Relational Database Design
Part I

CPS 116
Introduction to Database Systems

Announcements (September 6)

- ❖ DB2 accounts created; change your password!
  - ▪ Let me know if you have NOT received the email
- ❖ Homework #1 out today
  - ▪ Due next Thursday (September 15) at 11:59pm
  - ☞ Start early!
- ❖ Read instructions on Gradiance carefully
- ❖ Make use of office hours

Relational model: review

- ❖ A database is a collection of relations (or tables)
- ❖ Each relation has a list of attributes (or columns)
- ❖ Each attribute has a domain (or type)
- ❖ Each relation contains a set of tuples (or rows)

Keys

❖ A set of attributes K is a key for a relation R if
  ▪ In no instance of R will two different tuples agree on all attributes of K
    • That is, K is a "tuple identifier"
  ▪ No proper subset of K satisfies the above condition
    • That is, K is minimal
❖ Example: *Student* (*SID*, *name*, *age*, *GPA*)
  ▪ *SID* is a key of *Student*
  ▪ *age* is not a key (not an identifier)
  ▪ {*SID*, *name*} is not a key (not minimal)

Schema vs. data

Student

SID	name	age	GPA
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3

❖ Is *name* a key of *Student*?
  ▪ Yes? Seems reasonable for this instance
  ▪ No! Student names are not unique in general
❖ Key declarations are part of the schema

More examples of keys

❖ *Enroll* (*SID*, *CID*)

❖ *Address* (*street_address*, *city*, *state*, *zip*)

Usage of keys

❖ More constraints on data, fewer mistakes
❖ Look up a row by its key value
  ▪ Many selection conditions are "key = value"
❖ "Pointers"
  ▪ Example: *Enroll* (*SID*, *CID*)
    • *SID* is a key of *Student*
    • *CID* is a key of *Course*
    • An *Enroll* tuple "links" a *Student* tuple with a *Course* tuple
  ▪ Many join conditions are "key = key value stored in another table"

Database design

❖ Understand the real-world domain being modeled
❖ Specify it using a database design model
  ▪ Design models are intuitive and convenient for schema design, but are not necessarily implemented by DBMS
  ▪ Popular ones include
    • Entity/Relationship (E/R) model
    • Object Definition Language (ODL)
    • UML (Unified Modeling Language)
❖ Translate specification to the data model of DBMS
  ▪ Relational, XML, object-oriented, etc.
❖ Create DBMS schema

Entity-relationship (E/R) model

❖ Historically and still very popular
❖ Can think of as a "watered-down" object-oriented design model
❖ Primarily a design model—not directly implemented by DBMS
❖ Designs represented by E/R diagrams
  ▪ We use the style of E/R diagram covered by GMUW; there are other styles/extensions
  ▪ Very similar to UML diagrams

E/R basics

- ❖ Entity: a "thing," like a record or an object
- ❖ Entity set: a collection of things of the same type, like a relation of tuples or a class of objects
  - ▪ Represented as a rectangle
- ❖ Relationship: an association among entities
- ❖ Relationship set: a set of relationships of the same type (associations among same entity sets)
  - ▪ Represented as a diamond
- ❖ Attributes: properties of entities or relationships, like attributes of tuples or objects
  - ▪ 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

---

Attributes of relationships

- ❖ Example: students take courses and receive grades
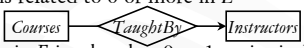

- ❖ Where do the grades go?

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 Bart and CPS116, there can be at most one *Enroll* relationship and at most one *TA* relationship
  ☞ What if Bart took CPS116 twice and got two different grades?

## 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:

| Students | — ◇ Enroll ◇ — | Courses |

❖ 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:

| Courses | — ◇ TaughtBy ◇ → | Instructors |

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

| Students | ← ◇ Own ◇ → | AcpubAccounts |

❖ "One" (0 or 1) is represented by an arrow ——→
❖ "Exactly one" is represented by a rounded arrow ——⟩

N-ary relationships

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

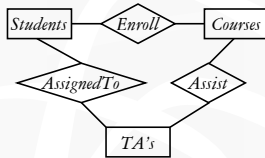| Students | — ◇ Enroll ◇ — | Courses |
| | ↓ | |
| | TA's | |
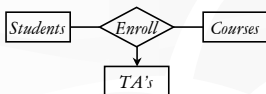
❖ Meaning of an arrow into *E*: Pick one entity from each of the other entity sets; together they must be related to either 0 or 1 entity in *E*

N-ary versus binary relationships

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

Students — ⟨Enroll⟩ — Courses

⟨AssignedTo⟩   ⟨Assist⟩

TA's

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

husband

⟨Roommate⟩ — Persons — ⟨Marry⟩

wife
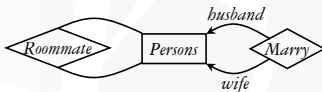
Weak entity sets

❖ 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
  - Represented by double rectangle
- Many-one (or one-one) relationship sets required
  - Represented by double diamonds
  - With many-many, we would not know which entity provides the key value
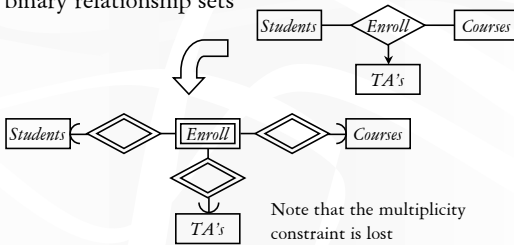
Weak entity set examples

❖ Seats in rooms in buildings
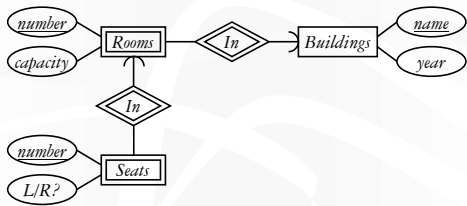
Modeling *n*-ary relationships

❖ An *n*-ary relationship set can be replaced by a weak entity set (called a connecting entity set) and *n* binary relationship sets

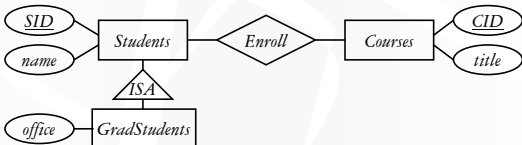Note that the multiplicity constraint is lost

ISA relationships

❖ Similar to the idea of subclasses in object-oriented programming: subclass = special case, fewer entities, and possibly more properties
  ▪ 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
  - Binary versus *N*-ary relationships
    - Modeling *N*-ary relationships with weak entity sets and binary relationships
  - ISA relationships
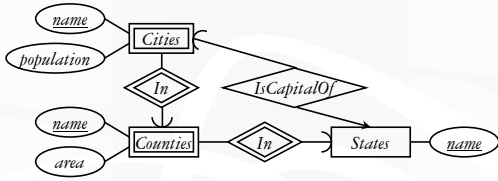

Case study 1

- ❖ 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
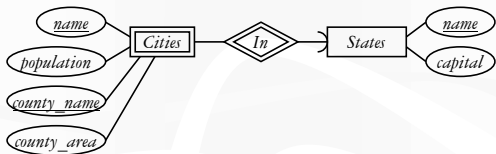
Case study 1: second design

name
population
Cities
In
IsCapitalOf
name
area
Counties
In
States
name

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
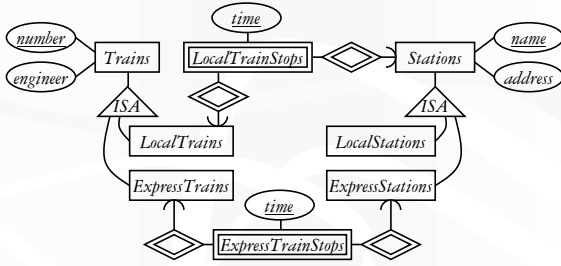- Train schedules are the same everyday

Case study 2: first design

number
engineer
E/L?
Trains
StopsAt
time
Stations
name
address
E/L?
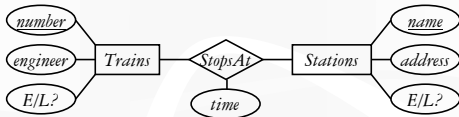
Case study 2: second design

Is the extra complexity worth it?