Chapter 4
High-Level Database Models

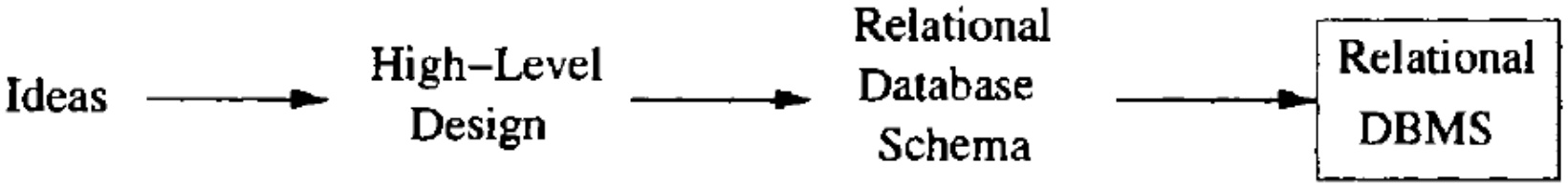
Université Grenoble Alpes

Bahareh Afshinpour

bahareh.afshinpour@univ-grenoble-alpes.fr

Main reference:
A First Course in Database Systems (and associated material) by J. Ullman and J. Widom, Prentice-Hall

- In practice, it is often easier to start with a higher-level model and then convert the design to the relational model.



The database modeling and implementation process

- There are several options for the notation in wich the design is expressed.
  - Entity-relationship diagram
  - UML (class diagram)
  - ODL(object description language)

Entity-Relationship Model

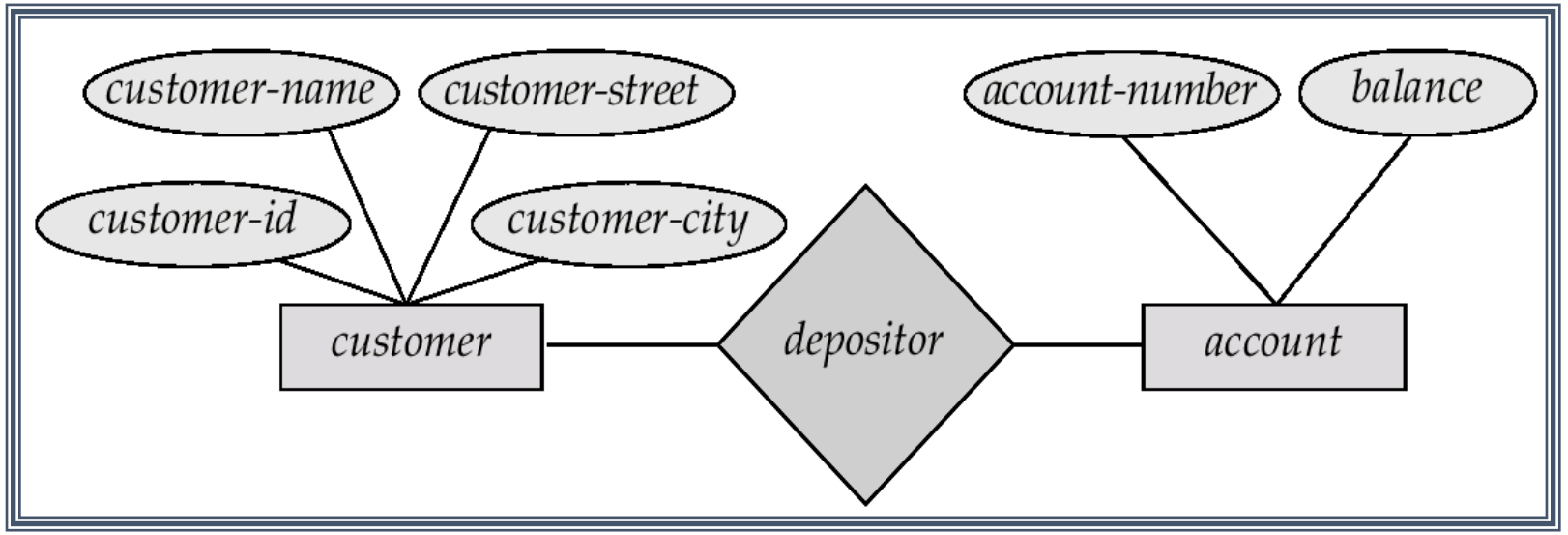
- The major activity of this model is identifying entities, attributes, and their relationships to construct model using the Entity Relationship Diagram.
  - Entity → table
  - Attribute → column
  - Relationship → line

Entity-Relationship Model

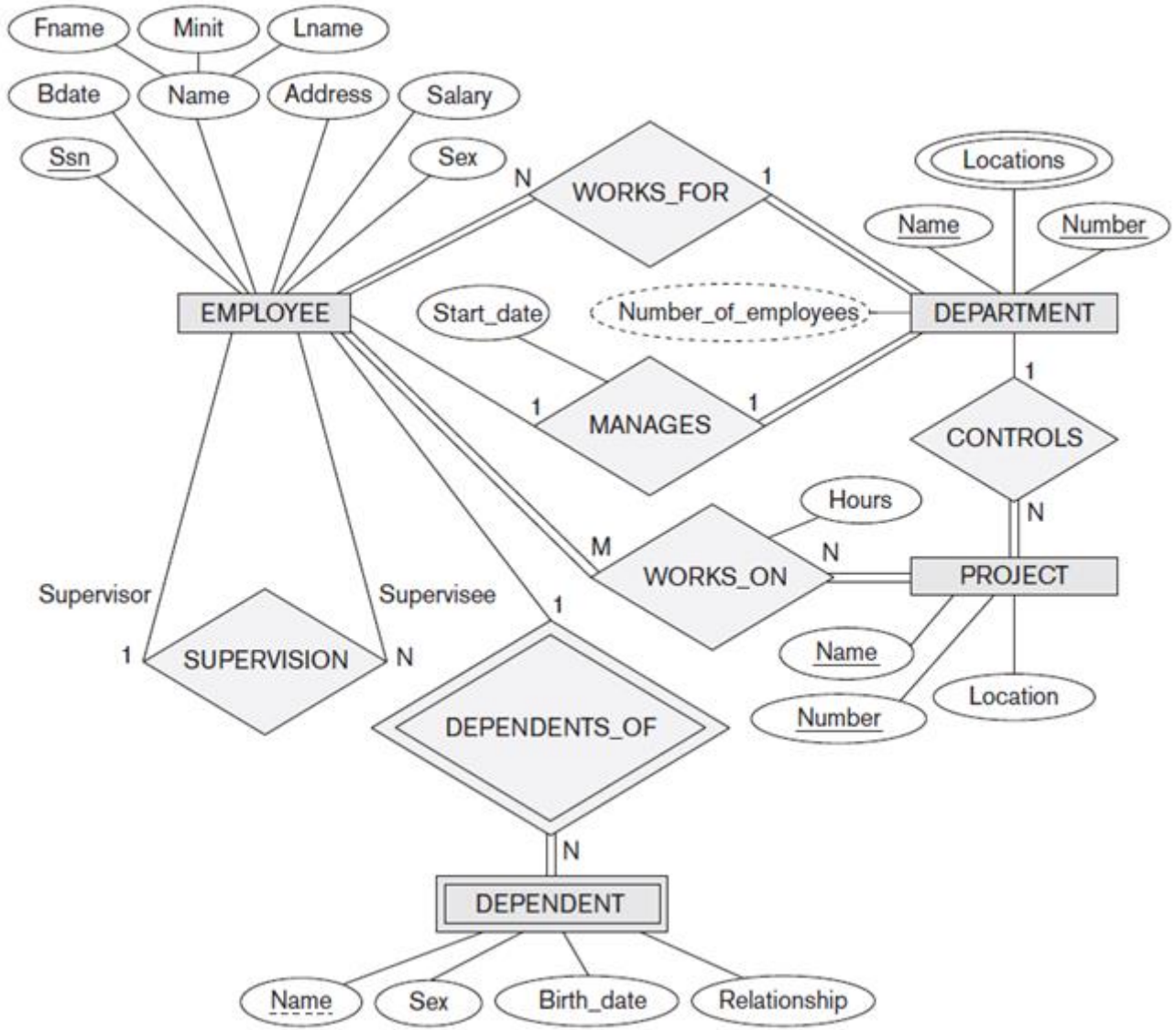
- ERD is a data modeling technique used in software engineering to produce a conceptual data model of an information system.
- So, ERDs illustrate the logical structure of databases.
- ERD development using a CASE tool
  - Powerdesigner by SAP
  - Data Modeler by Orcale

Entity-Relationship Model

- This is a way to model your tables.
- You can model the relationships between your tables.
- You can pick out your entities, attributes, and all sorts of things.
- Example of schema in the entity-relationship model



This is an example of what an entity relationship diagram looks like.



Entity Relationship Model

- Entities (objects)
  - It is used to describe a person, place, object, event or concept about which data are to be maintained
  - E.g. customers, accounts, bank branch

An *entity* is an abstract object of some sort, and a collection of similar entities forms an *entity set*. An entity in some ways resembles an "object" in the sense of object-oriented programming. Likewise, an entity set bears some resemblance to a class of objects.

How to find entities?

- Entity:
  - "...anything (people, places, objects, events, etc.) about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.)."
  - Tangible: customer, product
  - Intangible: order, accounting receivable
  - Look for singular nouns (beginner) BUT a proper noun is not a good candidate….

  - It's important to differentiate between an entity and an entity instance.
  - So, the entity itself might be called an employee but a particular entity instance would be a specific employee, like Ken.

  - For example: Movie database example
  - Each movie is an entity likewise the starts and studio are entities

Relationships

- Relationship:
  - Relationships are associations between entities.
  - Relationships are the glue that holds entities together.
  - Typically, a relationship is indicated by a verb connecting two or more entities.
  - Relationships should be classified in terms of cardinality.
    - One-to-one, one-to-many, etc.

Relationships

- **A relationship is an association between entities.**
- **Relationships are represented by diamond-shaped symbols.**

Figure :An Entity Relationship

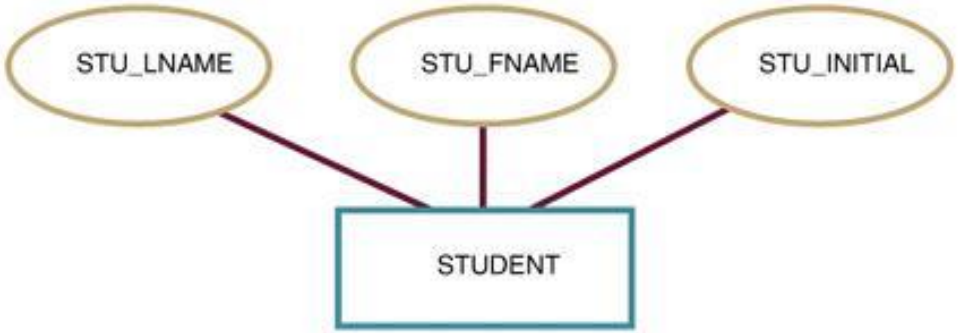
Relationships are connections among two or more entity sets. For instance, if *Movies* and *Stars* are two entity sets, we could have a relationship *Stars-in* that connects movies and stars.

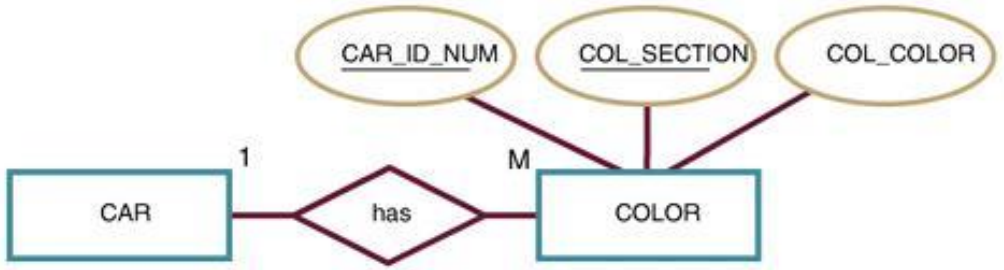
Attributes

- Attribute:
  - Attributes are data objects that either identify or describe entities (property of an entity).
- So, for example, students might have a student ID, a name, a home address, and so on. An automobile might have a vehicle ID, color, weight, and so on and so forth.

An attribute is a property or characteristic of an entity or relationship that is of interest to the organization.

Attributes examples





The entity Movies might be given attributes such as title and length

Example

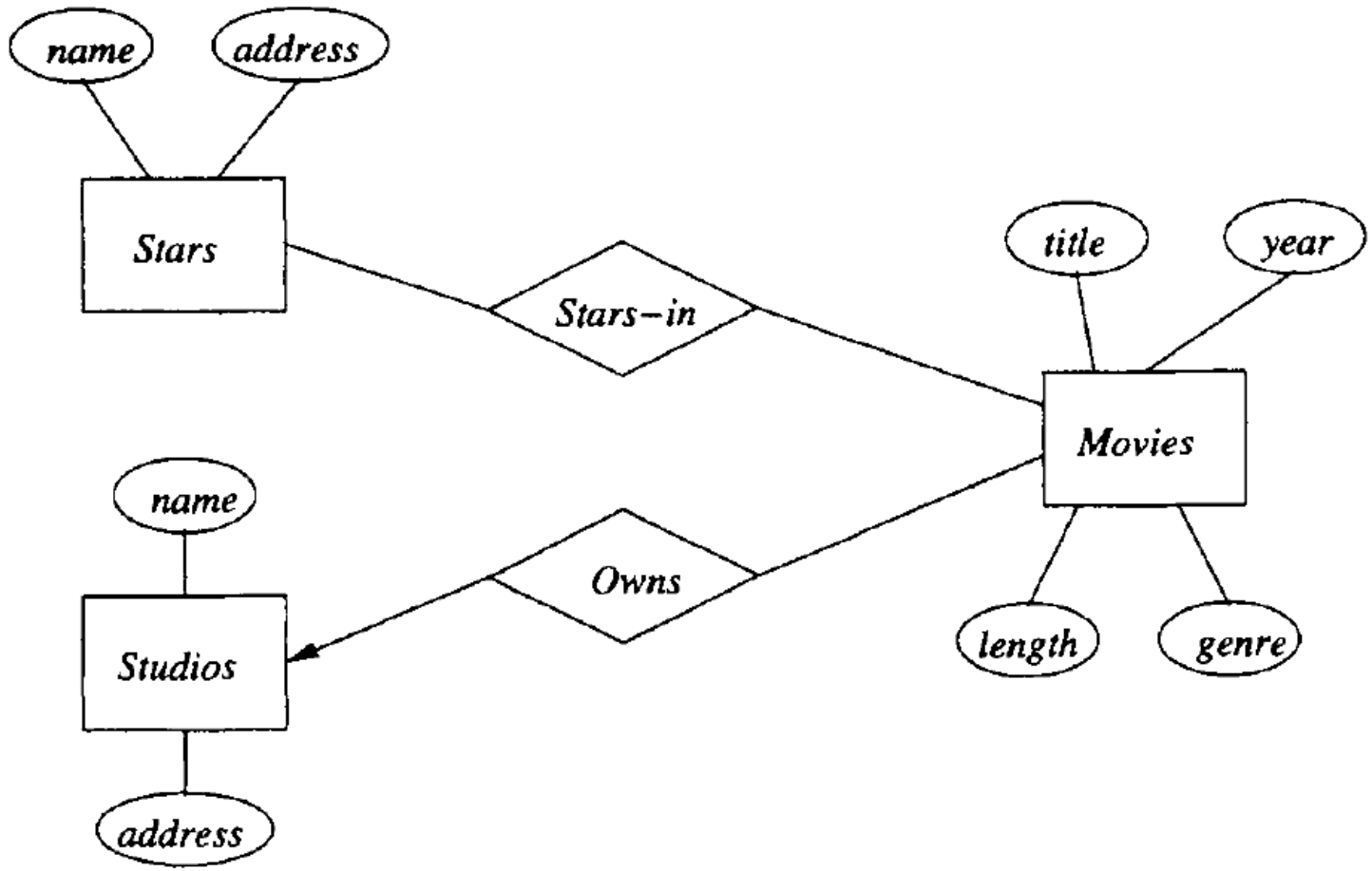


Figure 4.2: An entity-relationship diagram for the movie database

Classes of attributes

- Simple attribute
- Composite attribute
- Derived attributes
- Single-valued attribute
- Multi-valued attribute

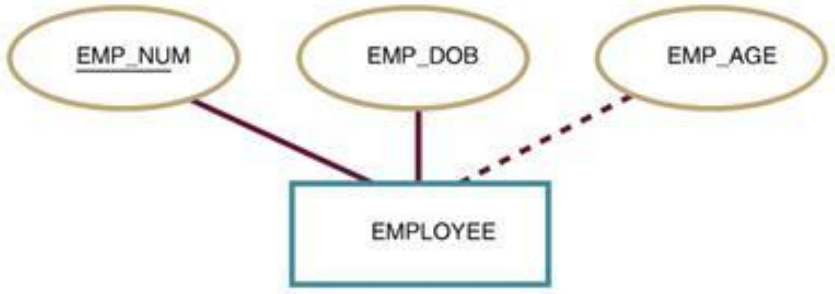
Simple/Composite attribute

- A **simple attribute** cannot be subdivided.
  - Examples: Age, Gender, and Marital status
- A **composite attribute** can be further subdivided to yield additional attributes.
  - Examples:
    - ADDRESS --→ Street, City, State, Zip

Derived attribute

A derived attribute is not physically stored within the database; instead, it is derived by using an algorithm.

- **Example: AGE can be derived from the data of birth and the current date.**



Single-valued attribute

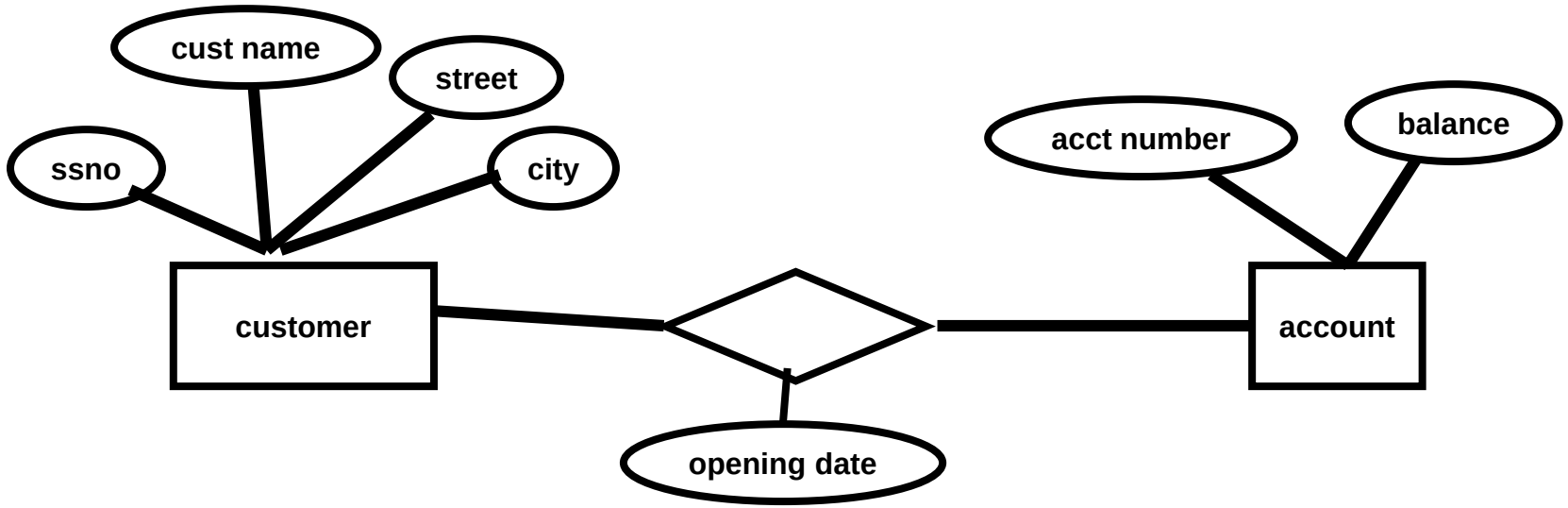
- can have only a single (atomic) value.
  - Examples:
    - A person can have only one social security number.
    - A manufactured part can have only one serial number.
  - **A single-valued attribute is not necessarily a simple attribute.**
    - Part No: CA-08-02-189935
    - Location: CA, Factory#:08, shift#: 02, part#: 189935

Multi-valued attributes

- can have many values.
  - Examples:
    - A person may have several college degrees.
    - A household may have several phones with different numbers
    - A car color

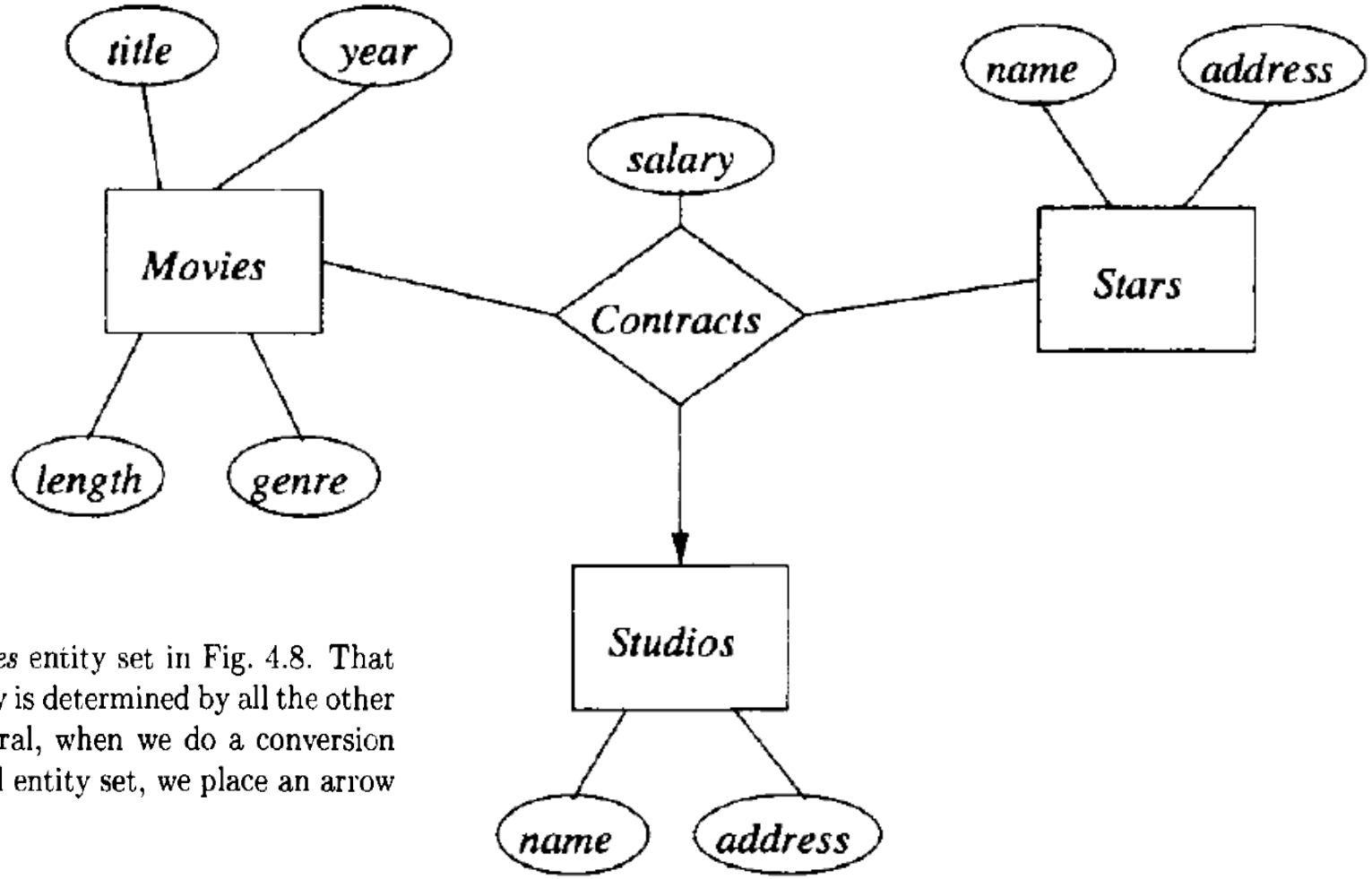
- Entity set -- rectangles; attributes -- ellipses; relationship set -- diamonds;
- dashed ellipse -- derived attribute;
- double ellipse -- multivalued attribute;

Relationship Attributes



Sometimes it is convenient, or even essential, to associate attributes with a relationship, rather than with any one of the entity sets that the relationship connects.

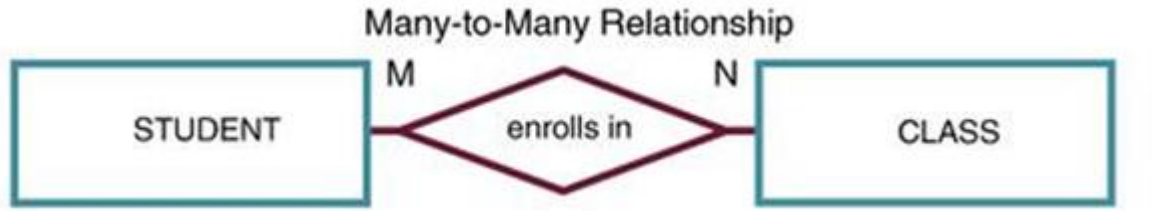
Relationship Attributes



Notice that there is an arrow into the *Salaries* entity set in Fig. 4.8. That arrow is appropriate, since we know that the salary is determined by all the other entity sets involved in the relationship. In general, when we do a conversion from attributes on a relationship to an additional entity set, we place an arrow into that entity set. □

Multiplicity on Relationship Sets

- The cardinality is the number of occurrences in one entity which are associated to the number of occurrences in another.
  - There are three basic cardinalities (degrees of relationship).
  - one-to-one (1:1), one-to-many (1:M), and many-to-many (M:N)



It refers to the maximum number of times an instance in one entity can be associated with instances in the related entity.

4.1.6 Multiplicity of Binary E/R Relationships

In general, a binary relationship can connect any member of one of its entity sets to any number of members of the other entity set. However, it is common for there to be a restriction on the "multiplicity" of a relationship. 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. Note that in a many-one relationship from E to F, each entity in F can be connected to many members of E. Similarly, if instead a member of F can be connected by R to at most one member of E, then we say R is many-one from F to E (or equivalently, one-many from E to F).

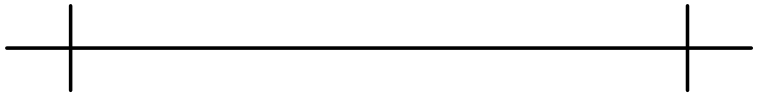
- If R is both many-one from E to F and many-one from F to E, then we say that R is *one-one.* In a one-one relationship an entity of either entity set can be connected to at most one entity of the other set.

- If R is neither many-one from E to F or from F to E, then we say R is *many-many.*

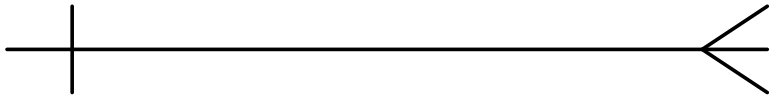
Example 4.4: A one-one relationship between entity sets E and F is represented by arrows pointing to both E and F. For instance, Fig. 4.3 shows two entity sets, *Studios* and *Presidents*, and the relationship *Runs* between them (attributes are omitted). We assume that a president can run only one studio and a studio has only one president, so this relationship is one-one, as indicated by the two arrows, one entering each entity set.

Basic Cardinality Type

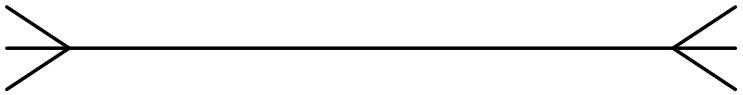
- 1-to-1 relationship



- 1-to-M relationship

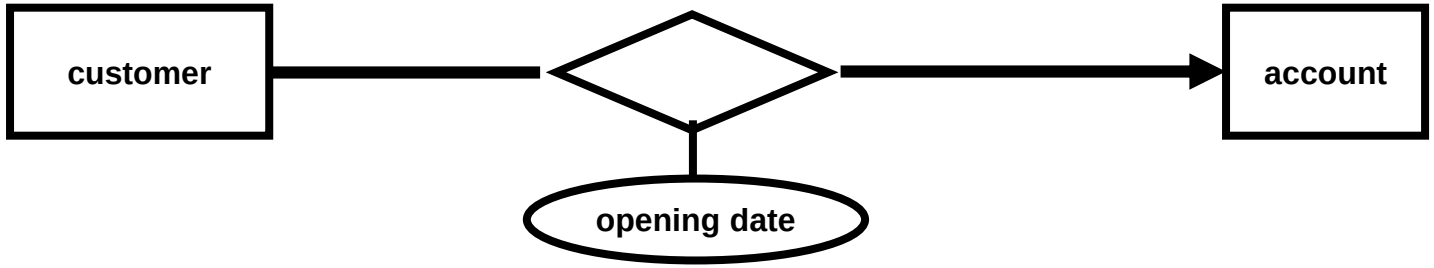


- M-to-N relationship



We typically use a **crow's foot**, as you see on the right-hand side of this relationship here, to represent a many relationship.

One to One Relationship



- 1 customer can have 1 account.
- One account can be owned by 1 customer
- relationship attributes can be shifted to either of the entity sets

Multiway relationship

The E/R model makes it convenient to define relationships involving more than two entity sets. In practice, ternary (three-way) or higher-degree relationships are rare, but they occasionally are necessary to reflect the true state of affairs. A multiway relationship in an E/R diagram is represented by lines from the relationship diamond to each of the involved entity sets.

Example 4.5: In Fig. 4.4 is a relationship *Contracts* that involves a studio, a star, and a movie. This relationship represents that a studio has contracted with a particular star to act in a particular movie. In general, the value of an E/R relationship can be thought of as a relationship set of tuples whose components are the entities participating in the relationship, as we discussed in Section 4.1.5. Thus, relationship *Contracts* can be described by triples of the form (studio, star, movie).

In Fig. 4.4 we have an arrow pointing to entity set *Studios*, indicating that for a particular star and movie, there is only one studio with which the star has contracted for that movie. However, there are no arrows pointing to entity sets *Stars* or *Movies*. A studio may contract with several stars for a movie, and a star may contract with one studio for more than one movie. □

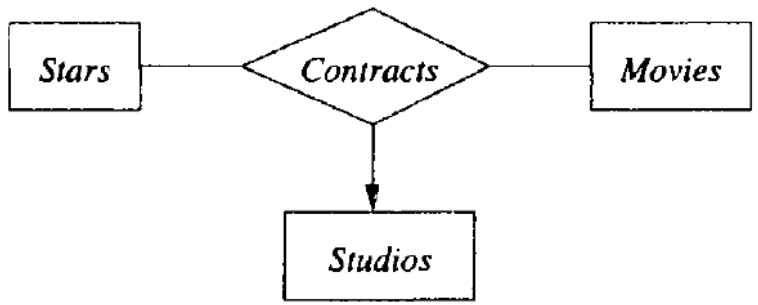


Figure 4.4: A three-way relationship

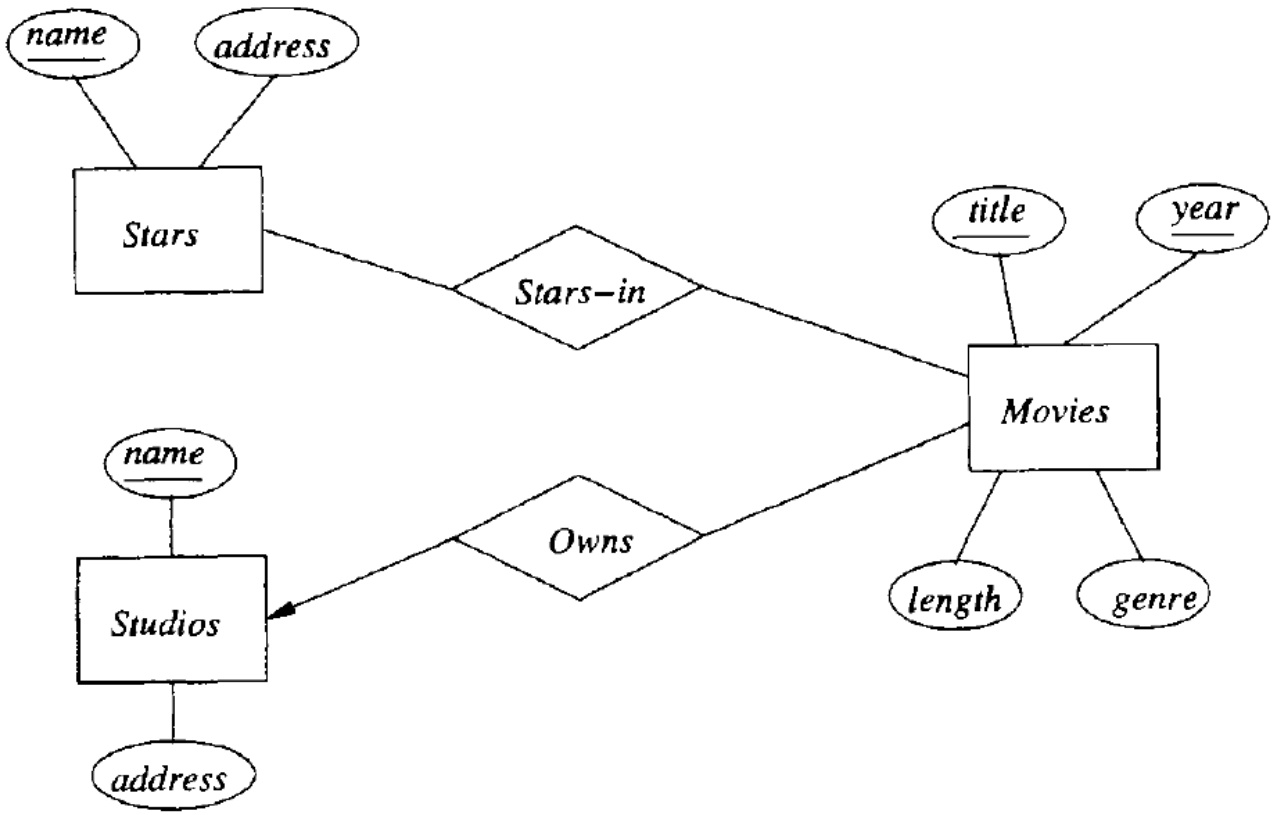
ERD vs Class diagram

- The more technical types of ERDs which you might see before is class diagram. That's when we add information like datatypes and other fields that we may not need at the analysis level.

- So, what you're going to see here is largely a simplified version of the ERD notation

- In our future lesson, we'll talk about how you transform the ERD on the left into the ERD on the right through a series of processes. But for now, it's just important to understand we'll be covering a simplified notation.

Representing Keys in the E/R Model



ER Diagrams, Naming Conventions, and Design Issues

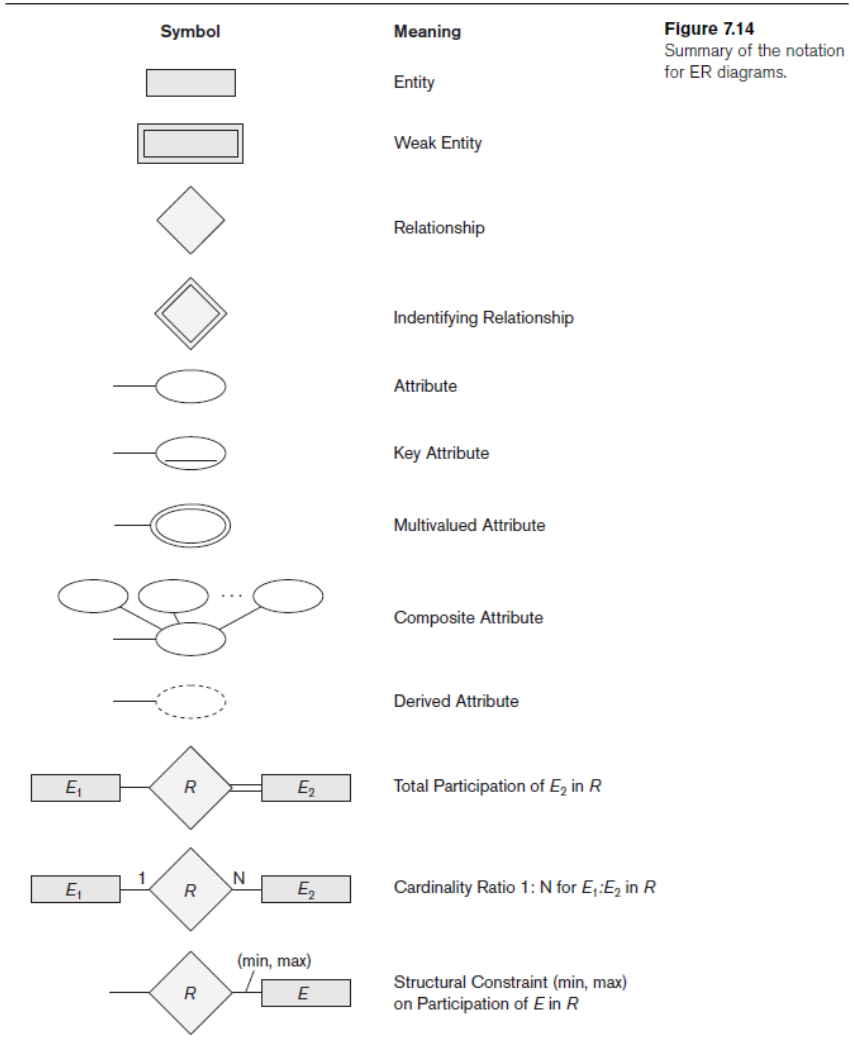


Figure 7.14
Summary of the notation for ER diagrams.

4.1.12 Exercises for Section 4.1

Exercise 4.1.1: Design a database for a bank, including information about customers and their accounts. Information about a customer includes their name, address, phone, and Social Security number. Accounts have numbers, types (e.g., savings, checking) and balances. Also record the customer(s) who own an account. Draw the E/R diagram for this database. Be sure to include arrows where appropriate, to indicate the multiplicity of a relationship.

Exercise 4.1.2: Modify your solution to Exercise 4.1.1 as follows:

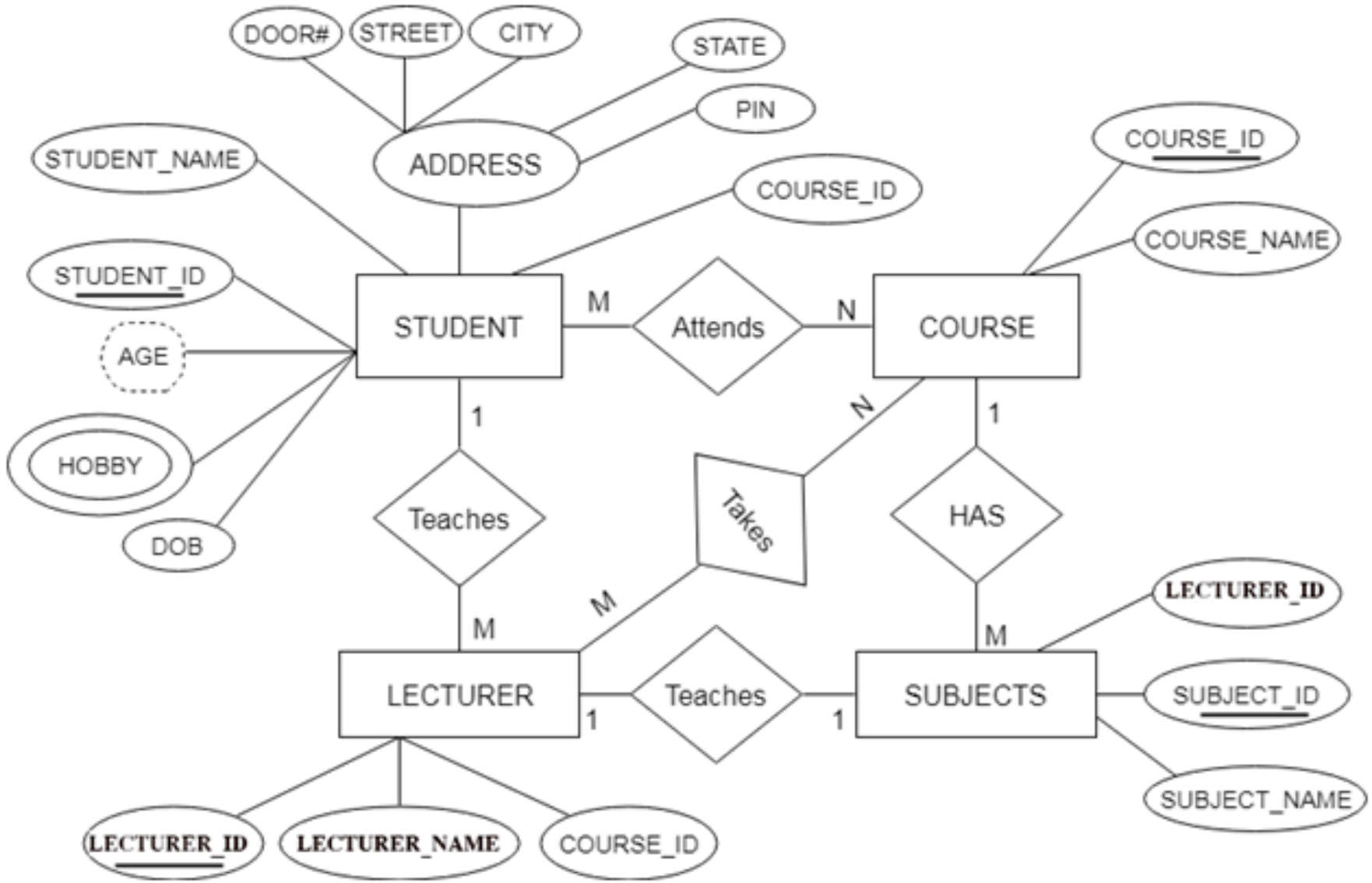
Exercise 4.1.3: Give an E/R diagram for a database recording information about teams, players, and their fans, including:

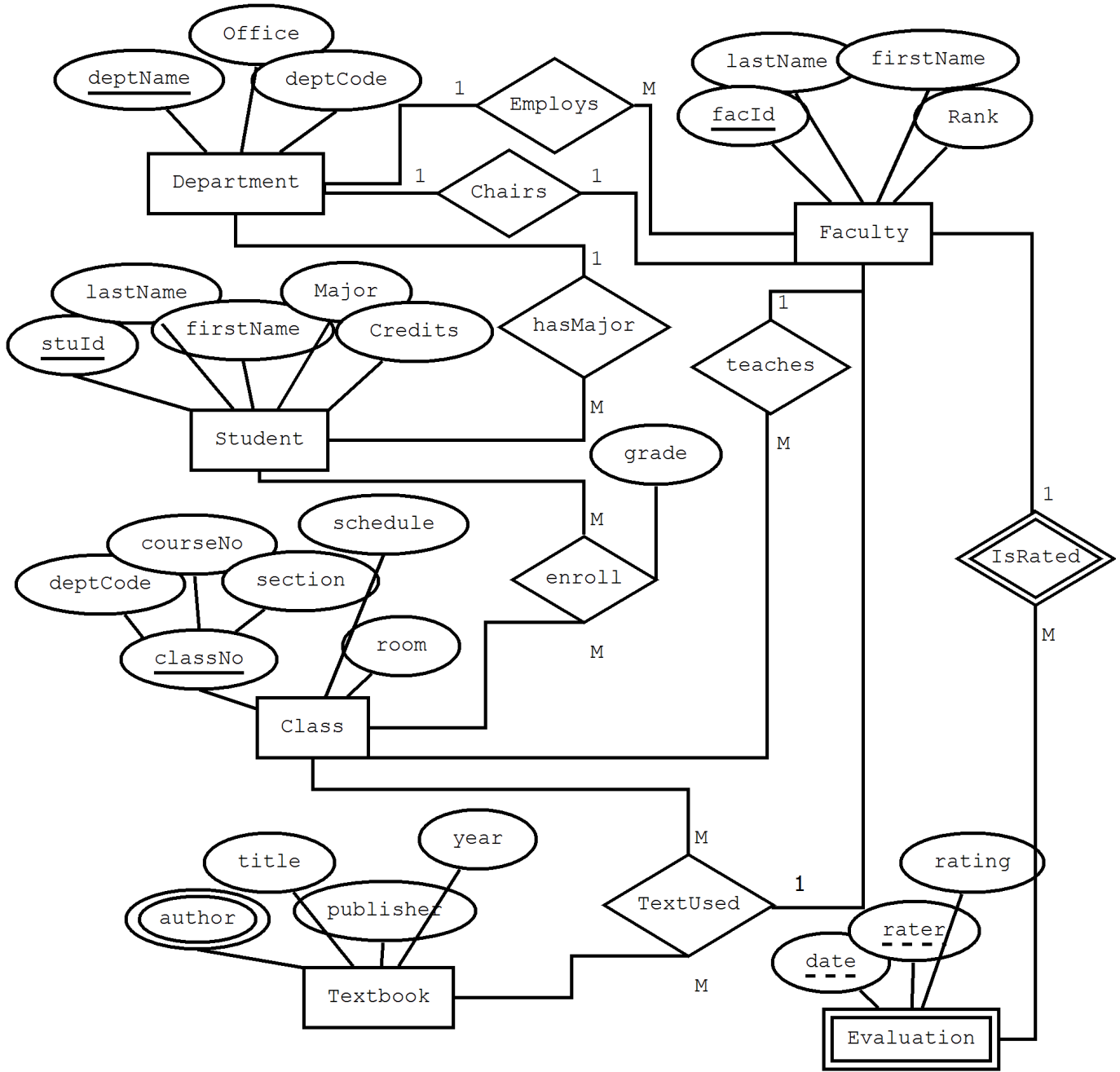
1. For each team, its name, its players, its team captain (one of its players), and the colors of its uniform.
2. For each player, his/her name.
3. For each fan, his/her name, favorite teams, favorite players, and favorite color.

Remember that a set of colors is not a suitable attribute type for teams. How can you get around this restriction?

Exercise 4.1.5: Modify Exercise 4.1.3 to record for each player the history of teams on which they have played, including the start date and ending date (if they were traded) for each such team.

Exercise 4.1.9: Design a database suitable for a university registrar. This database should include information about students, departments, professors, courses, which students are enrolled in which courses, which professors are teaching which courses, student grades, TA's for a course (TA's are students), which courses a department offers, and any other information you deem appropriate. Note that this question is more free-form than the questions above, and you need to make some decisions about multiplicities of relationships, appropriate types, and even what information needs to be represented.


DOOR#
STREET
CITY
STATE
PIN
ADDRESS
STUDENT_NAME
COURSE_ID
COURSE_ID
COURSE_NAME
STUDENT_ID
AGE
HOBBY
DOB
STUDENT
M
Attends
N
COURSE
1
N
1
Teaches
Takes
HAS
M
M
M
LECTURER
Teaches
SUBJECTS
1
1
LECTURER_ID
SUBJECT_ID
SUBJECT_NAME
LECTURER_ID
LECTURER_NAME
COURSE_ID


Office
deptName
deptCode
Employs
1
M
lastName
firstName
facId
Rank
Department
1
Chairs
1
Faculty
1
lastName
Major
stuId
firstName
Credits
hasMajor
1
teaches
M
Student
M
grade
M
schedule
courseNo
deptCode
section
enroll
M
1
IsRated
M
classNo
room
Class
M
year
title
author
publisher
TextUsed
1
rating
rater
date
M
Textbook
Evaluation

Design Principles

- **Faithfulness**

➢ **Entity sets and their attributes should reflect reality.**

Example 4.12: If we define a relationship *Stars-in* between *Stars* and *Movies*, it should be a many-many relationship. The reason is that an observation of the real world tells us that stars can appear in more than one movie, and movies can have more than one star. It is incorrect to declare the relationship *Stars-in* to be many-one in either direction or to be one-one. □

Example 4.13: On the other hand, sometimes it is less obvious what the real world requires us to do in our E/R design. Consider, for instance, entity sets *Courses* and *Instructors*, with a relationship *Teaches* between them. Is *Teaches* many-one from *Courses* to *Instructors*? The answer lies in the policy and intentions of the organization creating the database. It is possible that the school has a policy that there can be only one instructor for any course. Even if several instructors may "team-teach" a course, the school may require that exactly one of them be listed in the database as the instructor responsible for the course. In either of these cases, we would make *Teaches* a many-one relationship from *Courses* to *Instructors*.

- **Avoiding Redundancy**

➢We should be careful to say everything once only.

  - Need extra space
  - Update-anomaly may happen(ex: change relationship but not attribute)

- **Choosing the Right Relationships**

➢Adding to our design every possible relationship is not often a good idea.

➢It can lead to redundancy, update anomalies, and deletion anomalies, where the connected pairs or sets of entities for one relationship can be deduced from one or more other relationships.

In summary, we cannot tell you whether a given relationship will be redundant. You must find out from those who wish the database implemented what to expect.

- **Simplicity counts**

➢Avoid introducing more elements into your design than is absolutely necessary.

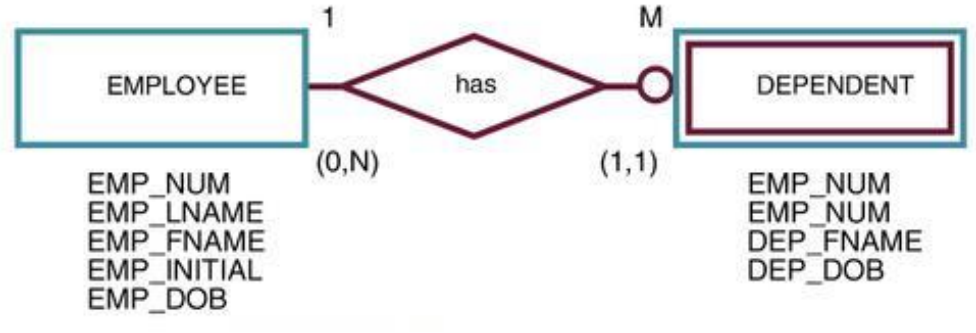
- **Picking the Right kind of element**
  - ➢In general, an attribute is simpler to implement than either an entity set or a relationship.
  - ➢However, making everything an attribute will usually get us into trouble.

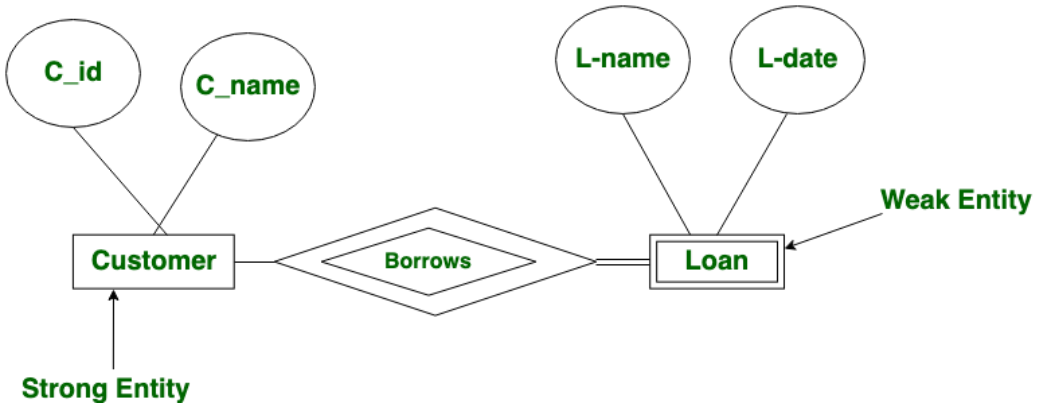
Example 4.17: Let us consider a specific problem. In Fig. 4.2, were we wise to make studios an entity set? Should we instead have made the name and address of the studio be attributes of movies and eliminated the *Studio* entity set? One problem with doing so is that we repeat the address of the studio for each movie. We can also have an update anomaly if we change the address for one movie but not another with the same studio, and we can have a deletion anomaly if we delete the last movie owned by a given studio.

On the other hand, if we did not record addresses of studios, then there is no harm in making the studio name an attribute of movies. We have no anomalies in this case. Saying the name of a studio for each movie is not true redundancy, since we must represent the owner of each movie somehow, and saying the name of the studio is a reasonable way to do so. □

Weak entity

- Weak Entities
  - **A weak entity is an entity that**
    - **Is existence-dependent and**
    - **Has a primary key that is partially or totally derived from the parent entity in the relationship.**
  - **The existence of a weak entity is indicated by a double rectangle.**
  - **The weak entity inherits all or part of its primary key from its strong counterpart.**





Example 4.20: A movie studio might have several film crews. The crews might be designated by a given studio as crew 1, crew 2, and so on. However, other studios might use the same designations for crews, so the attribute *number* is not a key for crews. Rather, to name a crew uniquely, we need to give both the name of the studio to which it belongs and the number of the crew. The situation is suggested by Fig. 4.20. The double-rectangle indicates a weak entity set, and the double-diamond indicates a many-one relationship that helps provide the key for the weak entity set. The notation will be explained further in Section 4.4.3. The key for weak entity set *Crews* is its own *number* attribute and the *name* attribute of the unique studio to which the crew is related by the many-one *Unit-of* relationship. □

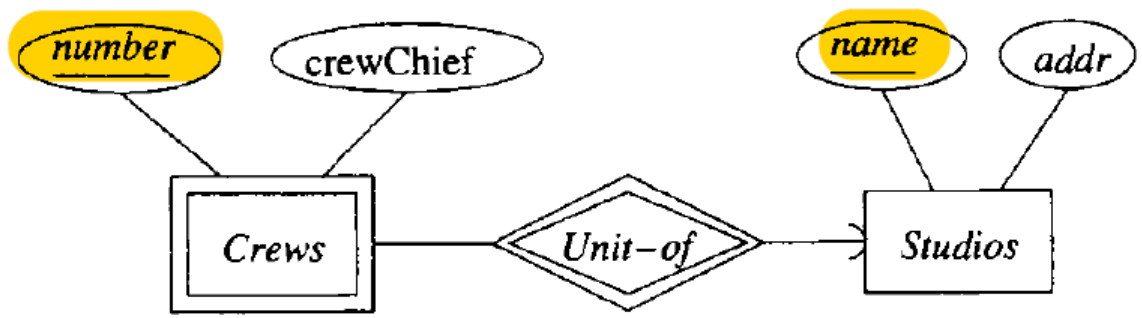


Figure 4.20: A weak entity set for crews, and its connections

Exercise 4.4.4: Draw E/R diagrams for the following situations involving weak entity sets. In each case indicate keys for entity sets.

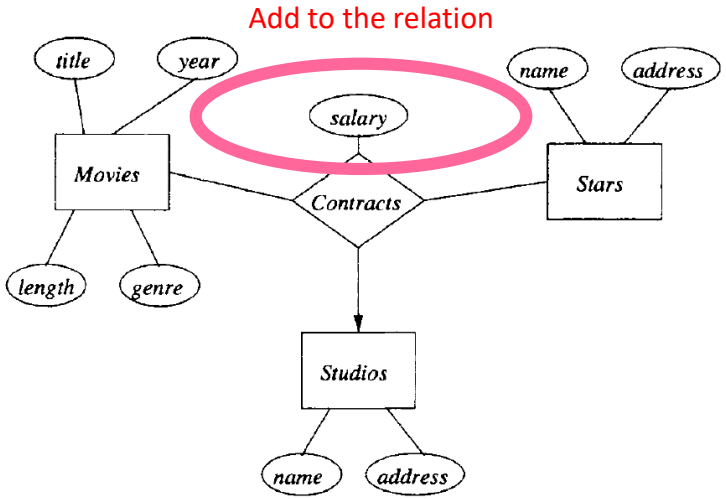
a) Entity sets *Courses* and *Departments*. A course is given by a unique department, but its only attribute is its number. Different departments can offer courses with the same number. Each department has a unique name.

Combining relations

Because R is many-one, all these attributes are functionally determined by the key for E, and we can combine them into one relation with a schema consisting of:

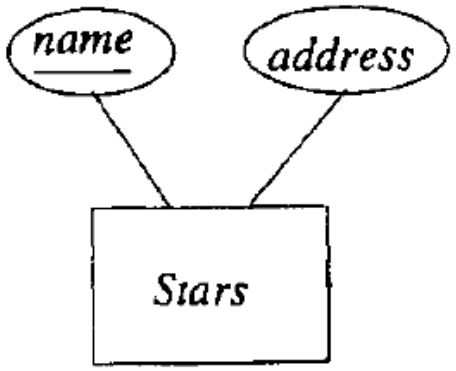
1. All attributes of E.
2. The key attributes of F.
3. Any attributes belonging to relationship R.

title	*year*	*length*	*genre*	*studioName*
Star Wars	1977	124	sciFi	Fox
Gone With the Wind	1939	239	drama	MGM
Wayne's World	1992	95	comedy	Paramount



From E/R diagrams to Relational designs

1. **Turn each entity set into a relation with the same set of attributes**



name	*address*
Carrie Fisher	123 Maple St., Hollywood
Mark Hamill	456 Oak Rd., Brentwood
Harrison Ford	789 Palm Dr., Beverly Hills

2. **Replace a relationship by a relation whose attributes are the keys for the connected entity sets**

   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.