CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #8: E/R Models

Till the Midterm Examination

- **Exam is on March 8, during class**
- Relational Data Models
  - The Entity-Relationship (ER) model
  - The relational model
  - Converting E/R diagram to relational designs.
- You should know how to
  - Identify all entities and relationships and describe them using an E/R diagram
  - Convert the E/R model to a number of relations in a relational schema.
- Use all these ideas to design your own database application in your project.

Database Design

- Requirements Analysis
- Conceptual Design
- Logical Design
- Schema Refinement
- Physical Design
- Security Design

Database Design

▪ Requirements Analysis	*user's needs*
▪ Conceptual Design	*high level (E/R)*
▪ Logical Design	*tables (schema)*
▪ Schema Refinement	*normalization*
▪ Physical Design	*indices etc.*
▪ Security Design	*access controls*

Basic Database Terminology

- Data model : describes high-level conceptual structuring of data
  - Example: Data is set of student records, each with ID, name, address, and courses
  - Example: Data is a graph where nodes represent people and edges represent friendship relations
- Schema describes how data is to be structured and stored in a database
  - Defined during creation of the database
  - Schemas rarely change
- Data is actual "instance" of database
  - Updated continuously
  - Changes rapidly

Why Learn About Database Modeling?

- The way in which data is stored is very important for subsequent access and manipulation by SQL.

- Properties of a good data model:
  - It is easy to write correct and easy to understand queries.
  - Minor changes in the problem domain do not change the schema.
  - Major changes in the problem domain can be handled without too much difficulty.
  - Can support efficient database access.

Purpose of E/R Model

- The E/R model allows us to sketch the design of a database informally.
  - Represent different types of data and how they relate to each other
- Designs are drawings called *entity-relationship diagrams*.
- Fairly mechanical ways to convert E/R diagrams to real implementations like relational databases exist.

Ideas → E/R design → Relational schema → Relational DBMS

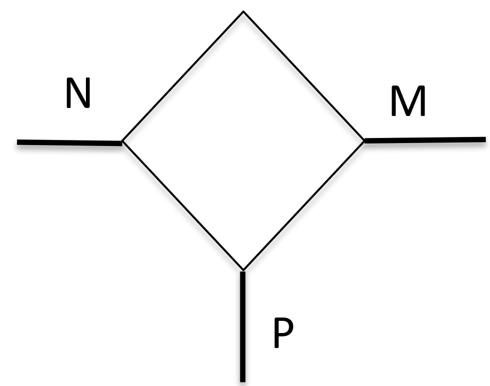
Purpose of E/R Model

- When designing E/R diagrams,
  - forget about relations/tables!
  - only consider how to model the information you need to represent in your database.

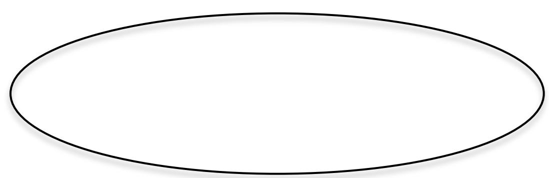
Tools

- Entities ('entity sets')

- Relationships ('rel. sets') and mapping constraints



- Attributes



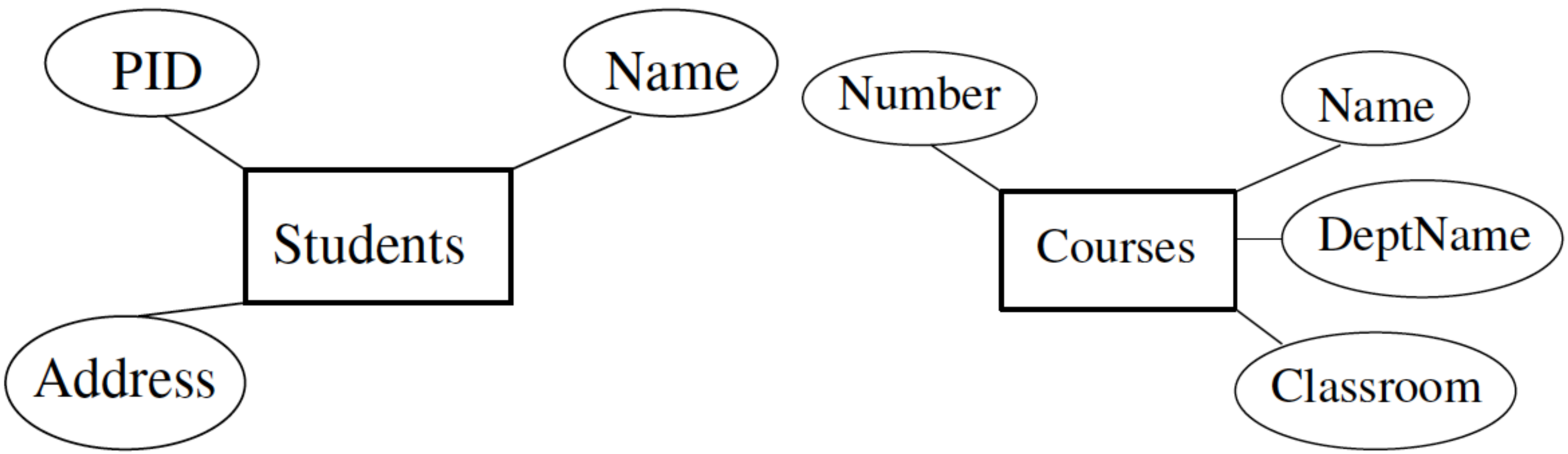
Entity Sets

- *Entity* = "thing" or objects
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- Attribute = property of an entity set.
  - Generally, all entities in a set have the same properties.
  - Our convention is to use 'atomic attributes' e.g. integers, character strings etc.

E/R Diagrams

- In an entity-relationship diagram, each entity set is represented by a rectangle.
- Each attribute of an entity set is represented by an oval, with a line to the rectangle representing its entity set.

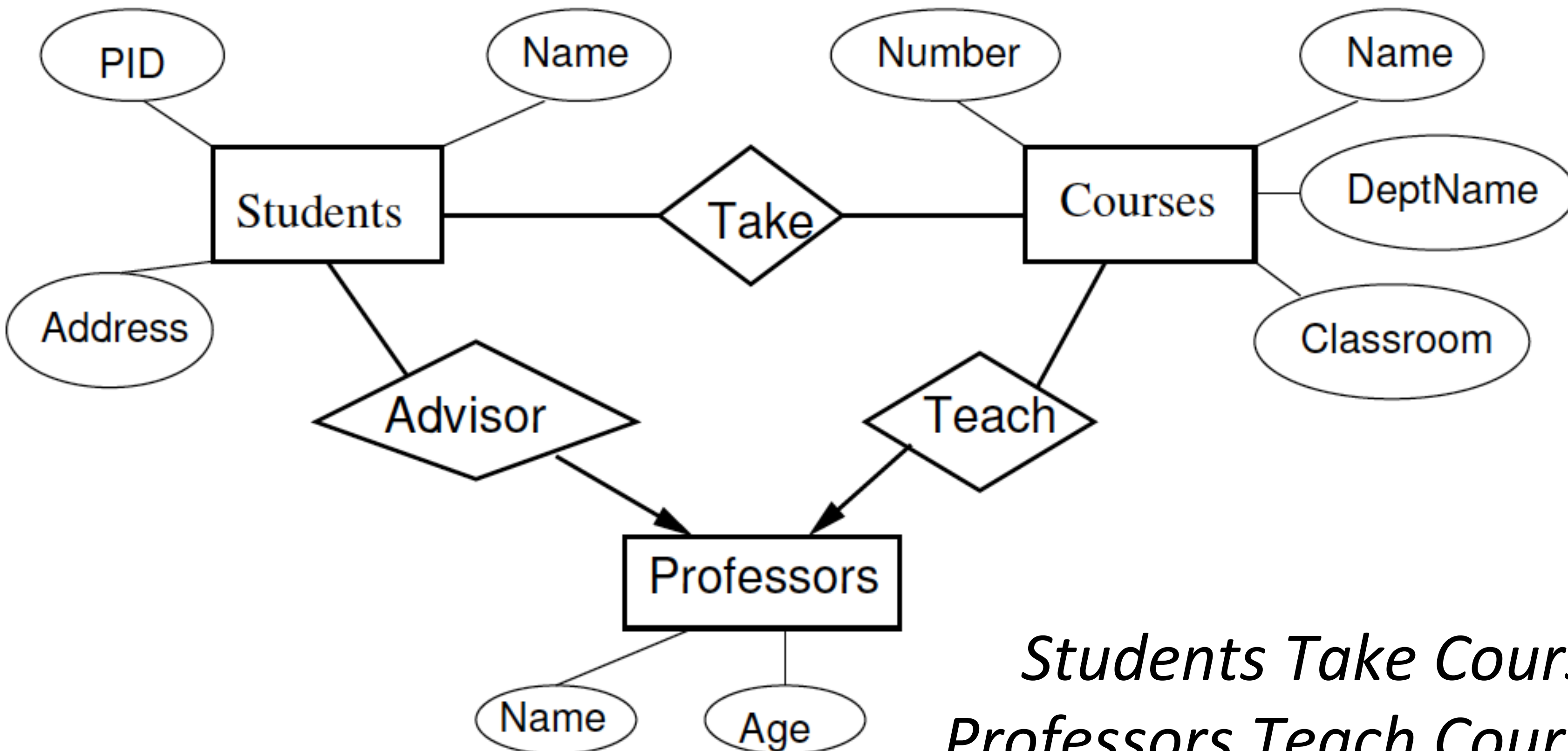
Example: Entity Sets



Relationships

- A relationship connects two or more entity sets.
- It is represented by a diamond, with lines to each of the entity sets involved.
- Don't confuse 'Relationships' with 'Relations'!

Example: Relationships



Students Take Courses
Professors Teach Courses
Professors Advise Students

Instance of an E/R Diagram

- An E/R is NOT an implementation of the DB
  – Just a notation for specifying structure

- Still useful to think of instance of an E/R Diagram === the particular data stored in a database

Instance of an Entity Set

- For each entity set, the instance stores a specific set of entities
- Each entity is a tuple containing specific values for each attribute
- Example: Instance of Entity set Students

Name	*PID*	*Address*
Hermione Grainger	HG	Gryffindor Tower
Draco Malfoy	DM	Slytherin Tower
Harry Potter	HP	Gryffindor Tower
Ron Weasley	RW	Gryffindor Tower

Instance of a Relationship

- Example: Instance of relationship Takes (no DeptName)

Student	*PID*	*Address*	*CourseName*	*Enrollment*	*Grade*
Hermione Grainger	HG	Gryffindor	Potions	∞	A-
Draco Malfoy	DM	Slytherin	Potions	∞	B
Harry Potter	HP	Gryffindor	Potions	∞	A
Ron Weasley	RW	Gryffindor	Potions	∞	C

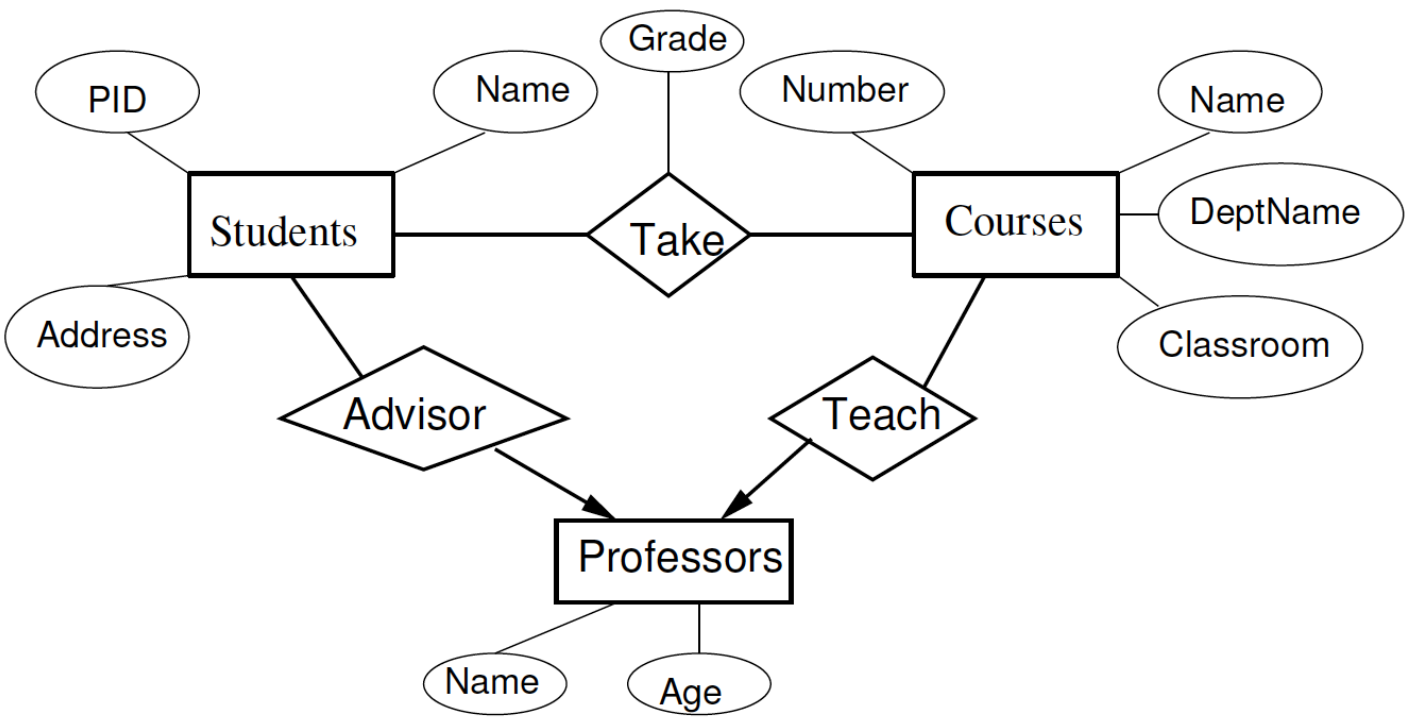
- Relationship R between (entity sets) E and F
  – Relates some *entities* in E to some *entities* in F

Instance of a Relationship

- Instance is a set of pairs of tuples (e; f) where e is in E and f is in F
  - Instance need not relate every tuple in E with every tuple in F
  - Relationship set for R: the pairs of tuples (e; f) related by R
- (Conceptually) An instance of R is simply the 'concatentation' of the attribute lists for all pairs of tuples (e; f) in the relationship set for R
- 'Tuples' in R have two components, one from E and one from F

Attributes for a Relationship

- Question: What is Grade an attribute of?
- Such an attribute is a property of the entity-pairs in the relationship



VirginiaTech

Many-Many Relationships

- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.

Many-One Relationships

- Some binary relationships are *many -one* from one entity set to another .
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.

One-One Relationships

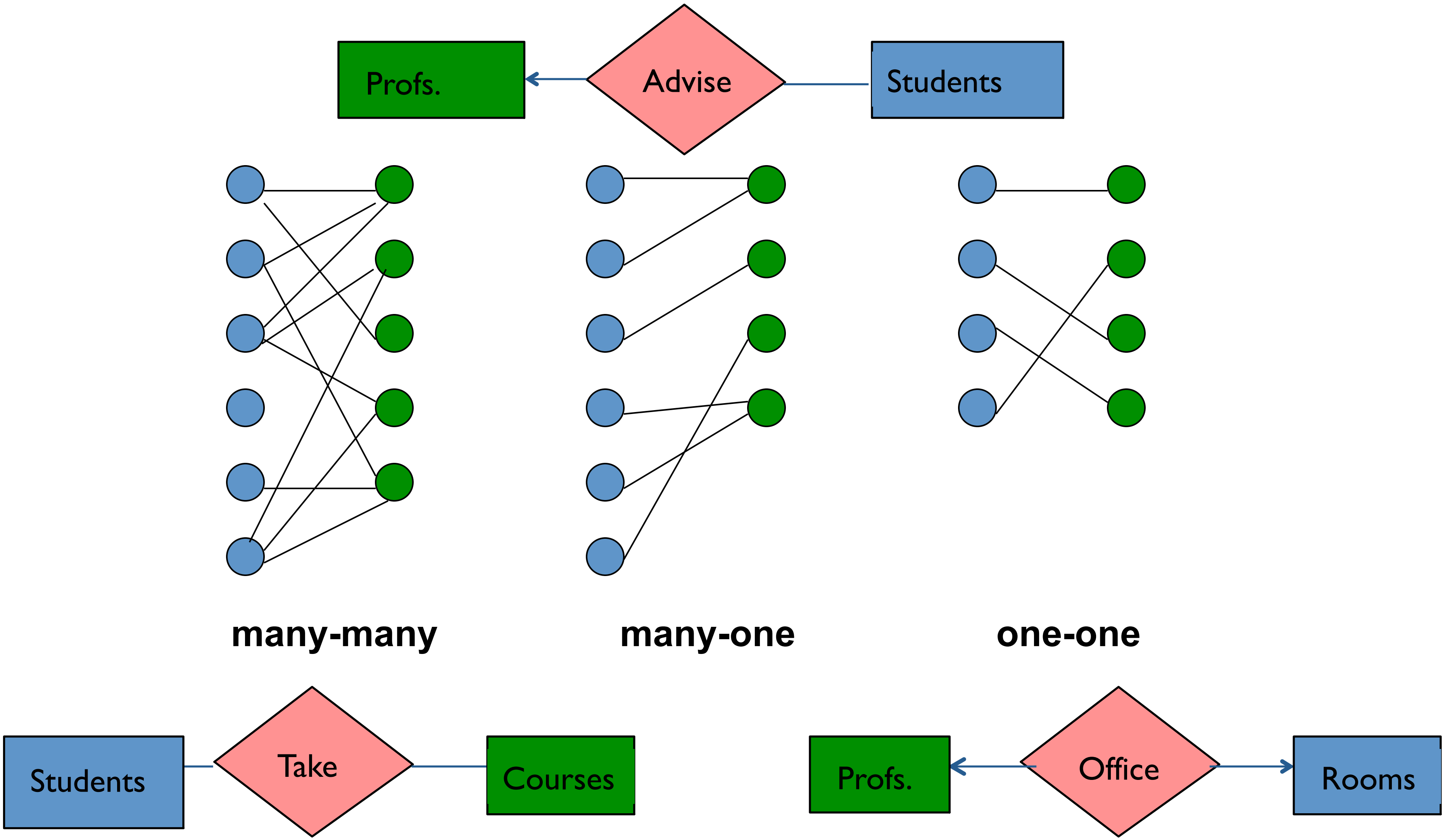
- In a one-one relationship, each entity of either entity set is related to at most one entity of the other set.

- The schema defines the multiplicity of relationships. Don't use the instances of the schema to determine multiplicity.

Representing "Multiplicity"

- Show a many-one relationship by an arrow entering the "one" side.
- Show a one-one relationship by arrows entering both entity sets.

Different kinds of relationships



Exactly one

- In some situations, we can also assert "exactly one," i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow.

VirginiaTech

Example: Exactly One

- Consider *Best-course* between *Profs* and *Courses*.
- Some courses are not the best-course of any professor, so a rounded arrow to *Profs* would be inappropriate.
- But a professor has to have a best-course