

# CS3743 Introduction to Databases

## Entity-Relationship Model

Prof. Weining Zhang

August 31, 2009

## Steps to Design a Database

- 1 Requirement analysis. Decide what subjects and relationships in the application domain should be represented by data, and what information needs to be represented
- 2 Conceptual design. Describe a conceptual (abstract) schema of the database using a conceptual data model and language
- 3 Logical design. Design a formal schema of the database according to the conceptual schema and describe it using the data model supported by a type of DBMSs
- 4 Physical design. Describe the logical schema using the language supported by a specific DBMS
- 5 Database creation. Load data into the database using methods supported by the DBMS

## Outline

- 1 Conceptual Database Design
  - Overview
- 2 The Entity-Relationship Model
  - Entity Types
  - Relationship Types
  - Constraints
  - Weak Entity Type
- 3 Extended E/R Model
  - Structural Constraint
  - Generalization and Specialization
  - More Examples
- 4 Summary

## Requirement Analysis

- The purpose is to determine what in the application domain needs to be modeled by data and what information is needed in this model.
- The outcome is a document describing data modeling requirements, such as, “A student has id, name, GPA, address, and age”, “A course has title, course number and hours”, “A student may take 1 to 5 courses”, etc.
- The quality of requirement analysis determine the quality of the database and has a profound impact on the quality of database applications

# Conceptual Design

- Design an abstract schema from the requirements that conceptually describe all data contained in the database
- A conceptual schema is independent of any specific DBMS, is easier to design, and is based on concepts more familiar to domain experts
- The conceptual schema is often described using a conceptual modeling language, such as,
  - the Entity-Relationship (ER) Diagram (most popular)
  - the Object Description Language (ODL)
  - or the Universal Modeling Language (UML) (getting more popular)

# Data Requirements

## Example

Some requirements of a university information system:

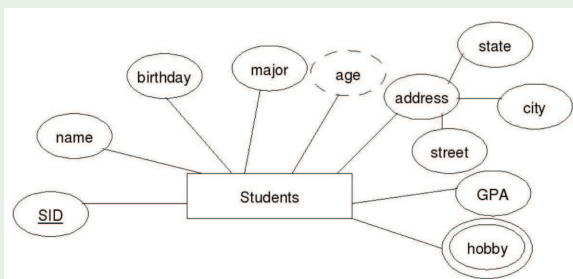
- There are Students, Courses, Faculty, Departments, Students consists of Graduate students and Undergraduate students, Courses has Sections
- Students enroll in courses, Faculty teaches courses, Courses are offered by Departments, Faculty supervise Graduate students
- Each course must have 5 to 40 students, a Students can register 1 to 5 courses
- ...

# Entity Type

Represent properties of a set of entities

## Example

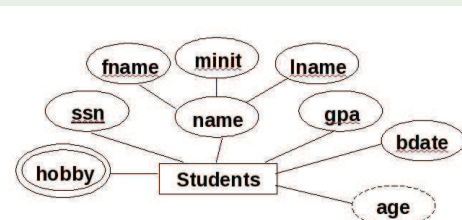
- A student entity has ID, name, birthdate, major, GPA, city, etc.



# Attribute

- An attribute can be composite (with components), multi-valued (with set of values), derived (computed from other attributes), and key (unique for each entity)
- The set of valid values of the attribute is its domain

## Example



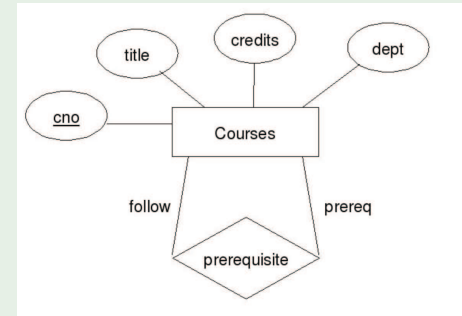
# Relationship Type

- A relationship type defines the properties of a set of relationship instances
- Important relationship properties include
  - Degree: the number of entities each relationship involves
    - Unary, binary, tertiary, etc.
  - Constraints
    - Cardinality, participation, and many more

# Unary Relationship

## Example

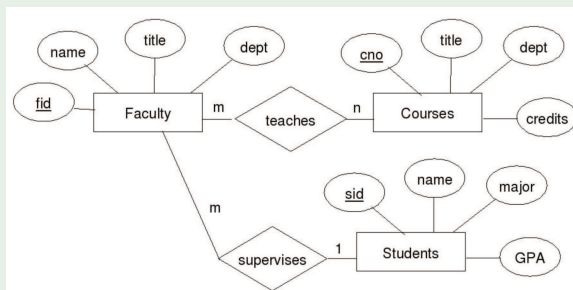
- A course may have one or more prerequisite courses.
- This relationship type involves only the Courses entity type (unary)



# Binary Relationship

## Example

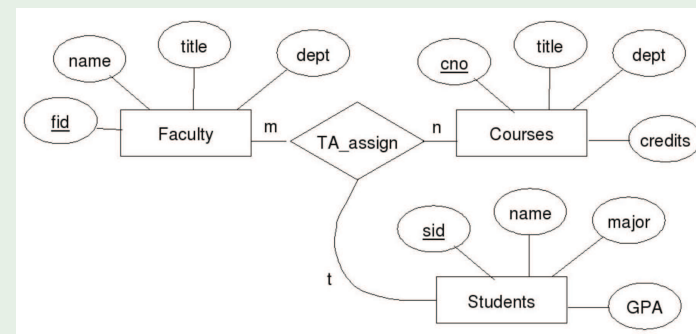
- A faculty may teach more than one course and a course may be taught by more than one faculty
- A student may be supervised by at most one faculty



# Tertiary Relationship

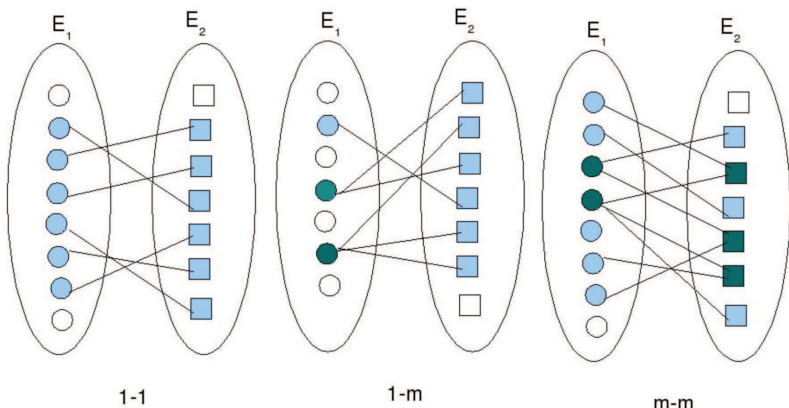
## Example

- Teaching assistants (TAs) are assigned to Courses and Faculty teaching the courses



# Cardinality Constraint

- How many entities from other entity types will be related to an entity of a given type, one-to-one, one-to-many, or many-to-many?

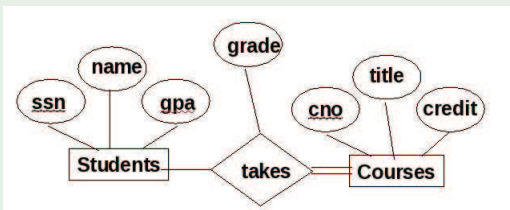


# Participation Constraint

Whether or not every entity in an entity type participates in a relationship type.

## Example

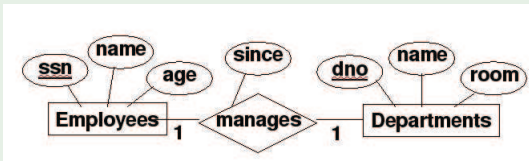
Every course must have some student (total participation)  
 Some student may not take any course (partial participation)



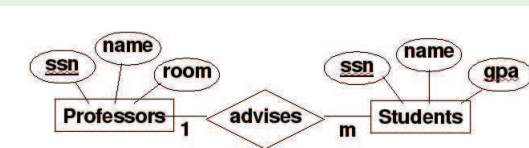
# Cardinality Examples

## Examples

Each department may have one manager

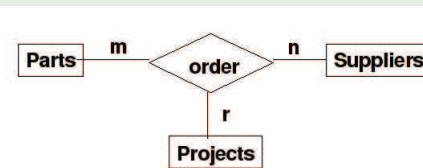


Each professor may advise many students



# Multi-way Relationship

## Example



## What does it really say?

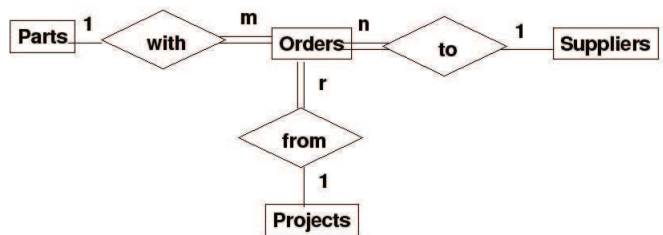
All of the following:

- A part can be supplied by many suppliers to many projects
- A supplier can supply many parts to many projects
- A project can use many parts from many suppliers

# Alternative Multi-way Relationship

## Example

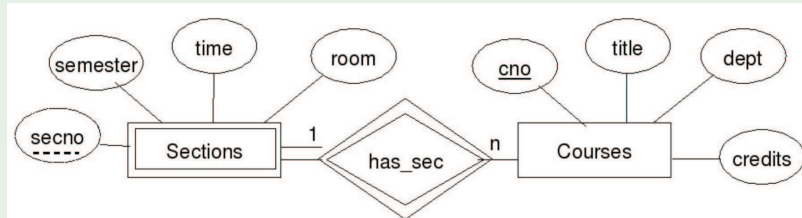
Using an entity type with total participation



# Weak Entity Type

## Example

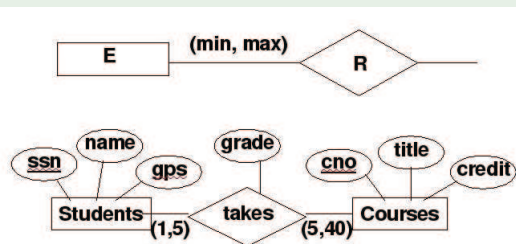
- A Course may have one or more sections
- Sections are weak entities since they do not have a key. They must be identified by a course (a strong entity) through an identifying relationship. This requires a **total participation** on sections.



# Structural Constraint

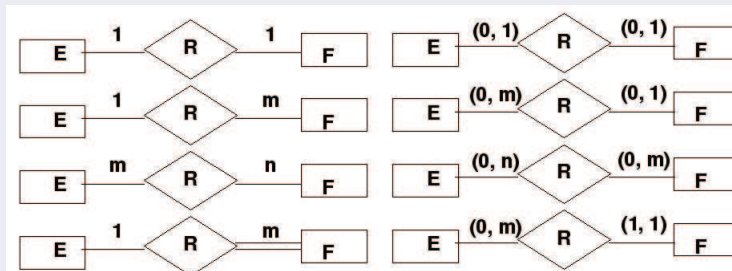
## Example

A student must register for at least 1 and at most 5 courses  
 A course must be have at least 5 and at most 40 students



# Cardinality vs. Structural Constraints

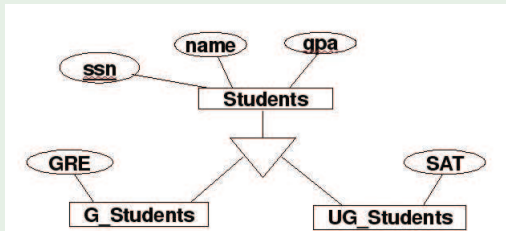
## Comparison



# IS-A Relationship

A subclass is an entity type contains a subset of entities of another entity type (the superclass). Every entity in the subclass is a member of the superclass with all attributes of the superclass.

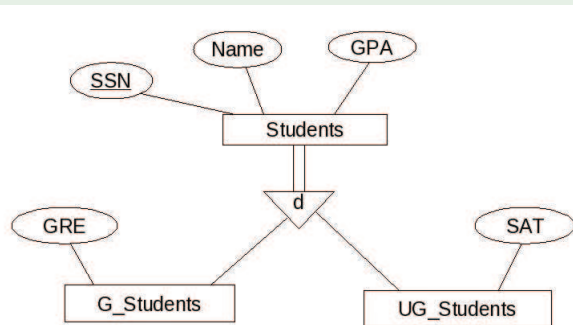
## Example



# Generalization

## Example

Graduate and undergraduate students are students



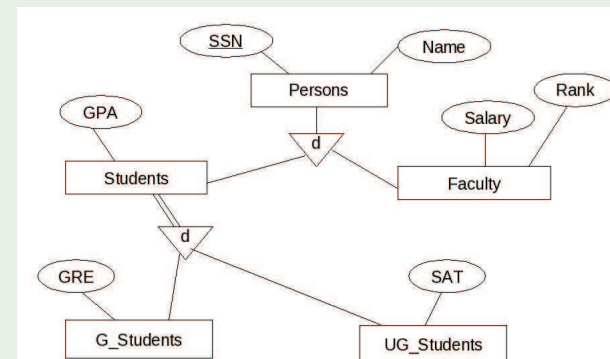
# Types of IS-A Relationships

- Defined by predicate or by attributes
- Subclasses can be disjoint or overlap
- Superclass can participate partially (specialization) or totally (generalization)

# ISA Hierarchy

## Example

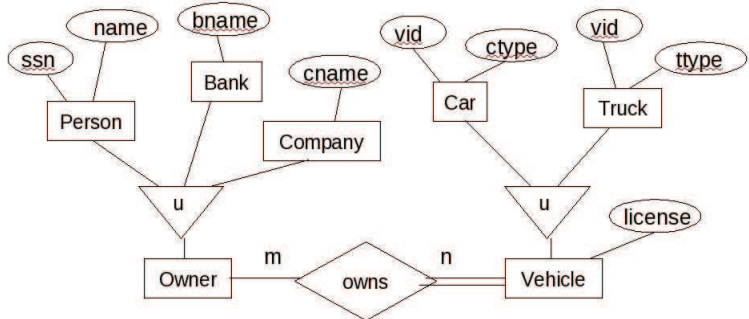
Graduate and undergraduate students are students, students and faculty are persons



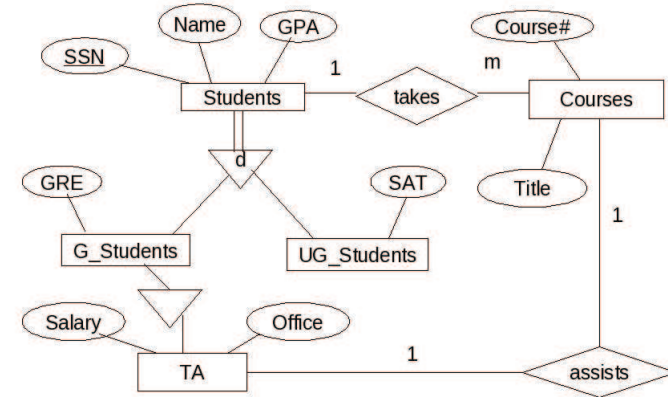
# Category

## Example

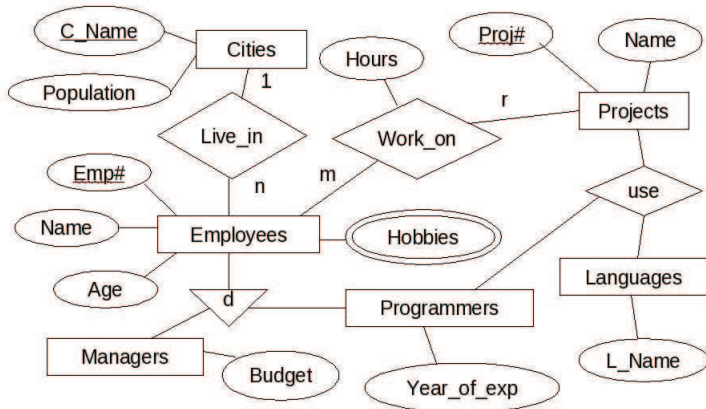
Owners consist of persons, banks, and companies; vehicles consist of cars and trucks



# A Sample EER



# Another Sample EER



# Design Choices

When design an ER model, one needs to make many design choices, such as,

- When to use entity type and when to use attribute
- Should one uses several binary relationships or a multi-way relationships
- When should one use IS-A relationships
- Should a entity type be strong or weak

These decisions may impact the complexity of the schema and the quality of the database