# Database Design – Entity Relationship Diagrams

- Goal: specification of database schema
- Methodology:
  - Use *E-R model* to get a high-level graphical view of essential components of enterprise and how they are related
  - Convert E-R diagram to DDL
- *E-R Model*: enterprise is viewed as a set of
  - *Entities*
  - *Relationships* among entities



## Example 1

1. Consider the following enterprise, which includes books, authors and publishers. Authors are people with normal attributes, like name, date of birth, etc., but in addition they wrote one or more books. A book has the usual attributes, such as title, ISBN, publication date, etc. Publishers are companies that publish books. They have an address, phone numbers (typically more than one), name, etc.

- A book can be written by more than one author, but it can be published by only one publisher. Books do not write themselves and do not publish themselves (hint: these are constraints). An author can write more than one book and to be called an author one, of course, has to write at least one book.
- Represent the above as an E-R diagram; include all relevant constraints.
- Assumptions: the author name is unique and the publisher name is unique.



# Entities

- *Entity*: an object that is involved in the enterprise
  - Ex: John, CSE305
- *Entity Type*: set of similar objects
  - Ex: students, courses
- *Attribute*: describes one aspect of an entity type
  - Ex: *name*, *maximum enrollment*

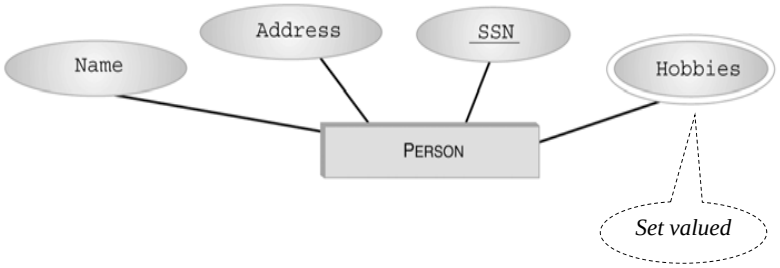


# Entity Type

- Entity type described by set of attributes
  - Person: *Id*, *Name*, *Address*, *Hobbies*
- *Domain*: possible values of an attribute
  - Value can be a set (in contrast to relational model)
    - (111111, John, 123 Main St, {stamps, coins})
- *Key*: minimum set of attributes that uniquely identifies an entity (candidate key)
- *Entity Schema*: entity type name, attributes (and associated domain), key constraints

# Entity Type (con't)

- Graphical Representation in E-R diagram:






# Relationships

- *Relationship*: relates two or more entities
  - John *majors in* Computer Science
- *Relationship Type*: set of similar relationships
  - Student (entity type) related to Department (entity type) by MajorsIn (relationship type).
- Distinction:
  - *relation* (relational model) - set of tuples
  - *relationship* (E-R Model) – describes relationship between entities of an enterprise
  - Both entity types and relationship types (E-R model) may be represented as relations (in the relational model)



# Attributes and Roles

- *Attribute* of a relationship type describes the relationship
  - e.g., John majors in CS *since* 2000
    - John and CS are related
    - 2000 describes relationship - value of SINCE attribute of MajorsIn relationship type
- *Role* of a relationship type names one of the related entities
  - e.g., John is value of *Student* role, CS value of *Department* role of MajorsIn relationship type
  - (John, CS; 2000) describes a relationship



# Relationship Type

- Described by set of attributes and roles
  - e.g., MajorsIn: *Student*, *Department*, *Since*
  - Here we have used as the role name (*Student*) the name of the entity type (Student) of the participant in the relationship, but ...

# Roles

- *Problem*: relationship can relate elements of same entity type
  - e.g., *ReportsTo* relationship type relates two elements of Employee entity type:
    - Bob reports to Mary since 2000
  - We do not have distinct names for the roles
  - It is not clear who reports to whom



# Roles (con't)

- *Solution*: role name of relationship type need not be same as name of entity type from which participants are drawn
  - ReportsTo has roles *Subordinate* and *Supervisor* and attribute *Since*
  - Values of *Subordinate* and *Supervisor* both drawn from entity type Employee



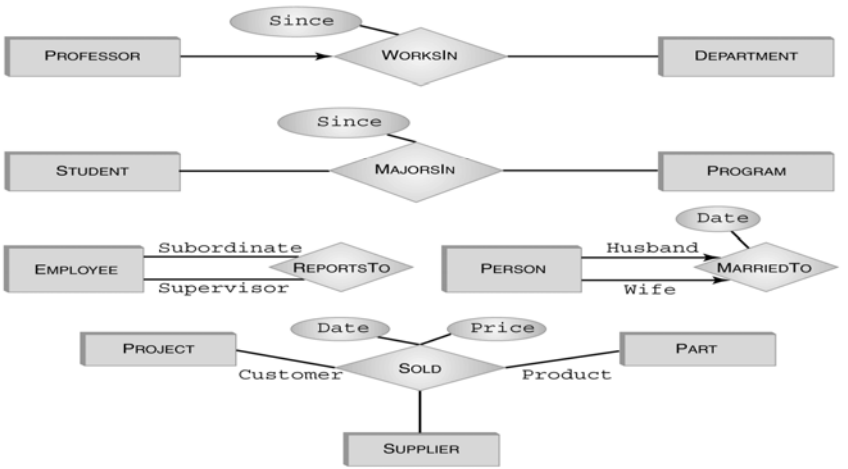
# Schema of a Relationship Type

- *Role names*, $R_i$, and their corresponding entity sets. Roles must be single valued (number of roles = degree of relationship)
- *Attribute names*, $A_j$, and their corresponding domains. Attributes may be set valued
- *Key*: Minimum set of roles and attributes that uniquely identify a relationship
- *Relationship*: $<e_1, \ldots e_n; a_1, \ldots a_k>$
  - $e_i$ is an entity, a value from $R_i$'s entity set
  - $a_j$ is a set of attribute values with elements from domain of $A_j$



# Graphical Representation

- Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.



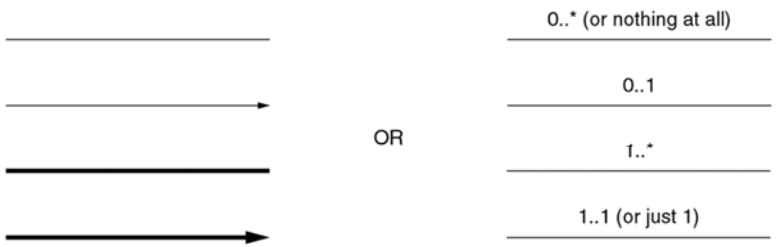



**FIGURE 4.9** Line-based representation vs. cardinality constraints.



# Single-role Key Constraint

- If, for a particular participant entity type, each entity participates in *at most* one relationship, corresponding role is a key of relationship type
  - E.g., *Professor* role is unique in WorksIn
- Representation in E-R diagram: arrow

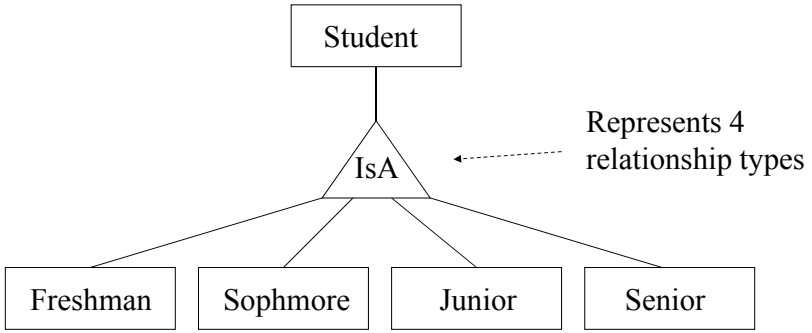



# Entity Type Hierarchies

- One entity type might be subtype of another
  - Freshman is a subtype of Student
- A relationship exists between a Freshman entity and the corresponding Student entity
  - e.g., Freshman John is related to Student John
- This relationship is called *IsA*
  - Freshman IsA Student
  - The two entities related by IsA are always descriptions of the same real-world object



# IsA



# Properties of IsA

- *Inheritance* - Attributes of supertype apply to subtype.
  - E.g., *GPA* attribute of Student applies to Freshman
  - Subtype *inherits* all attributes of supertype.
  - Key of supertype is key of subtype
- *Transitivity* - Hierarchy of IsA
  - Student is subtype of Person, Freshman is subtype of Student, so Freshman is also a subtype of Student

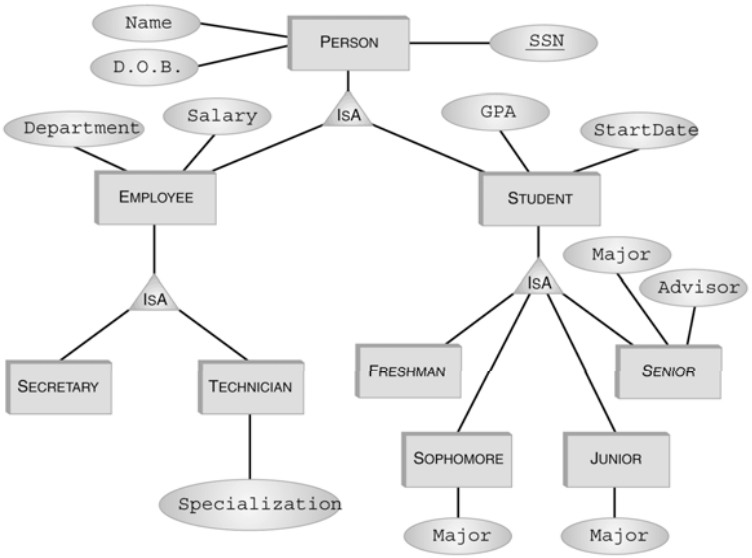


# Advantages of IsA

- Can create a more concise and readable E-R diagram
  - Attributes common to different entity sets need not be repeated
  - They can be grouped in one place as attributes of supertype
  - Attributes of (sibling) subtypes can be different



## IsA Hierarchy - Example

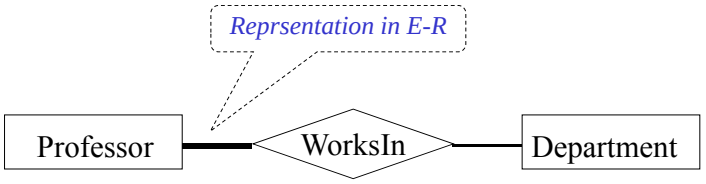



# Constraints on Type Hierarchies

- Might have associated constraints:
  - *Covering constraint*: Union of subtype entities is equal to set of supertype entities
    - Employee is either a secretary or a technician (or both)
  - *Disjointness constraint*: Sets of subtype entities are disjoint from one another
    - Freshman, Sophomore, Junior, Senior are disjoint set

# Participation Constraint

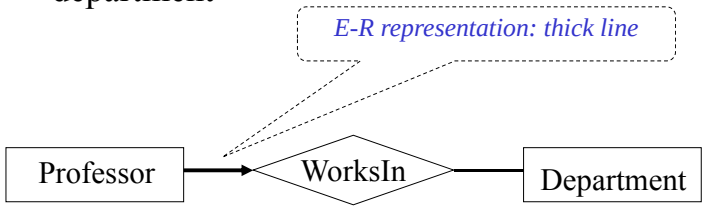
- If every entity participates in *at least* one relationship, a *participation constraint* holds:
  - A participation constraint of entity type E having role ρ in relationship type R states that for *e* in E there is an *r* in R such that ρ(*r*) = *e*.
  - e.g., every professor works in *at least* one department






# Participation *and* Key Constraint

- If every entity participates in *exactly* one relationship, both a participation and a key constraint hold:
  - e.g., every professor works in *exactly one* department






# Representation of Entity Types in the Relational Model

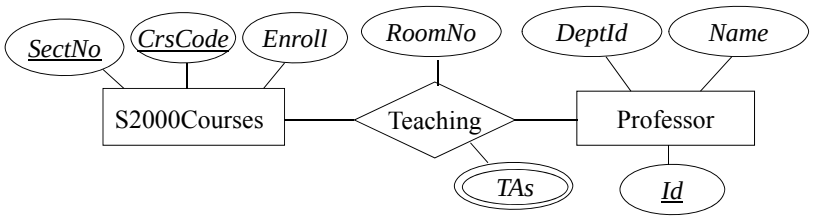
- An entity type corresponds to a relation
- Relation's attributes = entity type's attributes
  - *Problem*: entity type can have set valued attributes, e.g.,

    Person: *Id*, *Name*, *Address*, *Hobbies*
  - *Solution*: Use several rows to represent a single entity
    - (111111, John, 123 Main St, stamps)
    - (111111, John, 123 Main St, coins)
  - Problems with this solution:
    - Redundancy
    - Key of entity type (Id) not key of relation
    - Hence, the resulting relation must be further transformed (Chapter 6)



# Representation of Relationship Types in the Relational Model

- Typically, a relationship becomes a relation in the relational model
- Attributes of the corresponding relation are
  - Attributes of relationship type
  - For each role, the primary key of the entity type associated with that role
- *Example*:




  - S2000Courses (*CrsCode*, *SectNo*, *Enroll*)
  - Professor (*Id*, *DeptId*, *Name*)
  - Teaching (*CrsCode*, *SecNo*, *Id*, *RoomNo*, *TAs*)

# Representation of Relationship Types in the Relational Model

- Candidate key of corresponding table = candidate key of relation
  - Except when there are set valued attributes
  - Example: Teaching (*CrsCode, SectNo, Id, RoomNo, TAs*)
    - Key of relationship type = (*CrsCode, SectNo*)
    - Key of relation = (*CrsCode, SectNo, TAs*)

| *CrsCode* | *SectNo* | *Id* | *RoomNo* | *TAs* |
|---|---|---|---|---|
| CSE305 | 1 | 1234 | Hum 22 | Joe |
| CSE305 | 1 | 1234 | Hum 22 | Mary |

*Set valued*

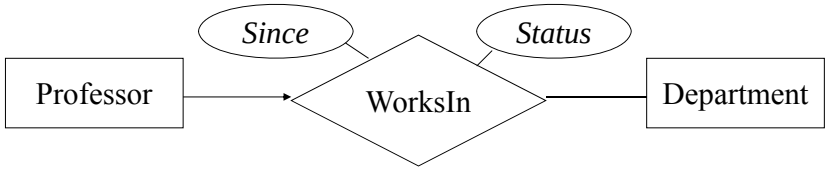


# Representation in SQL

- Each role of relationship type produces a foreign key in corresponding relation
  - Foreign key references table corresponding to entity type from which role values are drawn



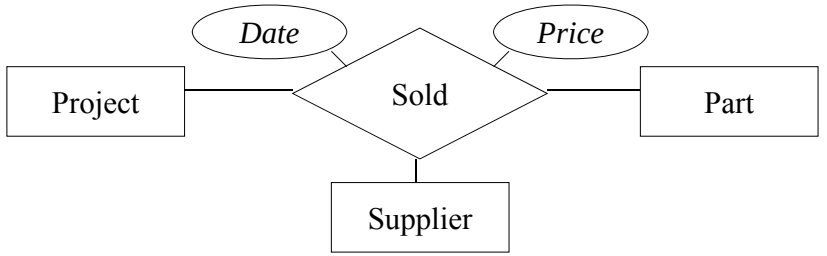
# Example 1




```
CREATE TABLE WorksIn  (
   Since DATE,                -- attribute
   Status  CHAR (10),         -- attribute
   ProfId  INTEGER,           -- role (key of  Professor)
   DeptId  CHAR (4),          -- role (key of  Department)
   PRIMARY KEY (ProfId),  -- since a professor works in at most one department
   FOREIGN KEY (ProfId) REFERENCES Professor (Id),
   FOREIGN KEY (DeptId) REFERENCES Department )
```



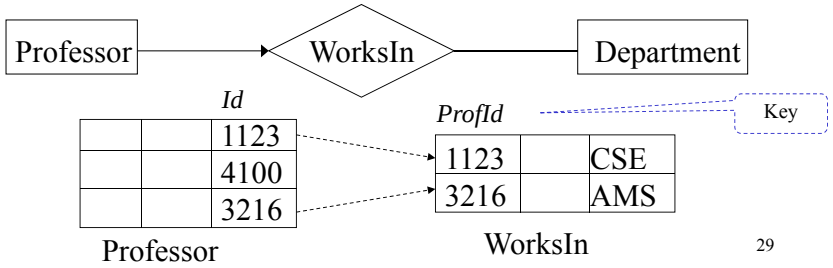
# Example 2




```
CREATE TABLE  Sold  (
   Price  INTEGER,            -- attribute
   Date  DATE,                -- attribute
   ProjId  INTEGER,           -- role
   SupplierId  INTEGER,       -- role
   PartNumber  INTEGER,       -- role
   PRIMARY KEY (ProjId, SupplierId, PartNumber, Date),
   FOREIGN KEY (ProjId) REFERENCES Project,
   FOREIGN KEY (SupplierId) REFERENCES Supplier (Id),
   FOREIGN KEY (PartNumber) REFERENCES Part (Number)  )
```

## Representation of Single Role Key Constraints in the Relational Model

- *Relational model representation*: key of the relation corresponding to the entity type is key of the relation corresponding to the relationship type
  - *Id* is primary key of Professor; *ProfId* is key of WorksIn. Professor 4100 does not participate in the relationship.
  - Cannot use foreign key in Professor to refer to WorksIn since some professors may not work in any dept. (But *ProfId* is a foreign key in WorksIn that refers to Professor.)





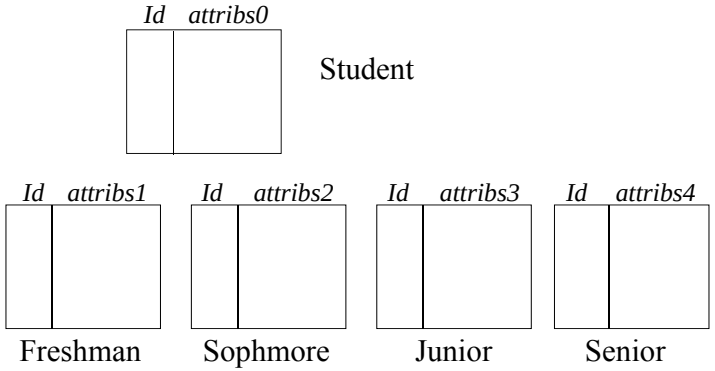

## Representing Type Hierarchies in the Relational Model

- Supertypes and subtypes can be realized as separate relations
  - Need a way of identifying subtype entity with its (unique) related supertype entity
    - *Choose a candidate key and make it an attribute of all entity types in hierarchy*



## Type Hierarchies and the Relational Model

- Translated by adding the primary key of supertype to all subtypes. Plus foreign key from subtypes to the supertype.




FOREIGN KEY *Id* REFERENCES Student

in Freshman, Sophomore, Sunior, Senior



## Type Hierarchies and the Relational Model

- Redundancy eliminated if IsA is not disjoint
  - For individuals who are both employees and students, Name and DOB are stored only once

Person

| *SSN* | *Name* | *DOB* |
|---|---|---|
| 1234 | Mary | 1950 |

Employee

| *SSN* | *Department* | *Salary* |
|---|---|---|
| 1234 | Accounting | 35000 |

Student

| *SSN* | *GPA* | *StartDate* |
|---|---|---|
| 1234 | 3.5 | 1997 |

## Type Hierarchies and the Relational Model

- Other representations are possible in special cases, such as when all subtypes are disjoint
- See in the book



## Representing Participation Constraints in the Relational Model

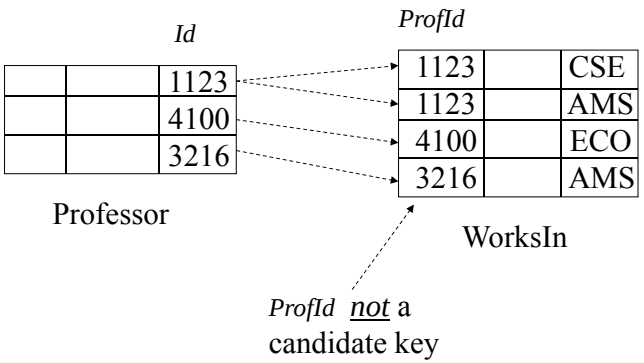

- *Inclusion dependency*: Every professor works in *at least* one dep't.
  - in the relational model: (easy)
    - Professor (*Id*) references WorksIn (*ProfId*)
  - in SQL:
    - Simple case: *If ProfId is a key in* **WorksIn** (i.e., every professor works in *exactly one* department) then it is easy:
      - FOREIGN KEY *Id* REFERENCES WorksIn (*ProfId*)
    - General case – *ProfId* is *not* a key in **WorksIn**, so can't use foreign key constraint (not so easy):

```
CREATE ASSERTION  ProfsInDepts
   CHECK ( NOT EXISTS (
      SELECT  *  FROM  Professor P
      WHERE NOT EXISTS (
         SELECT  *  FROM  WorksIn W
         WHERE   P.Id = W.ProfId ) ) )
```



## Representing Participation Constraint in the Relational Model

- Example (can't use foreign key in Professor if ProfId is not a candidate key in WorksIn)





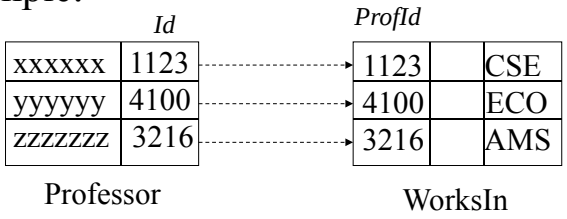

## Representing Participation *and* Key Constraint in SQL

- If both participation and key constraints apply, use foreign key constraint in entity table (but beware: if candidate key in entity table is not primary, presence of nulls violates participation constraint).

```
CREATE TABLE Professor  (
   Id   INTEGER,
        ……
   PRIMARY KEY (Id),      -- Id can't be null
   FOREIGN KEY (Id) REFERENCES WorksIn  (ProfId)
                          --all professors participate
)
```

# Participation *and* Key Constraint in the Relational Model

- Example:






# Participation *and* Key Constraint in Relational Model (again)

- Alternative solution if both key and participation constraints apply: merge the tables representing the entity and relationship sets
  - Since there is a 1-1 and onto relationship between the rows of the entity set and the relationship sets, might as well put all the attributes in one table



# Participation *and* Key Constraint in Relational Model

- Example

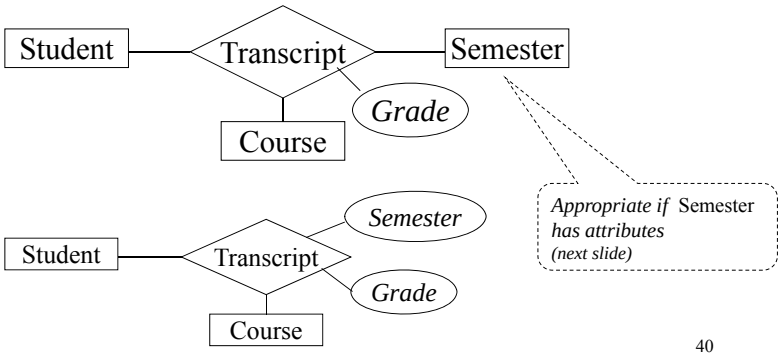
| *Name* | *Id* | *DeptId* |
|---|---|---|
| xxxxxxx | 1123 | CSE |
| yyyyyyy | 4100 | ECO |
| zzzzzzzz | 3216 | AMS |

Prof_WorksIn

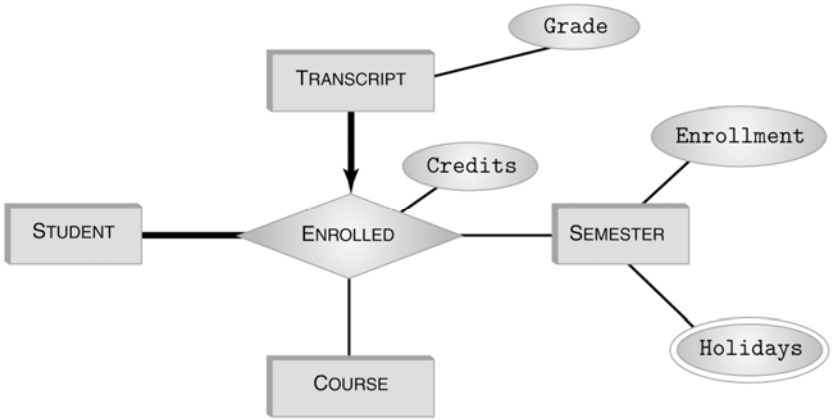


# Entity or Attribute?

- Sometimes information can be represented as either an entity or an attribute.



# Entity or Relationship?






# (Non-) Equivalence of Diagrams

- Transformations between binary and ternary relationships.