Database Systems

Lecturer: Anxiao (Andrew) Jiang
Chapter 4. High-Level Database Models
Steps in database design:

1. Show structure of data using a diagram

2. Turn the diagram into relations
Diagram for data:

George  Pipi  Homer  Peter  Carlson

name  hair color  fly

People
Every entity has three attributes: name, hair color, fly

“name” is a key. (Given “name”, “hair color” and “fly” are fixed.)
Relation Schema:

People

<table>
<thead>
<tr>
<th>name</th>
<th>hair color</th>
<th>fly</th>
</tr>
</thead>
</table>

entity set

attribute

attribute

name

hair color

fly
Relation:

People

<table>
<thead>
<tr>
<th>name</th>
<th>hair color</th>
<th>fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>black</td>
<td>no</td>
</tr>
<tr>
<td>Pipi</td>
<td>red</td>
<td>no</td>
</tr>
<tr>
<td>Homer</td>
<td>black</td>
<td>no</td>
</tr>
<tr>
<td>Peter</td>
<td>red</td>
<td>yes</td>
</tr>
<tr>
<td>Carlson</td>
<td>red</td>
<td>yes</td>
</tr>
</tbody>
</table>
In life, entities are related to each other. So there are relationships.
Diagram for data:

- People:
  - name
  - hair color
  - fly

- Likes

- Foods:
  - food name
  - calories
Diagram for data:

- **People** (entity set)
  - Attributes:
    - name
    - hair color
    - fly

- **Likes** (relationship)

- **Foods** (entity set)
  - Attributes:
    - food name
    - calories
Schemas:

<table>
<thead>
<tr>
<th>Entity Set</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>name, hair color, fly</td>
</tr>
<tr>
<td>Foods</td>
<td>food name, calories</td>
</tr>
<tr>
<td>Likes</td>
<td>name, food name</td>
</tr>
</tbody>
</table>
Relations:

<table>
<thead>
<tr>
<th>People</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>hair color</td>
<td>fly</td>
</tr>
<tr>
<td>George</td>
<td>black</td>
<td>no</td>
</tr>
<tr>
<td>Pipi</td>
<td>red</td>
<td>no</td>
</tr>
<tr>
<td>Homer</td>
<td>black</td>
<td>no</td>
</tr>
<tr>
<td>Peter</td>
<td>red</td>
<td>yes</td>
</tr>
<tr>
<td>Carlson</td>
<td>red</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Foods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>food name</td>
<td>calories</td>
</tr>
<tr>
<td>strawberry</td>
<td>low</td>
</tr>
<tr>
<td>orange</td>
<td>low</td>
</tr>
<tr>
<td>pineapple</td>
<td>low</td>
</tr>
<tr>
<td>pie</td>
<td>high</td>
</tr>
<tr>
<td>cherry</td>
<td>low</td>
</tr>
<tr>
<td>banana</td>
<td>high</td>
</tr>
<tr>
<td>apple</td>
<td>low</td>
</tr>
</tbody>
</table>

Here are two relations for the two entity sets.

Remember: there is also a relation for the relationship (shown on the next page).
<table>
<thead>
<tr>
<th>name</th>
<th>food name</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>strawberry</td>
</tr>
<tr>
<td>George</td>
<td>pie</td>
</tr>
<tr>
<td>Pipi</td>
<td>orange</td>
</tr>
<tr>
<td>Pipi</td>
<td>cherry</td>
</tr>
<tr>
<td>Homer</td>
<td>pie</td>
</tr>
<tr>
<td>Homer</td>
<td>banana</td>
</tr>
<tr>
<td>Peter</td>
<td>strawberry</td>
</tr>
<tr>
<td>Carlson</td>
<td>cherry</td>
</tr>
<tr>
<td>Carlson</td>
<td>apple</td>
</tr>
</tbody>
</table>
The Entity/Relationship Model

In the **entity-relationship model** (or **E/R model**), the structure of data is represented graphically, as an “entity-relationship diagram,” using three principle element types:

1. Entity sets,
2. Attributes, and
An **entity** is an abstract object of some sort, and a collection of similar entities forms an **entity set**.

**Example**

Let us consider the design of our running movie-database example. Each movie is an entity, and the set of all movies constitutes an entity set.

Likewise, the stars are entities, and the set of stars is an entity set.

A studio is another kind of entity, and the set of studios is a third entity set.

Entity sets have associated **attributes**, which are properties of the entities in that set. For instance, the entity set **Movies** might be given attributes such as **title** and **length**.
**Relationships** are connections among two or more entity sets.

For instance, if **Movies** and **Stars** are two entity sets, we could have a relationship **Stars-in** that connect movies and stars.

While binary relationships – those between two entity sets – are by far the most common type of relationship, the E/R model allows relationships to involve any number of entity sets.
An **E/R diagram** is a graph representing entity sets, attributes, and relationships. Elements of each of these types are represented by nodes of the graph, and we use a special shape of node to indicate the kind, as follows:

- **Entity sets** are represented by **rectangles**.
- **Attributes** are represented by **ovals**.
- **Relationships** are represented by **diamonds**.

Edges connect an entity set to its attributes, and also connect a relationship to its entity sets.
Attribute: oval
Relationship: diamond
Here is an E/R diagram that represents a simple database about movies. The entity sets are **Movies**, **Stars**, and **Studios**.

We will talk about key later (which should be underlined)
Suppose $R$ is a relationship connecting entity sets $E$ and $F$. Then:

- If each member of $E$ can be connected by $R$ to at most one member of $F$, then we say that $R$ is **many-one** from $E$ to $F$.
- If $R$ is both many-one from $E$ to $F$ and many-one from $F$ to $E$, then we say that $R$ is **one-one**.
- If $R$ is neither many-one from $E$ to $F$ or from $F$ to $E$, then we say $R$ is **many-many**.

We put an arrow on the “one” side.
Many-one relationship:

Every student attends at most one university:

```
Students   attends   Universities
```
A one-one relationship:

Example

If every country can have at most one president, and every president can be the president of at most one country, then we have a one-one relationship:
Multiway Relationships

A **multiway relationship** is a relationship involving more than two entity sets. It is represented by lines from the relationship diamond to each of the involved entity sets.
Example

Here is a three-way relationship:

- Stars
- Contracts
- Movies
- Studios
Roles in Relationships

It is possible that one entity set appears two or more times in a single relationship. If so, we draw as many lines from the relationship to the entity set as the entity set appears in the relationship. Each line to the entity set represents a different role that the entity set plays in the relationship. We therefore label the edges between the entity set and relationship by names, which we call “roles.”
Example

Here is a relationship with roles:

```
Original

Sequel-of

Sequel

Movies
```
Attributes on Relationships

Sometimes it is convenient, or even essential, to associate attributes with a relationship.
Example

- Movies
  - title
  - year
  - length
  - genre

- Contracts
  - salary

- Stars
  - name
  - address

- Studios
  - name
  - address
Often, an entity set contains certain entities that have special properties not associated with all members of the set. If so, we find it useful to define certain special-case entity sets, or subclasses, each with its own special attributes and/or relationships. We connect an entity set to its subclasses using a relationship called isa.

Example: “chocolate” is a special kind of “candy”
An **isa** relationship is a special kind of relationship, and to emphasize that it is unlike other relationships, we use a special notation: a triangle. One side of the triangle is attached to the subclass, and the opposite point is connected to the superclass. Every **isa** relationship is one-one, although we shall not draw the two arrows that are associated with other one-one relationships.
Subclasses in the E/R Model

Example:

```
name     address
  |         |
  v         v
Stars

  isa  isa
  v    v
Voices Movies

  isa
  v
Cartoons

  isa
  v
Murder-Mysteries

  weapon
```
Design Principles

- Faithfulness.
- Avoiding redundancy.
- Simplicity counts.
- Choosing the right relationships.
- Picking the right kind of element.
Constraints
The E/R model has several ways to express the common kinds of constraints on the data that will populate the database being designed.

Like the relational model, there is a way to express the idea that an attribute or attributes are a key for an entity set.

We have already seen how an arrow connecting a relationship to an entity set serves as a “functional dependency.”

There is also a way to express a referential-integrity constraint, where an entity in one set is required to have an entity in another set to which it is related.
4.3 Constraints in the E/R Model

The E/R model has several ways to express the common kinds of constraints on the data that will populate the database being designed.

Like the relational model, there is a way to express the idea that an attribute or attributes are a key for an entity set.

We have already seen how an arrow connecting a relationship to an entity set serves as a “functional dependency.”

There is also a way to express a referential-integrity constraint, where an entity in one set is required to have an entity in another set to which it is related.
The E/R model has several ways to express the common kinds of constraints on the data that will populate the database being designed.

Like the relational model, there is a way to express the idea that an attribute or attributes are a key for an entity set.

We have already seen how an arrow connecting a relationship to an entity set serves as a “functional dependency.”

There is also a way to express a referential-integrity constraint, where an entity in one set is required to have an entity in another set to which it is related.
The E/R model has several ways to express the common kinds of constraints on the data that will populate the database being designed.

Like the relational model, there is a way to express the idea that an attribute or attributes are a key for an entity set.

We have already seen how an arrow connecting a relationship to an entity set serves as a “functional dependency.”

There is also a way to express a referential-integrity constraint, where an entity in one set is required to have an entity in another set to which it is related.
Keys in the E/R Model

A **key** for an entity set $E$ is a set $K$ of one or more attributes such that, given any two distinct entities $e_1$ and $e_2$ in $E$, $e_1$ and $e_2$ cannot have identical values for each of the attributes in the key $K$. Some points to remember are:

- Every entity set must have a key, although in some cases – isa-hierarchies and “weak” entity sets (which we will discuss later), the key actually belongs to another entity set.
- There can be more than one possible key for an entity set. However, it is customary to pick one key as the “primary key,” and to act as if that were the only key.
- When an entity set is involved in an isa-hierarchy, we require that the root entity set have all the attributes needed for a key.
A **key** for an entity set $E$ is a set $K$ of one or more attributes such that, given any two distinct entities $e_1$ and $e_2$ in $E$, $e_1$ and $e_2$ cannot have identical values for each of the attributes in the key $K$.

Some points to remember are:

- Every entity set must have a key, although in some cases — isa-hierarchies and “weak” entity sets (which we will discuss later), the key actually belongs to another entity set.

- There can be more than one possible key for an entity set. However, it is customary to pick one key as the “primary key,” and to act as if that were the only key.

- When an entity set is involved in an isa-hierarchy, we require that the root entity set have all the attributes needed for a key.
A key for an entity set $E$ is a set $K$ of one or more attributes such that, given any two distinct entities $e_1$ and $e_2$ in $E$, $e_1$ and $e_2$ cannot have identical values for each of the attributes in the key $K$. Some points to remember are:

- Every entity set must have a key, although in some cases – isa-hierarchies and “weak” entity sets (which we will discuss later), the key actually belongs to another entity set.

- There can be more than one possible key for an entity set. However, it is customary to pick one key as the “primary key,” and to act as if that were the only key.

- When an entity set is involved in an isa-hierarchy, we require that the root entity set have all the attributes needed for a key.
A key for an entity set $E$ is a set $K$ of one or more attributes such that, given any two distinct entities $e_1$ and $e_2$ in $E$, $e_1$ and $e_2$ cannot have identical values for each of the attributes in the key $K$. Some points to remember are:

- Every entity set must have a key, although in some cases – isa-hierarchies and "weak" entity sets (which we will discuss later), the key actually belongs to another entity set.
- There can be more than one possible key for an entity set. However, it is customary to pick one key as the "primary key," and to act as if that were the only key.
- When an entity set is involved in an isa-hierarchy, we require that the root entity set have all the attributes needed for a key.
Representing Keys in the E/R Model

In the E/R diagram notation, we underline the attributes belonging to a key for an entity set.

Example
Referential Integrity

Sometimes, in a **many-one** relationship, it is not just “at most one”; it is actually “exactly one.” This is a “referential integrity” constraint.

The arrow notation in E/R diagrams is able to indicate whether a relationship is expected to support referential integrity in one or more directions.

Suppose $R$ is a relationship from entity set $E$ to entity set $F$. A **rounded arrowhead** pointing to $F$ indicates only that

- the relationship is many-one from $E$ to $F$,

but that

- the entity set $F$ related to a given entity of set $E$ is required to exist.

The same idea applies when $R$ is a relationship among more than two entity sets.
Referential Integrity

Sometimes, in a many-one relationship, it is not just “at most one”; it is actually “exactly one.” This is a “referential integrity” constraint.

The arrow notation in E/R diagrams is able to indicate whether a relationship is expected to support referential integrity in one or more directions.

Suppose \( R \) is a relationship from entity set \( E \) to entity set \( F \). A rounded arrowhead pointing to \( F \) indicates only that

- the relationship is many-one from \( E \) to \( F \),

but that

- the entity set \( F \) related to a given entity of set \( E \) is required to exist.

The same idea applies when \( R \) is a relationship among more than two entity sets.
Referential Integrity

Sometimes, in a **many-one** relationship, it is not just “at most one”; it is actually “exactly one.” This is a “referential integrity” constraint.

The arrow notation in E/R diagrams is able to indicate whether a relationship is expected to support referential integrity in one or more directions.

Suppose \( R \) is a relationship from entity set \( E \) to entity set \( F \). A **rounded arrowhead** pointing to \( F \) indicates only that

- the relationship is many-one from \( E \) to \( F \),

but that

- the entity set \( F \) related to a given entity of set \( E \) is required to exist.

The same idea applies when \( R \) is a relationship among more than two entity sets.
Referential Integrity

Sometimes, in a many-one relationship, it is not just “at most one”; it is actually “exactly one.” This is a “referential integrity” constraint.

The arrow notation in E/R diagrams is able to indicate whether a relationship is expected to support referential integrity in one or more directions.

Suppose $R$ is a relationship from entity set $E$ to entity set $F$. A rounded arrowhead pointing to $F$ indicates only only that

- the relationship is many-one from $E$ to $F$,

but that

- the entity set $F$ related to a given entity of set $E$ is required to exist.

The same idea applies when $R$ is a relationship among more than two entity sets.
A president is the president of **exactly one country**. A country has **at most one president**.
In the E/R model, we can attach a bounding number to the edges that connect a relationship to an entity set, indicating limits on the number of entities that can be connected to any one entity of the related entity set.

Example

We choose to place a constraint on the degree of a relationship, such as that a movie entity cannot be connected by relation **Stars-in** to more than 10 star entities.
Weak Entity Set

Police: Who are you?

Dog: I am Bart Simpson’s dog.
Weak Entity Sets

It is possible for an entity set’s key to be composed of attributes, some or all of which belong to another entity set. Such an entity set is called a weak entity set.

One important reason for the existence of weak entity sets:

- Sometimes entity sets fall into a hierarchy based on classifications unrelated to the “isa hierarchy.” If entities of set E are subunits of entities in set F, then it is possible that the names of E-entities are not unique until we take into account the name of the F-entity to which the E entity is subordinate.

Example: “Students” with ID numbers and “Schools.”
Woman: My UIN is 12345.
Man: My UIN is 12345, too.
Woman: I am from University of Oxford.
Man: I am from University of Cambridge.
If $E$ is a weak entity set, then its key consists of:

1. Zero or more of its own attributes, and
2. Key attributes from entity sets that are reached by certain many-one relationships from $E$ to other entity sets. These many-one relationships are called supporting relationships for $E$, and the entity sets reached from $E$ are supporting entity sets. (Here the “many-one” is referential integrity constraint, namely “exactly one.”)
To identify a student, we need:

1. UIN of the student
2. Name of his/her university
If $E$ is a weak entity set, then its key consists of:

1. Zero or more of its own attributes, and
2. Key attributes from entity sets that are reached by certain many-one relationships from $E$ to other entity sets. These many-one relationships are called **supporting relationships** for $E$, and the entity sets reached from $E$ are **supporting entity sets**. (Here the “many-one” is referential integrity constraint, namely “exactly one.”)
Every student should belong to exactly one university.
We adopt the following conventions to indicate that an entity set is weak and to declare its key attributes:

1. If an entity set is weak, it will be shown as a rectangle with a double border.
2. Its supporting many-one relationships will be shown as diamonds with a double border.
3. If an entity set supplies any attributes for its own key, then those attributes will be underlined.

Whenever we use an entity set $E$ with a double border, it is weak. The key for $E$ is whatever attributes of $E$ are underlined plus the key attributes of those entity sets to which $E$ is connected by many-one relationships with a double border.
Weak Entity Set Notation

We adopt the following conventions to indicate that an entity set is weak and to declare its key attributes:

1. If an entity set is weak, it will be shown an a rectangle with a double border.
2. Its supporting many-one relationships will be shown as diamonds with a double border.
3. If an entity set supplies any attributes for its own key, then those attributes will be underlined.

Whenever we use an entity set $E$ with a double border, it is weak. The key for $E$ is whatever attributes of $E$ are underlined plus the key attributes of those entity sets to which $E$ is connected by many-one relationships with a double border.
We adopt the following conventions to indicate that an entity set is weak and to declare its key attributes:

1. If an entity set is weak, it will be shown as a rectangle with a double border.
2. Its supporting many-one relationships will be shown as diamonds with a double border.
3. If an entity set supplies any attributes for its own key, then those attributes will be underlined.

Whenever we use an entity set $E$ with a double border, it is weak. The key for $E$ is whatever attributes of $E$ are underlined plus the key attributes of those entity sets to which $E$ is connected by many-one relationships with a double border.
UIN

double border

attends

name

exactly one (referential integrity)
We adopt the following conventions to indicate that an entity set is weak and to declare its key attributes:

1. If an entity set is weak, it will be shown as a rectangle with a double border.
2. Its supporting many-one relationships will be shown as diamonds with a double border.
3. If an entity set supplies any attributes for its own key, then those attributes will be underlined.

Whenever we use an entity set $E$ with a double border, it is weak. The key for $E$ is whatever attributes of $E$ are underlined plus the key attributes of those entity sets to which $E$ is connected by many-one relationships with a double border.
UIN

attends

name

underline key attributes
We adopt the following conventions to indicate that an entity set is weak and to declare its key attributes:

1. If an entity set is weak, it will be shown an a rectangle with a double border.
2. Its supporting many-one relationships will be shown as diamonds with a double border.
3. If an entity set supplies any attributes for its own key, then those attributes will be underlined.

Whenever we use an entity set $E$ with a double border, it is weak. The key for $E$ is whatever attributes of $E$ are underlined plus the key attributes of those entity sets to which $E$ is connected by many-one relationships with a double border.
UIN

attends

name

key for students (name and UIN)

key for universities (name)
Example of E/R diagram:

Key for “Universities”: name

Key for “Students”: name (borrowed key from “universities”) ad UIN (its own attribute)
How to turn E/R diagram into relations?
4.5 From E/R Diagrams to Relational Designs

Converting an E/R design to a relational database schema is straightforward:

- Turn each entity set into a relation with the same set of attributes;
- Replace a relationship by a relation whose attributes are the keys for the connected entity sets.
E/R diagram:

- **name**
- People
- hair color
- fly

**Relation:**

People (**name**, hair color, fly)
4.5 From E/R Diagrams to Relational Designs

Converting an E/R design to a relational database schema is straightforward:

- Turn each entity set into a relation with the same set of attributes;
- Replace a relationship by a relation whose attributes are the keys for the connected entity sets.
Schemas:

People (name, hair color, fly)

Foods (food name, calories)

Likes (name, food name)
We get these relation schemas from the above E/R diagram:

1. **Stars**(*name, address*).
2. **Studios**(*name, address*).
3. **Movies**(*title, year, length, genre*).
4. **Stars-in**(*title, year, name*).
5. **Owns**(*title, year, name*).
What if a relationship has its own attributes?

Answer: Include those attributes in the corresponding relation as well.
The relation for a given relationship $R$ has the following attributes:

1. For each entity set involved in relationship $R$, we take its key attribute or attributes as part of the schema of the relation for $R$.
2. If the relationship has attributes, then these are also attributes of relation $R$. 
If one entity set is involved several times in a relationship, in different roles, then its key attributes each appear as many times as there are roles. We must rename the attribute to avoid name duplications.

More generally, should the same attribute name appear twice or more among the attributes of $R$ itself and the keys of the entity sets involved in relationship $R$, then we need to rename to avoid duplication.
Example:

For the relationship “Contracts”, we get the relation schema:

Contracts(title, year, starName, studioOfStar, producingStudio, salary).
Those are the (two) rules for turning entity sets and relationships into relations.

Simple enough?
While these two rules cover much of the ground, there are also several special situations that we need to deal with, including:

1. Weak entity sets cannot be translated straightforwardly to relations.
3. Sometimes, we do well to combine two relations, especially the relation for an entity set $E$ and the relation that comes from a many-one relationship from $E$ to some other entity set.
Handling weak entity sets: do 3 things differently

1. The relation for the weak entity set \( W \) itself must include not only the attributes of \( W \) but also the key attributes of the supporting entity sets.

Schema for "Students":

Students (university name, UIN, student name)
Handling weak entity sets: do 3 things differently

1. The relation for any relationship in which the weak entity set \( W \) appears must use as a key for \( W \) all of its key attributes, including those of other entity sets that contribute to \( W \)'s key.

**Diagram:**

- **Students**
  - student name
- **Universities**
  - name
  - address
- **Likes**
  - fruit name
- **Fruits**

Schema for “Likes”:

\[
\text{Likes (university name, UIN, fruit name)}
\]
Handling weak entity sets: do 3 things differently

However, a supporting relationship $R$, from the weak entity set $W$ to a supporting entity set, need not be converted to a relation at all.

Do we need a schema for “attends”? No.

`attends` (university name, UIN, university name) → redundant
A side question:

Do we need a relation for “Fruits”? It seems that the information on “fruit names” also appears in the relation for “likes”.

Answer: Yes, we do. That is because there may be fruits that no student here likes. Then that fruit’s name will be missing in the relation for “Students”.
While these two rules cover much of the ground, there are also several special situations that we need to deal with, including:

1. Weak entity sets cannot be translated straightforwardly to relations.
3. Sometimes, we do well to combine two relations, especially the relation for an entity set $E$ and the relation that comes from a many-one relationship from $E$ to some other entity set.
When we have an “isa” relationship, we have several choices for conversion to relations.

1. E/R-style conversion: One relation for each subclass. Its attributes include the key attributes and the subclass’s own attributes.

2. An object-oriented approach: One relation per subset of subclasses, with all relevant attributes.

3. Using null values to combine relations: One relation; entities have NULL in attributes that do not belong to them.
E/R-Style Conversion

Example:

![Diagram showing a relationship between Candies and Chocolates with properties name and manufacturer for Candies, and color for Chocolates.]

We have the following (example) relations:

**Candies:**

<table>
<thead>
<tr>
<th>name</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twizzler</td>
<td>Hershey</td>
</tr>
<tr>
<td>Snickers</td>
<td>M&amp;M/Mars</td>
</tr>
</tbody>
</table>

**Chocolates:**

<table>
<thead>
<tr>
<th>name</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snickers</td>
<td>light</td>
</tr>
</tbody>
</table>
Example:

We have the following (example) relations:

Candies:

<table>
<thead>
<tr>
<th>name</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twizzler</td>
<td>Hershey</td>
</tr>
</tbody>
</table>

Chocolates:

<table>
<thead>
<tr>
<th>name</th>
<th>manufacturer</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snickers</td>
<td>M&amp;M/Mars</td>
<td>light</td>
</tr>
</tbody>
</table>
Using Null Values to Combine Relations

Example:

We have the following (example) relation:

Candies:

<table>
<thead>
<tr>
<th>name</th>
<th>manufacturer</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twizzler</td>
<td>Hershey</td>
<td>NULL</td>
</tr>
<tr>
<td>Snickers</td>
<td>M&amp;M/Mars</td>
<td>light</td>
</tr>
</tbody>
</table>
While these two rules cover much of the ground, there are also several special situations that we need to deal with, including:

1. Weak entity sets cannot be translated straightforwardly to relations.
3. Sometimes, we do well to combine two relations, especially the relation for an entity set $E$ and the relation that comes from a many-one relationship from $E$ to some other entity set.
Which is better?

Solutions 1: Use two relations for “Movies” and “Owns”

Movies (title, year, length, genre)
Owns (title, year, studioName)

Solutions 2: Combine into one relation

Movies (title, year, length, genre, studioName)
Example:

Solutions 1: Use two relations for “Movies” and “Owns”

Movies (title, year, length, genre)
Owns (title, year, studioName)

Solutions 2: Combine into one relation

Movies (title, year, length, genre, studioName)  This is better.
Example:

Movies (title, year, length, genre, studioName)

2. Like “length” and “genre”, “studioName” also has (at most) one value for any given movie (namely, given “title” and “year”).
3. If a movie is not owned by any studio, its “studioName” takes NULL value in the above combined relation.