Entity Relationship Model

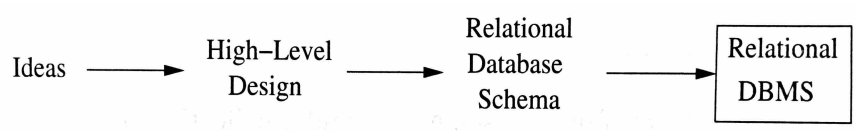
Dr Eugenia Ternovska
Associate Professor

Simon Fraser University

Purpose of E/R Model

Entity-relationship (E/R) model is a notation for describing **schemas in databases**

- A high-level sketch
- Includes some constraints, but not operations
- Designs are pictures called entity-relationship (E/R) diagrams
- Converted to relational DB schemas



Entity Relationship Model (ERM)

1. The Entity Relationship Model
   - Database design
   - Entity, Entity Set, Attribute, Relationship
2. E/R Design Considerations
   - Constraints: Key, Referential, Degree
   - Relationship Conditions: Multiplicity of Relationships, Multiway Relationships
3. More E/R Concepts
   - Combining Relations, Constraints, Subclasses, Weak Entity Sets
4. Conversion to SQL

This lecture covers parts 1 and 2

Design of the Database (1)

1. Step 1: Requirement Analysis
   - What data is going to be stored?
   - What are we going to do with the data?
   - Who should access the data?
2. Step 2: Conceptual Design $\leftarrow$ **Where E/R Model Fits**
   - A high-level description of the database
   - Sufficiently precise that technical people can understand it
   - But, not so precise that non-technical people can not participate

Design of the Database (2)

Usually one starts the design of a database with a conceptual model in E/R form

It is more intuitive, and it models in more general terms

It is like a sketch

Later, in the DB design phase, the conceptual model is transformed into a logical model, usually a relational model

At this transformation stage and also after obtaining the first set of relations, some techniques are used to produce the right collections of tables and their logical interconnections

Design of the Database (3)

The resulting set of tables is **improved by additional transformations**, obtaining a second set of tables

avoiding, e.g. redundancy of data or updates anomalies: **normalization process** – covered later in the course

Finally, the resulting relational model is implemented in a RDBMS

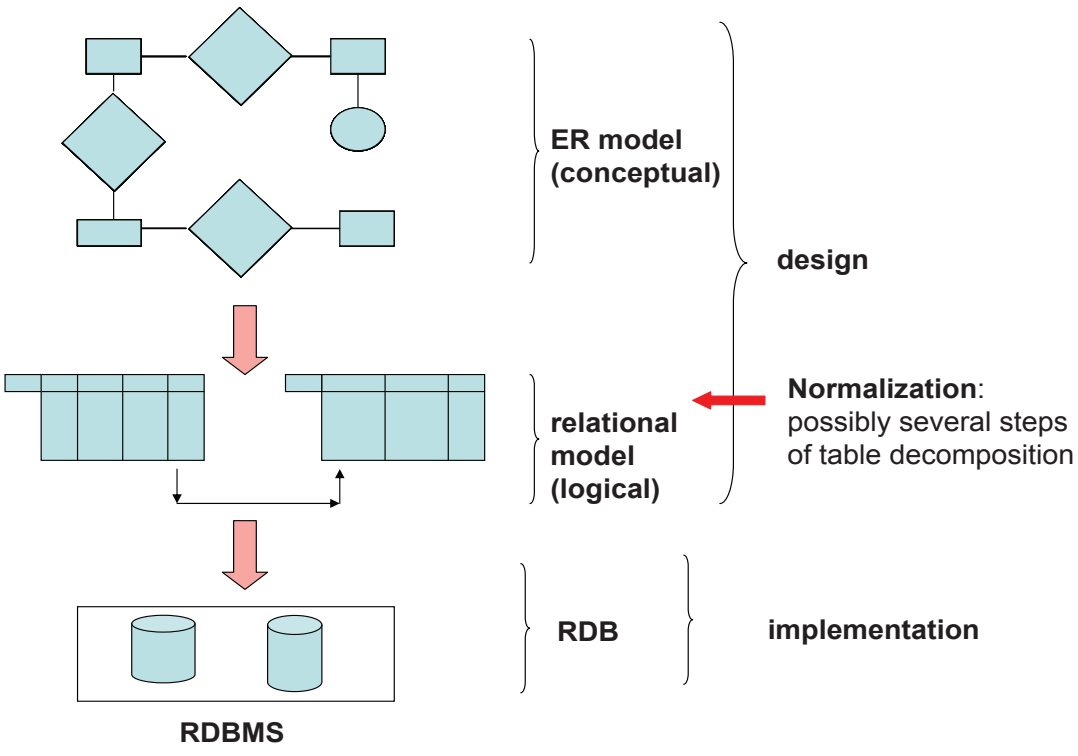
The physical representation usually differs from the logical representation

The former addresses efficiency in terms of storage, access, updates, I/O, ..., which is crucial for applications

Design of the Database (4)

3. Step 3: Detailed Design
   - A Logical Database Design
   - Physical Database Design
   - Security Design

Design Stages



Principal element types (1)

Entity = Individual object.
Similar to an instance of a class in object-oriented languages.

Entity set = collection of similar entities.
Similar to a class in object-oriented languages. But, E/R model is a static concept, involving the structure of data and **not the operations on data**

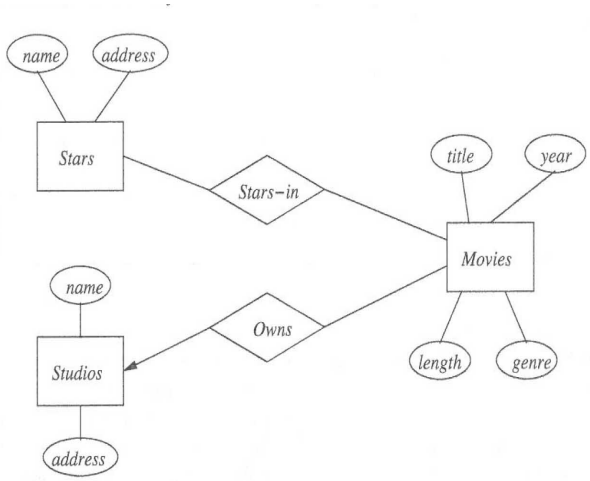
Attribute = Entity sets have attributes, showing properties of the entities in that set
Attributes are simple values, e.g. integers or character strings.

Relationships = Connections among two or more entity sets

Example: Movie Database

- Each movie is an entity, and the set of all movies constitutes an entity set.
- 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.
- The entity set Movies might be given attributes such as title and length.
- Movies and Stars are two entity sets, we could have a relationship Stars-in that connects movies and stars
- Entities are not explicitly represented in E/R diagrams!

E/R Diagrams: Notations

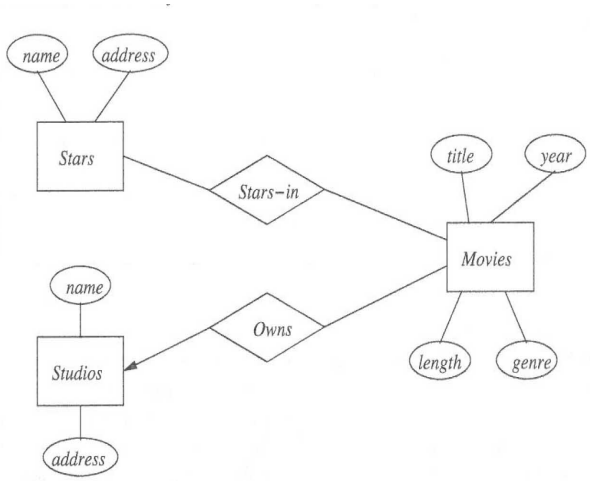


Entity set = **rectangle**

Attribute = **oval**, with a line to its entity set.

Relationship = **diamond**,
with lines to each of the two or more entity sets involved.

Example



Entity sets: *Movies*, *Stars* and *Studios*, with attributes in ovals

Relationships: *Stars-in* and *Owns*

Arrow indicates that each movie is owned by **at most one** studio (uniqueness constraint)

Keys and Candidate Keys (1)

Recall: A key is a set of attributes that uniquely identifies an entity.

A **candidate key** has to satisfy two conditions:

- Be a unique identifier
- Be **minimal** (under set inclusion)

A **super key** is a key that is not required to be minimal

Then: Candidate Key $\Rightarrow$ Super Key
But: Super Key $\not\Rightarrow$ Candidate Key

Note: a Super Key is often called just "Key"

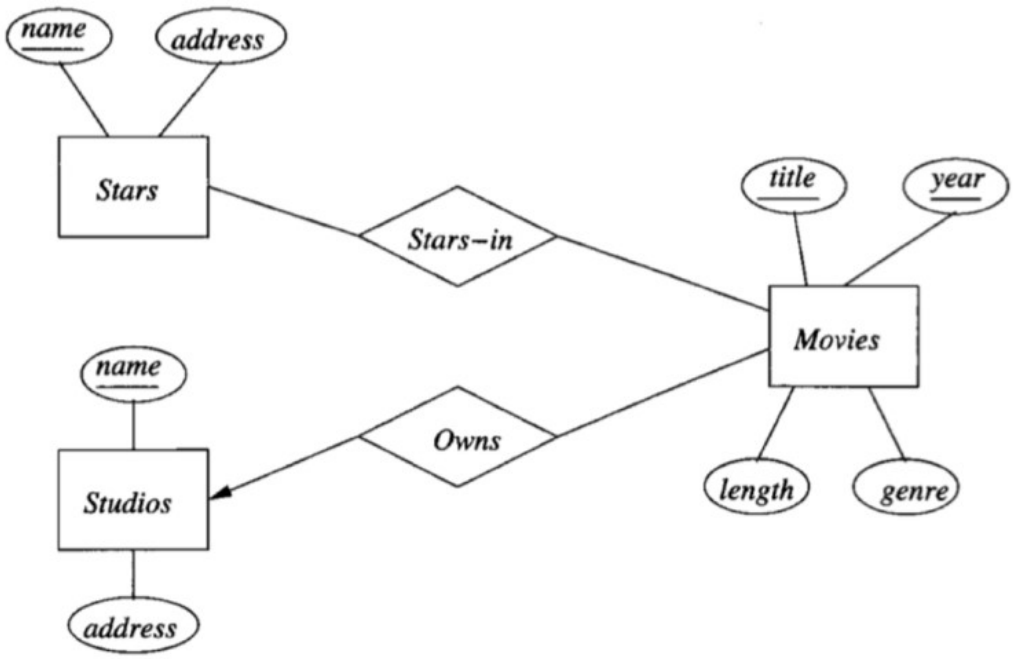
A **primary key** is the one chosen by the database designer

Keys and Candidate Keys (2)

- Every entity must have a key, although in some cases (we will see later) the key actually belongs to another entity set.
- Denote elements of the primary key by underlining.
- There is no notation for representing the situation where there are several keys for an entity set.
- We underline only the primary key.
- Multiple underlining: they are each member of the key

Keys and Candidate Keys (3)

Example: the attributes title and year together form the primary key for Movies.



What is a Relationship? (1)

Let **A** and **B** be sets
A $= \{1, 2, 3\}$, B $= \{a, b, c, d\}$

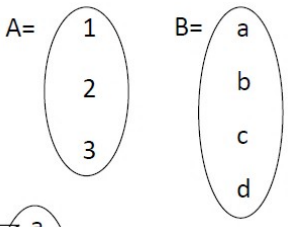
A $\times$ B (the cross-product) is the set of all pairs (a, b)

$$\text{A} \times \text{B} = \{(1, a), (1, b), (1, c), (1, d), (2, a), (2, b), (2, c), (2, d), (3, a), (3, b), (3, c), (3, d)\}$$

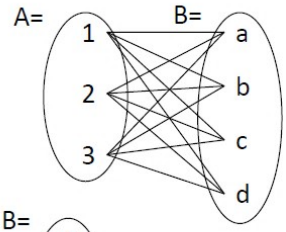
We define a relationship to be a subset of A $\times$ B, e.g.,
R $= \{(1, a), (2, c), (2, d), (3, b)\}$

What is a Relationship? (2)

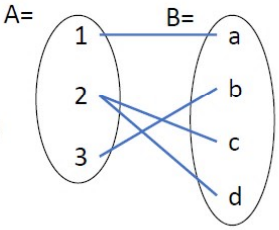
- **A={1,2,3}**
  B={a,b,c,d}



- **A × B = {(1,a), (1,b), (1,c), (1,d),**
  (2,a), (2,b), (2,c), (2,d),
  (3,a), (3,b), (3,c), (3,d)}



- **R = {(1,a), (2,c), (2,d), (3,b)}**



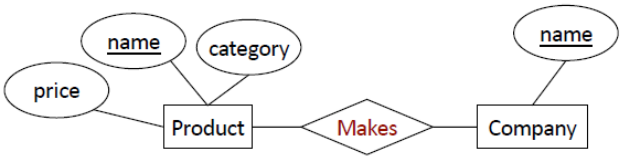
What is a Relationship? (3)

Company

name
Apple
Microsoft

Product

name	category	price
iPhone 8	Electronics	$700
iPad 4	Electronics	$300
Office	Software	$120



A relationship between entity sets Product and Company is a **subset** of all possible pairs of entities in Product and Company

There can only be one relationship for every specific combination of entities

The tuples of the relationship Makes are **uniquely identified by** Product and Company's **keys**

What is a Relationship? (4)

Company

name
Apple
Microsoft

Product

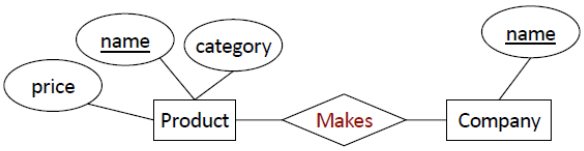
name	category	price
iPhone 8	Electronics	$700
iPad 4	Electronics	$300
Office	Software	$120

Company C × Product P

C.name	P.name	P.category	P.price
Apple	iPhone 8	Electronics	$700
Apple	iPad 4	Electronics	$300
Apple	Office	Software	$120
Microsoft	iPhone 8	Electronics	$700
Microsoft	iPad 4	Electronics	$300
Microsoft	Office	Software	$120

Makes

C.name	P.name
Apple	iPhone 8
Apple	iPad 4
Microsoft	Office

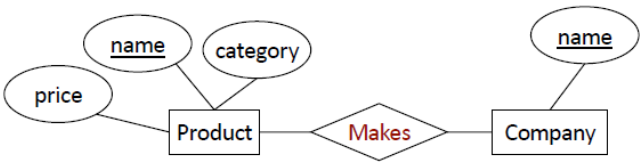


What is a Relationship? (5)

The relationship is uniquely determined by the keys of its entities

Makes

C.name	P.name
Apple	iPhone 8
Apple	iPad 4
Microsoft	Office



Example: the key for Makes is {Product.name, Company.name}

Entity Relationship Model

E/R Design Considerations:

- Constraints: Referential Constraints, Degree Constraints
- Relationship Conditions: Multiplicity of Relationships, Multiway Relationships

Constraints in Modelling

At data modelling time, we may not have instances or specific data values yet

May have only a partial idea about the kind of data values involved

We start by identifying

- entities,
- relationships,
- attributes,
- key constraints,

The latter are **semantic constraints**, expected to be satisfied by the data values when a database is populated (with data)

Referential Integrity

A value appearing in one context must also appear in another

A rounded arrow head pointing to Studios

- Not only the relationship is many-one from, e.g., Owns to Studios
- But also the entity set Studios related to a given entity set Owns is required to exist (be non-empty)

An example E/R diagram showing referential integrity constraints

Degree Constraints

A **bounding number on the edges** that connect a relationship to an entity set

indicates limits on the number of entities that can be connected to any one entity of the related entity set

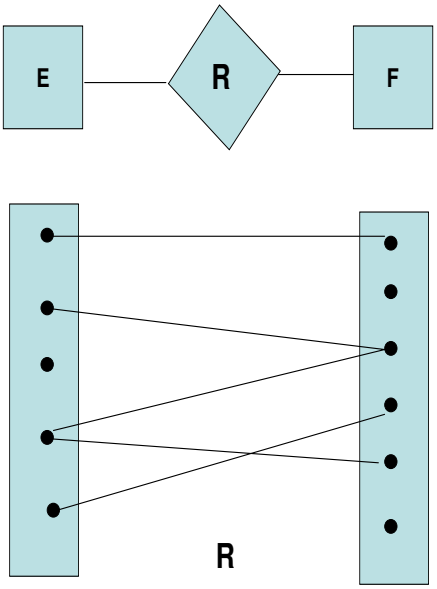
An example representing a constraint on the number of stars per movie

Cardinality Constraints (1)

Cardinality constraints is another example of semantic constraints. They impose lower (min) and upper (max) bounds, so they are satisfied with everything in-between.

They can be partially captured in the E/R model.

Cardinality Constraints (2)



At modelling time, it is possible to impose conditions on the number of connections (via R) between entities in E and F

Multiplicity of E/R Relations (1)

For a **binary** relationship, we can restrict its multiplicity to be:

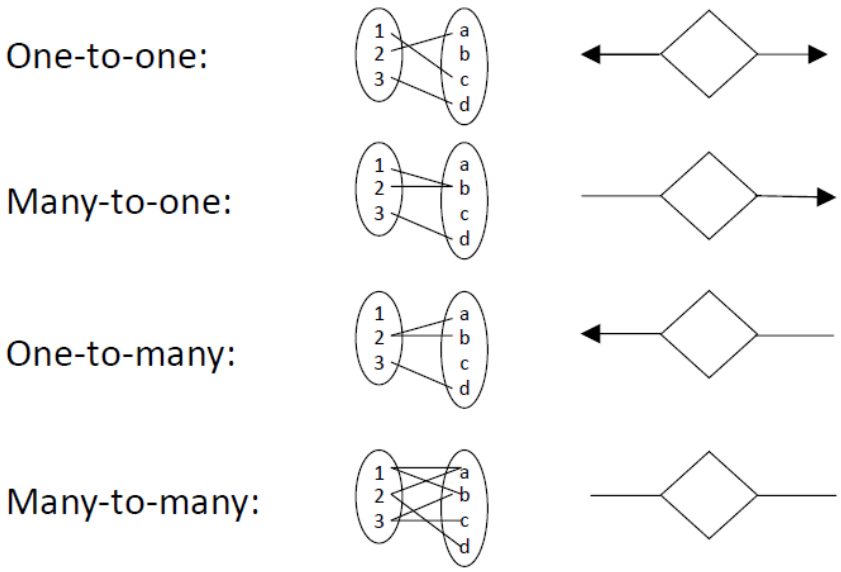
many-one If each member of E can be connected by R to at most one member of F: R is many-one from E to F.

one-one If R is both many-to-one from E to F and many-one from F to E: R is one-one.

many-many If R is neither many-to-one from E to F nor from F to E: R is many-to-many.

An example one-one relationship:

Multiplicity of E/R Relations (2)



Usage of arrows: X $\rightarrow$ Y means
There exists a function mapping from X to Y (Recall the definition of a function)

Multiway Relationships (1)

Sometimes, we need a relationship that connects more than two entity sets.

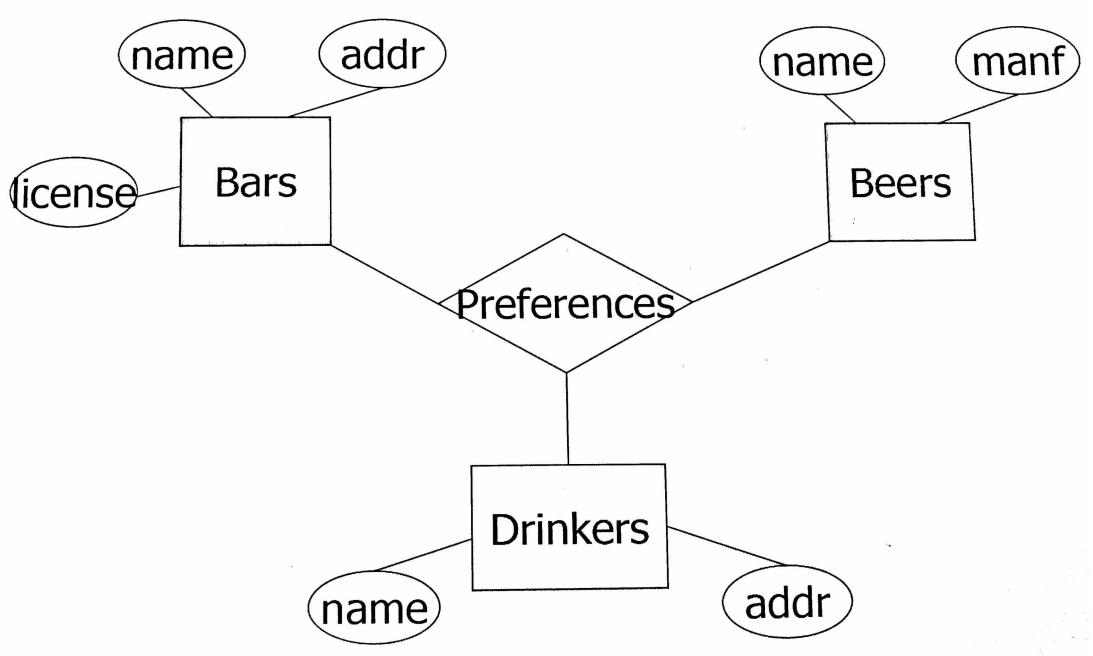
Suppose that drinkers will only drink certain beers at certain bars.

Binary relationships such as *Likes*, *Sells*, and *Frequents* do not allow us to make this distinction.

A 3-way relationship is much more suitable.

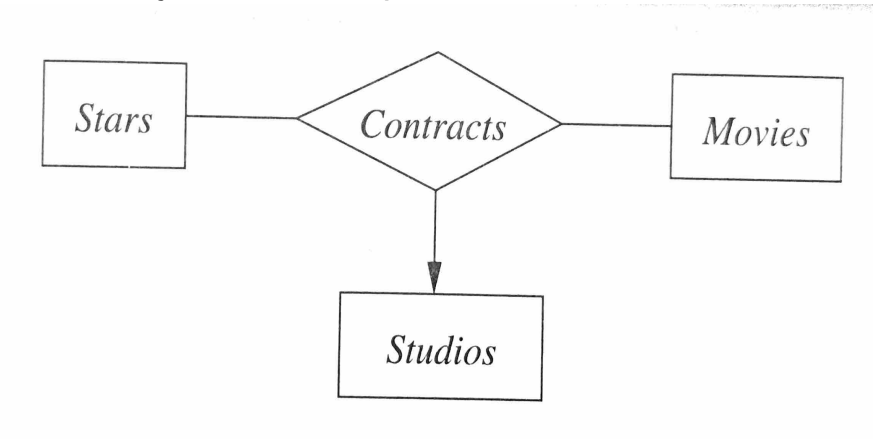
Multiway Relationships (2)

Example: 3-Way Relationship *Preferences*:



Multiway Relationships (3)

Example: 3-Way Relationship:



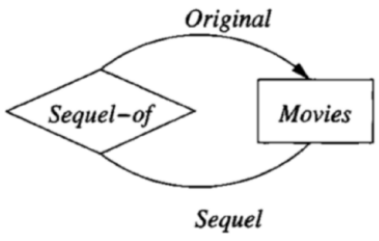
Arrow indicates that for a particular star and movie, there is only one studio with which the start has contracted for that movie

Roles in Relationships

It is possible that one entity set appears two or more times in a single relationship.

Each line to the entity set represents a different role that the entity set plays in the relationship.

We label the edges between the entity set and relationship by names (roles).

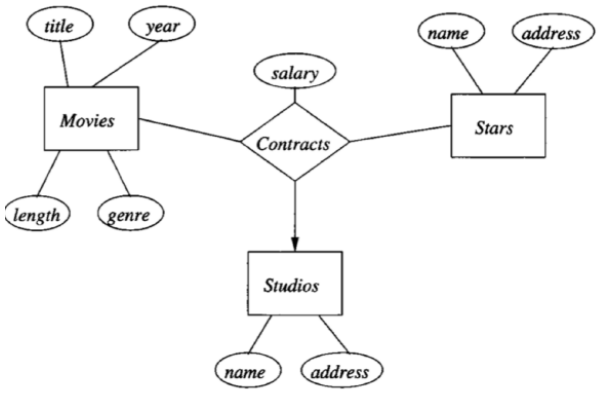


A relationship with roles

Attributes in Relationships (1)

It is never necessary to place attributes on relationships.

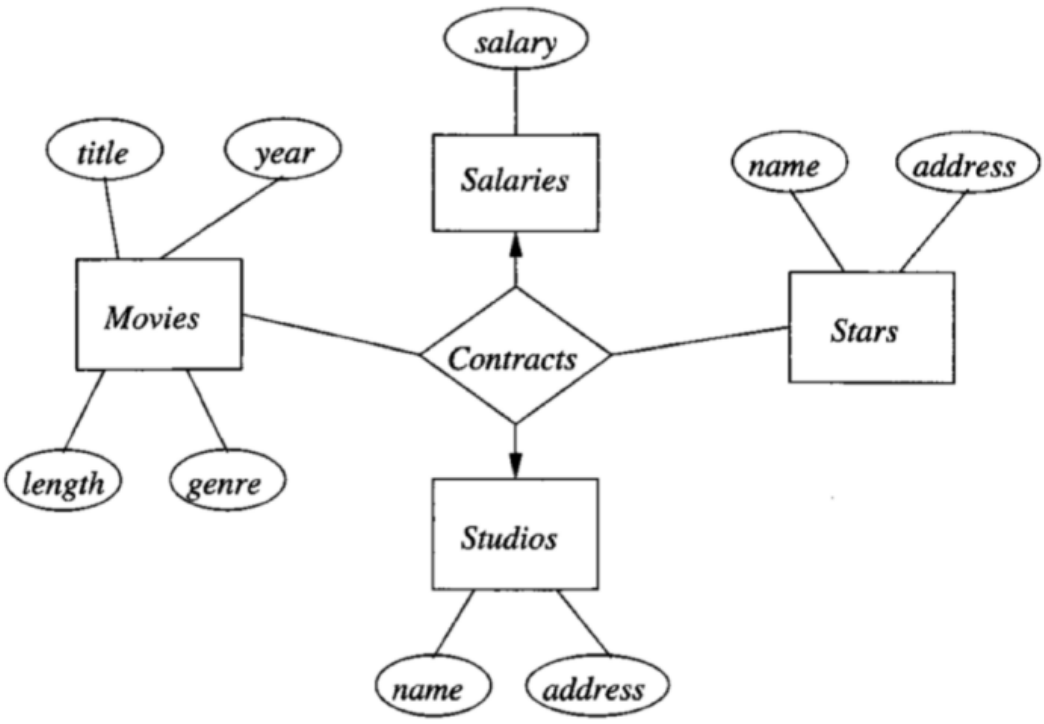
Could be replaced by a new entity set with the attributes.



A relationship with an attribute

On the next slide, we add a new entity set Salaries

Attributes in Relationships (2)



Moving the attribute to an entity set

Is-A Hierarchies (1)

As in C++, or other PLs, some attributes are inherited.

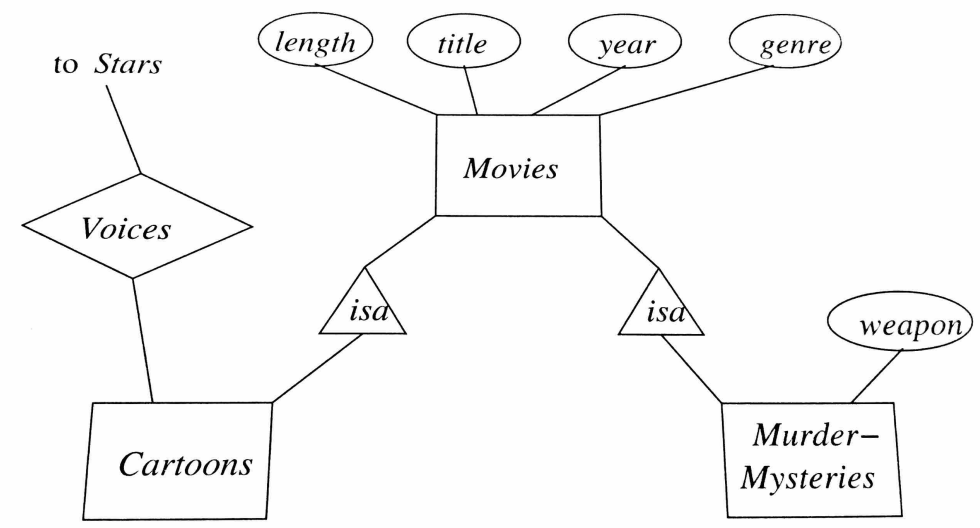
If we declare A Is-A B, every A entity is also considered to be a B entity.

Reasons for using Is-A ("is a")

- To add descriptive attributes specific to a subclass.
- To identify entities that participate in a relationship.

Is-A Hierarchies (2)

Example: Is-A Hierarchy



Sub-entities (subclasses of classes) can be specified using "Is-A"-arrows connecting entities

Is-A Hierarchies (3)

Attributes (properties) are inherited by sub-entities, but sub-entities may have specialized attributes

Participation in relationships are also inherited by sub-entities

The "Is-A" labels (for links) have a fixed, **built-in semantics**, as opposed to the other labels for roles we have used so far

They mean **set-theoretic inclusion** of entities (or entity sets)

Summary of Conceptual Design

- Conceptual design follows requirements analysis,
  - Yields a high-level description of data to be stored
- E/R model is popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: entities, relationships, and attributes (of entities and relationships).
- Some additional constructs: weak entities, Is-A hierarchies, and aggregation.
- Note: There are many variations on E/R model.

Summary of E/R (1)

- Several kinds of integrity constraints can be expressed in the E/R model:
  - key constraints,
  - participation constraints, and
  - overlap/covering constraints for Is-A hierarchies.
- Some foreign key constraints are also implicit in the definition of a relationship set.
- Some constraints (notably, functional dependencies) cannot be expressed in the E/R model.
- Constraints play an important role in determining the best database design for an enterprise.

Summary of E/R (2)

- E/R design is **subjective**. There are often many ways to model a given scenario. Analyzing alternatives can be tricky, especially for a large enterprise.

  Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or N-nary relationships, whether or not to use Is-A hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further.
- Functional Dependencies information and normalization techniques are **especially useful**. They will be studied later in the course.

Acknowledgements

[1] Database Systems: The Complete Book, 2nd EditionHector Garcia-Molina, Jeffrey D. Ullman, Jennifer WidomPrentice Hall, 2009

[2] Database System Concepts, Seventh EditionAvi Silberschatz, Henry F. Korth, S. SudarshanMcGraw-Hill, March 2019www.db-book.com

Additional references and resources used in preparation of this course are listed on the course webpage or mentioned in slides.