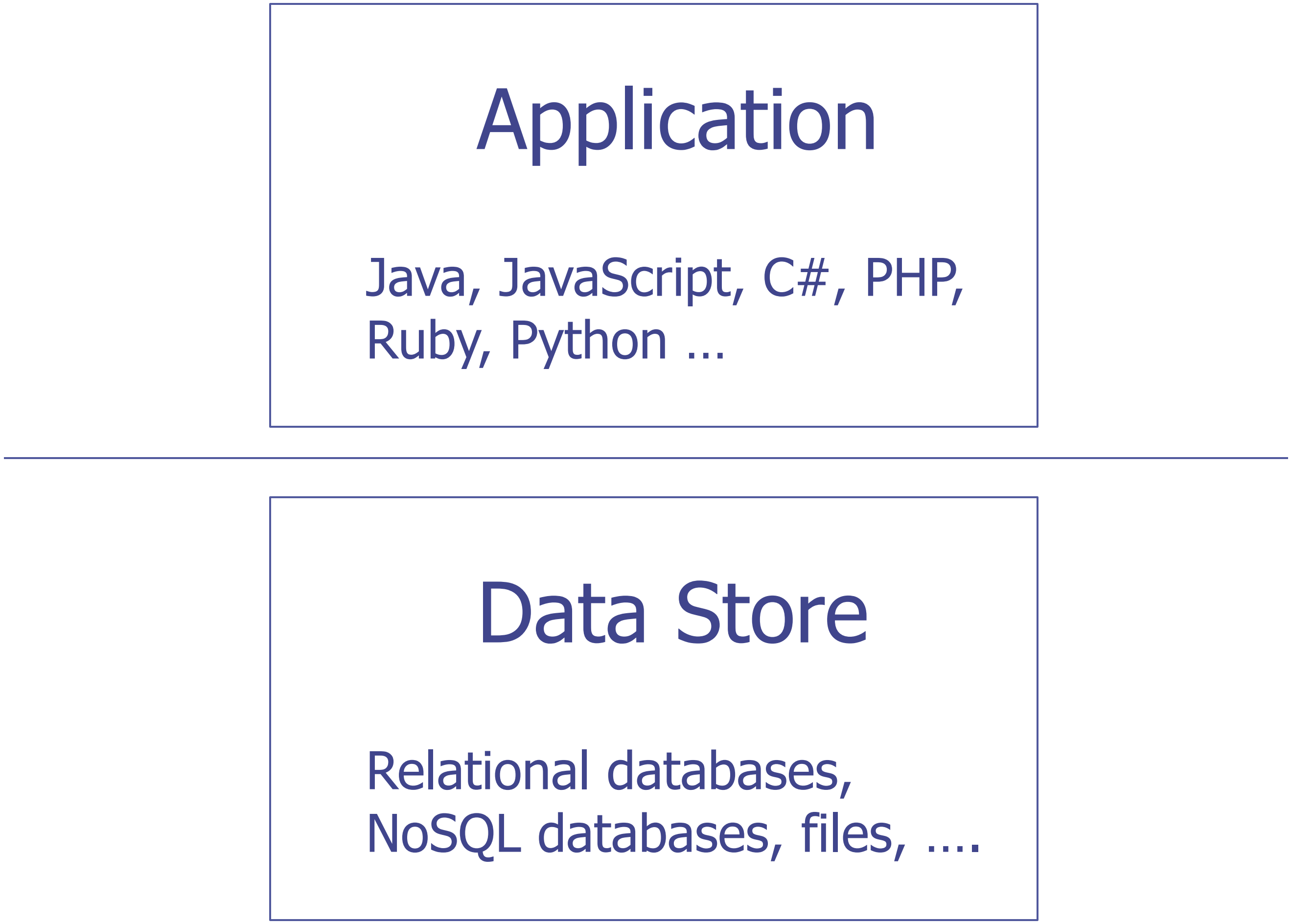
CS5220 Advanced Topics in Web Programming

Data Modeling for Relational Databases

Chengyu Sun
California State University, Los Angeles

Typical Web Application



Application

Java, JavaScript, C#, PHP, Ruby, Python …

Data Store

Relational databases, NoSQL databases, files, ….

Data Modeling

- How to represent the data in the application language, e.g. class design
- How to represent the data in the data storage, e.g. schema design

⬇

Better design leads to better code and better performance

Relational Database Terminology

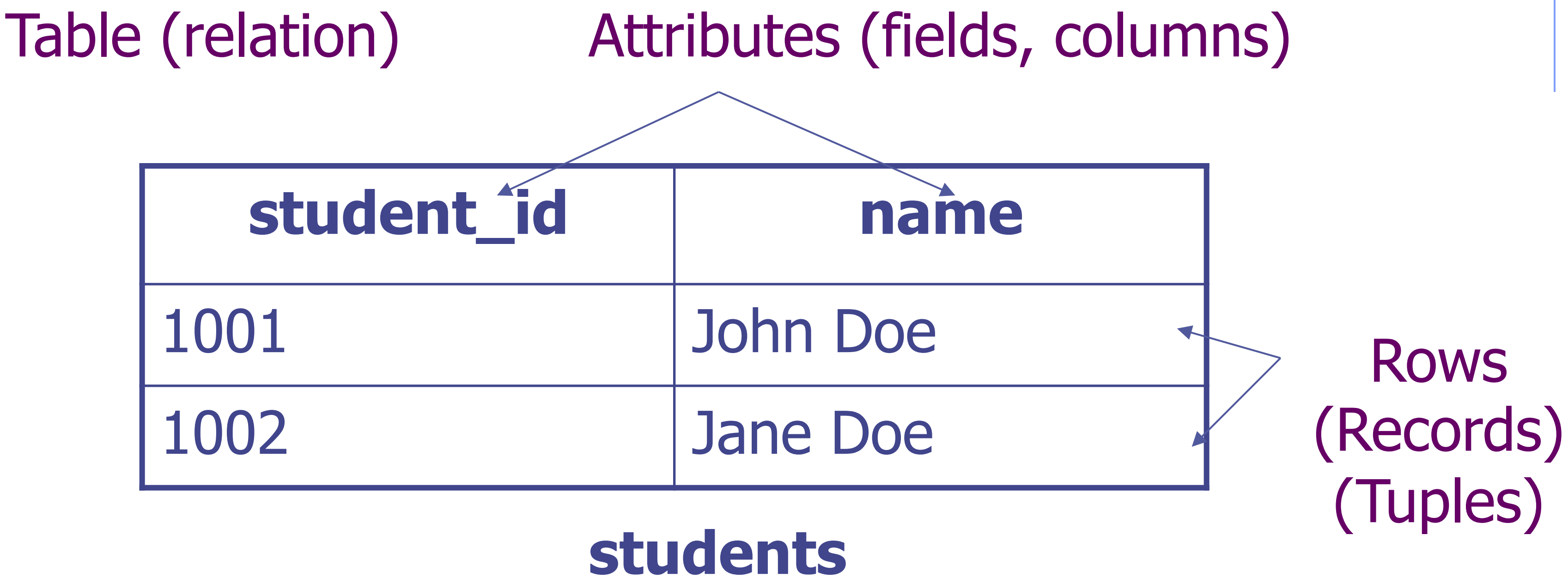


Table (relation)

Attributes (fields, columns)

student_id	name
1001	John Doe
1002	Jane Doe

students

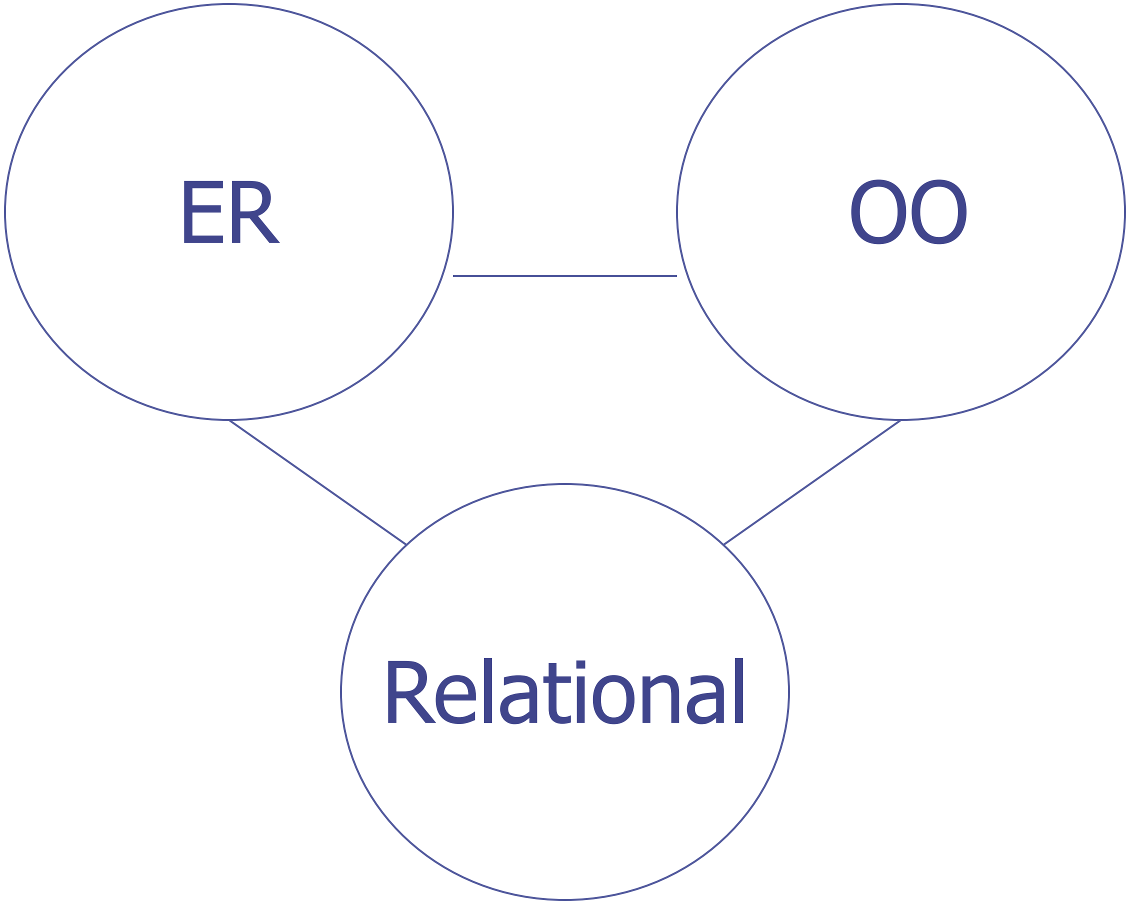
Rows
(Records)
(Tuples)

Table (relation) schema:

students(student_id, name)

Database schema: database name + table schemas

Data Modeling for Relational Database



Entity-Relationship (ER) Model

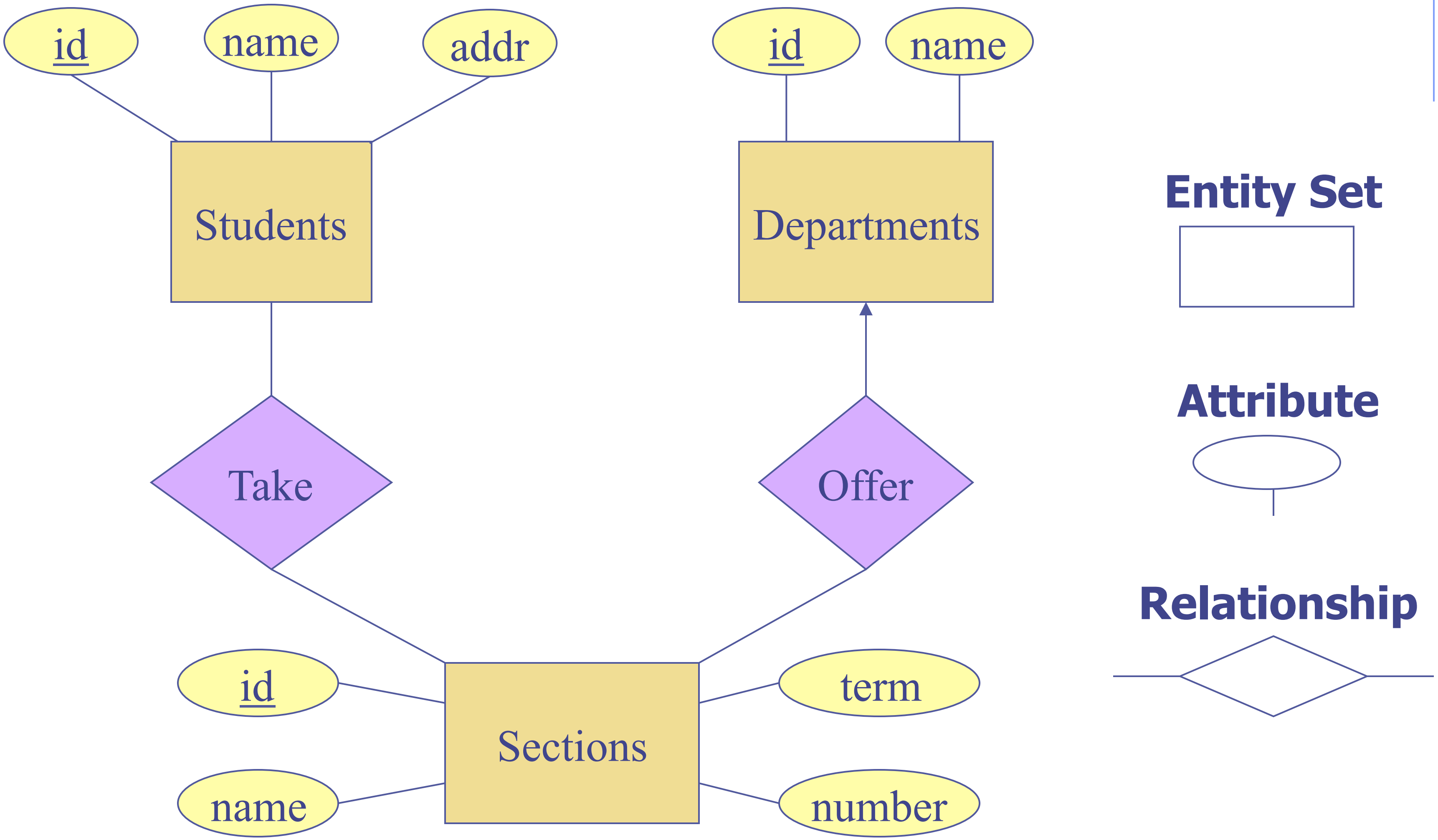
Problem → ER Model → Tables

- An *OO-like* approach
- Easily converted to relational model
- A visual representation of the design – ER Diagram

ER Example: Problem Description

- Student
  - id, name, address
- Department
  - id, name
- Class sections
  - id, name, term, section number
- Class offerings and enrollment

Example: ER Diagram



Entity Set and Attributes

- Entity Set is similar to *class* in an OO language
- Attributes are the properties of an entity set
  - Similar to the *class fields* in an OO language
  - Must have simple values like numbers or strings, i.e. *cannot be collection or composite type*

Keys

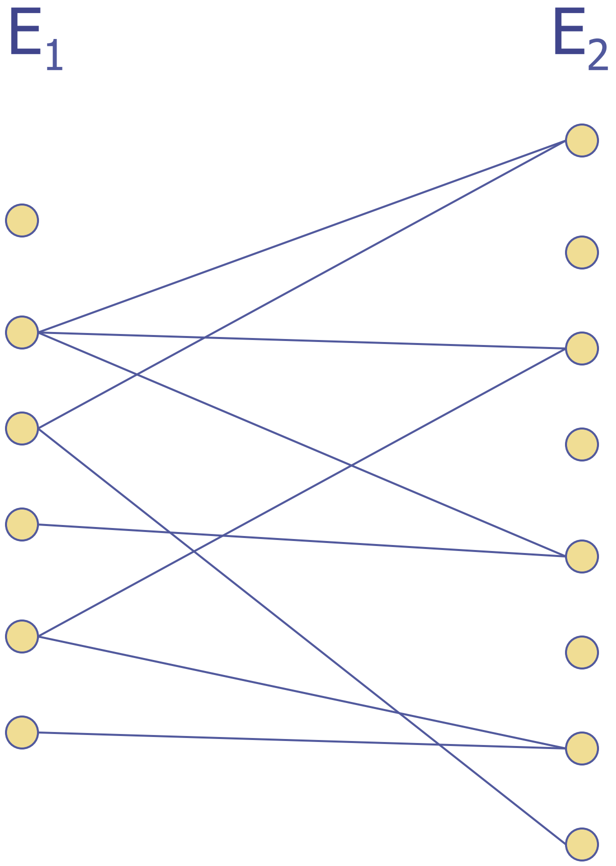
- A key is an attribute or a set of attributes that *uniquely* identify an entity in an entity set.
- Each entity set must have a key
- If there are multiple keys, choose one of them as the *primary key* (i.e. the underlined attribute(s) in an ER diagram)

Types of Relationships

- Many-to-Many
- Many-to-One / One-to-Many
- One-to-One

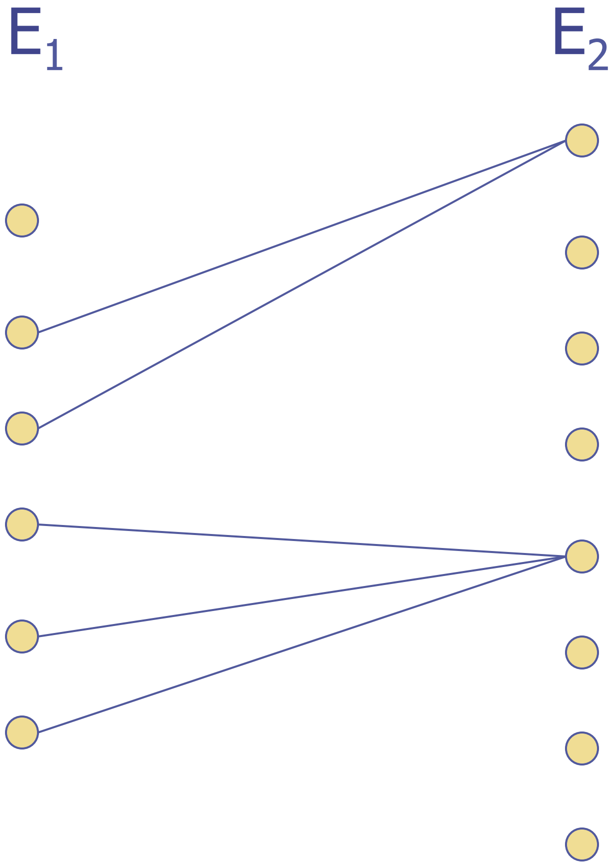
Many-to-Many Relationship

- Each entity in E_1 can be related to many entities in E_2
- Each entity in E_2 can be related to many entities in E_1



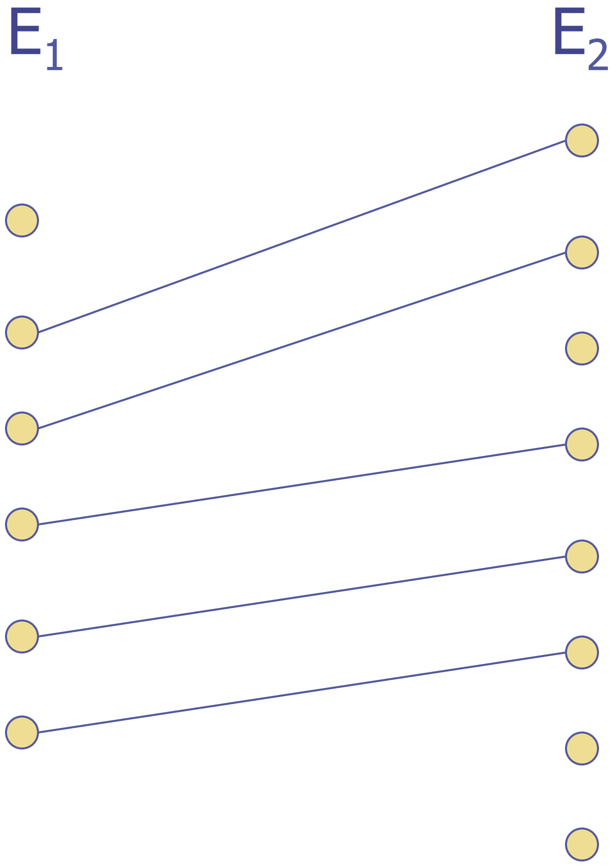
Many-to-One Relationship

- Each entity in E_1 can be related to one entities in E_2
- Each entity in E_2 can be related to many entities in E_1



One-to-One Relationship

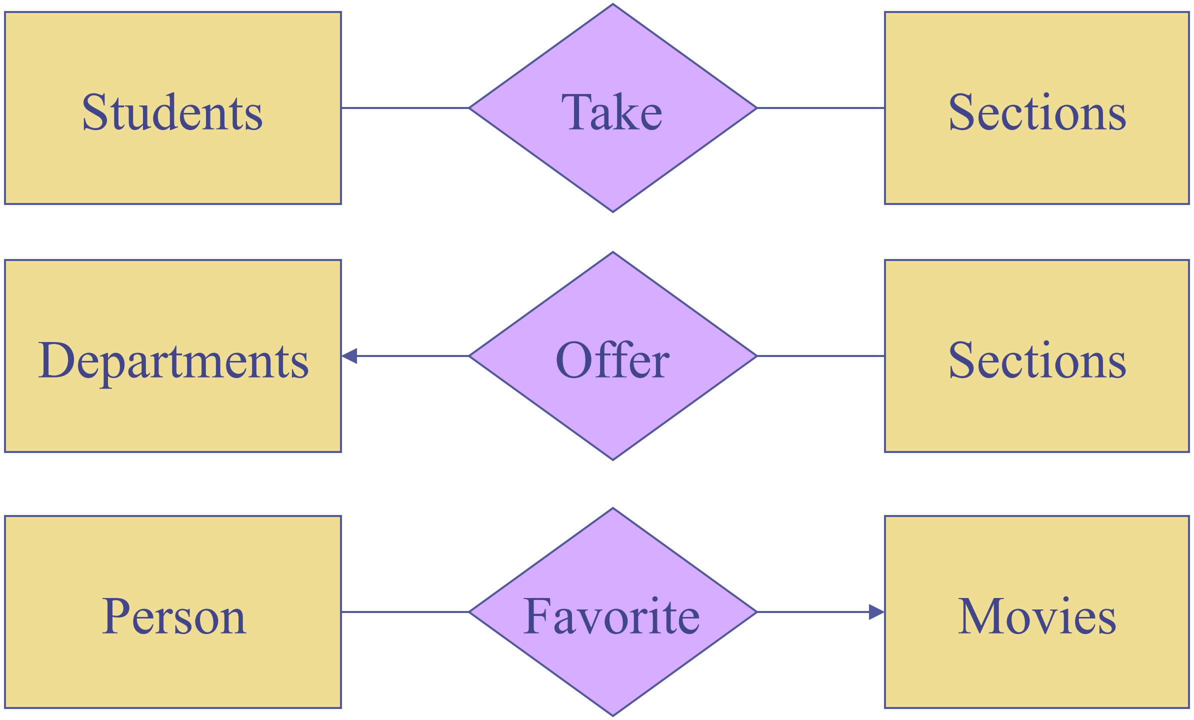
- Each entity in E_1 can be related to one entities in E_2
- Each entity in E_2 can be related to one entities in E_1



Relationship Type Examples

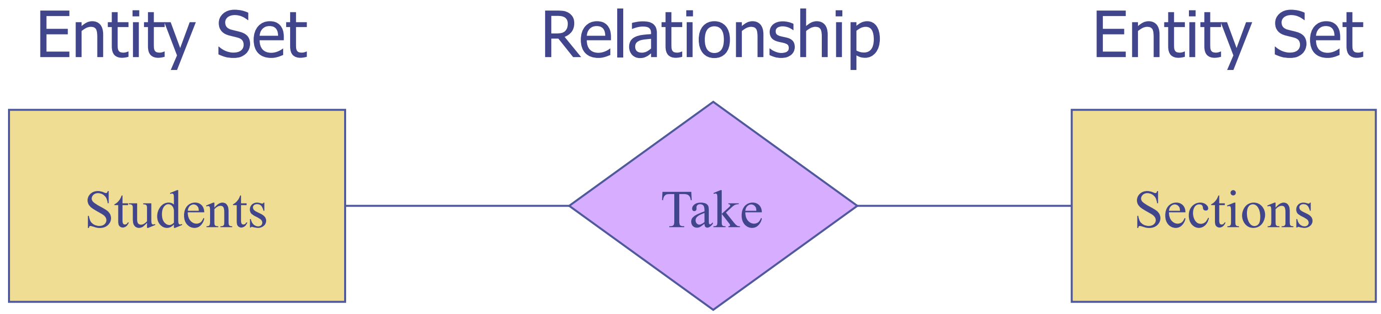
- Students and sections??
- Departments and sections??
- Person and Favorite movie??

Relationship Types in ER Diagram



- An arrow is used to indicate the “one” side

Data in a Relationship



Entity Set

Relationship

Entity Set

Students

Take

Sections

Joe
Amy
Tom
…

(Joe,CS1222)
(Amy,CS3220)
(Amy,CS4222)
(Tom,CS4222)
…

CS1222
CS3220
CS4222
…

Design Example: Department Database

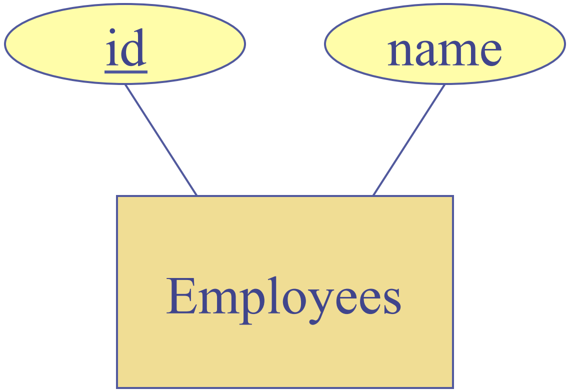
- Design a database to store the information about departments and faculty. Each department has a id and a name. Each faculty has a id and a name. A department has a number of faculty and a chairperson, who is also a faculty.

ER Design

- Step 1: identify entity sets, attributes, and relationships.
- Step 2: determine relationship types
- Step 3: complete entity sets
  - Identify/create keys
  - Add additional attributes if necessary
- Some common problems:
  - Wrong relationship types
  - Collection/composite attributes

Employees and Supervisors

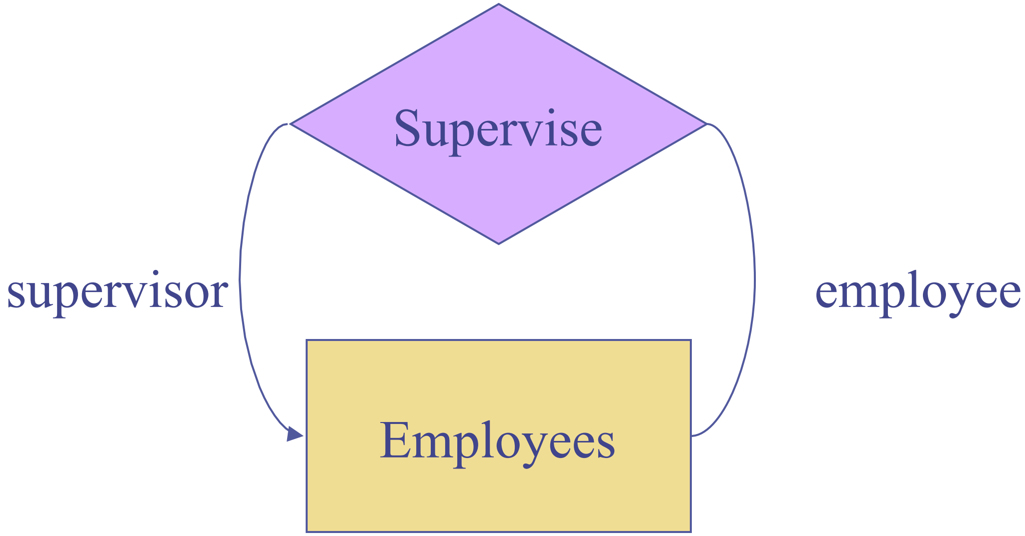
- Each employee has a supervisor
- A supervisor is an employee



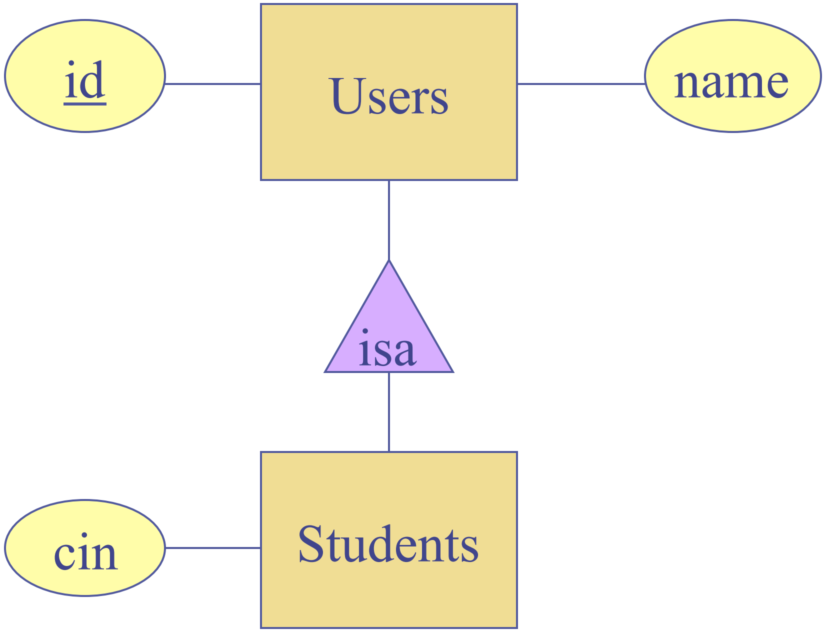
??

Roles

- An entity set may appear in the same relationship more than once.
- Label the edges with names called Roles



Subclass



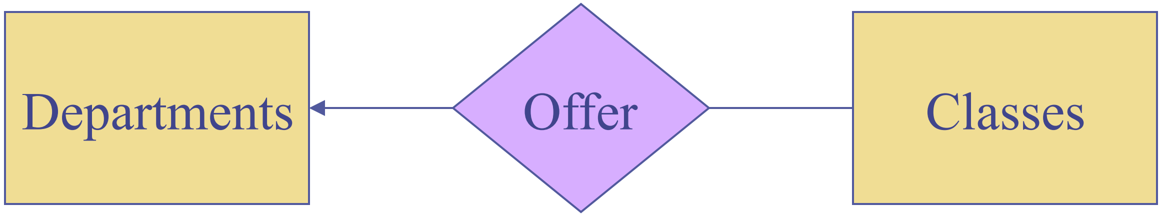
- In ER design, a subclass is only needed if it has more attributes than the superclass.

Basic Rules of ER to Relational Conversion …

- A entity set is converted to a table
  - Entity set name → table name
  - Entity set attributes → table columns
  - Entity set key → table key
- A many-to-many relationship is also converted to a table that includes the key attributes from the associated entity sets

… Basic Rules of ER to Relational Conversion

- A many-to-one relationship is converted to a foreign key column on the "many" side referencing the "one" side



Classes (id, name, term, section, department_id)

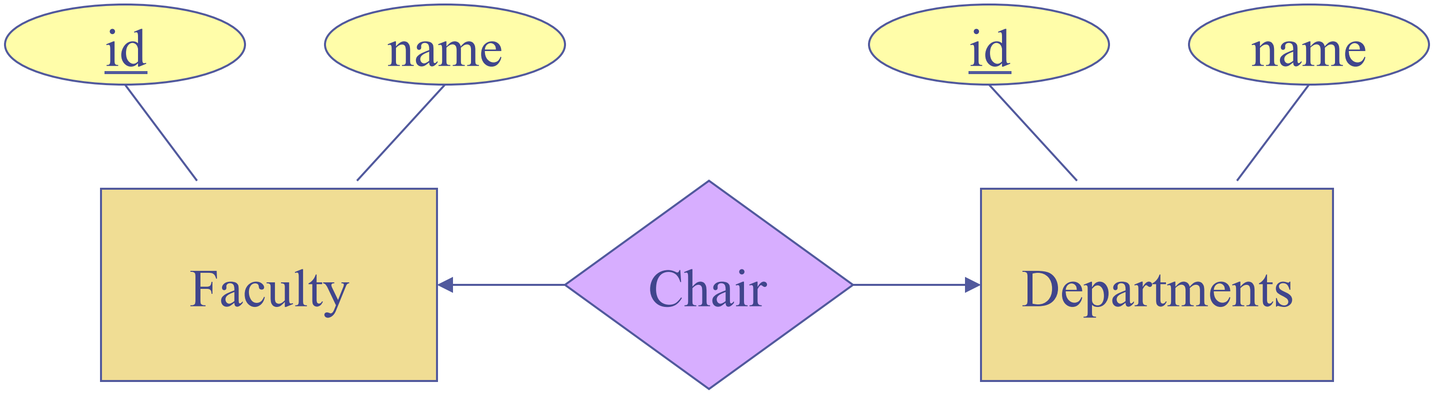
About Foreign Key

- Foreign key in relational model
  - A link (or association) between two tables – a foreign key column is like an object reference in a Java class
  - A data integrity constraint
- There is NO foreign key in ER model, *because the association is already expressed as a relationship*

Basic ER to Relational Conversion Steps

- Step 1: convert entity sets to tables
- Step 2: convert relationships
  - Many-to-many → table
  - Many-to-one → foreign key column
- Step 3: rename tables and columns when necessary

Converting One-to-One Relationship …



… Converting One-to-One Relationship

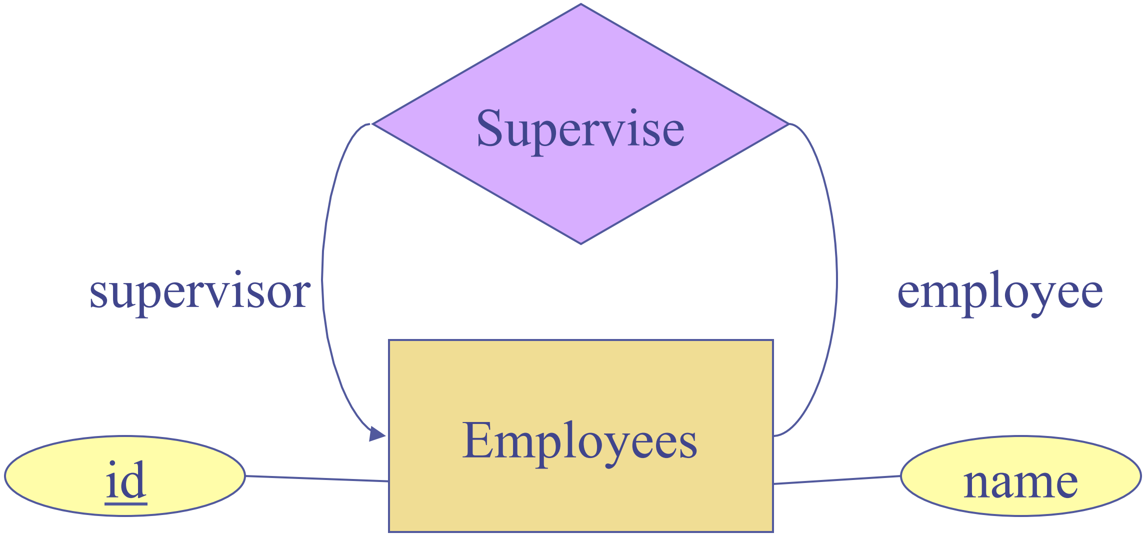
- Which one of the following is better??

Faculty(id, name, chair_of_department)
Departments(id, name)

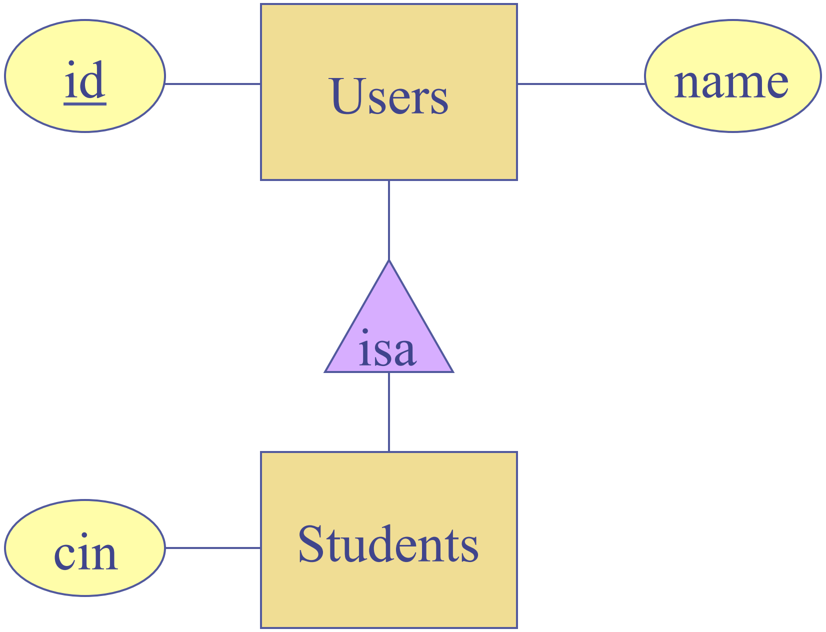
or

Faculty(id, name)
Departments(id, name, department_chair)

Converting Relationship with Roles



Converting Subclass …



… Converting Subclass

- Object-oriented approach
  - One table per concrete class
  - Each entity belongs to exact one table
- ER approach
  - One table per subclass
  - Each entity may appear in multiple tables
- NULL approach
  - One table per class hierarchy

Object-Oriented Approach

id	name
1000	John

Users

id	name	cin
1001	Jane	212345678

Students

ER Approach

id	name
1000	John
1001	Jane

Users

user_id	cin
1001	212345678

Students

NULL Approach …

id	name	cin
1000	John	NULL
1001	Jane	212345678

Users

… NULL Approach

Discriminator field

id	user_type	name	cin
1000	staff	John	NULL
1001	student	Jane	212345677

Users

Comparison of Subclass Conversion Approaches

- Constraints and data integrity
- Query performance

Q1: list all students
Q2: list all non-student users
Q3: list all users

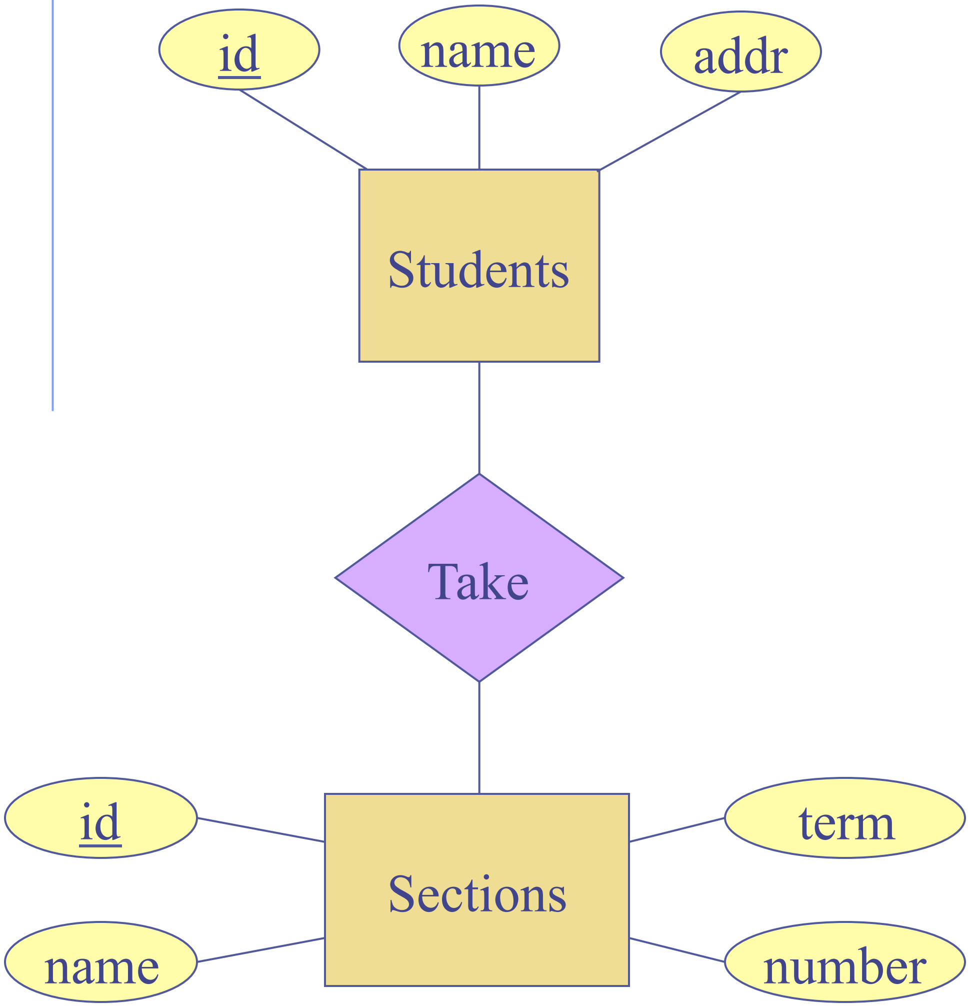
About OO Design

- The *starting point* of OO design should be creating classes that closely model after their real-world counterparts
  - Why?
  - The “English Test”
  - Modifications/optimizations may be necessary after the initial design
- Example: Project vs ProjectForm

OO Example: Problem Description

- Student
  - id, name, address
- Department
  - id, name
- Class sections
  - id, name, term, section number
- Class offerings and enrollment

OO vs. ER (I) …



```
class Student {
    Integer id;
    String name;
    String addr;
}

class Section {
    Integer id;
    String name;
    String term;
    int number;
    List<Student> students;
}
```

… OO vs ER (I)

- Classes are similar to Entity Sets, except that …
- There is no Relationship in OO. However …
- OO does allow fields of class/collection types, which are used to express relationships (a.k.a. *associations*) between classes

Which is the “Correct” OO Design? …

```
class Student {

    Integer id;
    String name;
    String address;

}
```

```
public class Section {

    Integer id;
    String name;
    String term;
    int number;

    List<Student> students;

}
```

… Which is the “Correct” OO Design? …

```
class Student {

    Integer id;
    String name;
    String address;

    List<Section> sectionsEnrolled;

}
```

```
public class Section {

    Integer id;

    String name;
    String term;
    int number;

}
```

… Which is the “Correct” OO Design?

```
class Student {

    Integer id;
    String name;
    String address;

    List<Section> sectionsEnrolled;

}
```

```
public class Section {

    Integer id;

    String name;
    String term;
    int number;

    List<Student> students;

}
```

OO vs. ER (II)

- There are different ways represent a relationship in OO design: unidirectional association and bidirectional association
- In the case of bidirectional association, the two fields in two classes are simply the two “ends” of the same relationship
- Same relationship → same database schema

Determine Relationship Type in OO Design

- Can we determine the relationship type in OO design by just looking at code?

```
class Student {

    Integer id;
    String name;
    String address;

}
```

```
public class Section {

    Integer id;
    String name;
    String term;
    int number;

    List<Student> students;

}
```

OO vs. ER (III)

- Relationship types are explicit in ER design
- Relationship types are implicit in OO design – must be determined by the *semantics* of the application

From OO to Relational

- Example: design a database to store the information about departments and faculty. Each department has a id and a name. Each faculty has a id and a name. A department has a number of faculty and a chairperson, who is also a faculty.