Entity-Relationship Model

- Diagrams
- Class hierarchies
- Weak entity sets

Purpose of E/R Model

- The E/R model allows us to sketch the design of a database informally.
- Designs are pictures called entity-relationship diagrams.
- Fairly mechanical ways to convert E/R diagrams to real implementations like relational databases exist.

Entity Sets

- *Entity* = “thing” or object.
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- *Attribute* = property of an entity set.
  - Generally, all entities in a set have the same properties.
  - Attributes are simple values, e.g. integers or character strings.

E/R Diagrams

- In an entity-relationship diagram, each entity set is represented by a rectangle.
- Each attribute of an entity set is represented by an oval, with a line to the rectangle representing its entity set.

Example

- Entity set Beers has two attributes, name and manf (manufacturer).
- Each Beer entity has values for these two attributes, e.g. (Bud, Anheuser-Busch)

Relationships

- A relationship connects two or more entity sets.
- It is represented by a diamond, with lines to each of the entity sets involved.
Example

**Example**

- For the relationship *Sells*, we might have a relationship set like:

<table>
<thead>
<tr>
<th>Bar</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe’s Bar</td>
<td>Bud</td>
</tr>
<tr>
<td>Joe’s Bar</td>
<td>Miller</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Bud</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Pete’s Ale</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Bud Lite</td>
</tr>
</tbody>
</table>

**Multiway Relationships**

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that drinkers will only drink certain beers at certain bars.
  - Our three binary relationships Likes, Sells, and Frequent do not allow us to make this distinction.
  - But a 3-way relationship would.

**A Typical Relationship Set**

<table>
<thead>
<tr>
<th>Bar</th>
<th>Drinker</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe’s Bar</td>
<td>Ann</td>
<td>Miller</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Ann</td>
<td>Bad</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Ann</td>
<td>Pete’s Ale</td>
</tr>
<tr>
<td>Joe’s Bar</td>
<td>Bob</td>
<td>Bud</td>
</tr>
<tr>
<td>Joe’s Bar</td>
<td>Bob</td>
<td>Miller</td>
</tr>
<tr>
<td>Joe’s Bar</td>
<td>Cal</td>
<td>Miller</td>
</tr>
<tr>
<td>Sue’s Bar</td>
<td>Cal</td>
<td>Bud Lite</td>
</tr>
</tbody>
</table>
Many-Many Relationships

- Think of a relationship between two entity sets, such as Sells between Bars and Beers.
- In a many-many relationship, an entity of either set can be connected to many entities of the other set.
  - E.g., a bar sells many beers; a beer is sold by many bars.

Many-One Relationships

- Some binary relationships are many-one from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.

Example

- Favorite, from Drinkers to Beers is many-one.
- A drinker has at most one favorite beer.
- But a beer can be the favorite of any number of drinkers, including zero.

One-One Relationships

- In a one-one relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Beers.
  - A beer cannot be made by more than one manufacturer, and no manufacturer can have more than one best-seller (assume no ties).

In Pictures:

- Many-many
- Many-one
- One-one

Representing “Multiplicity”

- Show a many-one relationship by an arrow entering the “one” side.
- Show a one-one relationship by arrows entering both entity sets.
- In some situations, we can also assert “exactly one,” i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow.
Example

- Consider Best-seller between Manfs and Beers.
- Some beers are not the best-seller of any manufacturer, so a rounded arrow to Manfs would be inappropriate.
- But a manufacturer has to have a best-seller (we assume they are beer manufacturers).

In the E/R Diagram

Attributes on Relationships

- Sometimes it is useful to attach an attribute to a relationship.
- Think of this attribute as a property of tuples in the relationship set.

Example

- Create an entity set representing values of the attribute.
- Make that entity set participate in the relationship.

Price is a function of both the bar and the beer, not of one alone.
Example

- **Bars** — **Sells** — **Beers**
  - **Prices** — **price**

Note convention:
arrow from multiway relation/hip = “gi other
title entity sets determine a
unique one of these.”

Roles

- Sometimes an entity set appears more
  than once in a relationship.
- Label the edges between the
  relationship and the entity set with
  names called *roles*.

Example

- **Married**
  - **Husband**
    - Bob
    - Joe
  - **Wife**
    - Ann
    - Sue

Example

- **Buddies**
  - **Drinkers**
    - 1
      - Buddy1
        - Bob
        - Joe
      - Buddy2
        - Sue
        - Moe
    - 2
      - Joe
      - Moe

Subclasses

- Subclass = special case = fewer
  entities = more properties.
- Example: Ales are a kind of beer.
  - Not every beer is an ale, but some are.
  - Let us suppose that in addition to all the
    properties (attributes and relationships) of
    beers, ales also have the attribute *color*.

Subclasses in E/R Diagrams

- Assume subclasses form a tree.
  - I.e., no multiple inheritance.
- Isa triangles indicate the subclass
  relationship.
  - Point to the superclass.
E/R Vs. Object-Oriented Subclasses

- In the object-oriented world, objects are in one class only.
  - Subclasses inherit properties from superclasses.
- In contrast, E/R entities have components in all subclasses to which they belong.
  - Matters when we convert to relations.

Keys

- A key is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key.
  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set.

Keys in E/R Diagrams

- Underline the key attribute(s).
- In an Isa hierarchy, only the root entity set has a key, and it must serve as the key for all entities in the hierarchy.
**Example: a Multi-attribute Key**

- **Courses**
  - dept, number, hours, room

  - Note that hours and room could also serve as a key, but we must select only one key.

**Weak Entity Sets**

- Occasionally, entities of an entity set need “help” to identify them uniquely.
- Entity set $E$ is said to be weak if in order to identify entities of $E$ uniquely, we need to follow one or more many-one relationships from $E$ and include the key of the related entities from the connected entity sets.

**Example**

- **name** is almost a key for football players, but there might be two with the same name.
- **number** is certainly not a key, since players on two teams could have the same number.
- But **number**, together with the **Team** related to the player by **Plays-on** should be unique.

**In E/R Diagrams**

- Double diamond for supporting many-one relationship.
- Double rectangle for the weak entity set.

**Weak Entity-Set Rules**

- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  - Not every many-one relationship from a weak entity set need be supporting.
- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  - E.g., player-number and team-name is a key for Players in the previous example.

**Design Techniques**

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.
Avoiding Redundancy

- Redundancy occurs when we say the same thing in two different ways.
- Redundancy wastes space and (more importantly) encourages inconsistency.
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other, related version.

Example: Good

This design gives the address of each manufacturer exactly once.

Example: Bad

This design states the manufacturer of a beer twice: as an attribute and as a related entity.

Example: Bad

This design repeats the manufacturer’s address once for each beer; loses the address if there are temporarily no beers for a manufacturer.

Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.
  - It is the “many” in a many-one or many-many relationship.

Example: Good

- *Manfs* deserves to be an entity set because of the nonkey attribute *addr*.
- *Beers* deserves to be an entity set because it is the “many” of the many-one relationship *ManBy*.
Example: Good

There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Example: Bad

Since the manufacturer is nothing but a name, and is not at the “many” end of any relationship, it should not be an entity set.

Don’t Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself.
  - They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique ID’s for entity sets.
  - Examples include social-security numbers, automobile VIN’s etc.

When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID’s.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.