Entity-Relationship Models: Good Design and Constraints

T. M. Murali

September 30, 2009
Guidelines

- Be faithful to the specification of the application.
- Avoid redundancy.
- Keep the entities and relationship simple.
- Select the right relationships.
- Select the right type of element.
Be Faithful

- Do not use meaningless or unnecessary attributes.
- Define the multiplicity of a relationship appropriately.
  - What is the multiplicity of the relationship *Take* between *Students* and *Courses*?
  - What is the multiplicity of the relationship *Teach* between *Professors* and *Courses*?
Avoid Redundancy

- Redundancy occurs when we express the same fact in two or more ways.
- Redundancy wastes space.
- Redundancy can lead to inconsistency if we change one instance but not the other.
Select the Right Relationships

▶ Do not add unnecessary relationships.
▶ It may be possible to deduce one relationship from another.
Select the Right Relationships

- Do not add unnecessary relationships.
- It may be possible to deduce one relationship from another.
- Do we need the relationship *Instruct* between *Professors* and *Students*?

No. We can deduce this relationship from *Take* and *Teach*.

- Do we need the relationships *Take* and *Teach*?
  - Yes. Why?
Select the Right Relationships

- Do not add unnecessary relationships.
- It may be possible to deduce one relationship from another.
- Do we need the relationship *Instruct* between *Professors* and *Students*?
  - No. We can deduce this relationship from *Take* and *Teach*. 
Select the Right Relationships

- Do not add unnecessary relationships.
- It may be possible to deduce one relationship from another.
- Do we need the relationship *Instruct* between *Professors* and *Students*?
  - No. We can deduce this relationship from *Take* and *Teach*.
- Do we need the relationships *Take* and *Teach*?
Select the Right Relationships

- Do not add unnecessary relationships.
- It may be possible to deduce one relationship from another.
- Do we need the relationship *Instruct* between *Professors* and *Students*?
  - No. We can deduce this relationship from *Take* and *Teach*.
- Do we need the relationships *Take* and *Teach*?
  - Yes. Why?
Select the Right Type of Element

- Attribute or Entity or Relationship?
Select the Right Type of Element

- Attribute or Entity or Relationship?
- Can we make *Professor* an attribute of *Courses* and remove the relationship *Teach*?

![Entity-Relationship Diagram]

- Name
- Address
- Name
- Classroom
- Enrollment
- Name
- Age
- Name
- PID
- PID
- DeptName

T. M. Murali September 30, 2009 CS 4604: E/R Design and Constraints
Select the Right Type of Element

- Attribute or Entity or Relationship?
- Can we make Professor an attribute of Courses and remove the relationship Teach?
- What if we add the relationship Evaluation?

Diagram:

- Students
  - Take
    - Evaluation
      - Teach
        - Professors
          - Name
          - Age
          - Professor
          - Classroom
          - Enrollment
          - Professor
          - Name
          - Address
          - PID
          - DeptName
          - Courses
            - Name
            - Address
            - PID
            - Number
          - Take
          - Teach
          - Evaluation
          - Classroom
          - Enrollment

T. M. Murali September 30, 2009 CS 4604: E/R Design and Constraints
Select the Right Type of Element

- Attribute or Entity or Relationship?
- Can we make Professor an attribute of Courses and remove the relationship Teach?
- What if we add the relationship Evaluation?
- What if we add the relationship Research signifying a research project the student is working on with a professor?
Converting an Entity Set into an Attribute

If an entity set $E$ satisfies the following properties:

1. All relationships involving $E$ have arrows entering $E$.
2. The attributes of $E$ collectively identify an entity (i.e., no attribute depends on another).
3. No relationship involves $E$ more than once

then we can replace $E$ as follows:

1. If there is a many-one relationship $R$ from an entity set $F$ to $E$, remove $R$ and make the attributes of $E$ be attributes of $F$.
2. If there is a multiway relationship $R$ with an arrow to $E$, make $E$’s attributes be new attributes of $R$ and remove the arrow from $R$ to $E$. 
Types of Constraints

- **Keys** are attributes or sets of attributes that uniquely identify an entity within its entity set.

- **Single-value constraints** require that a value be unique in certain contexts.

- **Referential integrity constraints** require that a value referred to actually exists in the database.

- **Domain constraints** specify what set of values an attribute can take. Read the textbook.

- **General constraints** are arbitrary constraints that should hold in the database.

- **Constraints are part of the schema of a database.**
A key for an entity set $E$ is a set $K$ of one or more attributes such that given any two entities $e_1$ and $e_2$ in $E$, $e_1$ and $e_2$ cannot have identical values for all the attributes in $K$.

$E$ can have multiple keys. We designate one as the primary key.

In an isa-hierarchy, the root entity set must have all the attributes needed for a key.

In an E/R diagram, underline the attributes that form the primary key.
Examples of Keys

- **Students**: Name, Address, PID
- **Professors**: Name, Age
- **Courses**: Name, Classroom, DeptName, Number
- **Take**: Evaluation
- **Teach**: Research

Diagram:

- Students → Take → Courses
- Professors → Take → Courses
- Students → Evaluation → Professors
- Professors → Teach → Research
Examples of Keys
Single-Value Constraints

- There is at most one value in a given context.

1. Each attribute of an entity set has a single value.
   - If the value is missing, we can invent a “null” value.
   - E/R models cannot represent the requirement that an attribute cannot have a null value.

2. A many-one relationship implies a single value constraint.
**Referential Integrity Constraint**

- Asserts that exactly one value exists in a given context.
- Usually used in the context of relationships.
- Example: Many-one *Advises* relationship between *Students* and *Professors*.
  - Many-one requirement says that no student may have more than one advising professor.
  - Referential integrity constraint says that each student must have exactly one advising professor *and* that professor must be present in the database.

- If $R$ is a (many-to-one or one-to-one) relationship from $E$ to $F$, we use a rounded arrowhead pointing to $F$ to indicate that we require that the entity in $F$ related by $R$ to an entity in $E$ must exist.
Example of Referential Integrity

- Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson).
- Each chairperson can be the head of at most one department and this department must exist in the database.
Example of Referential Integrity

▸ Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson).

▸ Each chairperson can be the head of at most one department and this department must exist in the database.

▸ Where do we put the arrows?
Example of Referential Integrity

- Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson).
- Each chairperson can be the head of at most one department and this department must exist in the database.
- Where do we put the arrows?
Enforcing Referential Integrity Constraints

- We forbid the deletion of a referenced entity (e.g., a professor) until the professor advises no students.
- We require that if we delete a referenced entity, we delete all entities that reference it.
- When we insert a student entity, we must specify an existing professor entity connected to the student by the \textit{Advises} relationship.
Weak Entity Sets

- A *weak entity set* is an entity set whose key contains attributes from one or more other entity sets.
- It is possible that all attributes in a weak entity set’s key come from other entity sets.
- Primary causes for weak entity sets:
  - Hierarchy of entity sets (not caused by inheritance).
  - Conversion of multiway into binary relationships. (Read Example 4.22 on page 153 of the textbook.)
Example of Weak Entity Set

- Each department teaches multiple courses. Each course has a number. What is the key for the entity set *Courses*?
Each department teaches multiple courses. Each course has a number. What is the key for the entity set *Courses*?
Finding the Key for a Weak Entity Set

- *E* is a *weak entity set* if its key consists of
  1. Zero or more of its own attributes and
  2. Key attributes from *supporting relationships* for *E*.

- A relationship *R* from a weak entity set *E* to *F* is *supporting* if
  1. *R* is a binary, many-one relationship from *E* to *F*,
  2. *R* has referential integrity from *E* to *F*.

- How does *F* help *E*?
  1. *F* supplies its key attributes to define *E*’s key.
  2. If *F* is itself a weak entity set, some of its key attributes come from entity sets to which *F* is connected by supporting relationships.

- Representation in the E/R diagram.
  - Weak entity set: rectangle with a double border.
  - Supporting relationship: diamond with a double border.