These lecture notes include some material from Professors Bertossi, Ullman, Widom, Ramakrishnan, Gehrke

# Entity Relationship Model

Dr Evgenia Ternovska
Associate Professor

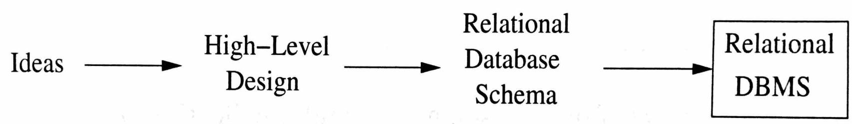
Simon Fraser University

Spring 2018

## Purpose of ER Model

Entity-relationship (ER) 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 diagrams
- Converted to relational DB schemas



## Design of the Database (1)

Usually one starts the design of a database with a conceptual model in ER form

It is more intuitive and models in more general terms

It is less of a model of the DB to come, more 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 connections**

## Design of the Database (2)

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

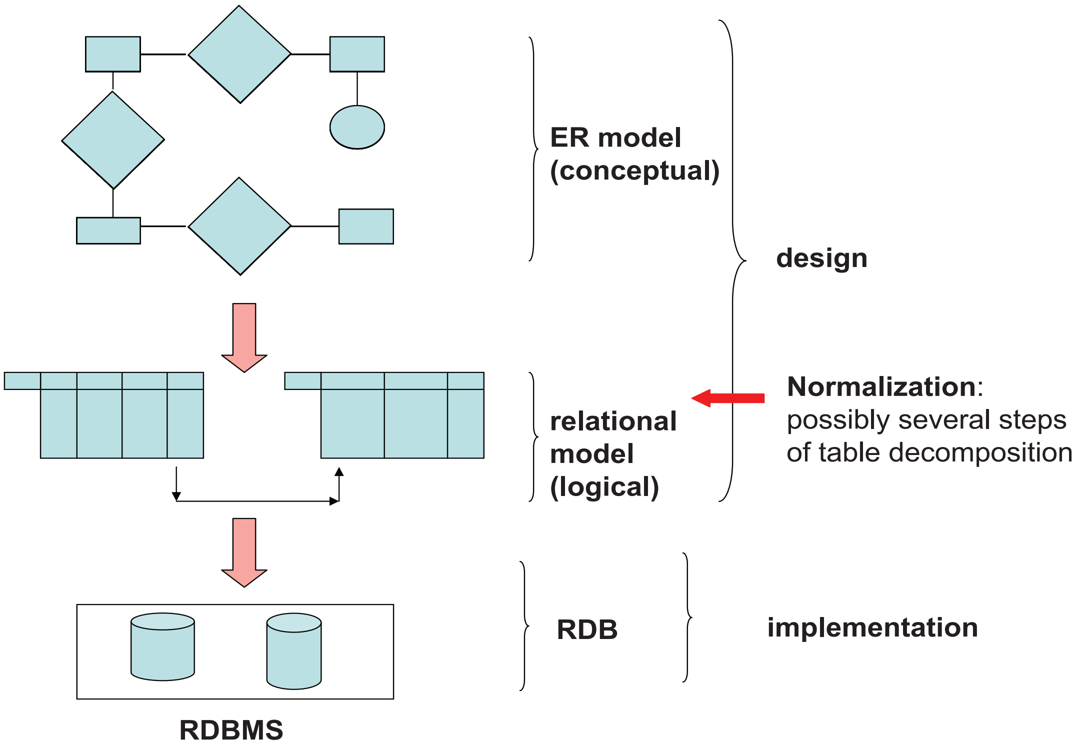
avoiding, e.g. redundancy of data or updates anomalies: **normalization process**

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 Stages




# Entity Sets

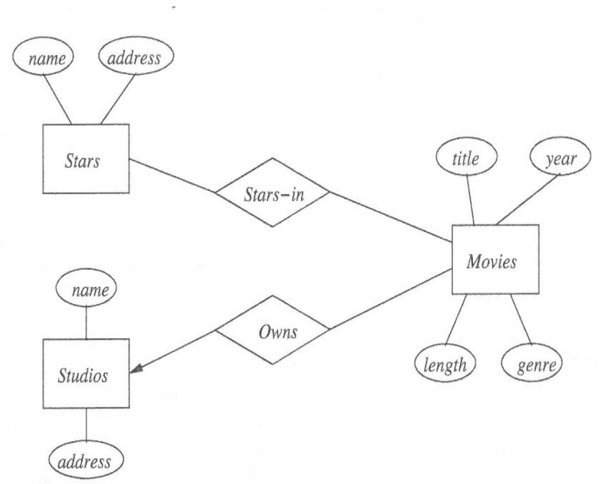
Entity = thing or object.

Entity set = collection of similar entities.

Similar to a class in object-oriented languages.

Attribute = property of (the entities of) an entity set.
Attributes are simple values, e.g. integers or character strings.

# ER Diagrams: Notations

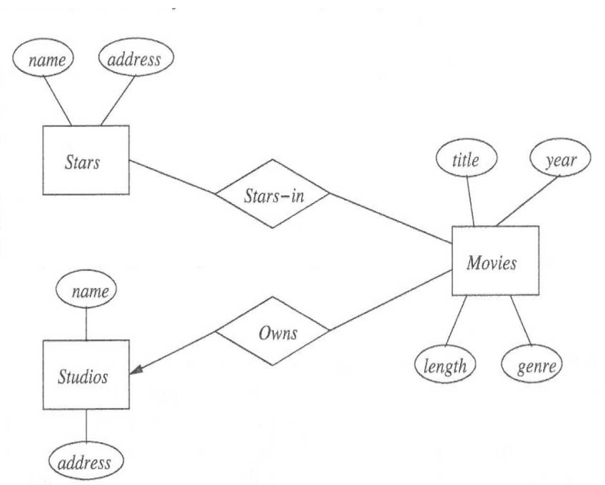



Entity set = **rectangle**

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)

## Relationship Set

The current value of an entity set is the set of entities that belong to it.

Example: the set of all stars in our database.

The value of a relationship is a **relationship set**, a set of tuples with one component for each related entity set.

For the relationship Stars-in, we might have a relationship set like:

| **Movies** | **Stars** |
|---|---|
| Basic Instincts | Sharon Stone |
| Total Recall | Arnold Schwarzenegger |
| Total Recall | Sharon Stone |

## Constraints in Modelling (1)

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)

# Constraints in Modelling (2)

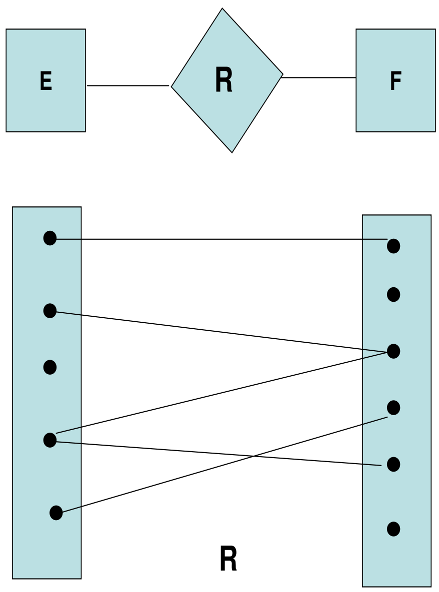
In the ER model we have a **limited way** of capturing/representing semantic constraints

whereas in the Relational model, originated in Ted Codd's work, we have at least the **full power of predicate logic** to express ICs, plus more

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 ER model

# Cardinality Constraints




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

# Multiplicity of Binary ER Relationships

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

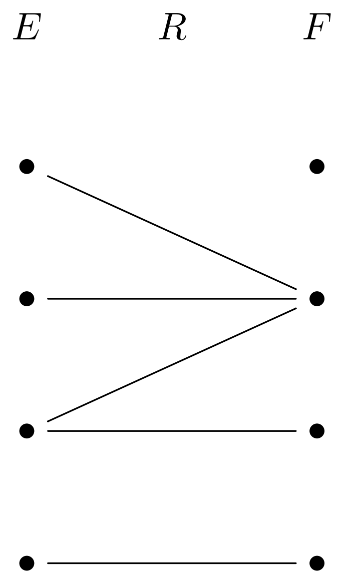
many-one

one-one

many-many

# Many-One Relationships

Each member of E can be connected to at most one member of F




**N-to-1 Constraints:**

min-card(E,R) $= 0$
max-card(E,R) $=$ N ($> 1$)

min-card(F,R) $= 0$
max-card(F,R) $= 1$

The cardinality constraints declare **admissible bounds** for any instance of the model, but those extreme values may not be taken

## Example: Many-One Relationships

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: Many-One Relationship

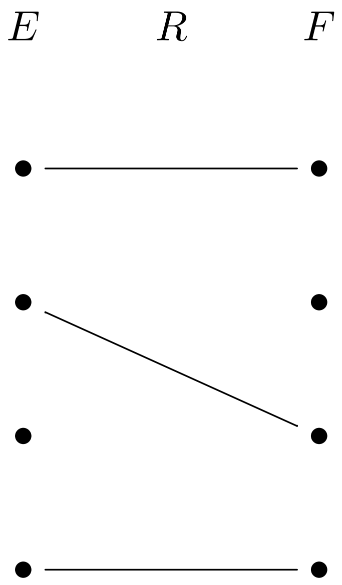
*Favourite*, from Drinkers to Beers is many-one.

A drinker has at most one favourite beer.

But a beer can be the favourite of any number of drinkers, including zero.

## One-One Relationships

Both many-one from E to F and many-one from F to E




**1-to-1 Constraints:**

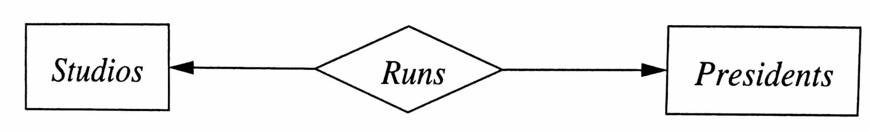
min-card(E,R) $=0$
max-card(E,R) $=1$

min-card(F,R) $=0$
max-card(F,R) $=1$

# Example: One-One Relationships

Example 1: 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).

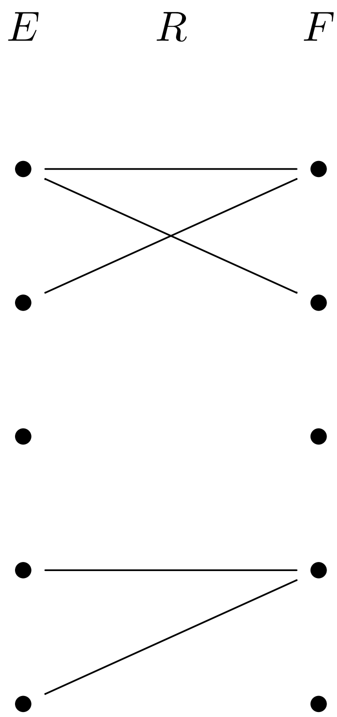
Example 2: Relationship $Runs$




These arrows mean “at most one”

# Many-Many Relationships

Neither many-one from E to F nor many-one from F to E




**N-to-N Constraints:**

min-card(E,R) = 0
max-card(E,R) = N

min-card(F,R) = 0
max-card(F,R) = N

## Example: Many-Many Relationships

An entity of either set can be connected to many entities of the other set.

Example: Binary relationship *Sells*
between entity sets *Bars* and *Beers*.

a bar sells many beers; a beer is sold by many bars.

## Representing Multiplicity (1)

Show a many-one relationship by an arrow entering the one side.

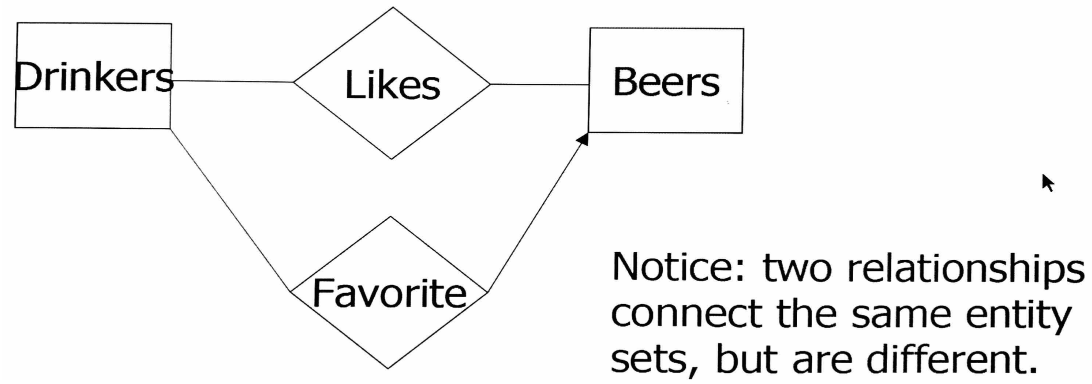
Show a one-one relationship by arrows entering both entity sets.

**Regular arrow = at most one**, i.e., each entity of the first set is related to at most one entity of the target set

**Rounded arrow = exactly one**, i.e., each entity of the first set is related to exactly one entity of the target set

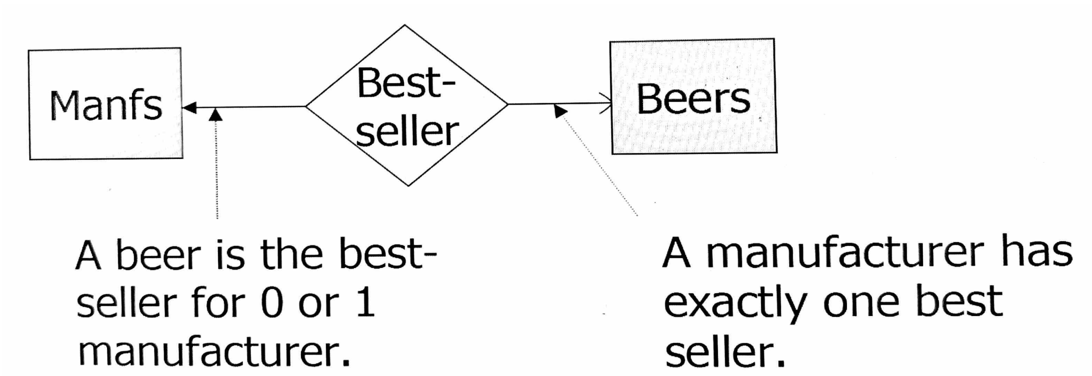
## Representing Multiplicity (2)

Example: Many-One Relationship *Favorite*




## Representing Multiplicity (3)

Example: One-One Relationship
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 beer manufacturer has to have a best-seller.



# Multiway Relationships (1)

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

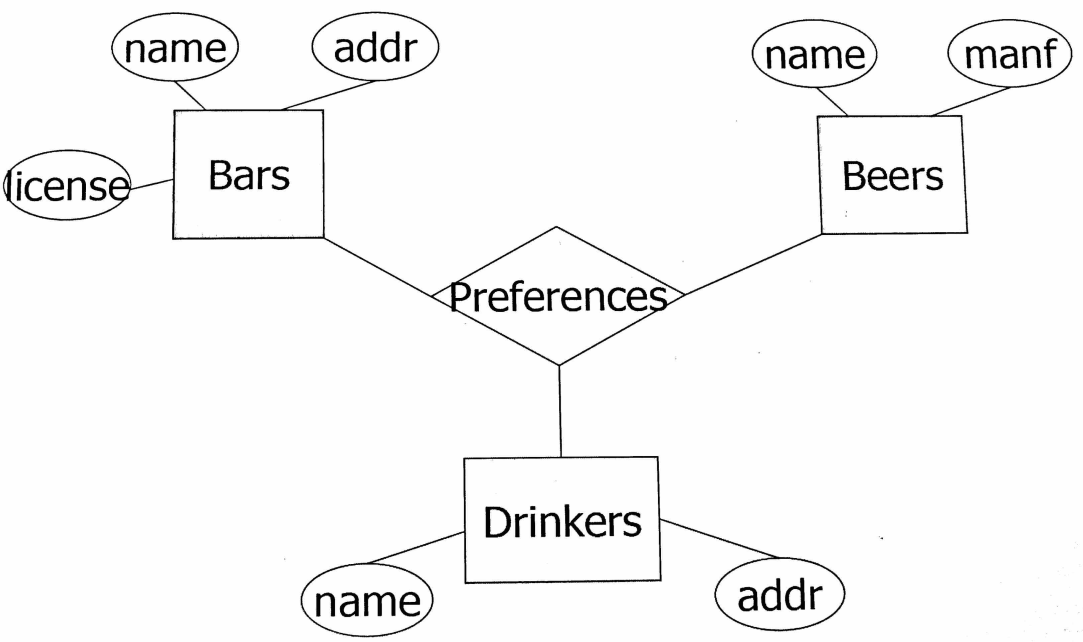
Suppose that drinkers will only drink certain beers at certain bars.

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

A 3-way relationship is much more suitable.

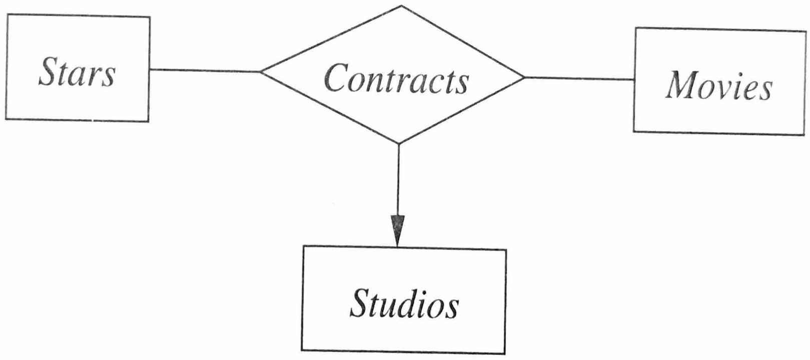
# Multiway Relationships (2)

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

# Is-A Hierarchies (1)

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

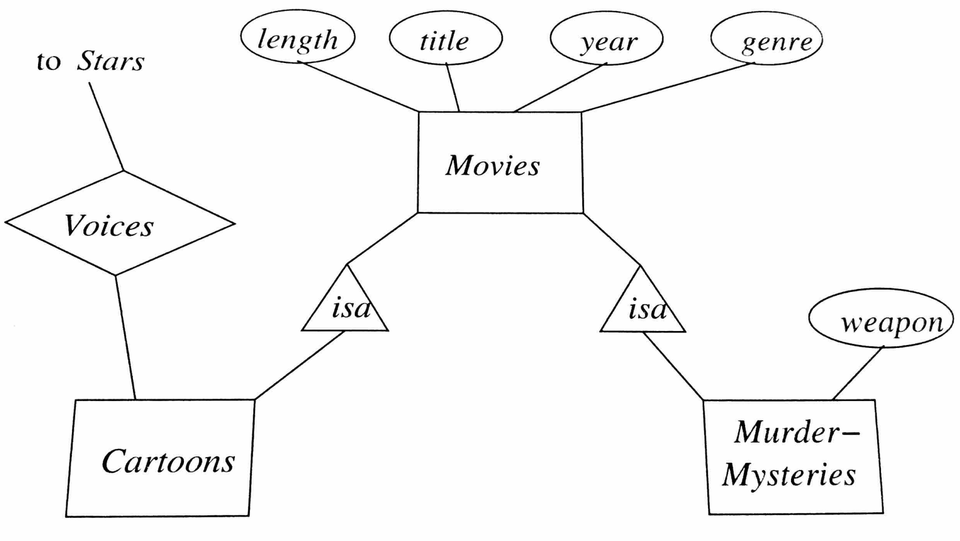
If we declare A Is-A B, every A entity is also considered to be a B entity.

## 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)

# Keys and Candidate Keys (1)

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

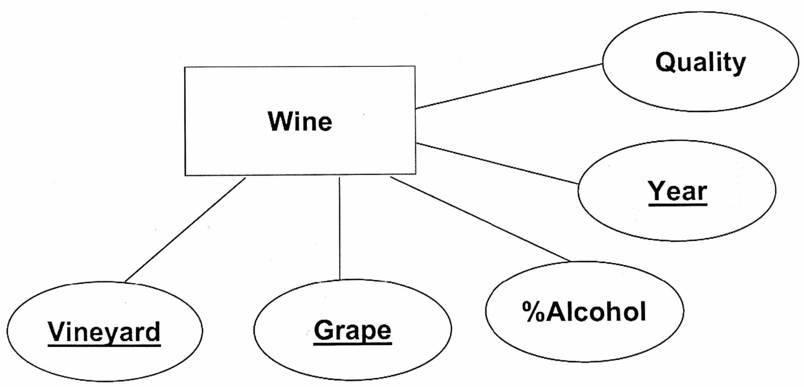
- Be a unique identifier
- Be **minimal** (under set inclusion)
  No proper subset of $\{Vineyard, Grape, Year\}$ is a key in the example below

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

Note that Candidate Keys are called just Keys in Ullman & Widom
Keys are called Super Keys (for Superset Keys) in Ullman & Widom

# Keys and Candidate Keys (2)

Example:




The three underlined attributes **together** form the key

# Summary of Conceptual Design

- Conceptual design follows requirements analysis,
  - Yields a high-level description of data to be stored
- ER 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 ER model.

# Summary of ER (1)

- Several kinds of integrity constraints can be expressed in the ER 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 ER model.
- Constraints play an important role in determining the best database design for an enterprise.

# Summary of ER (2)

- ER 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**.