CS 4604: Introduction to Database Management Systems

Entity/Relationship Models II

Virginia Tech CS 4604 Sprint 2021

Instructor: Yinlin Chen

Today's Topics

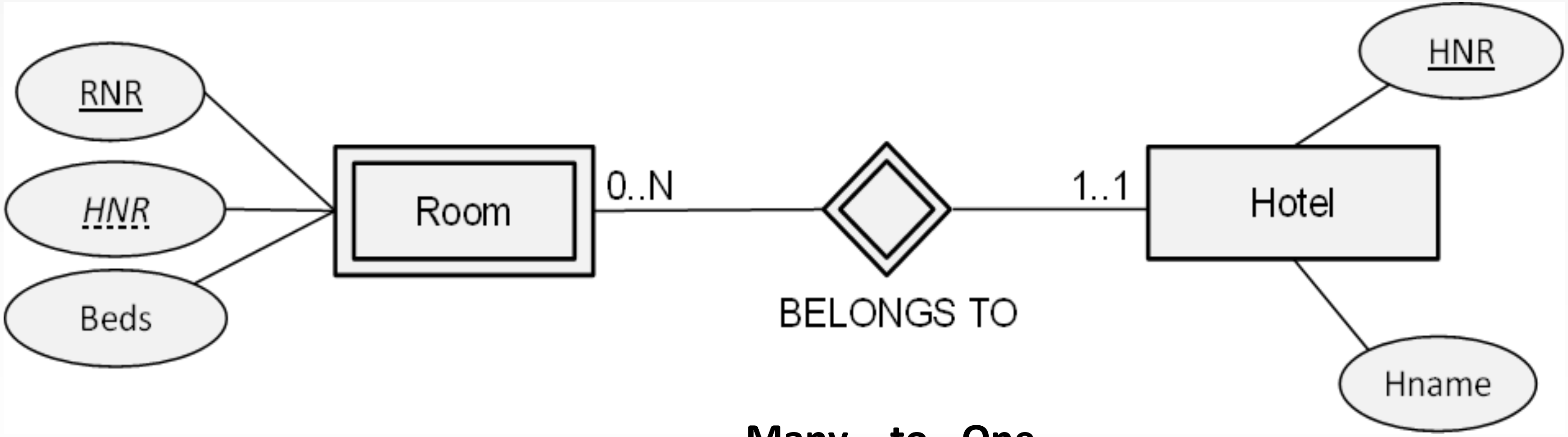
- Design ER models
- ER to Relational

Weak Entities

- A **strong entity** set is an entity set that has a **key** attribute type
- A **weak entity** set is an entity set whose key contains attributes from one or more other entity sets.
  - Owner entity set and weak entity set must participate in a **one-to-many** relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this identifying relationship set.
- Weak entities have a “partial key” (dashed underline)

Weak Entity Set

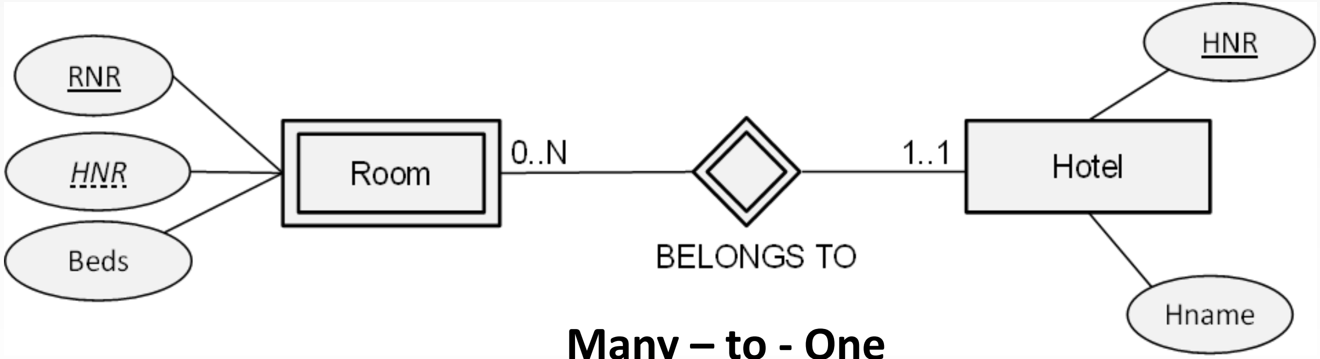
- **Weak entity** set is always existentially dependent from owner entity set (not vice versa!)
- Representation: a rectangle with a double border in the E/R diagram
- **Supporting relationship**: diamond with a double border



Many – to - One

Finding the Key for a Weak Entity Set

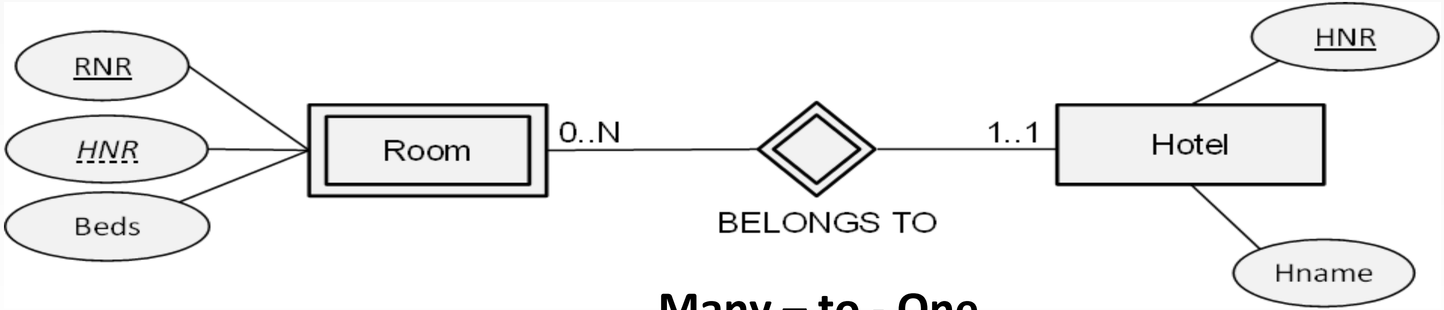
- Room is a weak entity set if its key consists of
  - Zero or more of its own attributes
  - Key attributes from supporting relationships for Room



Many – to - One

Supporting relationship

- A relationship R from a weak entity set Room to Hotel is supporting if
  - R is a binary, many-to-one relationship from Room to Hotel
  - R has referential integrity from Hotel to Room
- How does Hotel help Room?
  - Hotel supplies its key attributes to define Room's key
  - If Hotel is itself a weak entity set, some of its key attributes come from entity sets to which Hotel is connected by supporting relationships

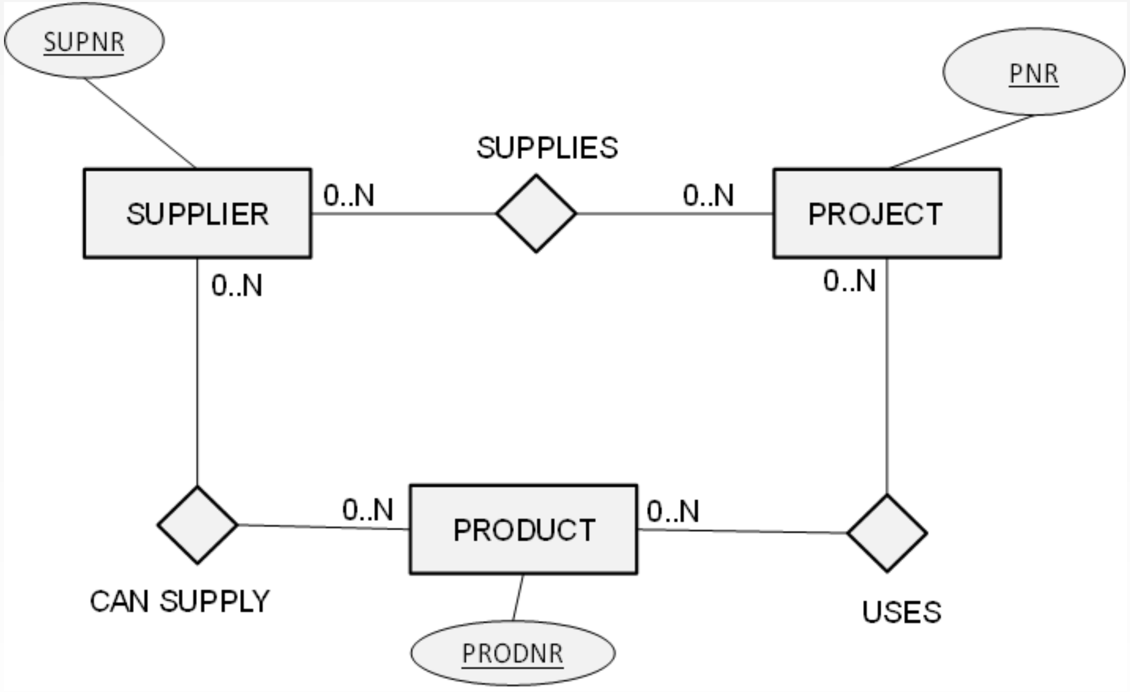


Many – to - One

Ternary Relationship Types Example

- Assume that we have a situation where suppliers can supply products for projects.
- A supplier can supply a particular product for multiple projects.
- A product for a particular project can be supplied by multiple suppliers.
- A project can have a particular supplier supply multiple products.
- The model must also include the quantity and due date for supplying a particular product to a particular project by a particular supplier.

Binary Relationship Types



Binary Relationship Types

- Say we have two projects: project 1 uses a pencil and a pen, and project 2 uses a pen
- Supplier Peters supplies the pencil for project 1 and the pen for project 2
- Supplier Johnson supplies the pen for project 1
- From the binary relationship types, it is not clear who supplies the pen for project 1!

SUPPLIES

Supplier	Project
Peters	Project 1
Peters	Project 2
Johnson	Project 1

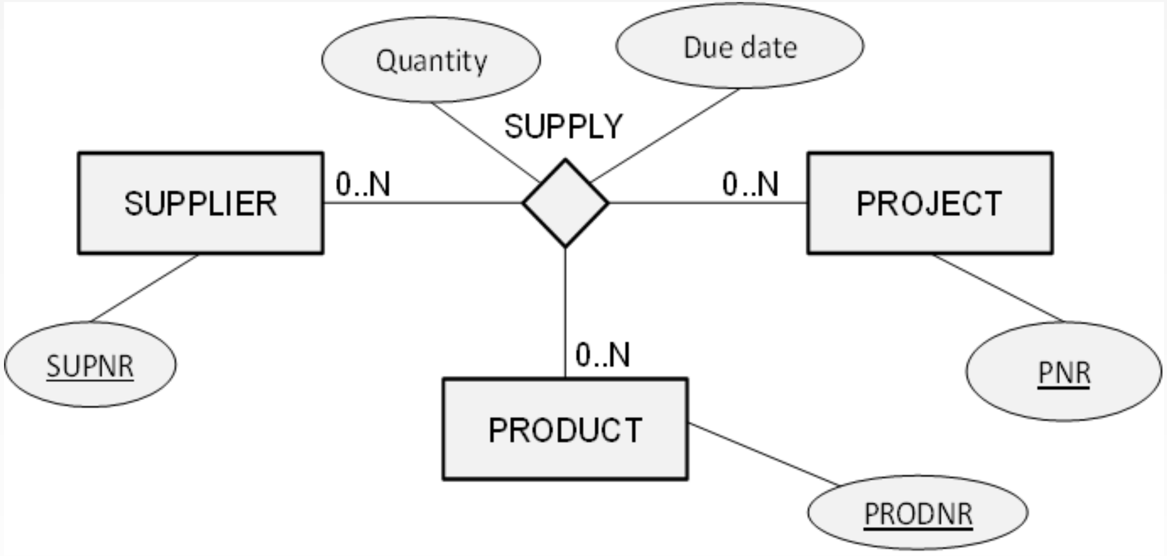
USES

Product	Project
Pencil	Project 1
Pen	Project 1
Pen	Project 2

CAN SUPPLY

Supplier	Product
Peters	Pencil
Peters	Pen
Johnson	Pen

Ternary Relationship Types

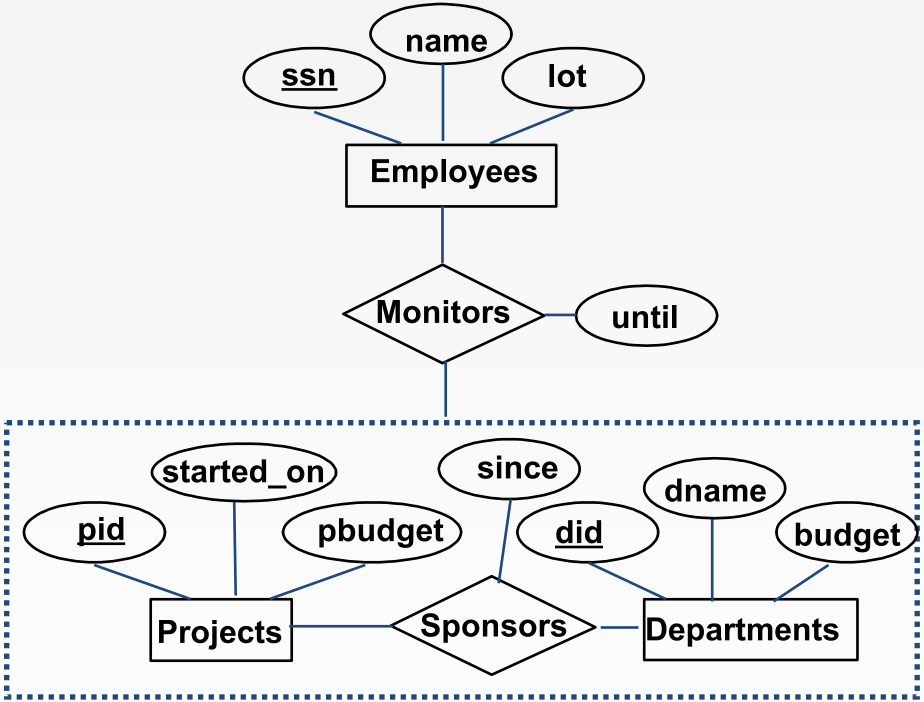


SUPPLY

Supplier	Product	Project
Peters	Pencil	Project 1
Peters	Pen	Project 2
Johnson	Pen	Project 1

Aggregation

- Allows relationships to have relationships
- Entity types that are related by a particular relationship type can be combined or aggregated into a higher-level aggregate entity type
- Aggregation is especially useful when the aggregate entity type has its own attribute types and/or relationship types



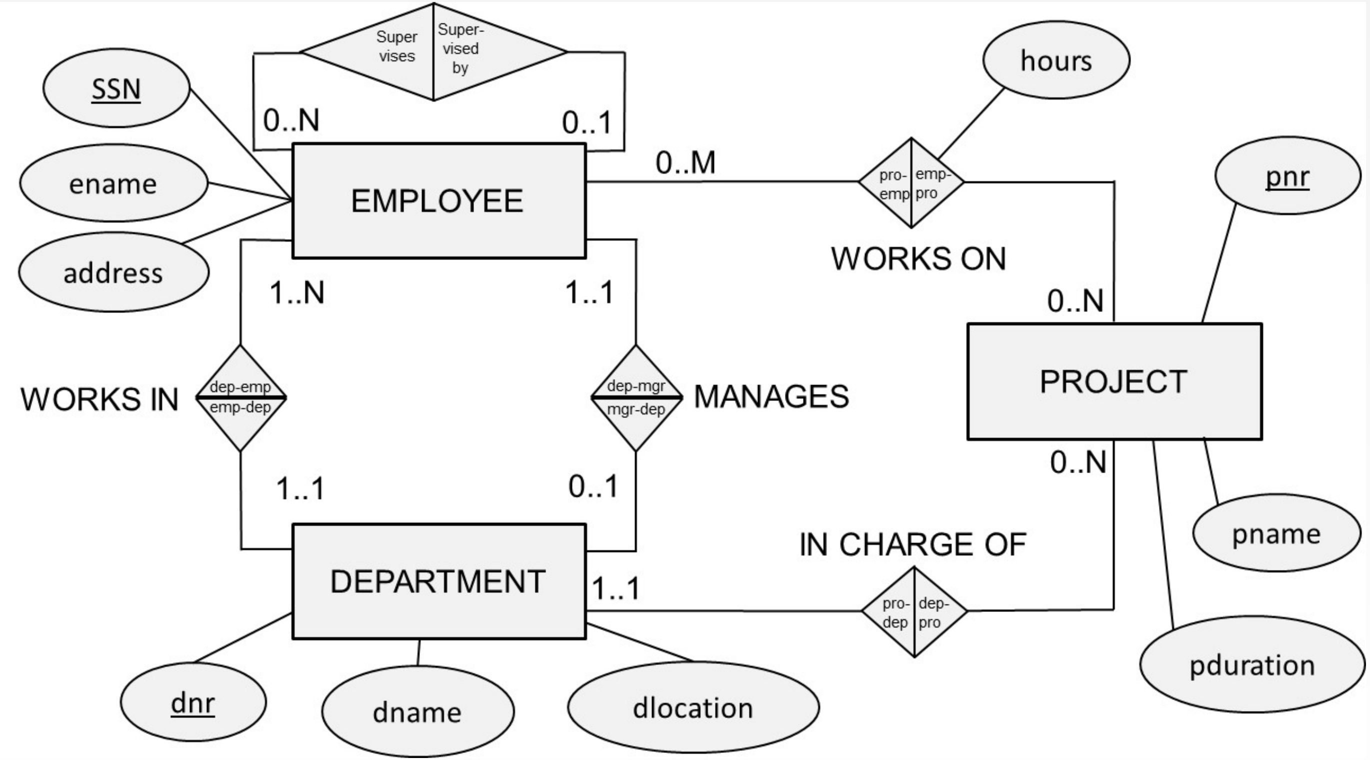
Limitations of the ER model

- ER model presents a temporary snapshot and cannot model **temporal** constraints
  - Examples: a project needs to be assigned to a department after one month, a purchase order must be assigned to a supplier after two weeks, etc.
- ER model cannot guarantee the consistency across multiple relationship types
  - Examples: an employee should work in the department that he/she manages, suppliers can only be assigned to purchase orders for products they can supply

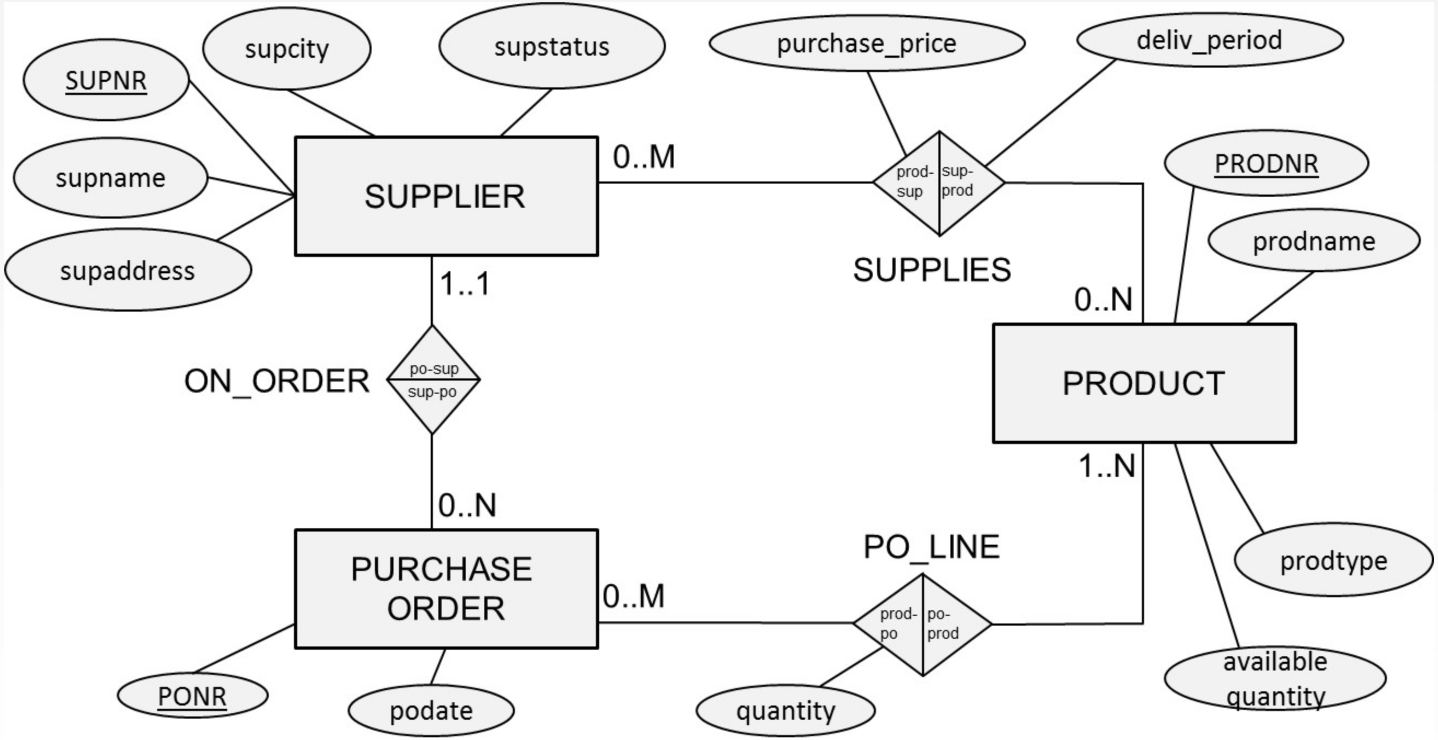
Limitations of the ER model

- Domains are not included in the ER model
  - Examples: hours should be positive; prodtype must be red, white or sparkling, supstatus is an integer between 0 and 100
- Functions are not included in the ER model
  - Examples: calculate average number of projects an employee works on; determine which supplier charges the maximum price for a product

Examples of the ER Diagram (Model)



Examples of the ER Diagram (Model)



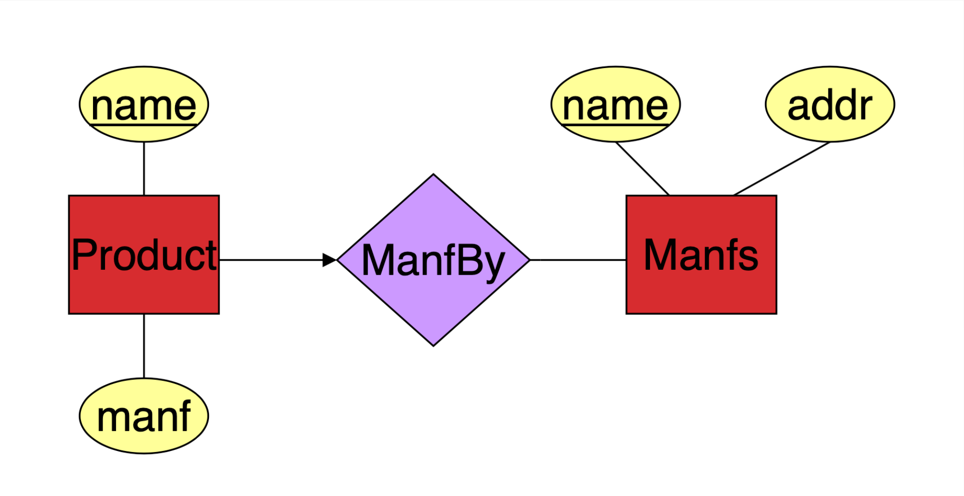
Conceptual Design Using the ER Model

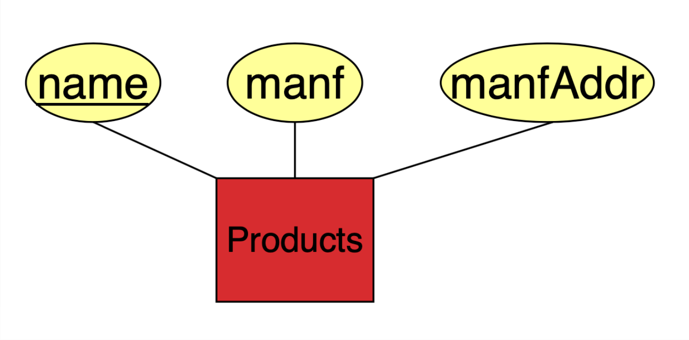
- ER modeling can get tricky!
- Design choices:
  - **Entity** or **attribute**?
  - **Entity** or **relationship**?
  - Relationships: **Binary or ternary? Aggregation?**
- ER Model goals and limitations:
  - Lots of semantics can (and should) be captured.
  - Some constraints cannot be captured in ER.
    - We will refine things in our logical (relational) design

Principle 1: Avoiding Redundancy

- Redundancy occurs when we say the same thing in two different ways
- Redundancy wastes space and causes inconsistency
  - The two instances of the same fact may become inconsistent if we change one & forget to change the other

Example: Avoiding Redundancy





Principle 2: Entity vs. Attribute

- "Address":
  - attribute of Employees?
  - Entity of its own?
- It depends! Semantics and usage.
  - Several addresses per employee?
    - must be an entity
    - atomic attribute types (no set-valued attributes!)
  - Care about structure? (city, street, etc.)
    - must be an entity! (or at least multiple attributes)
    - atomic attribute types (no tuple-valued attributes!)

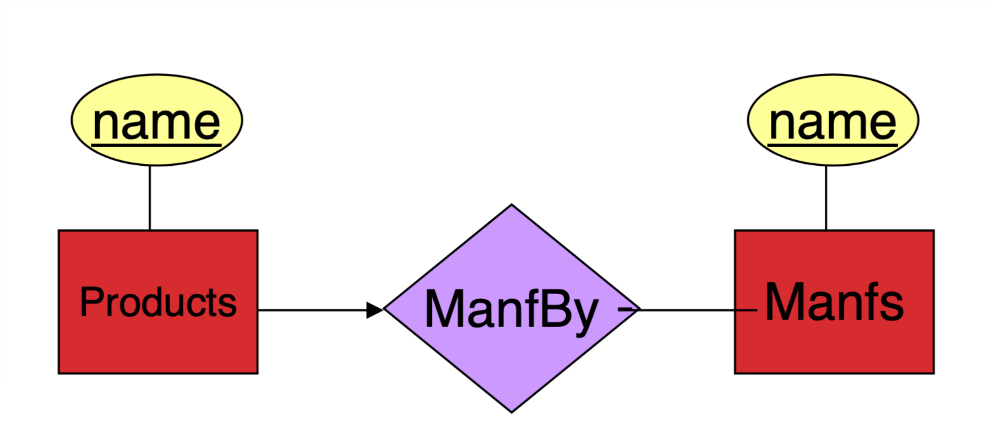
Entity Sets Versus Attributes

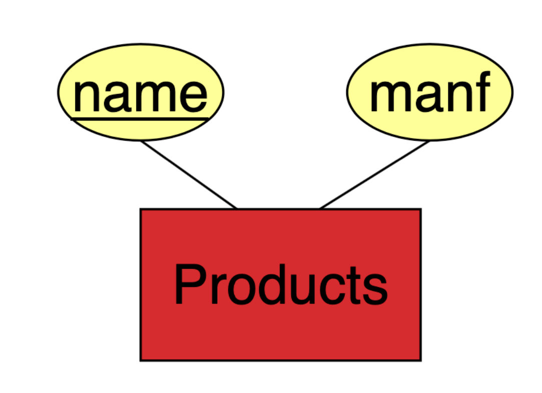
- Rule: An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something;
    - i.e., it has at least one non-key attribute.

    Or
  - It is the “many” in a many-one or many-many relationship.

Example: Entity vs. Attribute

- If we had no manufacturer address information





Summary of Conceptual Design

- **Conceptual design** follows requirements analysis
  - Yields a high-level description of data to be stored
- **ER model** popular for conceptual design
  - Constructs are expressive, close to the way we think about applications.
  - Note: There are many variations on ER model
    - Both graphically and conceptually
- Basic constructs: **entities**, **relationships**, and **attributes** (of entities and relationships).
- Some additional constructs**: weak entities** and aggregation.

Summary of ER (Cont.)

- Basic integrity constraints
  - **key constraints**
  - **participation constraints**

- Some **foreign key** constraints are also implicit in the definition of a relationship set.

- Many other constraints (notably, **functional dependencies**) cannot be expressed.

- Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Cont.)

- ER design is **subjective**. Many ways to model a given scenario!
- Analyzing alternatives can be tricky! Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use aggregation
- For good DB design: resulting relational schema should be **analyzed** and **refined** further.
  - Functional Dependency information
    + normalization coming in subsequent lecture.

Guidelines

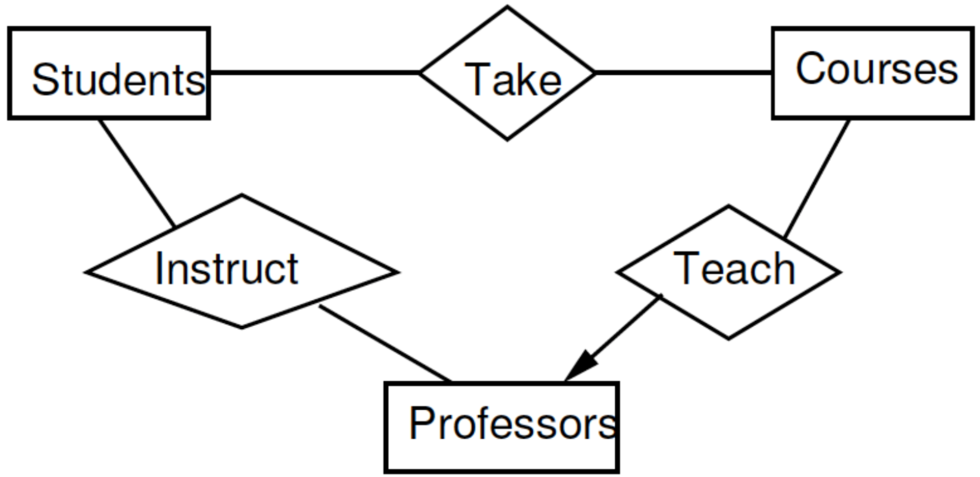
- Be faithful to the specification of the application
- Avoid redundancy
- Keep the entities and relationship **simple**
- Select the right relationships
- Select the right type of element

Be Faithful to the Specification

- Do not use meaningless or unnecessary attributes
- Define the multiplicity of a relationship appropriately
  - What is the multiplicity of the relationship Take between Students and Courses?
  - What is the multiplicity of the relationship Teach between Professors and Courses?

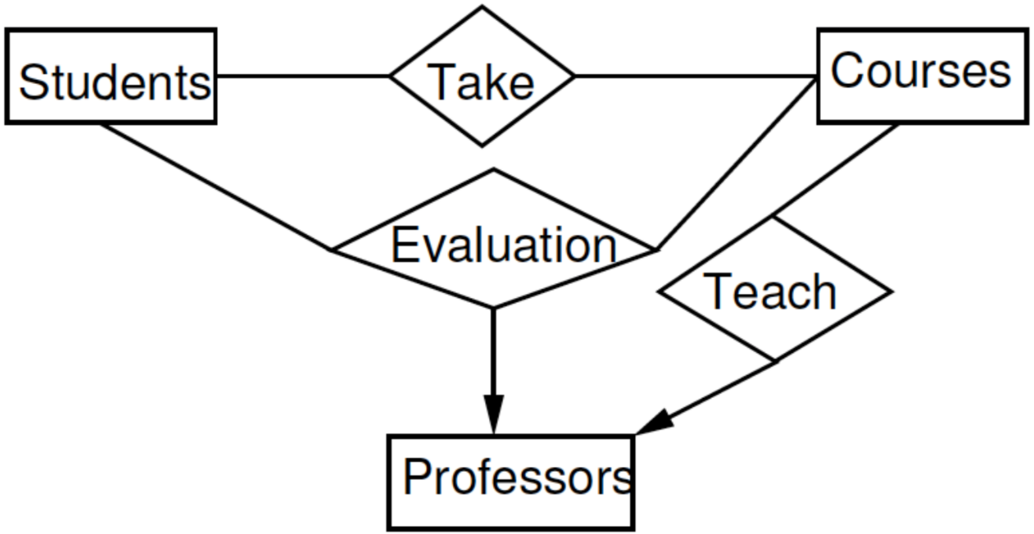
Select the Right Relationships

- Do not add unnecessary relationships
- It may be possible to deduce one relationship from another
- Do we need the relationship Instruct between Professors and Students?



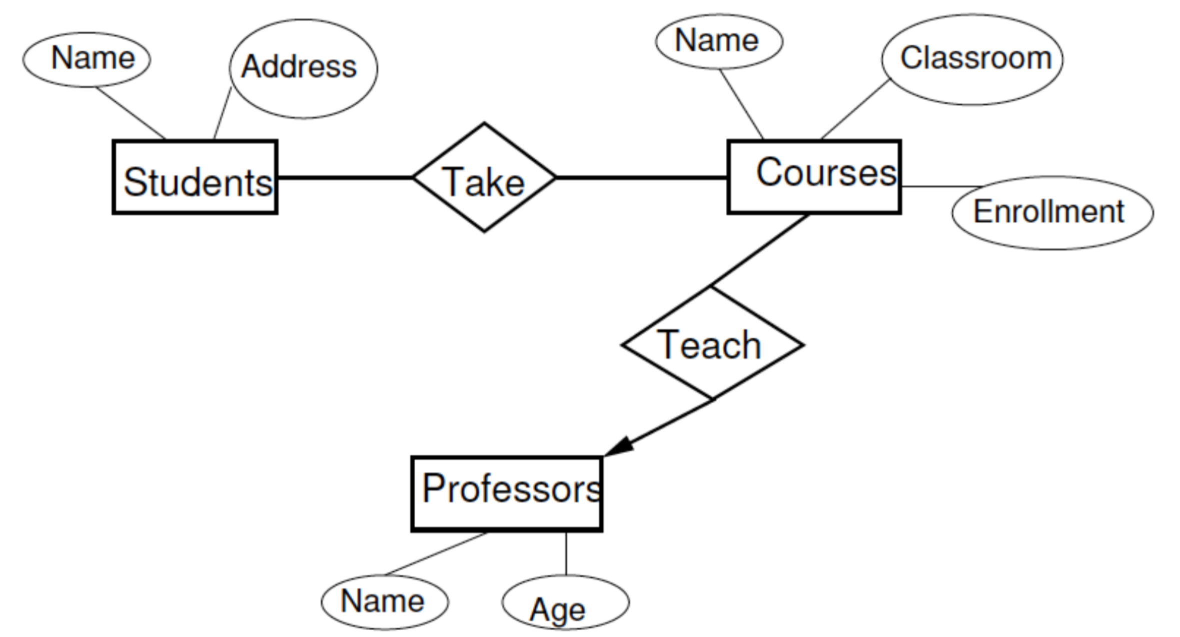
Select the Right Relationships

- Do we need the relationships Take and Teach?



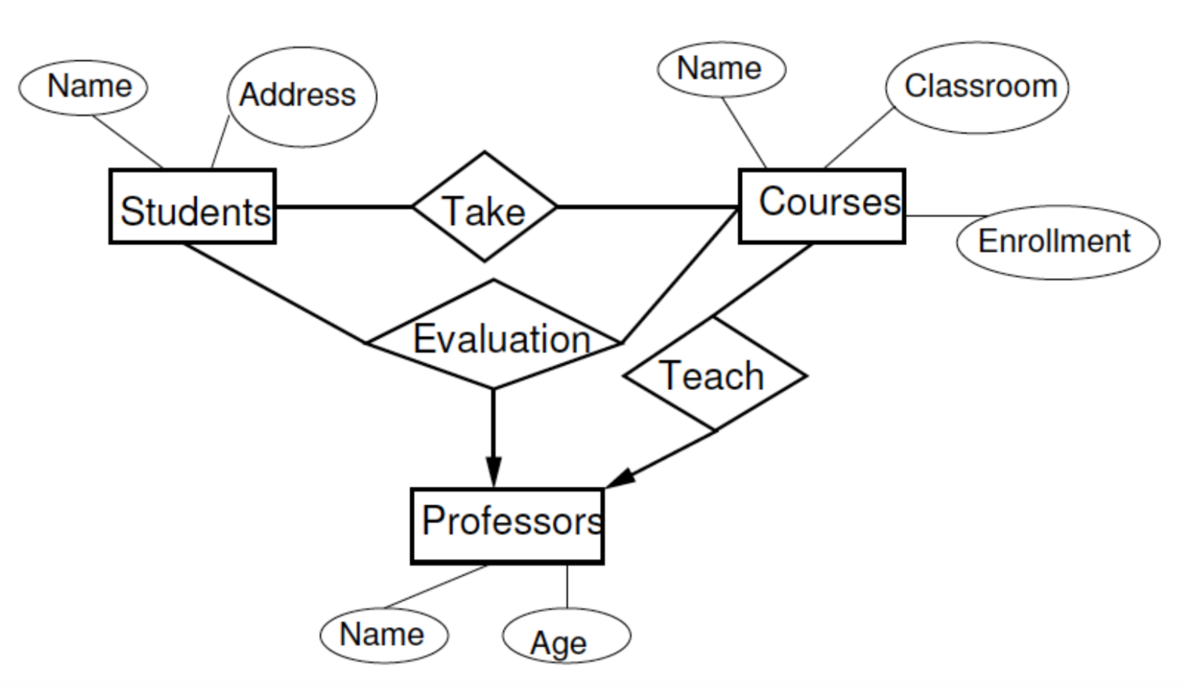
Select the right kind of element

- Attribute or Entity or Relationship
- Can we make Professor an attribute of Courses and remove the relationship Teach?



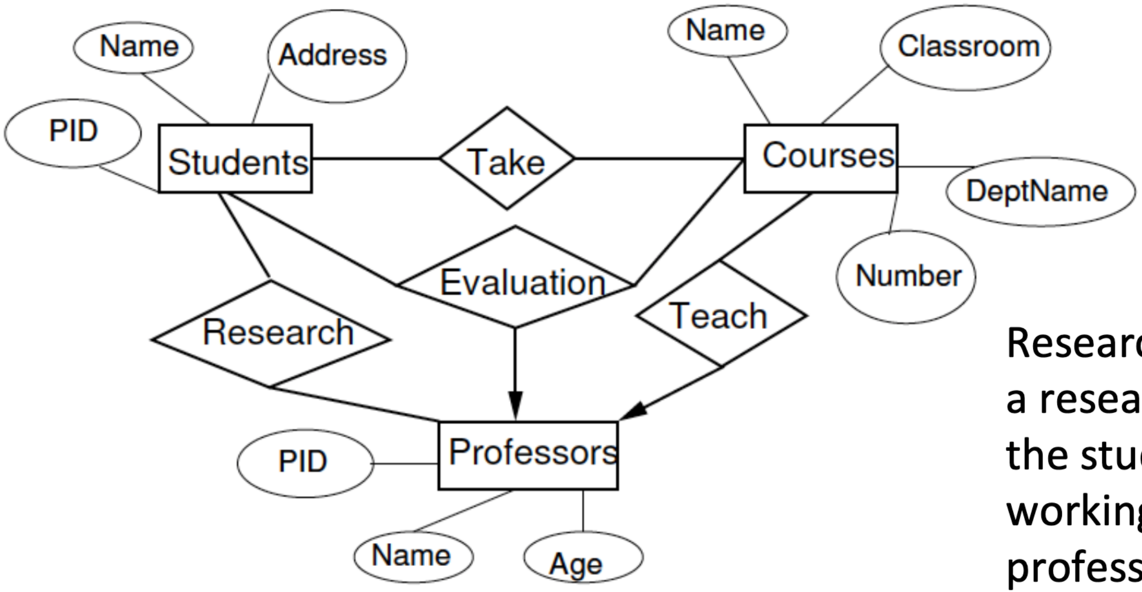
Select the right kind of element

- Attribute or Entity or Relationship
- What about this?



Select the right kind of element

- Attribute or Entity or Relationship
- What about this?



Research signifies a research project the student is working on with a professor

Converting an Entity Set into an Attribute

- **If** an entity set E satisfies the following properties:
  - All relationships involving E have arrows entering E
  - The attributes of E collectively identify an entity (i.e., no attribute depends on another)
  - No relationship involves E more than once
- **Then** we can replace E as follows:
  - If there is a many-one relationship R from an entity set F to E, remove R and make the attributes of E be attributes of F
  - If there is a multiway relationship R with an arrow to E, make E's attributes be new attributes of R and remove the arrow from R to E

Recap: Types of Constraints

- **Keys** are attributes or sets of attributes that uniquely identify an entity within its entity set
- **Single-value constraints** require that a value be unique in certain contexts
- **Referential integrity constraints** require that a value referred to actually exists in the database
- **Degree constraints** specify what set of values an attribute can take
- **General constraints** are arbitrary constraints that should hold in the database
- **Constraints are part of the schema of a database**

Single Value Constraints

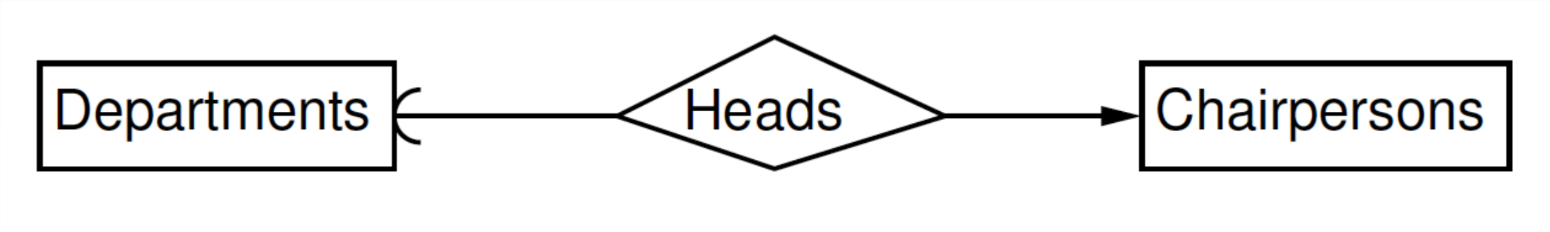
- There is at most one value in a given context
- Each attribute of an entity set has a single value
  - If the value is missing, we can invent a “null" value
  - E/R models cannot represent the requirement that an attribute cannot have a null value
- A many-one relationship **implies** a single value constraint

Referential Integrity Constraint

- **Asserts** that exactly one value exists in a given context
  - Usually used in the context of relationships
- Example: Many-one Advises relationship between Students and Professors
  - Many-one requirement says that no student may have more than one advising professor
  - Referential integrity constraint says that each student must have exactly one advising professor and that professor **must be present** in the database
- If R is a (many-to-one or one-to-one) relationship from E to F, we use a **rounded arrowhead** pointing to F to indicate that we require that the entity in F related by R to an entity in E **must exist**

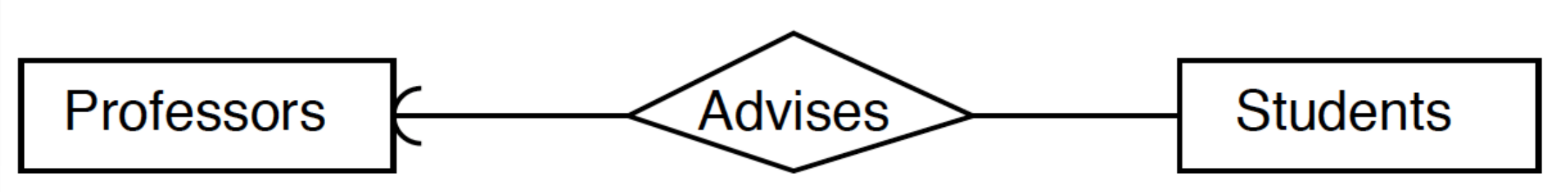
Example: Referential Integrity Constraint

- Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson)
- Each chairperson can be the head of at most one department and this department must exist in the database



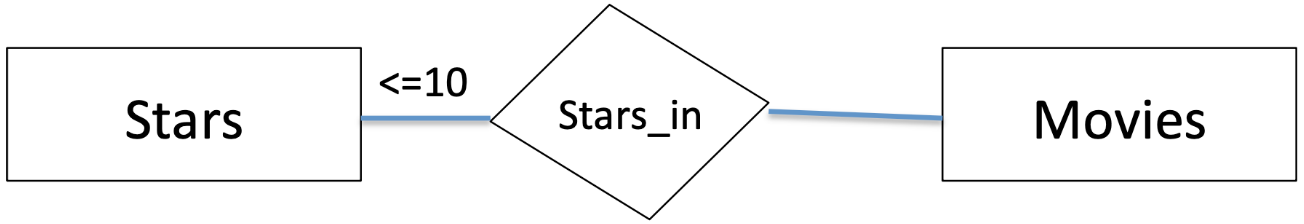
Enforcing Referential Integrity Constraints

- We forbid the deletion of a referenced entity (e.g., a professor) until the professor advises no students
- We require that if we delete a referenced entity, we delete all entities that reference it
- When we insert a (student, professor) pair into the Advises relationship, the professor must exist in the Professors entity set



Degree Constraints

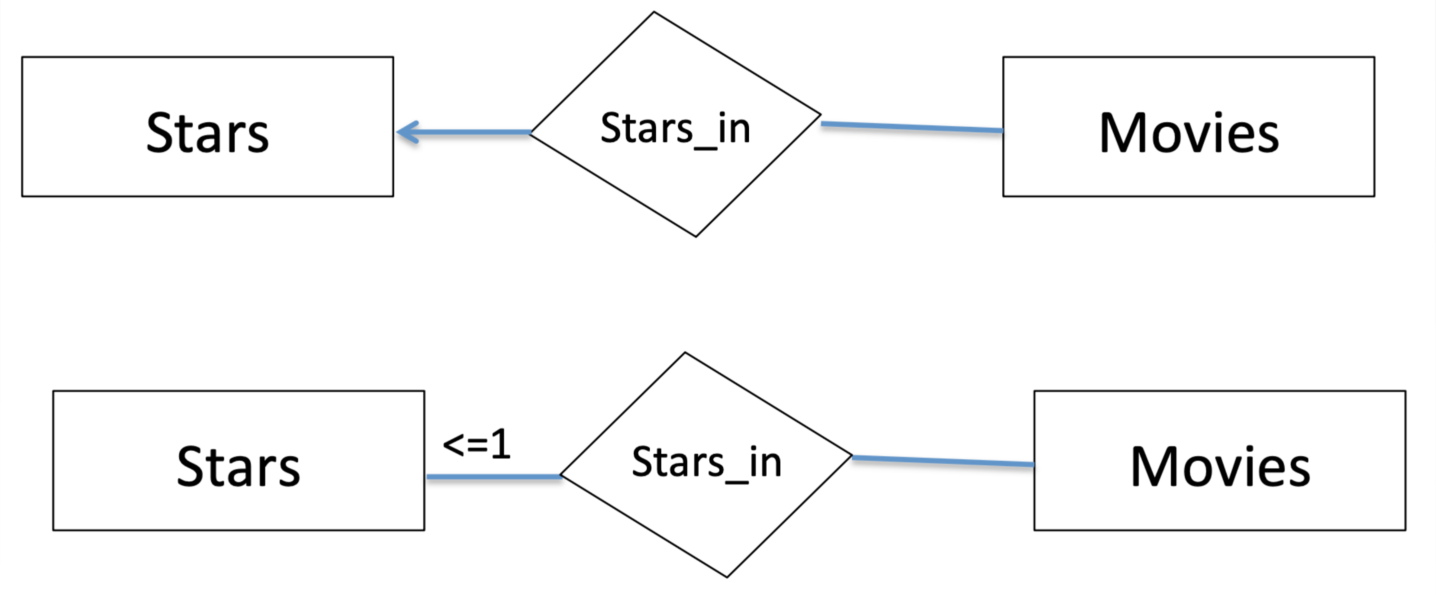
- Indicates limits on the # of entities that can be connected
- For example,



- Limits number of stars in each move to <=10

Degree Constraints

- Indicates limits on the # of entities that can be connected



Steps in Database Design

- **Requirements Analysis**
  - user needs; what must database do?
- **Conceptual Design** ⟸ Just finished
  - *high level description (often done w/ER model)*
  - Object-Relational Mappings (ORMs: Hibernate, Rails, Django, etc.) encourage you to program here
- **Logical Design**
  - translate ER into DBMS data model
  - ORMs often require you to help here too

  ⟸ We are here
- **Schema Refinement**
  - consistency, normalization
- **Physical Design** - indexes, disk layout
- **Security Design** - who accesses what, and how

Recap: Relational Model

- Built around a single concept for modelling data: the relation or table
- Supports high-level programming language (SQL)
- Has an elegant mathematical design theory
- Most current DBMS are relational

Recap: The Relation

- A relation is a two-dimensional table:
  - **Relation ⇔ Table**
  - **Attribute ⇔ Column name**
  - **Tuple ⇔ Row (not the header row)**
  - **Database ⇔ Collection of relations**

CoursesTaken

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

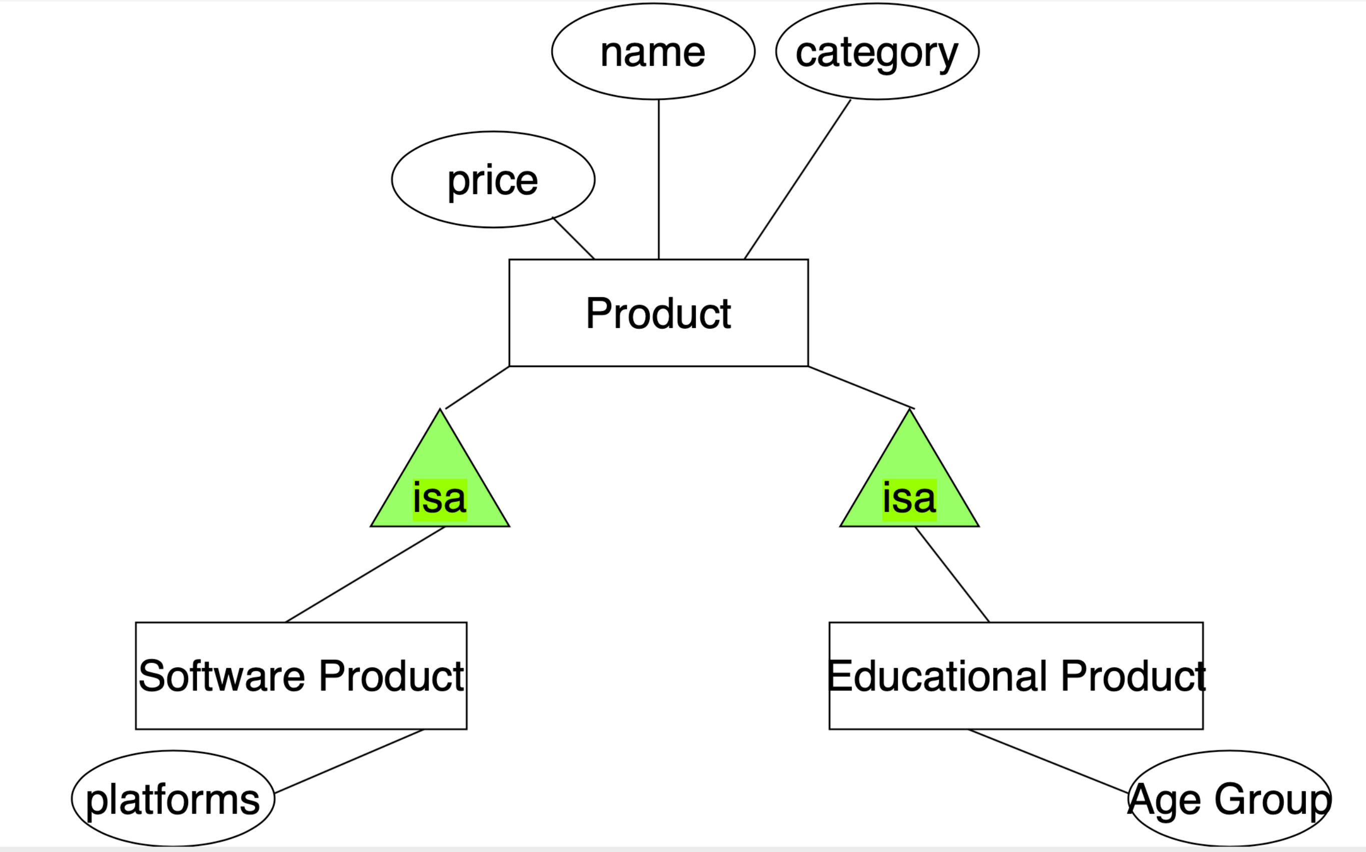
Recap: The Schema

- The schema of a relation is the name of the relation followed by a parenthesized list of attributes
  - CoursesTaken(Student, Course, Grade)
- A design in a relational model consists of a set of schemas.
  - Such a set of schemas is called a relational database schema

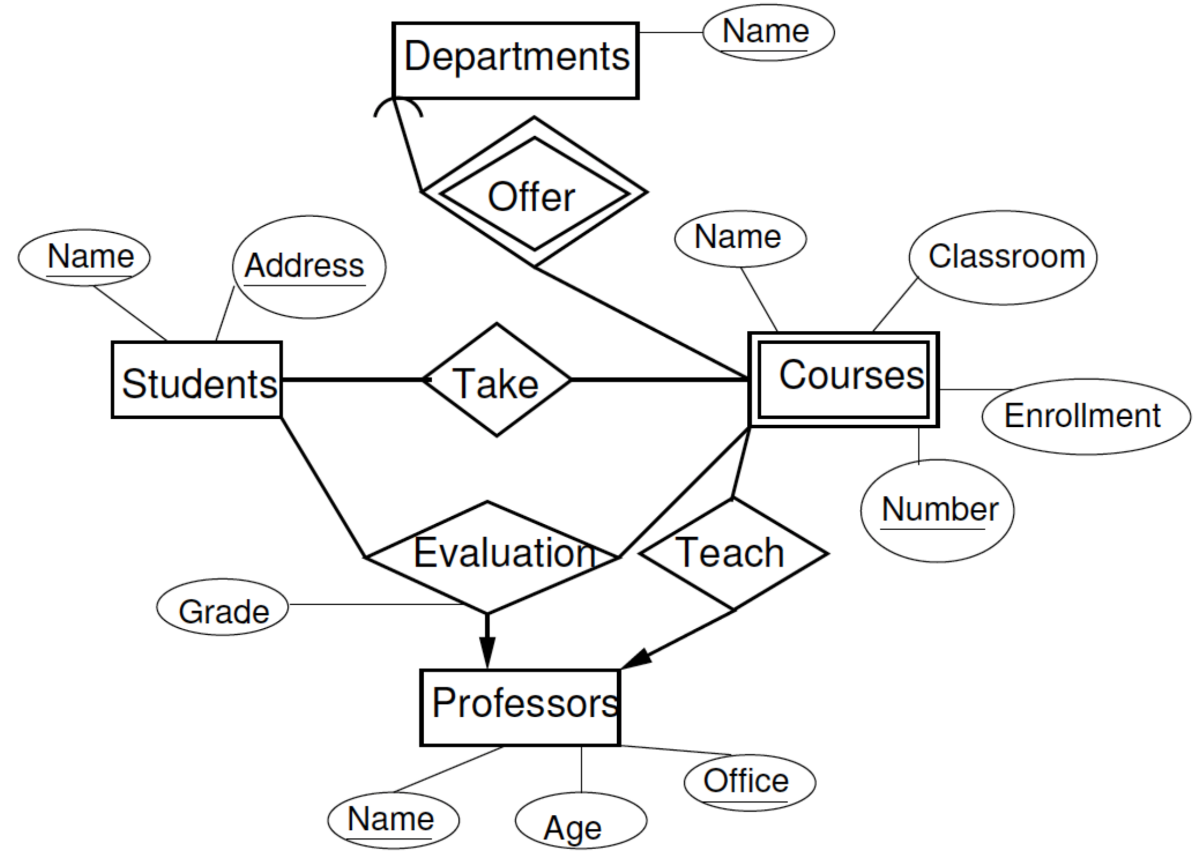
Converting ER Diagram to Relational Design

- Entity Set → Relation
  - Attribute of Entity Set → Attribute of a Relation
- Relationship → relation whose attributes are
  - Attribute of the relationship itself
  - Key attributes of the connected entity sets
- Several special cases:
  - Weak entity sets.
  - Combining relations (especially for many-one relationships)
  - *ISA* relationships and subclasses

Subclasses in ER Diagrams



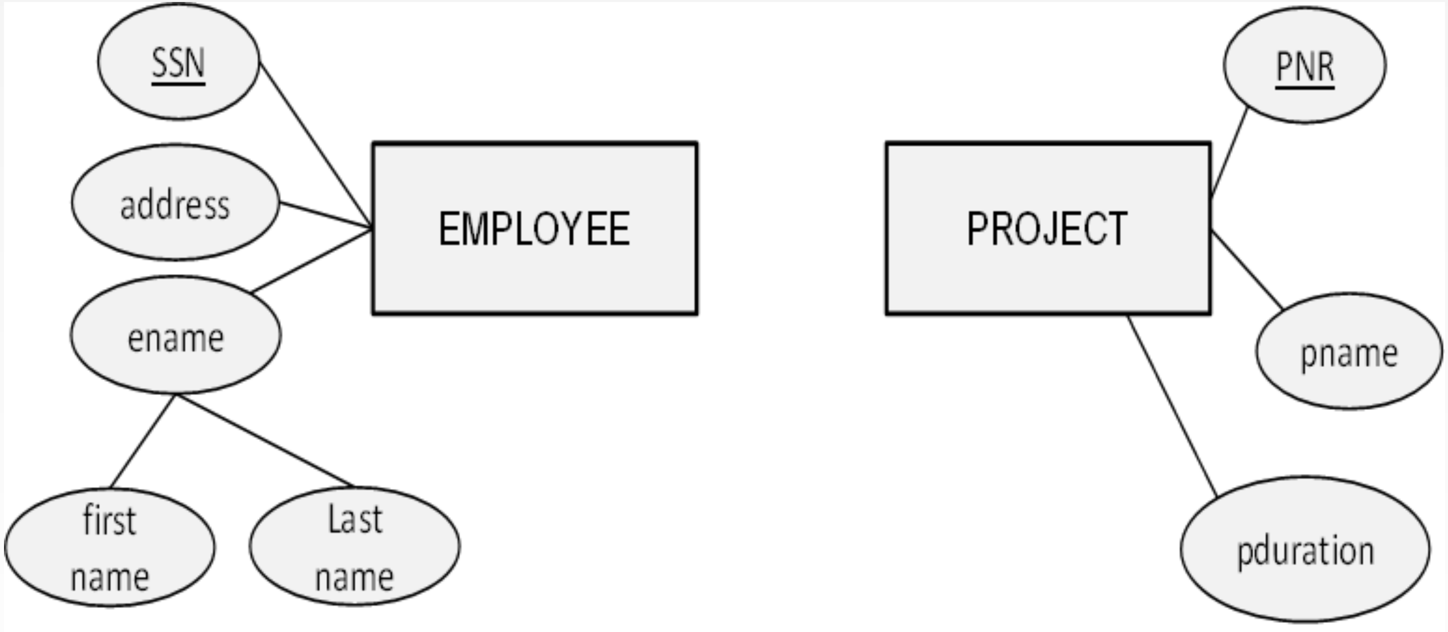
Example: ER Diagram



Schemas for Entity Sets

- For each entity set, create a relation with the same name and with the same set of attributes
- Students (Name, Address)
- Professors (Name, Office, Age)
- Departments (Name)

Mapping Entity Types

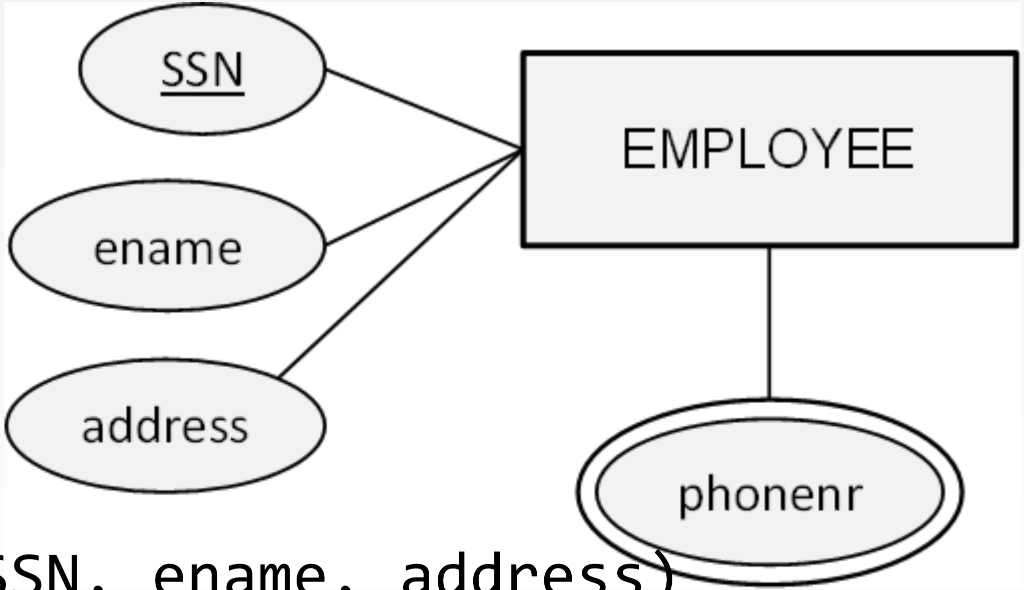


EMPLOYEE(SSN, address, first name, last name)
PROJECT(PNR, pname, pduration)

Mapping Multi-Valued Attribute Types

- For each multi-valued attribute type, we create a new relation R
- We put the multi-valued attribute type in R together with a foreign key referring to the primary key of the original relation
- Multi-valued composite attribute types are again decomposed into their components
- The primary key can then be set based upon the assumptions

Mapping Multi-Valued Attribute Types



EMPLOYEE(<u>SSN</u>, ename, address)

EMP-PHONE(<u>PhoneNr</u>, <u>*SSN*</u>)

- SSN foreign key refers to SSN in EMPLOYEE, NULL NOT ALLOWED

Mapping Multi-Valued Attribute Types

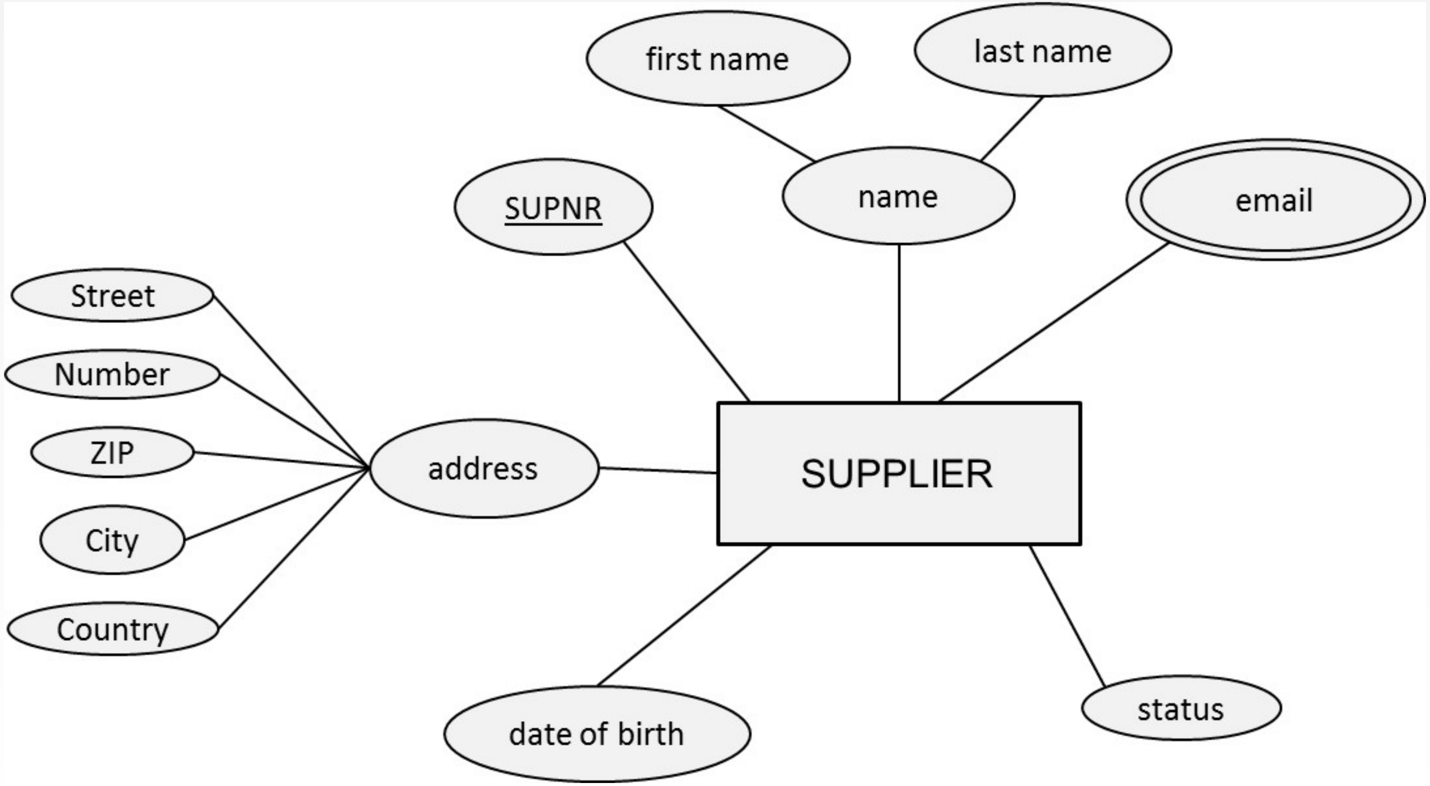
EMPLOYEE(<u>SSN</u>, ename, address, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	001
356	Emma Lucas	432 Wacker Drive, Chicago	002

EMP-PHONE(<u>PhoneNR</u>, <u>*SSN*</u>)

900-244-8000	511
900-244-8000	289
900-244-8002	289
900-246-6006	356

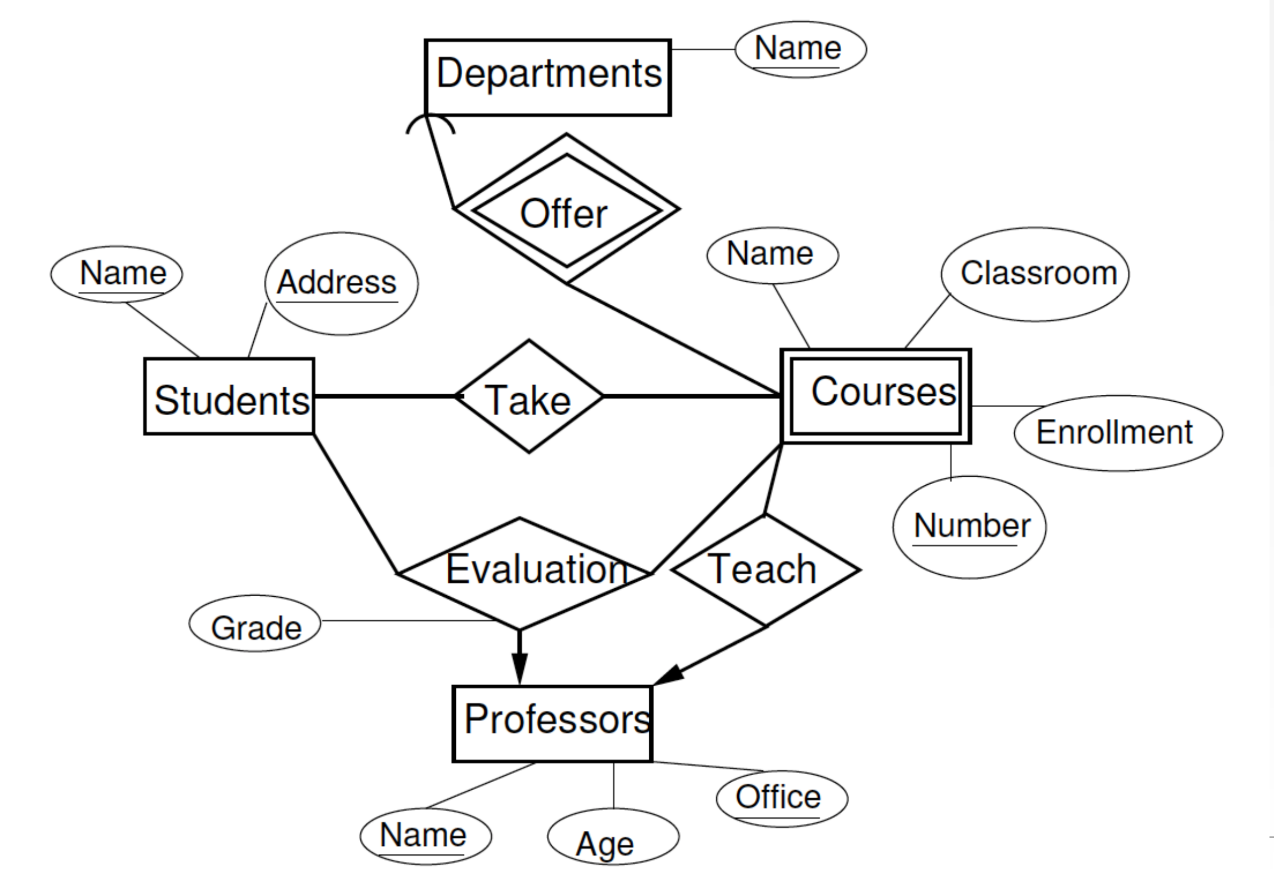
Composite Attributes Types



Schemas for Relationships

- For each relationship, create a relation with the same name whose attributes are
  - Attributes of the relationship itself
  - Key attributes of the connected entity sets (even if they are weak)
- Take (StudentName, Address, Number, DepartmentName)
- Teach (ProfessorName, Office, Number, DepartmentName)
- Evaluation (StudentName, Address, ProfessorName, Office, Number, DepartmentName, Grade)

Example: ER Diagram



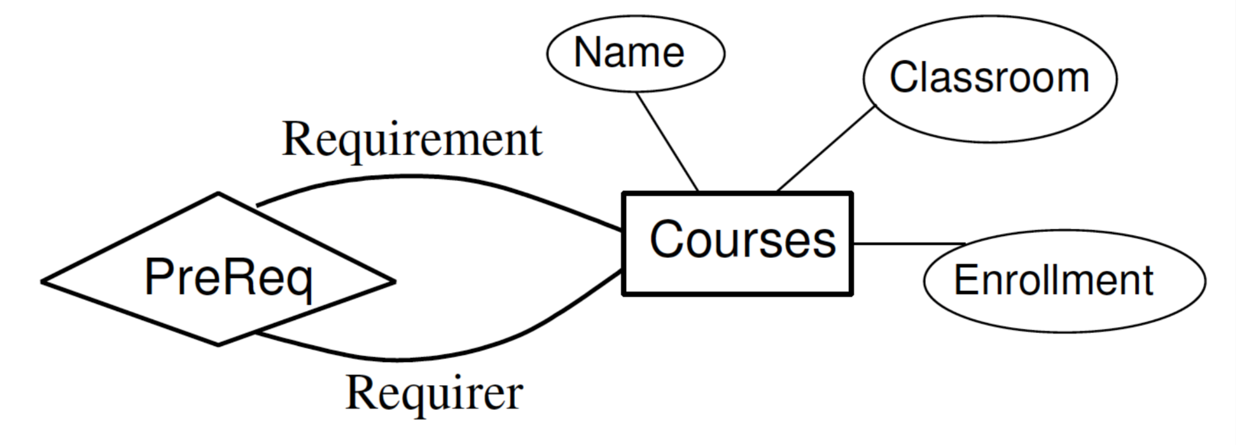
Combining Relations

- Consider many-one Teach relationship from Courses to Professors
- Schemas are:
  - Courses(Number, DepartmentName, CourseName, Classroom, Enrollment)
  - Professors(Name, Office, Age)
  - Teach(Number, DepartmentName, ProfessorName, Office)
- The key for Courses uniquely determines all attributes of Teach
- We can combine the relations for Courses and Teach into a single relation whose attributes are
  - All the attributes for Courses
  - Any attributes of Teach
  - The key attributes of Professors

Rules for Combining Relations

- We can combine into one new relation Q
  - The relation for an entity set E
  - all many-to-one relationships R1, R2, ..., Rk from E to other entity sets E1, E2, ..., Ek respectively
- The attributes of Q are
  - All the attributes of E
  - Any attributes of R1, R2, ..., Rk
  - The key attributes of E1, E2, ..., Ek

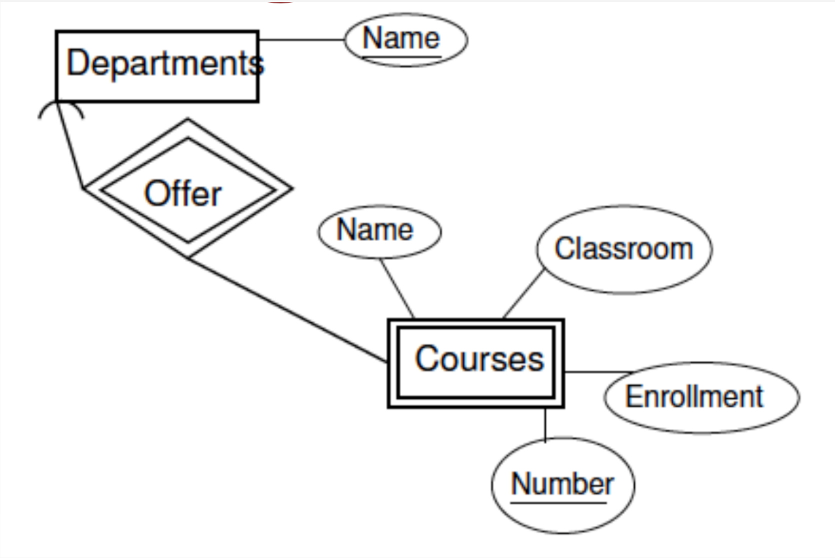
Roles in Relationships



- If an entity set E appears k > 1 times in a relationship R (in different roles), the key attributes for E appear k times in the relation for R, appropriately renamed
- PreReq (RequirerNumber, RequirerDeptName, RequirementNumber, RequirementDeptName)

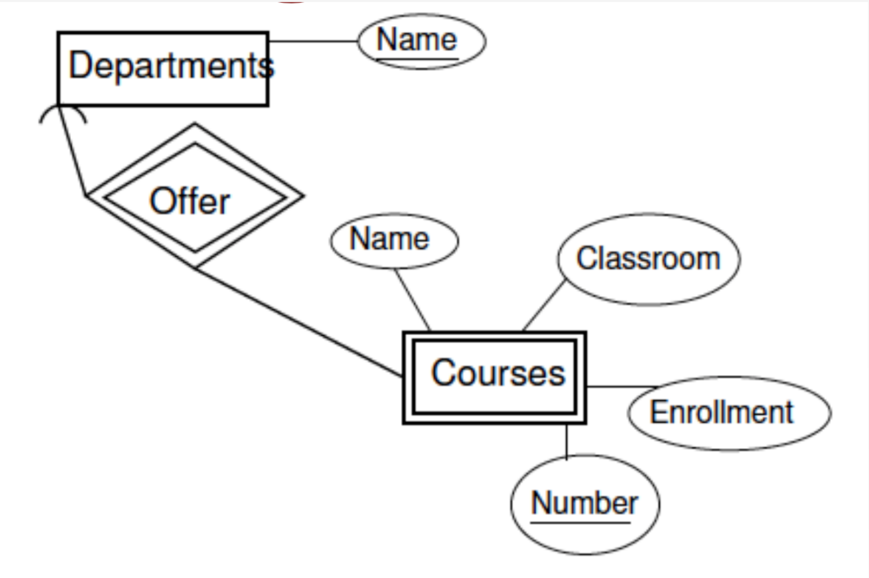
Supporting Relationships

- **Departments**(Name)
- **Courses**(Number, DepartmentName, CourseName, Classroom, Enrollment)
- **Offer**(Name, Number, DepartmentName)
  - But Name and DepartmentName are identical, so the schema for Offer is **Offer**(**Number, DepartmentName**)
  - The schema for Offer is a subset of the schema for the weak entity set, so we can dispense with the relation for Offer



Rules for Supporting Relationships

- If W is a weak entity set, the relation for W has a schema whose attributes are
  - all attributes of W
  - all attributes of supporting relationships for W
  - for each supporting relationship for W to an entity set E
  - the key attributes of E
- There is no relation for any supporting relationship for W



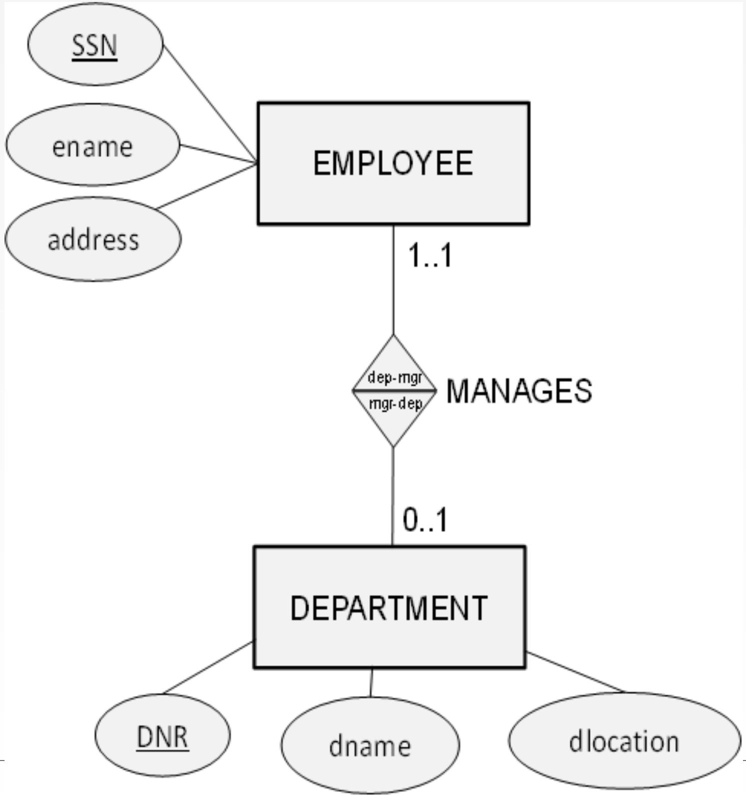
Mapping Relationship Types

- Mapping a binary 1:1 relationship type
- Mapping a binary 1:N relationship type
- Mapping a binary N:M relationship type
- Mapping unary relationship types
- Mapping n-ary relationship types

Mapping a Binary 1:1 Relationship Type

- Create two relations: one for each entity type participating in the relationship type
- The connection can be made by including a foreign key in one of the relations to the primary key of the other
- In case of existence dependency (participation constraint), put the foreign key in the existence-dependent relation and declare it as NOT NULL
- The attribute types of the 1:1 relationship type can then be added to the relation with the foreign key

Mapping a Binary 1:1 Relationship Type



Mapping a Binary 1:1 Relationship Type

EMPLOYEE(SSN, ename, address)

DEPARTMENT(DNR, dname, dlocation, *SSN*)

- SSN foreign key refers to SSN in EMPLOYEE, NULL NOT ALLOWED
- SSN is an alternate key

EMPLOYEE(SSN, ename, address)

511	John Smith	14 Avenue of the Americas, New York
289	Paul Barker	208 Market Street, San Francisco
356	Emma Lucas	432 Wacker Drive, Chicago

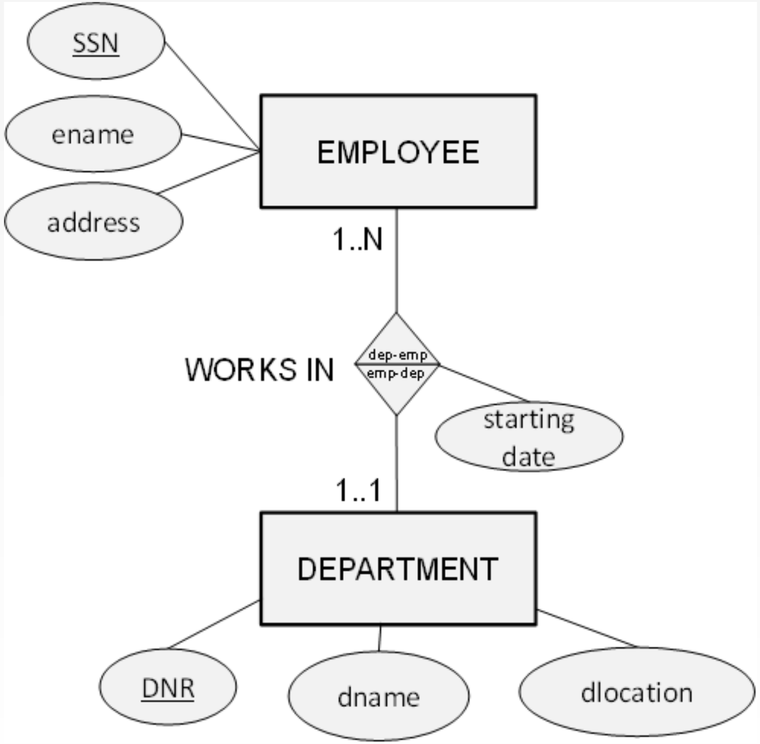
DEPARTMENT(DNR, dname, dlocation, SSN)

001	Marketing	3th floor	511
002	Call center	2nd floor	511
003	Finance	basement	289
004	ICT	1st floor	511

Mapping a Binary 1:N Relationship Type

- Binary 1:N relationship types can be mapped by including a foreign key in the relation corresponding to the participating entity type at the N-side of the relationship type
- The foreign key refers to the primary key of the relation corresponding to the entity type at the 1-side of the relationship type
- Depending upon the minimum cardinality, the foreign key can be declared as NOT NULL or NULL ALLOWED
- The attribute types of the 1:N relationship type can be added to the relation corresponding to the participating entity type

Mapping a Binary 1:N Relationship Type



EMPLOYEE(<u>SSN</u>, ename, address, starting date, *<u>DNR</u>*)

- DNR foreign key refers to DNR in DEPARTMENT, NULL NOT ALLOWED

DEPARTMENT(<u>DNR</u>, dname, dlocation)

Mapping a Binary 1:N Relationship Type

EMPLOYEE(SSN, ename, address, starting date, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	01/01/2000	001
289	Paul Barker	208 Market Street, San Francisco	01/01/1998	001
356	Emma Lucas	432 Wacker Drive, Chicago	01/01/2010	002

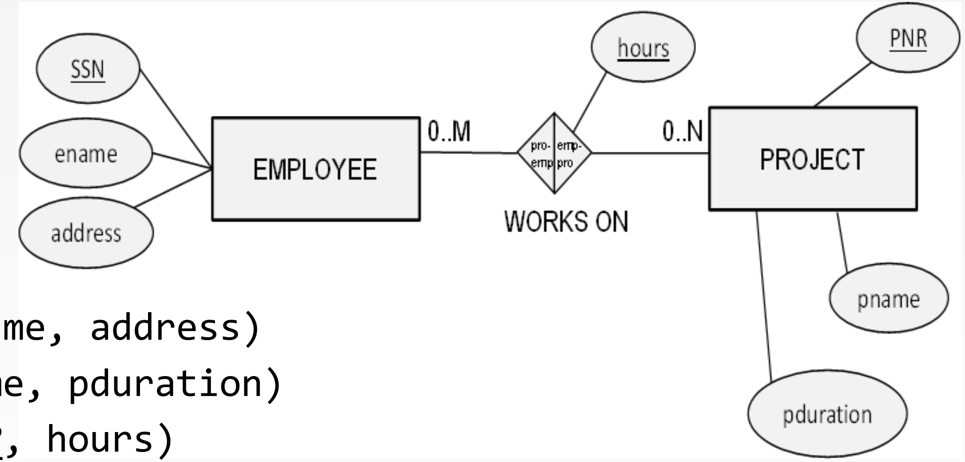
DEPARTMENT(DNR, dname, dlocation, *SSN*)

001	Marketing	3th floor	511
002	Call center	2nd floor	511
003	Finance	basement	289
004	ICT	1st floor	511

Mapping a Binary N:M Relationship Type

- M:N relationship types are mapped by introducing a new relation R
- The primary key of R is a combination of foreign keys referring to the primary keys of the relations corresponding to the participating entity types
- The attribute types of the M:N relationship type can also be added to R

Mapping a Binary N:M Relationship Type



EMPLOYEE(SSN, ename, address)

PROJECT(PNR, pname, pduration)

WORKS_ON(*SSN*, *PNR*, hours)

- SSN foreign key refers to SSN in EMPLOYEE, NULL NOT ALLOWED
- PNR foreign key refers to PNR in PROJECT, NULL NOT ALLOWED

Mapping a Binary N:M Relationship Type

EMPLOYEE(SSN, ename, address, DNR)

511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	001
356	Emma Lucas	432 Wacker Drive, Chicago	002

PROJECT(PNR, pname, pduration)

1001	B2B	100
1002	Analytics	660
1003	Web site	52
1004	Hadoop	826

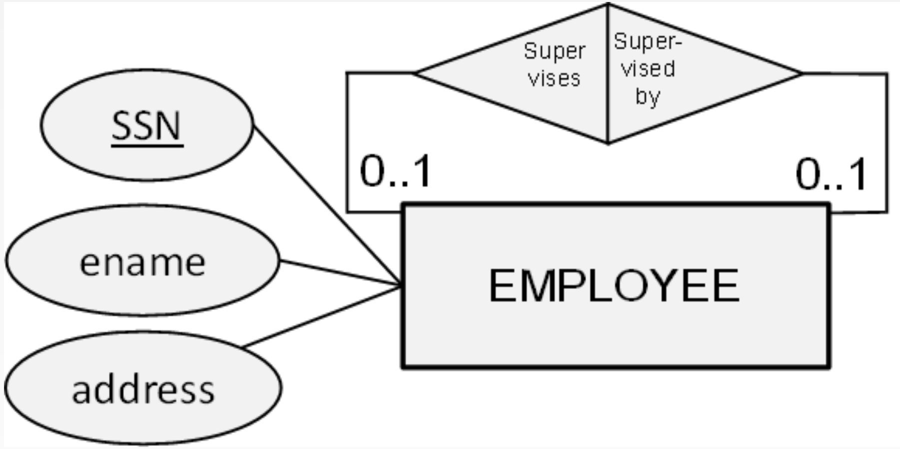
WORKS_ON(*SSN*, *PNR*, hours)

511	1001	10
289	1001	80
289	1003	50

Mapping Unary Relationship Types

- A recursive 1:1 or 1:N relationship type can be implemented by adding a foreign key referring to the primary key of the same relation
- For an N:M recursive relationship type, a new relation R needs to be created with two NOT NULL foreign keys referring to the original relation

Mapping Unary Relationship Types



EMPLOYEE(SSN, ename, address, *supervisor*)

- supervisor foreign key refers to SSN in EMPLOYEE, NULL ALLOWED
- supervisor is an alternate key

Mapping Unary Relationship Types

EMPLOYEE(<u>SSN</u>, ename, address, *supervisor*)

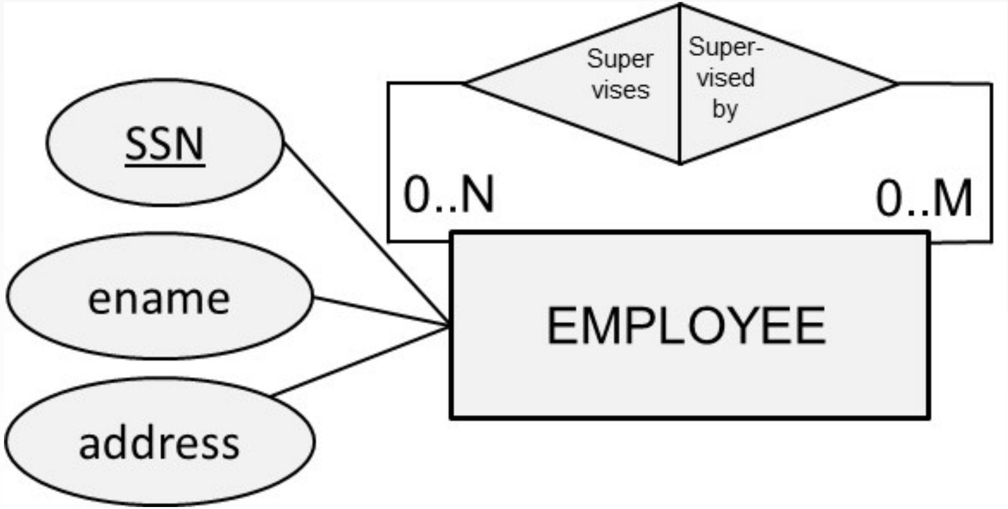
511	John Smith	14 Avenue of the Americas, New York	289
289	Paul Barker	208 Market Street, San Francisco	412
356	Emma Lucas	432 Wacker Drive, Chicago	289
412	Dan Kelly	668 Strip, Las Vegas	NULL

Mapping Unary Relationship Types

EMPLOYEE(<u>SSN</u>, ename, address)

SUPERVISION(<u>*Supervisor*</u>, <u>*Supervisee*</u>)

- Supervisor foreign key refers to SSN in EMPLOYEE, NULL NOT ALLOWED
- Supervisee foreign key refers to SSN in EMPLOYEE, NULL NOT ALLOWED



Mapping Unary Relationship Types

EMPLOYEE(SSN, ename, address)

SUPERVISION(Supervisor, Supervisee)

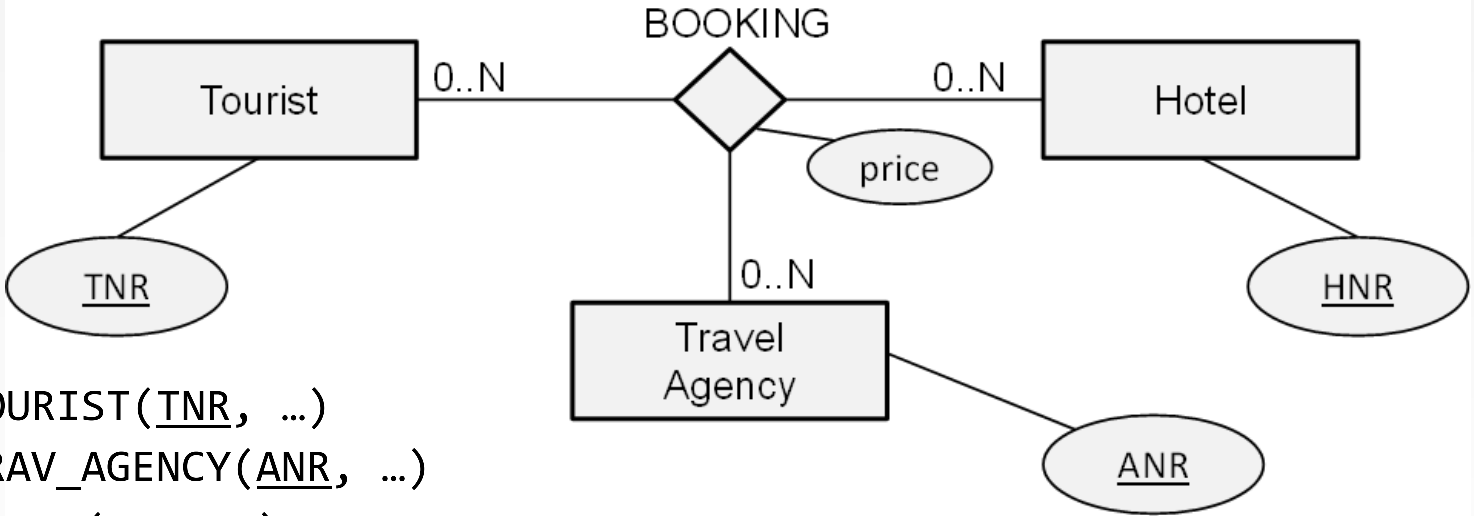
511	John Smith	14 Avenue of the Americas, New York
289	Paul Barker	208 Market Street, San Francisco
356	Emma Lucas	432 Wacker Drive, Chicago
412	Dan Kelly	668 Strip, Las Vegas

289	511
289	356
412	289
412	511

Mapping n-ary Relationship Types

- To map an n-ary relationship type, we first create relations for each participating entity type
- We then also define one additional relation R to represent the n-ary relationship type and add foreign keys referring to the primary keys of each of the relations corresponding to the participating entity types
- The primary key of R is the combination of all foreign keys which are all NOT NULL
- Any attribute type of the n-ary relationship can also be added to R

Mapping n-ary Relationship Types



TOURIST(TNR, …)

TRAV_AGENCY(ANR, …)

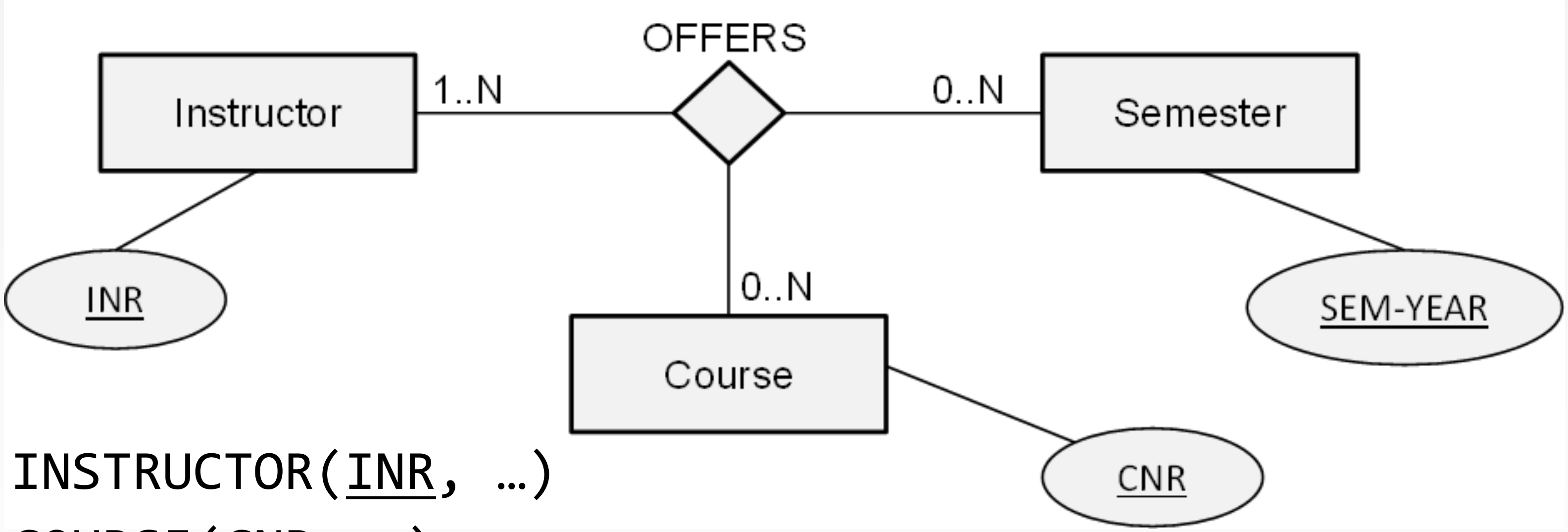
HOTEL(HNR, …)

BOOKING(*TNR*, *ANR*, *HNR*, price)

- TNR foreign key refers to TNR in Tourist, NULL NOT ALLOWED

...

Mapping n-ary Relationship Types



INSTRUCTOR(INR, …)

COURSE(CNR, …)

SEMESTER(SEM-YEAR, …)

OFFERS(*INR*,*CNR*,*SEM-YEAR*)

- INR foreign key refers to ...

Mapping n-ary Relationship Types

INSTRUCTOR(<u>INR</u>, iname, ….)

10	Bart
12	Wilfried
14	Seppe

COURSE(<u>CNR</u>, cname, ….)

100	Database Management
110	Analytics
120	Java Programming

SEMESTER(<u>SEM-YEAR</u>, ….)

1-2015
2-2015
1-2016

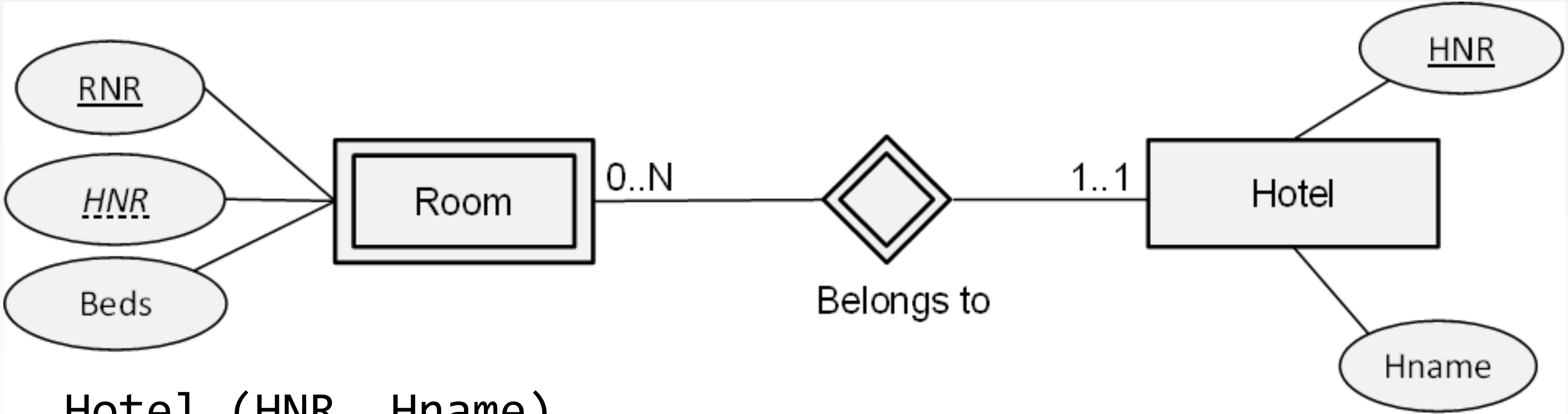
OFFERS(<u>*INR, CNR, SEM-YEAR*</u>)

10	100	1-2015
12	100	1-2016
10	120	1-2015
14	120	1-2015

Mapping Weak Entity Sets

- A weak entity set should be mapped into a relation R with all its corresponding attribute types
- A foreign key must be added referring to the primary key of the relation corresponding to the owner entity type
- Because of the existence dependency, the foreign key is declared as NOT NULL
- The primary key of R is then the combination of the partial key and the foreign key

Mapping Weak Entity Sets



Hotel (<u>HNR</u>, Hname)

Room (<u>RNR</u>, <u>*HNR*</u>, beds)

- HNR foreign key refers to HNR in Hotel, NULL NOT ALLOWED

Mapping Weak Entity Sets

ROOM (<u>RNR</u>, <u>*HNR*</u>, Beds)

2	101	2
6	101	4
8	102	2

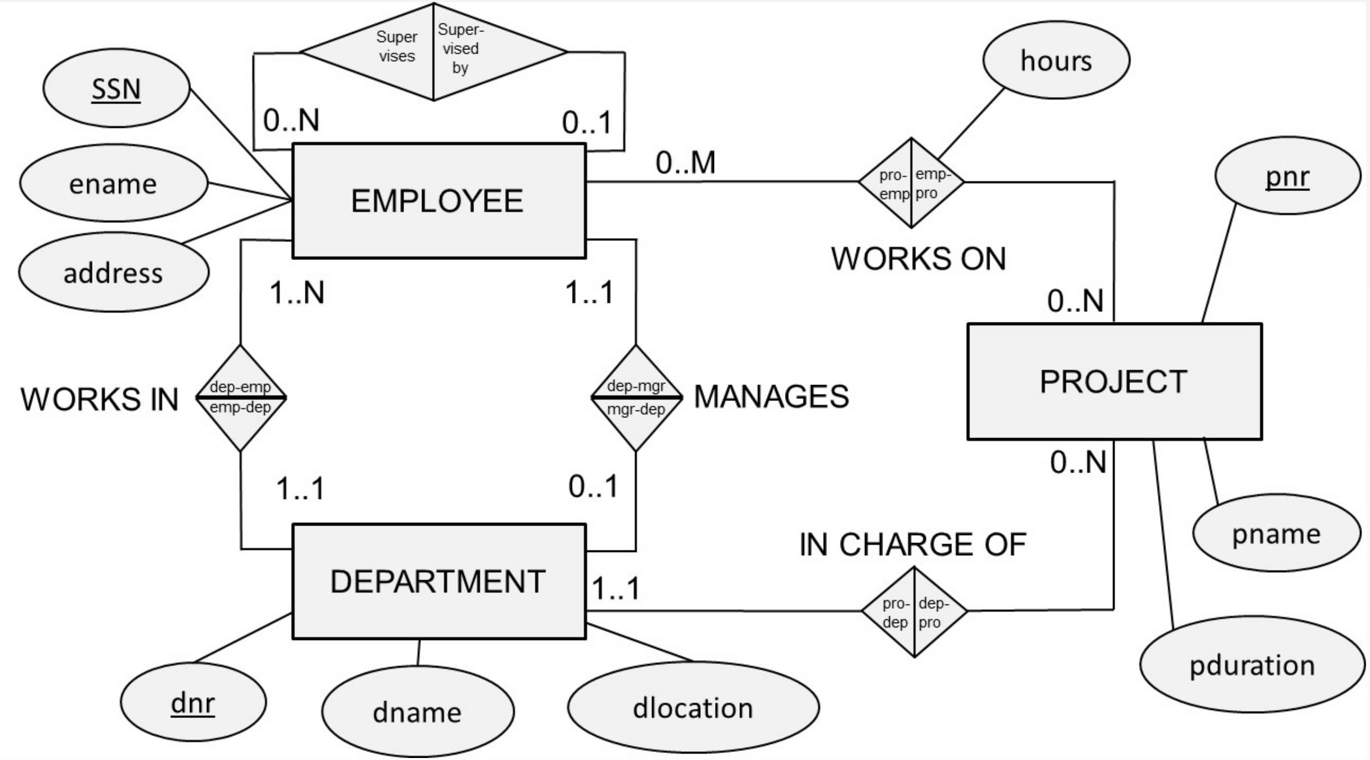
HOTEL(<u>HNR</u>, Hname)

100	Holiday Inn New York
101	Holiday Inn Chicago
102	Holiday Inn San Francisco

Putting It All Together

ER model	Relational model
Entity set	Relation
Weak entity set	Foreign key
1:1 or 1:N relationship type	Foreign key
M:N relationship type	New relation with two foreign keys
n-ary relationship type	New relation with n foreign keys
Simple attribute type	Attribute type
Composite attribute type	Component attribute type
Multi-valued attribute type	Relation and foreign key
Key attribute type	Primary or alternative key

Putting it All Together



Putting It All Together

- EMPLOYEE(<u>SSN</u>, ename, streetaddress, city, sex, dateofbirth, *SUPERVISOR*, *DNR*)
  - SUPERVISOR foreign key refers to SSN in EMPLOYEE, NULL ALLOWED
  - DNR foreign key refers to DNR in DEPARTMENT, NOT NULL
- DEPARTMENT (<u>DNR</u>, dname, dlocation, *MGNR*)
  - MGNR: foreign key refers to SSN in EMPLOYEE, NOT NULL
  - MGNR is an alternate key
- PROJECT (<u>PNR</u>, pname, pduration, *DNR*)
  - DNR: foreign key refers to DNR in DEPARTMENT, NOT NULL
- WORKS-ON (<u>*SSN*</u>, <u>*PNR*</u>, HOURS)
  - SSN foreign key refers to SSN in EMPLOYEE, NOT NULL
  - PNR foreign key refers to PNR in PROJECT, NOT NULL

Reading and Next Class

- Entity/Relationship Models II
  - Ch 2, 3
- Next: SQL I: Ch 5