The Entity Relationship Model

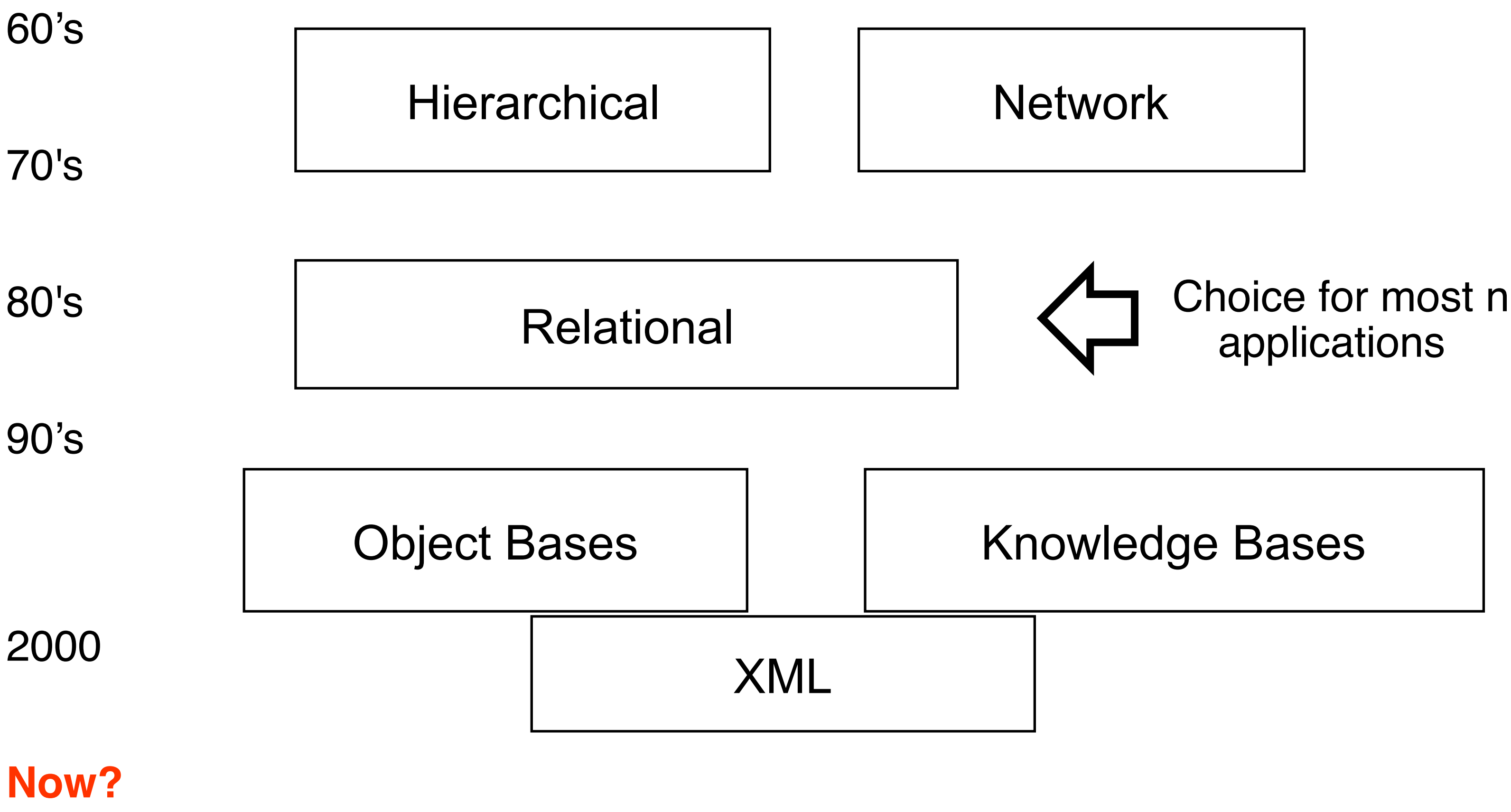
RDBMS

What is a Database Management System?

1. Manages very large amounts of data.
2. Supports efficient access to very large amounts of data.
3. Supports concurrent access to very large amounts of data.
   Example: bank and its ATM machines.
4. Supports secure, atomic access to very large amounts of data.
   Example: Contrast two people editing the same UNIX file – last to write “wins” – with the problem if two people deduct money from the same account via ATM machines at the same time – new balance is wrong whichever writes last.

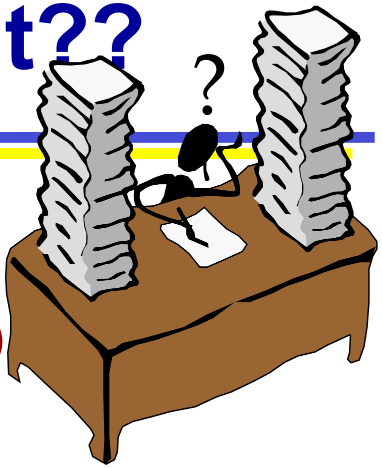
Data Models: Historic Overview

A data model says what information is to be contained in a database, how the information will be used, and how the items in the database will be related to each other



Now?

Why Study Data Management??



- Shift from *computation* to *information*
  - at the “low end”: scramble to webspace (a mess!)
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic

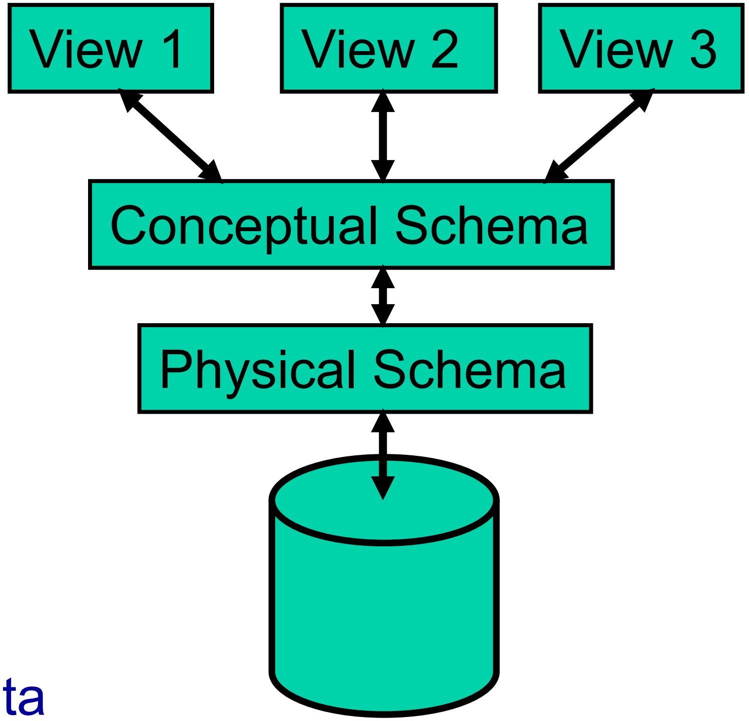
Levels of Abstraction

Many *views*,

single *conceptual schema*

and *physical schema*.

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes how data are stored



Example: University Database

- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*
- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on student id (sid).

Relational Model Example

Relational model is based on tables…

e.g. *CustomerTable* in a Bank Database

acct #	name	balance
12345	Tasos	$1000.2
34567	Yannis	$285.48
…	…	…

Today used in *most* DBMS's…

The DBMS Marketplace

- Relational DBMS companies – Oracle, Sybase – are among the largest software companies in the world.
- IBM offers its relational DB2 system. With IMS, a nonrelational system, IBM is by some accounts the largest DBMS vendor in the world.
- Microsoft offers SQL-Server, plus Microsoft Access for the cheap DBMS on the desktop, answered by “lite” systems from other competitors.
- Relational companies also challenged by “object-oriented DB” companies.
- But countered with “object-relational” systems, which retain the relational core while allowing type extension as in OO systems.

Structured Query Language (SQL)

acct #	name	balance
12345	Tasos	$1000.2
34567	Yannis	$285.48
…	…	…

SELECT *name*
FROM *CustomerTable*
WHERE *balance* >= 500

SELECT *acct#*
FROM *CustomerTable*
WHERE *name* = “Yannis” **AND**
balance >= 200

Three Aspects to Studying DBMS's

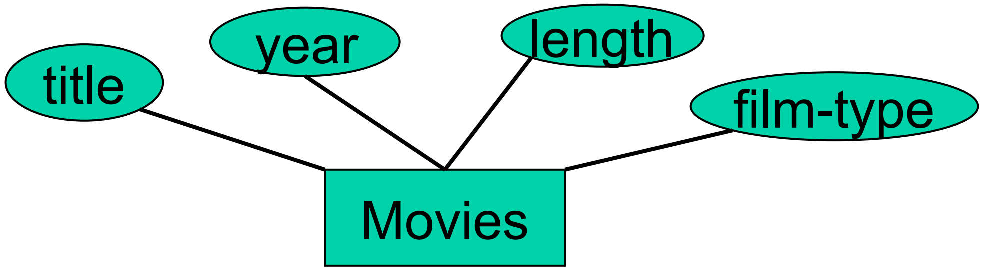
- Modeling and design of databases Management System.
  - Allows exploration of issues before committing to an implementation.
- Programming: queries and DB operations like update.
  - SQL = “intergalactic dataspeak.”
  - Is it enough?
- Data Management System Implementation.
  - Is a database enough? Or we need something more?

The Entity Relationship Model

Entity/Relationship Model

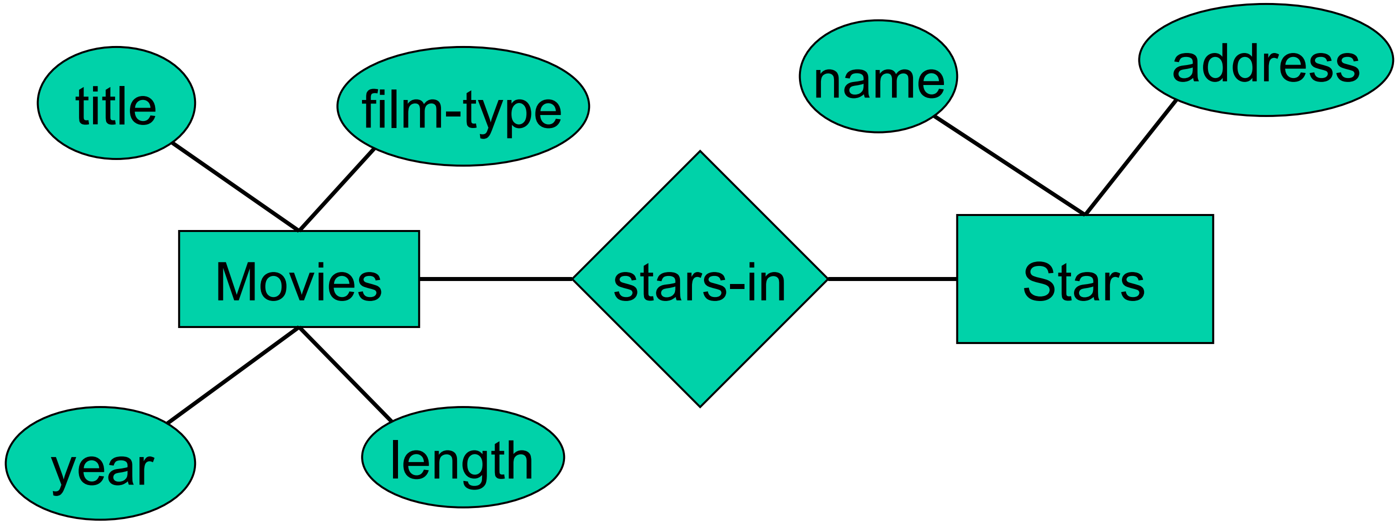
Main idea: Diagrams to represent designs.

- *Entity* like object, = “thing”
- *Entity set* like class = set of “similar” entities/objects.
- *Attribute* = property of entities in an entity set, similar to fields of a struct.
- In diagrams, entity set → rectangle; attribute → oval.



Relationships

- Connect two or more entity sets.
- Represented by diamonds.



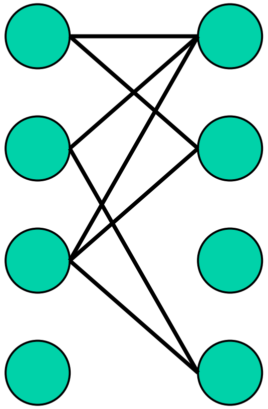
Relationship Set

- Think of the “value” of a relationship set as a table.
- One column for each of the connected entity sets.
- One row for each list of entities, one from each set, that are connected by the relationship.
- Example: stars-in relationship set

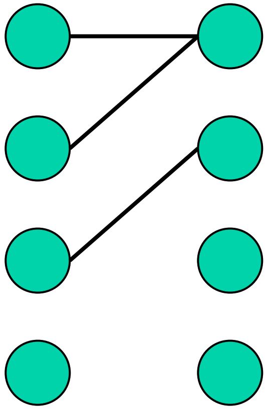
Movies	Stars
Basic Instinct	Sharon Stone
Total Recall	Arnold Schwarzenegger
Total Recall	Sharon Stone
…	…

Why are relationship sets important?

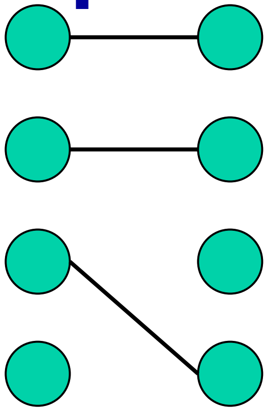
Multiplicity of Relationships



Many-many



Many-one



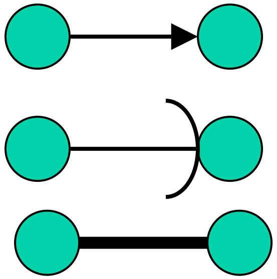
One-one

Representation of Many-One:

Arrow pointing to "one."

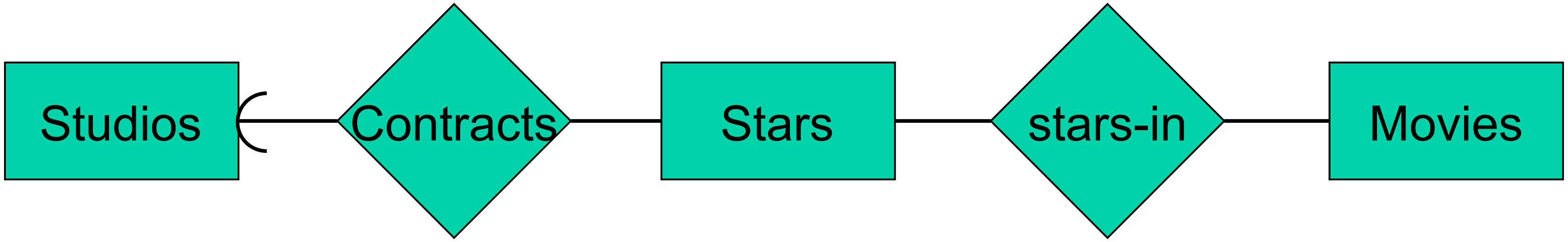
Rounded arrow = "exactly one."

… or simply



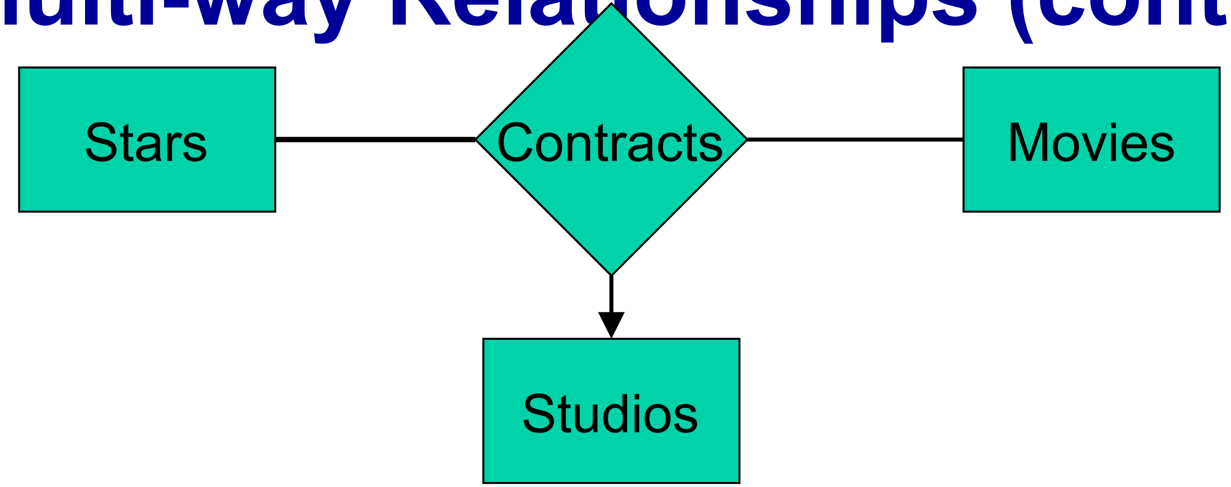
Multi-way Relationships

Usually binary relationships suffice.



- However, there are some cases where three or more E.S. must be connected by one relationship.
- Example: *Stars* have *Contracts* with *Studios* for particular *Movies*.

Multi-way Relationships (cont.)



Stars	Movies	Studios
Sharon Stone	Basic Instinct	Sony
Arnold Schwarzenegger	Total Recall	Columbia
Sharon Stone	Total Recall	Columbia
…	…	…

Multi-way Relationships (cont.)

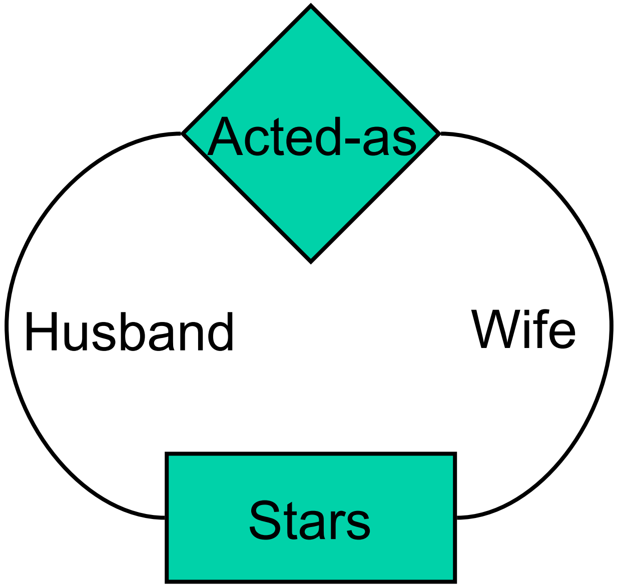
Notice arrow convention for multi-way relationships:

“all other E.S. determine one of these.”

- Not sufficiently general to express any possibility.
- Assume that a studio, say, depended only on the movie, then we could use two 2-way relationships: *Studios-Movies* and *Movies-Stars*.
- Or better: just make *studio* an attribute of *Movies*.

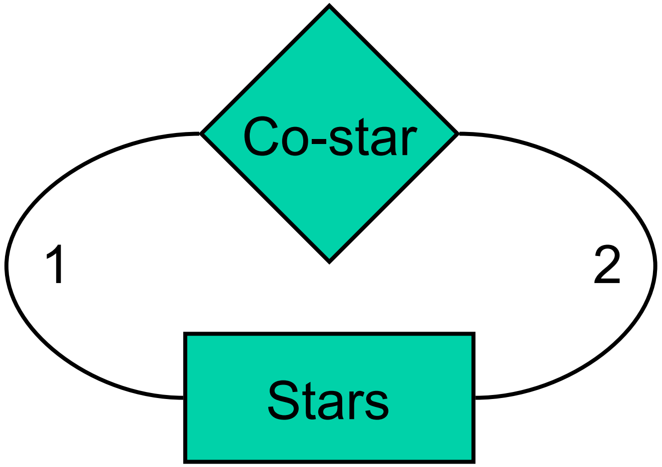
Roles in relationships

Sometimes an E.S. participates more than once in a relationship. Thus, we need to label edges with *roles* to distinguish.



Husband	Wife
Arnold Schw	Sharon Stone
Arnold Schw	Jamie Lee Curtis
…	…

Roles in relationships (cont.)



Actor_1	Actor_2
Tom Cruise	Nicole Kidman
Nicole Kidman	Tom Cruise
…	…

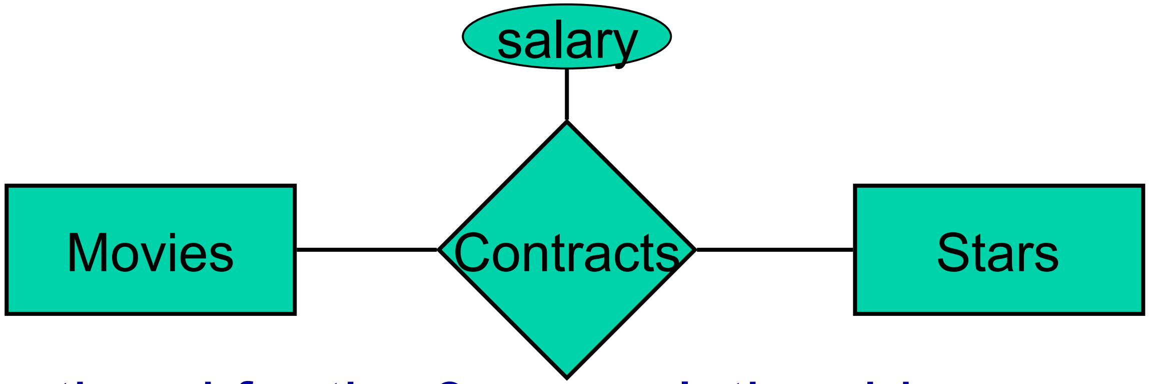
Note: *Co-star* is symmetric, *Acted-as* was not.

There is no way to say “symmetric” in E/R.

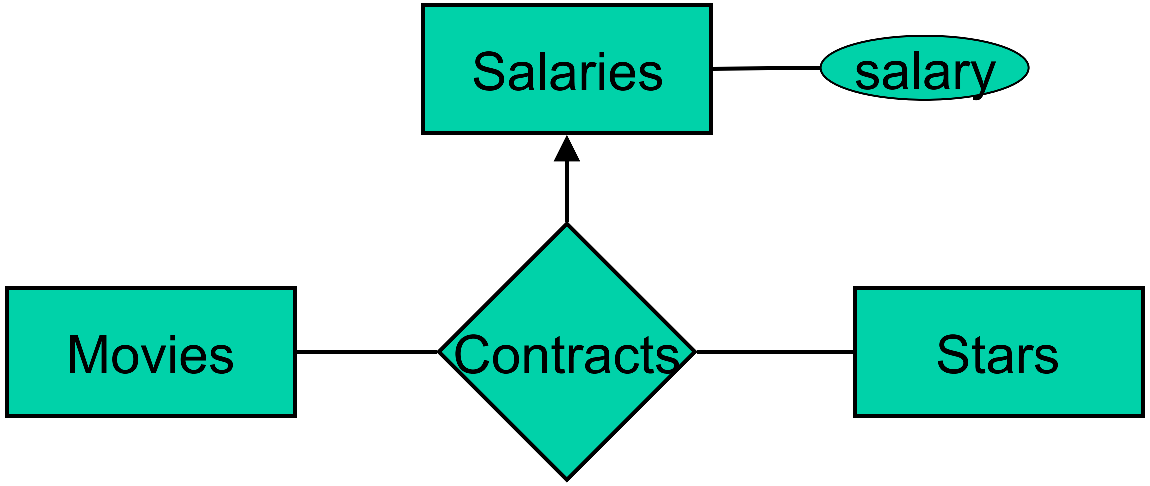
Design Question:

What if we replace *Husband* and *Wife* by one role, e.g., *couple*?

Attributes on Relationships



… a shorthand for the 3-way relationship:

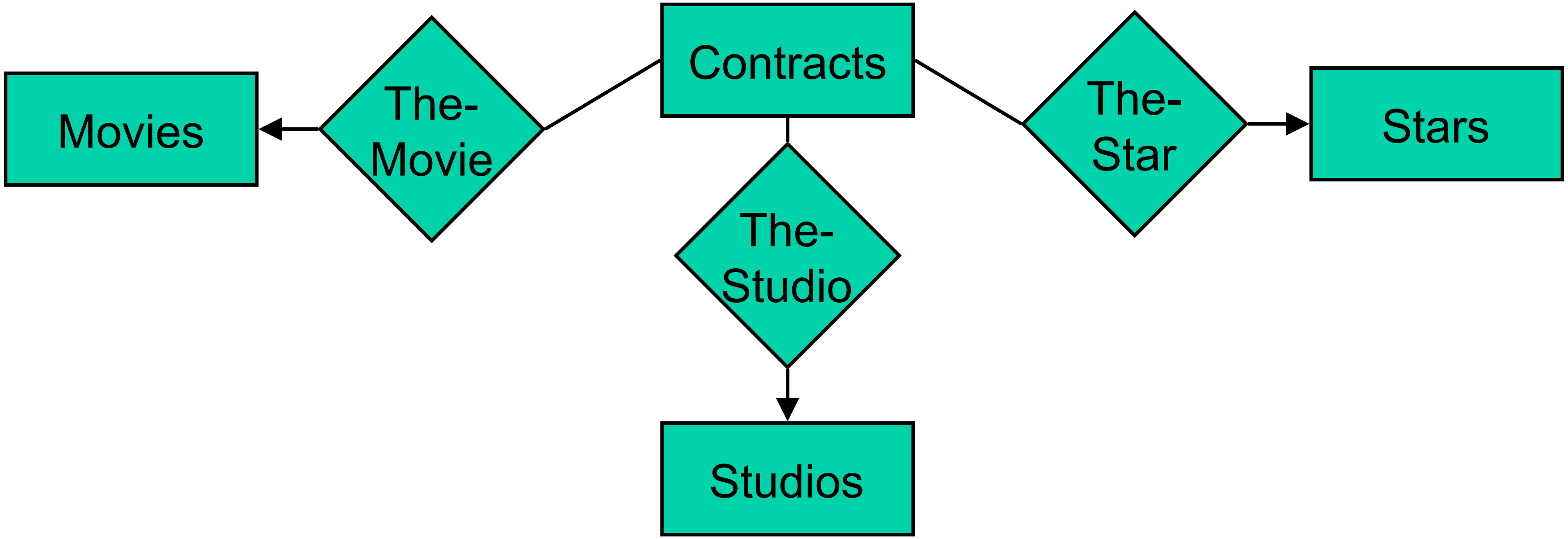


Converting Multi-way to 2-Way

Baroque in E/R, but necessary in certain “object-oriented”

models.

- Create a new E.S. to represent rows of a relationship set.
- Add many-one relationships from the new E.S. to the E.S.’s that participated in the original relationship.



More Design Issues

1. Subclasses.
2. Keys.
3. Weak entity sets.

Subclasses

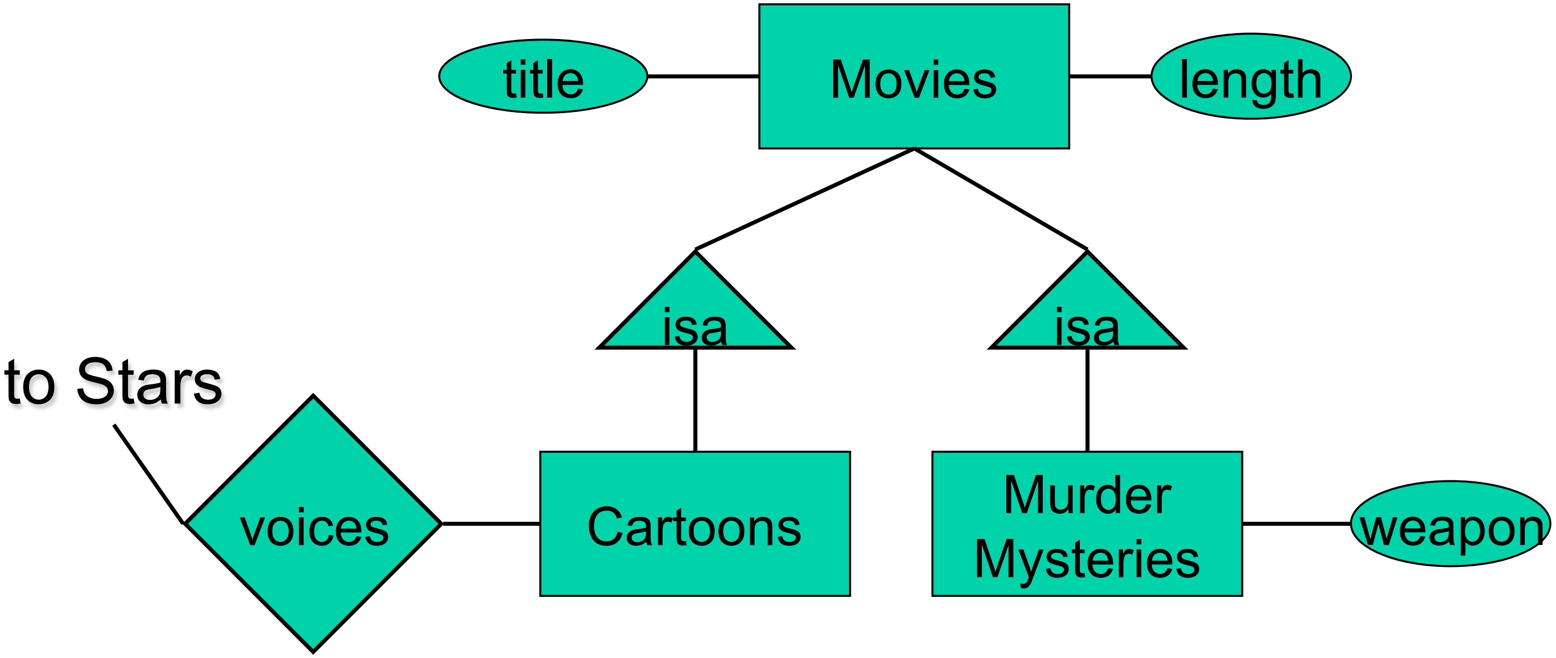
Subclass = special case = fewer entities =

more properties.

- Example: A cartoon is a kind of movie. In addition to the properties (= attributes and relationships) of movies, there is a *voices* attribute for cartoons.

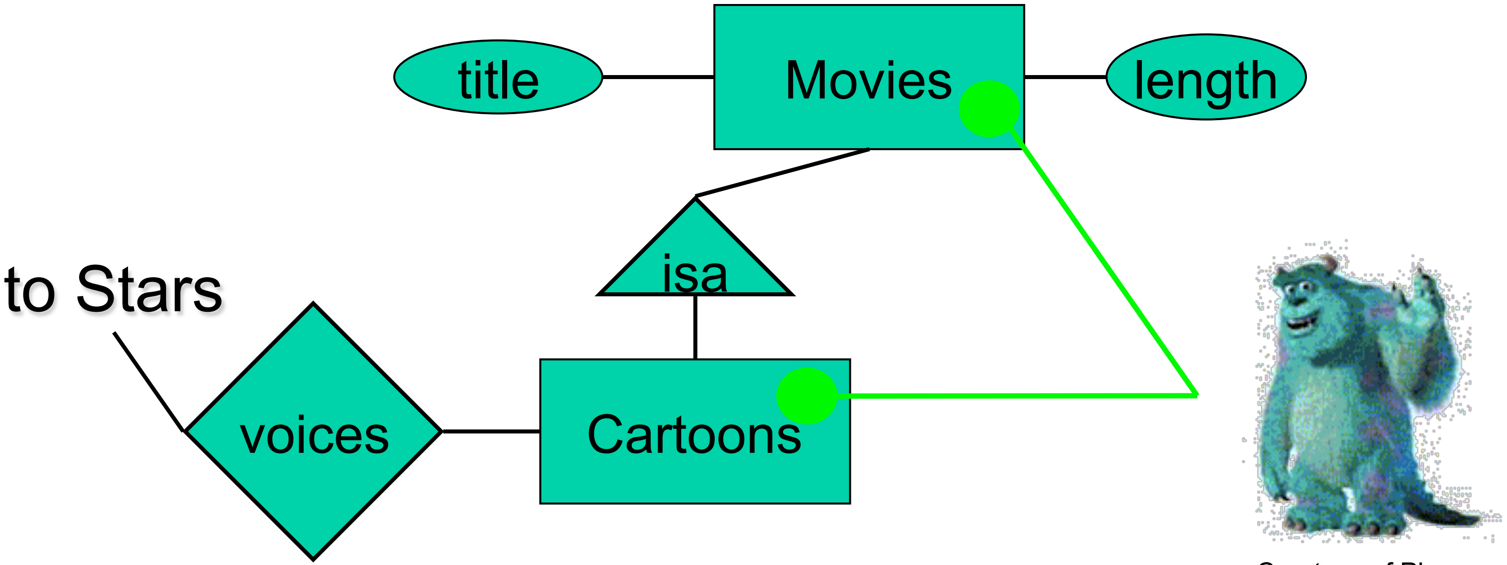
E/R Subclasses

- Subclasses form a tree (no multiple inheritance).
- *isa* relationships indicate the subclass relation
- they are one-one



Different Subclass Viewpoints

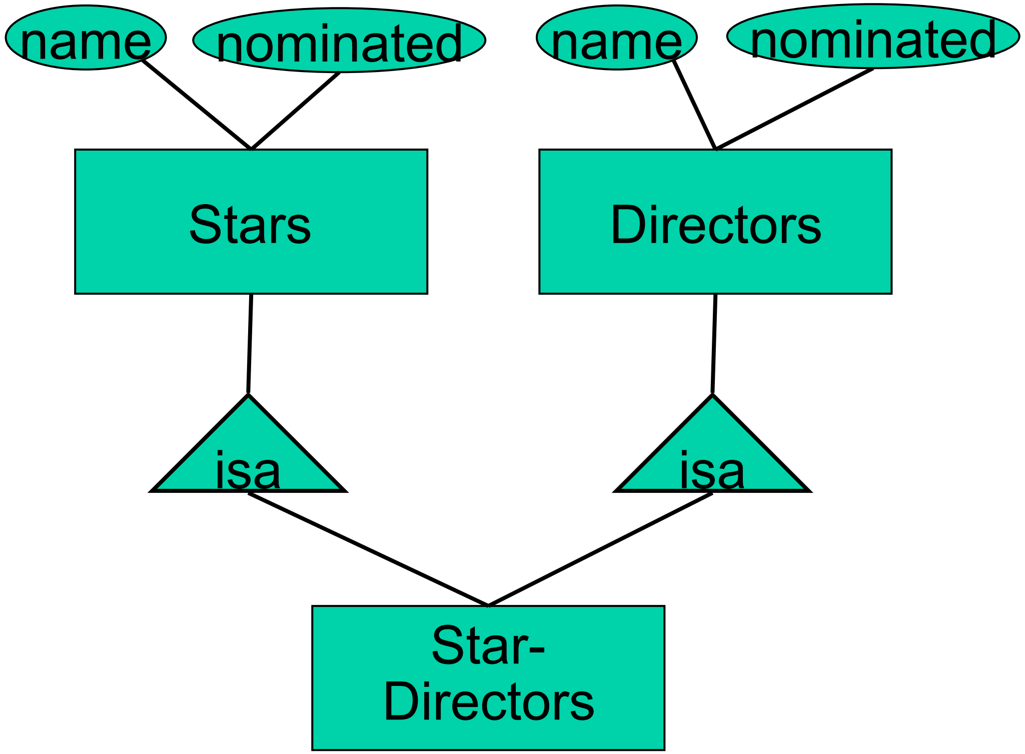
- *E/R viewpoint*: An entity has a *component* in each entity set to which it logically belongs. Its properties are the union of the properties of these E.S.
- *Object-oriented viewpoint*: An object (entity) belongs to exactly one class. It *inherits* properties of its superclasses.



Courtesy of Pixar

Multiple Inheritance

Theoretically, an E.S. could be a subclass of several other entity sets.



Problems

How should conflicts be resolved?

- Example: nominated means Oscar nominated as an actor for *Stars*, and as a director for *Directors*. What does it mean for *Star-Directors*?
- Need ad-hoc notation to resolve meanings.
- In practice, we shall assume a tree of entity sets connected by *isa*, with all “isa’s” pointing from child to parent.

Keys

Consider an entity set *E*. A *key* is a set of attributes *K*

of *E* such that, given two entities e_1 and e_2 of *E*,

e_1 and e_2 cannot have identical values in *K*

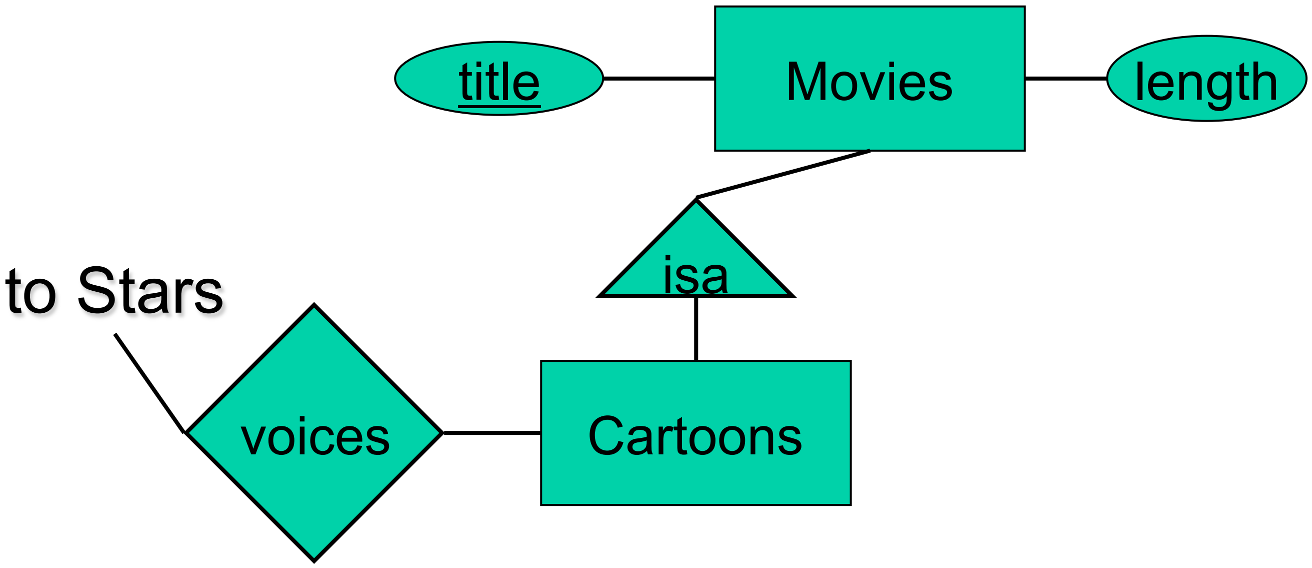
- In E/R model, every E.S. must have a key.
  - It could have more than one key, but one set of attributes is the “designated” key.
- In E/R diagrams, you should underline all attributes of the designated key.

Example

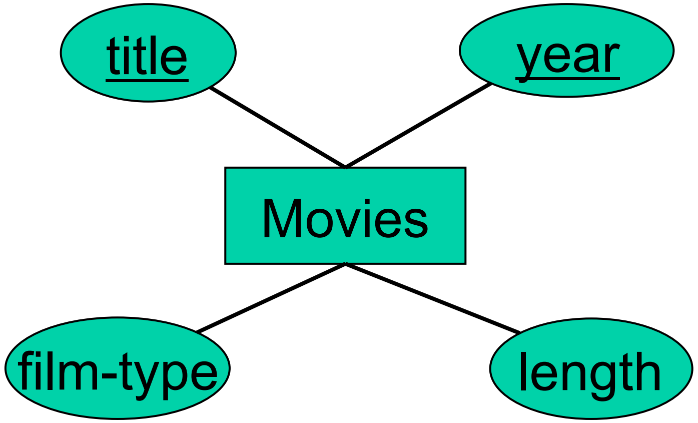
Suppose *title* is the key for *Movies*.

It makes sense to use *title* as key for *Cartoons* also.

Thus, we use root key as key for all subclasses.



Example: A Multi-attribute Key



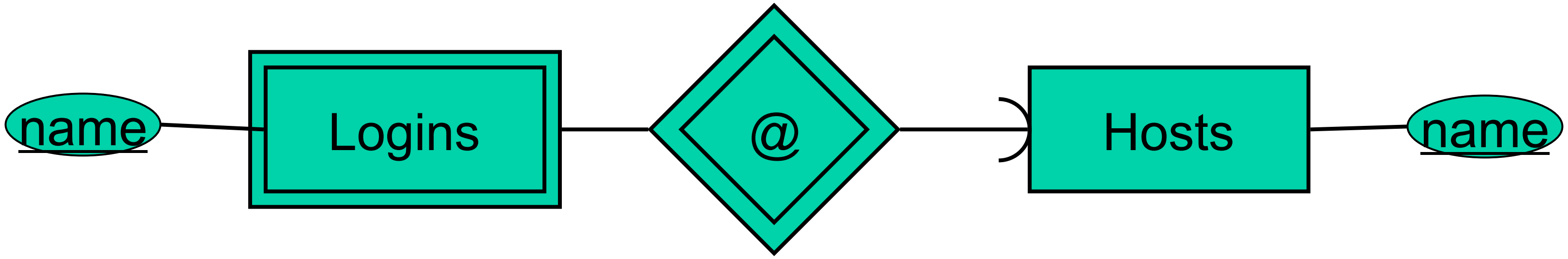
Possibly, the combination of *title* + *length* also forms

a key, but we have not designated it as such.

Example: Logins (Email Addresses)

Login name = user name + host name, e.g. velgias@disi.unitn.eu

- A “login” entity corresponds to a user name on a particular host. The *Login* entity doesn’t record the host, just the user name, e.g., tasos.
- Key for a login = the user name at the host (which is unique for that host only) + the IP address of the host (which is unique globally).



- Design issue: Under what circumstances could we simply make login-name and host-name be attributes of logins, and dispense with the weak E.S.?

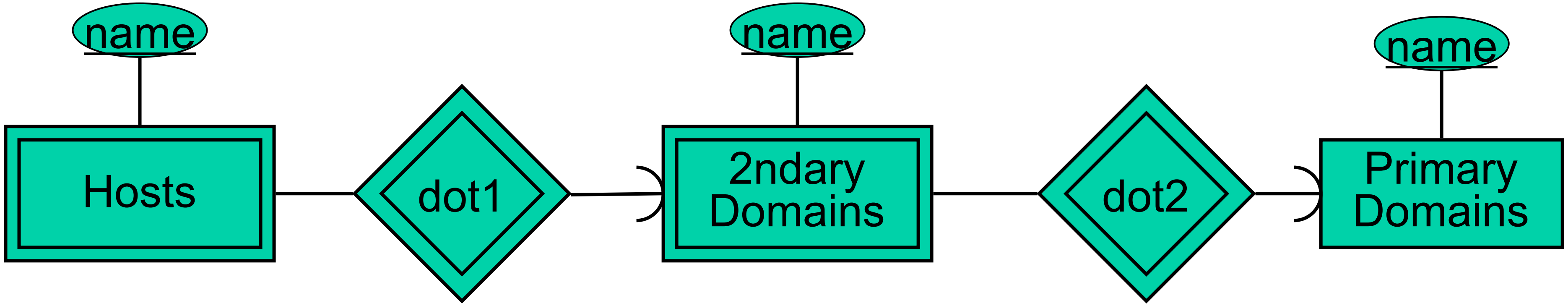
Weak Entity Sets

Consider an entity set *E.* It's possible that *E's* key comes not (completely) from its own attributes, but also from the keys of one or more E.S.'s to which *E* is linked (through a *supporting* many-one relationship).

- Entity set *E* is called a *weak* E.S.
- It is represented by putting double rectangle around *E* and a double diamond around each supporting relationship.
- Many-one-ness of supporting relationship (includes 1-1) essential.
  - With many-many, we wouldn't know which entity provided the key value.
- "Exactly one" also essential, or else we might not be able to extract key attributes by following the supporting relationship.

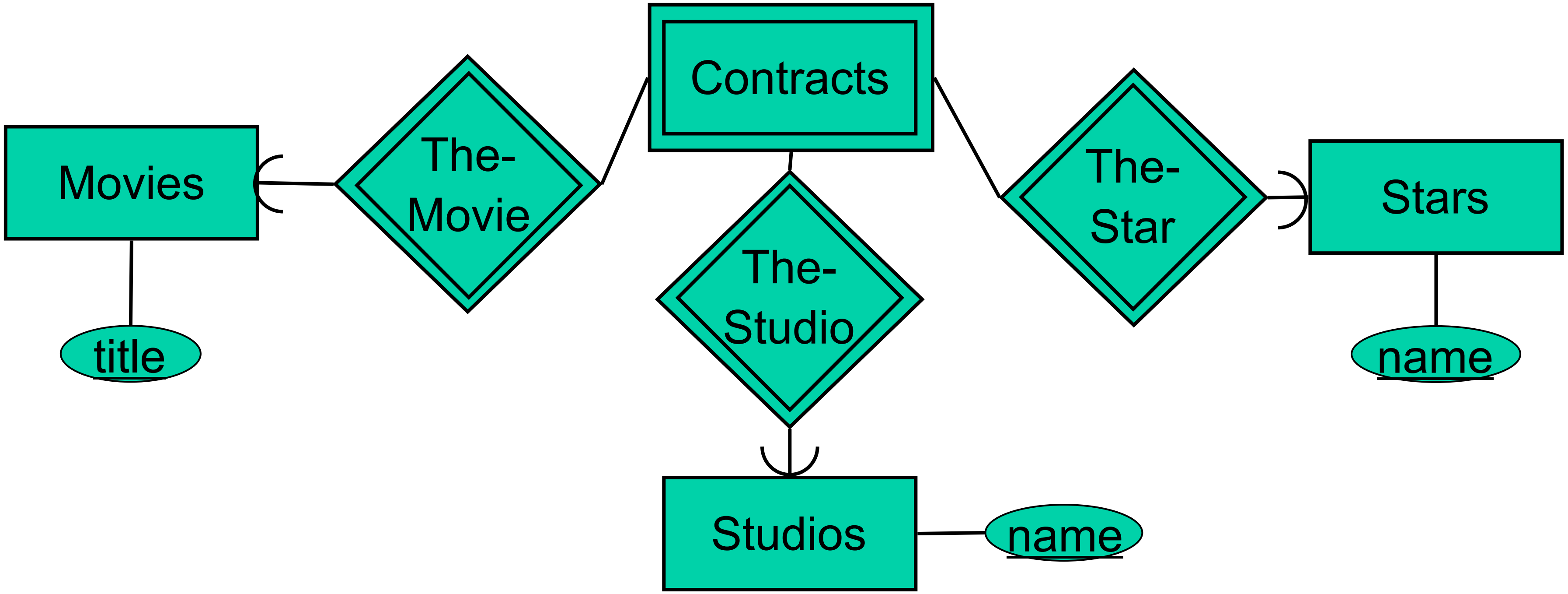
Example: Chain of “Weakness”

Consider IP addresses consisting of a primary domain (e.g., edu), subdomain (e.g., cdf.toronto), and host (e.g., eddie).



- Key for primary domain = its name.
- Key for secondary domain = its name + name of primary domain.
- Key for host = its name + key of secondary domain = its name + name of secondary domain + name of primary domain.

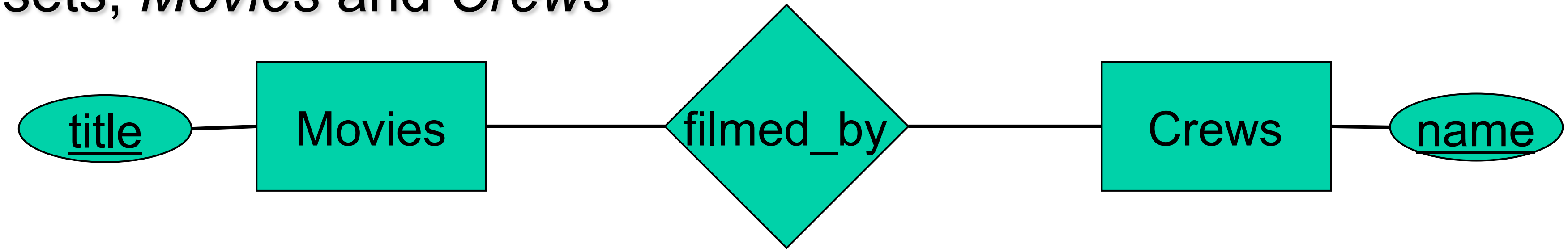
All “Connecting” Entity Sets Are Weak



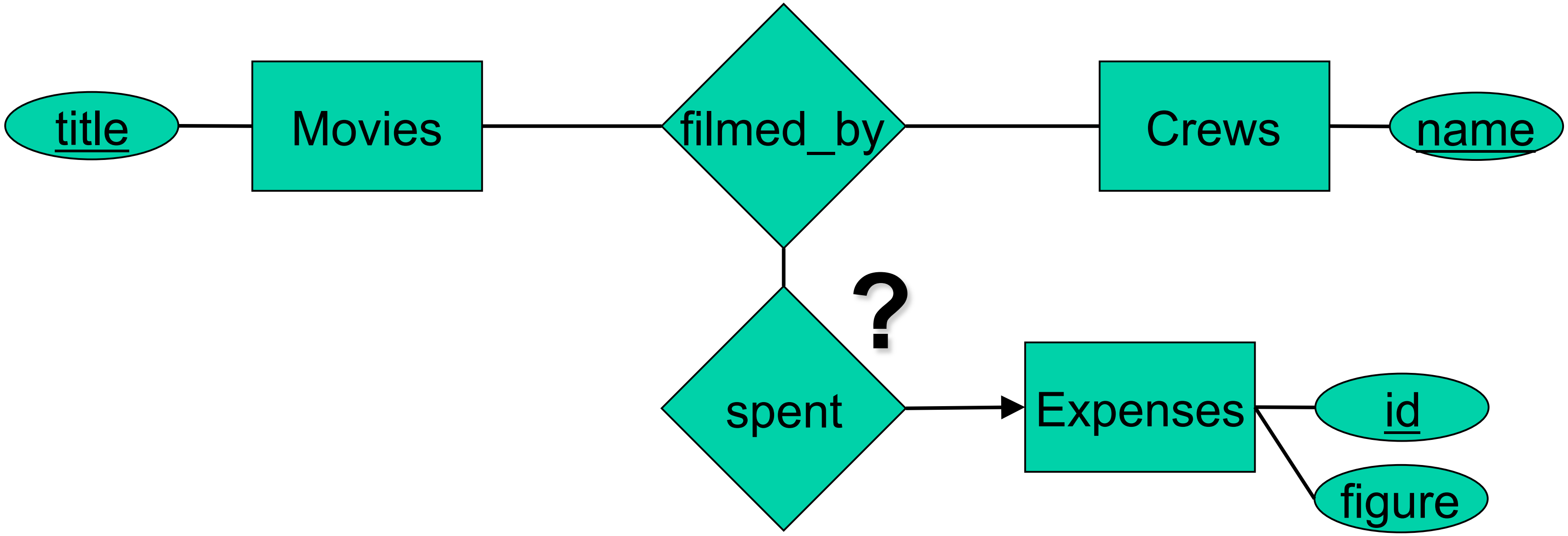
- If *Movies* determine *Studios*, we can omit *Studios name* from the key, and remove the double diamond from *Studios*.
- Better: *Studios* is attribute of *Contracts*.

Aggregation

Consider a relationship, *filmed_by*, between two entity sets, *Movies* and *Crews*

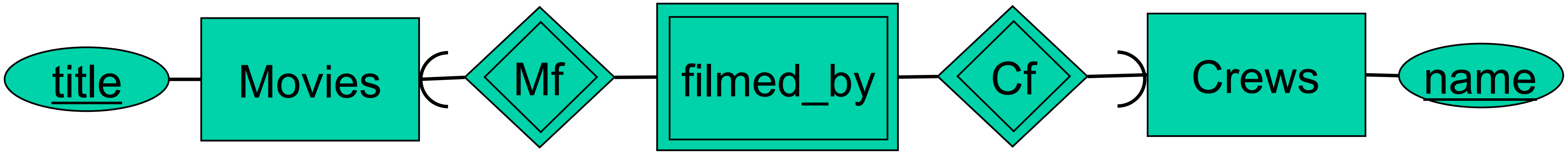


How can we add *Expenses* to the mix?

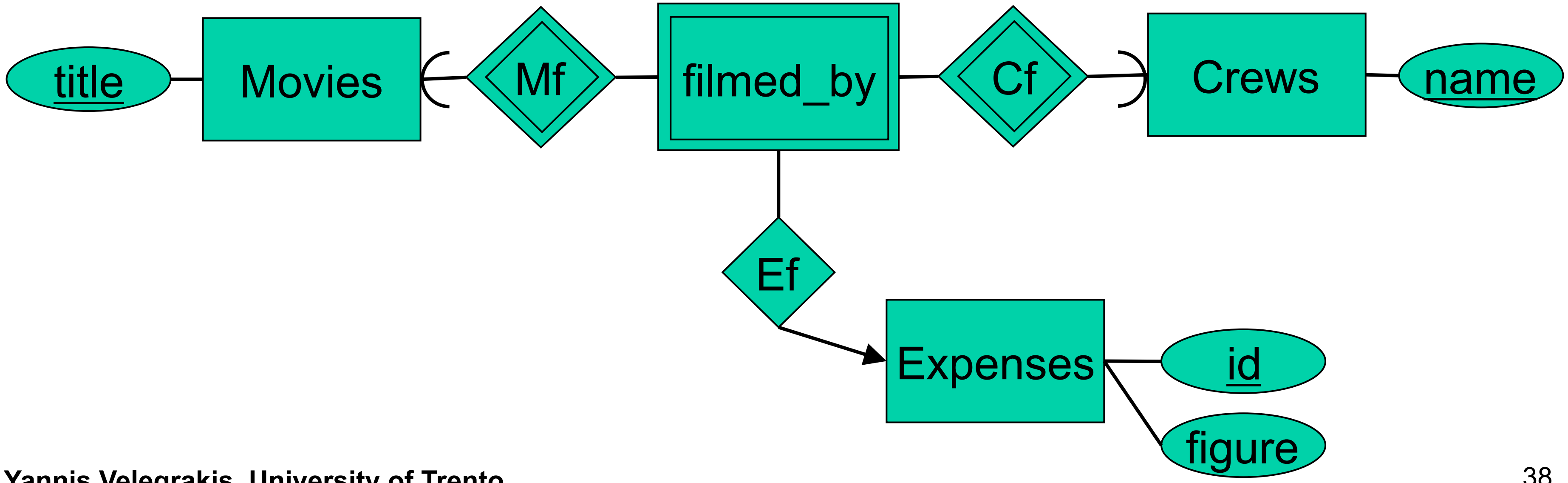


Simulating Aggregation

Solution: convert *filmed_by* into a weak entity set.



And then add *Expenses* to it…



Design Principles

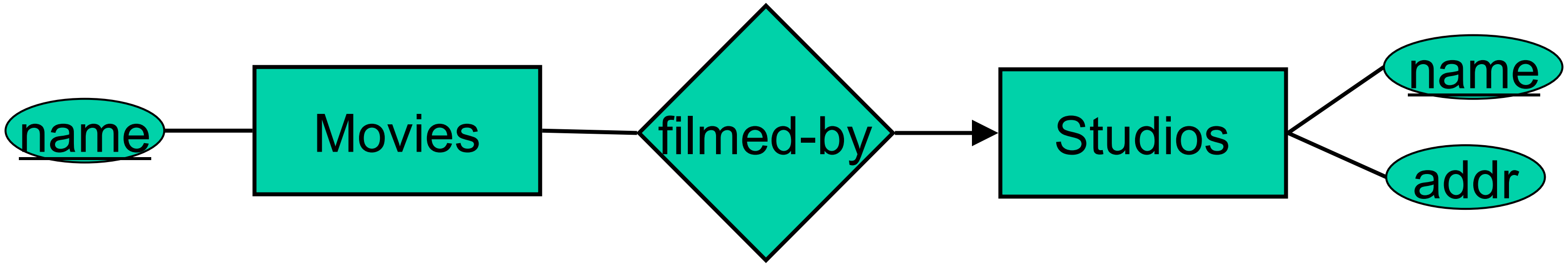
Setting: client has (possibly vague) idea of what he/she wants. You must design a database that represents these thoughts and only these thoughts.

Important note:

Avoid redundancy,i.e., saying the same thing more than once. It wastes space and encourages inconsistency.

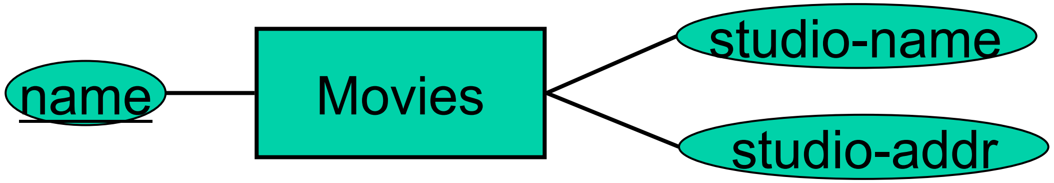
Example

Good:

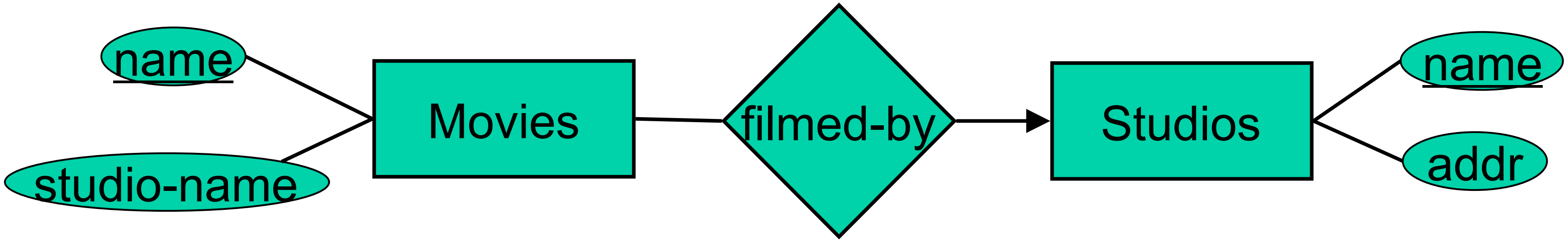


Example

This is a bad design…



Yeap! That's also a bad design…



Use Schema to Enforce Constraints

The design *schema* should enforce as many

constraints as possible.

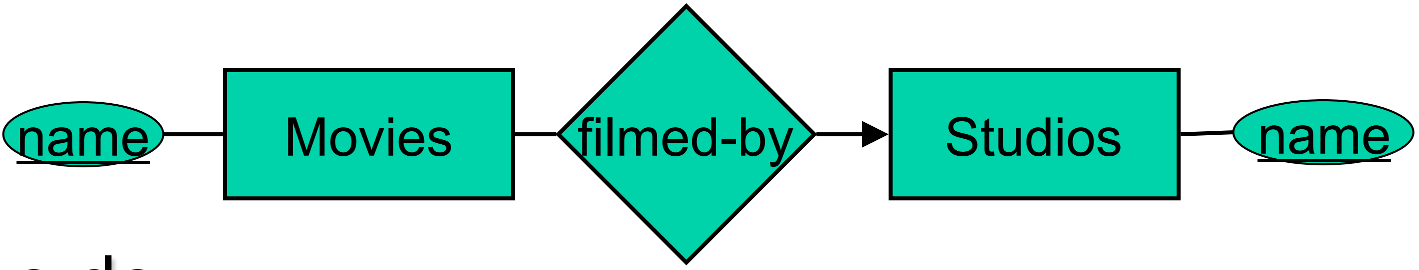
- Don't rely on future data to follow assumptions.

Example:

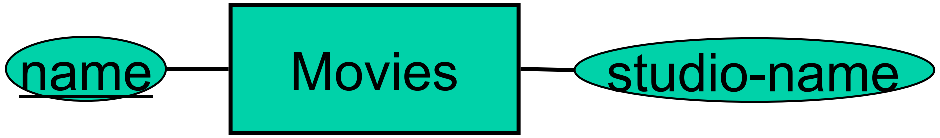
- If a user must associate only one director with a movie, don't allow sets of directors and count on users to enter only one director per movie.

Entity Sets Vs. Attributes

- You may be unsure which concepts are worthy of being entity sets, and which are handled more simply as attributes.
- Especially tricky for the class design project, since there is a temptation to create needless entity sets to make project "larger."



Which one do you like?!!?



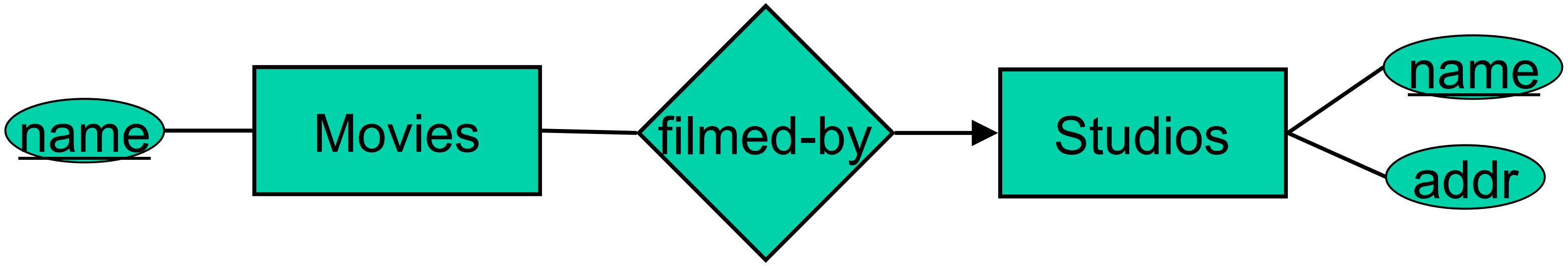
Intuitive Rule for E.S. Vs. Attribute

Make an entity set only if it either:

1. Is more than a name of something; *i.e*., it has non-key attributes or it has relationships with a number of different entity sets, or
2. Is the “many” in a many-one relationship.

Example

The following design illustrates both points:



- *Studios* deserves to be an E.S. because we record *addr*, a non-key attribute.
- *Movies* deserves to be an E.S. because it is at the “many” end.

Don't Overuse Weak E.S.

- There is a tendency to feel that no E.S. has its entities uniquely determined without following some relationships.
- However, in practice, we almost always create unique ID's to compensate: social-security numbers, OHIP's, etc.
- The only times weak E.S.'s seem necessary are when:
  a) We can't easily create such ID's; e.g., no one is going to accept a “species ID” as part of the standard nomenclature (species is a weak E.S. supported by membership in a genus).
  b) There is no global authority to create them, *e.g.*, crews and studios.

The Relational Model

Relational Model

- Table = relation.
- Column headers = attributes.
- Row = tuple

Relation stars-in

Movies	Stars
Basic Instinct	Sharon Stone
Total Recall	Arnold Schwarzenegger
Total Recall	Sharon Stone
…	…

- Relation schema = name(attributes) + other structure info., e.g., keys, other constraints.
  Example: *stars-in*(*movie-name*, *star-name*)
  - Order of attributes is arbitrary, but in practice we need to assume the order given in the relation schema.
- Relation instance is current set of rows for a relation schema.
- Database schema is a collection of relation schemas.

Why Relations?

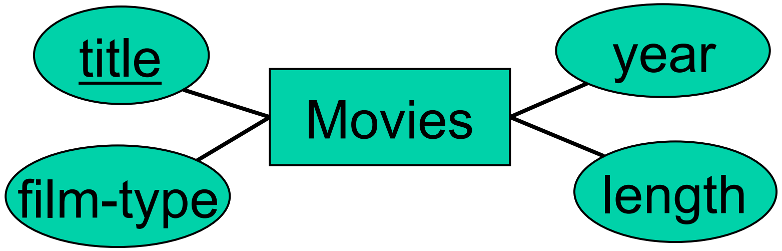
- Very simple model.
- *Often* a good match for the way we think about our data.
- Abstract model that underlies SQL, the most important language in DBMS's today.
  - But SQL uses "bags" while the abstract relational model is set-oriented.

Relational Design

We start with an E/R design. Then, the simplest approach (but not always best) is :

- convert each E.S. to a relation and
- convert each relationship to a relation.

Example:



Movies(*title*, *year*, *length*, *film-type*)

Note: E.S. attributes become relational attributes

Keys in Relations

An attribute (or set of attributes) *K* is a *key* for a relation *R* if we expect that in no instance of *R* will two different tuples agree on all the attributes of *K*.

We indicate a key by underlining the key attributes.

(pretty much what we did in E/R)

Example:

If *title* is a key for *Movies, then*

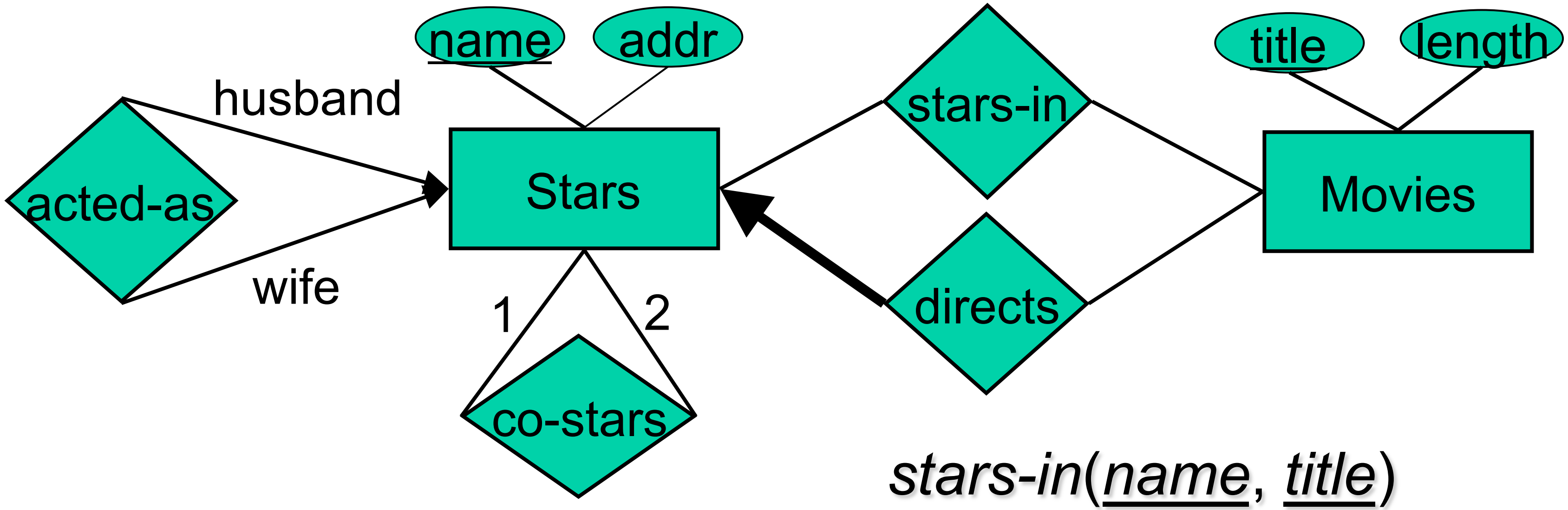
Movies(*title*, *year*, *length*, *film-type*)

From E/R Relationships to Relations

The constructed relation has the following attributes:

- The *key* attributes of each E.S. that participates in the relationship.
- Any attributes that belong to the relationship itself.
- Renaming attributes is necessary if an E.S. has multiple roles in the relationship

From E/R Relationships to Relations (cont.)



stars-in(*name*, *title*)
directs(*title*, *name*)
*acted-as**(*husband*, *wife*)
co-stars(*name1*, *name2*)

(*) For *acted-as*, we can choose either *husband* or *wife* as key

Combining Relations

Common case: Combine relation for an E.S. *E* with the relation for some many-one relationship from *E* to another E.S. *F*

Example: Combine *Movies*(title, length) with *directs*(title, name) to get *Movies-new*(title, *length*, *director-name*).

However danger in pushing this idea too far: redundancy

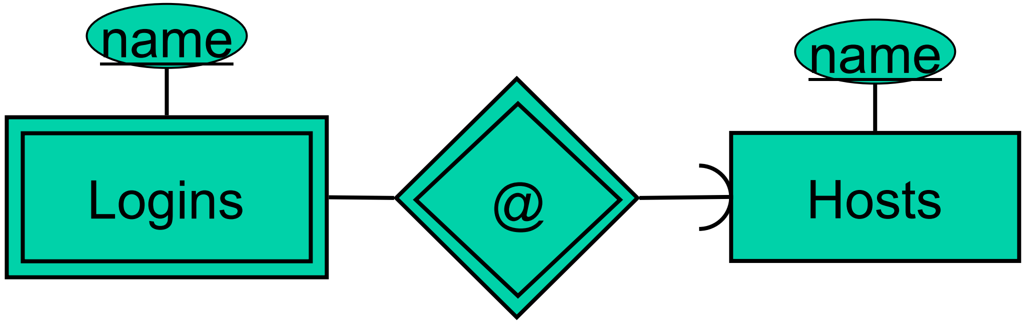
Example: Combine *Movies*(title, length) with *stars-in*(name, title) to get *Movies-stars*(title, name, *length*,)

title	name	length
Star Wars II	Ewan McGregor	142
Star Wars II	Natalie Portman	142

Weak Entities & Relations

- Relation for a weak E.S. must include its full key (*i.e.*, attributes of related entity sets) as well as its own attributes.
- A supporting (double-diamond) relationship yields a relation that is actually redundant and should be deleted from the database schema.

Weak Entity Sets & Relationships to Relations (cond.)



Hosts(*<u>name</u>*)

Logins(*<u>name</u>*, *<u>hname</u>*)

At(*<u>lname</u>*, *<u>hname</u>*, *hname2*)

- In *At*, *hname* and *hname2* must be the same host, so delete one of them.
- Then, *Logins* and *At* become the same relation; delete one of them.
- In this case, *Hosts*' schema is a subset of Logins' schema. Delete *Hosts*?

Subclasses to Relations

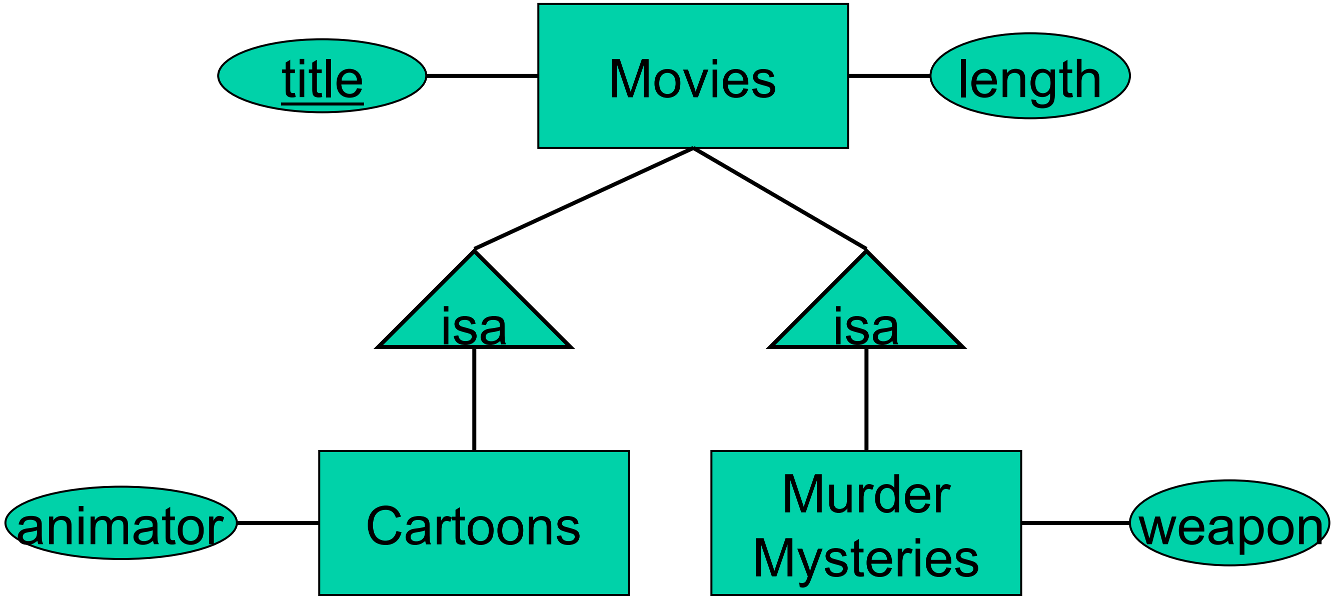
Three approaches:

1. Object-oriented: each entity is in one class. Create a relation for each class, with all the attributes for that class.
   - Don't forget inherited attributes.

2. E/R style: an entity is in a network of classes related by *isa*. Create one relation for each E.S.
   - An entity is represented in the relation for each subclass to which it belongs.
   - Relation has only the attributes attached to that E.S. + key.

3. Use nulls. Create one relation for the root class or root E.S., with all attributes found anywhere in its network of subclasses.
   - Put NULL in attributes not relevant to a given entity.

Example


title
Movies
length
isa
isa
animator
Cartoons
Murder Mysteries
weapon

Subclasses to Relations: An Example

OO Style:

Title	*Length*	*Weapon*
Episode I	133	Lightsaber
…	…	

Title	*Length*	*Animator*
Monsters	92	Yannis
…	…	

Title	*Length*
MXXXXX	92
…	…

E/R Style:

Title	*Length*
Episode I	133
Monsters	92
…	…

Title	*Weapon*
Episode I	Lightsaber
…	…

Title	*Animator*
Monsters	Yannis
…	…

Using *Nulls*:

Title	*Length*	*Weapon*	*Animator*
Episode I	133	Lightsaber	Null
Monsters	92	Null	Yannis