Recitation for CPS216, Advanced Database Systems, Spring 2003

# E/R Model & Database Design

CPS 216

Advanced Database Systems

January 17, 2003

Thanks Dr. Jun Yang for providing the slides! ☺



## Database design

- Understand the real-world domain being modeled
  - Student? Faculty? Courses? What relationships?
- Specify it using a database design model
  - Design models are especially convenient for schema design, but are not necessarily implemented by DBMS
  - Popular ones include
    - Entity/Relationship (E/R) model
    - Object Definition Language (ODL) – Covered by Jun's class
- Translate specification to the data model of DBMS
  - Relational, XML, object-oriented, etc.
- Create DBMS schema



## Entity-relationship (E/R) model

- Historically very popular
- Can think of as a "watered-down" object-oriented design model
- E/R diagrams represent designs
- Primarily a design model—not implemented by any major DBMS



## E/R basics

- Entity: a "thing," like a record or an object
  - Graduate student Junyi, Professor Jun Yang
- Entity set: a collection of things of the same type, like a relation of tuples or a class of objects
  - Student(Junyi, Zhihui, Rebecca….), Professor(Jun, Amin, Jeff…..)
  - Represented as a rectangle
- Relationship: an association among two or more entities
  - Professor Jun Yang ADVISES graduate student Junyi
- Relationship set: a set of relationships of the same type; an association among two or more entity sets
  - Professor ADVISES Student
  - Represented as a diamond



## E/R basics(cont.)

- Attributes: properties of entities or relationships, like attributes of tuples or objects
  - Entity Set
    - Student(Name, SID, Department)
  - Relationship Set
    - Advise(Professor, Student, ResearchTopic)
  - Represented as ovals



## An example E/R diagram

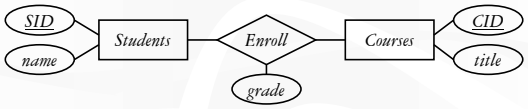
- Students enroll in courses


- A key of an entity set is represented by underlining all attributes in the key
  - A key is a set of attributes whose values can belong to at most one entity in an entity set—like a key of a relation
    - Social security number, student ID, etc
  - Very simple, right?

## Attributes of relationships



- Example: students take courses and receive grades




- Where do the grades go?
  - With *Students*?
    - But a student can have different grades for multiple courses
  - With *Courses*?
    - But a course can assign different grades for multiple students
  - With *Enroll*!

## More on relationships



- There could be multiple relationship sets between the same entity sets
  - Example: *Students Enroll Courses*; *Students TA Courses*
- In a relationship set, each relationship is uniquely identified by the entities it connects
  - Example: Between Tom and CPS216, there can be at most one *Enroll* relationship and at most one *TA* relationship

## Multiplicity of relationships



- *E* and *F*: entity sets
- Many-many: Each entity in *E* is related to 0 or more entities in *F* and vice versa
  - Example:


  - Tom enrolls courses CPS216, CPS240, etc.
  - Tom, Bart, Jeff enroll in CPS216

## Multiplicity of relationships(cont.)



- Many-one: Each entity in *E* is related to 0 or 1 entity in *F*, but each entity in *F* is related to 0 or more in *E*
  - Example:

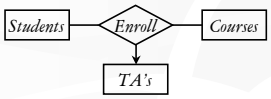

- One-one: Each entity in *E* is related to 0 or 1 entity in *F* and vice versa
  - Example:


- Notation: "One" (0 or 1) is represented by an arrow

## *N*-ary relationships



- Example: Each course has multiple TA's; each student is assigned to one TA

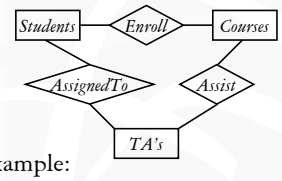



- Meaning of an arrow into *E*: Pick one entity from each other entity set; together they must be related to 0 or 1 entity in *E*
  - Tom is assigned ONE TA Lisa in CPS216

## *N*-ary versus binary relationships



- Can we model *n*-ary relationships using just binary relationships?

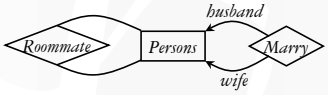



- No; for example:
  - Bart takes CPS196 and CPS114
  - Lisa TA's CPS196 and CPS114
  - Bart is assigned to Lisa in CPS196, but not in CPS114

## Roles in relationships



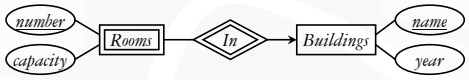
- An entity set may participate more than once in a relationship set
  - May need to label edges to distinguish roles
- Examples
  - People are married as husband and wife; label needed
  - People are roommates of each other; label not needed




## Weak entity set



- Example: rooms in buildings

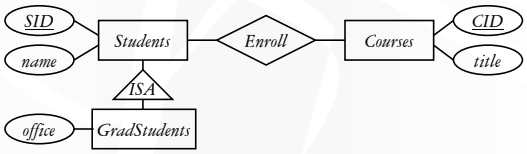



- Sometimes the key of an entity set *E* comes not completely from its own attributes, but from the keys of other (one or more) entity sets to which *E* is linked by many-one (or one-one) relationship sets
  - *E* is called a weak entity set(double rectangle)
  - Many-one (or one-one) relationship sets required(double diamonds)
    - With many-many, we would not know which entity provides the key value

## ISA relationships



- Similar to the idea of subclasses in object-oriented programming: subclass = special case, more properties, and fewer entities
  - Represented as a triangle (direction is important)
- Example: Graduate students are students, but they also have offices




## Summary of E/R concepts

- Entity sets
  - Keys
  - Weak entity sets
- Relationship sets
  - Attributes of relationships
  - Multiplicity
  - Roles
  - ISA relationships

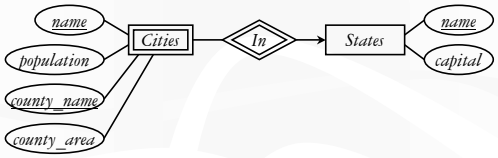
## Case study 1



- Design a database representing cities, counties, and states
  - For states, record name and capital (city)
  - For counties, record name, area, and location (state)
  - For cities, record name, population, and location (county and state)
- Assume the following:
  - Names of states are unique
  - Names of counties are only unique within a state
  - Names of cities are only unique within a county
  - A city is always located in a single county
  - A county is always located in a single state

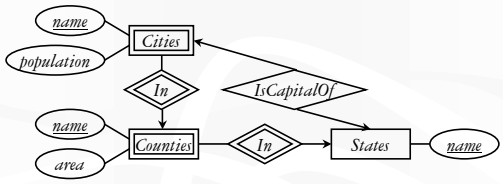
## Case study 1: first design






- County area information is repeated for every city in the county
  - Redundancy is bad (why?)
- State capital should really be a city
  - "Reference" entities through explicit relationships

## Case study 1: second design




- Technically, nothing in this design could prevent a city in state *X* from being the capital of another state *Y*, but oh well…

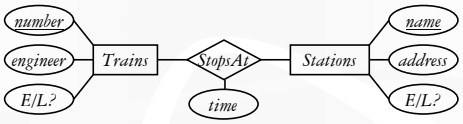


## Case study 2

- Design a database consistent with the following:
  - A station has a unique name and an address, and is either an express station or a local station
  - A train has a unique number and an engineer, and is either an express train or a local train
  - A local train can stop at any station
  - An express train only stops at express stations
  - A train can stop at a station for any number of times during a day
  - Train schedules are the same everyday



## Case study 2: first design




- Nothing in this design prevents express trains from stopping at local stations
  - ☞Capture all constraints if possible
- A train can stop at a station only once during a day
  - ☞Do not introduce constraints



## Case study 2: second design