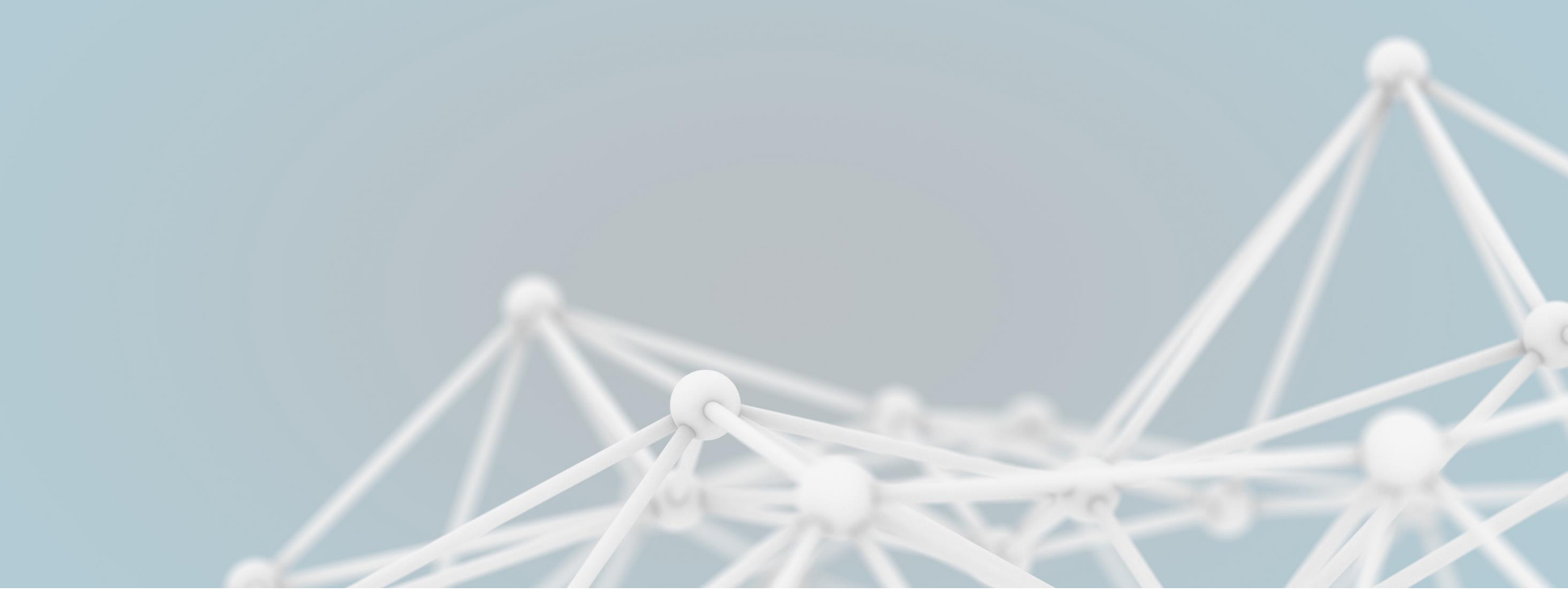


02 CONCEPTUAL DESIGN (ENTITY RELATIONSHIP MODEL)

CS 340: INTRODUCTION TO DATABASE SYSTEMS

Adapted from: Dr. Omar Alomeir 2023

Course instructor:
Dr. Maram Alajlan.

DATABASES SO FAR

We learned that databases are useful because:

- They store a lot of data for applications.
- Give desirable guarantees for transactions.
- Are efficient and reliable.
- *Many more reasons..*

Before any of this is possible, we need to carefully design a database.

LEARNING GOALS

CLO2: Develop conceptual modeling concepts and notations using Entity-Relationship diagrams to analyze and design complex database applications. (Cognitive Skill)

Chapter objectives:

- Define some basic concepts related to databases (abstraction, schemas, and instances).
- Explain the main components of ER diagrams (entities, attributes, and relationships).
- Given a description of a problem, create an ER diagram. Justify your choice of entities, relationships, keys, cardinality constraints, participation constraints, and weak entities.

PRELIMINARIES

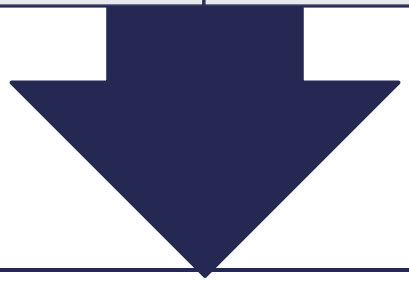
Database systems provide an abstract view of the data.

The 3 levels of abstraction are as follows:

- Physical level: how data are stored
- Logical level: what data are stored
- External level (views): Shows different parts of the db to different users
  - Example: Student view is different from registrar or database admin.
- We refer to the suppression of data storage details as data abstraction.
- Data abstraction is one of the key advantages of using relational databases.

Data abstraction

Suppression of details of data organization and storage	Highlighting of the essential features for an improved understanding of data.



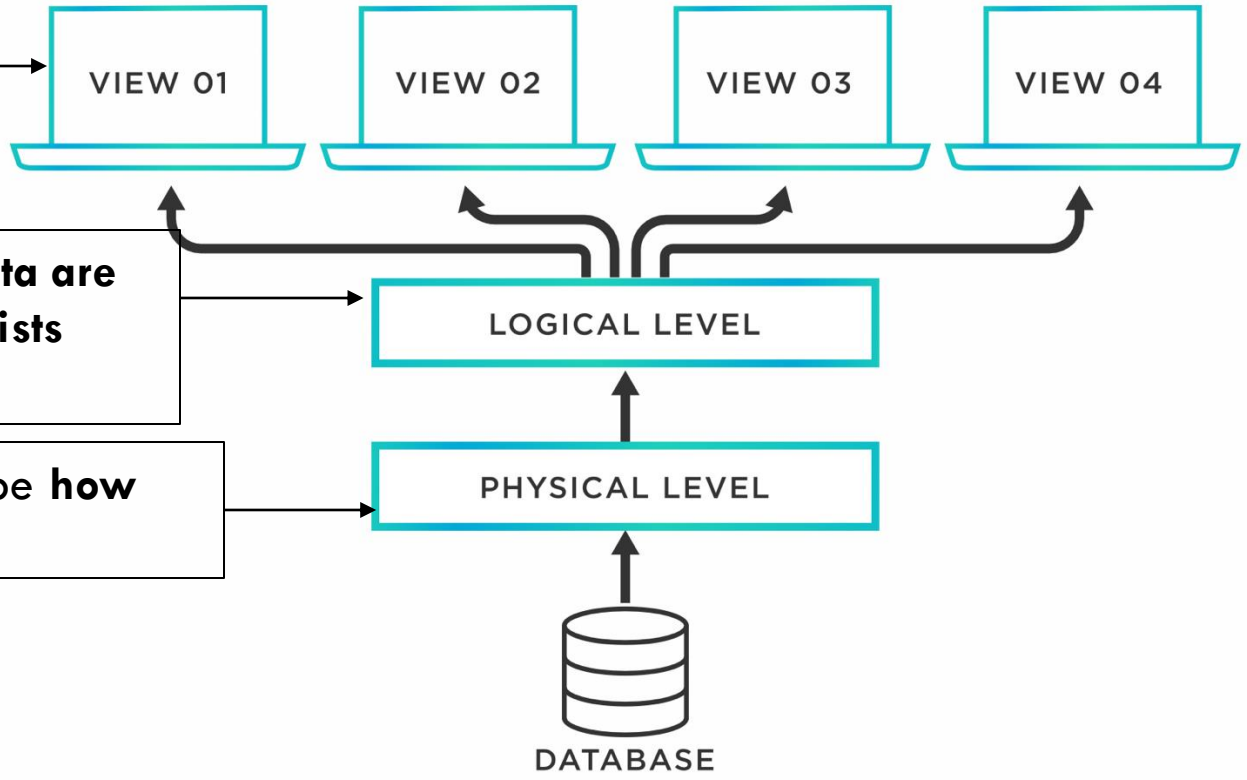
One of the main characteristics of the database approach is to support data abstraction so that different users can perceive data at their preferred level of detail.

LEVELS OF ABSTRACTION

Next highest level of abstraction describe **only part of entire db.** Many users of the db system do not need all the information.

Next higher level of abstraction describe **what data are stored** in the database, and **what relationship exists among those data.**

The lowest level of abstraction describe **how the data are actually stored.**



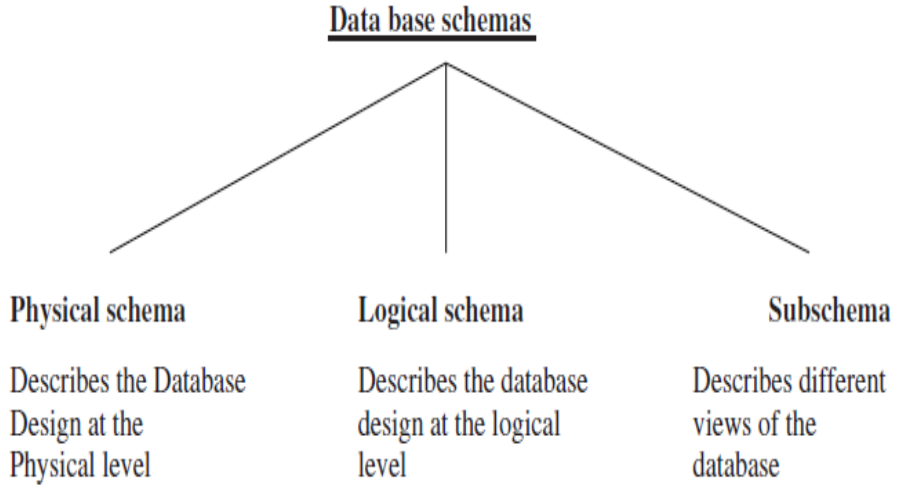
SCHEMAS AND INSTANCES

We create the schema, which is the logical structure of the database (e.g. Student takes course).

- **Logical schema:** describes the database design at the logical level.
- **Physical schema:** describes the database design at the physical (storage) level.
- **External Schema(Subschema): describes d**ifferent views of the database for various external users

After we create the schema, we fill the database with data. This is called creating instances of the database.

Instances: The data in the database at a particular moment in time



The database schema changes very infrequently.
The database state changes every time the database is updated.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

A database that stores student and course information.

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

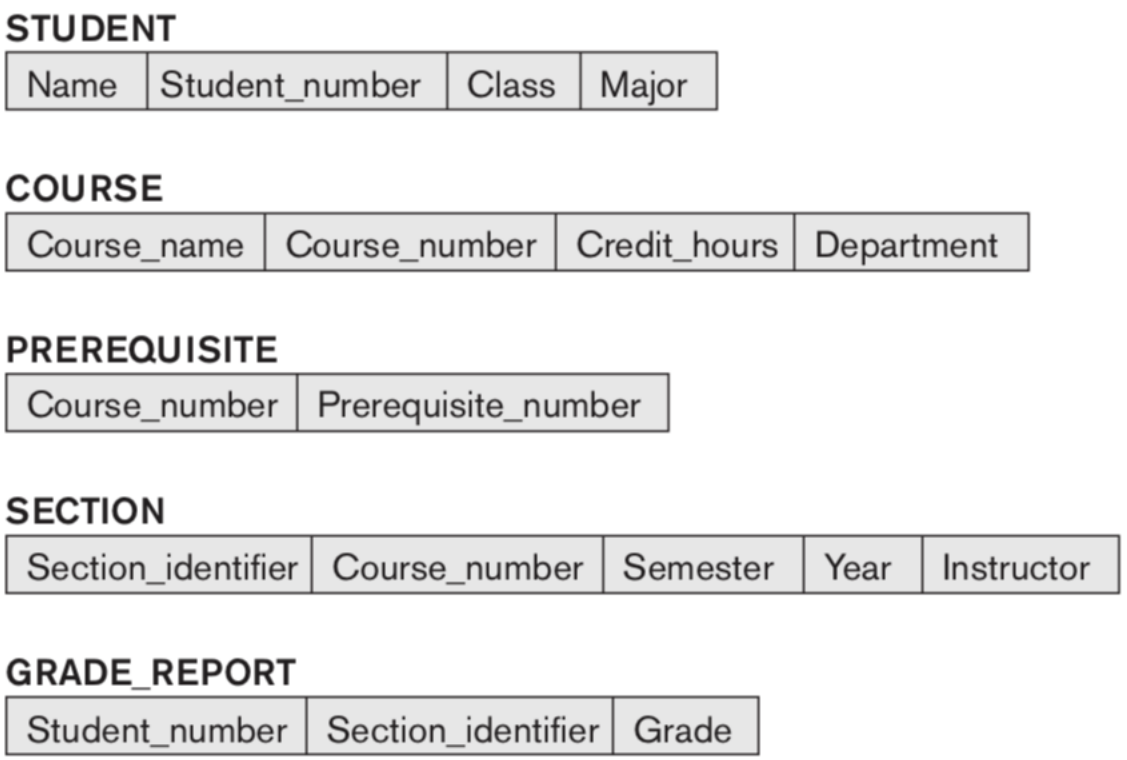
Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

SCHEMAS

Schema diagram for the previous database



DATA INDEPENDENCE

Data Independence refers to the protection of user applications to changes made in the definition and organization of data.

Application programs should not, ideally, be exposed to details of data representation and storage. The DBMS provides an abstract view of the data that hides such details.

There are two types of data independence: physical and logical data independence.

DATA INDEPENDENCE

Logical Data Independence:

The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not change.

- For example: consider two users A & B. Both are selecting the fields "EmployeeNumber" and "EmployeeName". If user B adds a new column (e.g. salary) to his table, it will not affect the external view for user A, though the internal schema of the database has been changed for both users A & B.

DATA INDEPENDENCE

Physical Data Independence:

The capacity to change the internal schema without having to change the conceptual schema.

- Modifications at the physical level are occasionally necessary to improve performance. For example, a change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

Logical data independence is more difficult to achieve than physical data independence, since application programs are heavily dependent on the logical structure of the data that they access.

CONCEPTUAL DESIGN

We have our enterprise we intend to model. What are the entities and relationships in this enterprise?

Entities: (usually) nouns that represent things in the enterprise (e.g. Student, course, etc)

Relationships: (usually) verbs that represent statements about 2 or more objects (e.g. Student takes course).

What information should we store about those entities and relationships?

What rules do we need to make sure our data is sound? **Integrity constraints.**

The encoding in the relational database realm is referred to as an Entity-Relationship (ER) diagram.

ER COMPONENTS: ENTITIES AND ATTRIBUTES

Entity: Real world object distinguishable from other objects.

A set of *attributes* describe an entity.

Entity set: A collection of similar entities. E.g. Employee is an entity set for all employees.

- All entities in an entity set have the same set of *attributes*.
- Each attribute has a *domain* (e.g. float, int, date).
- Each entity set has a *key* attribute.

ER COMPONENTS: ATTRIBUTE TYPES

Simple

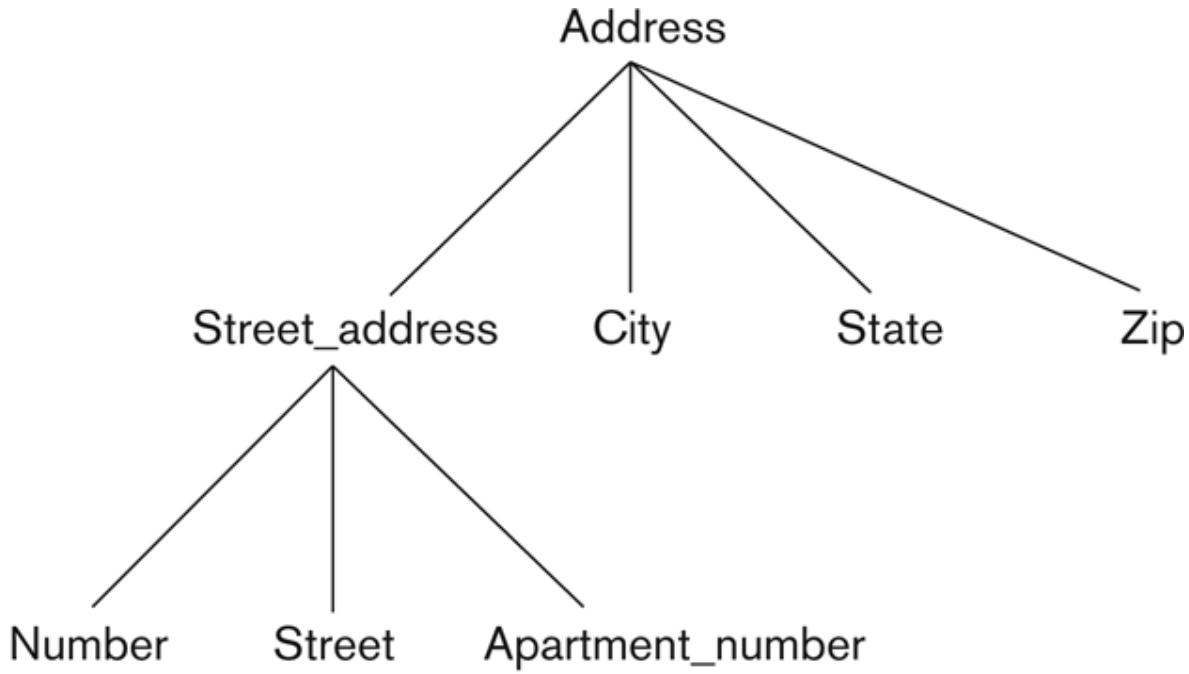
- Each entity has a single atomic value for the attribute. For example, SSN or Sex.

Composite

- The attribute may be divided into smaller parts. For example:
  - Address(Number, Street, City, Country), or
  - Name(FirstName, MiddleName, LastName).
  - Composition may form a hierarchy where some components are themselves composite.

EXAMPLE OF A COMPOSITE ATTRIBUTE

Address(Street_address
(number, street ,
apartment_number),
city, state, zip)



TYPES OF ATTRIBUTES: MULTI-VALUED

Multi-valued attributes:

- A set of values for the same entity.
  - E.g. colors attribute for a car - {Color} 1…3
  - College degrees attribute for a person - {college degree}.1…3
  - A multi-valued attribute may have lower and upper bounds to constrain the number of values allowed e.g. the colors attribute of a car may have between 1 and 3 values.

TYPES OF ATTRIBUTES: STORED AND DERIVED

- When two or more attributes are related, the value of one attribute can be obtained from the other.
- E.g. The age can be determined from the current date and the value of the birth date of a person, so age is a *derived attribute* and birthdate is a *stored attribute*.
- Some attribute values can be derived from related entities; e.g. an attribute Number_of_employees of a DEPARTMENT entity can be derived by counting the number of employees related to that department.

TYPES OF ATTRIBUTES: COMPLEX ATTRIBUTES

In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.

- For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
- Multiple PreviousDegrees values can exist
- Each has four subcomponent attributes:
  - College, Year, Degree, Field

This one is very rare and unlikely to occur. It is also not a good design technique as we will see later.

ER COMPONENTS: KEYS

Entities with the same basic attributes are grouped or typed into an entity type.

- For example, the entity type EMPLOYEE and PROJECT.

An attribute of an entity type for which each entity must have a unique value is called a *key attribute* of the entity type.

- For example, SSN of EMPLOYEE or student ID for a student.

ER COMPONENTS: KEYS

A key attribute may be composite.

- VehicleTagNumber is a key of the CAR entity type with components (Number, State).

An entity type may have more than one key.

- The CAR entity type may have two keys:
  - VehicleIdentificationNumber (popularly called VIN)
  - VehicleTagNumber (Number, State), aka license plate number.

Each key is underlined (Note: this is different from the relational schema where only one "primary key" is underlined).

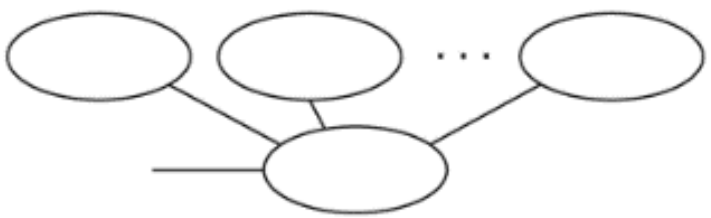
MODELING ENTITIES IN ER DIAGRAMS

In ER diagrams, an entity type is displayed in a **rectangular box**

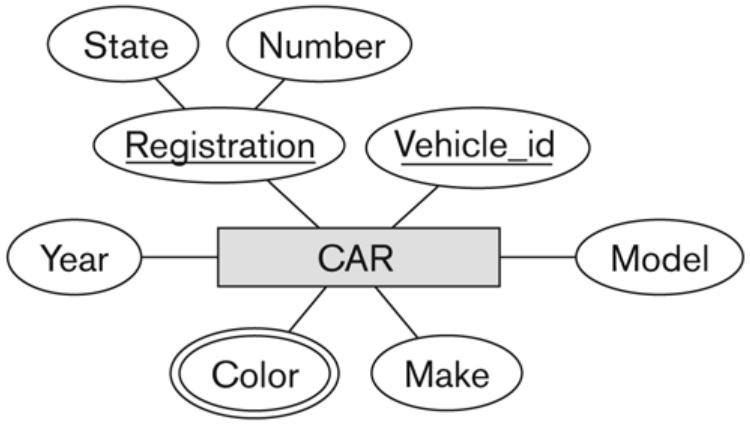
Attributes are displayed in **ovals**

- Each attribute is connected to its entity type
- Components of a composite attribute are connected to the oval representing the composite attribute
- Each key attribute is underlined
- Multivalued attributes displayed in double ovals

See the full ER notation in advance on the next slide

Symbol	Meaning
	Entity
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

EXAMPLE: CAR ENTITY AND CAR ENTITY SET



CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Car entity with two key attributes (left), Car entity set with 3 car entities (right).

ACTIVITY

Identify the entities and their corresponding attributes from the next 2 slides. Draw them and use proper notation.

I will give you 2 minutes per slide.

EXAMPLE: COMPANY DATABASE

We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

EXAMPLE COMPANY DATABASE (CONTINUED)

- The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - The DB will keep track of the number of hours per week that an employee currently works on each project.
  - It is required to keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

INITIAL CONCEPTUAL DESIGN OF ENTITY TYPES FOR THE COMPANY DATABASE SCHEMA

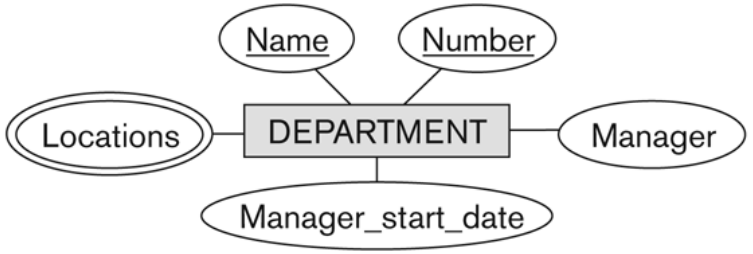
Based on the requirements, we can identify four initial entity types in the COMPANY database:

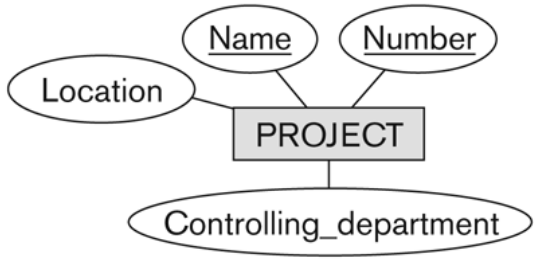
- DEPARTMENT
- PROJECT
- EMPLOYEE
- DEPENDENT

Their initial conceptual design is shown on the following slide

The initial attributes shown are derived from the requirements description

INITIAL DESIGN OF ENTITY TYPES

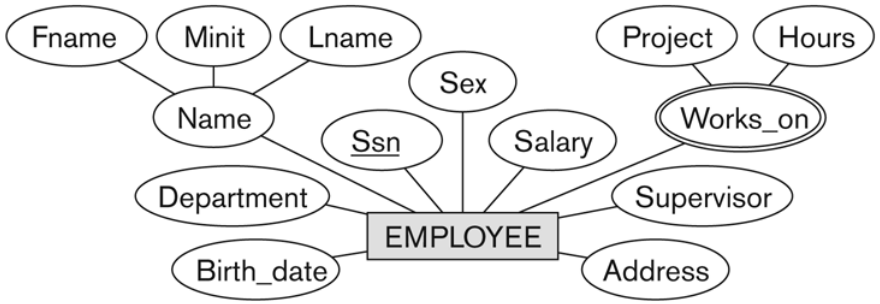




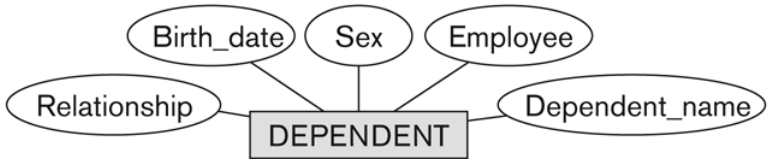
Each department has a name, number and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.

Each department controls a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

INITIAL DESIGN OF ENTITY TYPES



Each EMPLOYEE has a social security number, address, salary, sex, and birthdate.



Each employee may have a number of DEPENDENTs. For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee

ER COMPONENTS: RELATIONSHIPS

A *relationship* relates two or more distinct entities with a specific meaning.

- For example, EMPLOYEE John Smith *works on* the ProductX PROJECT, or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.

Relationships of the same type are grouped or typed into a *relationship type*.

- For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

RELATIONSHIP TYPE AND RELATIONSHIP SET

Relationship Type:

- Is the schema description of a relationship
- Identifies the relationship name and the participating entity types
- Also identifies certain relationship constraints

Relationship Set:

- The current set of relationship instances represented in the database
- The current *state* of a relationship type

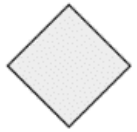
RELATIONSHIP DEGREES OR ARITY

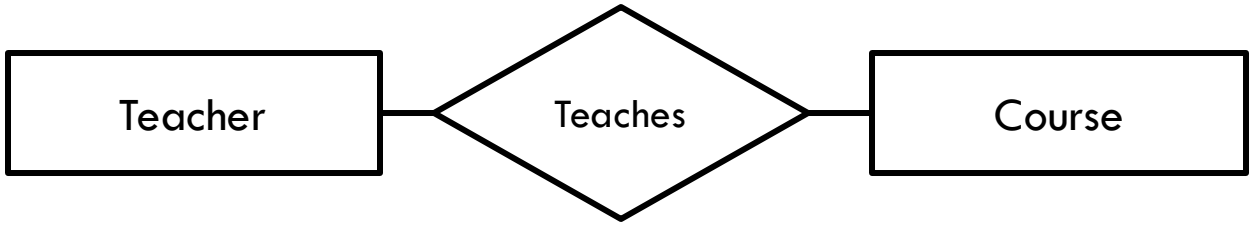
Degree or arity of a relationship is the number of entity types that participate in it:

- Unary (or recursive) relationship.
- Binary Relationship.
- Ternary relationship.

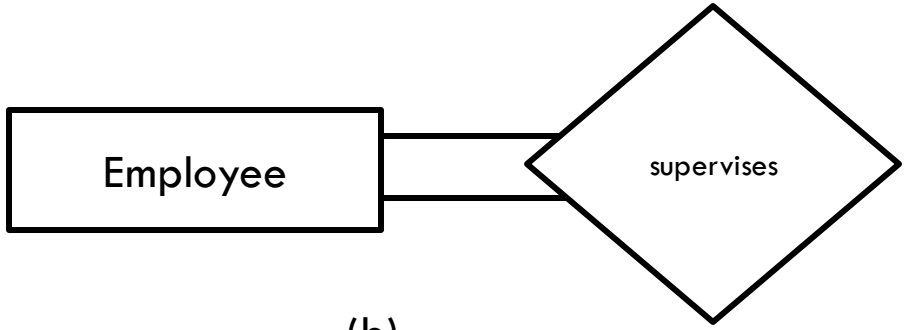
E.g. Both MANAGES and WORKS_ON are *binary* relationships.

RELATIONSHIP EXAMPLES

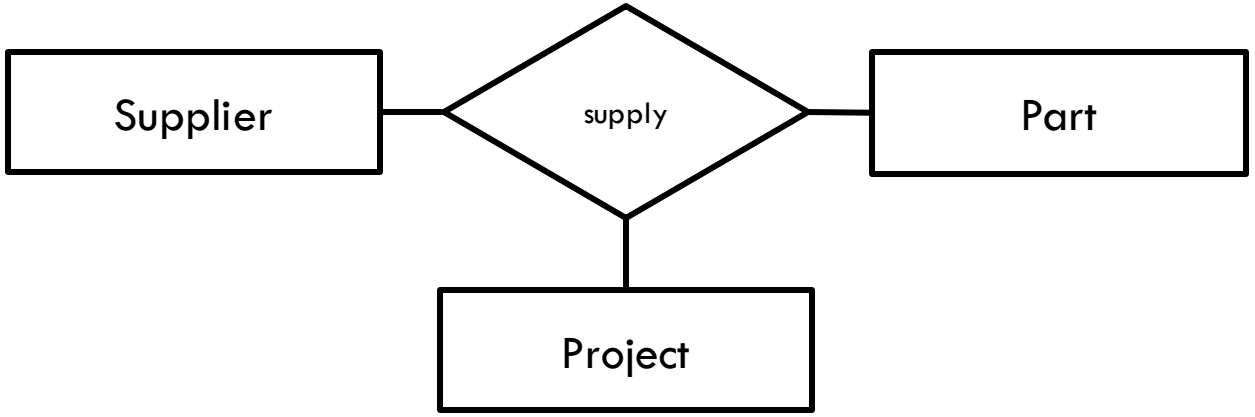
Symbol	Meaning
	Relationship



(a)



(b)



(c)

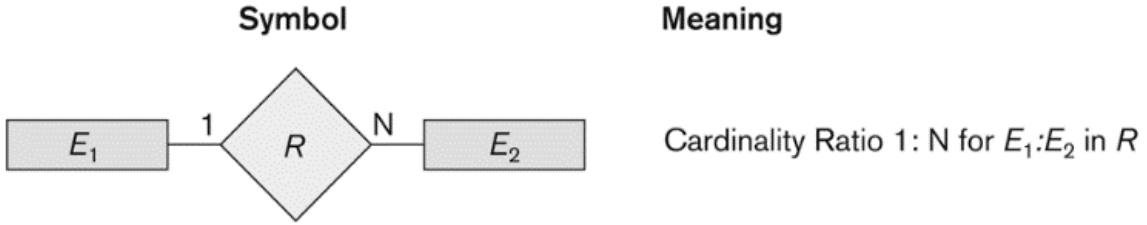
(a) Binary relationship
(b) Unary relationship
(c) Ternary relationship

RELATIONSHIP CARDINALITY CONSTRAINTS

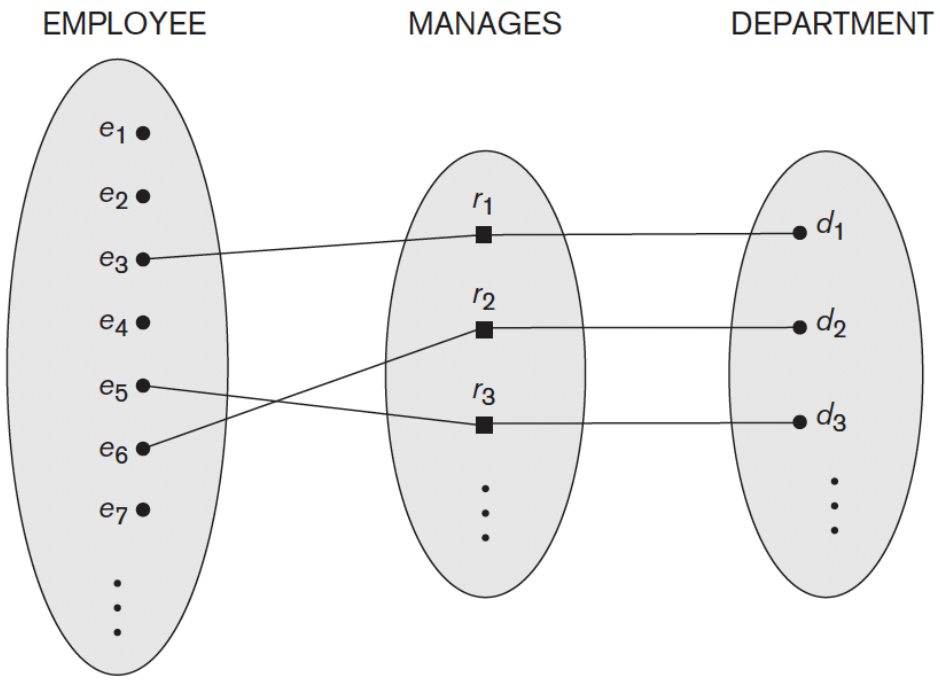
Cardinality ratio for a relationship set specifies the number of relationships in the set in which an entity can participate.

We will look at 4 types of relationships based on cardinality:

1. One-to-one relationships
2. Many-to-one relationships
3. one-to-many relationships
4. Many-to-many relationships



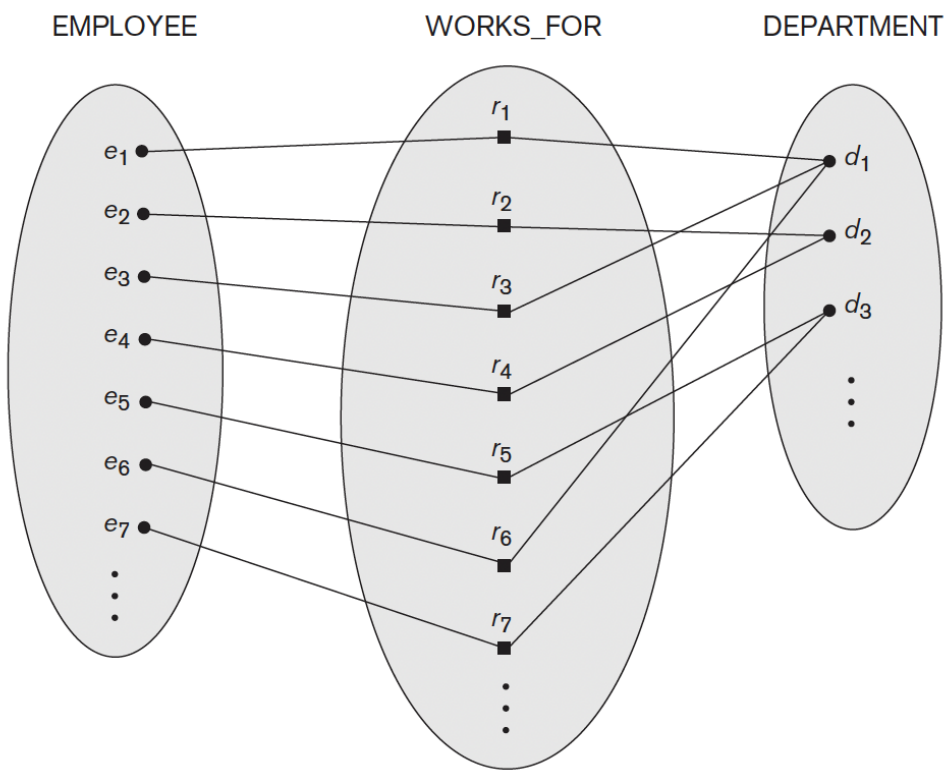
ONE TO ONE (1:1) RELATIONSHIPS



An entity in A is associated with at most one entity in B. Likewise, an entity in B is associated with at most one entity in A.

E.g., an employee manages at most one department. A department is managed at most by one employee.

MANY TO ONE (N:1) RELATIONSHIPS

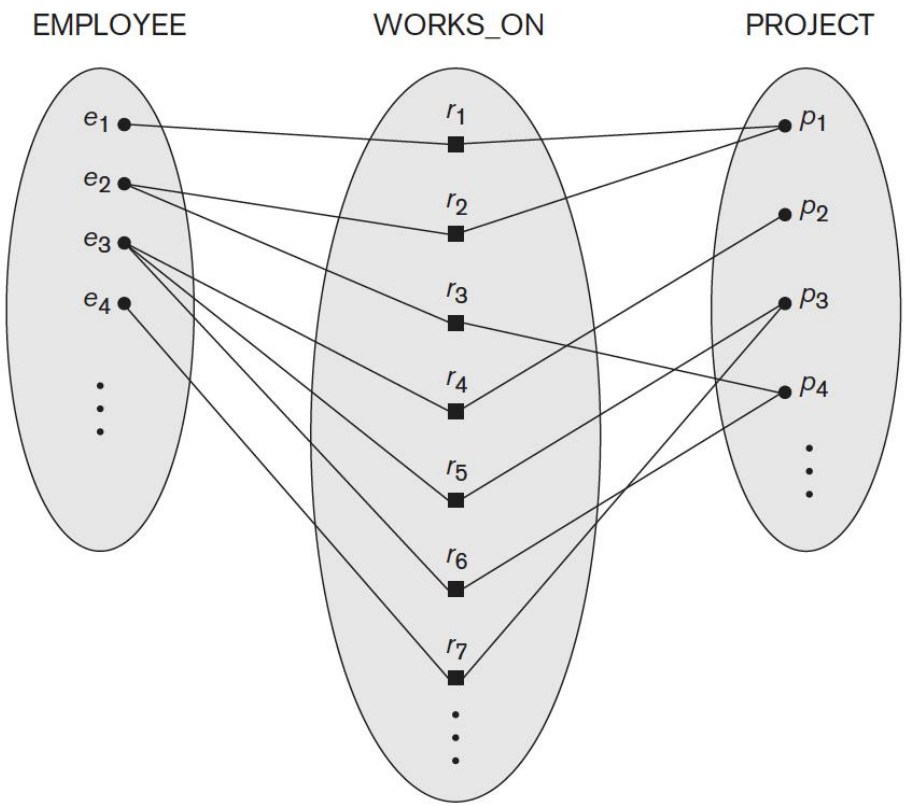


An entity in A is associated with at most one entity in B. An entity in B is associated with any number of entities in A.

E.g., an employee works for one department. A department can have many employees.

One–to-many (1:N) is the opposite of this relationship (Switch A and B).

MANY TO MANY (M:N) RELATIONSHIPS



An entity in A is associated with any number of entities in B and vice versa.

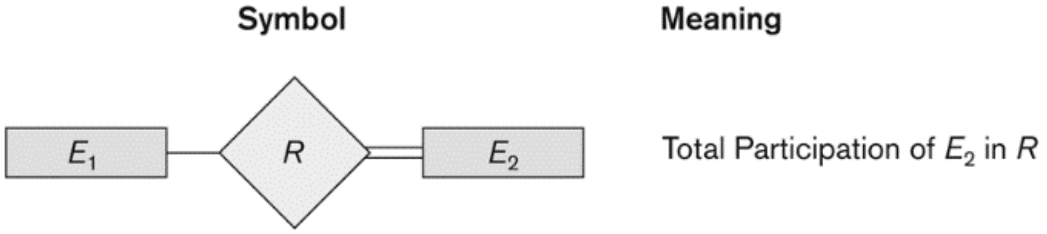
An employee works on many projects. A project is worked on by many employees.

OTHER CONSTRAINTS ON RELATIONSHIPS

Existence Dependency Constraint (also called **participation constraint**) specifies *minimum* participation:

- zero (optional participation, not existence-dependent)
- one or more (mandatory participation, existence-dependent, at least one)

In ER diagrams, mandatory participation (or existence dependency) is displayed as a double line, partial participation is represented by a single line.



TOTAL AND PARTIAL PARTICIPATION

Existence Dependency Constraints are referred to as **total or partial participation.**

Total participation: Every entity in the entity set must participate in at least one instance of the relationship set.

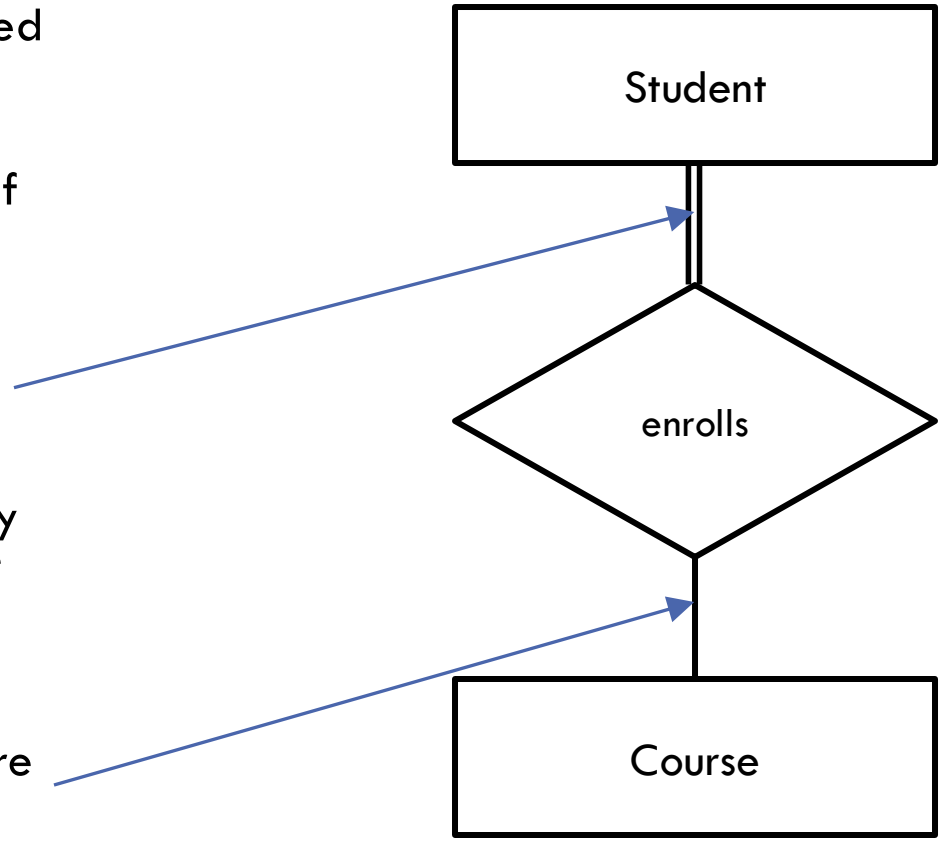
Represented with double line.

Example: each student must be enrolled in at least one course.

Partial participation: Every entity in the entity set may or may not participate in an instance of the relationship set.

Represented with one line.

Example: there might be courses in which there are no enrollments.



RECURSIVE RELATIONSHIP TYPE

A relationship type between the same participating entity type in **distinct roles**

Also called a **self-referencing** relationship type.

Example: the SUPERVISION relationship

EMPLOYEE participates twice in two distinct roles:

- supervisor (or boss) role
- supervisee (or subordinate) role

Each relationship instance relates two distinct EMPLOYEE entities:

- One employee in *supervisor* role
- One employee in *supervisee* role

DISPLAYING A RECURSIVE (UNARY) RELATIONSHIP

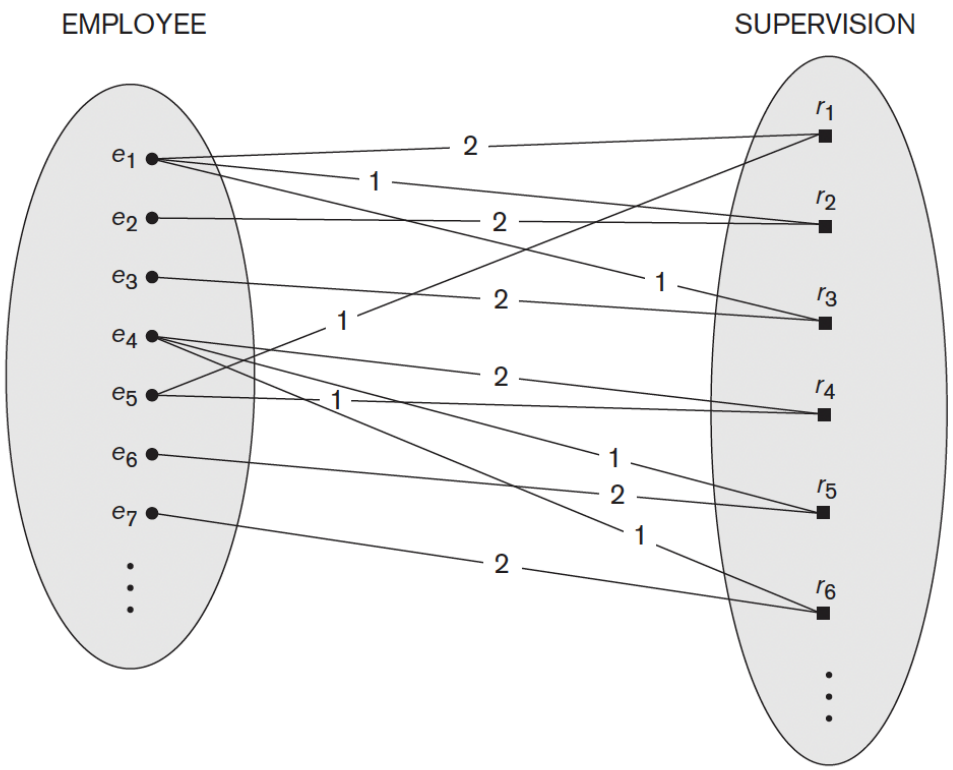
In a recursive relationship type.

- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).

In following figure, first role participation labeled with 1 and second role participation labeled with 2.

In ER diagram, need to display role names to distinguish participations.

RECURSIVE (UNARY) RELATIONSHIP



(1) the supervisor role

(2) the subordinate role

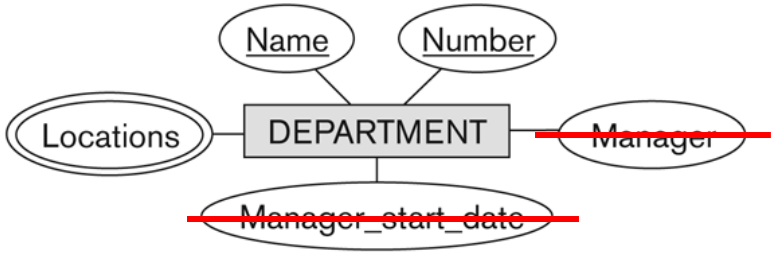
1 supervisor supervises many employees. Any employee is supervised by at most one supervisor.

E.g. e1 supervises e2 and e3, e4 supervises e6, and e7. e5 supervises e1 and e4.

DISCUSSION ON RELATIONSHIP TYPES

In the refined design, some attributes from the initial entity types are refined into relationships:

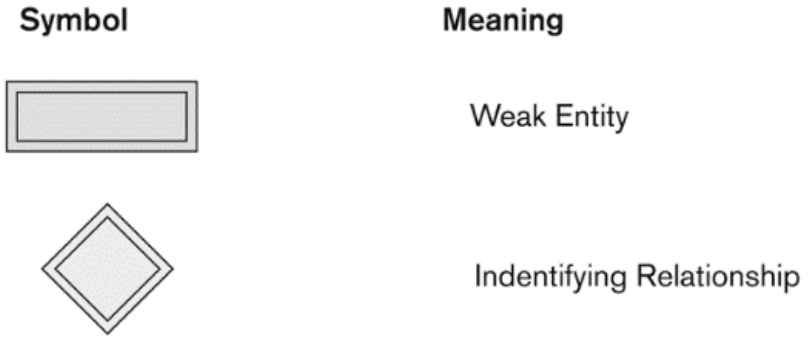
- Manager of DEPARTMENT -> MANAGES
- Works_on of EMPLOYEE -> WORKS_ON
- Department of EMPLOYEE -> WORKS_FOR
- etc



In general, more than one relationship type can exist between the same participating entity types

- MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
- Different meanings and different relationship instances.

WEAK ENTITY TYPES



An entity that does not have a key attribute and that is identification-dependent on another entity type.

A weak entity must participate in an identifying relationship type with an owner or identifying entity type

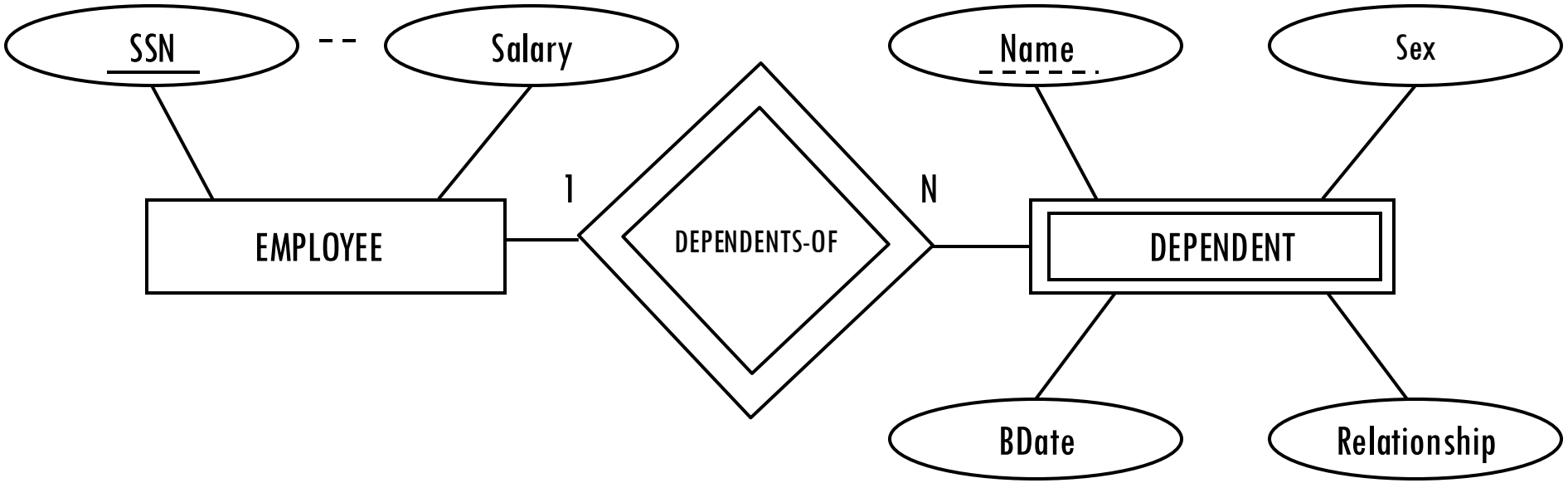
Weak entities are identified by the combination of:

- A partial key of the weak entity type
- The particular entity they are related to in the identifying relationship type

Example:

- A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
- Name of DEPENDENT is the *partial key*
- DEPENDENT is a *weak entity type*
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

WEAK ENTITY TYPE



ATTRIBUTES OF RELATIONSHIP TYPES

A relationship type can have attributes:

- For example, HoursPerWeek of WORKS_ON
- Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
  - A value of HoursPerWeek depends on a particular (employee, project) combination
- Most relationship attributes are used with M:N relationships
  - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

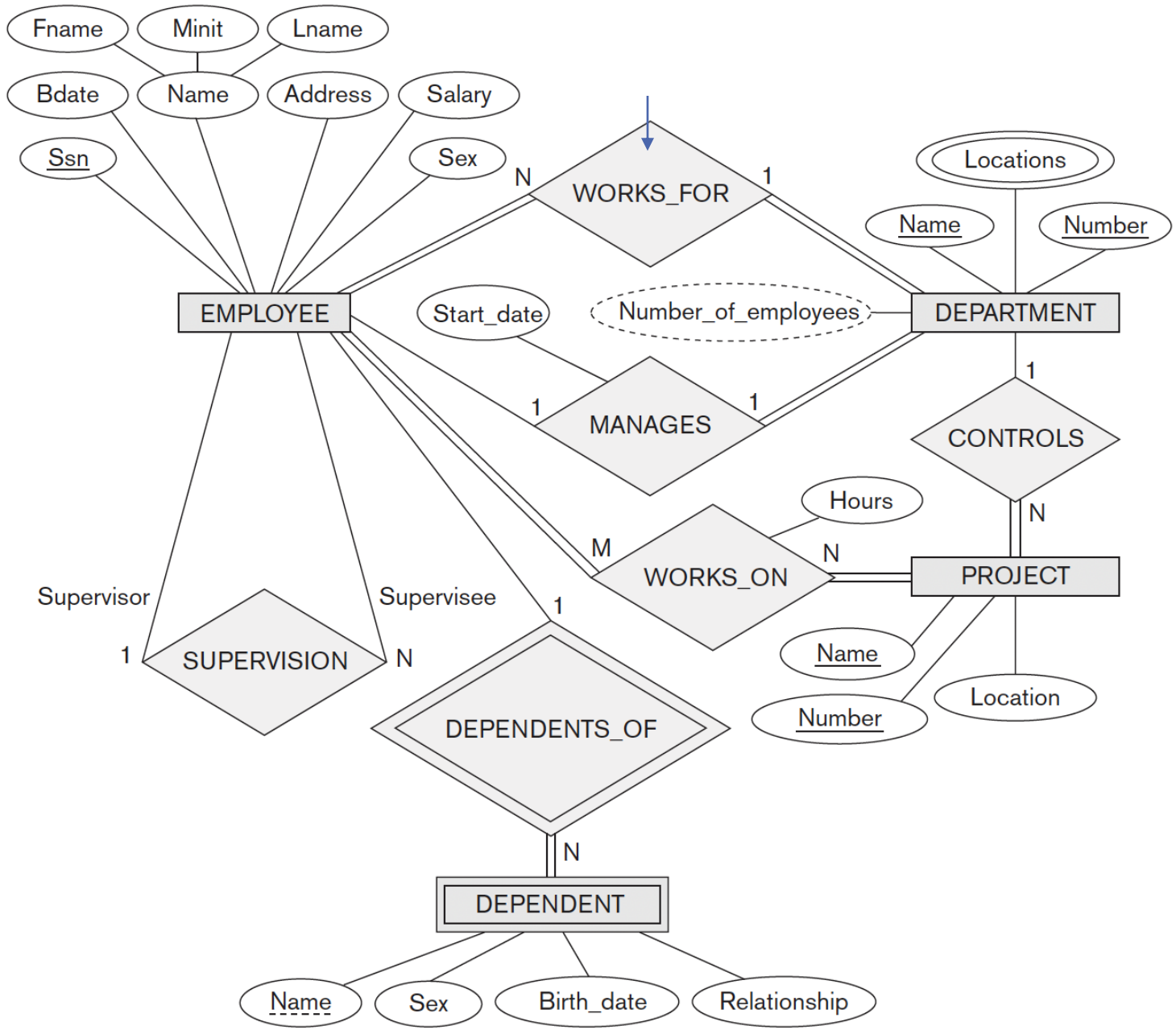
REFINING THE COMPANY DATABASE SCHEMA BY INTRODUCING RELATIONSHIPS

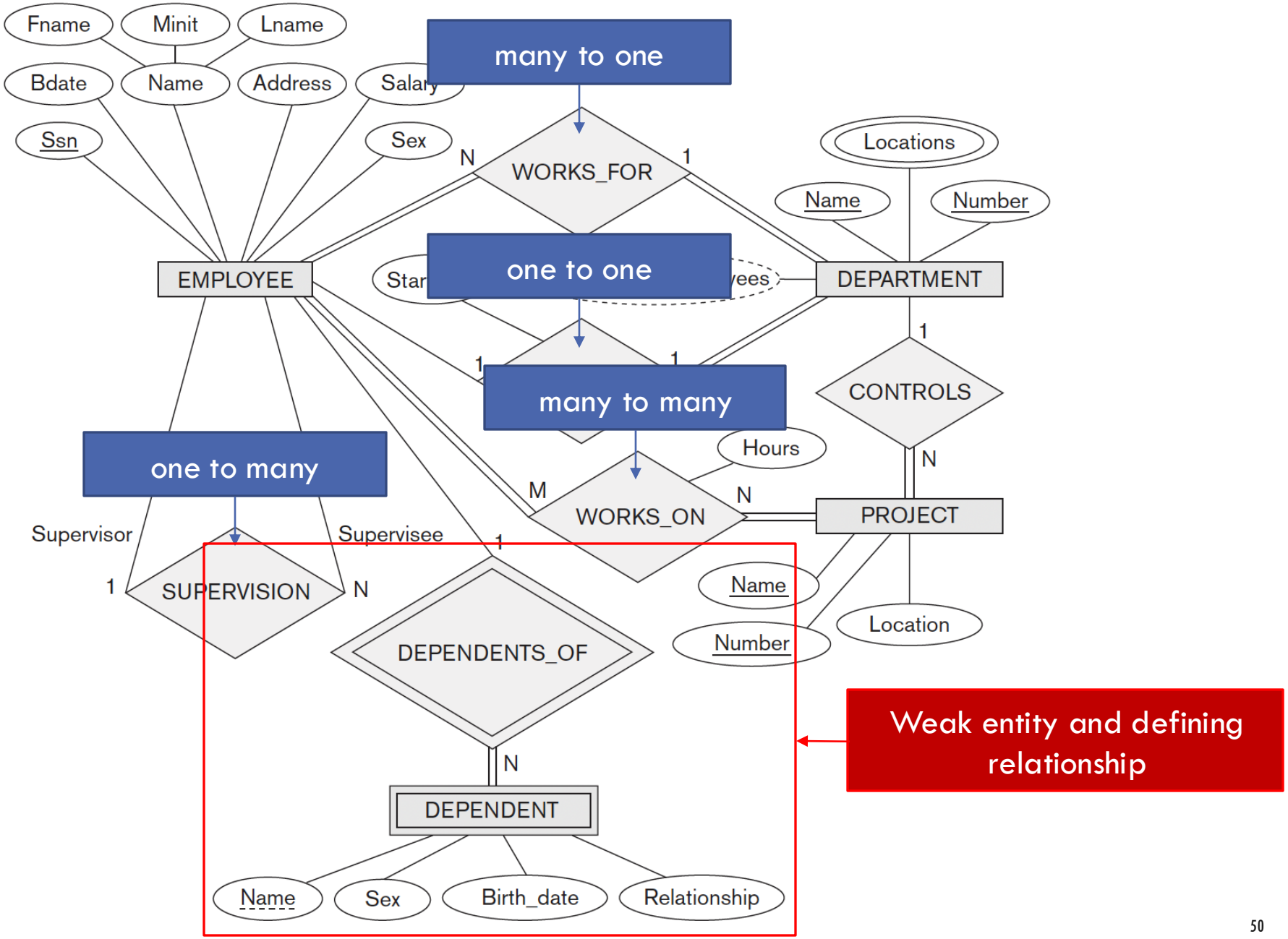
By examining the requirements, six relationship types are identified

All are *binary* relationships(degree 2)

Listed below with their participating entity types:

- WORKS_FOR (between EMPLOYEE, DEPARTMENT)
- MANAGES (also between EMPLOYEE, DEPARTMENT)
- CONTROLS (between DEPARTMENT, PROJECT)
- WORKS_ON (between EMPLOYEE, PROJECT)
- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
many to one
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
one to one
DEPARTMENT
1
1
1
many to many
CONTROLS
Hours
one to many
M
N
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship
Weak entity and defining relationship

ER DIAGRAM CONVENTIONS

One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema.

Nouns for entity type and attribute names and verbs for relationship type names.

Choosing binary relationship names to make the ER diagram readable from left to right and from top to bottom.

Entity type and relationship type names in uppercase letters, attribute names capitalized, and role names in lowercase.

ACTIVITY

Read the requirements carefully and analyze them

Make decisions on entities, attributes, and relationships and how to model them

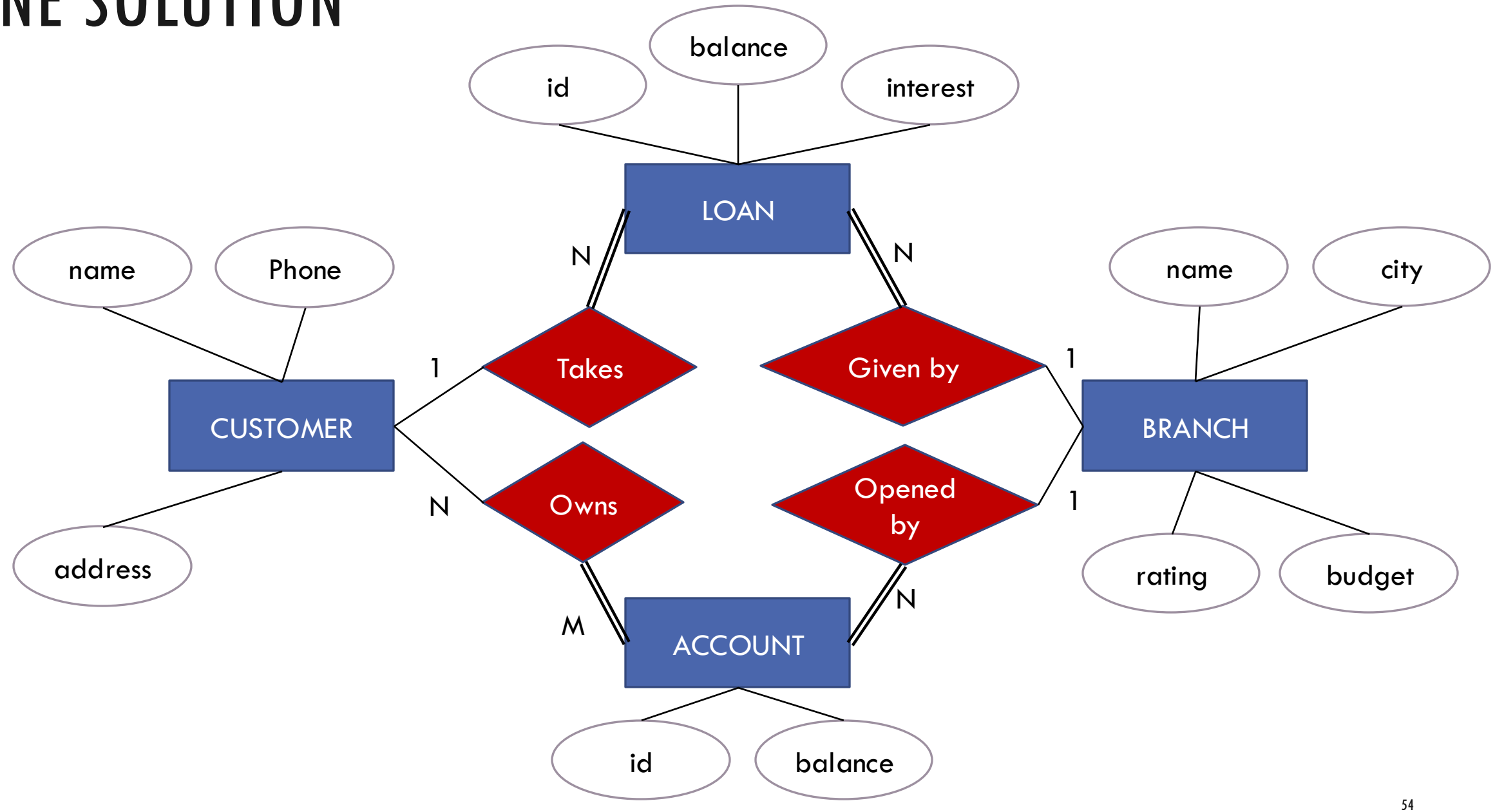
I will give you 5 minutes to produce a solution as a group.

CASE STUDY (SBS BANK)

We have a bank called SBS. With the following business requirements:

- The bank has multiple branches. Each branch is located in a specific city and has its own unique name. Every year the branch is assigned a yearly budget, and a rating (out of 10).
- A customer is identified by their name and phone number. The bank also keeps track of their address.
- The bank offers two services: accounts and loans to the customers. Each account and loan has a unique number and is created and maintained by a single branch.
- Each account is assigned to one or more customers and the balance cannot be negative.
- A loan is always assigned to a single customer, has interest rate and balance that cannot be negative.

ONE SOLUTION



ALTERNATIVE (MIN, MAX) NOTATION FOR RELATIONSHIP STRUCTURAL CONSTRAINTS:

Specified on each participation of an entity type E in a relationship type R

Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R

Default(no constraint): min=0, max=n (signifying no limit)

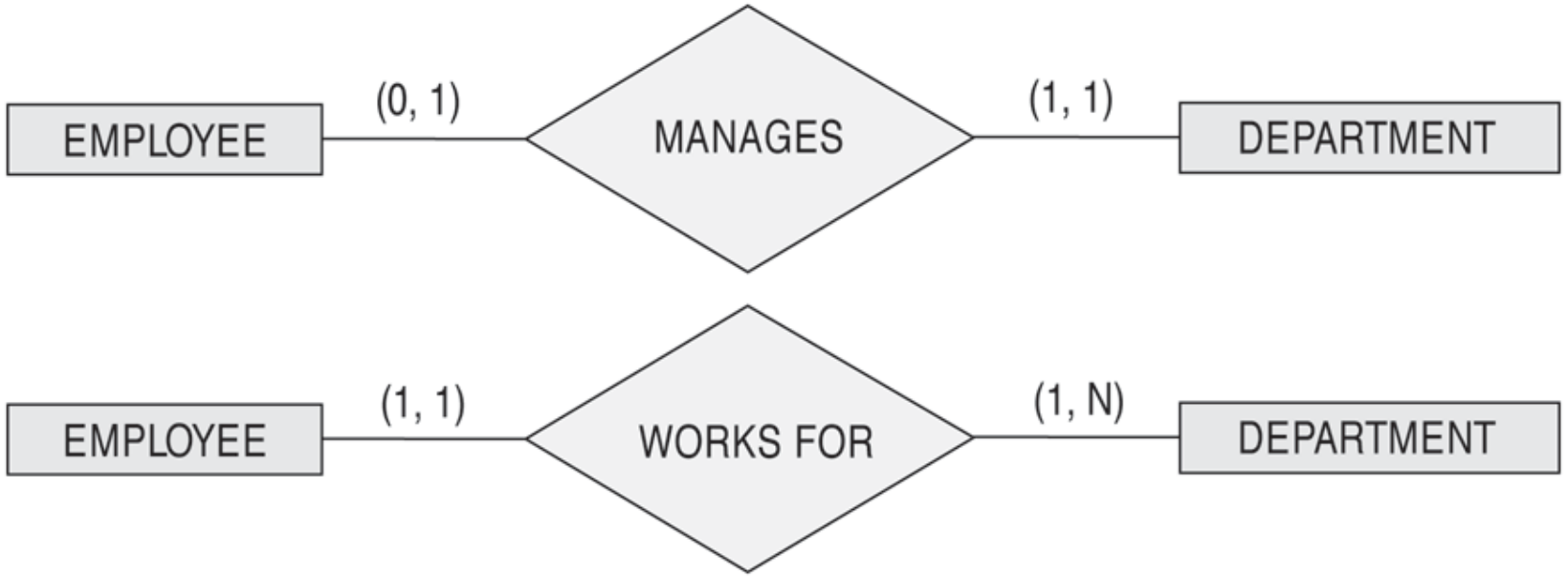
Must have min≤max, min≥0, max ≥1

Derived from the knowledge of mini-world constraints

Examples:

- A department has exactly one manager and an employee can manage at most one department.
  - Specify (0,1) for participation of EMPLOYEE in MANAGES
  - Specify (1,1) for participation of DEPARTMENT in MANAGES
- An employee can work for exactly one department but a department can have any number of employees.
  - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
  - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

THE (MIN,MAX) NOTATION FOR RELATIONSHIP CONSTRAINTS



Read the min,max numbers next to the entity type and looking **away from** the entity type

E.g. An Employee can manage a Department. A Department is managed by 1 Employee

E.g. Every Employee Works for a single Department. A Department has 1-N employees.

COMPANY ER SCHEMA DIAGRAM USING (MIN, MAX) NOTATION

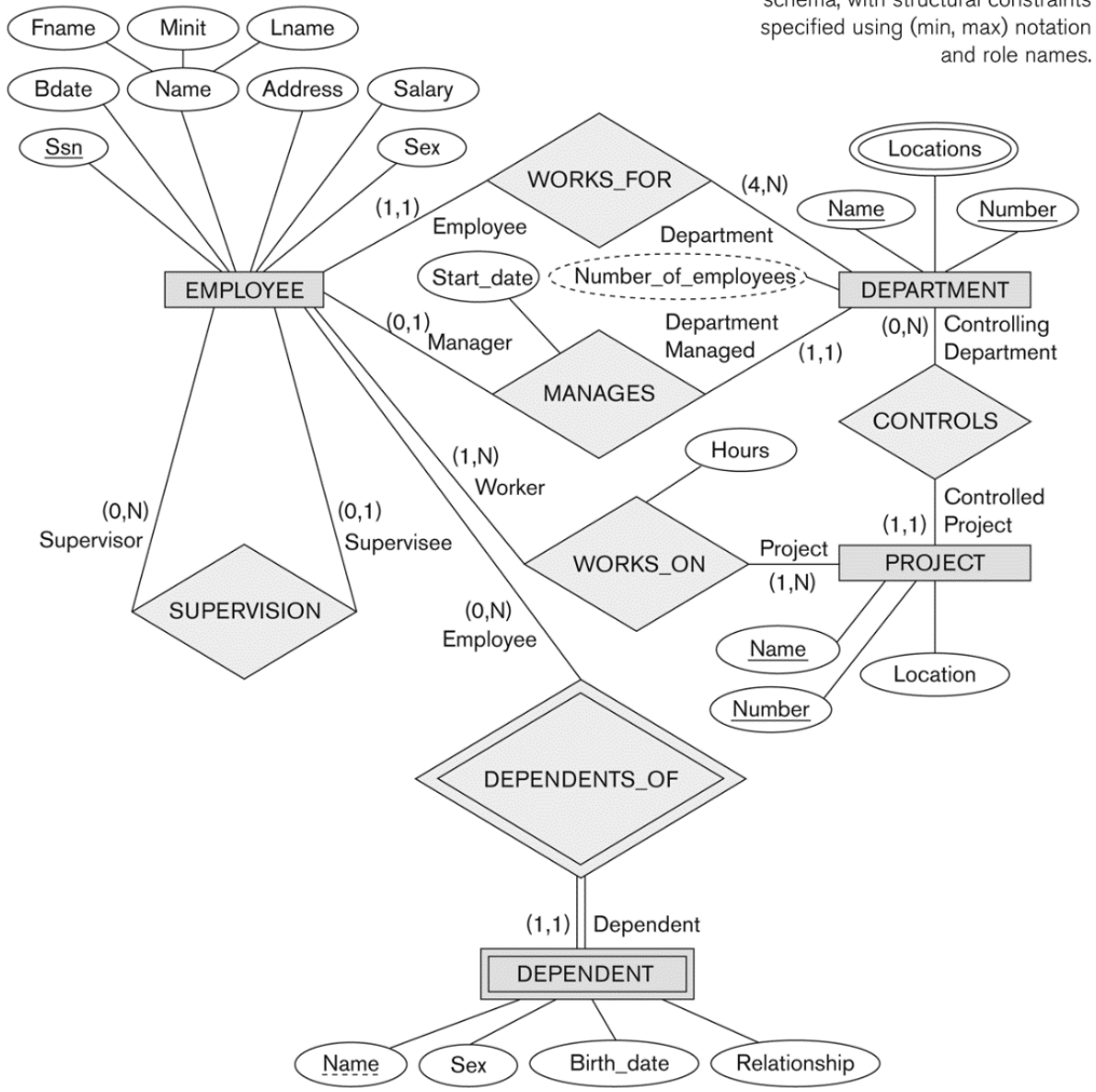
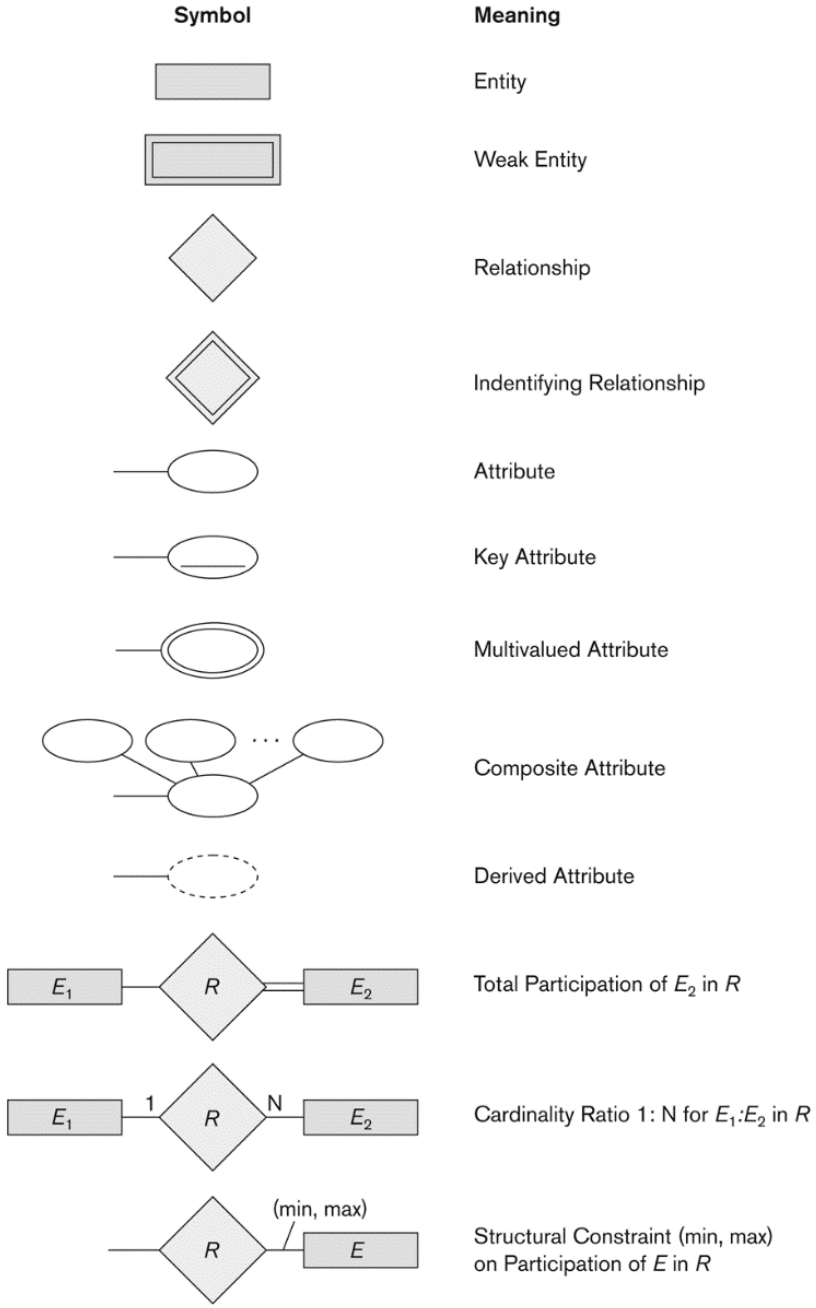


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

SUMMARY OF NOTATION FOR ER DIAGRAMS



RELATIONSHIPS OF HIGHER DEGREE

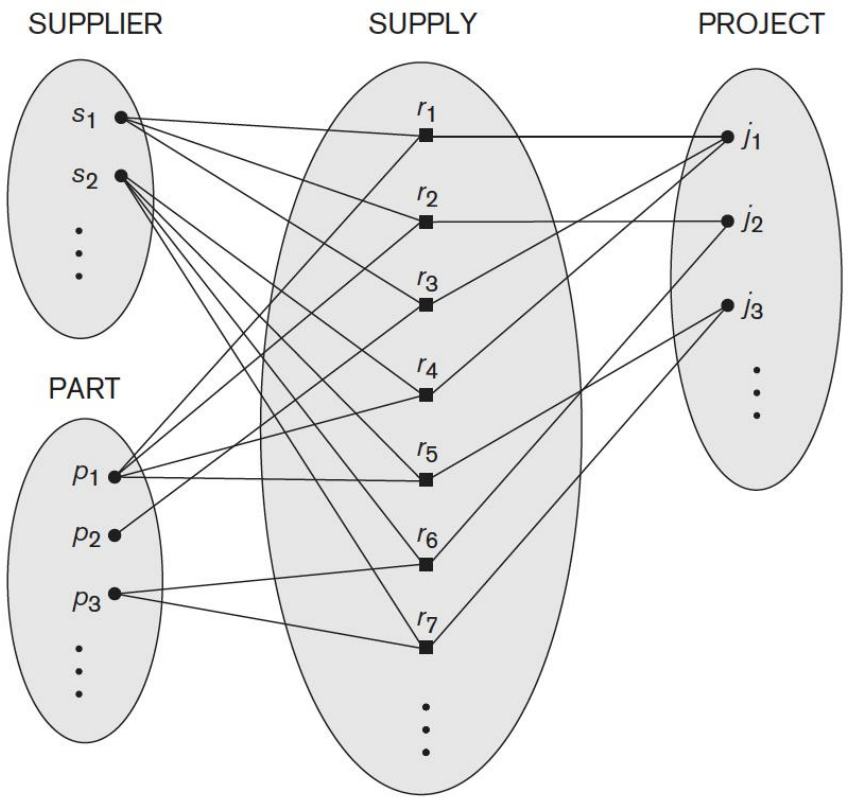
Relationship types of degree 2 are called binary

Relationship types of degree 3 are called ternary and of degree n are called n-ary

In general, an n-ary relationship is not equivalent to n binary relationships

Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships

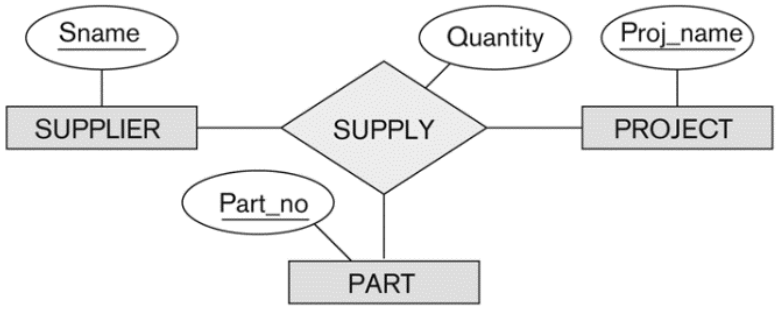
TERNARY RELATIONSHIPS



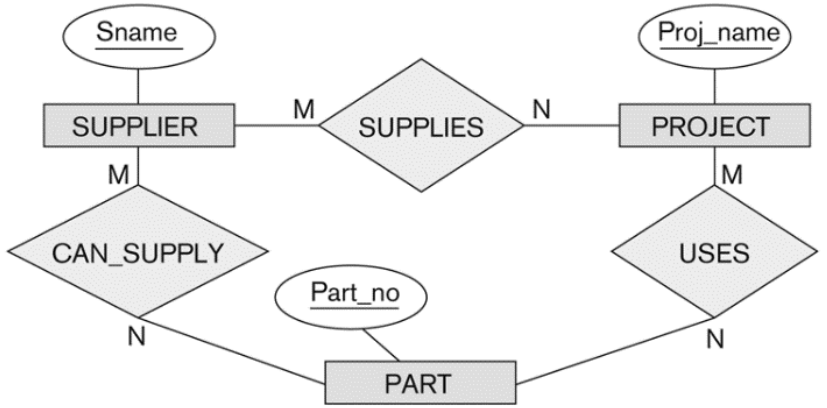
The relationship set of SUPPLY is a set of relationship instances (s, j, p), where the meaning is that s is a SUPPLIER who may SUPPLY one or many PARTs p to be used in one or many PROJECTs j. It is not necessary all SUPPLIERS supply parts or all PARTS are supplied

ALTERNATIVES TO TERNARY RELATIONSHIP

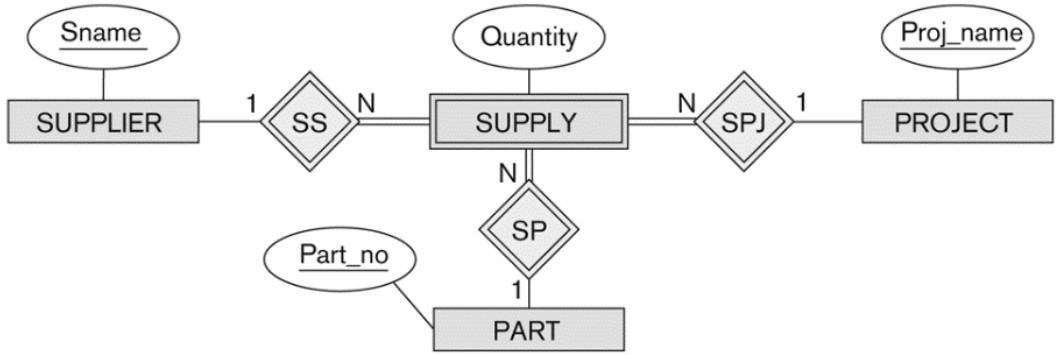
(a)



(b)



(c)



DISCUSSION OF N-ARY RELATIONSHIPS (N > 2)

In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on previous slide)

If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)

In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Figure 3.17c)

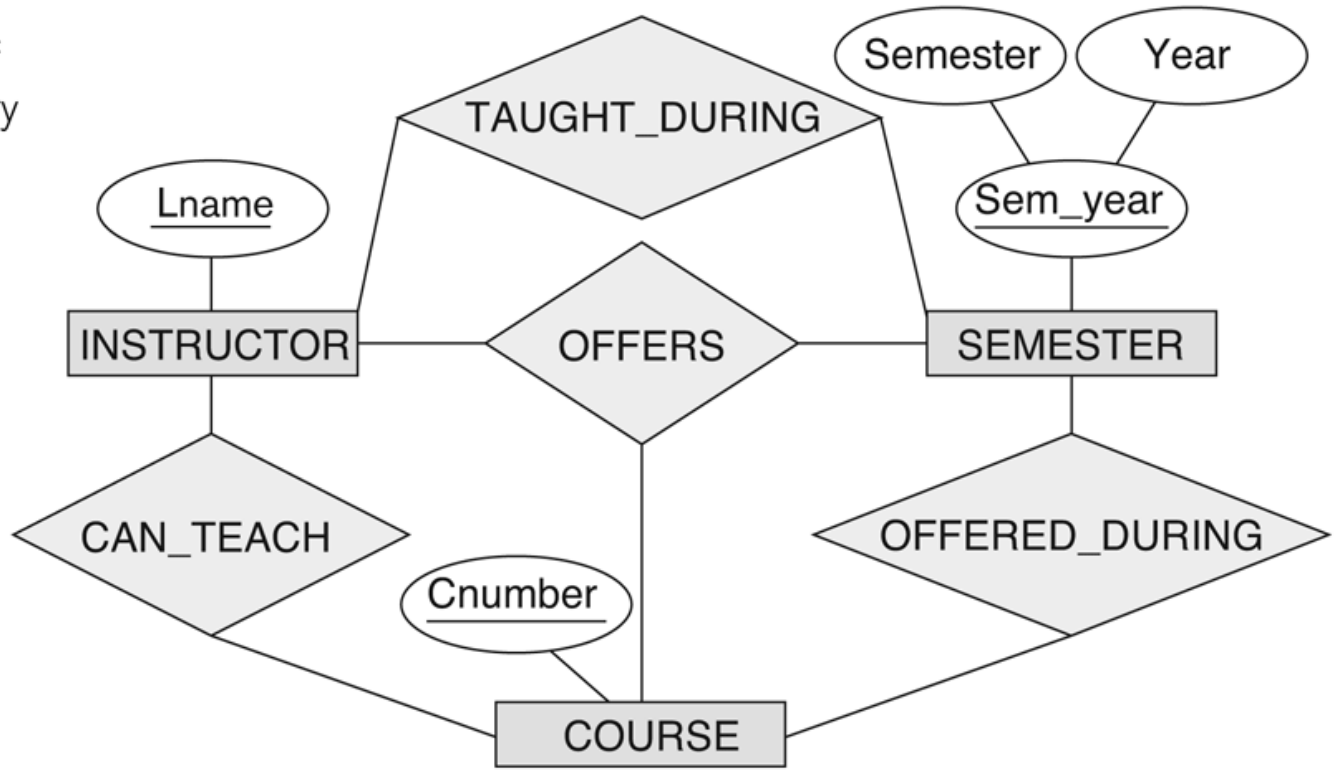
DISCUSSION OF N-ARY RELATIONSHIPS (N > 2)

If a particular binary relationship can always be derived from a higher-degree relationship, then it is redundant

For example, the TAUGHT_DURING, and OFFERED_DURING binary relationships in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships).

ANOTHER EXAMPLE OF A TERNARY RELATIONSHIP

Figure 3.18
Another example of ternary versus binary relationship types.



DISPLAYING CONSTRAINTS ON HIGHER-DEGREE RELATIONSHIPS

The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints.

A (min, max) on a participation here specifies that each entity is related to at least min and at most max relationship instances in the relationship set..

In general, both (min, max) and 1, M, or N are needed to describe the constraints fully.

Overall, the constraint specification is difficult and possibly ambiguous when we consider relationships of a degree higher than two.

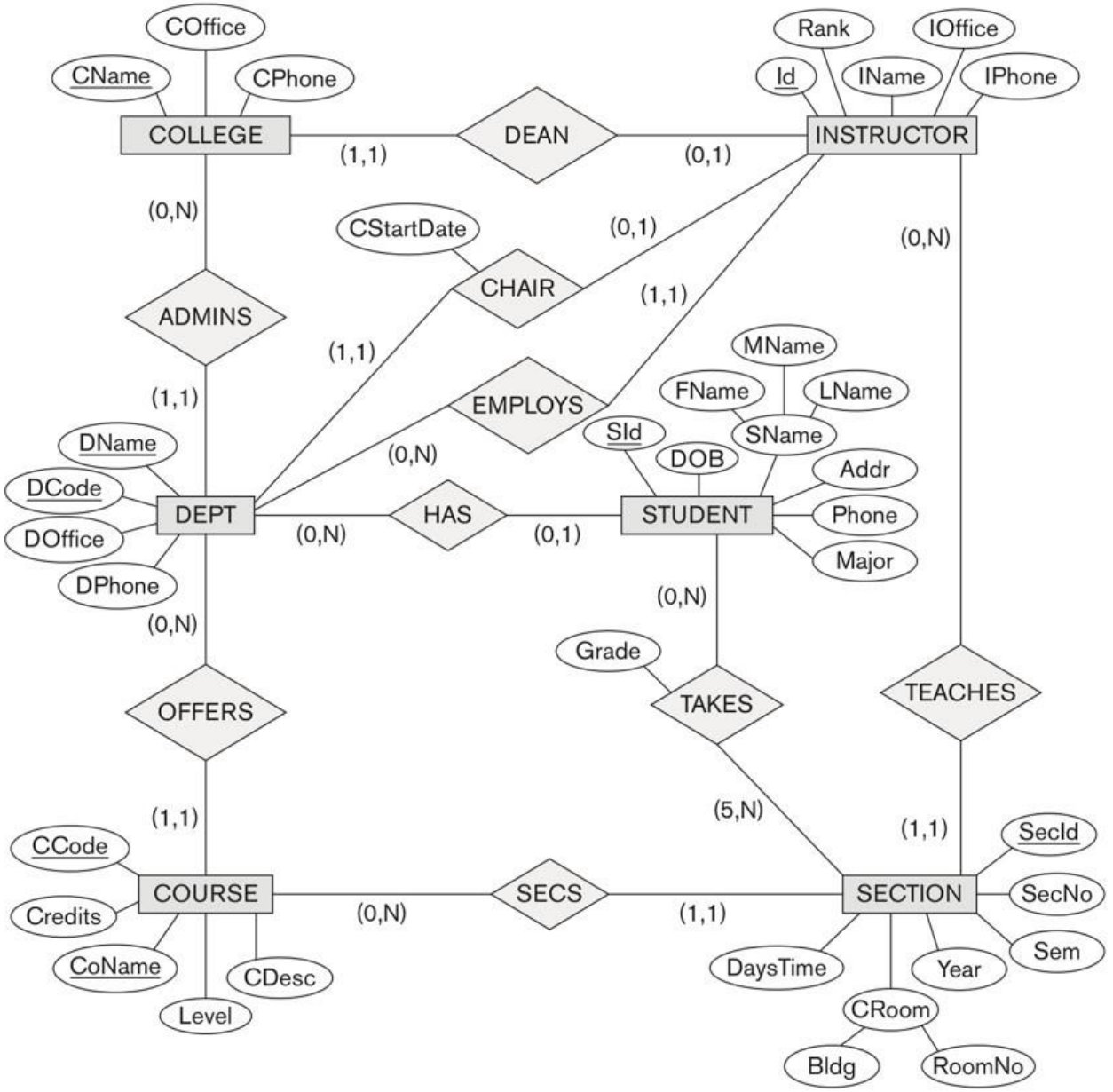
ANOTHER EXAMPLE: A UNIVERSITY DATABASE

To keep track of the enrollments in classes and student grades, another database is to be designed.

It keeps track of the COLLEGEs, DEPARTMENTs within each college, the COURSEs offered by departments, and SECTIONs of courses, INSTRUCTORs who teach the sections etc.

These entity types and the relationships among these entity types are shown on the next slide.

UNIVERSITY DATABASE ER DIAGRAM



CHAPTER SUMMARY

- ER Model Concepts: Entities, attributes, relationships
- We saw constraints in the ER model (Keys, participation, cardinality)
- The process of conceptual design is subjective, many possible designs
- Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship
- The important thing is to continue to refine your design. Your decisions will be more informed when we cover future chapters.

EXTENDED ENTITY-RELATIONSHIP (EER) MODEL (IN THE NEXT CHAPTER)

The entity relationship model in its original form did not support the specialization and generalization abstractions

Next chapter illustrates how the ER model can be extended with

- Type-subtype and set-subset relationships
- Specialization/Generalization Hierarchies
- Notation to display them in EER diagrams