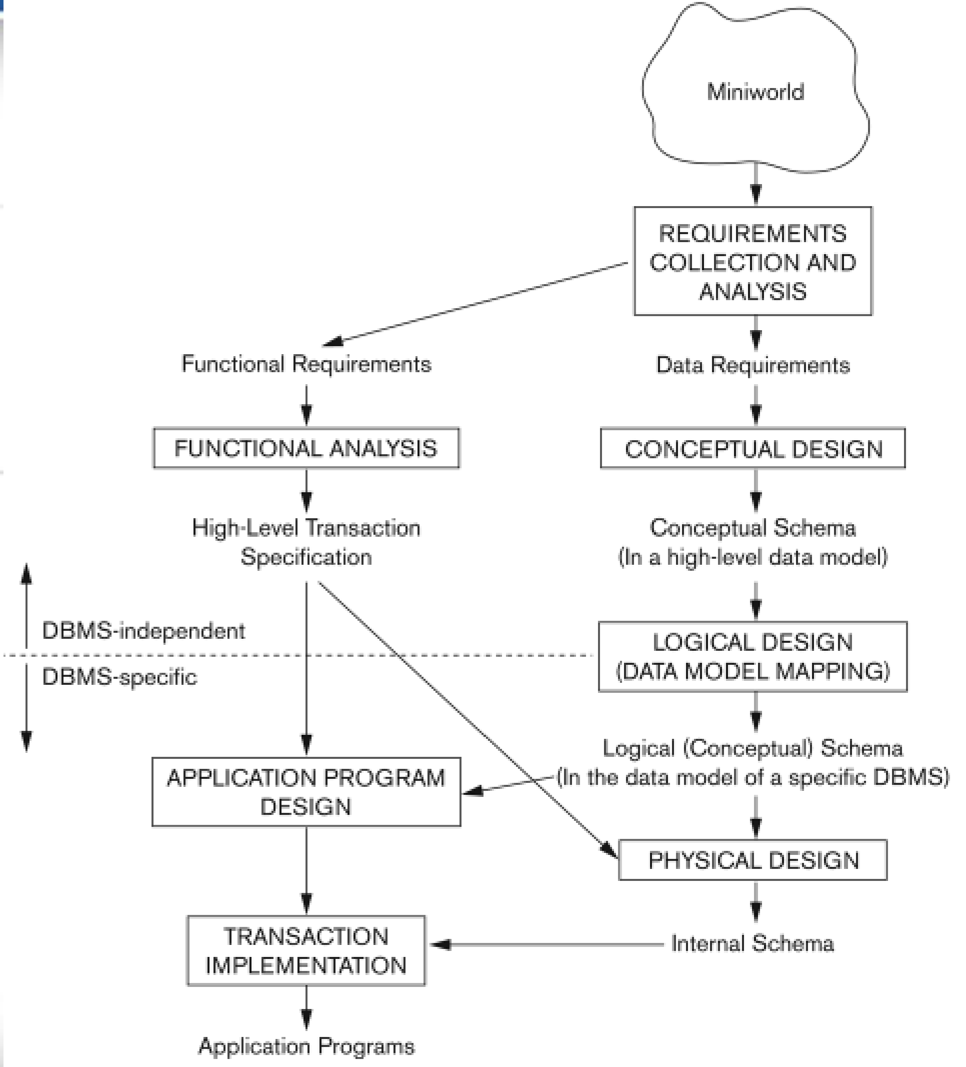
Data Modeling Using the Entity-Relationship (ER) Model



Miniworld

REQUIREMENTS COLLECTION AND ANALYSIS

Functional Requirements

Data Requirements

FUNCTIONAL ANALYSIS

CONCEPTUAL DESIGN

High-Level Transaction Specification

Conceptual Schema (In a high-level data model)

DBMS-independent

DBMS-specific

LOGICAL DESIGN (DATA MODEL MAPPING)

Logical (Conceptual) Schema (In the data model of a specific DBMS)

APPLICATION PROGRAM DESIGN

PHYSICAL DESIGN

TRANSACTION IMPLEMENTATION

Internal Schema

Application Programs

An entity-relationship diagram (ERD)

- a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities.
- a conceptual and representational model of data used to represent the entity framework infrastructure.
- crucial to creating a good database design.
- used as a high-level logical data model, which is useful in developing a conceptual design for databases.

Steps involved in creating an ERD include:

1. IDENTIFYING AND DEFINING THE ENTITIES.

2. DETERMINING ALL INTERACTIONS BETWEEN THE ENTITIES.

3. ANALYZING THE NATURE OF INTERACTIONS/DETERMINING THE CARDINALITY OF THE RELATIONSHIPS.

4. CREATING THE ERD.

Elements of ERD

Entity

An entity is a real-world item or concept that exists on its own.

Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

Attribute

An attribute of an entity is a particular property that describes the entity.

Relationship

A relationship is the association that describes the interaction between entities.

Cardinality

In the context of ERD, is the number of instances of one entity that can, or must, be associated with each instance of another entity. In general, there may be one-to-one, one-to-many, or many-to-many relationships.

Employee Entity

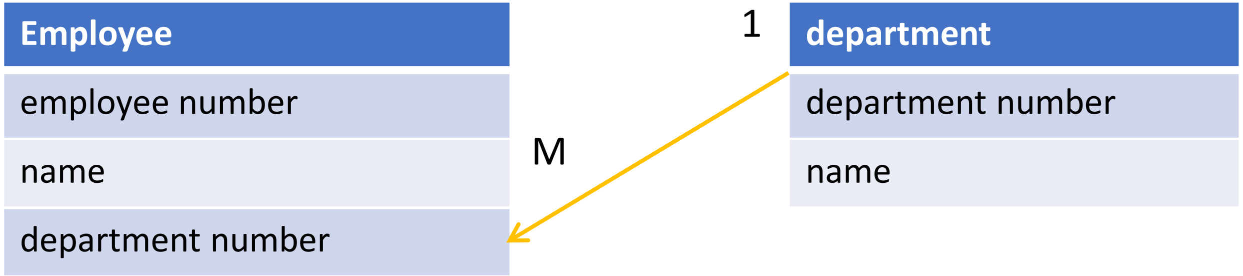
Employee Attribute :

employee number name
Department number

Department Entity

Department Attribute :

department number
name



Employee
employee number
name
department number

department
department number
name

Components of E-R Diagram

- **Entity relational diagram (ER Diagram)** is used to represent the requirement analysis at the conceptual design stage.
  - the database is designed from the ERD or ERD is converted to the database.
    - Each entity in the ERD corresponds to a table in the database.
    - The attributes of any an entity correspond to field of a table.
    - The ERD is converted to the database.

Elements of an ERD

ENTITIES

- ✓ Entities are objects or concepts that represent important data.
- ✓ They are typically nouns *(customer*, *supervisor*, *location*, or *promotion)*.

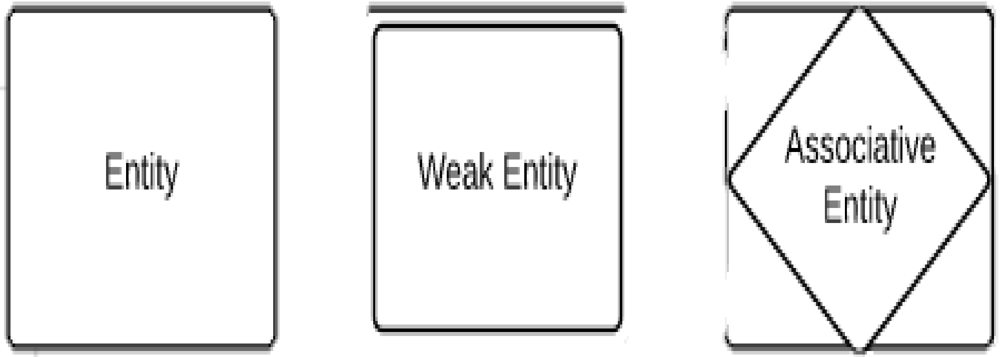
Entity

Strong entities exist independently from other entity types. They always possess one or more attributes that uniquely distinguish each occurrence of the entity.

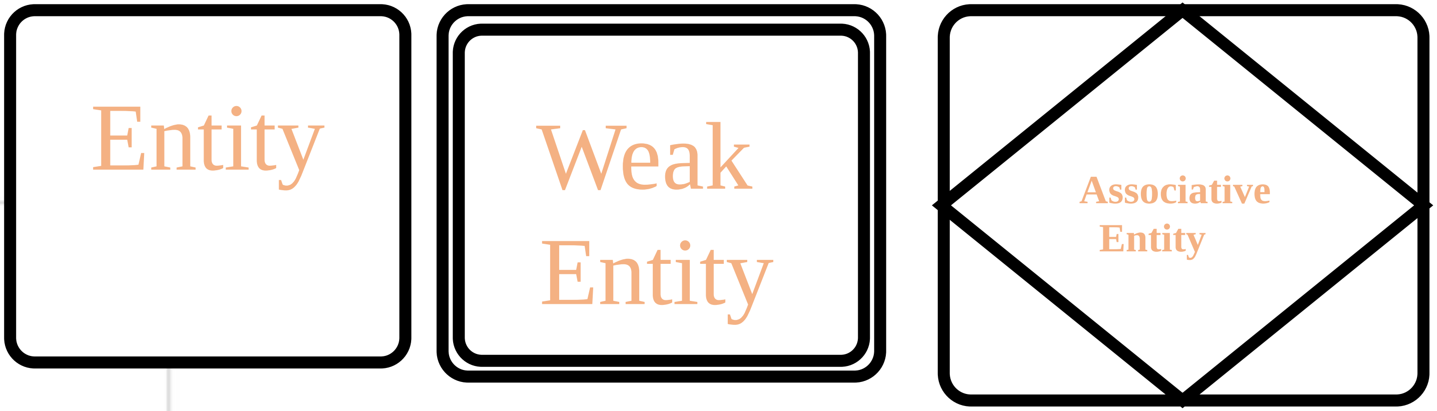
Weak entities depend on some other entity type. They don't possess unique attributes (also known as a primary key) and have no meaning in the diagram without depending on another entity. This other entity is known as the owner.

Associative entities are entities that associate the instances of one or more entity types. They also contain attributes that are unique to the relationship between those entity instances.

Entity



Entity



Entity

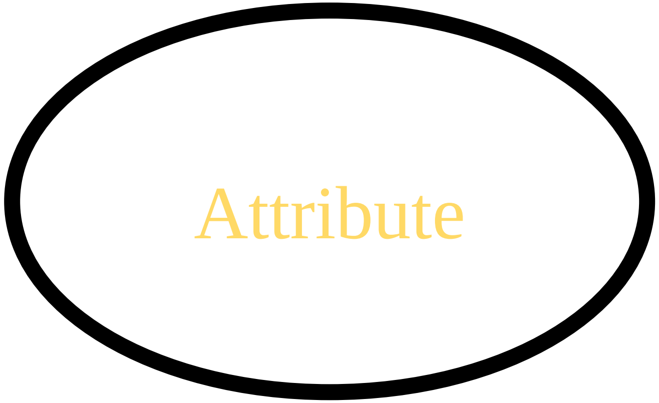
Weak Entity

Associative Entity

Elements of an ERD

ATTRIBUTES

- **Attributes** are characteristics of either an entity, a many-to-many relationship, or a one-to-one relationship.



Attribute

Types of Attributes

Simple

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

Composite

- The attribute may be composed of several components. For example:
  - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
  - Name(FirstName, MiddleName, LastName).
  - Composition may form a hierarchy where some components are themselves composite.

Multi-valued

- An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
  - Denoted as {Color} or {PreviousDegrees}.

Types of 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

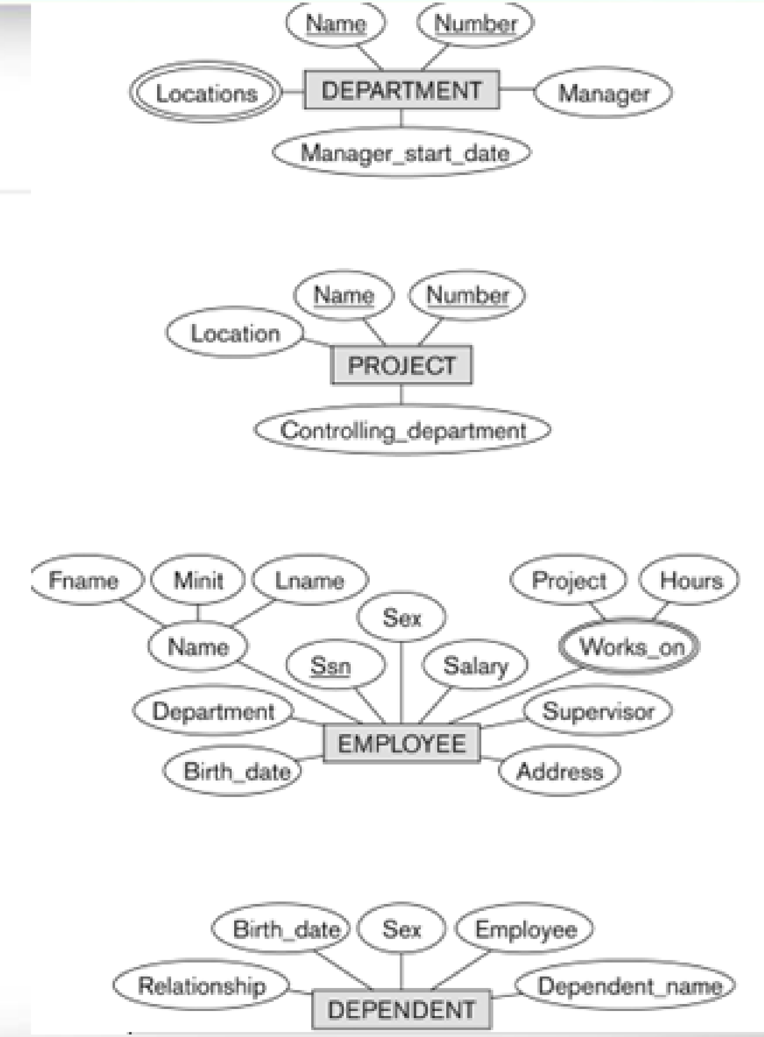
Example COMPANY Database

- 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 unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
  - A department controls several projects, each of which has a unique name, a unique number, and a single location.
  - The database will store each employee’s name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
  - The database will keep track of the dependents of each employee for insurance purposes, including each dependent’s first name, sex, birth date, 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

Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT



Entity Types and Key Attributes

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.

Entity Types and Key Attributes (2)

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).

Entity Set

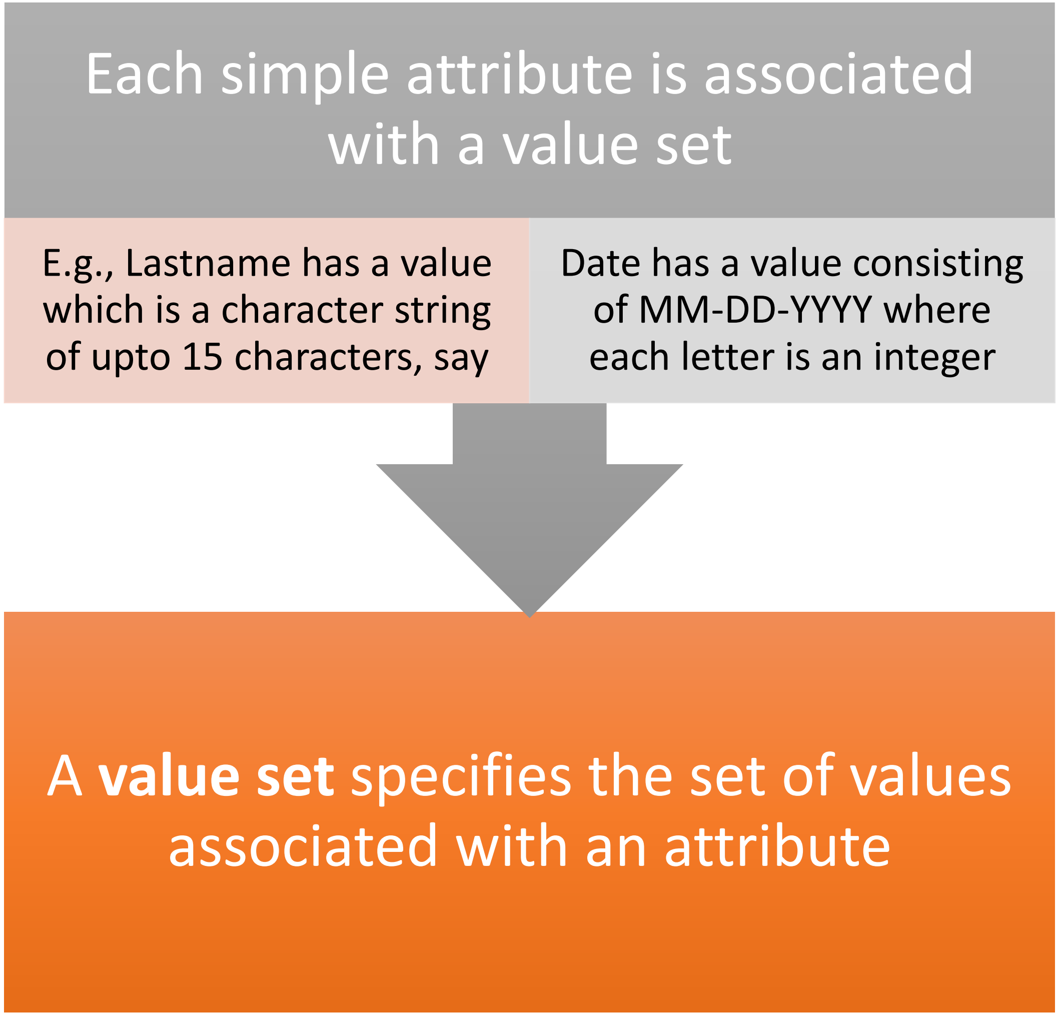
Each entity type will have a collection of entities stored in the database

- Called the **entity set** or sometimes **entity collection**

However, entity type and entity set may be given different names

Entity set is the current *state* of the entities of that type that are stored in the database

Value Sets (Domains) of Attributes



Each simple attribute is associated with a value set

E.g., Lastname has a value which is a character string of upto 15 characters, say

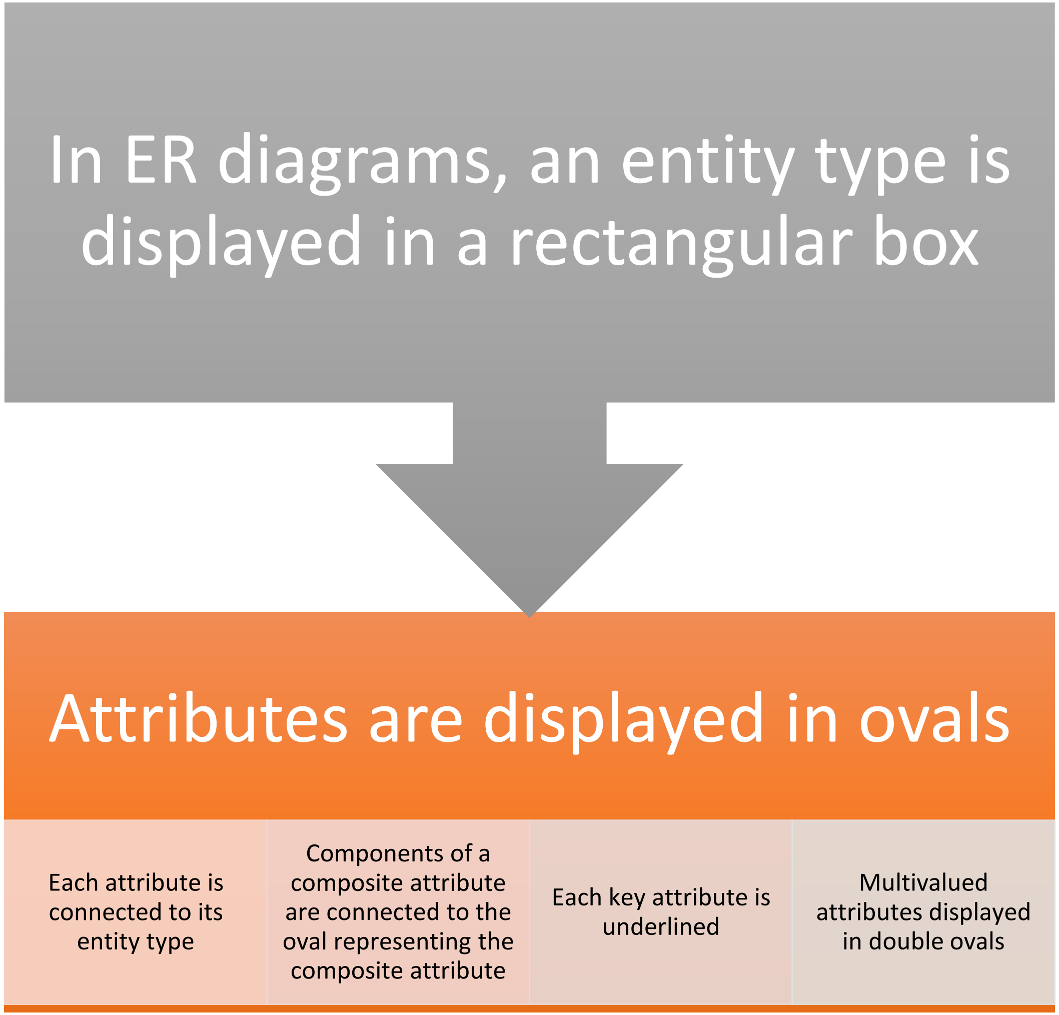
Date has a value consisting of MM-DD-YYYY where each letter is an integer

A **value set** specifies the set of values associated with an attribute

Attributes and Value Sets

- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit
- Mathematically, an attribute A for an entity type E whose value set is V is defined as a function
- A : E -> P(V)
- Where P(V) indicates a power set (which means all possible subsets) of V. The above definition covers simple and multivalued attributes.
- We refer to the value of attribute A for entity e as A(e).

Displaying an Entity type

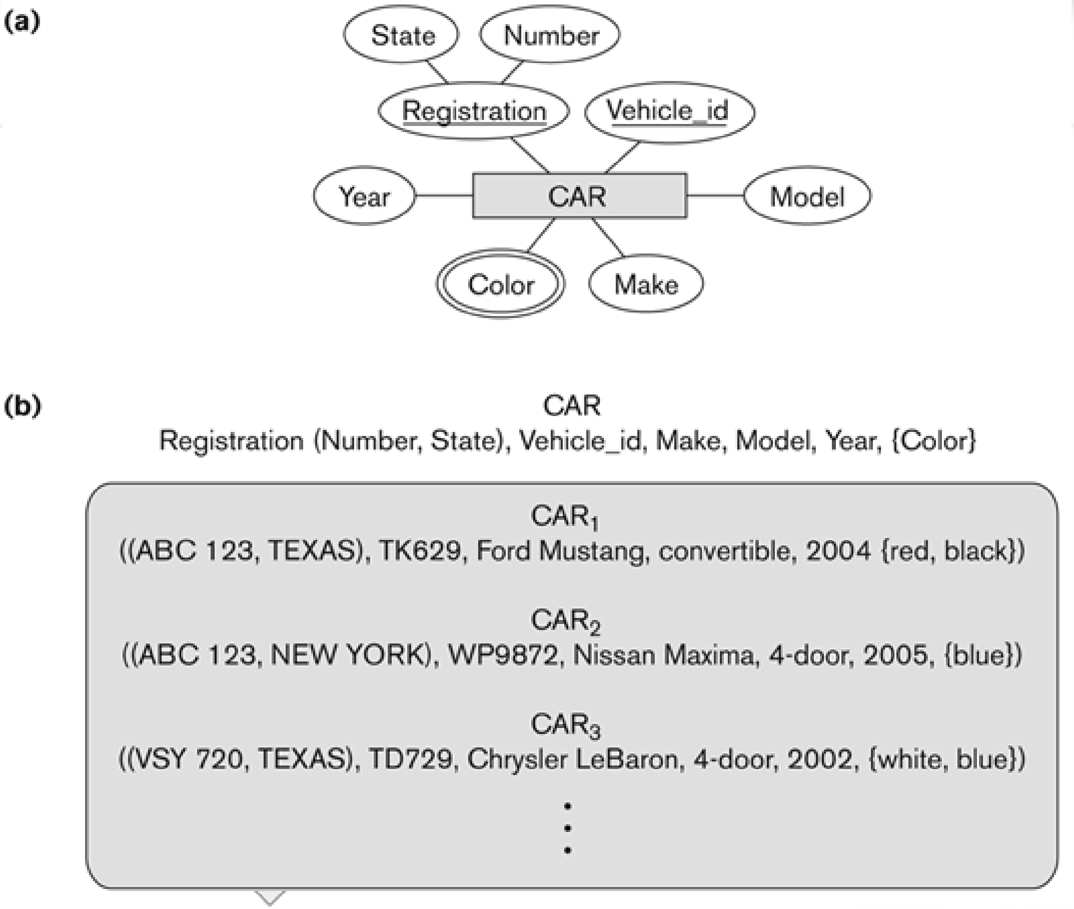


In ER diagrams, an entity type is displayed in a rectangular box

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

Entity Type CAR with two keys and a corresponding 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})

⋮

Elements of an ERD

RELATIONSHIPS

- **Relationships** are meaningful associations between or among entities.
- They are usually verbs, e.g. *assign*, *associate*, or *track*.
- A relationship provides useful information that could not be discerned with just the entity types.

Relationships and Relationship Types

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.

The degree of a relationship type is the number of participating entity types.

Both MANAGES and WORKS_ON are *binary* relationships.

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

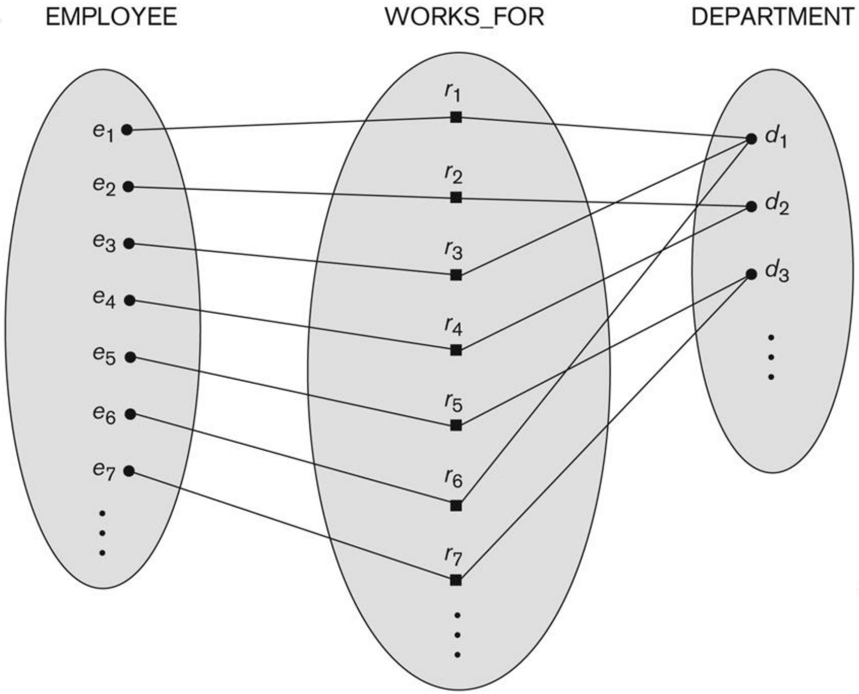


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

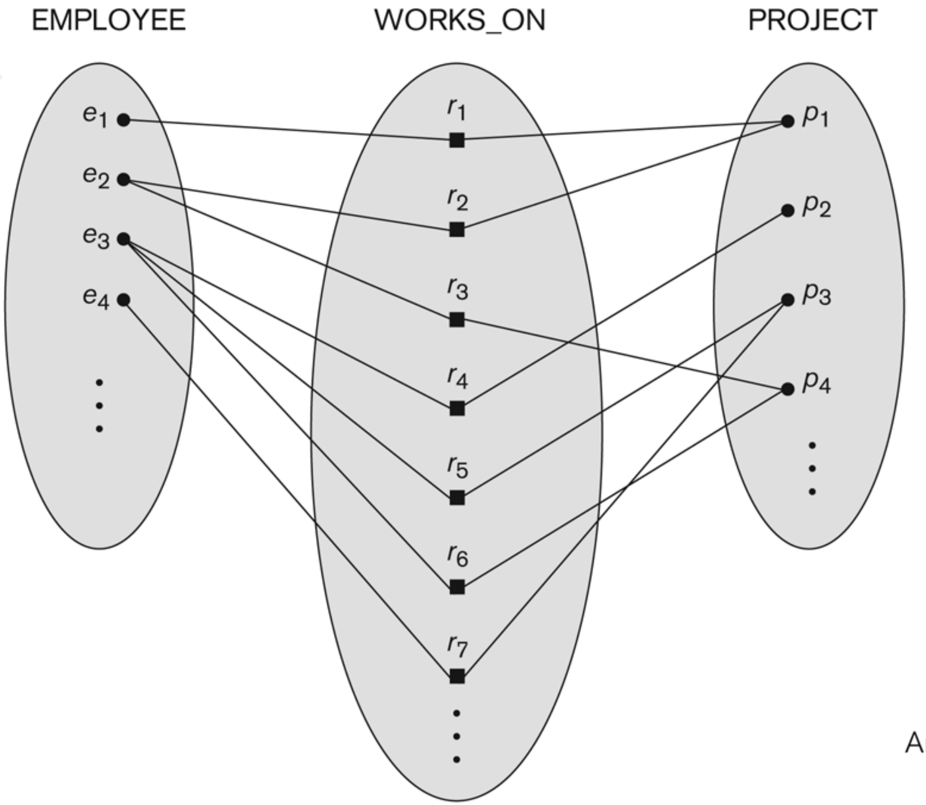


Figure 3.13
An M:N relationship, WORKS_ON.

Relationship type vs. relationship set

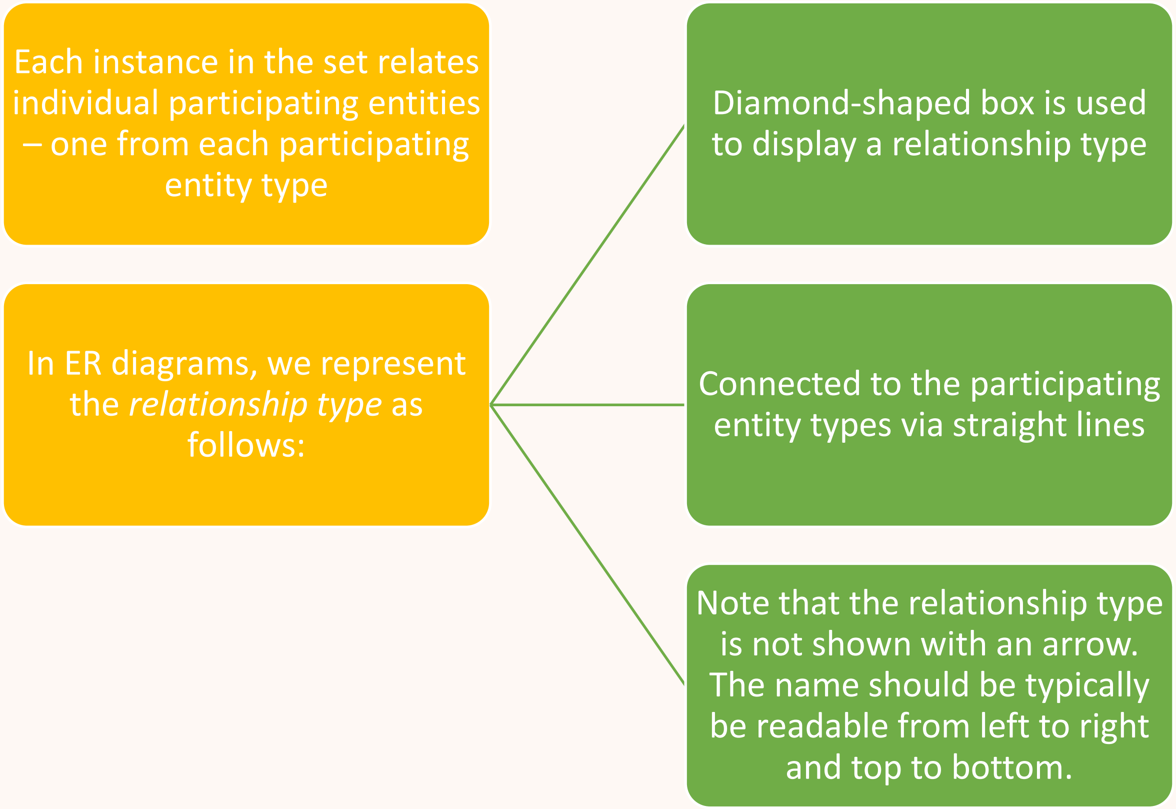
Relationship Type:

- 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 type vs. relationship set (2)



Each instance in the set relates individual participating entities – one from each participating entity type

In ER diagrams, we represent the *relationship type* as follows:

Diamond-shaped box is used to display a relationship type

Connected to the participating entity types via straight lines

Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.

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)

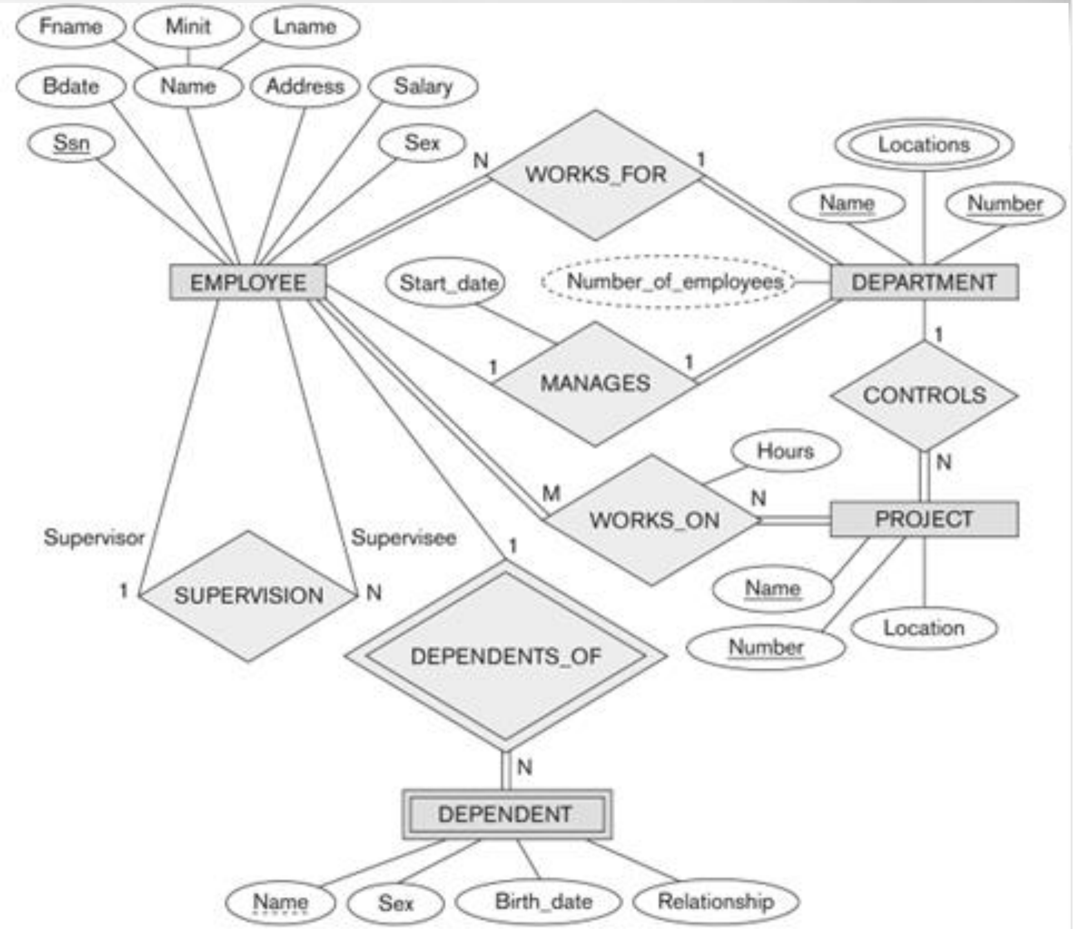
CONTROLS (between DEPARTMENT, PROJECT)

WORKS_ON (between EMPLOYEE, PROJECT)

SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))

DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are:
WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



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

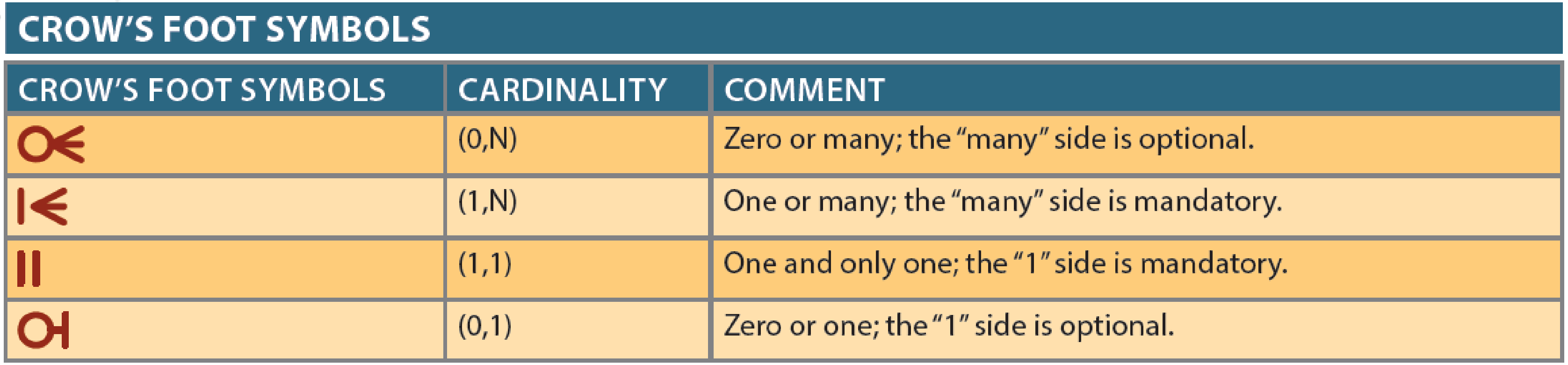
MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT

Different meanings and different relationship instances.

Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Cardinality Ratio (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many (M:N)
  - Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Crow's Foot Symbols



CROW'S FOOT SYMBOLS

CROW'S FOOT SYMBOLS	CARDINALITY	COMMENT
O⋲	(0,N)	Zero or many; the "many" side is optional.
|⋲	(1,N)	One or many; the "many" side is mandatory.
||	(1,1)	One and only one; the "1" side is mandatory.
O|	(0,1)	Zero or one; the "1" side is optional.

Many-to-one (N:1) Relationship

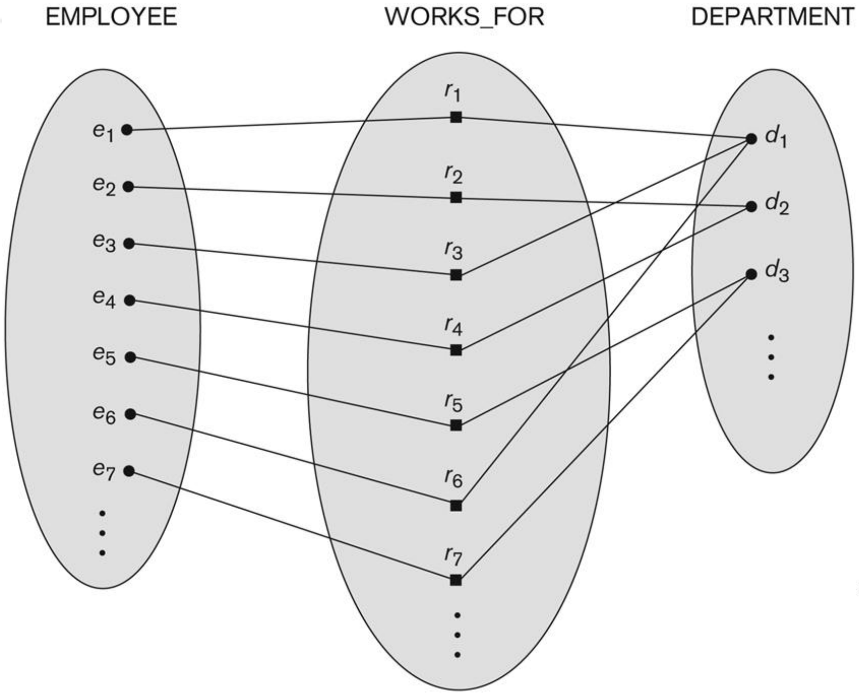


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many-to-many (M:N) Relationship

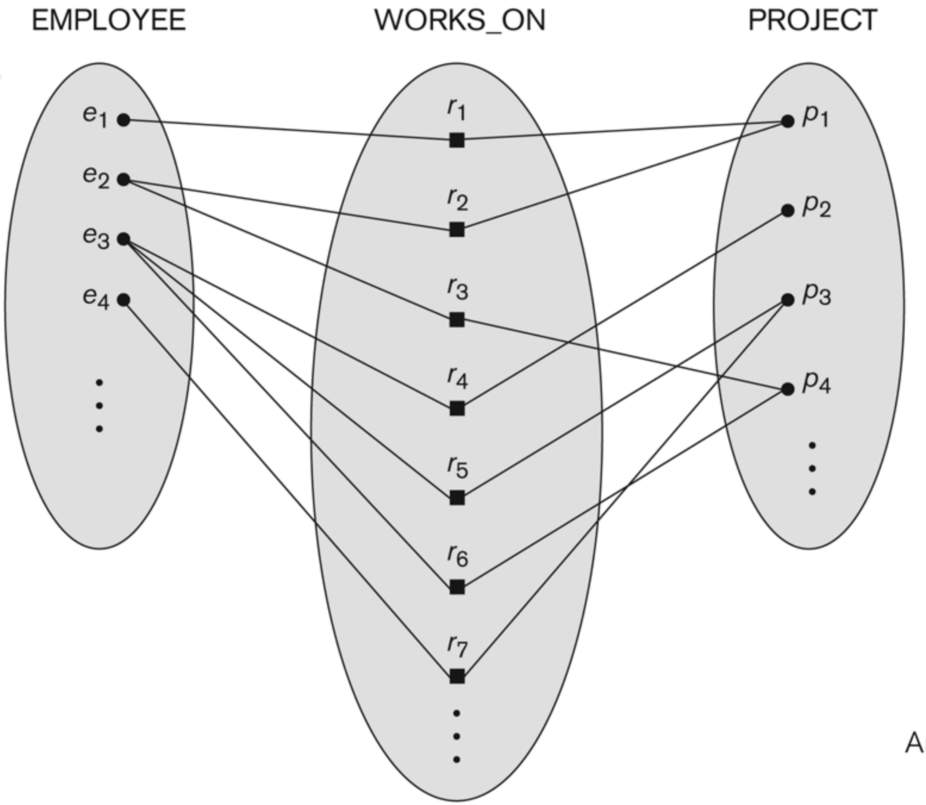


Figure 3.13
An M:N relationship, WORKS_ON.

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 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).

A Recursive Relationship Supervision

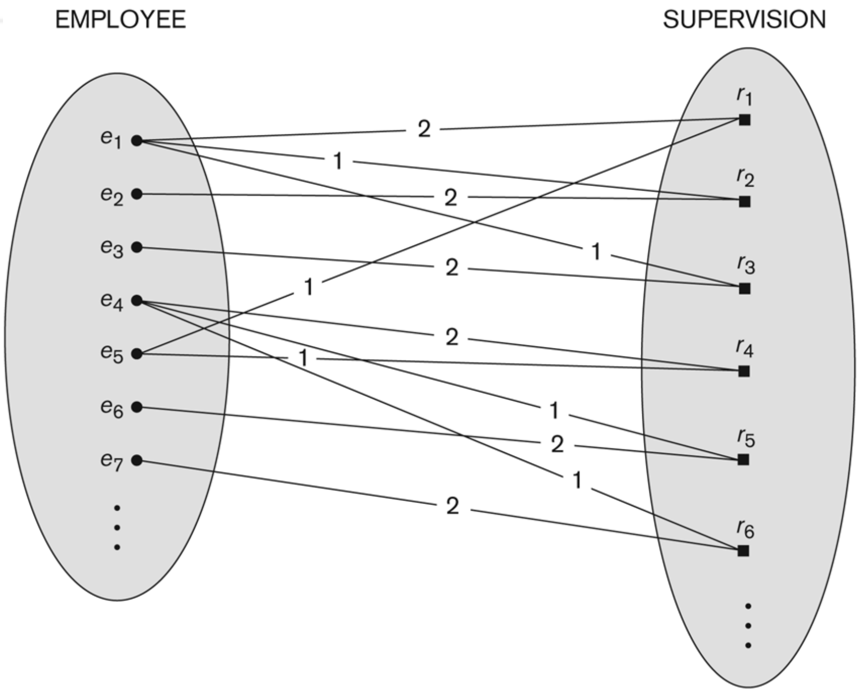


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Recursive Relationship Type is: SUPERVISION (participation role names are shown)

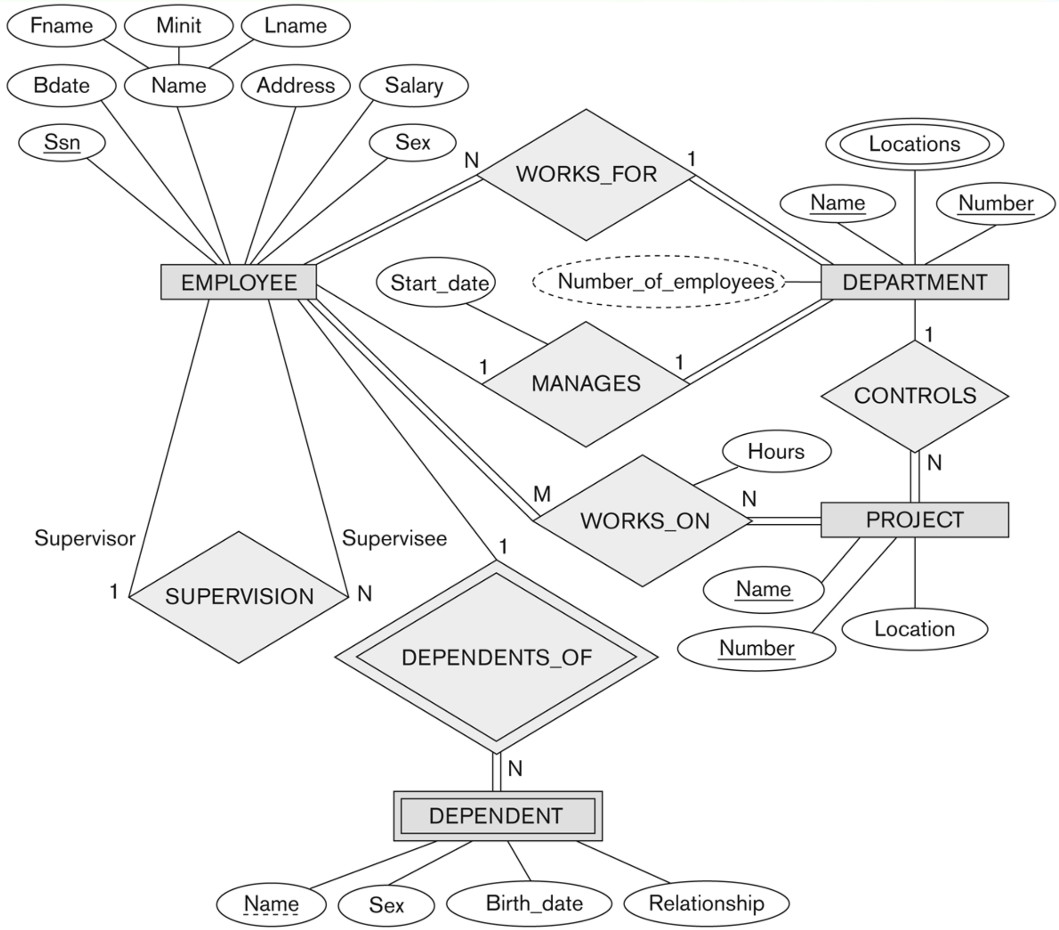


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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
- 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

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

Example Attribute of a Relationship Type: Hours of WORKS_ON

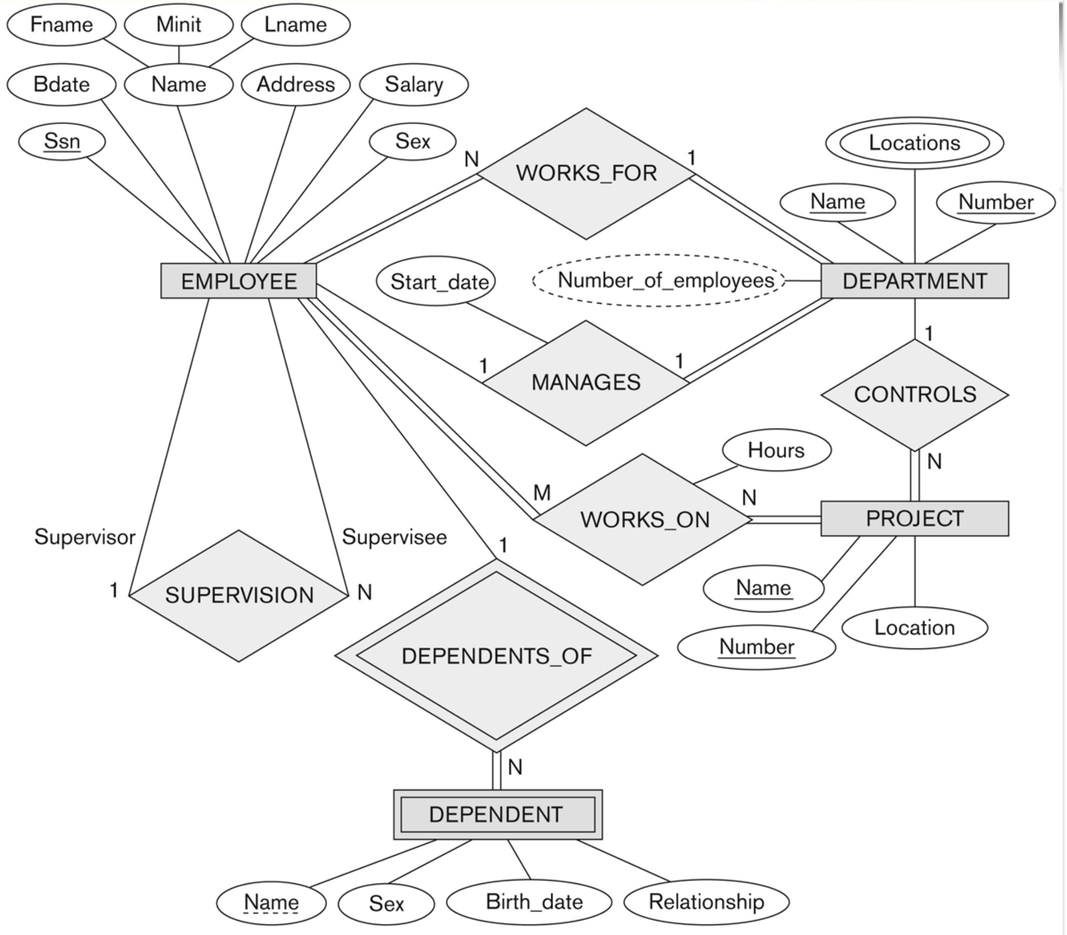


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N

- Shown by placing appropriate numbers on the relationship edges.

Participation constraint (on each participating entity type): total (called existence dependency) or partial.

- Total shown by double line, partial by single line.

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

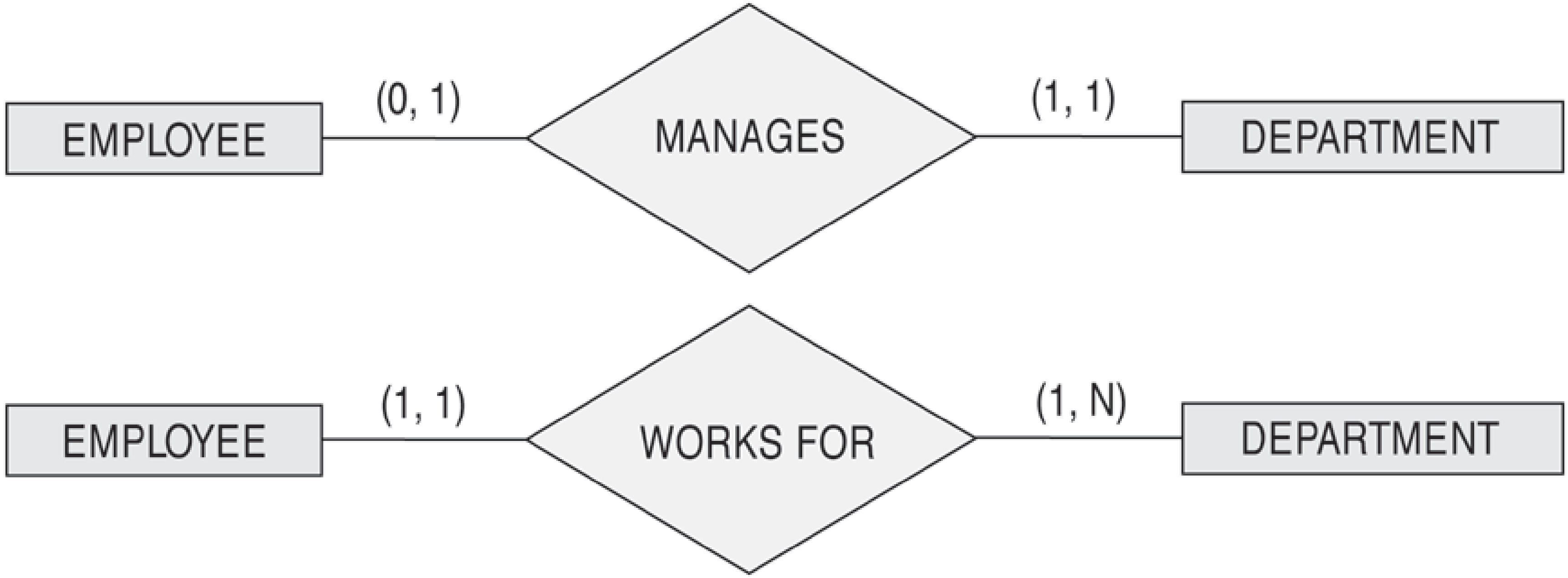
Derived from the knowledge of mini-world constraints

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



COMPANY ER Schema Diagram using (min, max) notation

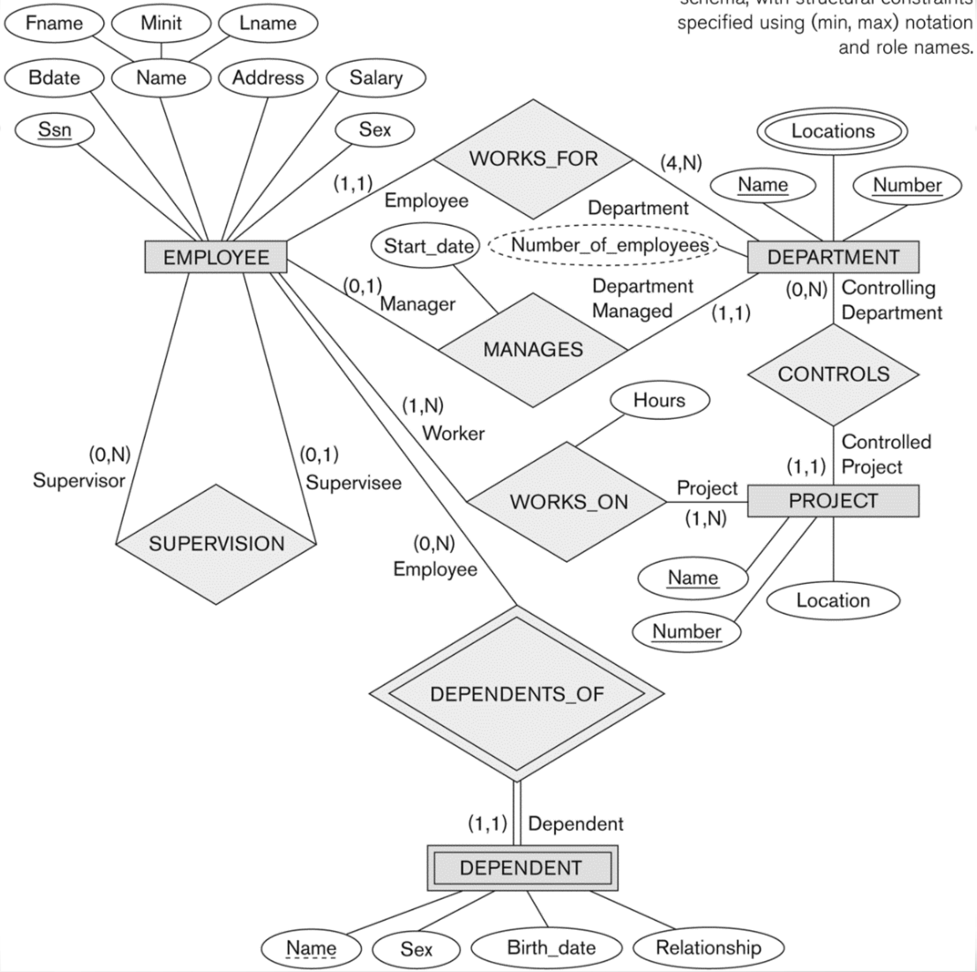


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Summary of notation for ER diagrams

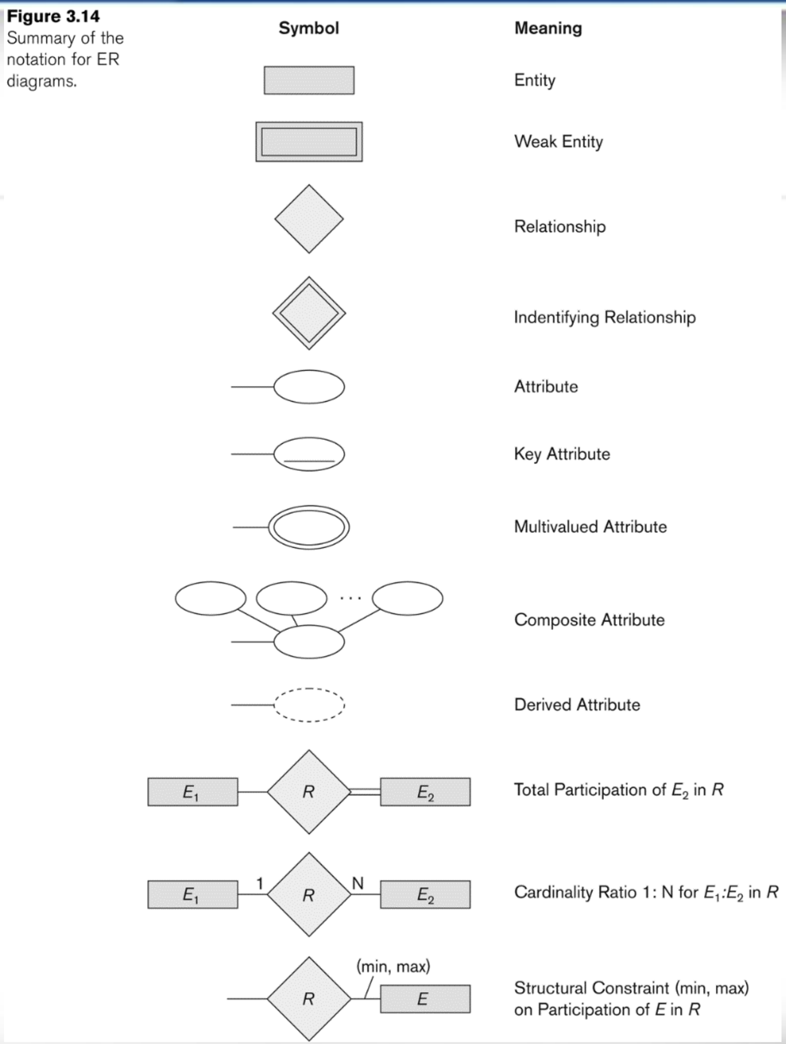


Figure 3.14
Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
. . .	Composite Attribute
	Derived Attribute
E_1 R E_2	Total Participation of E_2 in R
E_1 1 R N E_2	Cardinality Ratio 1: N for E_1:E_2 in R
R (min, max) E	Structural Constraint (min, max) on Participation of E in R

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

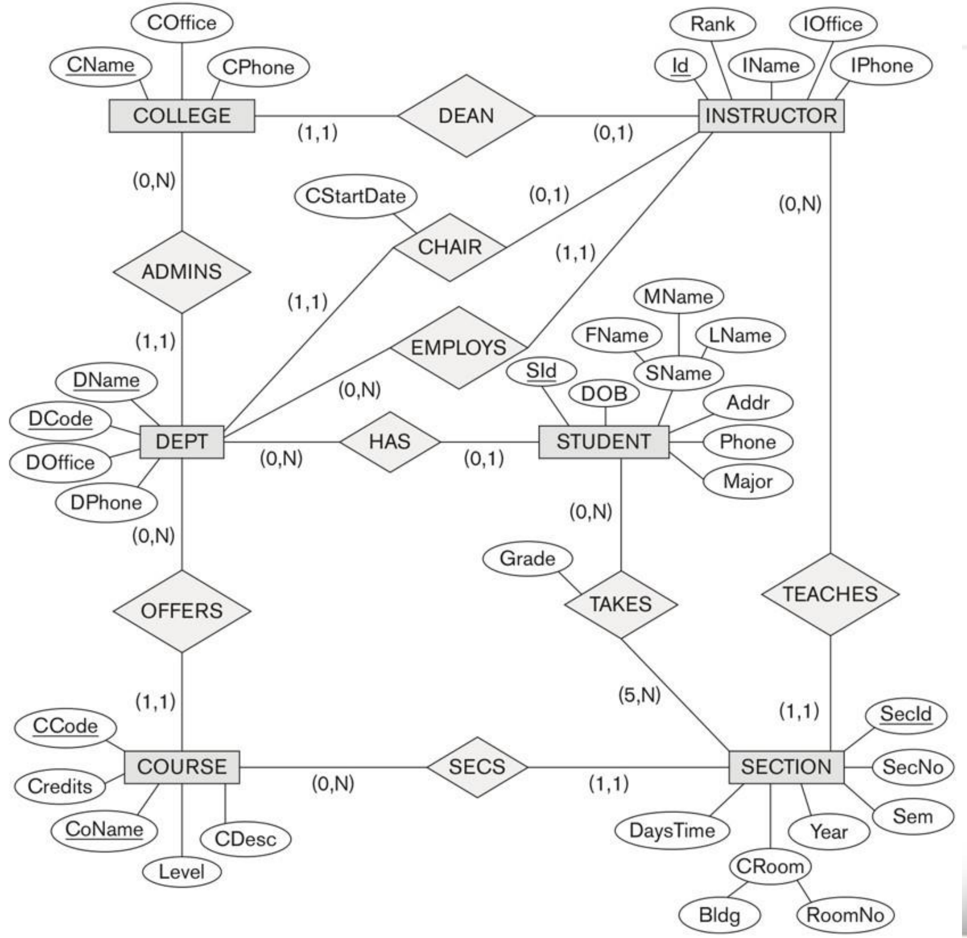
In general, an n-ary relationship is not equivalent to n binary relationships

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

Another Example: A UNIVERSITY Database

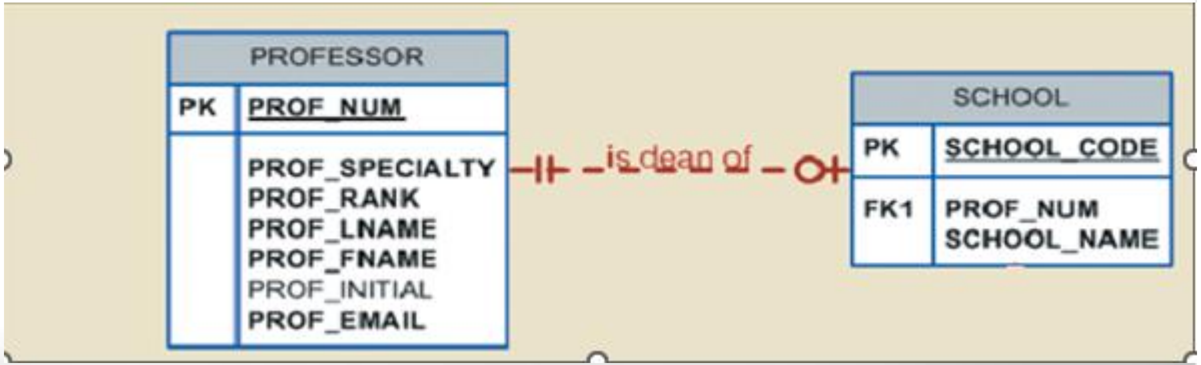
- To keep track of the enrollments in classes and student grades, another database is to be designed.
- It keeps track of the COLLEGEs, DEPARTMENTs within each college, the COURSEs offered by departments, and SECTIONs of courses, INSTRUCTORs who teach the sections etc.
- These entity types and the relationships among these entity types are shown on the next slide in Figure 3.20.

UNIVERSITY database conceptual schema



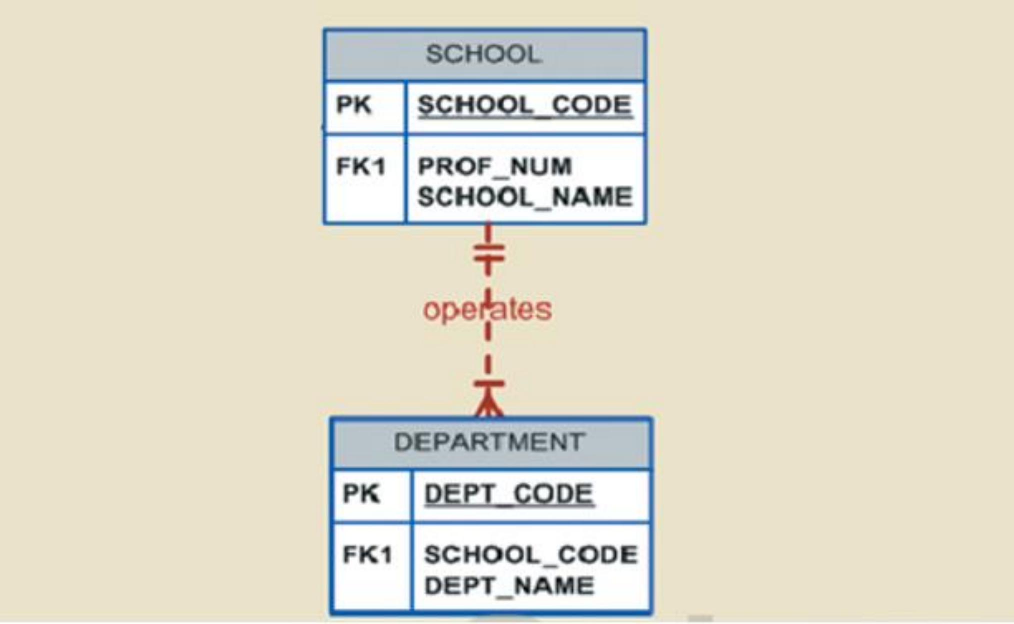
The First Tiny College ERD Segment

1. Each school is administered by a dean who is a professor.
Each professor can be the dean of only one school, and a professor is not required to be
the dean of any school.



The First Tiny College ERD Segment

2. Each school comprises several departments. For example, the school of business has an accounting department, a management/marketing department, an economics/finance department, and a computer information systems department.

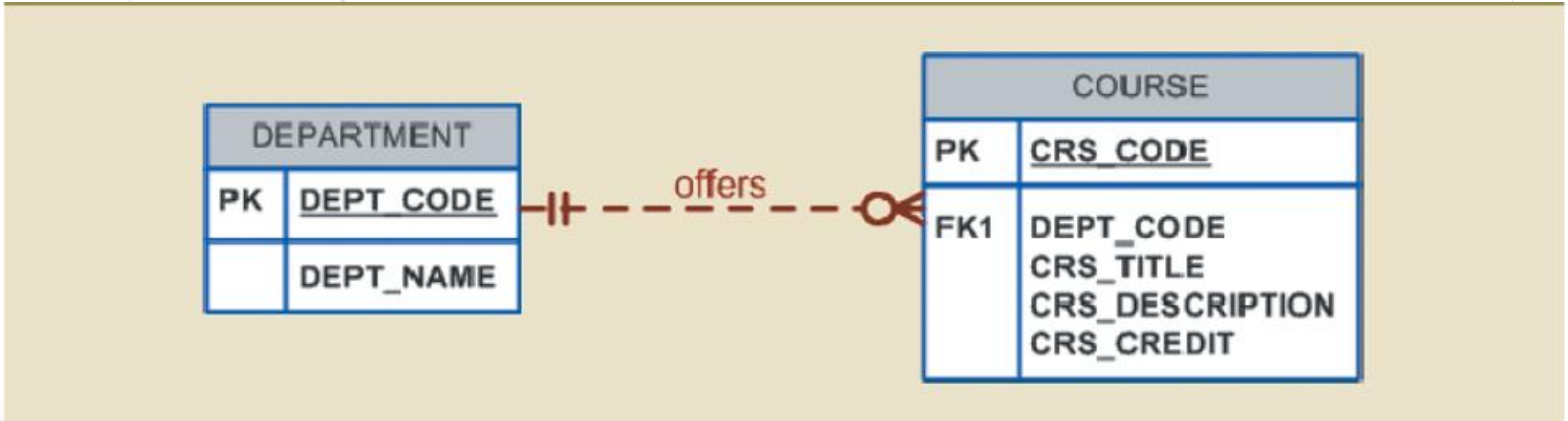


SCHOOL	
PK	SCHOOL_CODE
FK1	PROF_NUM SCHOOL_NAME

DEPARTMENT	
PK	DEPT_CODE
FK1	SCHOOL_CODE DEPT_NAME

The Second Tiny College ERD Segment

3. Each department may offer courses.



DEPARTMENT

PK DEPT_CODE

DEPT_NAME

offers

COURSE

PK CRS_CODE

FK1 DEPT_CODE

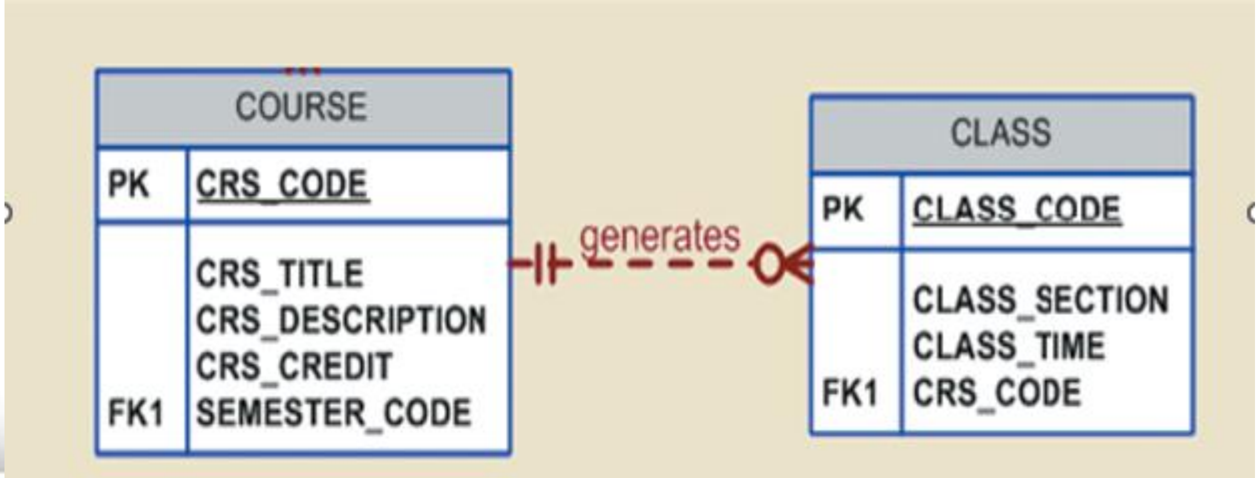
CRS_TITLE

CRS_DESCRIPTION

CRS_CREDIT

The Third Tiny College ERD Segment

4. That is, a department may offer several sections (classes) of the same database course. Each of those classes is taught by a professor at a given time in a given place.

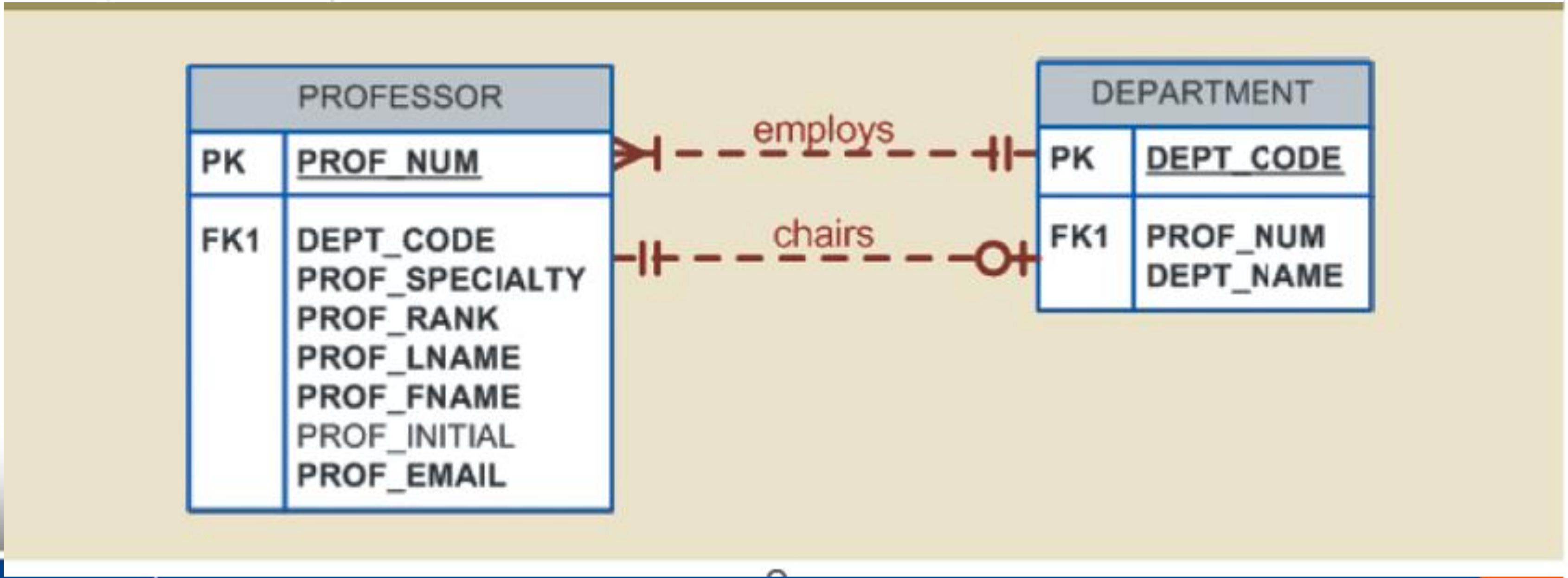


COURSE	
PK	CRS_CODE
	CRS_TITLE CRS_DESCRIPTION CRS_CREDIT
FK1	SEMESTER_CODE

CLASS	
PK	CLASS_CODE
	CLASS_SECTION CLASS_TIME
FK1	CRS_CODE

The Fourth Tiny College ERD Segment

5. Each department should have one or more professors assigned to it. One and only one of those professors chairs the department, and no professor is required to accept the chair position. Therefore, DEPARTMENT is optional to PROFESSOR in the “chairs” relationship.

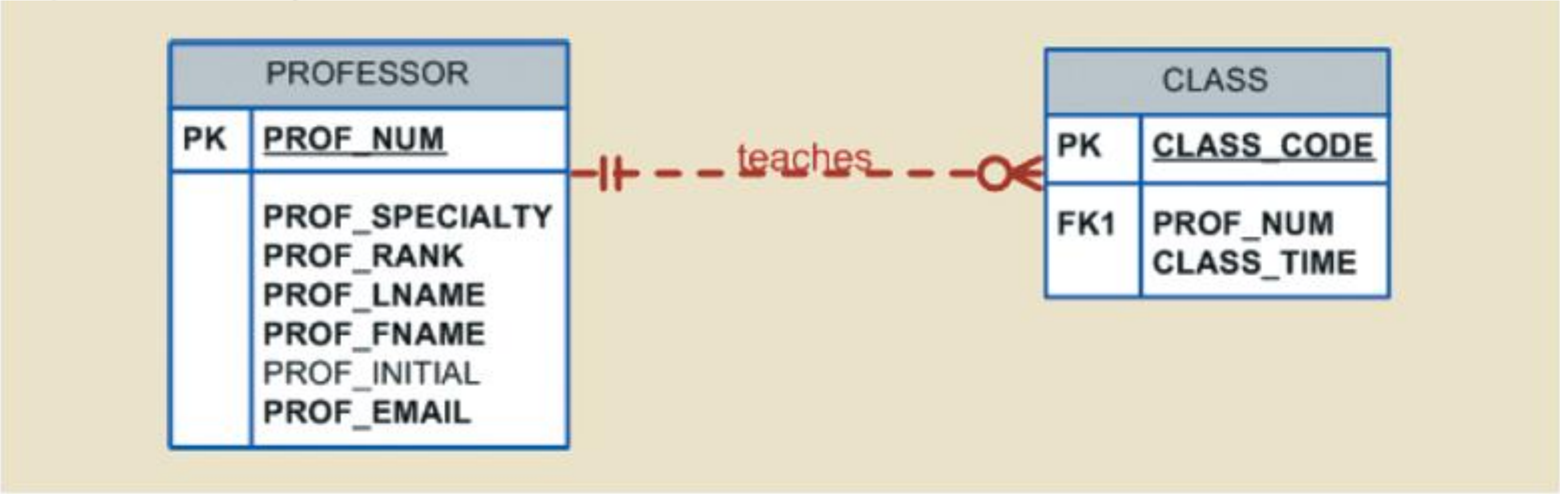


PROFESSOR	
PK	PROF_NUM
FK1	DEPT_CODE PROF_SPECIALTY PROF_RANK PROF_LNAME PROF_FNAME PROF_INITIAL PROF_EMAIL

DEPARTMENT	
PK	DEPT_CODE
FK1	PROF_NUM DEPT_NAME

The Fifth Tiny College ERD Segment

6. Each professor may teach up to four classes; each class is a section of a course. A professor may also be on a research contract and teach no classes at all.

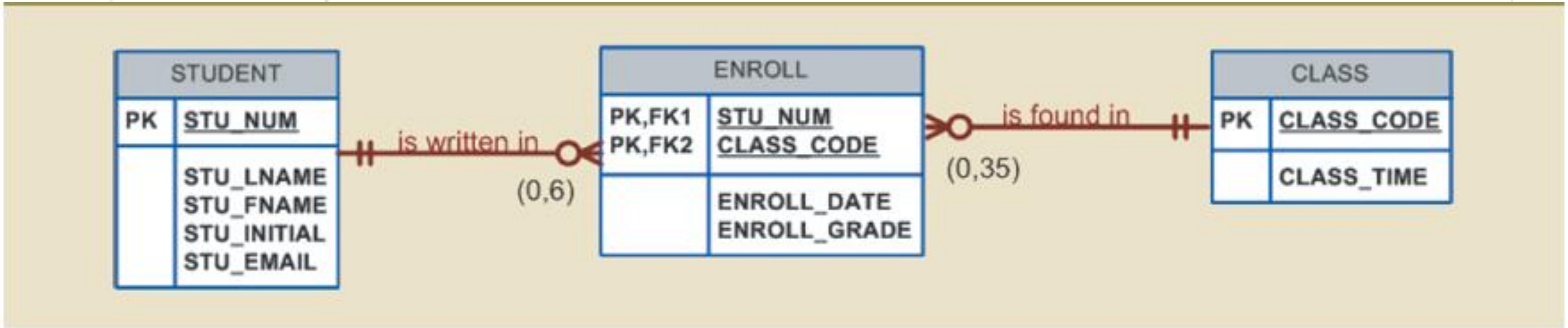


	PROFESSOR
PK	PROF_NUM
	PROF_SPECIALTY PROF_RANK PROF_LNAME PROF_FNAME PROF_INITIAL PROF_EMAIL

	CLASS
PK	CLASS_CODE
FK1	PROF_NUM CLASS_TIME

The Sixth Tiny College ERD Segment

7. A student may enroll in several classes but take each class only once during any given enrollment period.



STUDENT

PK	STU_NUM
	STU_LNAME STU_FNAME STU_INITIAL STU_EMAIL

is written in

(0,6)

ENROLL

PK,FK1 PK,FK2	STU_NUM CLASS_CODE
	ENROLL_DATE ENROLL_GRADE

is found in

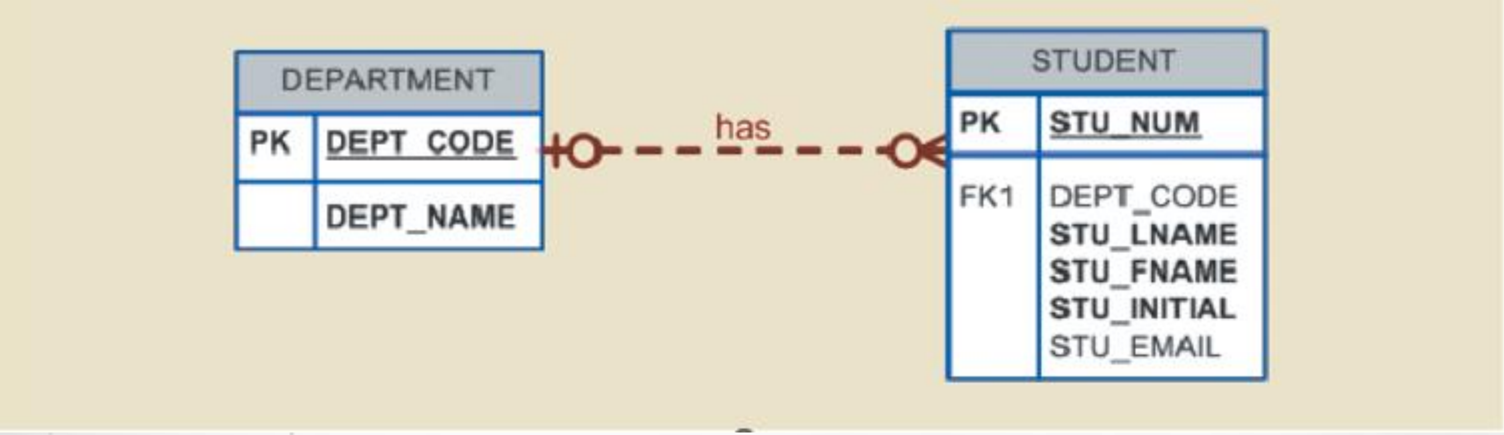
(0,35)

CLASS

PK	CLASS_CODE
	CLASS_TIME

The Seventh Tiny College ERD Segment

8. Each department has several (or many) students whose major is offered by that department. However, each student has only a single major and is therefore associated with a single department.

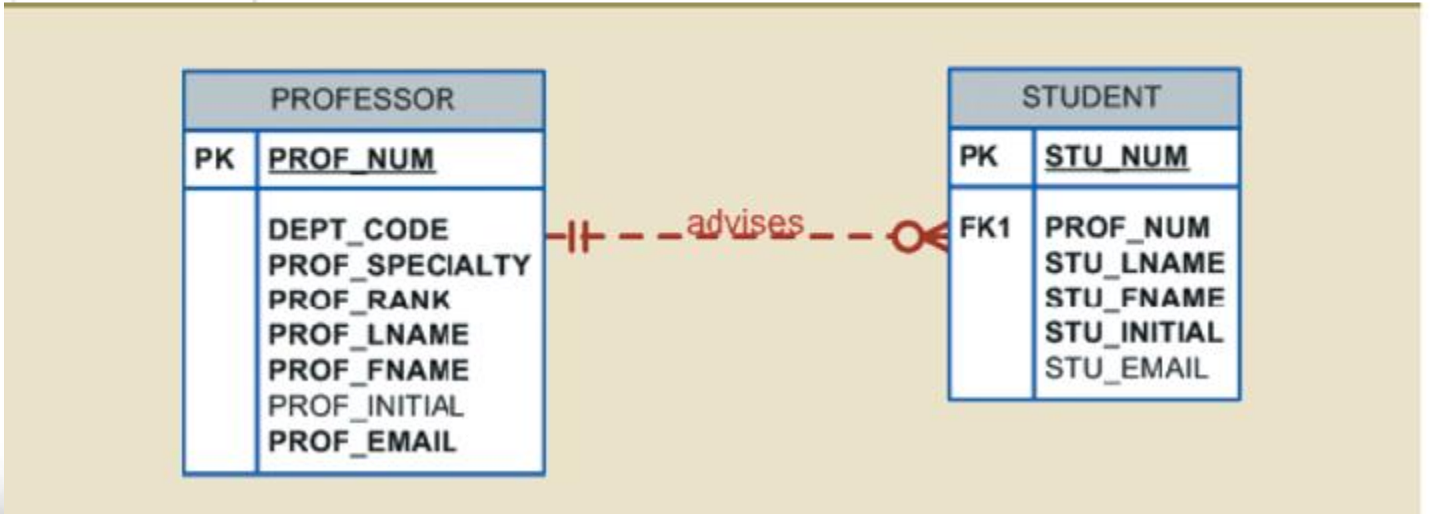


DEPARTMENT	
PK	DEPT_CODE
	DEPT_NAME

STUDENT	
PK	STU_NUM
FK1	DEPT_CODE STU_LNAME STU_FNAME STU_INITIAL STU_EMAIL

The Eighth Tiny College ERD Segment

9. Each student has an advisor in his or her department; each advisor counsels several students. An advisor is also a professor, but not all professors advise students. Therefore, STUDENT is optional to PROFESSOR in the “PROFESSOR advises STUDENT” relationship.

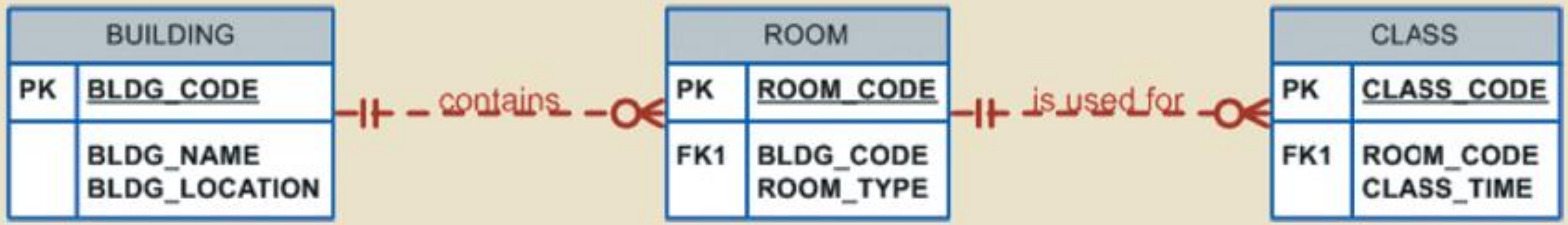


	PROFESSOR
PK	PROF_NUM
	DEPT_CODE PROF_SPECIALTY PROF_RANK PROF_LNAME PROF_FNAME PROF_INITIAL PROF_EMAIL

	STUDENT
PK	STU_NUM
FK1	PROF_NUM STU_LNAME STU_FNAME STU_INITIAL STU_EMAIL

The Ninth Tiny College ERD Segment

10. As you can see in Figure 4.34, the CLASS entity contains a ROOM_CODE attribute. Given the naming conventions, it is clear that ROOM_CODE is an FK to another entity. Clearly, because a class is taught in a room, it is reasonable to assume that the ROOM_CODE in CLASS is the FK to an entity named ROOM. In turn, each room is located in a building. So, the last Tiny College ERD is created by observing that a BUILDING can contain many ROOMs, but each ROOM is found in a single BUILDING. In this ERD segment, it is clear that some buildings do not contain (class) rooms.



Components of the ERM

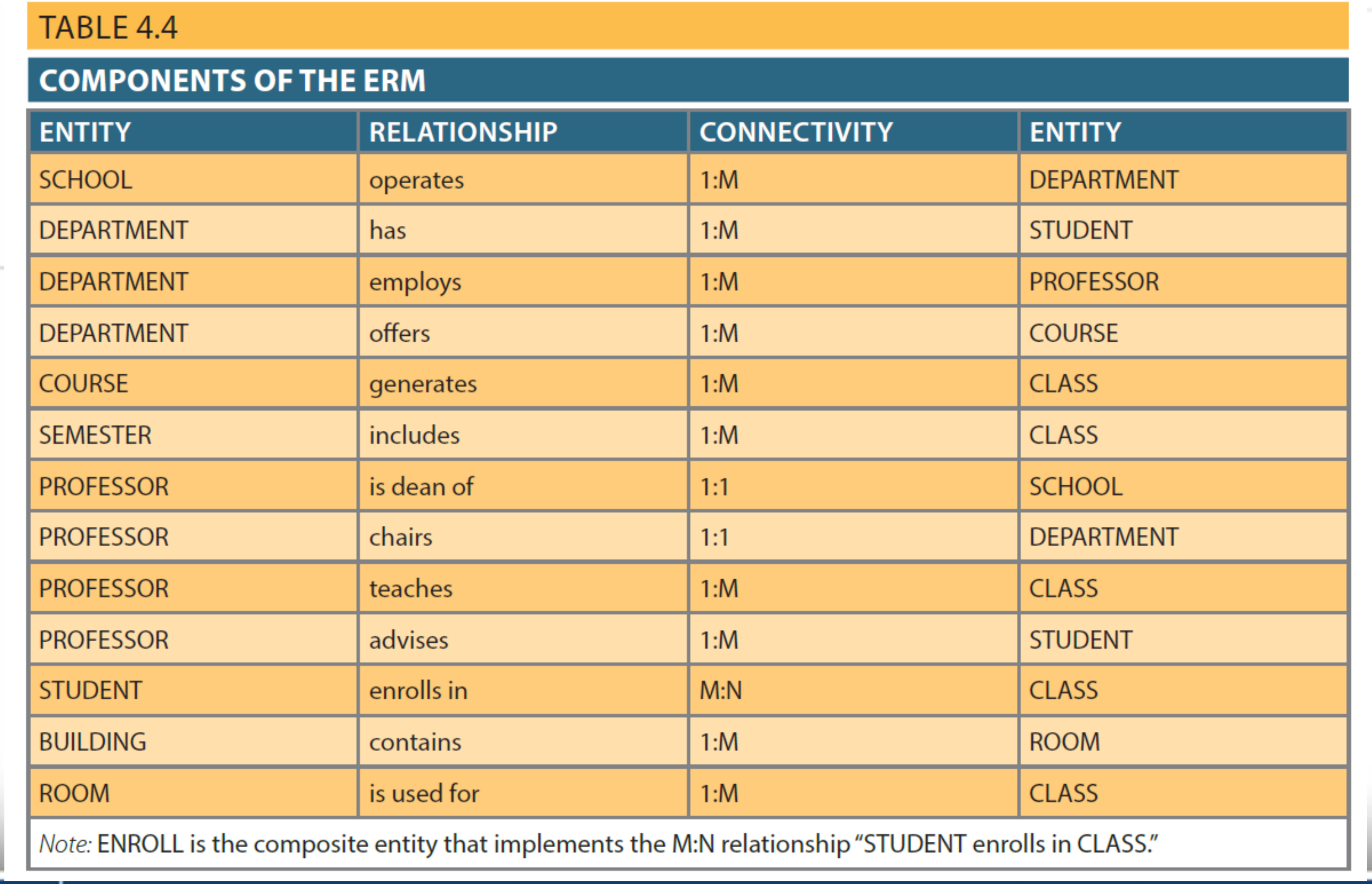
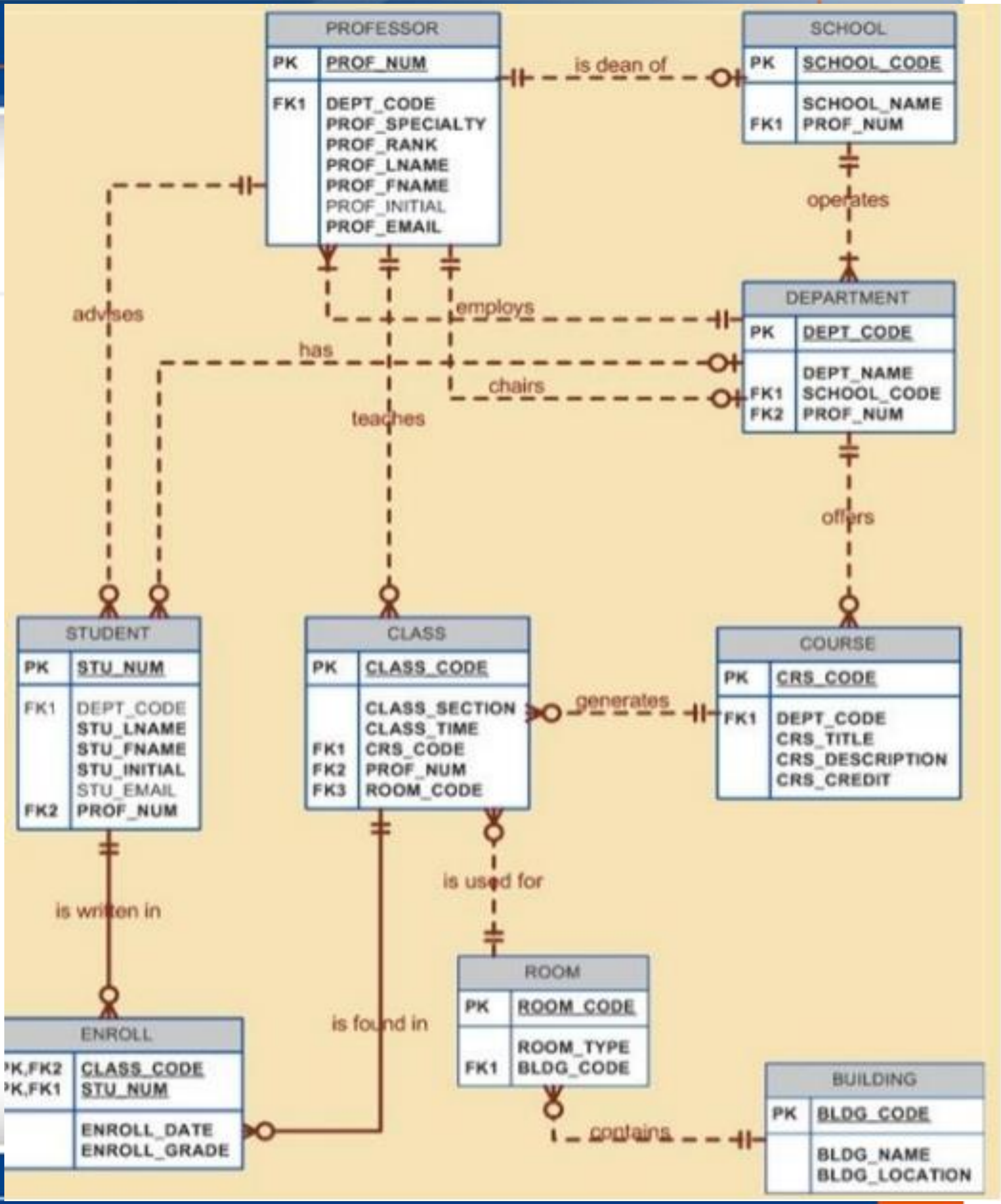


TABLE 4.4

COMPONENTS OF THE ERM

ENTITY	RELATIONSHIP	CONNECTIVITY	ENTITY
SCHOOL	operates	1:M	DEPARTMENT
DEPARTMENT	has	1:M	STUDENT
DEPARTMENT	employs	1:M	PROFESSOR
DEPARTMENT	offers	1:M	COURSE
COURSE	generates	1:M	CLASS
SEMESTER	includes	1:M	CLASS
PROFESSOR	is dean of	1:1	SCHOOL
PROFESSOR	chairs	1:1	DEPARTMENT
PROFESSOR	teaches	1:M	CLASS
PROFESSOR	advises	1:M	STUDENT
STUDENT	enrolls in	M:N	CLASS
BUILDING	contains	1:M	ROOM
ROOM	is used for	1:M	CLASS

Note: ENROLL is the composite entity that implements the M:N relationship "STUDENT enrolls in CLASS."



PROFESSOR	
PK	PROF_NUM
FK1	DEPT_CODE PROF_SPECIALTY PROF_RANK PROF_LNAME PROF_FNAME PROF_INITIAL PROF_EMAIL

SCHOOL	
PK	SCHOOL_CODE
	SCHOOL_NAME
FK1	PROF_NUM

DEPARTMENT	
PK	DEPT_CODE
	DEPT_NAME
FK1	SCHOOL_CODE
FK2	PROF_NUM

STUDENT	
PK	STU_NUM
FK1	DEPT_CODE STU_LNAME STU_FNAME STU_INITIAL STU_EMAIL
FK2	PROF_NUM

CLASS	
PK	CLASS_CODE
	CLASS_SECTION CLASS_TIME
FK1	CRS_CODE
FK2	PROF_NUM
FK3	ROOM_CODE

COURSE	
PK	CRS_CODE
FK1	DEPT_CODE CRS_TITLE CRS_DESCRIPTION CRS_CREDIT

ENROLL	
PK,FK2	CLASS_CODE
PK,FK1	STU_NUM
	ENROLL_DATE ENROLL_GRADE

ROOM	
PK	ROOM_CODE
	ROOM_TYPE
FK1	BLDG_CODE

BUILDING	
PK	BLDG_CODE
	BLDG_NAME BLDG_LOCATION