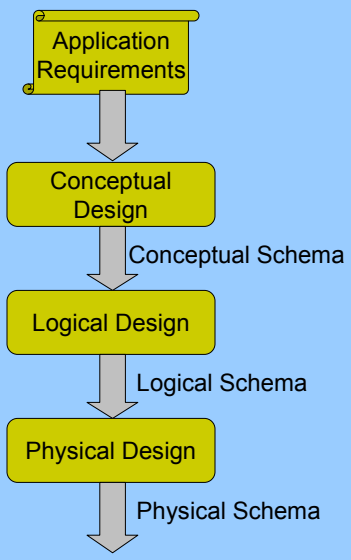
The Entity-Relationship Model

Murali Mani

WPI

Database Design Stages



Murali Mani

WPI

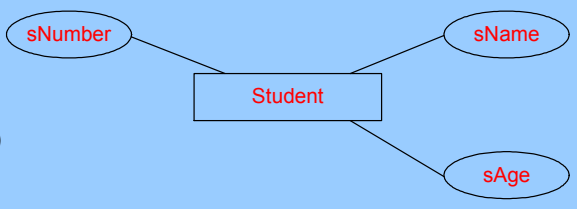
Conceptual Design

- What is conceptual Design?
  - Concise representation of our DB application requirements
- Conceptual Models
  - **ER (Entity Relationship)** Model, UML (Unified Modeling Language), ORM (Object Role Modeling) etc
- ER Model
  - Structures: entities and relationships
  - Constraints
  - An ER schema is represented as an ER diagram.

ER Model: Entity Types and Attributes

- **Entity:** "Object"
- **Entity Type:** "Class"
- **Attribute:** property of an entity, has a domain
- **In ER diagrams**
  - Entity Type → rectangle
  - Attribute → Oval.

Entity Type **Student**
with attributes
(sNumber, sName, sAge)



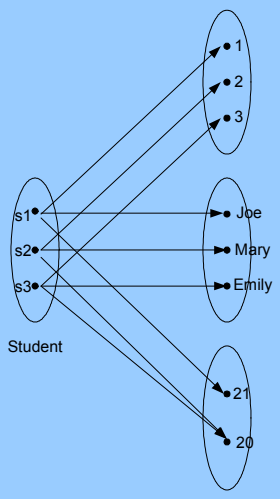
ER Example

- Consider DB instance with 3 students

  (1, Joe, 21),

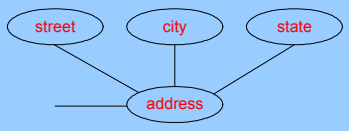
  (2, Mary, 20),

  (3, Emily, 20)



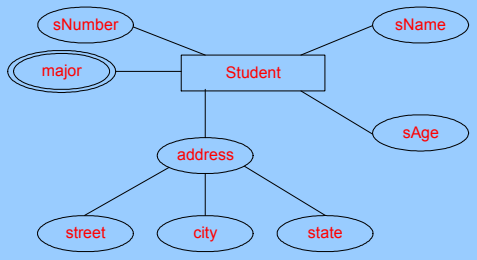
ER Model: Complex Attributes

Composite Attribute: **address**



Multivalued Attribute: **major**

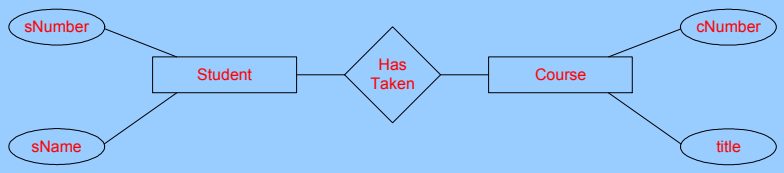
Student entity type with all its attributes



ER Model: Relationship Types

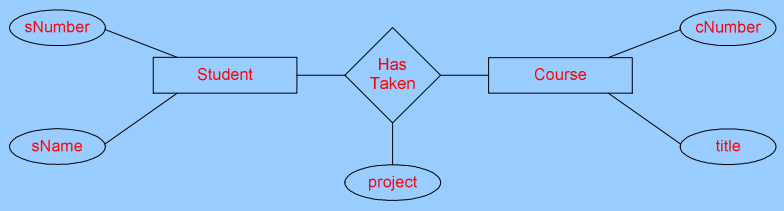
- **Relationship:** Association between entities
- **Relationship Type:** class of relationships
- Represented as diamond

Relationship type **HasTaken** to represent **Courses** taken by **Students**



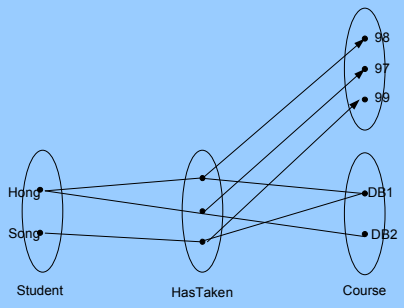
ER Model: Relationship Types with Attributes

Relationship **HasTaken** has an attribute **project** which is the project the **Student** did for the **Course**



Example: Relationship Instance

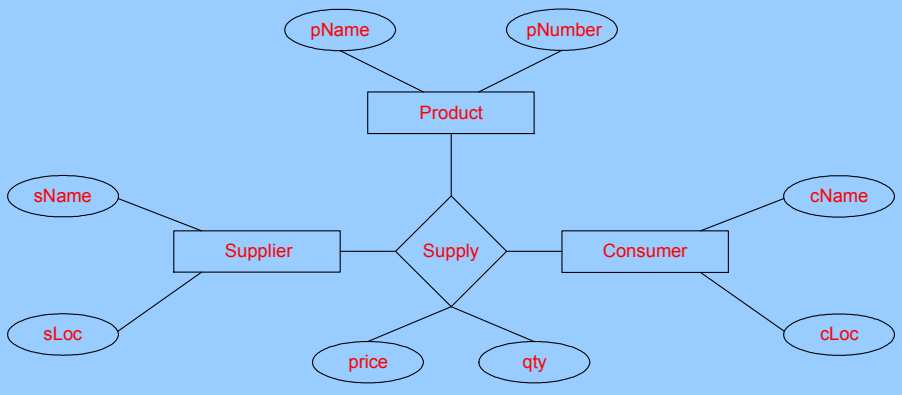
- Consider students {Hong, Song}, courses {DB1, DB2}, and the relationships {(Hong, DB1, 98), (Song, DB1, 99), (Hong, DB2, 97)}



N-ary relationship type

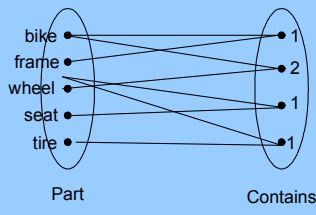
Ternary relationship type: **Supplier** supplies **Products** to **Consumers**

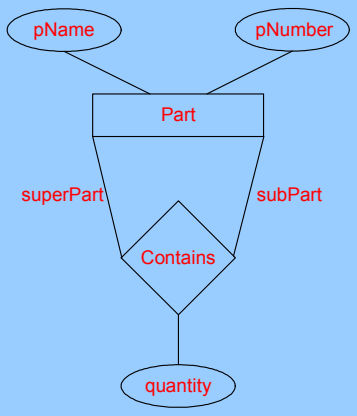
Note: This is **NOT** equivalent to 2 binary relationships



Recursive Relationship Types and Roles

Part-Subpart recursive relationship type
Roles: There are **Parts** that play the role of **superPart**
There are **Parts** that play the role of **subPart**





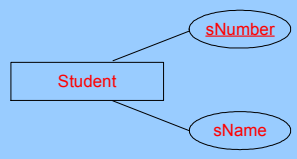
ER Model so far

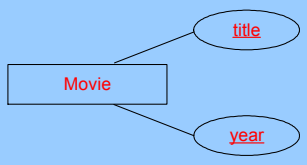
- Structures
  - Entity Types
  - Relationship Types
    - Binary, ternary, n-ary
    - Recursive
  - Attributes
    - For entity types and relationship types
    - Simple, composite, multivalued
  - Roles

ER Model: Key Constraints

Underline the key attribute/attributes

Key for **Student** is **sNumber**



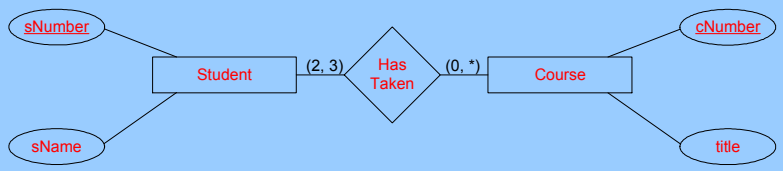


Key for **Movie** is **<title, year>**

Note:
We **can** represent key for an entity type consists of more than 1 attribute (eg: **Movie**)
We **cannot** represent multiple keys for an entity type (eg: key for **Student** can be either **sNumber** or **sName**)

ER Model: Cardinality Constraints

Expressed using **(min, max)**

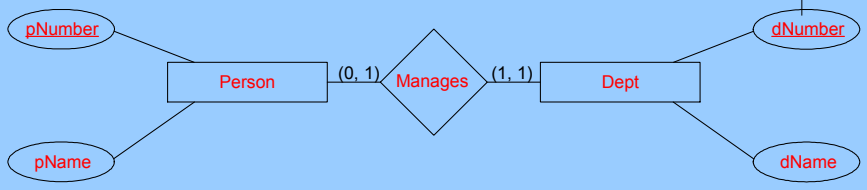


Student can take >= 2 and <= 3 **Courses**
Course can have >= 0 and <= * (infinity) **Students**

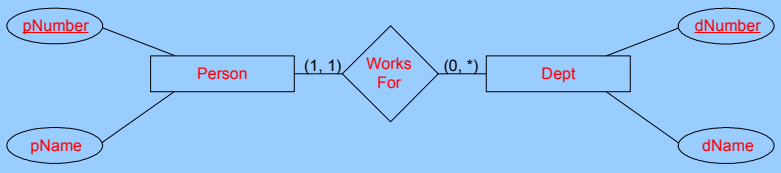
min and max are non-negative integers
max >= min

Cardinality Constraints

1:1 relationship type: A **Dept** has exactly one **Manager**,
A **Person** can manage atmost one **Dept**

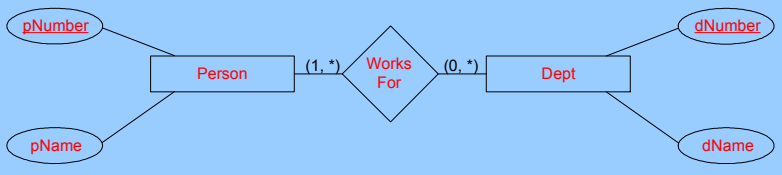


1:many (1:n) relationship type: A **Person** works for exactly one **Dept**,
A **Dept** can have any number of **Persons**

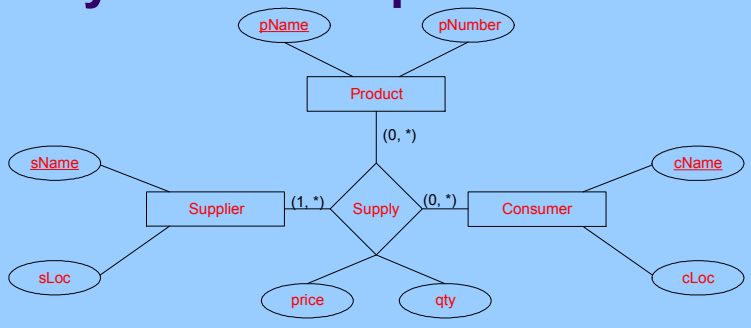


Cardinality Constraints

many:many (m:n) relationship type: A **Person** works for one or more **Depts**,
A **Dept** can have any number of **Persons**



Cardinality Constraints for n-ary relationships

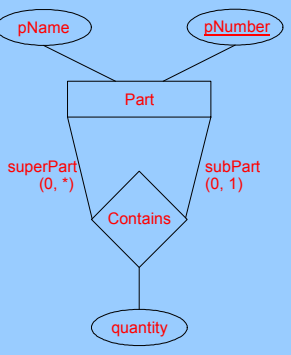


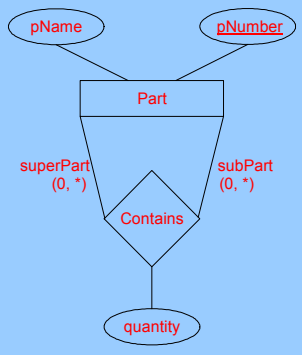
A **Supplier** supplies at least one **Product** to some **Consumer**

We **cannot** specify:
A **Consumer** gets a **Product** from only one **Supplier**
Each **Supplier** supplies exactly 2 **Products**

Cardinality Constraints for Recursive Relationships

A **Part** can be subpart of one **superPart**
A **Part** can have many **subParts**





A **Part** can be subpart of many **superParts**
A **Part** can have many **subParts**

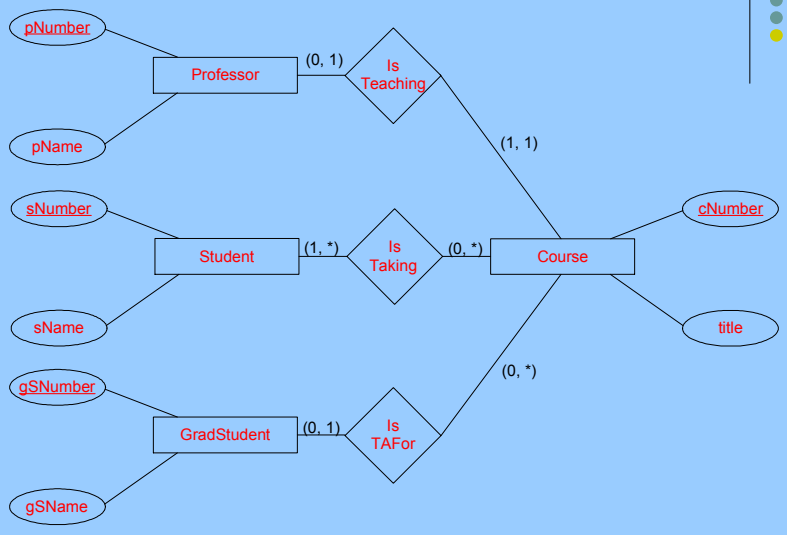
ER Model Constraints Summary

- Key Constraints
- Cardinality Constraints
  - Expressed using (min, max)
  - Binary relationship types are called 1:1, 1:many, many:many

An Application Example

- **Courses offered in CS Dept, WPI, in A term**
  - What entity types? – Student, Professor, Course, GradStudent
  - Attributes and key constraints for entity types
  - What relationship types?
  - Cardinality for Relationship Types

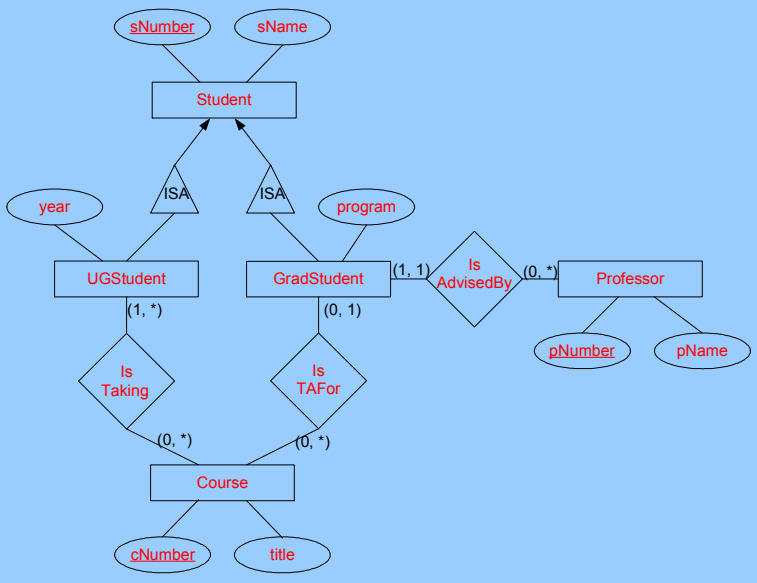
Possible Solution



ER Model: ISA Relationship Types

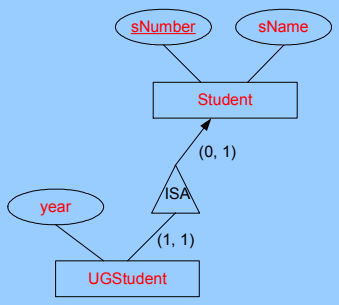
Similar to “subclass”

Students can be **UGStudents** or **GradStudents**
UGStudents take **Classes**, **GradStudents** are TAs for **Classes**
GradStudents are advised by **Professors**



Murali Mani

ISA

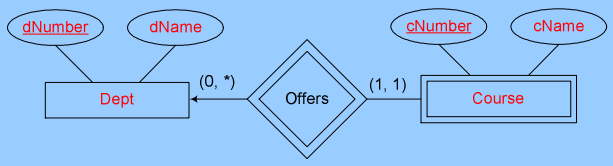


Note:
Implicit 1:1 relationship
Key for subtype is same as key for supertype
Subtypes can have additional attributes

Murali Mani

Weak Entity Types

Consider **Depts** and **Courses**
The **Courses** offered by a **Dept** are identified by Cnumber
Course is weak entity type
Its identifying relationship is **Offers**
Its identifying entity type is **Dept**
A weak entity type can have multiple identifying relationship types and entity types



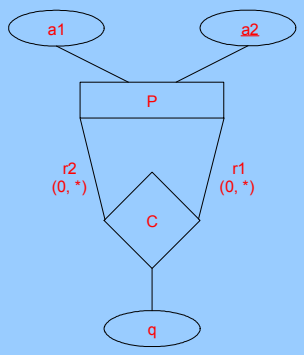
Note: The cardinality of the weak entity type in a identifying relationship type is (1, 1)

Summary of ER

- **Structures**
  - **Entity Types**
  - **Relationship types – binary, ternary, n-ary. recursive**
  - **Attributes**
    - **For entity types or relationship types**
    - **Simple, composite or multi-valued**
- **Constraints – key, cardinality**
- **Roles of entity types in a relationship type**
- **ISA relationship types**
- **Weak Entity Types – identifying relationship type, identifying entity type**

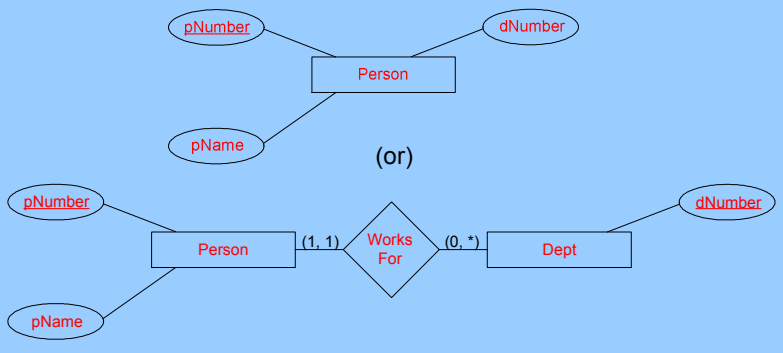
Coming up with a good design for your application

- Give good names to entity types, relationship types, attributes and roles



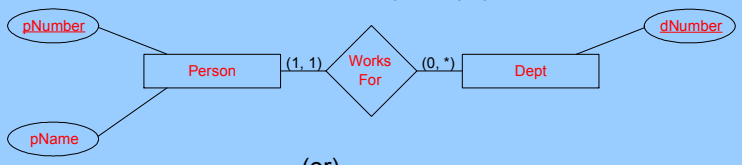
Good Design: Attribute or entity type?

Should we represent something as an attribute or entity type?
How should **dept** be represented?

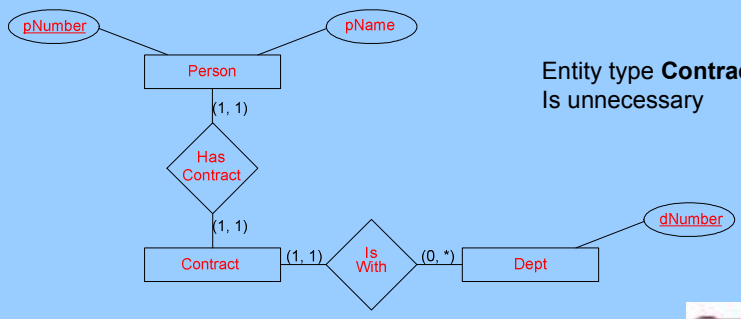


Good Design: Keep it simple

Do not introduce unnecessary entity types

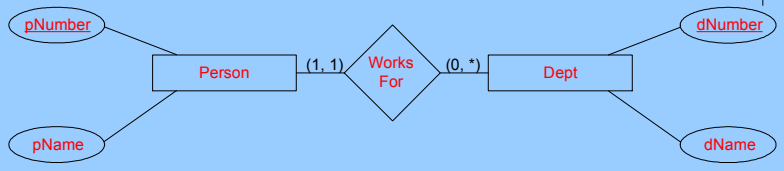


(or)

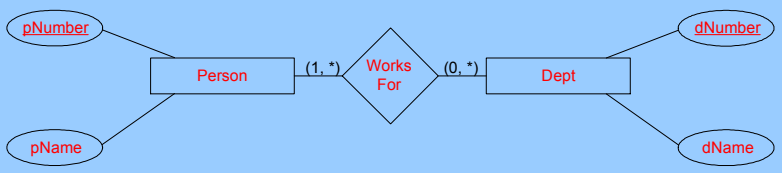


Entity type **Contract**
Is unnecessary

Good Design: Determine correct cardinality constraints

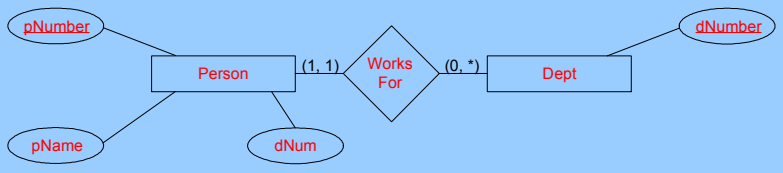


(or)



Good Design: Try to avoid redundancy

Redundant attribute
Attribute **dNum** is redundant



Good Design: Try to Avoid redundancy

Redundant relationship type
Relationship Type **IsObtainedBy** is redundant