Advanced ER Concepts

Orlando Karam
okaram@spsu.edu

SOUTHERN POLYTECHNIC STATE UNIVERSITY

By now you should be familiar with most of the basic ER modeling concepts; you know a lot about entities, relationships and their attributes; these concepts are enough to model many situations; however, there are several more complicated concepts that are necessary for other situations.

Introduction

- You now know the basics
  - –Entities
  - –Relationships
  - –Attributes
- There are some more complicated (but often used) concepts
  - –Weak entities
  - –Associative entities
  - –Ternary Relationships

2

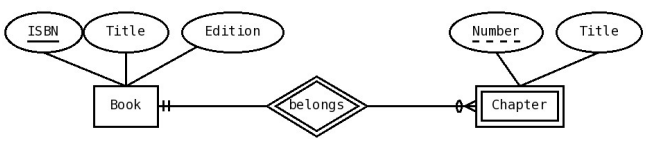
The first one is the concept of a weak entity; an entity type that cannot be identified by itself, but depends on another entity for identification.

A slightly more complex concept is that of an associative entity, which is something that starts up as a relationship, but we transform it into an entity so it can have other relationships.

The final concept is relationships of a degree higher than two, of which the most common ones are ternary relationships, which associate three entities at the same time.

Weak Entities

- Don't have an identifier (identification-dependent)
- Depend on another entity for identification
- Double lines for weak entity and identifying relationship
- Discriminator is dash-underlined



3

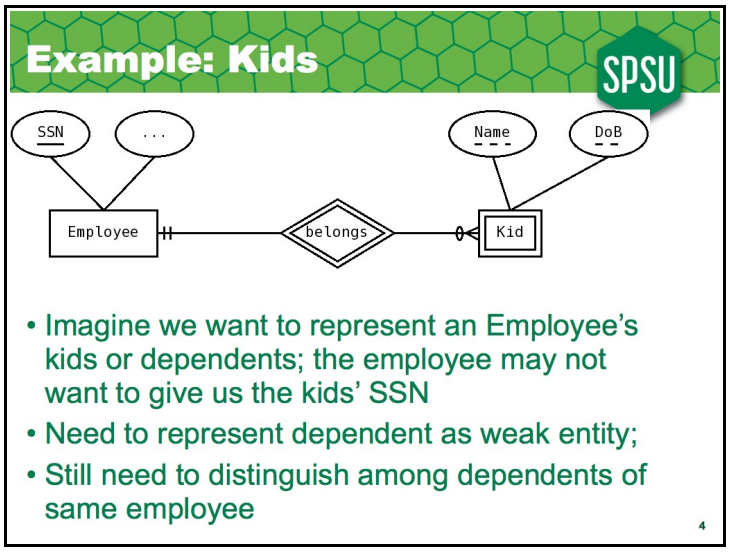
Weak entities are those that do not have an identifier, and they depend on another entity for its identification; for example, if we were representing book chapters, we may not have a way to identify individual chapters independently of a book; that is, we identify a chapter as, say, chapter 1 of 'Intro to Databases', we need to tie it to a book, and we use attributes of the chapter (in this case a number) to distinguish among chapters of the same book.

We use double lines for marking a weak entities, and double lines also for the identifying relationship (since weak entities

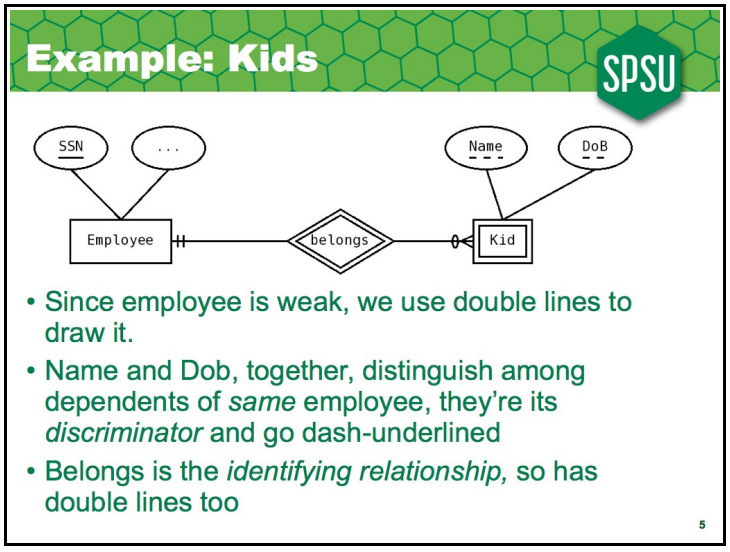
may be related to other entities besides their identifying entity); we use dash-underlining instead of full underlining to mark the *discriminator* of a weak entity, that is, the set of attributes that uniquely identify entities that depend on the same strong entity.

So on the diagram, we represent chapter as a weak entity, and we draw it with double lines; we mark number as its discriminator, or weak identifier; this means no two chapters of *the same book* will have the same number; the same book cannot have two 'Chapter 1' s, but chapters from different books may have the same number; most books will have a chapter 1. Notice that book is a regular, strong entity, with ISBN as its identifier; we draw 'belongs' with double lines, since it is the *identifying* relationship, it is the relationship that gives identity to the chapters; although this diagram doesn't show it, weak entities may participate in other relationships, and so we need to mark the identifying relationship.

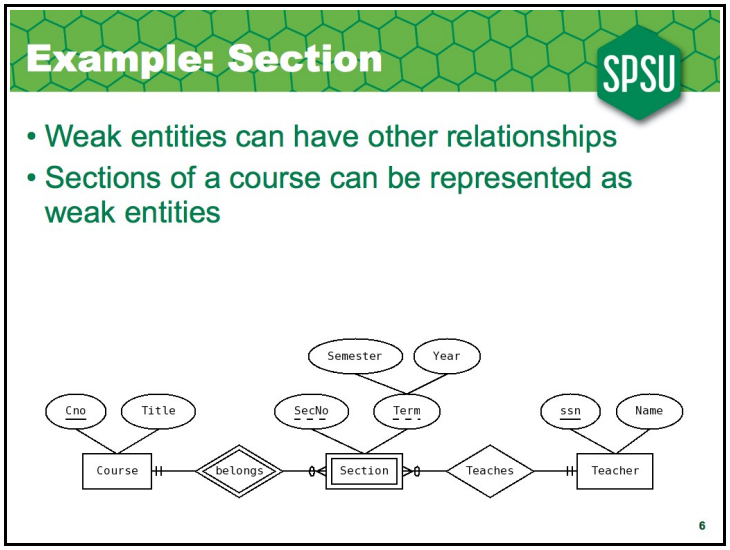
Now let's look at another example; imagine that the Human Resources department wants to keep information about an employee's kids, to, say, send them a card on their birthday or such; we could ask the employee to provide us with their kids SSN, but they may not want to; so we may need to model 'Kid' as a weak entity, depending on employee. We may not have anything to distinguish among *all* kids, but we still need to figure out a way to distinguish among kids of the *same* employee; we can assume that the name *and* date of birth, together, are unique for kids of the same employee.



So now lets look at the mechanics; We do Employee normally; Kid is a weak entity, so we draw it with double lines; Since Name and date-of-birth *together* form the discriminator, we draw them dash-underlined; they distinguish among kids of the same employee; Finally, since each Kid depends on the Employee it belongs to for its identity, we draw 'belongs' with double lines; it is the *identifying* relationship.



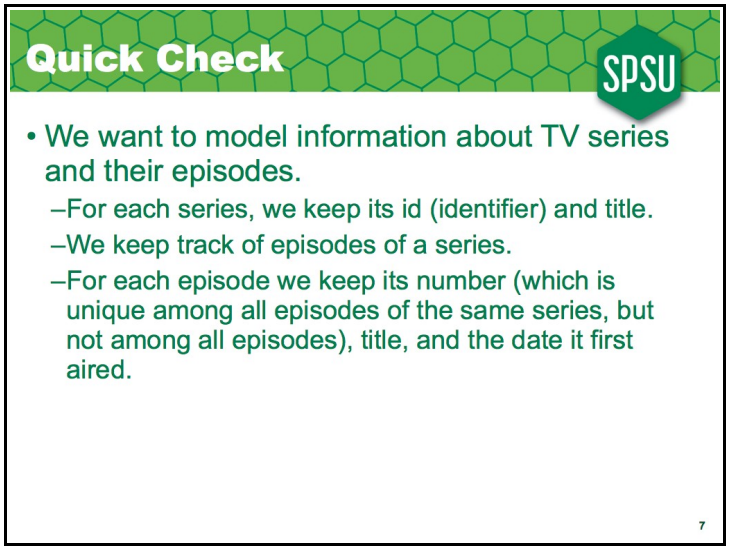
Now let's look at another example, which illustrates that weak entities participate in other relationships, besides their identifying one.



We want to model information about courses in a university; normally, the same course will be offered many different times; we call each of those offerings a section. Withing a given term, no two sections *of the same course* can have the same number; We will also keep track of which teachers teach each one of the sections.

So, we model Course as a strong entity, with Cno (the course number) as its identifier, and another field called title; We then add 'Section' as a weak entity; a section is identified with the course it belongs to, so 'belongs' is its identifying relationship; the section's number and term, *together* identify sections of the same course; finally, we add another strong entity, Teacher, with ssn and name, and another relationship, teaches, between section and teacher, representing that that teacher teaches that section. Notice that teaches is NOT and identifying relationship, since a section does NOT depend on its teacher for its identity; that is, it is section 1 of intro to databases, but not section 1 of Julia Jones.

OK, so now you try it. No peeking …

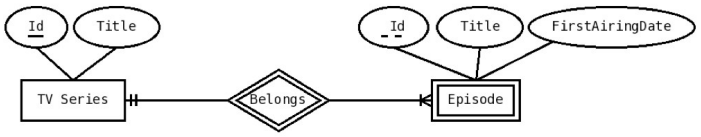


And here's the solution. Episode is a weak entity, depending on series, and with id as its discriminator.

Solution

SPSU

- We want to model information about TV series and their episodes.
  - –For each series, we keep its id (identifier) and title.
  - –We keep track of episodes of a series.
  - –For each episode we keep its number (which is unique among all episodes of the same series, but not among all episodes), title, and the date it first aired.



8

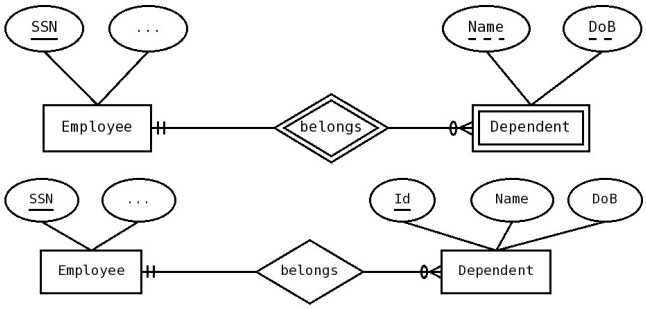
Sometimes, we don't want to deal with weak entities, and can transform a weak entity into a strong one by just inventing an identifier for it. The values for this identifier would have to be generated by the system, but most identifiers are generated by *some* system anyway.

Once we add an identifier, the entity is not special any more; we may be losing some information (that the combination of employee, kid's name and kid's date-of-birth is unique), which we would normally keep in a separate document, or as part of our data dictionary.

Can make weak strong

SPSU

- Sometimes, it is useful to transform a weak entity into a strong entity
- We can just create an identifier for it



9

Now we need to consider, should we convert weak entities into strong ones ? Well, the answer depends on the particular situation; strong entities are conceptually simpler, and will lead to simpler keys in the relational model; however, we need to add an extra attribute, generate its values, and keep track of the constraints separately; also, we need to think about what is natural for our customers, we normally want our model to be as close as possible to their mental model of the situation.

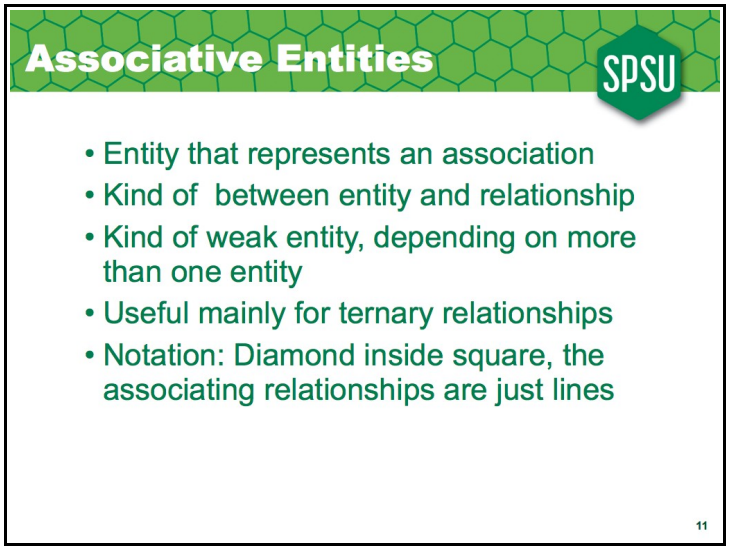
Should we make strong ?

SPSU

- Advantages
  - –Conceptual simplicity (weak entities are harder to understand)
  - –Eventually, simpler keys (for relational model)
- Disadvantages
  - –Extra attribute
  - –Extra work (generating it)
  - –Need to keep track of the constraints separately (the discriminator is unique among weak entities related to the same strong entity)

10

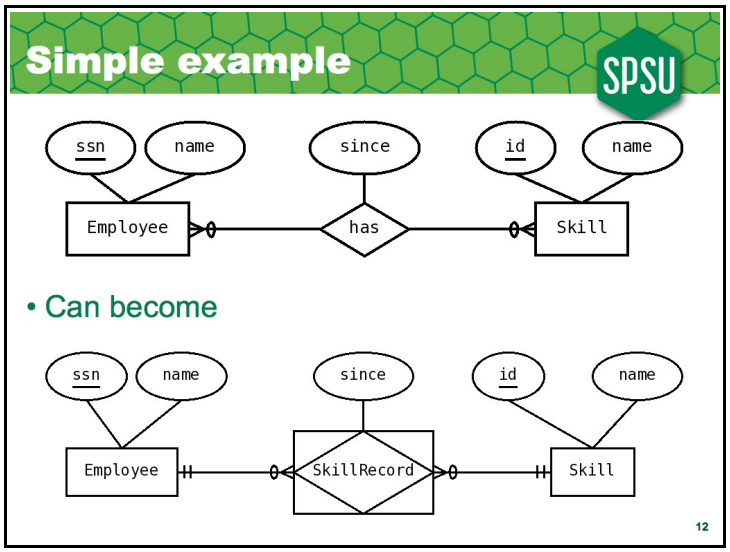
A more complicated concept is that of an associative entity; this is an entity, but it represents an association; we can also think of it as a kind of a weak entity that depends on more than one strong entity for its identification.



Associative entities are mainly used for representing ternary relationships, but they're also used when we start modeling something as a relationship, and then we realize we need it to participate in other relationships, but only entities participate in relationships.

The notation for an associative entity is to use a rectangle with a diamond inside, and we use lines only, without the diamond, to join it with the entities it associates. These lines represent relationships of this associative entity with the entities it associates, but we use a line to mark these relationships as special.

Let's look at a simple example of what an associative entity could look like. Imagine we want to represent employees' skills; we decide to represent the skills as entities; then we can represent the skills an employee have by using a relationship, say *has*, between employee and skill. Since an employee can have many skills and different employees can have the same skill, we make this a many to many relationship; we also keep track of when did the employee acquired the skill by adding an attribute, *since*, to the has relationship.



Instead of representing it this way, we could convert the has relationship into an associative entity; since it will now be an entity, we need to use a noun for its name, so we call it SkillRecord; now each instance of a SkillRecord associates one Employee and one Skill, and so it needs relationships joining them with those; however, we use single lines with the cardinality constraints, instead of using the normal diamonds, to mark these relationships as special. Notice that each SkillRecord is associated with exactly one Employee and exactly one Skill, since each instance of SkillRecord represents one relationship instance from the old has relationship; notice also that, just as you would never have two relationship instances for the same entity instances, that is, you wouldn't record twice in your DB the fact that Orlando has Programming Ability, the same restriction applies here; you can only have one SkillRecord instance for the same Employee and Skill.

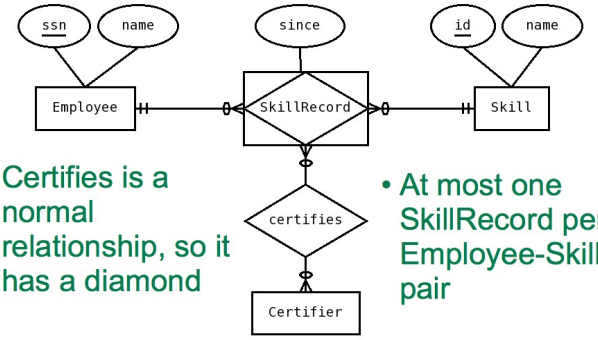
Now, representing this situation with an associative entity is more complicated than a regular relationship, so why would we do it ?

Well, now that SkillRecord is an entity, it can have other relationships; so now we can keep track of which entity certifies that the employee possesses that skill.

Notice that certifies is a normal relationship, so it has a diamond, just like any other relationship.

Why ?

- Associative entity is an entity, so it can now have other relationships !



- Certifies is a normal relationship, so it has a diamond
- At most one SkillRecord per Employee-Skill pair

14

Although we can use associative entities to represent situations like above, we also need them to represent ternary relationships, and in general relationships of degree higher than two.

Ternary relationships associate three entities *at the same time*; notice the entities could be related among themselves by binary relationships but this is different information.

To represent ternary relationships we need to use associative entities, that associate three entities, and similarly for relationships of even higher degrees.

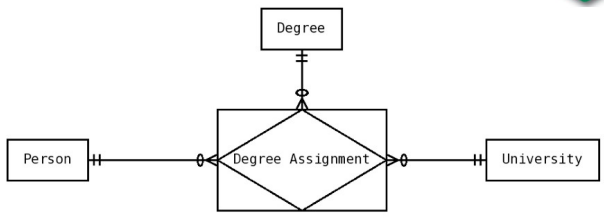
Ternary Relationships

- Most relationships are binary, but sometimes we need to relate three (or more :) entities at the same time
- Ternary relationships associate 3 entities *at the same time*
- Represent different information that having 3 binary relationships among the pairs of entities
- To represent them, we use associative entities
- At most ONE relationship instance for each trio of entity instances

15

For example, imagine we want to represent the fact that a person gets a specific degree from a university; say Orlando got a PhD from Tulane University; notice we need to associate those three things at the same time, Orlando, PhD and Tulane; in this case we have a ternary relationship, and to represent it we create an associative entity, say Degree Assignment, and we associate it with just lines with Person, Degree and University, representing the fact that one instance of DegreeAssignment represents an association between one Person, one Degree and one University.

Example: Degree



- A person gets a specific Degree from a University
- Degree assignment associates, at the same time, the person, degree and university

16