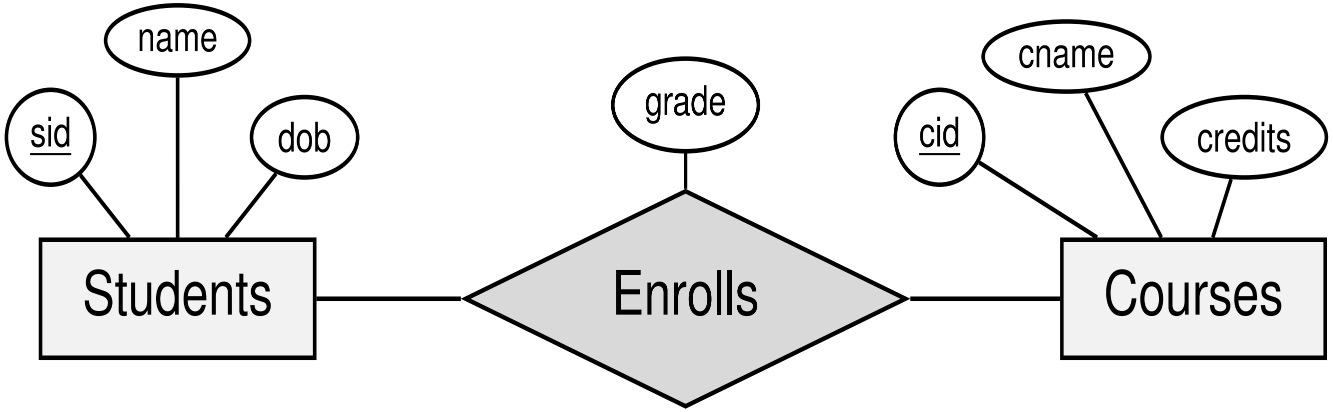
I would like my customers to be able to browse my catalog of books and place orders over the Internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the ISBN number of a book and a quantity; they often pay by credit card. I then prepare a shipment that contains the books they ordered. If I don't have enough copies in stock, I order additional copies and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its ISBN number, title, author, purchase price, sales price, and the year the book was published. Most of my customers are regulars, and I have records with their names and addresses. New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique customer identification number. Then they should be able to browse my catalog and to place orders online.

Conceptual Data Models

- Entity-Relationship (ER) Model
  - Developed by Peter Chen in 1976
  - Designed for conceptual data model specifications
- Unified Modelling Language (UML)
  - Developed by Grady Booch & James Rumbaugh in 1997
  - Goes beyond conceptual data modelling - software design specifications
  - Standardized by Object Management Group (OMG)

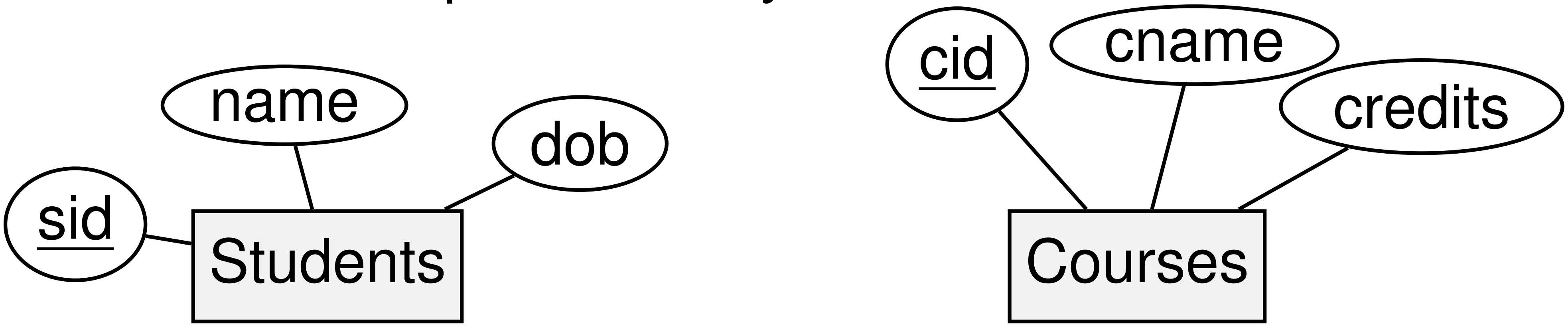
Entity-Relationship (ER) Model

- The **entity-relationship (ER) model** is the most common data model used for conceptual database design
- Data is described in terms of **entities** and their **relationships**
- Information about entities & relationships are described using **attributes**
- Certain data constraints are represented using additional annotations
- ER schemas are presented as **ER diagrams**



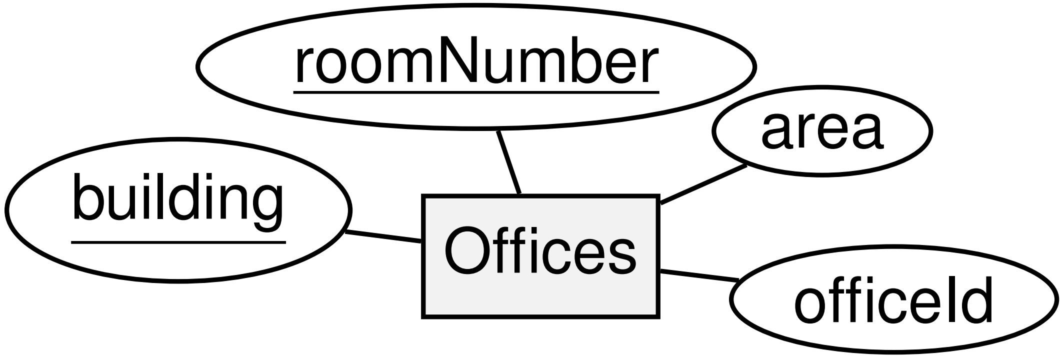
Entities & Attributes

- **Entity** = real-world object distinguishable from other objects
- **Attribute** = specific information describing an entity
  - Each attribute has an *atomic domain* (e.g., integer, string)
- **Entity Set** = a collection of similar entities
- Entity sets are represented by rectangles
- Attributes are represented by ovals



Entity Keys

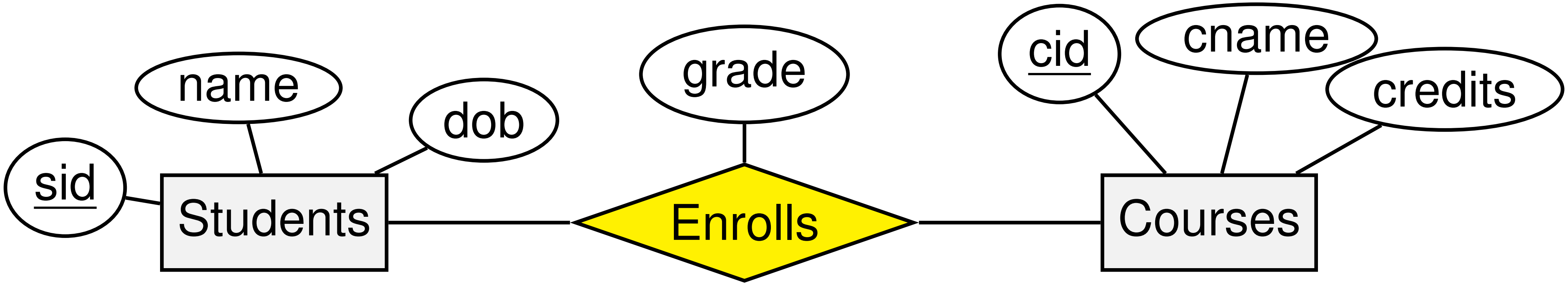
- Each entity set has a **key** = minimal set of attributes whose values uniquely identify an entity
- An entity set could have multiple keys called **candidate keys**
- **Example:**



- One of the candidate keys is chosen as the **primary key**
- The attributes that formed a primary key are underlined

ER Model: Relationships

- **Relationship** is an association among two or more entities
- **Relationship set** is a collection of similar relationships
- Attributes are used to describe information about relationships
- Relationship sets are represented by diamonds

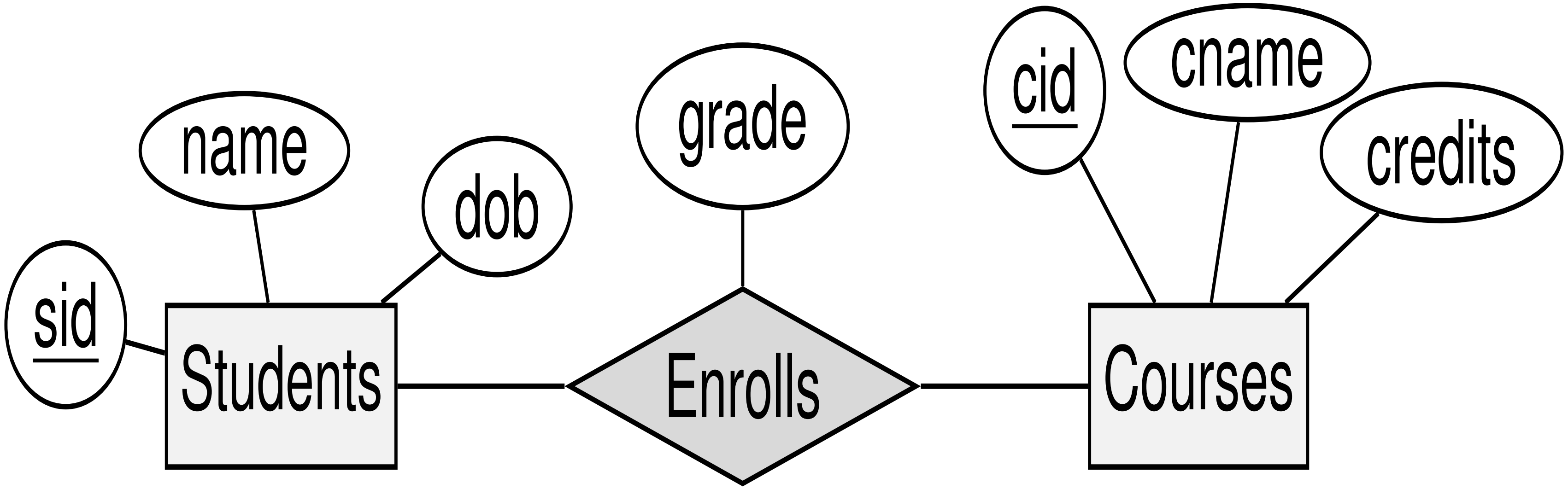


Entities, Relationships & Attributes

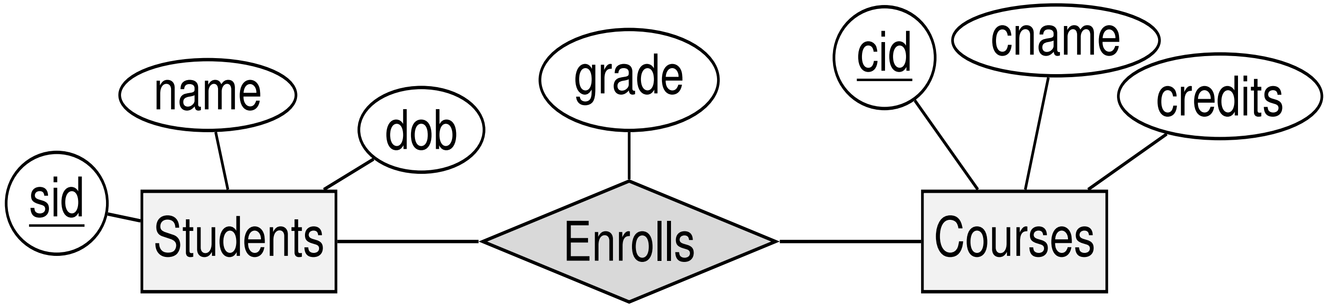
I would like my **customers** to be able to browse my catalog of **books** and **place orders** over the Internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the **ISBN number** of a book and a **quantity**; they often pay by **credit card**. I then prepare a shipment that contains the books they ordered. If I don't have enough **copies in stock**, I order additional copies and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its ISBN number, **title**, **author**, **purchase price**, **sales price**, and the **year** the book was published. Most of my customers are regulars, and I have records with their **names** and **addresses**. New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique **customer identification number**. Then they should be able to browse my catalog and to place orders online.

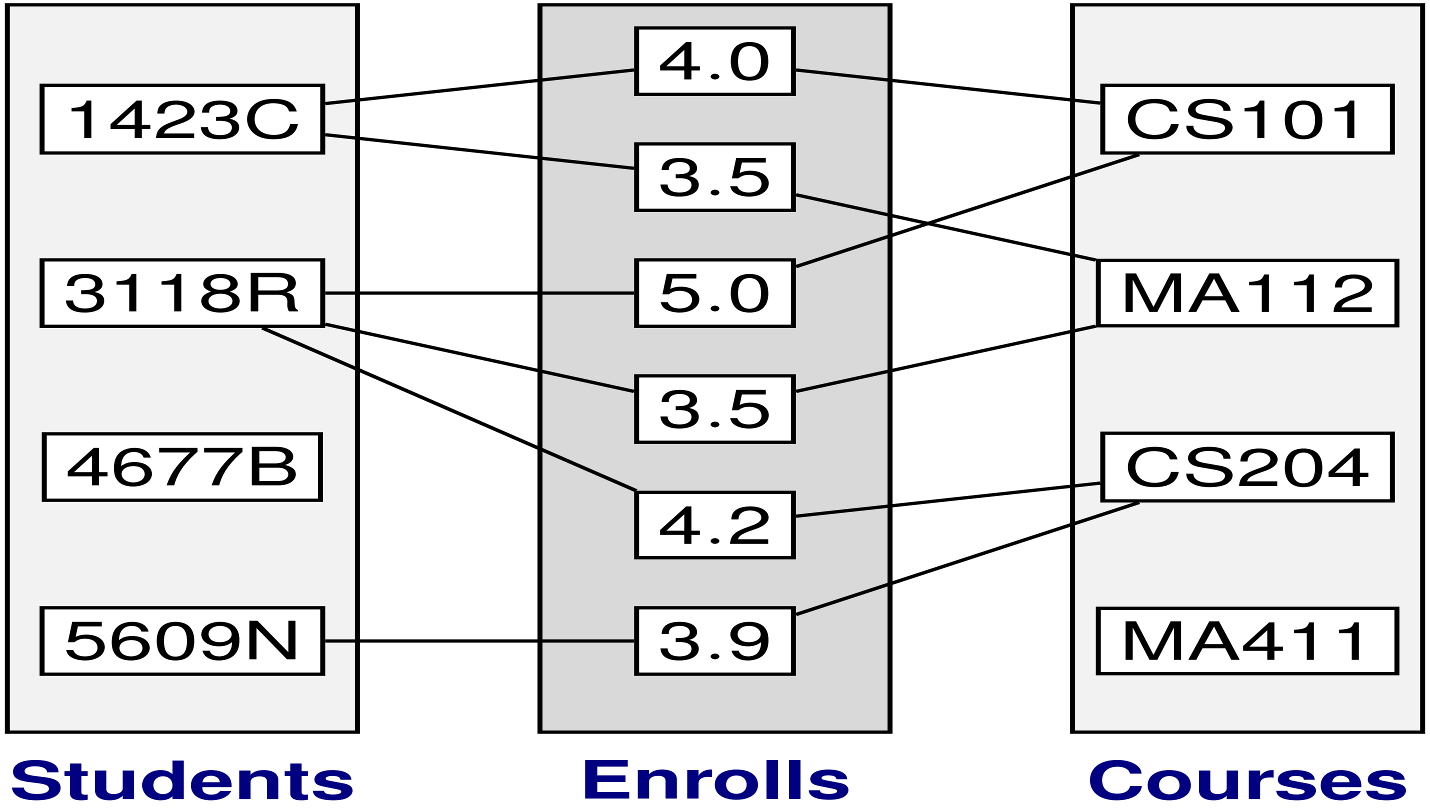
Many-to-Many Relationship Sets

- **Many-to-many** relationship between `Students` and `Courses`
- Each student can enroll in 0 or more courses
- Each course can by enrolled by 0 or more students



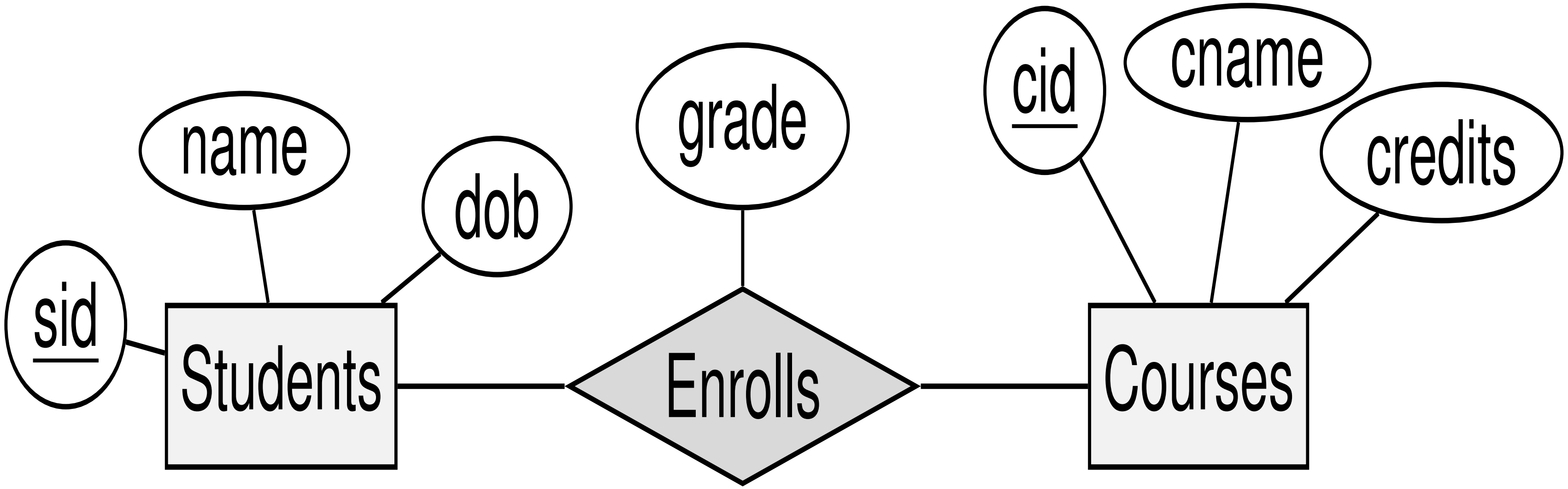
Many-to-Many Relationship Sets (cont.)





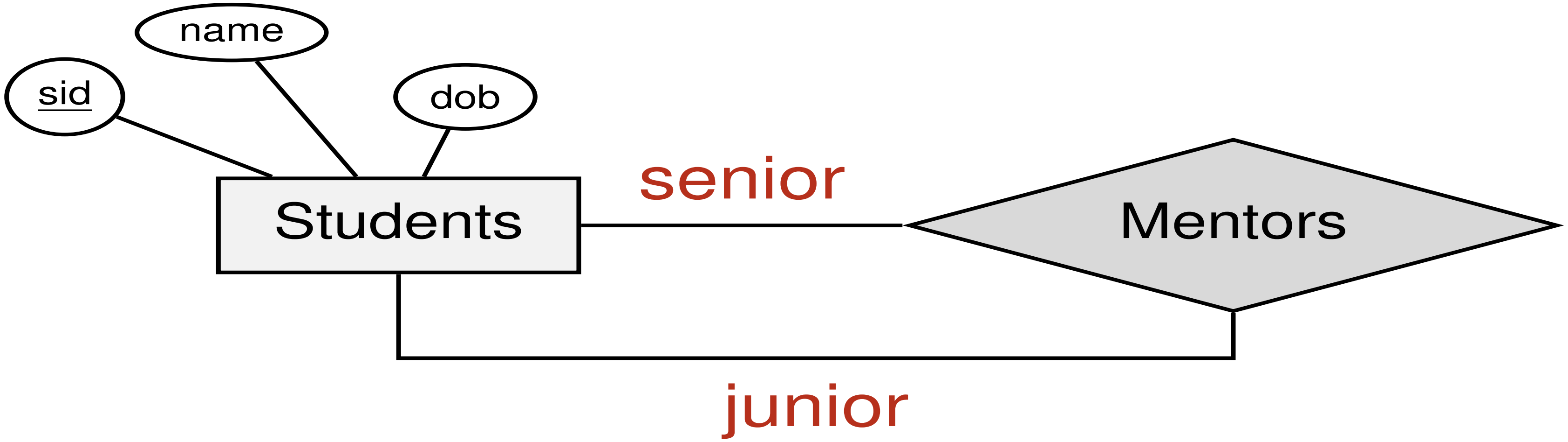
Relationship Roles

- Each entity set participating in a relationship set plays a certain **role**
- The role is typically named the same as the entity set name & is not shown explicitly



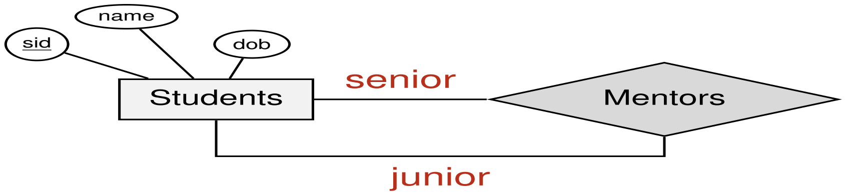
Relationship Roles (cont.)

- **Roles** are shown explicitly when one entity set appears two or more times in a relationship set

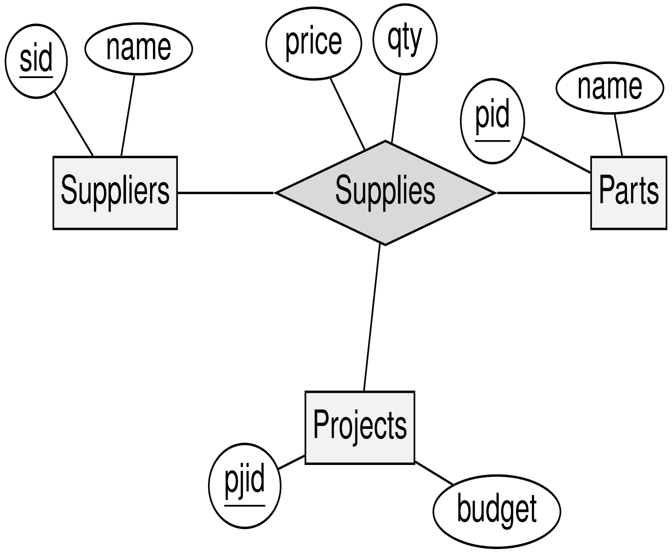


Degree of Relationship Sets

- An n-ary relationship set involves *n* entity roles
- n = **degree of relationship set**
- When n=2, we have a *binary relationship set*

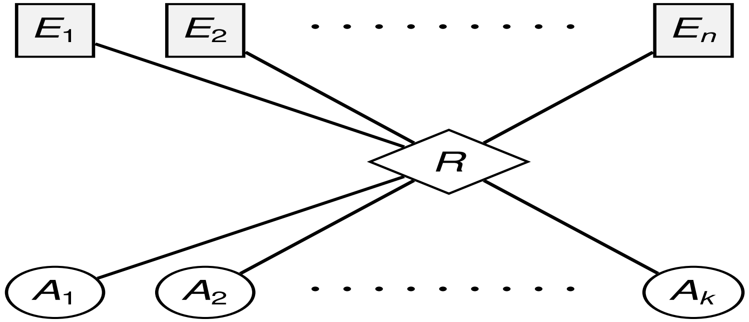


- When n=3, we have a *ternary relationship set*



Relationship Keys

- Consider a n-ary relationship set R involving entity sets $E_1, \cdots, E_n$ with relationship attributes $\{A_1, \cdots, A_k\}$



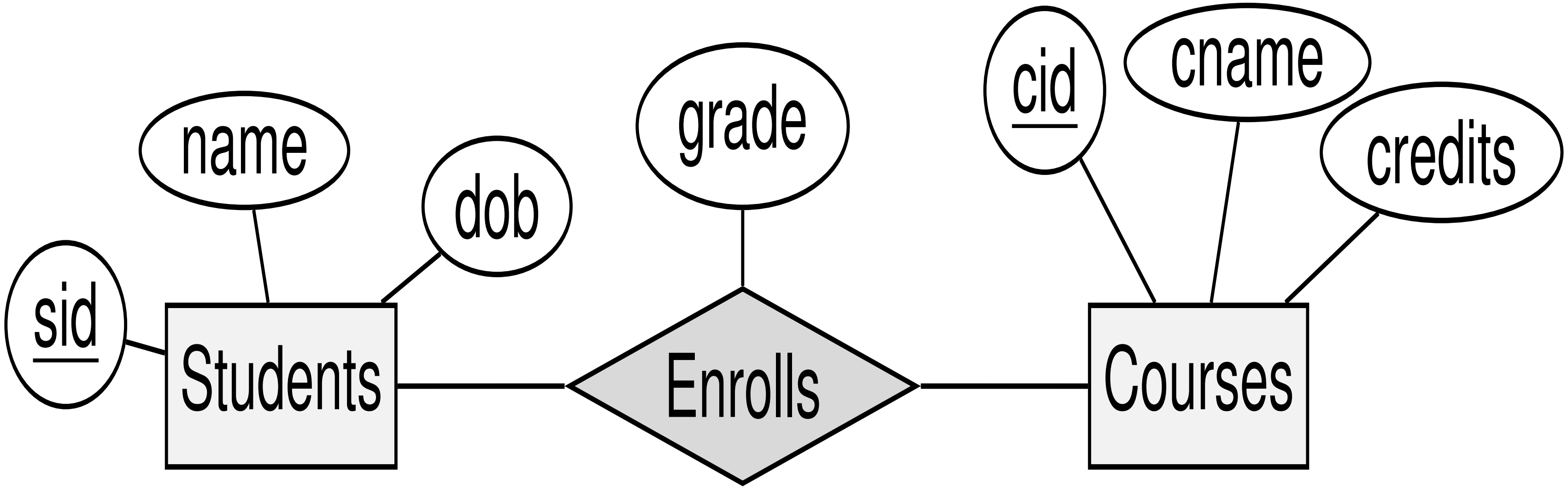
- Let $Key(E_i)$ denote the set of attributes that define the primary key of entity set E_i

- Each instance of R involves one instance of each E_i & have the following attributes:
    - Key(E_1), $\cdots$, Key(E_n)
    - $A_1, \cdots, A_k$

Relationship Keys (cont.)

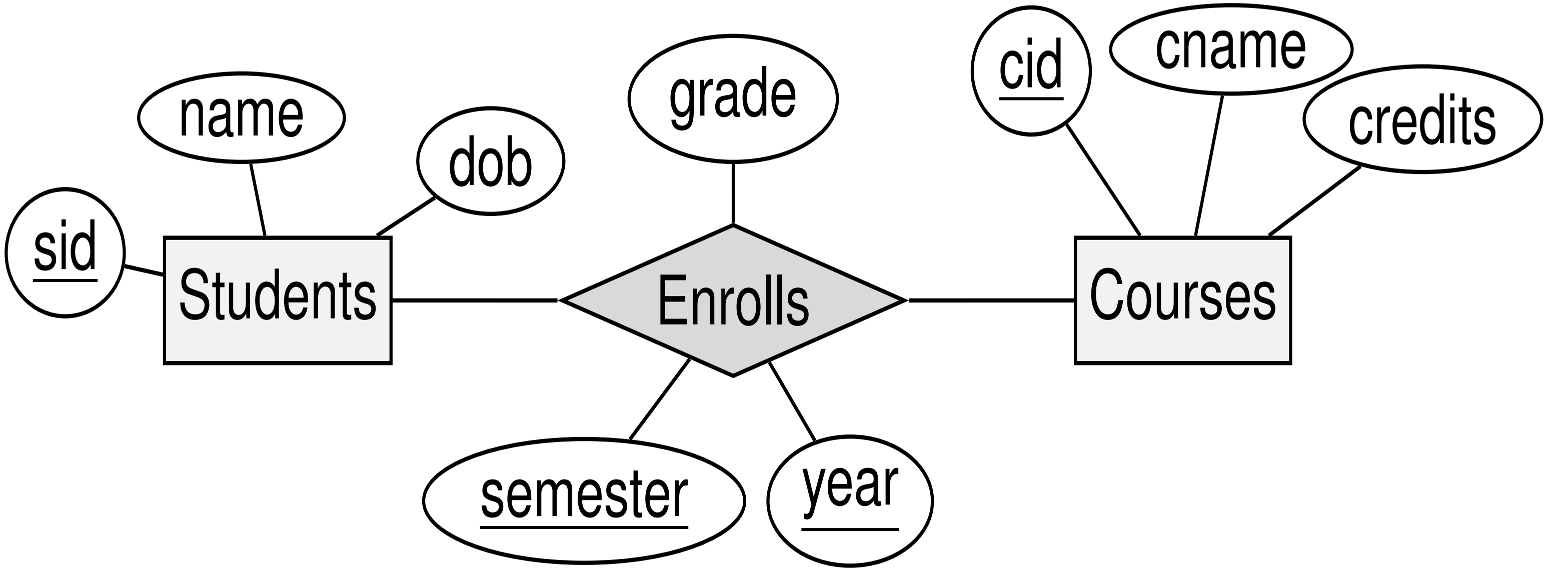
- The key for R (denoted by Key(R)) is specified by some subset $A' \subseteq \{A_1, \cdots, A_k\}$ and some subset $E' \subseteq \{E_1, \cdots, E_n\}$ such that
  - $Key(R) = A' \cup \bigcup_{E_i \in E'} Key(E_i)$ is a minimal subset of attributes whose values uniquely identify a relationship instance of R
- Relationship attributes in $\{A_1, \cdots, A_k\}$ that form part of the relationship key are underlined.
  - Each attribute in A' is underlined in ER diagram

Relationship Keys: Example 1



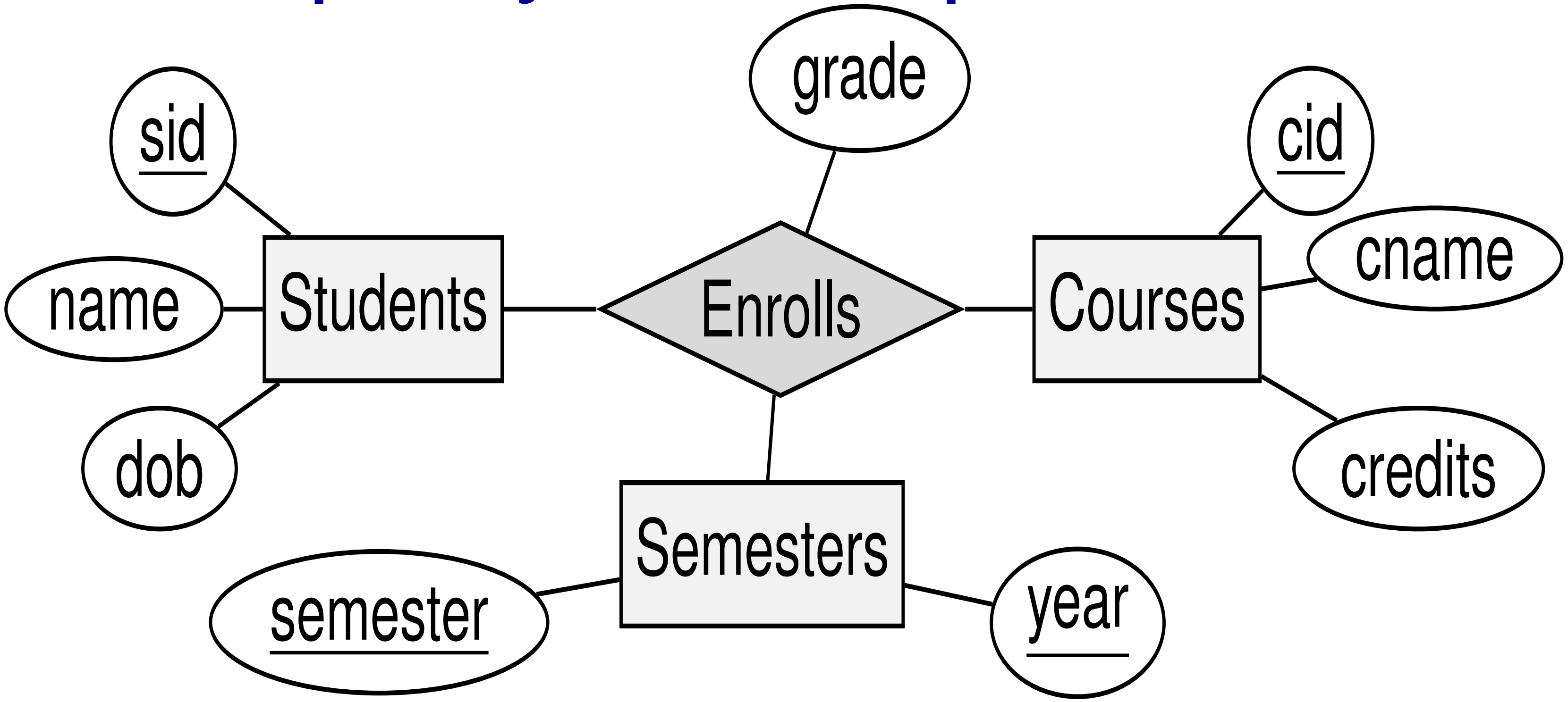
- Each instance of Enrolls has attributes $\{sid, cid, grade\}$
- $A' = \emptyset$, $E' = \{\text{Students}, \text{Courses}\}$
- Key(Enrolls) = $\{sid, cid\}$
- Each (sid,cid) appears at most once in `Enrolls` relationship set

Relationship Keys: Example 2



- Each instance of Enrolls has attributes $\{\mathit{sid}, \mathit{cid}, \mathit{year}, \mathit{semester}, \mathit{grade}\}$
- $A' = \{\text{year, semester}\}$, $E' = \{\text{Students, Courses}\}$
- Key(Enrolls) = $\{\mathit{sid}, \mathit{cid}, \mathit{year}, \mathit{semester}\}$
- Each (sid,cid,year,semester) appears at most once in `Enrolls` relationship set

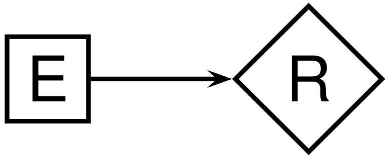
Relationship Keys: Example 3



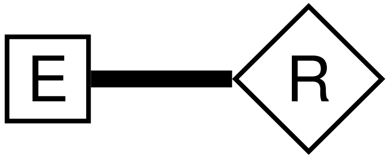
- Each instance of Enrolls has attributes $\{sid, cid, year, semester, grade\}$
- $A' = \emptyset, \quad E' = \{\text{Students}, \text{Courses}, \text{Semesters}\}$
- Key(Enrolls) = $\{sid, cid, year, semester\}$
- Each (sid,cid,year,semester) appears at most once in `Enrolls` relationship set

Relationship Constraints

- Let *R* be a relationship set that involves entity set *E*
- Key constraint on *E* wrt *R*
  - Each instance of *E* can participate in at most one instance of *R*

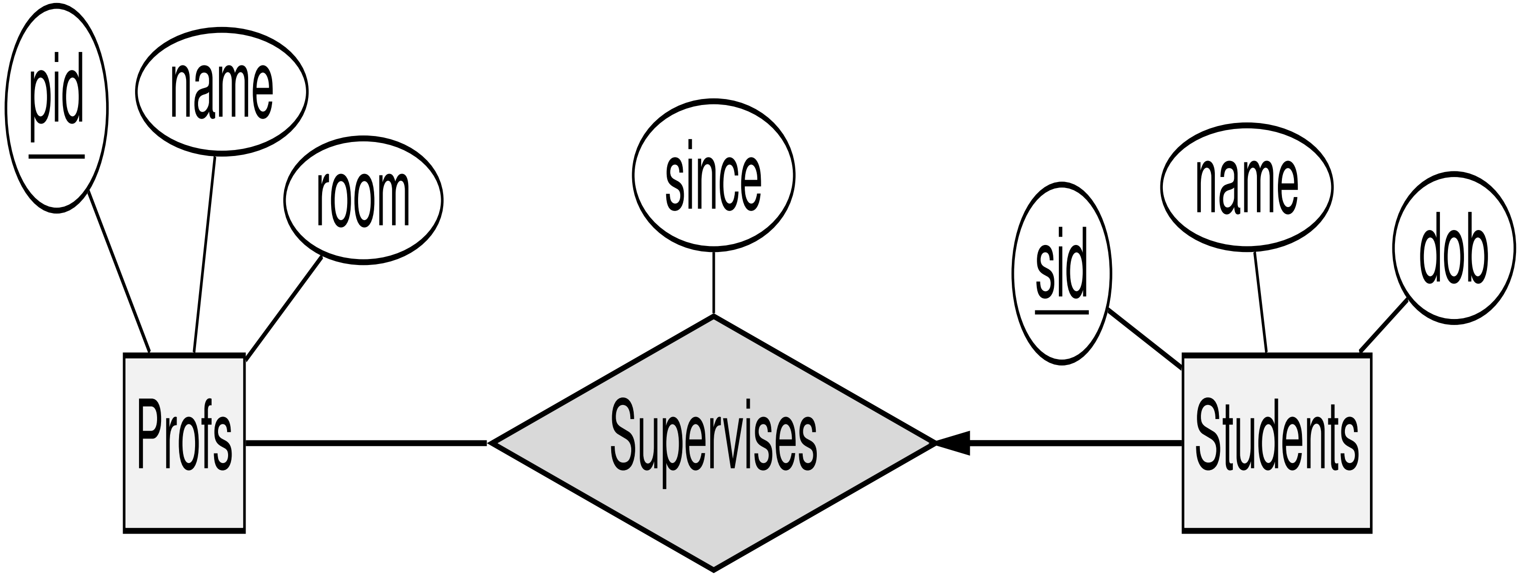


- Total participation constraint on *E* wrt *R*
  - Each instance of *E* must participate in at least one instance of *R*



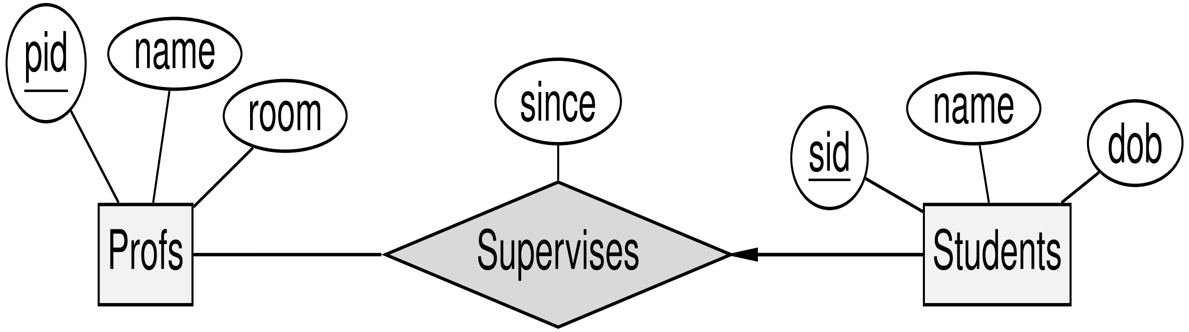
Key Constraints

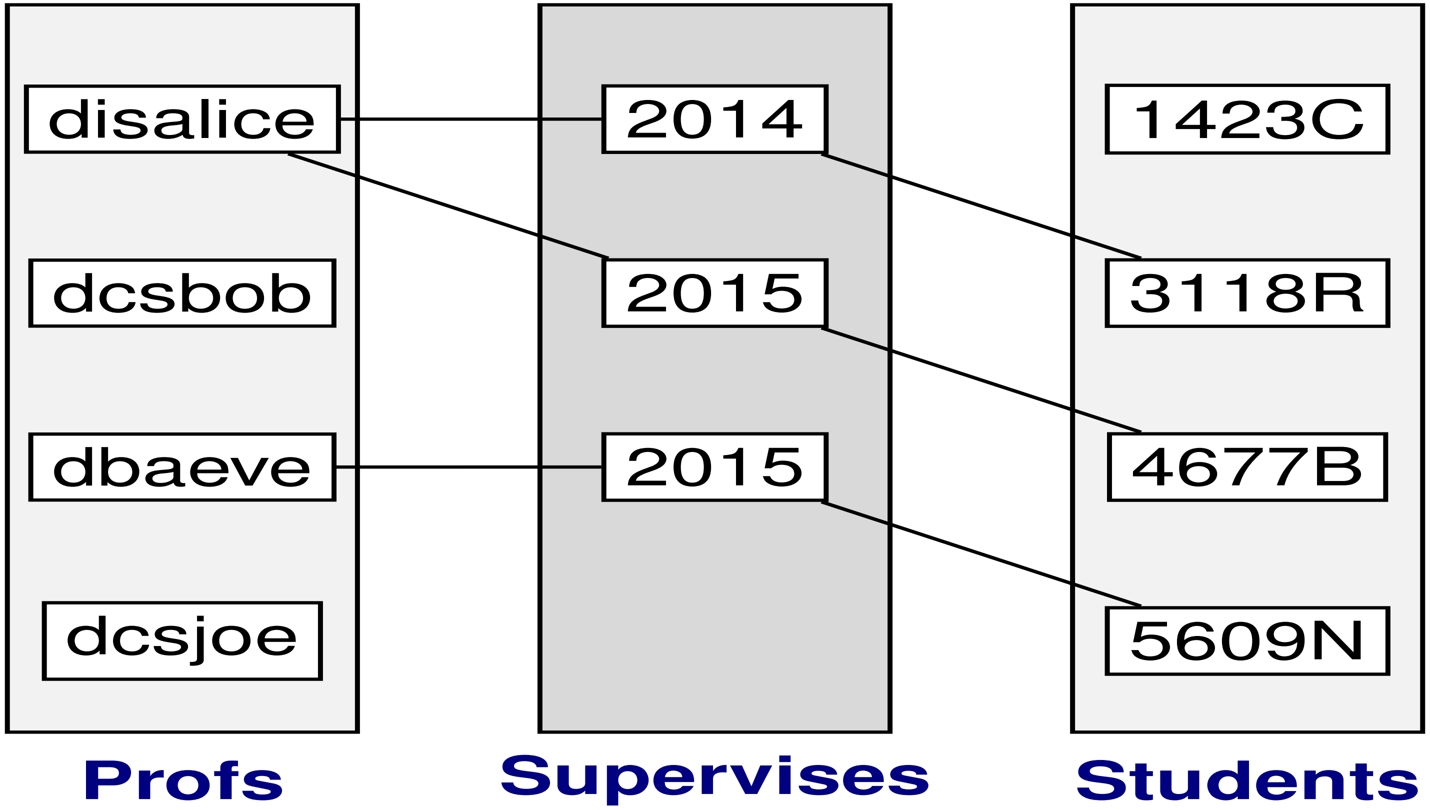
- Each student can be supervised by at most one professor
- **one-to-many** relationship from Profs to Students / **many-to-one** relationship from Students to Profs
    - Each professor can supervise many students
    - Each student can be supervised by one professor



- Key(Supervises) = $\{$sid$\}$

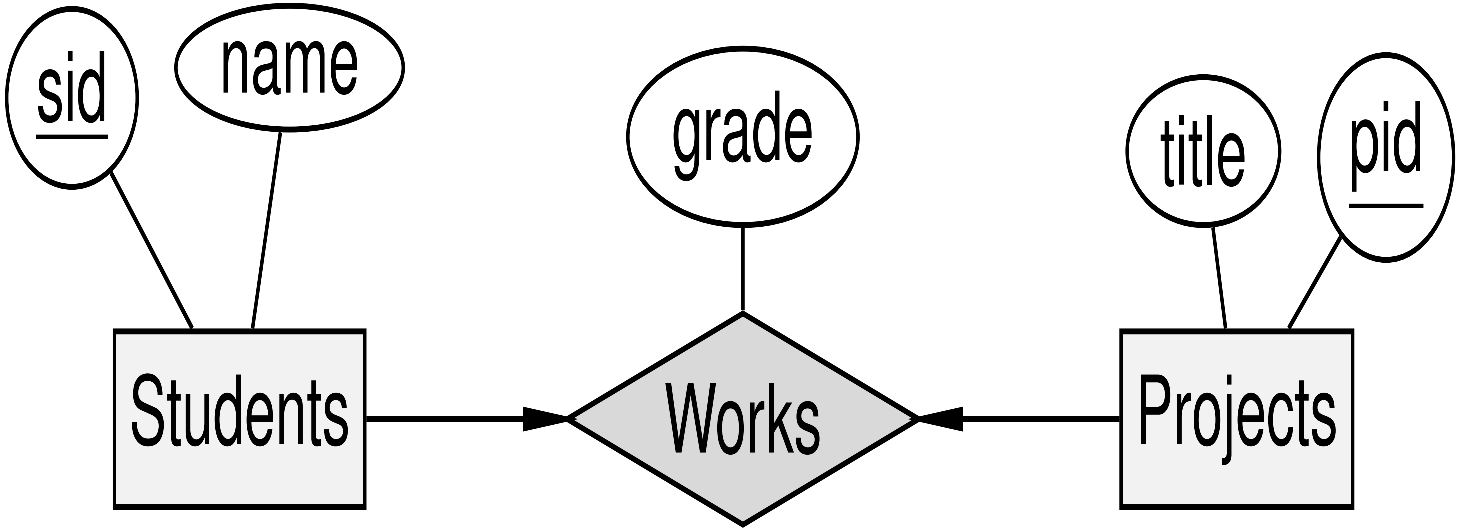
Key Constraints (cont.)





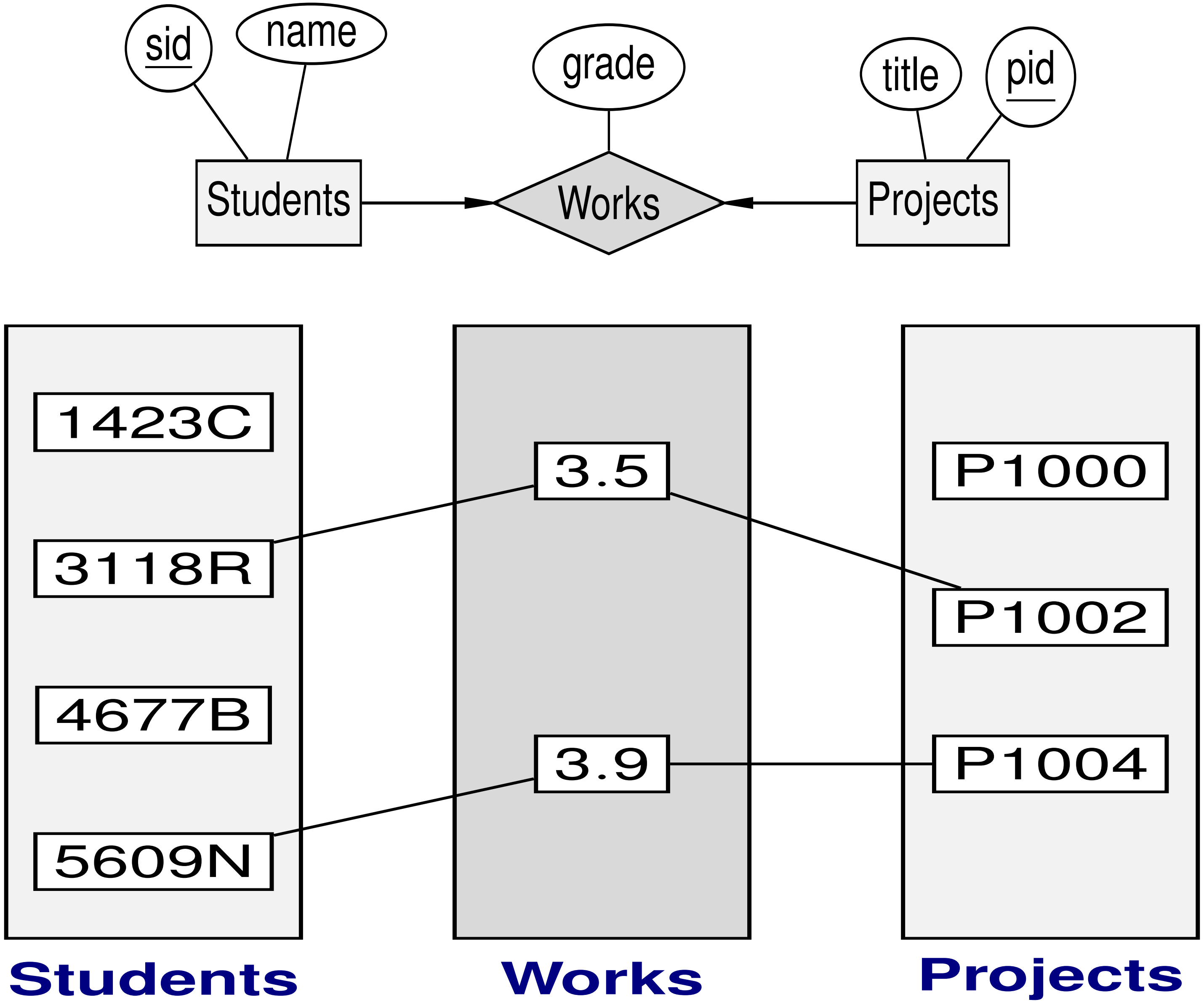
Key Constraints: 1-to-1 Relationships

- **One-to-one** relationship between `Students` and `Projects`
- Each student can work on at most one project
- Each project can be worked on by at most one student



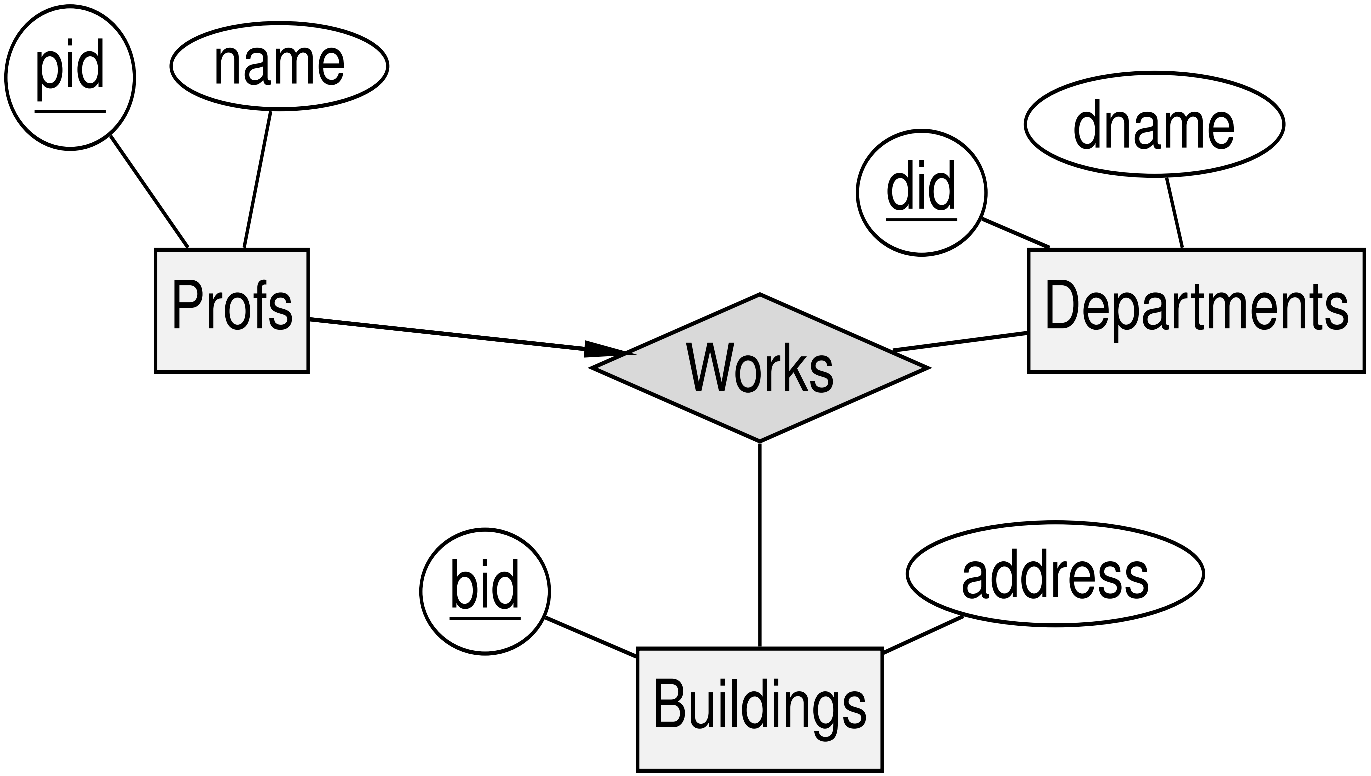
- Key(Works) = $\{$sid$\}$ or $\{$pid$\}$

Key Constraints: 1-to-1 Relationships (cont.)

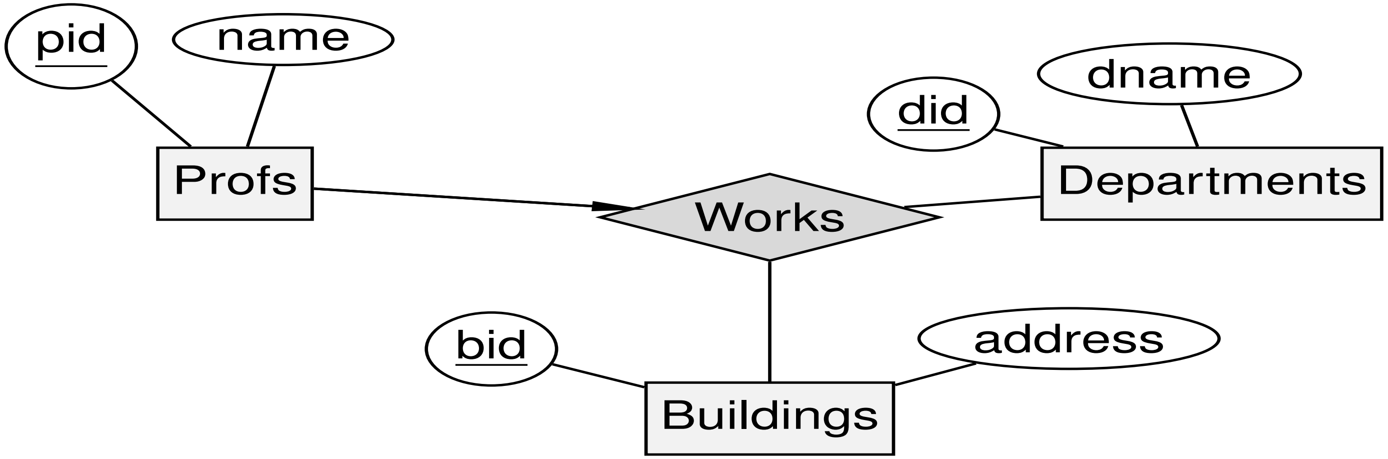


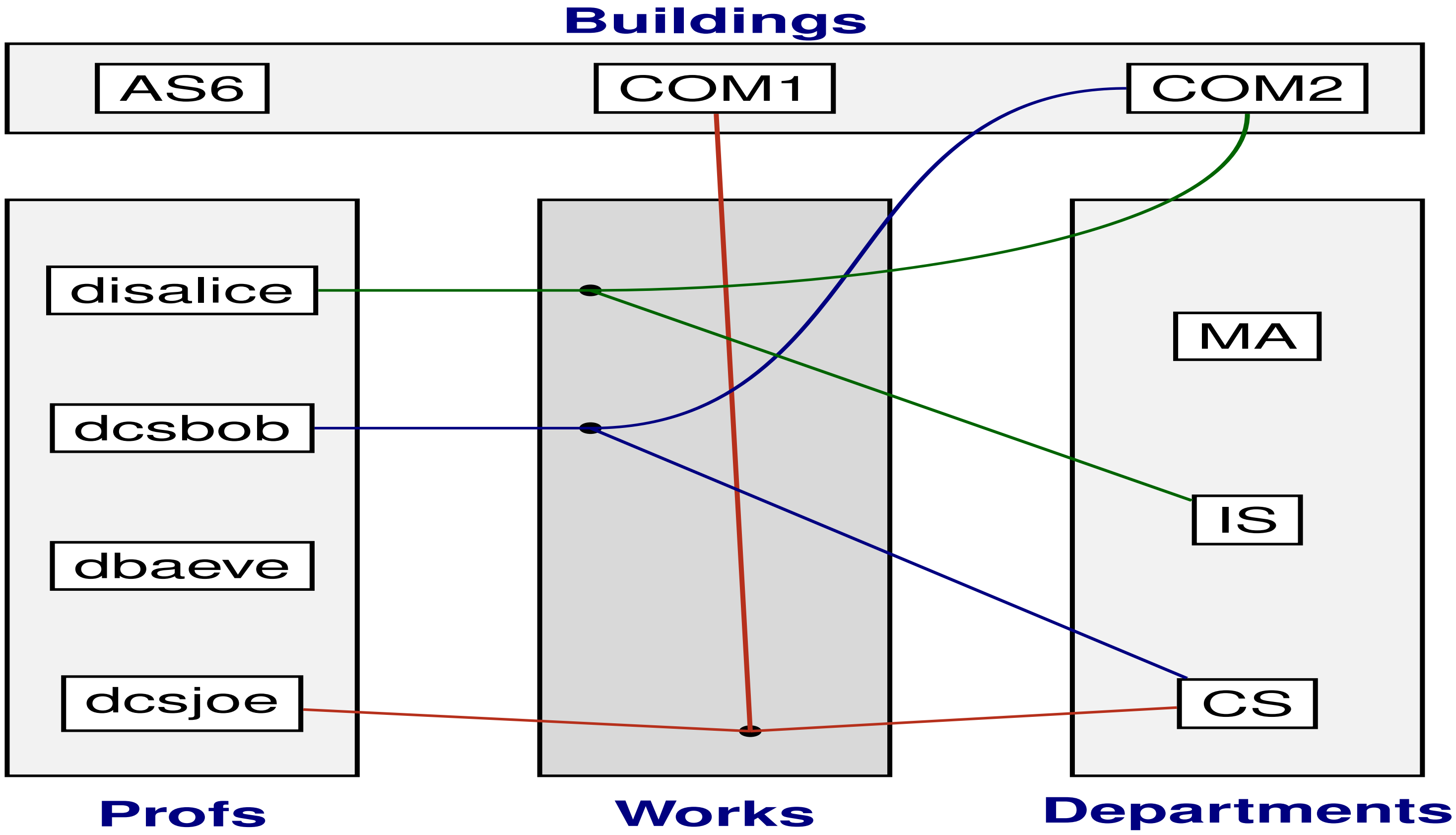
Key Constraints: N-ary Relationships

- Each professor can work in at most one department located at some building



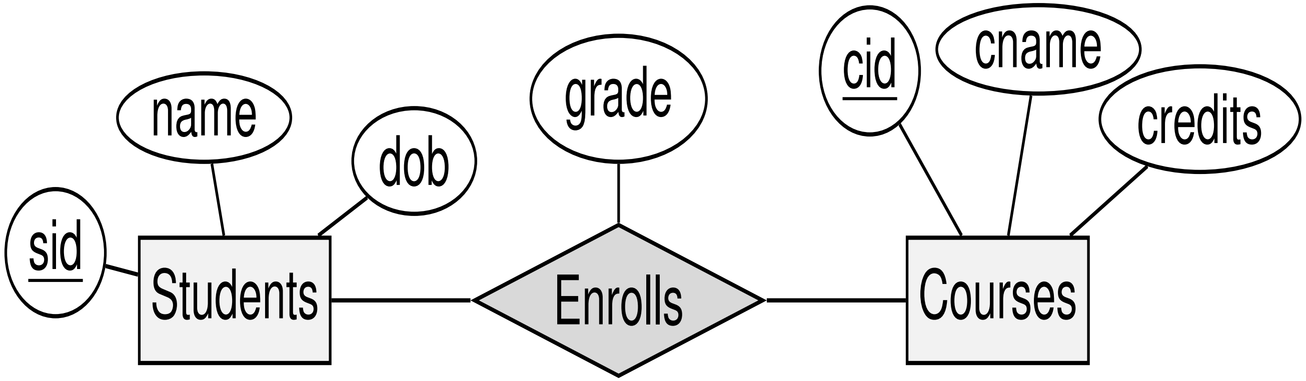
- Key(Works) = {pid}


pid
name
did
dname
Profs
Works
Departments
bid
address
Buildings

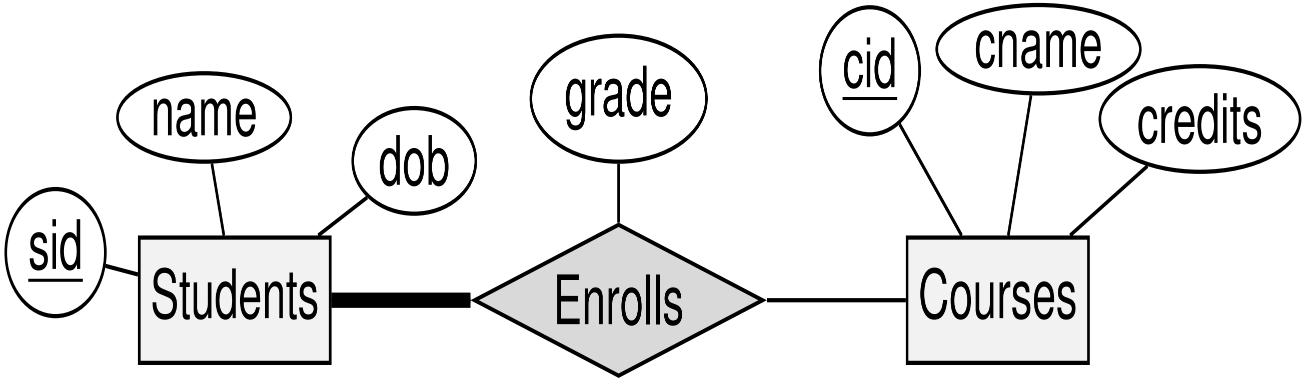

Buildings
AS6
COM1
COM2
disalice
dcsbob
dbaeve
dcsjoe
MA
IS
CS
Profs
Works
Departments

Participation Constraints

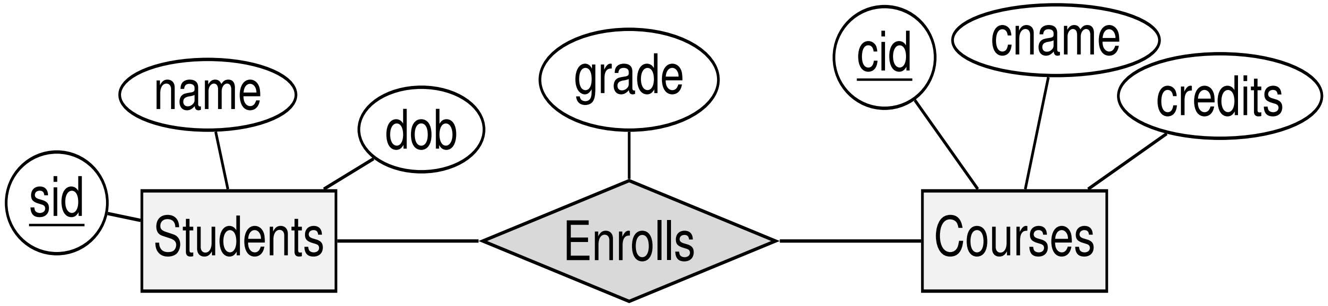
- **Participation constraint** - Is the participation of an entity set in a relationship set mandatory?
- **Partial participation constraint**: each student can enroll in 0 or more courses

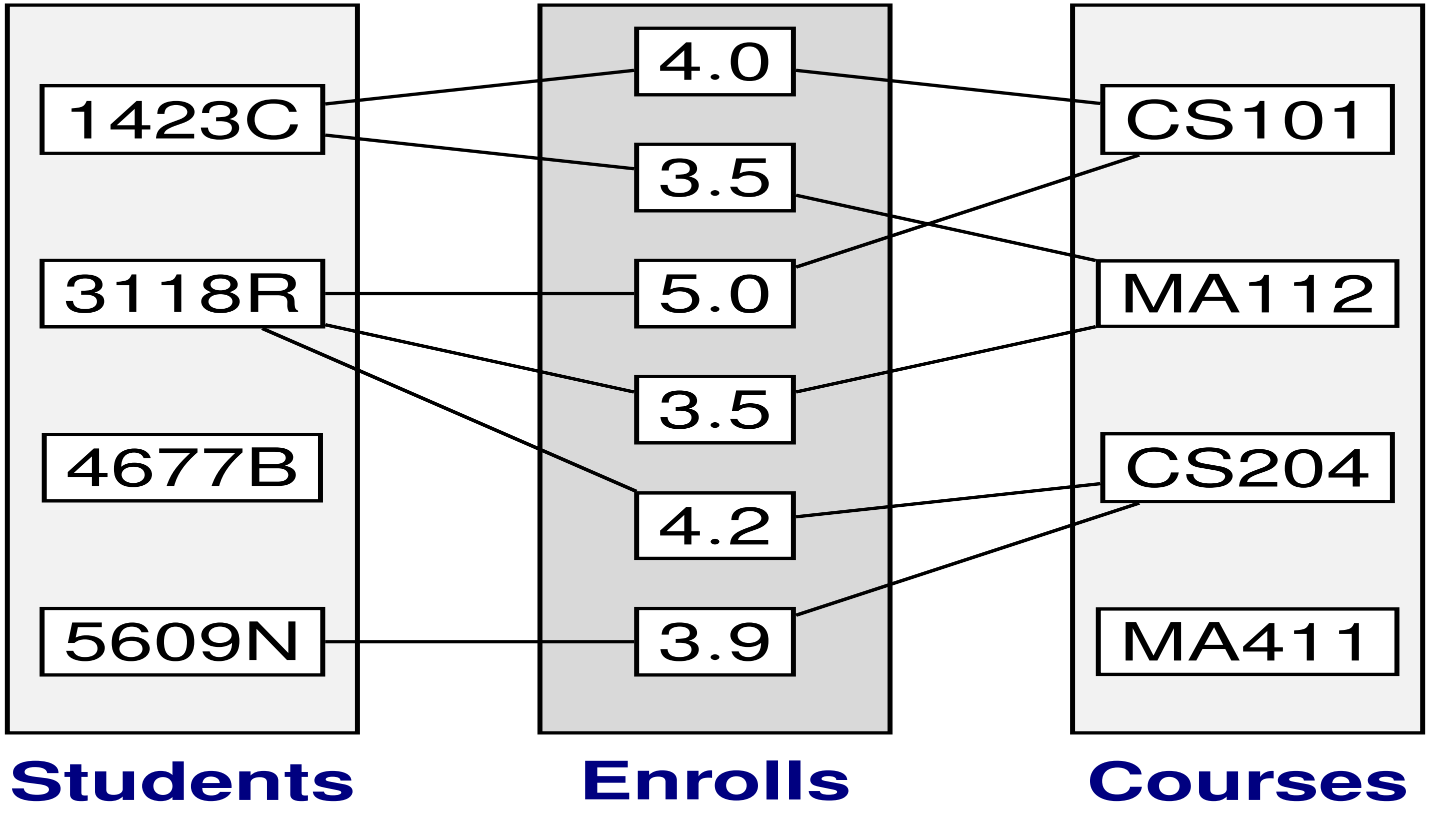


- **Total participation constraint**: each student must enroll in at least one course

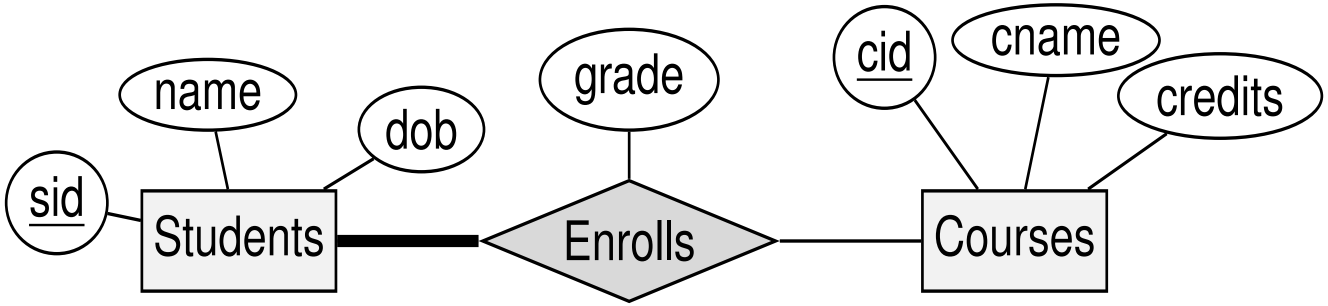


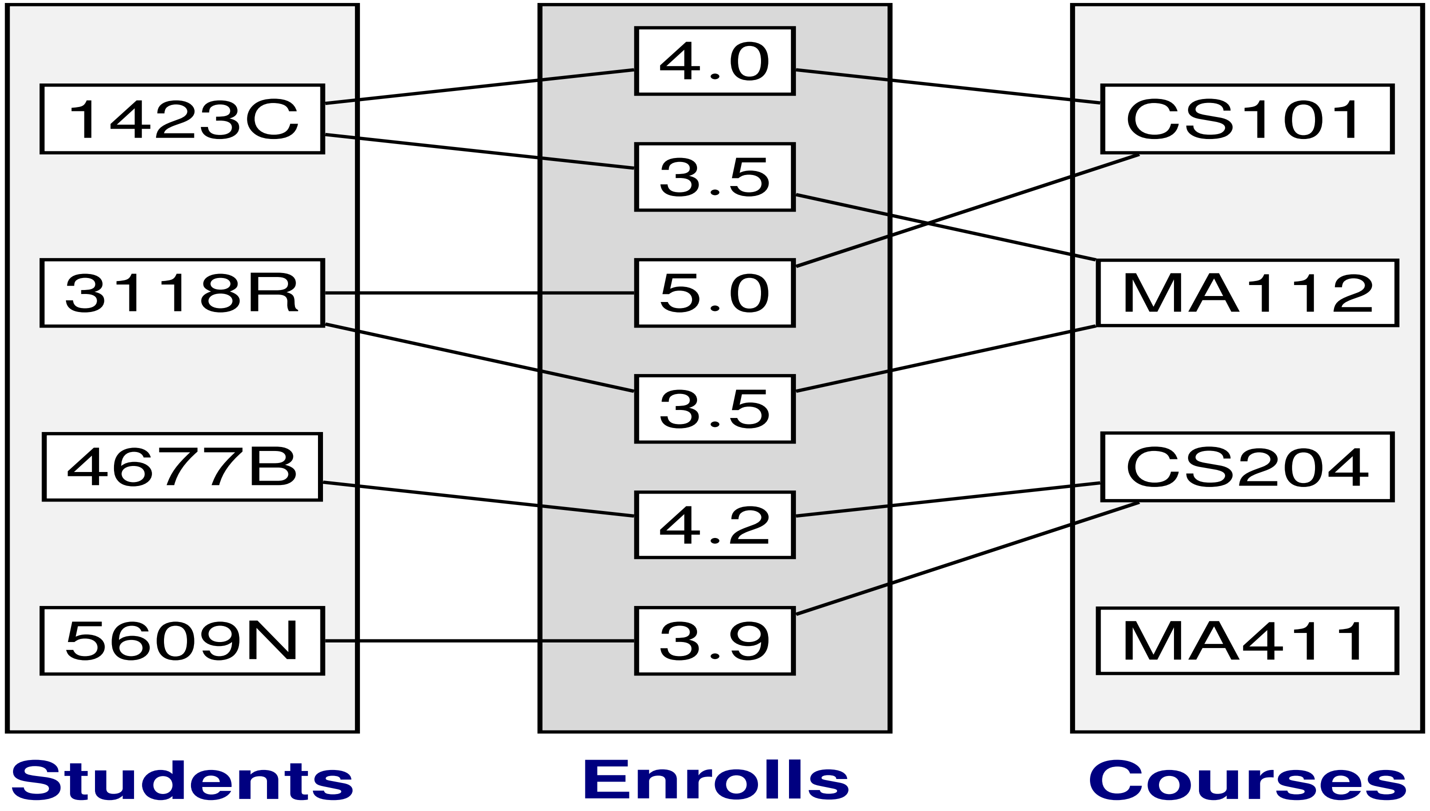
Partial Participation Constraints



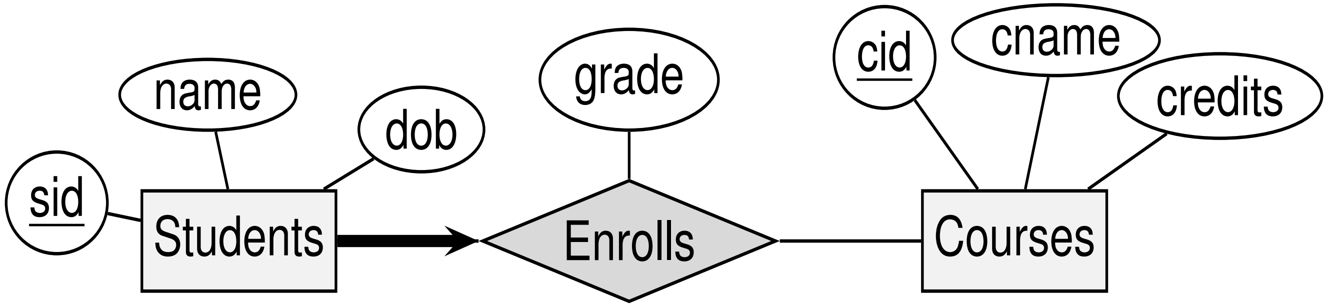


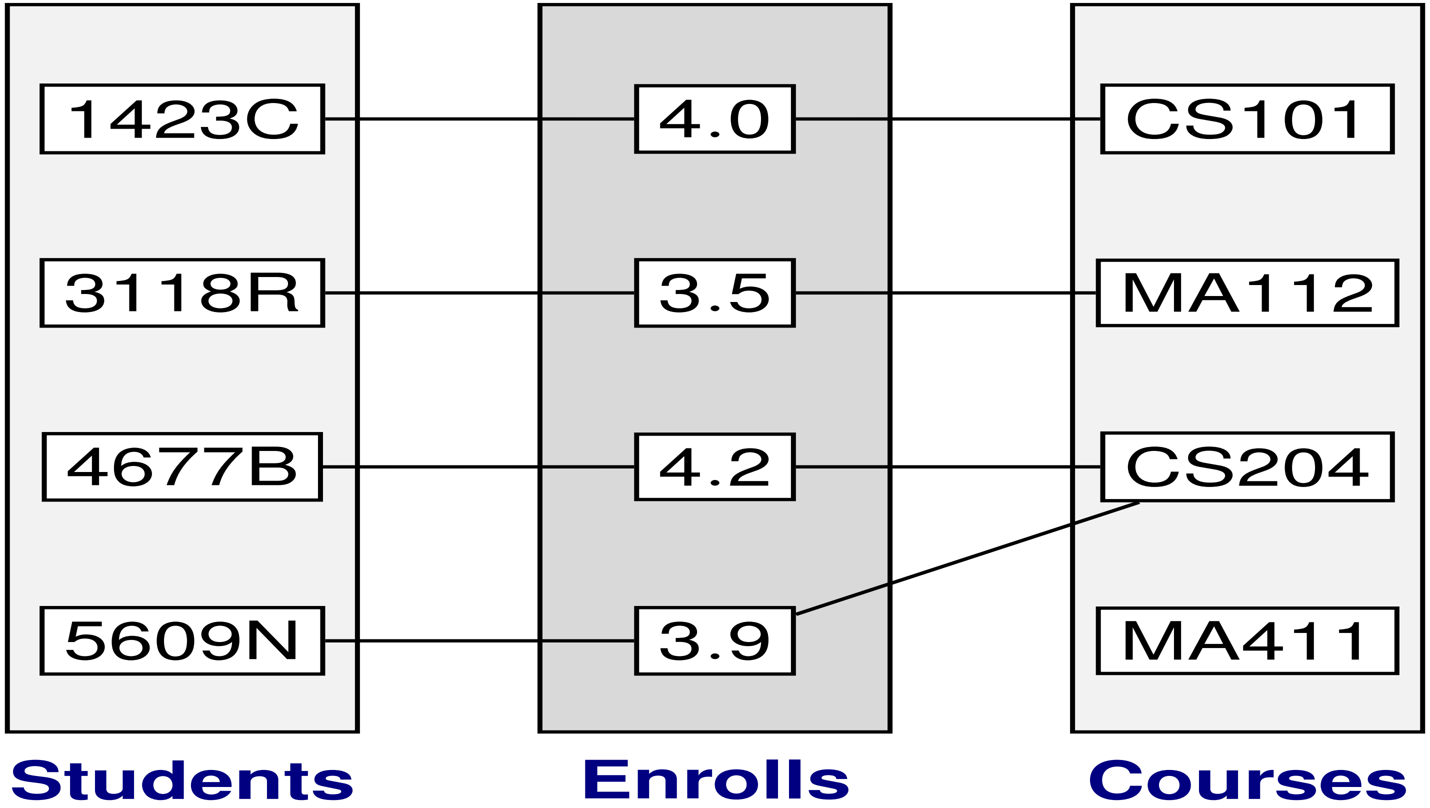
Total Participation Constraints





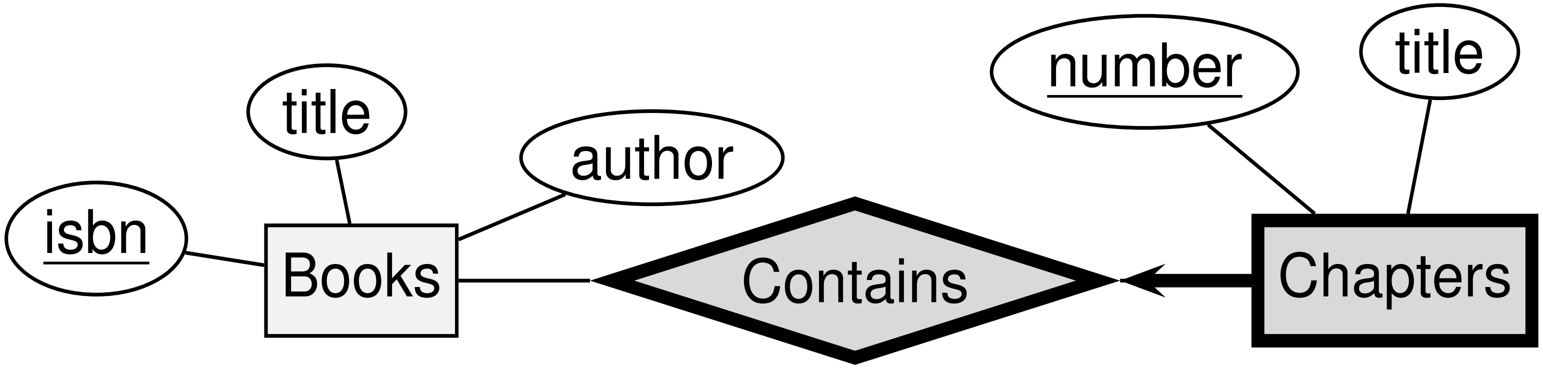
Key & Total Participation Constraints





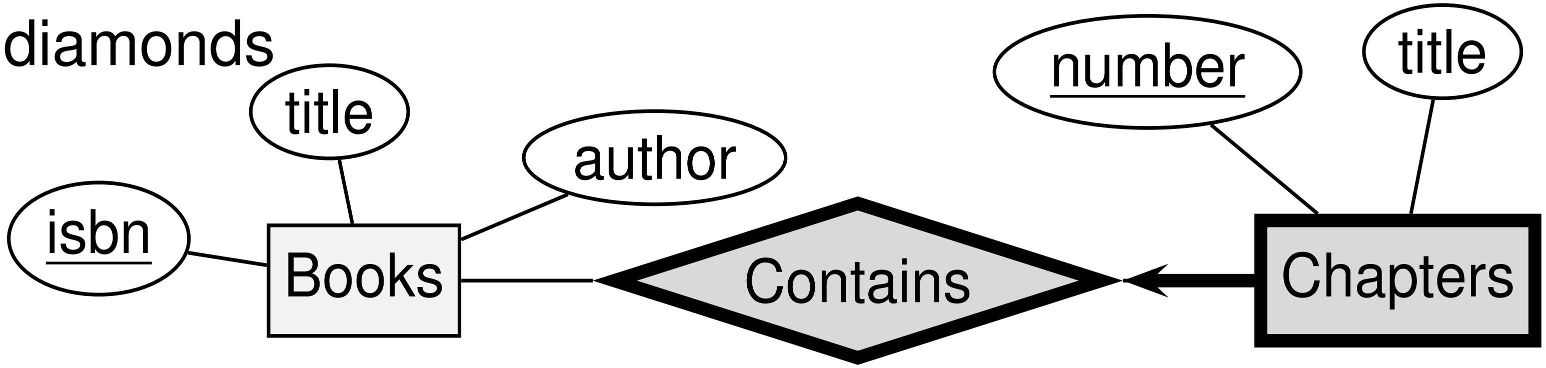
Weak Entity Sets

- **Weak entity set** is an entity set that does not have its own key
- A *weak entity* can only be uniquely identified by considering the primary key of another entity (called **owner entity**)
  - There must be a *many-to-one relationship* (called **identifying relationship**) from the weak entity set to an owner entity set
  - Weak entity set must have *total participation* in identifying relationship

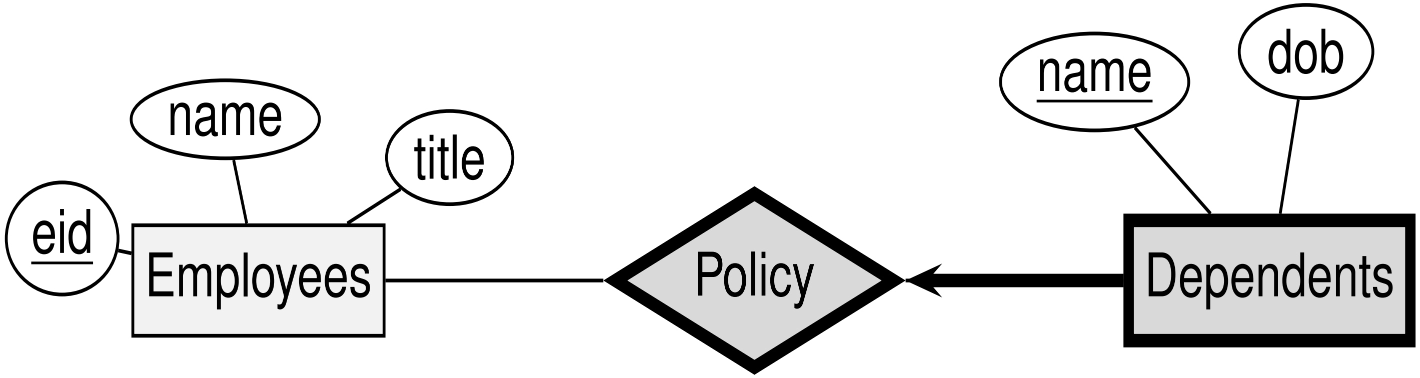


Weak Entity Sets (cont.)

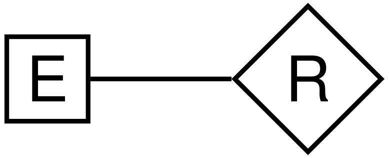
- **Partial key** of a weak entity set is a set of attributes of weak entity set that uniquely identifies a weak entity for a given owner entity
- A weak entity's existence is dependent on the existence of its owner entity
  - If an owner entity is deleted, then all the weak entities that are dependent on that owner entity will be deleted
- Weak entity sets are represented by dark lined rectangles
- Identifying relationship sets are represented by dark lined diamonds



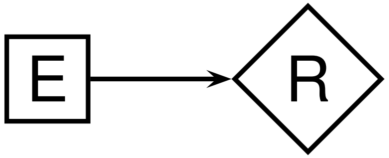
Weak Entity Sets: Another Example



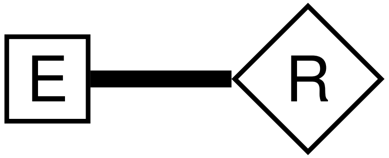
Summary of Relationship Constraints



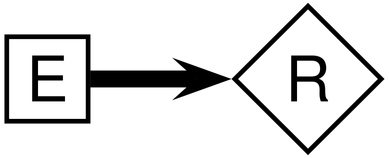
Each instance of E participates in 0 or more instances of R



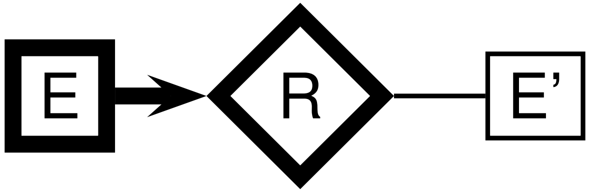
Each instance of E participates in at most one instance of R



Each instance of E participates in at least one instance of R



Each instance of E participates in exactly one instance of R

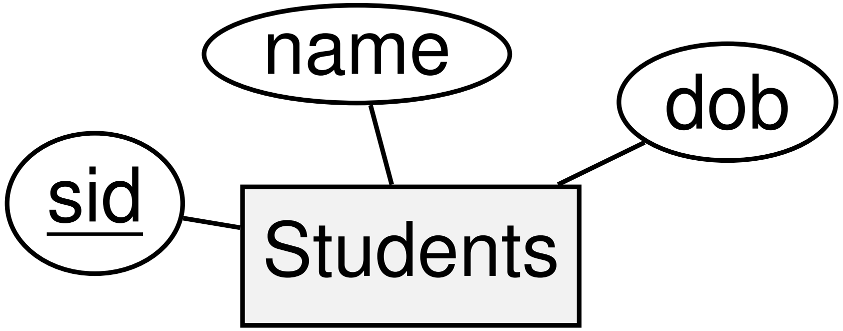


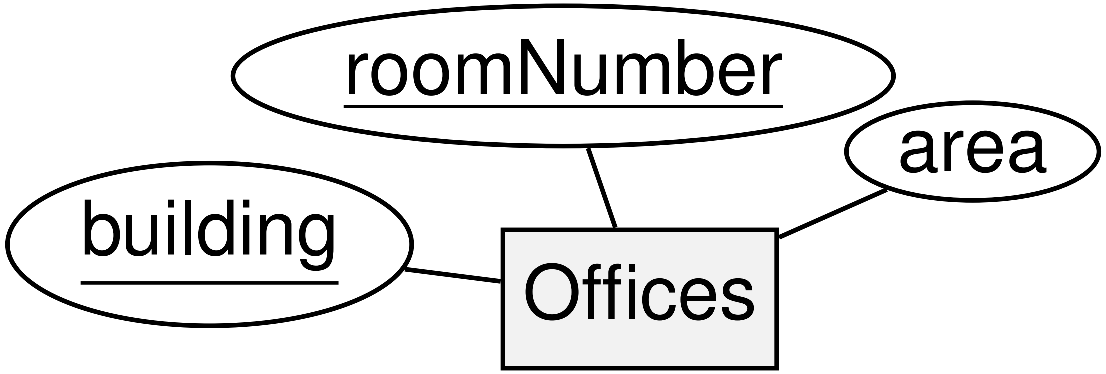
E is a weak entity set with identifying owner E' & identifying relationship set R

Database Design Process

1. **Requirement Analysis** - find out the data/application/performance requirements of the enterprise
2. **Conceptual Database Design** - capture data requirements using a conceptual schema
3. **Logical Database Design** - map conceptual schema to logical schema supported by DBMS
4. **Schema Refinement** - improve logical schema design using data constraints
5. **Physical Database Design** - use performance requirements to design physical schema
6. **Application & Security Design** - specify access control policies

Entity Sets

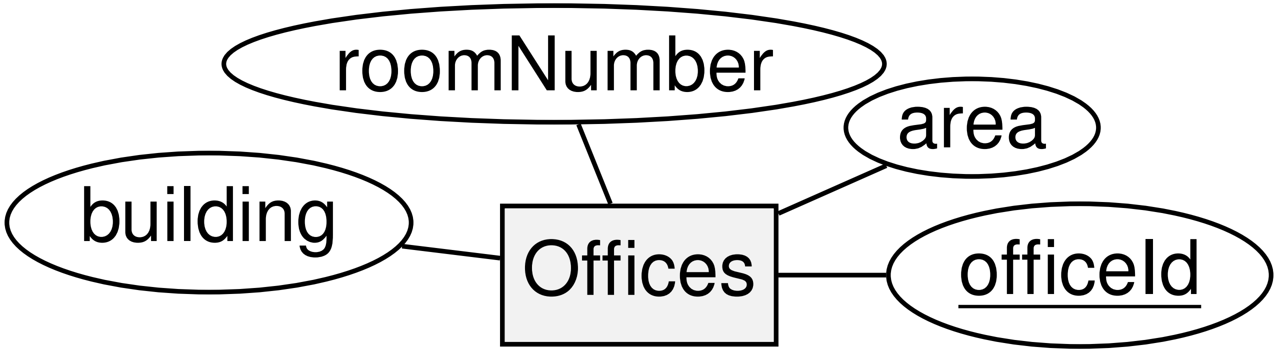




```
create table Students (
        sid     char(20),
        name    char(30),
        dob     date,
        primary key (sid)
);
```

```
create table Offices (
        building        char(10),
        roomNumber  char(7),
        area            integer,
        primary key (building, roomNumber)
);
```

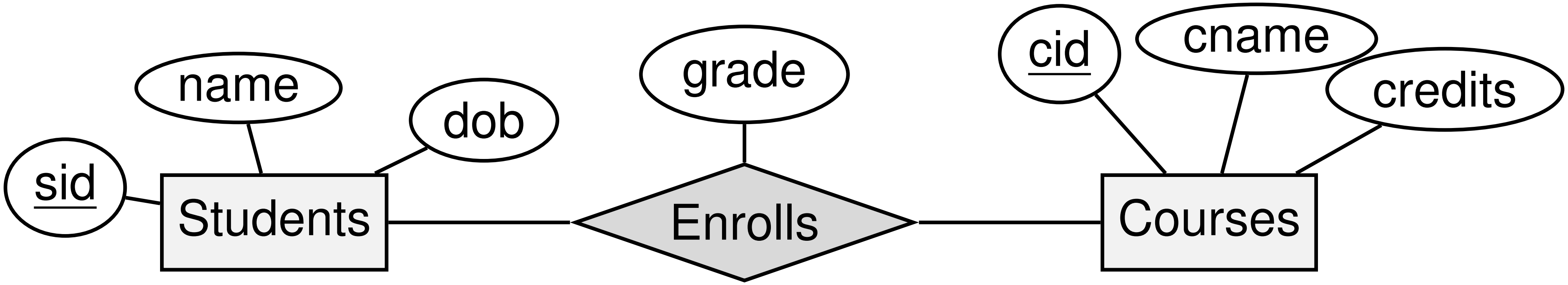
Entity Sets with Candidate Keys



Assume that (building, roomNumber) is a candidate key of Offices

```
create table Offices (
        officeId      char(10) primary key,
        building      char(10) not null,
        roomNumber char(7) not null,
        area          integer,
        unique (building, roomNumber)
);
```

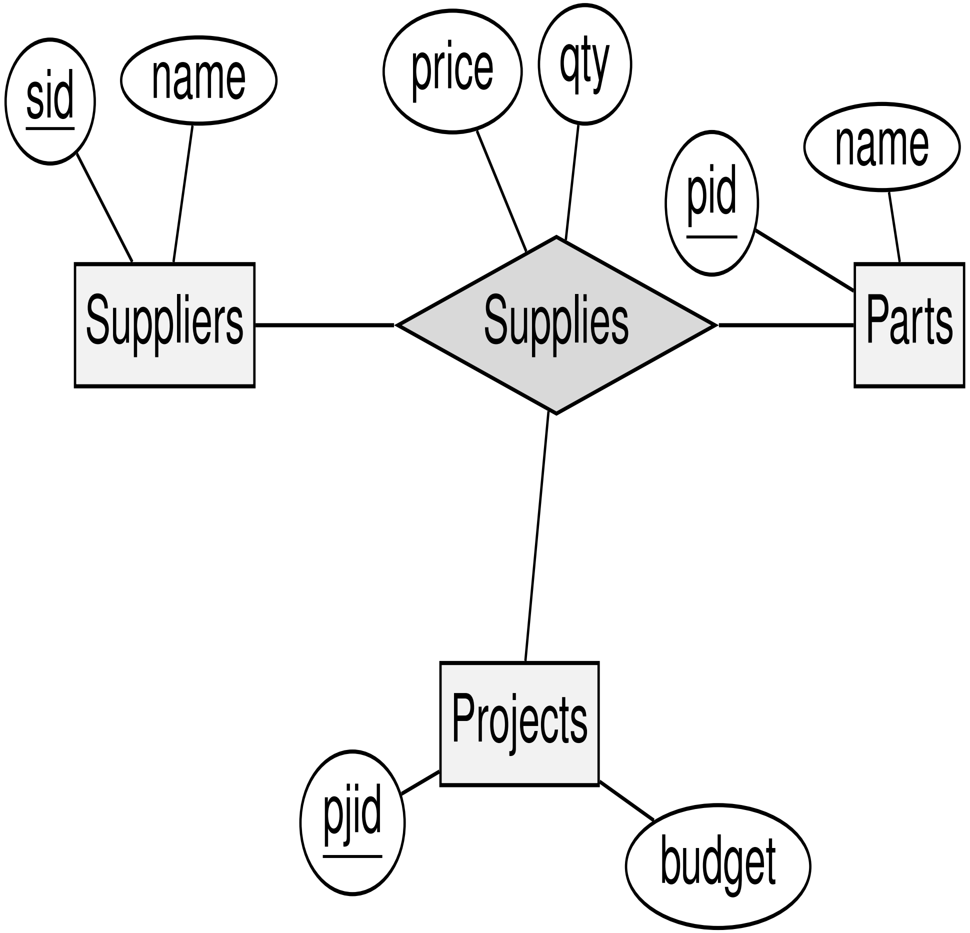
Relationship Sets w/o Constraints



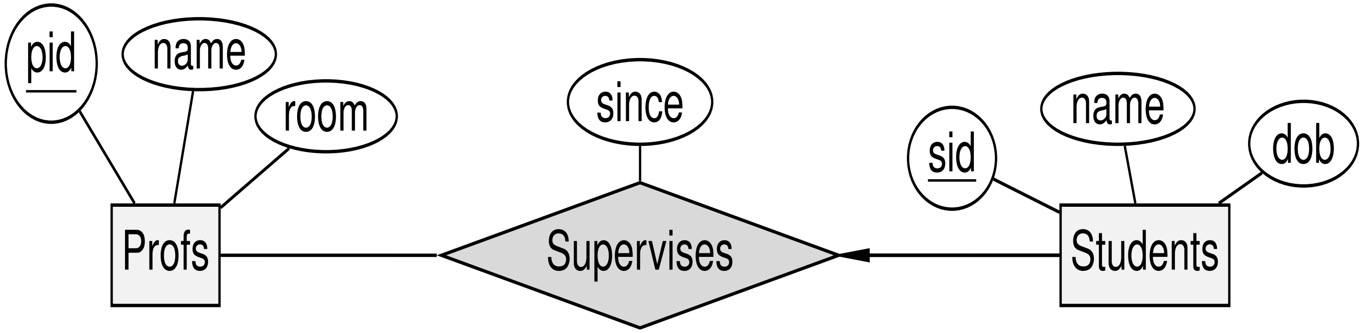
```
create table Enrolls (
        sid     char(20) references Students,
        cid     char(5),
        grade   numeric,
        primary key (sid, cid),
        foreign key  (cid) references Courses
);
```

Relationship Sets w/o Constraints

```
create table Supplies (
  sid      char(10),
  pid      char(10),
  pjid     char(10),
  price    real,
  qty      integer,
  primary key (sid, pid, pjid),
  foreign key  (sid)
      references Suppliers,
  foreign key  (pid)
      references Parts,
  foreign key  (pjid)
      references Projects
);
```

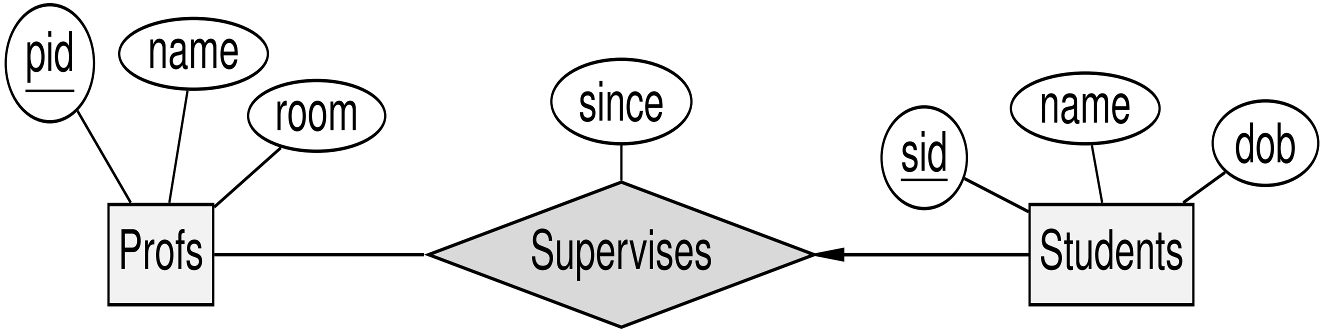


Relationship Sets with Key Constraints



- First approach: Represent `Supervises` using a separate table
  - **Profs** (pid, name, room)
  - **Students** (sid, name, dob)
  - **Supervises** (sid, pid, since)
- Second approach: Combine `Supervises` & `Students` into one table
  - **Profs** (pid, name, room)
  - **Students** (sid, name, dob, pid, since)

Relationship Sets with Key Constraints (cont.)



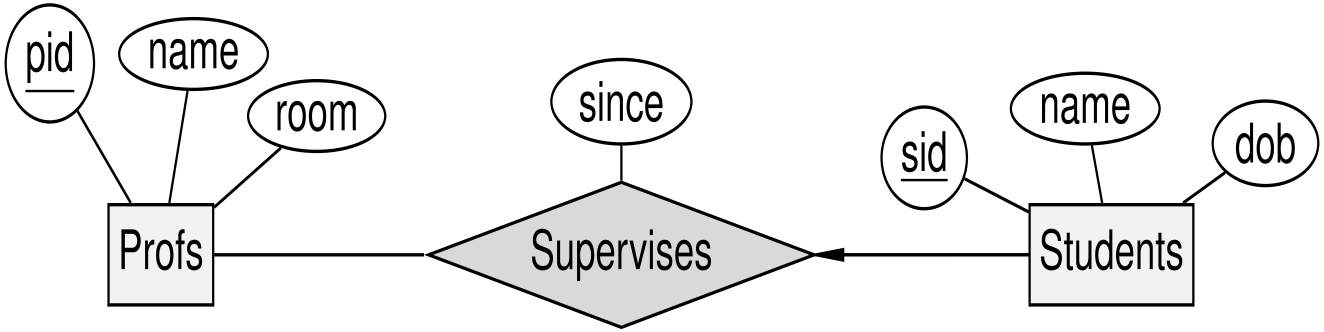
First approach: Represent `Supervises` using a separate table

```
create table Profs (
        pid     char(7),
        name    char(30),
        room    char(6),
        primary key (pid)
);
```

```
create table Students (
        sid     char(20),
        name    char(30),
        dob     date,
        primary key (sid)
);
```

```
create table Supervises (
        sid     char(20),
        pid     char(7) not null,
        since   date,
        primary key (sid),
        foreign key  (sid)
          references Students,
        foreign key  (pid)
          references Profs
);
```

Relationship Sets with Key Constraints (cont.)

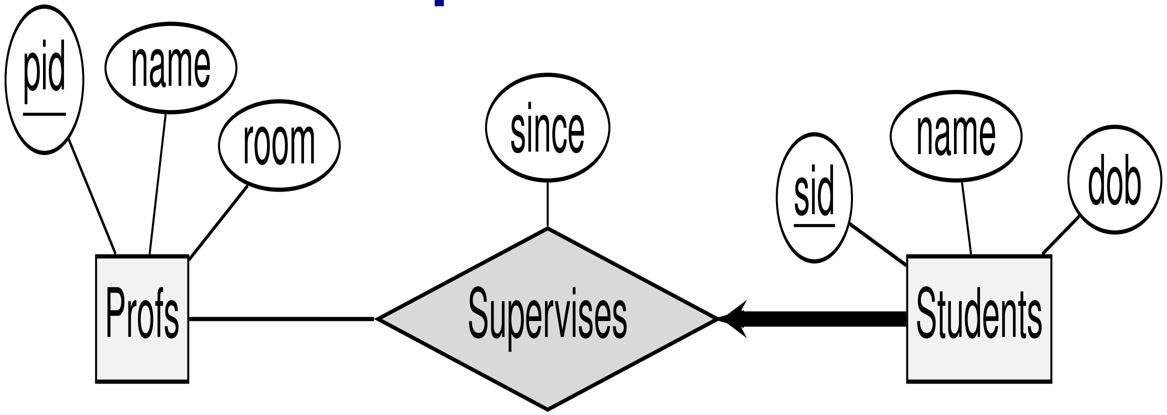


Second approach: Combine `Supervises` & `Students` into one table

```
create table Profs (
        pid     char(7),
        name    char(30),
        room    char(6),
        primary key (pid)
);
```

```
create table Students (
        sid     char(20),
        name    char(30),
        dob     date,
        pid     char(7),
        since   date,
        primary key (sid),
        foreign key  (pid)  references Profs
);
```

Key & Total Participation Constraints



First approach: Represent `Supervises` using a separate table

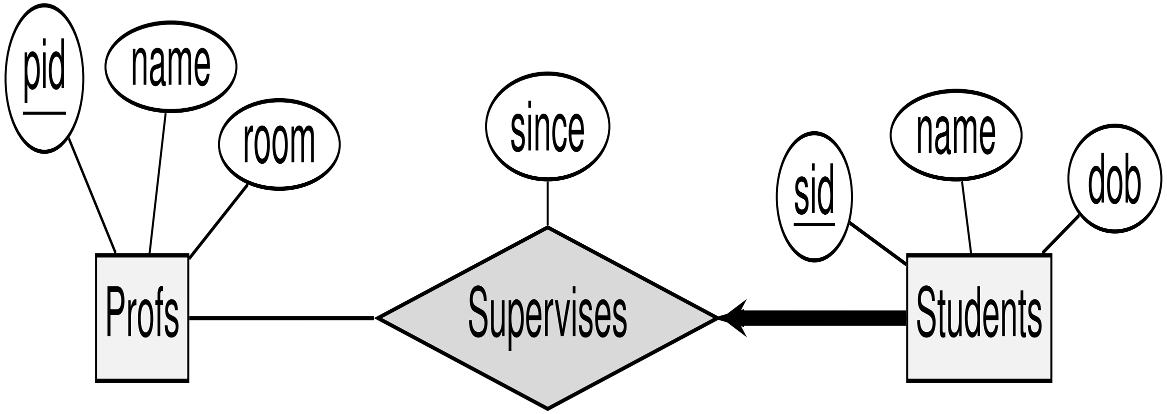
```
create table Profs (
        pid      char(7),
        name     char(30),
        room     char(6),
        primary key (pid)
);
```

```
create table Students (
        sid      char(20),
        name     char(30),
        dob      date,
        primary key (sid)
);
```

```
create table Supervises (
        sid      char(20),
        pid      char(7) not null,
        since    date,
        primary key (sid),
        foreign key  (sid)
          references Students,
        foreign key  (pid)
          references Profs
);
```

Total participation constraint of `Students` w.r.t. `Supervises` is not enforced by database schema!

Key & Total Participation Constraints (cont.)

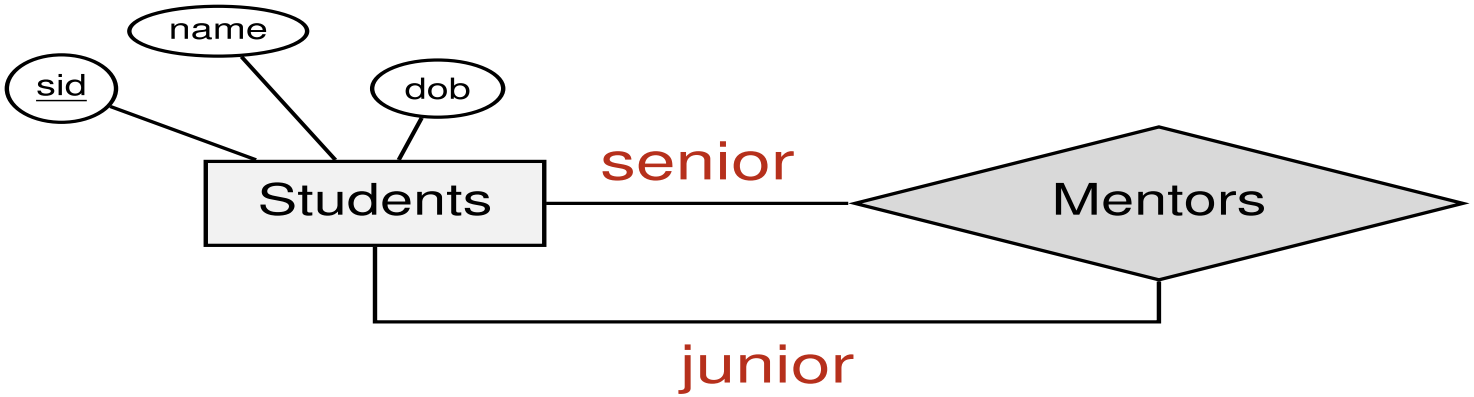


Second approach: Combine `Supervises` & `Students` into one table

```
create table Profs (
        pid     char(7),
        name    char(30),
        room    char(6),
        primary key (pid)
);
```

```
create table Students (
        sid     char(20),
        name    char(30),
        dob     date,
        pid     char(7) not null,
        since   date,
        primary key (sid),
        foreign key  (pid)  references Profs
);
```

Roles in Relationships

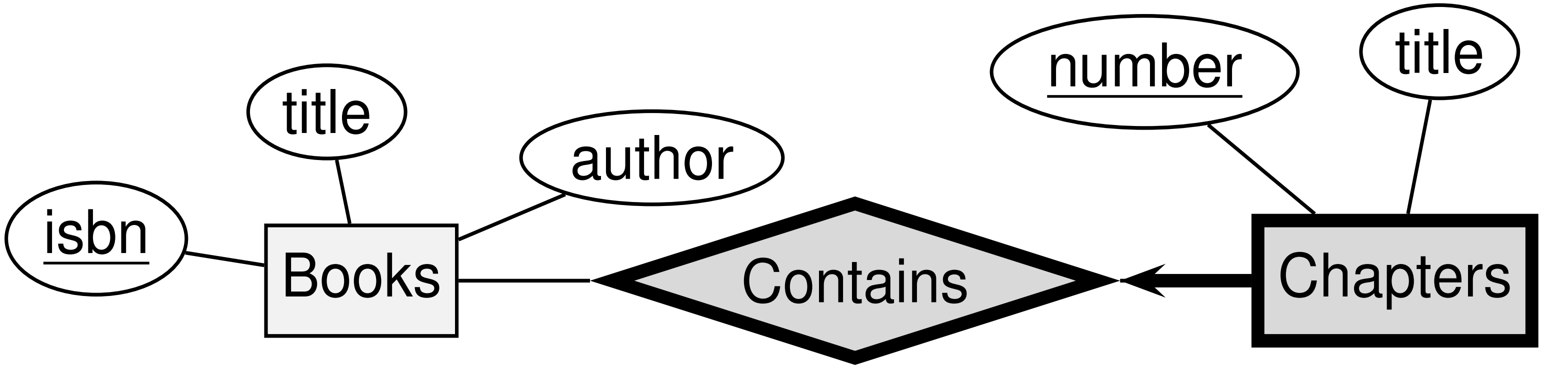


```
create table Students (
        sid     char(20),
        name    char(30),
        dob     date,
        primary key (sid),
);
```

```
create table Mentors (
        seniorSid     char(20),
        juniorSid     char(20),
        primary key (seniorSid, juniorSid),
        foreign key  (seniorSid)
           references Students(sid),
        foreign key  (juniorSid)
           references Students(sid),
        check          (juniorSid <> seniorSid)
);
```

Weak Entity Sets

- Weak entity set & its identifying relationship set are represented by a single relation



```
create table Books (
        isbn    char(30),
        title   char(50),
        author  char(60),
        primary key (isbn)
);
```

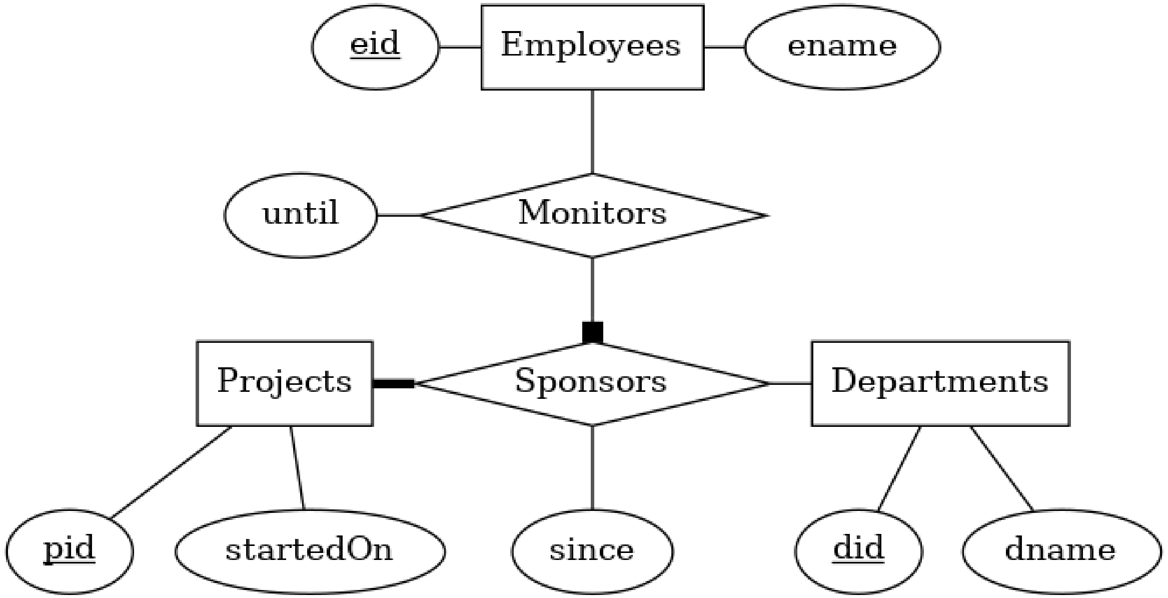
```
create table Chapters (
        number      integer,
        title       char(50),
        isbn        char(30),
        primary key (isbn, number),
        foreign key  (isbn) references Books
                     on delete cascade
);
```

ER Design & Relational Mapping

- **Guidelines for ER Design**
  - ER design should capture as many of the application's constraints as possible
  - ER design must not impose any constraint that is not required in the application
- **Guidelines for relational mapping**
  - Relational schema should enforce as many of the application's constraints as possible using column/table constraints
  - Relational schema must not impose any constraint that is not required in the application

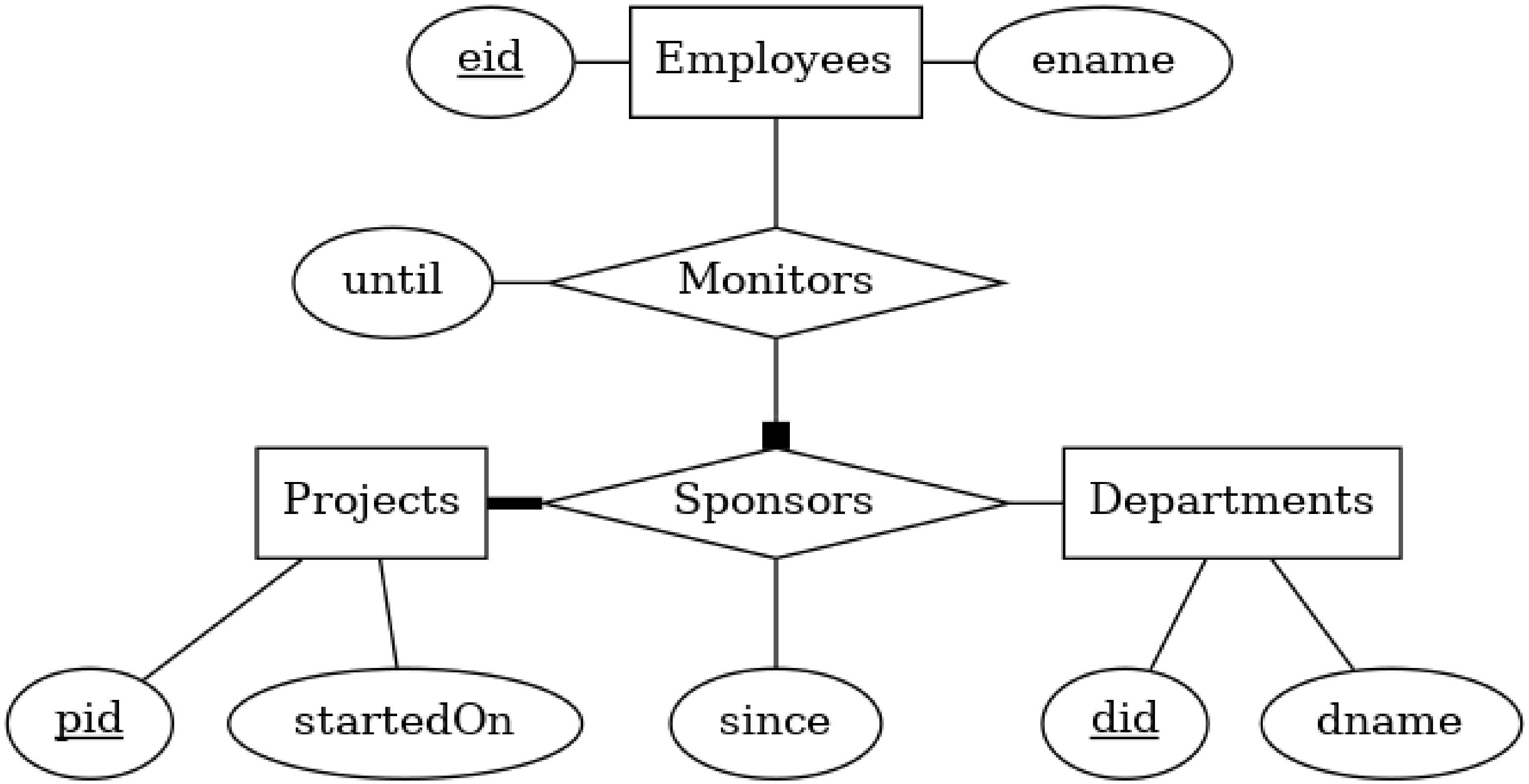
Aggregation

- How to model a relationship between entities & relationships?
- **Example**:
  - Every project is sponsored by at least one department
    - Each sponsorship has a "since" attribute
  - Each sponsorship might be monitored by 0 or more employees
    - Each monitoring has a "until" attribute



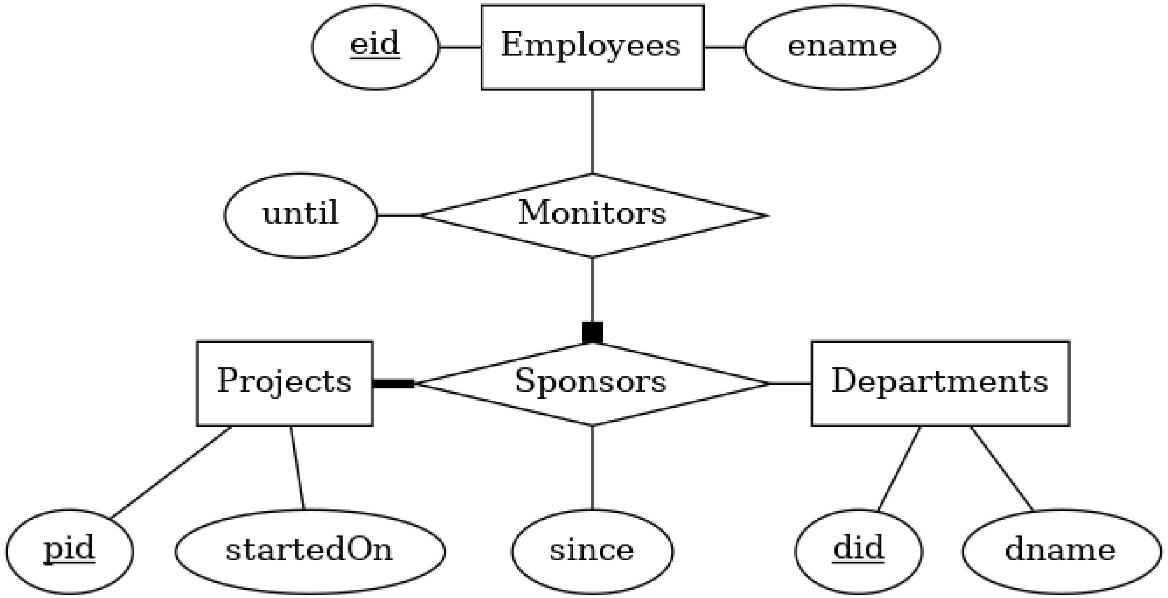
The ■ connected to Sponsors relationship denotes that Sponsors is participating in the Monitors relationship as an aggregation

Aggregation (cont.)



- Each instance of `Monitors` has the following attributes:
  - Key(Employees) = $\{$eid$\}$
  - Key(Sponsors) = $\{$pid, did$\}$
  - Relationship attributes = $\{$until$\}$

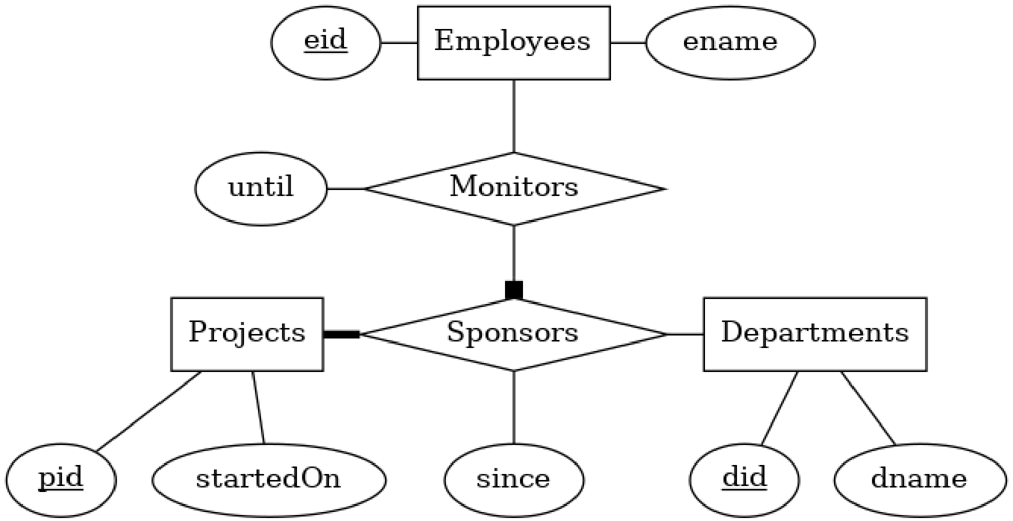
Aggregation: Relational Mapping

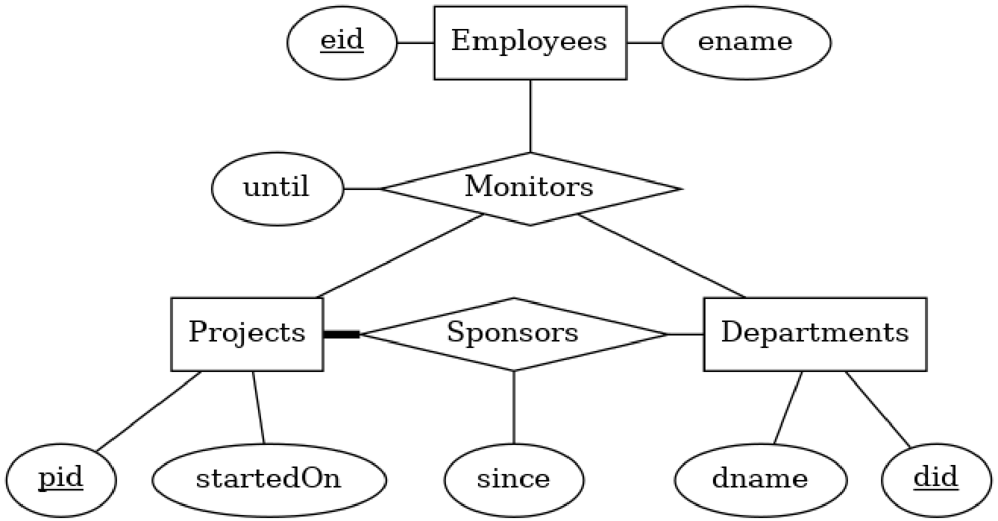


```
create table Sponsors (
    pid     char(30)
            references Projects,
    did     char(30),
            references Departments,
    since   date,
    primary key (pid,did)
);
```

```
create table Monitors (
    eid     char(20) references Employees,
    pid     char(30),
    did     char(30),
    until   date,
    primary key (eid,pid,did),
    foreign key (pid,did)
            references Sponsors (pid,did)
);
```

Aggregation: Is it necessary?



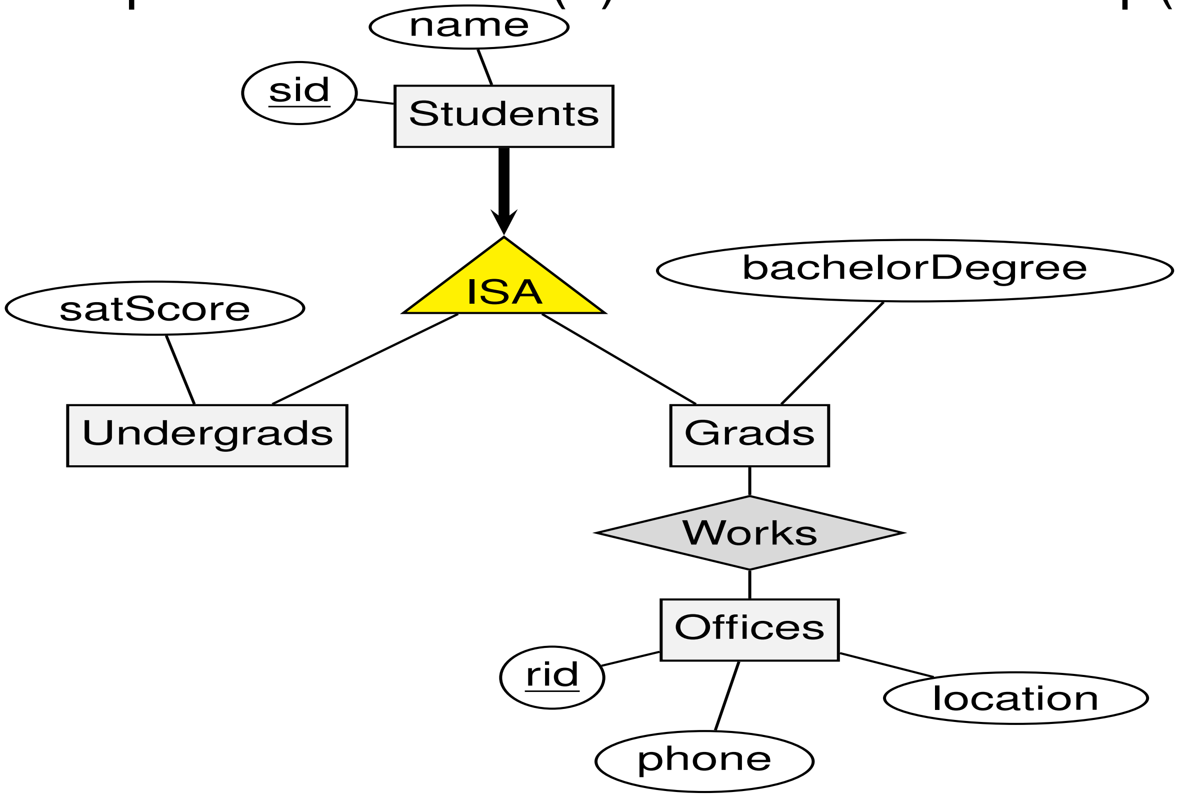


```
create table Monitors (
      eid      char(20) references Employees,
      pid      char(30),
      did      char(30),
      until    date,
      primary key (eid,pid,did),
      foreign key (pid,did)
               references Sponsors (pid,did)
);
```

```
create table Monitors (
      eid      char(20) references Employees,
      pid      char(30), references Projects,
      did      char(30) references Departments,
      until    date,
      primary key (eid,pid,did),
);
```

ISA Hierarchies

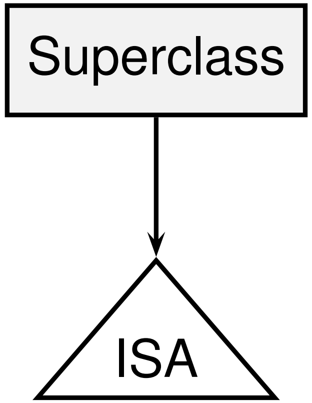
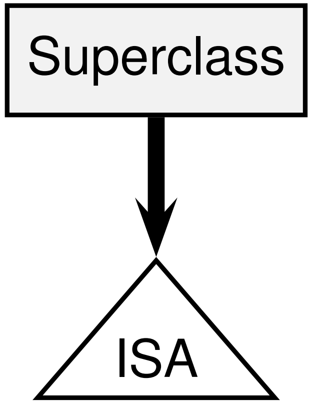
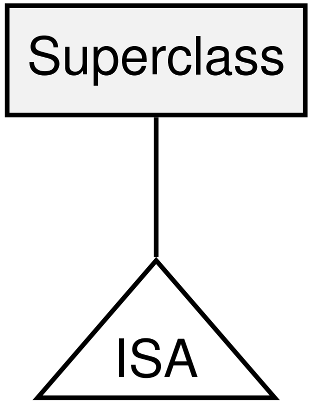
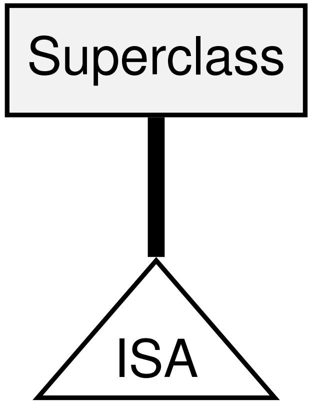
- Sometimes useful to classify an entity set into subclasses
- Every entity in a subclass entity set is an entity in its superclass entity set
- Each subclass has specific attribute(s) and/or relationship(s)



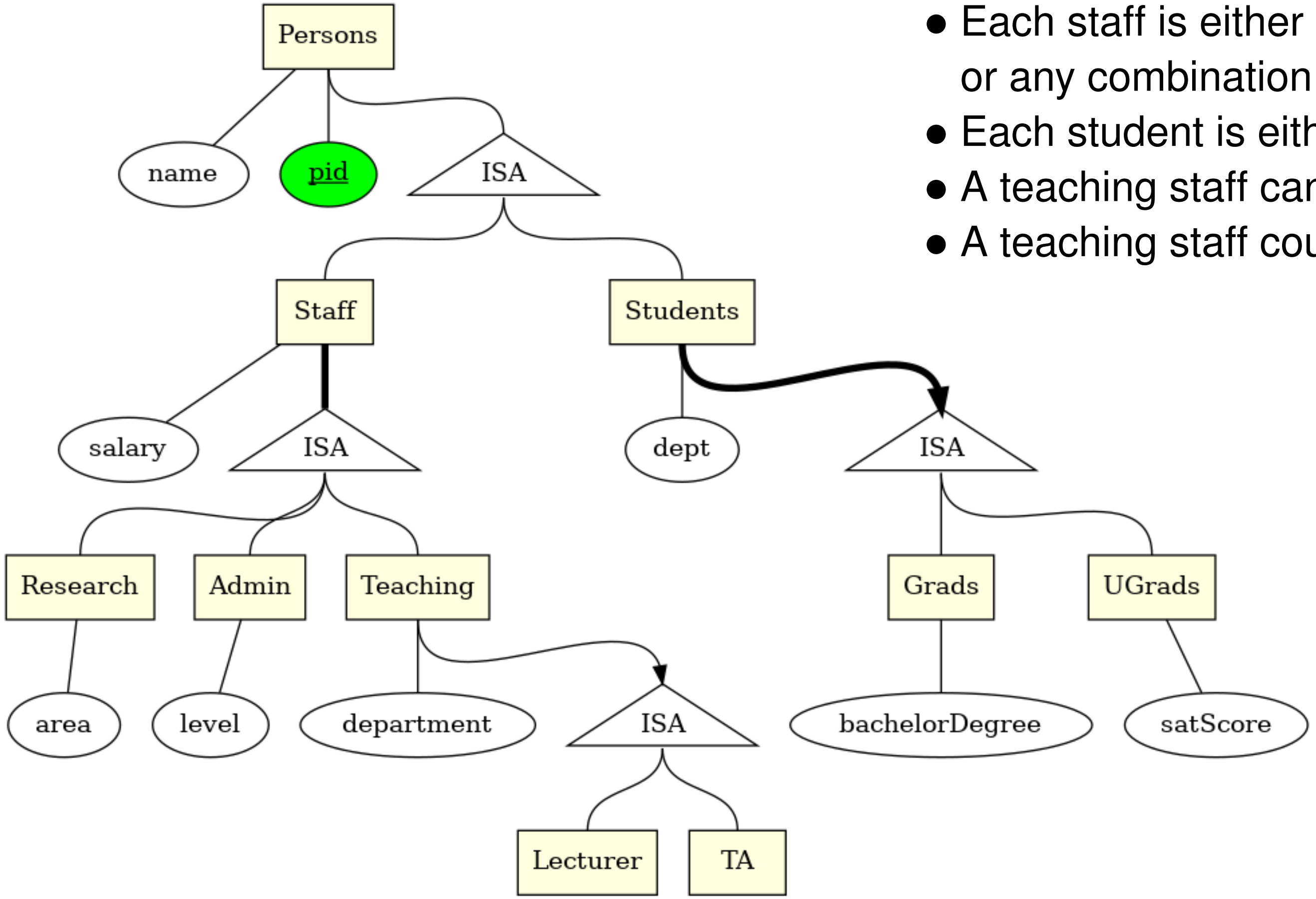
Constraints on ISA Hierarchies

- **Overlap constraint**: Can an entity belong to multiple subclasses?
    - A ISA hierarchy satisfies the overlap constraint if an entity in a superclass could belong to multiple subclasses
    - Satisfies: Undirected edge from superclass to ISA triangle
    - Does not satisfy: Directed edge from superclass to ISA triangle
- **Covering constraint** Does an entity in a superclass have to belong to some subclass?
    - A ISA hierarchy satisfies the covering constraint if every entity in a superclass has to belong to some subclass
    - Satisfies: thick edge from superclass to ISA triangle
    - Does not satisfy: thin edge from superclass to ISA triangle

ISA Hierarchies: Notation

Overlap Constraint	Covering Constraint	
	false	true
false		
true		

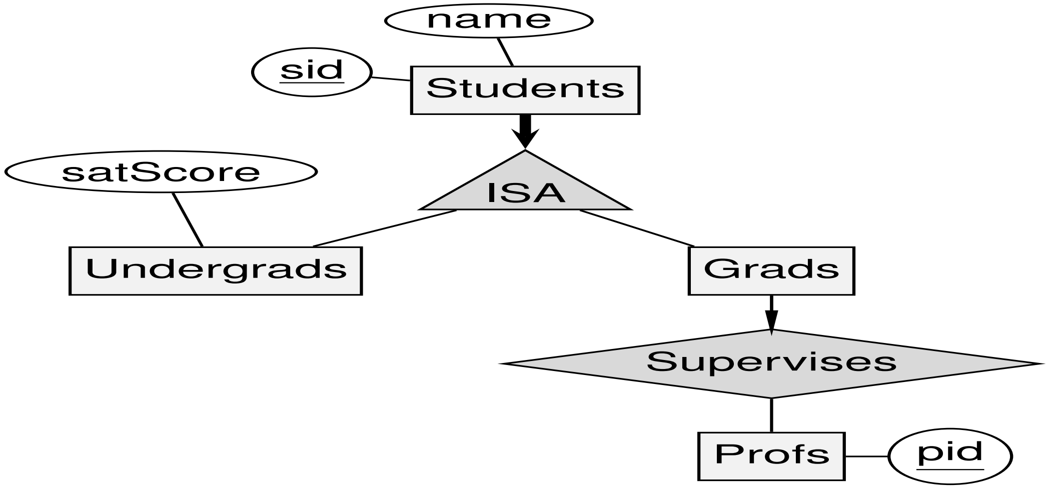
ISA Hierarchies: Example



- A person could be both a student & staff
- A person could be neither a student nor staff
- Each staff is either a research, teaching, admin, or any combination of these three roles
- Each student is either an undegraduate or a graduate student
- A teaching staff can’t be both a lecturer and TA
- A teaching staff could be neither a lecturer nor a TA

ISA Hierarchies: Relational Mapping

Simplest approach: One relation per subclass/superclass



```
create table Students (
  sid     char(20)
          primary key,
  name    char(30));
```

```
create table Undergrads (
  sid        char(20) primary key references Students
             on delete cascade,
  satScore   numeric);
```

```
create table Grads (
  sid         char(20) primary key references Students
              on delete cascade,
  supervisor  char(7) references Profs(pid));
```

Summary

- ER model has expressive constructs for conceptual data design
    - Concepts: entities, relationships, attributes, weak entities, aggregation, ISA hierarchies
    - Constraints: key constraints, participation constraints
- Rules for mapping entity-relationship model to relational model
    - Entity & relationship sets
    - Key constraints
    - Participation constraints
    - Relationship roles
    - Weak entity sets
    - Aggregation
    - ISA hierarchies

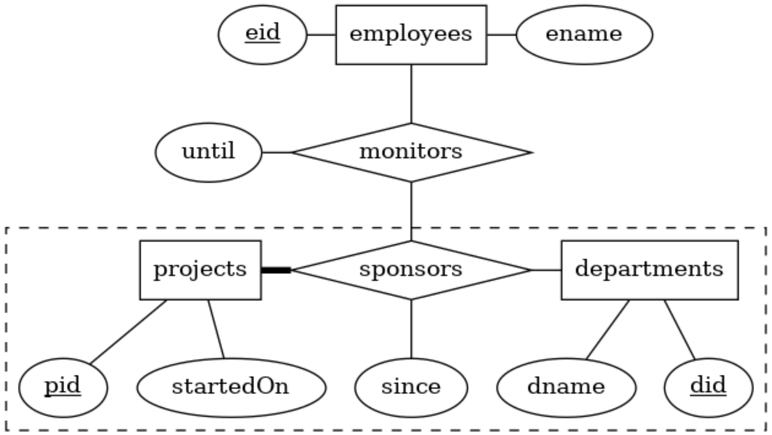
References

- R. Ramakrishnan & J. Gehrke, *Introduction to database design*, Database Management Systems, chapter 2. McGraw Hill, third edition, 2003.
- R. Ramakrishnan & J. Gehrke, *The relational model*, Database Management Systems, chapter 3. McGraw Hill, third edition, 2003.

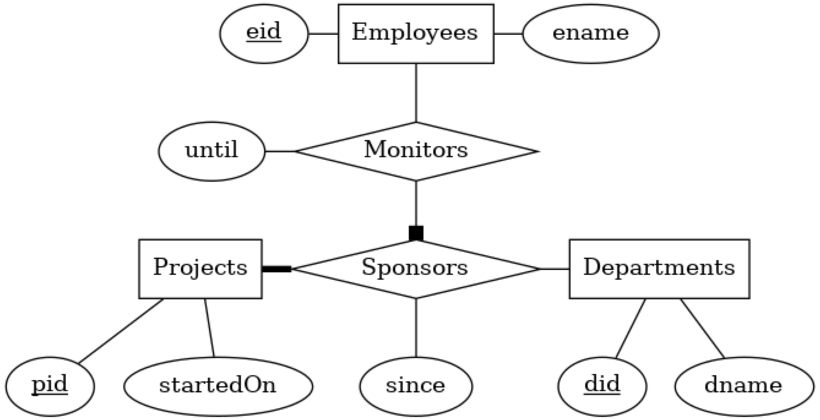
ER Model: R & G’s vs Lecture’s Notation

Lecture’s ER model uses different notation for aggregation

R & G’s Notation



Lecture’s Notation



Lecture’s ER model allows relationship attributes to form part of relationship key