E-R Diagram: Weak Entity, Subclass

CS 4750
Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.6]
[Ricardo and Urban, Database Illuminated, Ch.3]
Recap: Self-Referential, Multi-Way

Given the following E-R diagram. Discuss with your neighbors, come up with an example that can be represented by the diagram.

“Microsoft buys a printer from HP”
Recap: Multi-Way Relationships

Suggest how we may convert the 4-way relationship into an E-R diagram with binary relationships.
Recap: Multi-Way Relationships

Suggest how we may convert the 4-way relationship into an E-R diagram with binary relationships.
E-R Diagram: Building Blocks

(string) Entity set

Attribute

Relationship

Weak entity

Subclass

is a

Note: colors are not part of E-R Diagram. They simply are used to increase readability.
Strong and Weak Entity Sets

**Strong entity set**

- Entities can be identified by the values of their attributes (a primary key)
- (what we have been discussing so far)

**Weak entity set**

- Entities cannot be identified by the values of their attributes
- There is no primary key made from its own attributes
- An entity can be identified by a combination of their attributes ("discriminator") and the relationship they have with another entity set ("identifying relationship")
Weak Entity Set

By definition, there must be total participation between the weak entity set and its identifying relationship.

- **Discriminator**: payment_number
- **Total participation**: payment
- **Identifying relationship**: loan_payment
- **Strong entity set**: loan
  - **l_number**
  - **amount**

Does not have sufficient attributes to form a primary key. Depends on the strong entity set it is associated with. Needs a discriminator and a primary key of the strong entity set.
Homework cannot exist without a course.
Every homework must belong to a single class.
A course can have many homework.
Different courses may have the same homework number.
To identify a homework, we need c_number and hw_number.
Let’s try: Weak Entity Set

Draw an E-R diagram for the following scenario

A movie studio might have several film crews.
The crews might be designated by a crew’s number as crew1, crew2, and so on.
Each crew has multiple phone numbers.
Other studios might use the same designations for crews, so the attribute crew’s number is not a key for crews.
To name a crew uniquely, we need to give both the name of the studio to which it belongs and the crew’s number.
Let’s try: Weak Entity Set

Draw an E-R diagram for the following scenario

A movie studio might have several film crews. The crews might be designated by a crew’s number as crew1, crew2, and so on. Each crew has multiple phone numbers. Other studios might use the same designations for crews, so the attribute crew’s number is not a key for crews. To name a crew uniquely, we need to give both the name of the studio to which it belongs and the crew’s number.
Subclassing

• An entity set may contain entities that have special properties not associated with all members of the set

• Subclasses ~ special-case entity sets

• Isa (or Is-a) ~ special kind of relationship (one-to-one)

Similar to classes in OO (super-classes and sub-classes)
Subclassing

Subclasses are mutually exclusive

Subclasses implicitly inherit superclass key and attributes

Generalization / Specialization
The triangle points to the specialization

Product

edu_product

Software

Category
Price
Age group
Platforms

Name
Price
Category
Name
Done with the building blocks

Let’s transition to design decision

and converting E-R diagram into Relational designs
Recap: Entity vs. Attribute

What are main differences between entities and attributes?

• Entities can model situations that attribute cannot model naturally
• Entities can participate in relationships
• Entities can have attributes
• Attributes cannot do any of these
Design Decision

Should “price” be an entity or an attribute?

For each tuple (person, product, vendor), there is a value of price.
Since “price” is just the actual amount, treating it as an attribute is adequate. No need to make it an entity.
Design Decision (3)

How about this?

- A “person” is an attribute of “buy”
- A “vendor” is an attribute of “buy”
- A “product” is an attribute of “buy”
- Cannot model something about a “person” (or “vendor” or “product”) such as date-of-birth, address
- A “person” will involve in any relationship “buy” is associated with
Decisions to Make

- **Entity set vs. attributes**
  - Has more data \(\rightarrow\) entity set
  - Is the data \(\rightarrow\) attribute

- **Entity set vs. relationship set**
  - Entity set \(\rightarrow\) nouns (students, faculty, loads, ...)
  - Relationship \(\rightarrow\) possession verbs (teaches, advises, owns, works for, ...)

- **Binary vs. n-ary relationship sets**

- **Specialization / generalization**
Rules of Thumb

• Keep it simple
• Don’t over complicate things
• Choose the right elements (entities vs. attributes)
• Choose the right relationships
• Follow the specification of the application
• Avoid NULL value
• Avoid redundancy
• Consider small number of tables
There is a unique table which is assigned the name of the corresponding entity set or relationship set

```
product(pid, name, description)
company(cname, address)
makes(cname, pid)
```

E-R Diagrams to Relations

Schema statement
Strong Entity Set

Direct map:

- entity name → relation name
- Attributes → columns
- Primary key: same as entity

```
product(pid, name, description)
company(cname, address)
```
Strong Entity Set with Composite Attribute

Create separate attributes for each component
Don’t include the higher level attribute

person(pid, first_name, middle_name, last_name)
Strong Entity Set with Multivalued Attribute

Create a separate table for the multivalued attribute

Name the table with the concatenation, separated by “_”

entityname_attributename

Primary key: all attributes

person(pid, name)

person_phone(pid, phone_number)
Weak Entity Set

Let $A$ be a weak entity set and $B$ be the identifying strong entity set on which $A$ depends.

Create a table with primary key of $B$ and all $A$’s attributes.

Primary key: primary key of $B$ (strong entity) and discriminator of $A$.

```
homework(c_number, hw_number, due_date, total_scores)
```
Relationship Set: Many-to-Many

Table: primary keys of both participating entity sets and any attributes on the relationship itself

Primary key: primary keys of both participating entity sets

makes(pid, cname, quantity)

Primary keys of both entities
Relationship Set: Many-to-One / One-to-Many

Table: primary keys of both participating entity sets and any attributes on the relationship itself

Primary key: primary keys of the entity set on the “many” side

makes(pid, cname, quantity)
Because the total participation requires all entity to be participated in the relationship

→ add the primary key of the “one” side to the “many” side entity set, no table for relationship needed

Primary key: primary keys of the entity set on the “many” side

```
product(pid, name, description, cname, address, quantity)
```
Relationship Set: One-to-One

Table: Either side can be used as the main table
(Which side? doesn’t matter. Pick the one that makes the most sense)

Add the other side’s primary key to it

Primary key: primary keys of the entity set you pick

company(cname, address, pid, quantity)
Recap: E-R to Relations (1)

Convert the following E-R diagram into relations

stars(name, address)

movies(title, year, genre, length)

stars-in(title, year, starsName)

owns(title, year, studioName)
Recap: E-R to Relations (2)

Convert the following E-R diagram into relations

flights(number, day, aircraft)

customer(custID, name, phone, address)

bookings(number, day, custID, row, seat)
Subclass (Option 1)

Keep everything
Primary key of the lower level entity set: from the higher level

Drawback: need to access more tables to get info about the lower levels

product(name, price, category)
edu_product(name, age_group)
software(name, platforms)
Recap: Subclasses (option 1)

Convert the following E-R diagram into relations

depts(name, dept_chair)
courses(deptName, number, room)
labCourses(deptName, number, computerAllocation)
seminarCourses(deptName, number, research)
Subclass (Option 2)

- Keep specialization entity sets
- No table for generalization entity set
- Primary key of the lower level entity set: from the higher level
- Drawback: redundancy if entities have more than one specialization

```
edu_product(name, price, category, age_group)
software(name, price, category, platforms)
```
Recap: Subclasses (option 2)

Convert the following E-R diagram into relations

depts(name, dept_chair)
labCourses(deptName, number, room, computerAllocation)
seminarCourses(deptName, number, room, research)
Subclass (Option 3)

Keep generalization entity set
No table for specialization entity sets

Drawback: NULL in attributes from specialization entity sets

Although less duplication of data, need to handle NULL value

product(name, price, category, age_group, platforms)
Recap: Subclasses (option 3)

Convert the following E-R diagram into relations

\[ \text{depts} (\text{name, dept\_chair}) \]  
\[ \text{courses} (\text{deptName, number, room, computerAllocation, research}) \]
Subclass: Design Decision

Depending on the number of attributes of the generalization entity set and specialization entity set

- If balanced $\rightarrow$ do option 1 (create all)
- If more attributes in specialization $\rightarrow$ do option 2
- If more attributes in generalization $\rightarrow$ do option 3

In general, design decision depends on

- The number of attributes
- DB administrator’s decision

Overall goal: minimize duplication (there is no one correct way)
Wrap-Up

• Weak entity sets
• Subclasses
• Converting from E-R diagrams to relational designs
  • Turn each entity set into a relation with the the same set of attributes
  • Replace a relationship by a relation whose attributes are the keys for the connected entity sets
  • Weak entity sets cannot be translated straightforwardly to relations
  • “Is a” relationships and subclasses require careful treatment

What’s next?
• Apply the concept to database scenarios