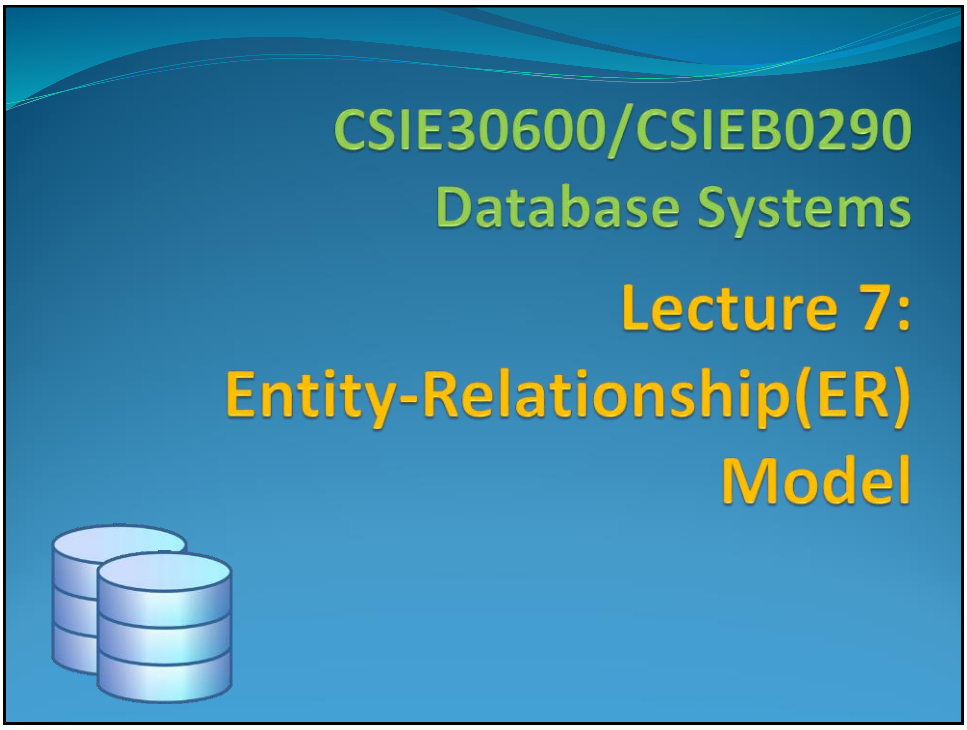
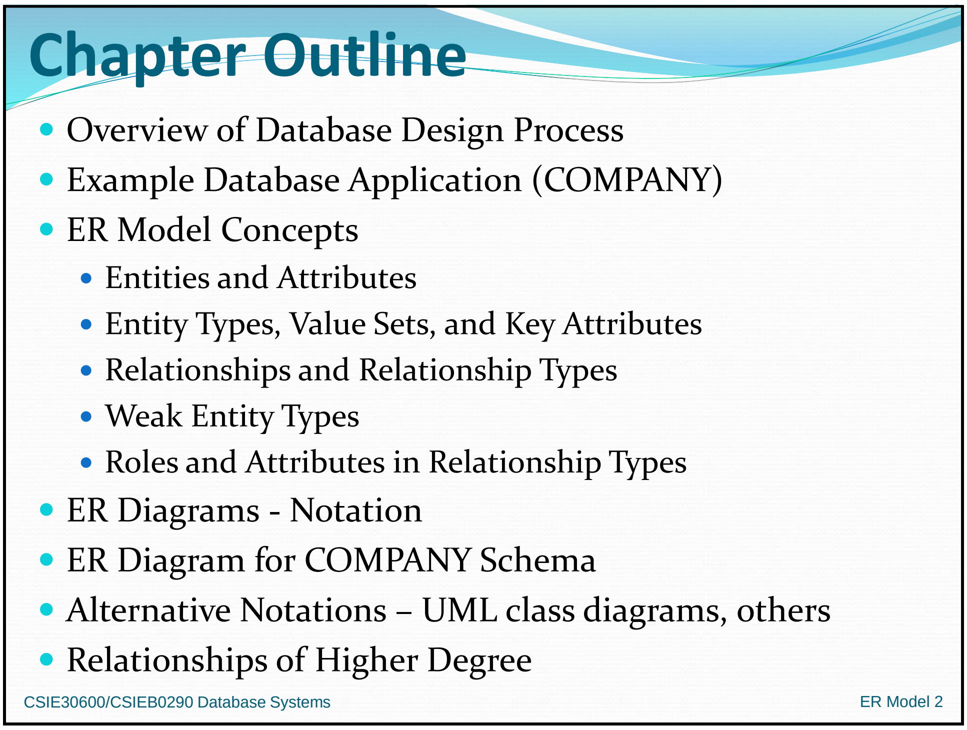


CSIE30600/CSIEB0290 Database Systems

Lecture 7: Entity-Relationship(ER) Model



Chapter Outline

- Overview of Database Design Process
- Example Database Application (COMPANY)
- ER Model Concepts
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram for COMPANY Schema
- Alternative Notations – UML class diagrams, others
- Relationships of Higher Degree

Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus in this chapter on conceptual database design
  - To design the conceptual schema for a database application
- Applications design focuses on the programs and interfaces that access the database
  - Generally considered part of software engineering

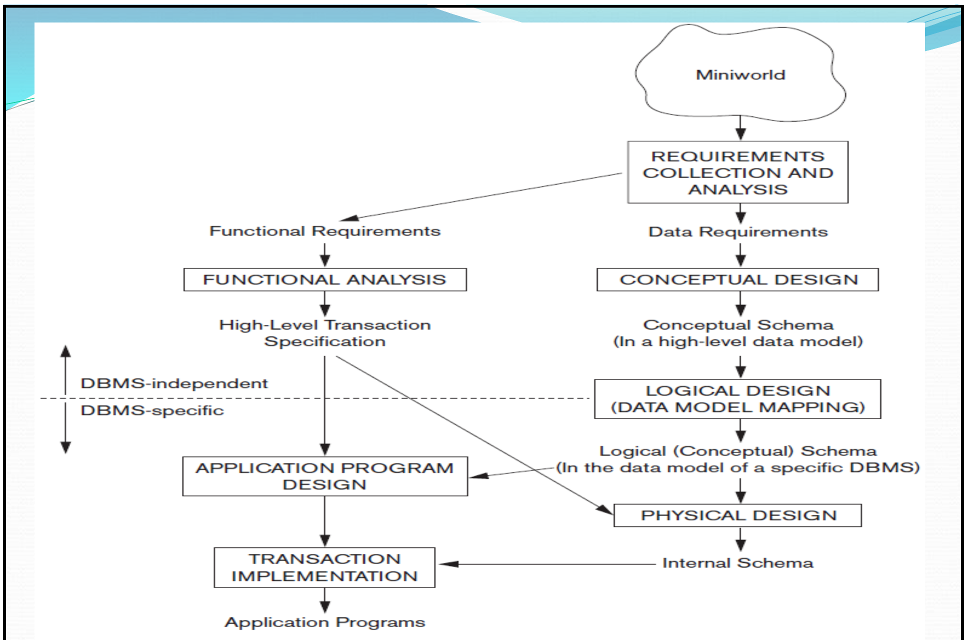
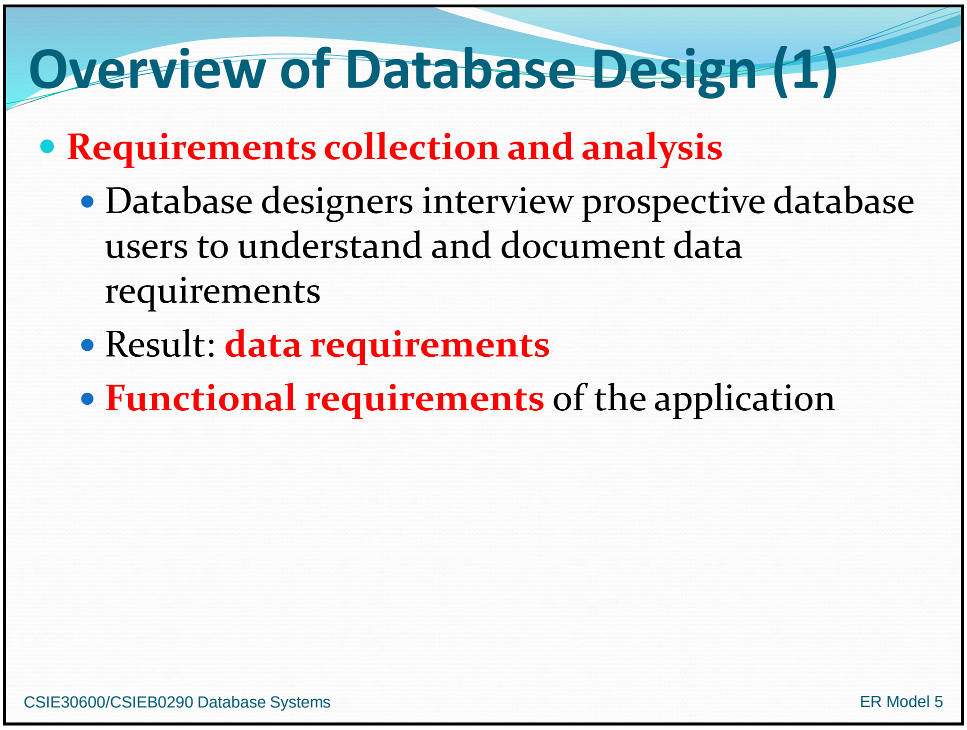
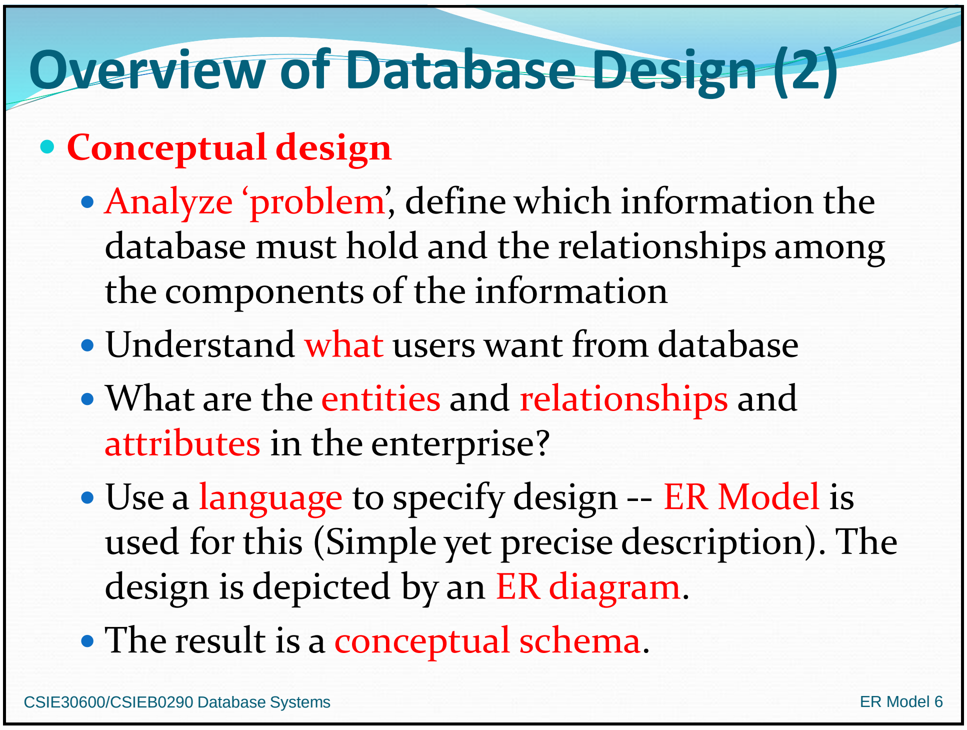

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.



Overview of Database Design (1)

- **Requirements collection and analysis**
  - Database designers interview prospective database users to understand and document data requirements
  - Result: **data requirements**
  - **Functional requirements** of the application



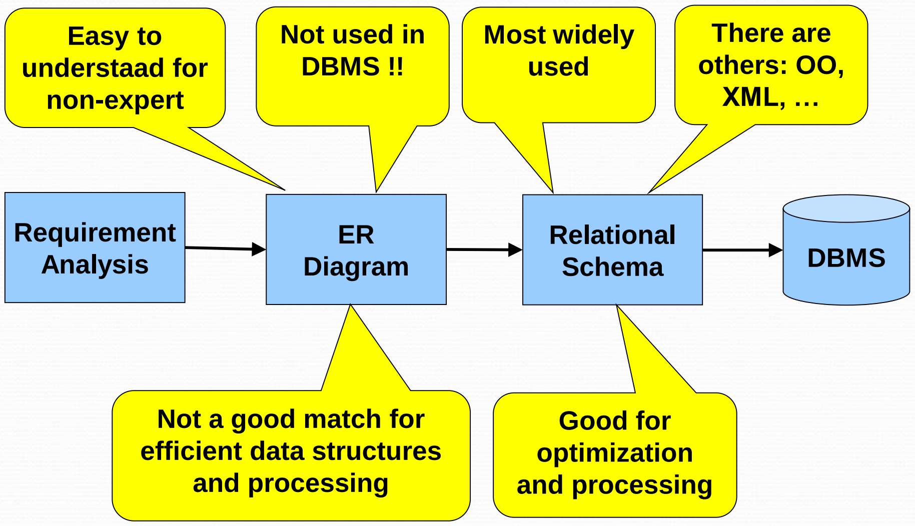
Overview of Database Design (2)

- **Conceptual design**
  - Analyze 'problem', define which information the database must hold and the relationships among the components of the information
  - Understand what users want from database
  - What are the entities and relationships and attributes in the enterprise?
  - Use a language to specify design -- ER Model is used for this (Simple yet precise description). The design is depicted by an ER diagram.
  - The result is a conceptual schema.

Overview of Database Design (3)

- **Logical design** or **data model mapping**
  - ER diagram is converted into a relational schema
  - Check relational schema for redundancies and related anomalies – Normalization
  - Input schema to DBMS
- **Physical database design** and **tuning**
  - Consider typical workloads and further refine the database design.
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

Overview of Database Design (4)



Easy to understaad for non-expert

Not used in DBMS !!

Most widely used

There are others: OO, XML, …

Requirement Analysis

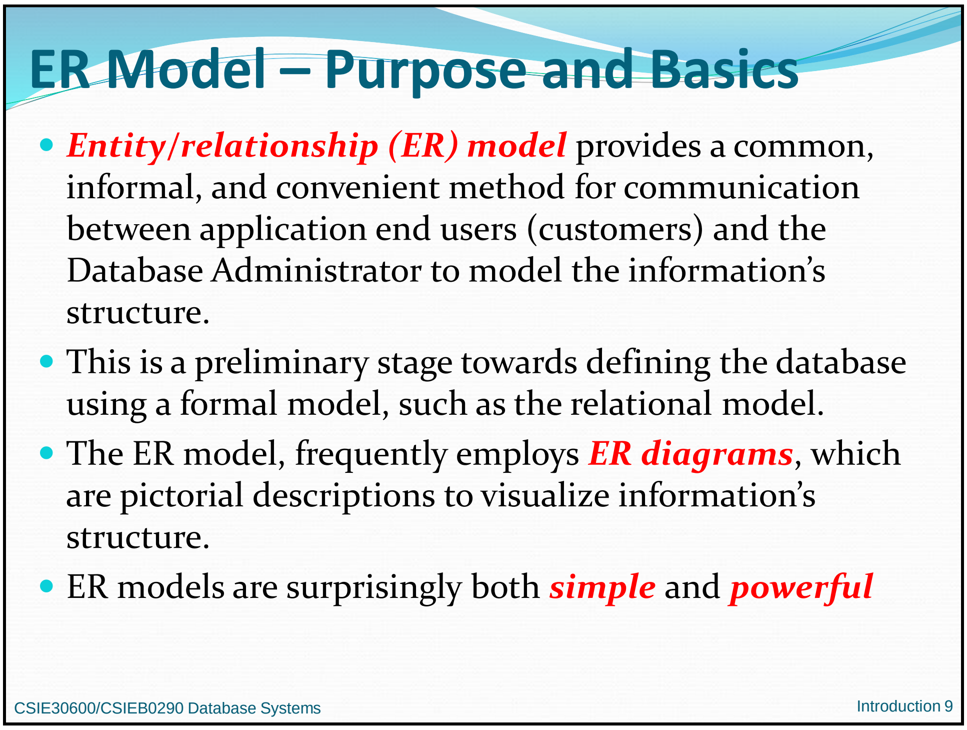
ER Diagram

Relational Schema

DBMS

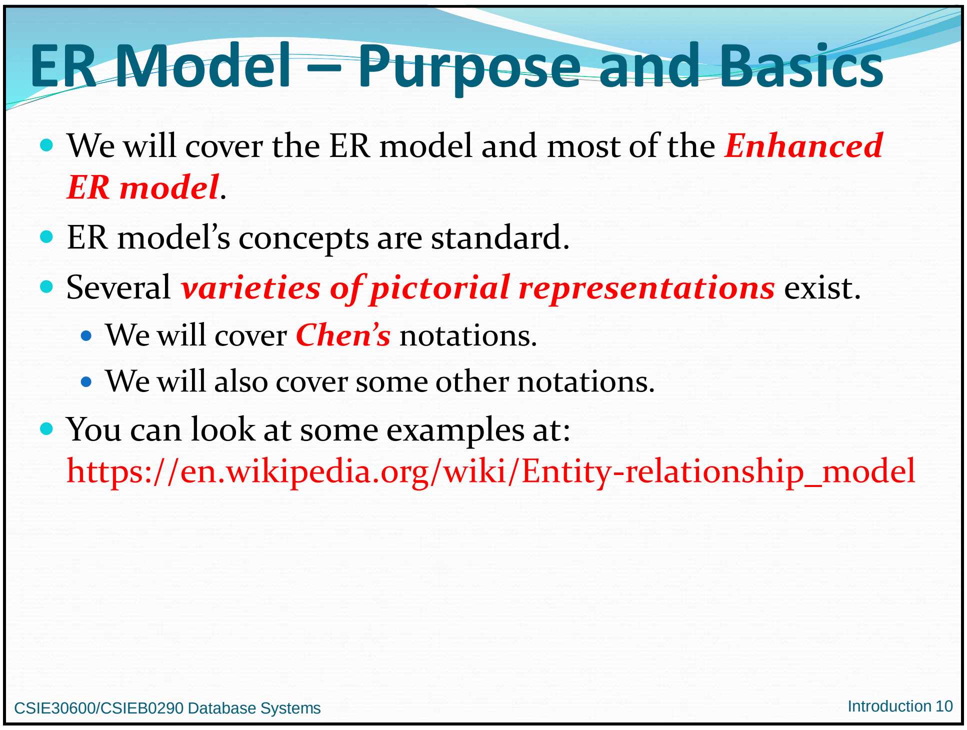
Not a good match for efficient data structures and processing

Good for optimization and processing



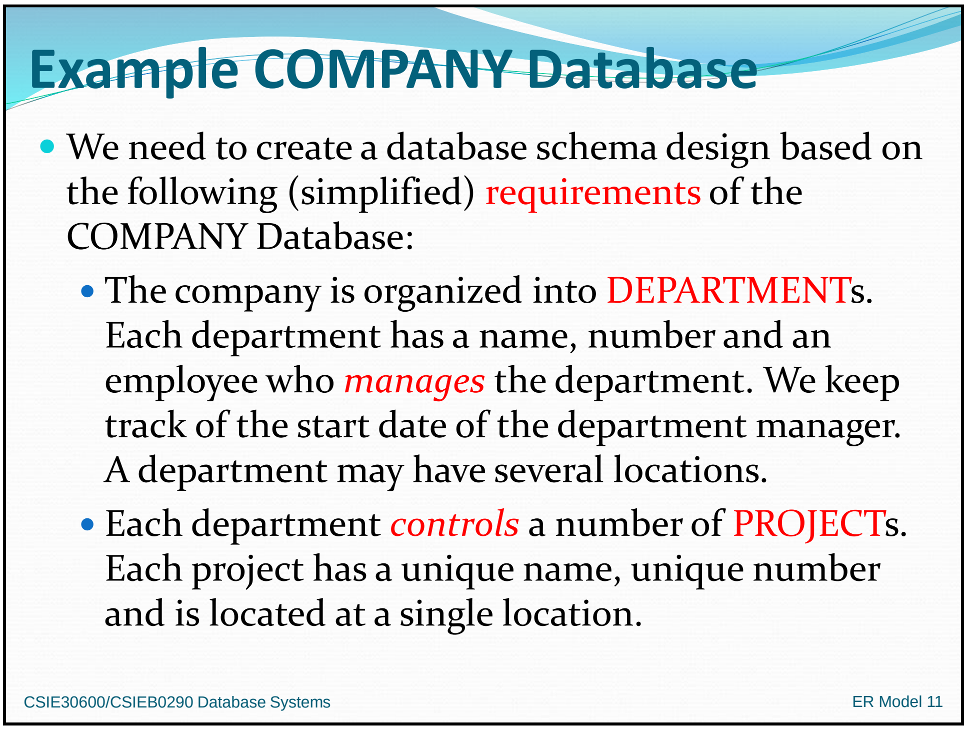
ER Model – Purpose and Basics

- ***Entity/relationship (ER) model*** provides a common, informal, and convenient method for communication between application end users (customers) and the Database Administrator to model the information's structure.
- This is a preliminary stage towards defining the database using a formal model, such as the relational model.
- The ER model, frequently employs ***ER diagrams***, which are pictorial descriptions to visualize information's structure.
- ER models are surprisingly both ***simple*** and ***powerful***



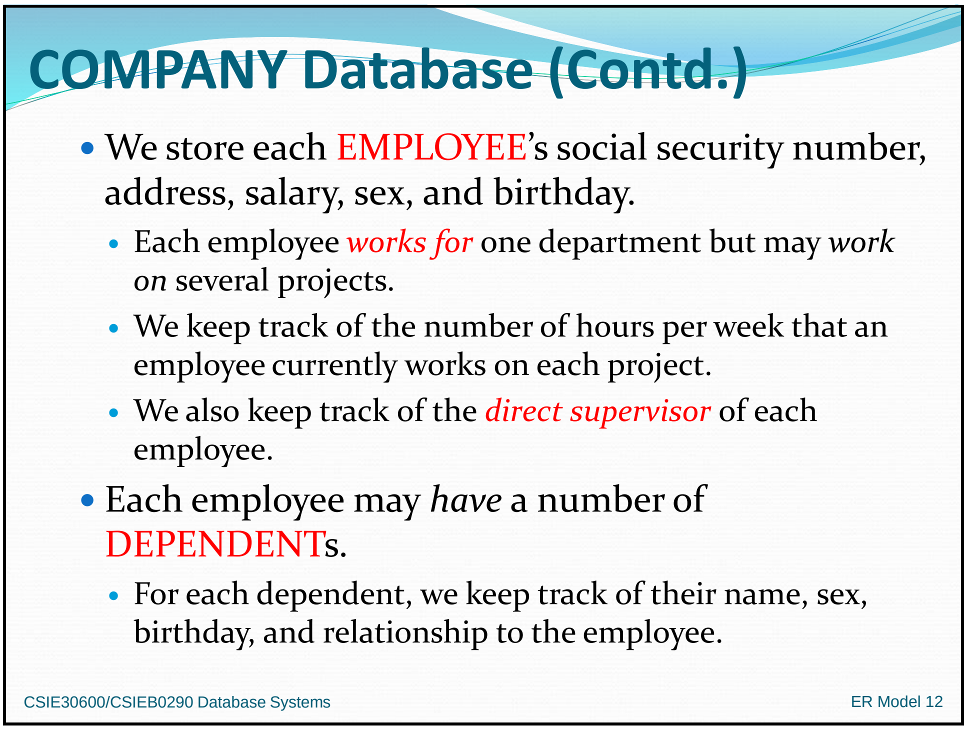
ER Model – Purpose and Basics

- We will cover the ER model and most of the ***Enhanced ER model***.
- ER model's concepts are standard.
- Several ***varieties of pictorial representations*** exist.
  - We will cover ***Chen's*** notations.
  - We will also cover some other notations.
- You can look at some examples at: https://en.wikipedia.org/wiki/Entity-relationship_model



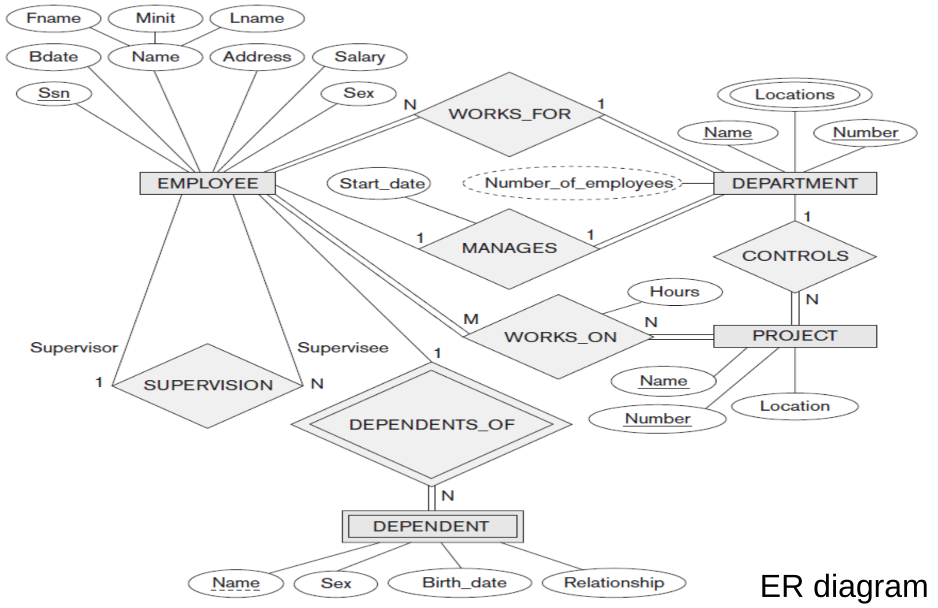
Example COMPANY Database

- 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.



COMPANY Database (Contd.)

- We store each EMPLOYEE's social security number, address, salary, sex, and birthday.
    - Each employee *works for* one department but may *work on* several projects.
    - We keep track of the number of hours per week that an employee currently works on each project.
    - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
    - For each dependent, we keep track of their name, sex, birthday, and relationship to the employee.



ER diagram

Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.

CSIE30600/CSIEB0290 Database Systems ER Model 13

Conceptual Modeling

- A database can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An entity is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have attributes
  - Example: people have *names* and *addresses*

CSIE30600/CSIEB0290 Database Systems ER Model 14

Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate

Entities and Attributes

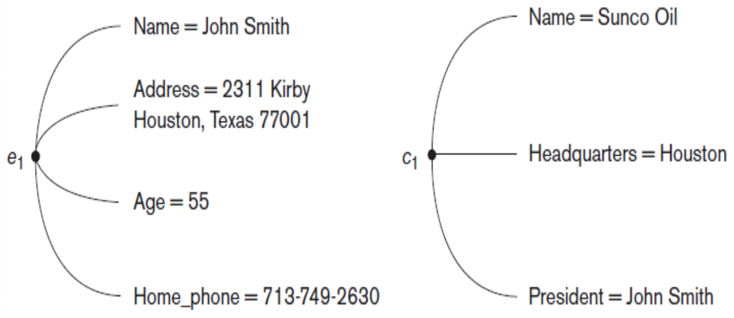
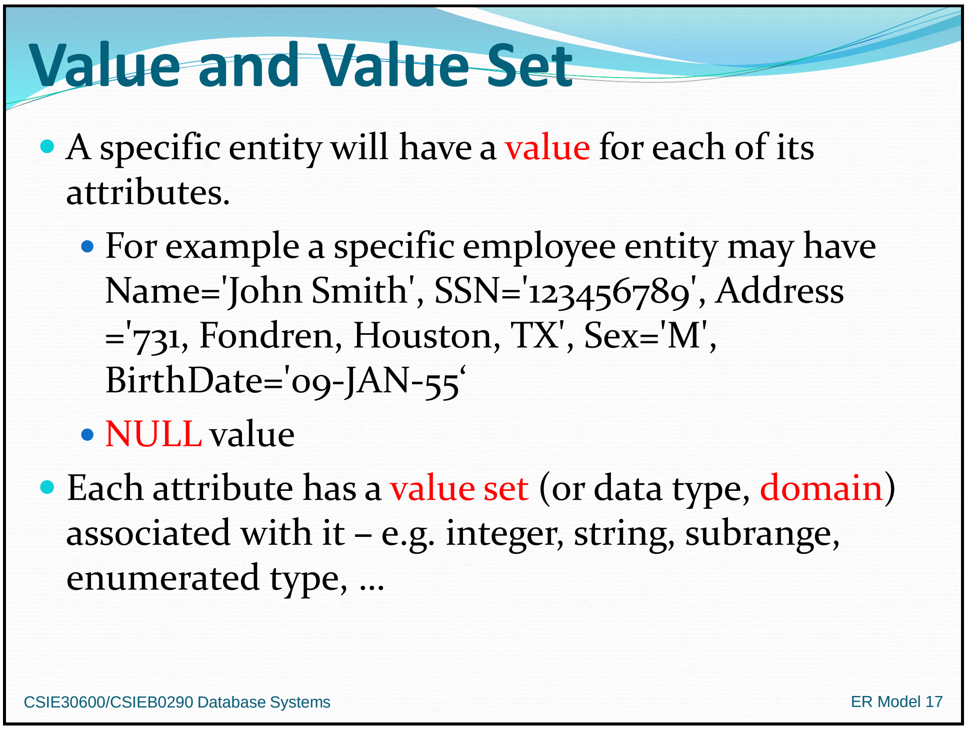
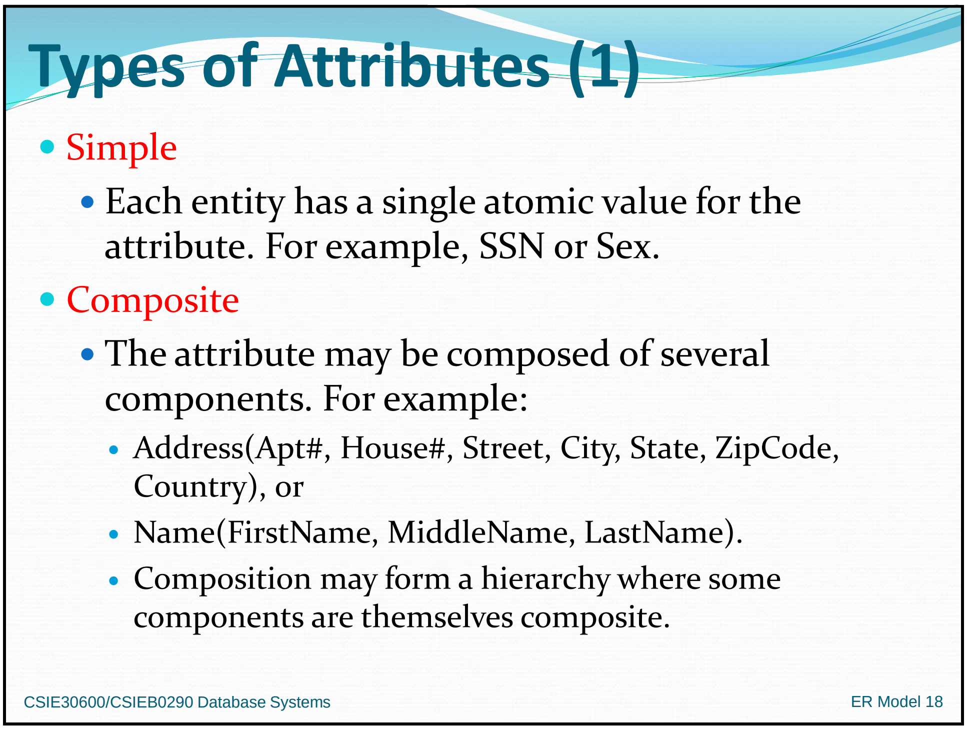


Figure 3.3 Two entities, EMPLOYEE e_1, and COMPANY c_1, and their attributes.



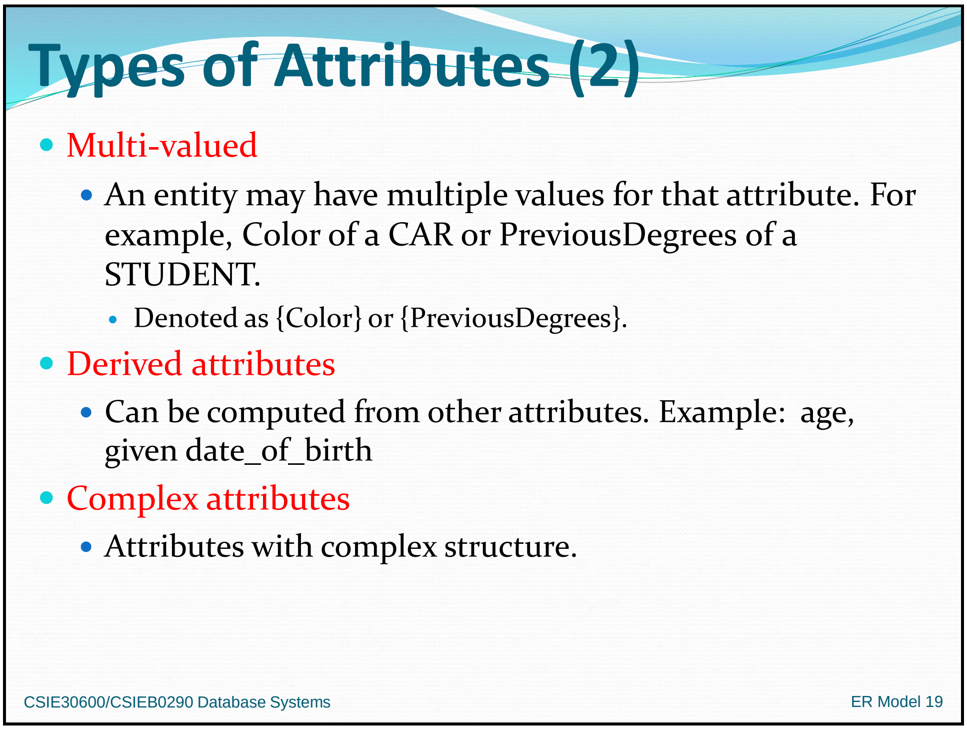
Value and Value Set

- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘
  - NULL value
- Each attribute has a value set (or data type, domain) associated with it – e.g. integer, string, subrange, enumerated type, …



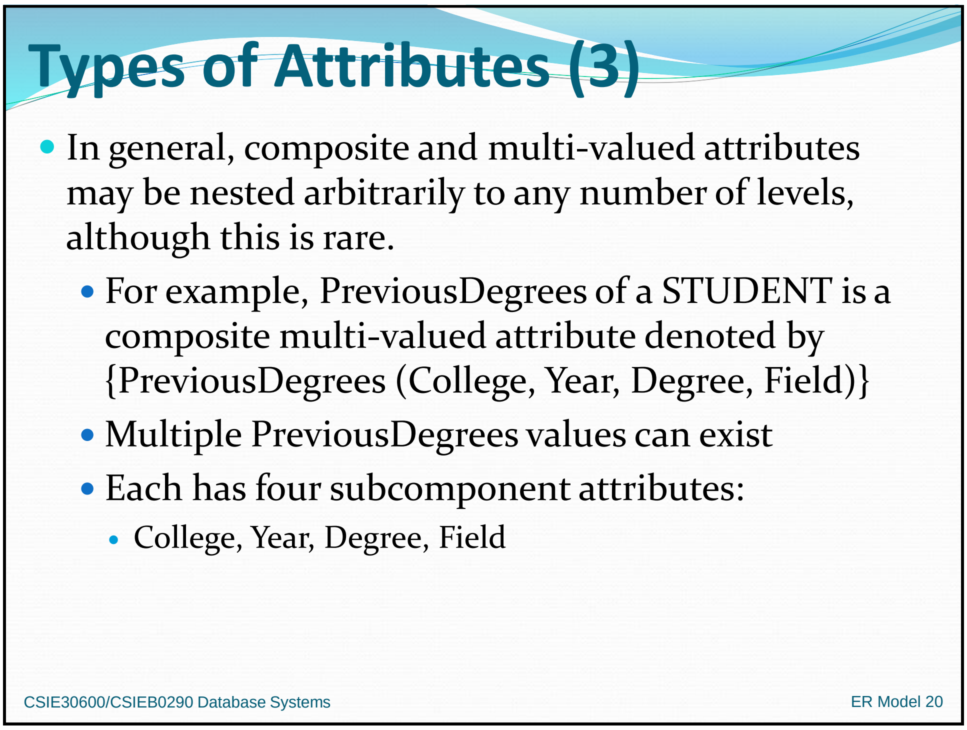
Types of Attributes (1)

- 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.



Types of Attributes (2)

- 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}.
- Derived attributes
  - Can be computed from other attributes. Example: age, given date_of_birth
- Complex attributes
  - Attributes with complex structure.



Types of Attributes (3)

- 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

Examples of Composite Attribute

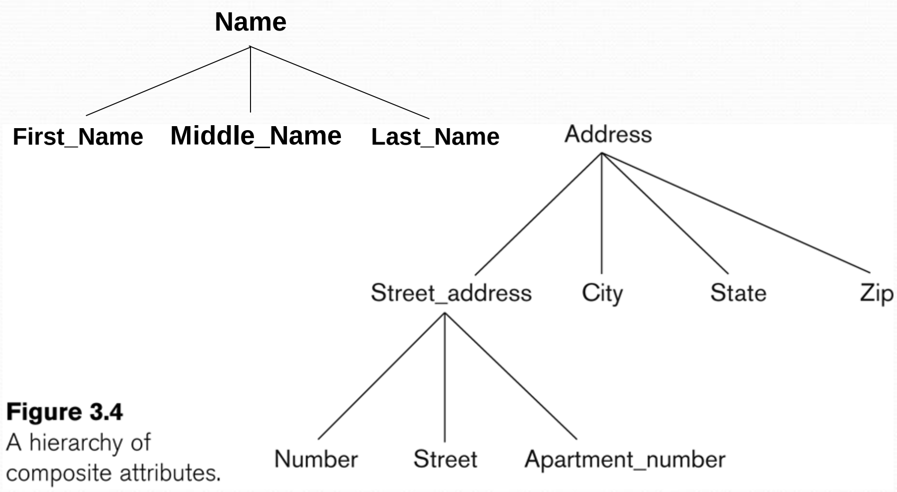
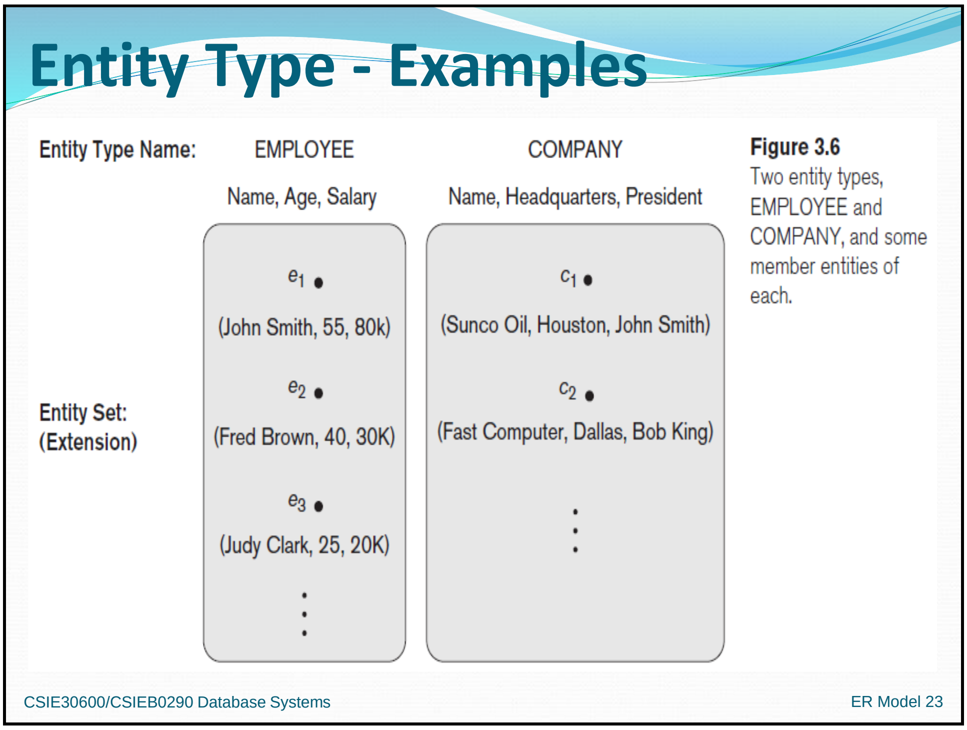


Figure 3.4
A hierarchy of composite attributes.

Entity Types and Key Attributes (1)

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and COMPANY (next slide)
- 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 Type - Examples

Entity Type Name: EMPLOYEE COMPANY

Name, Age, Salary

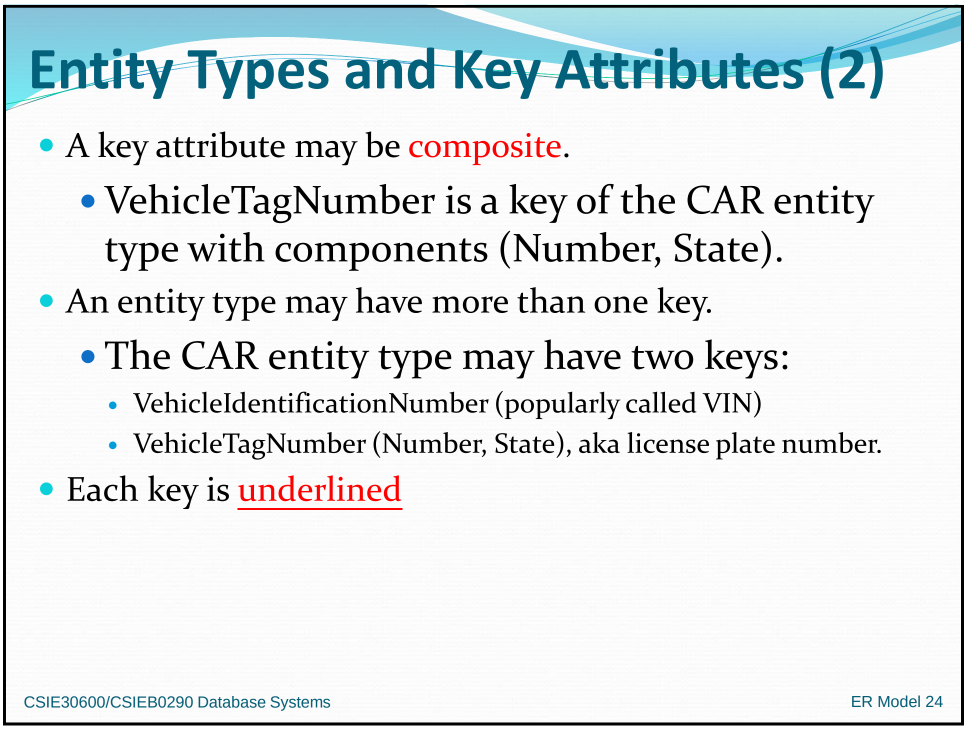
Name, Headquarters, President

Entity Set: (Extension)

e_1 •
(John Smith, 55, 80k)
e_2 •
(Fred Brown, 40, 30K)
e_3 •
(Judy Clark, 25, 20K)
⋮

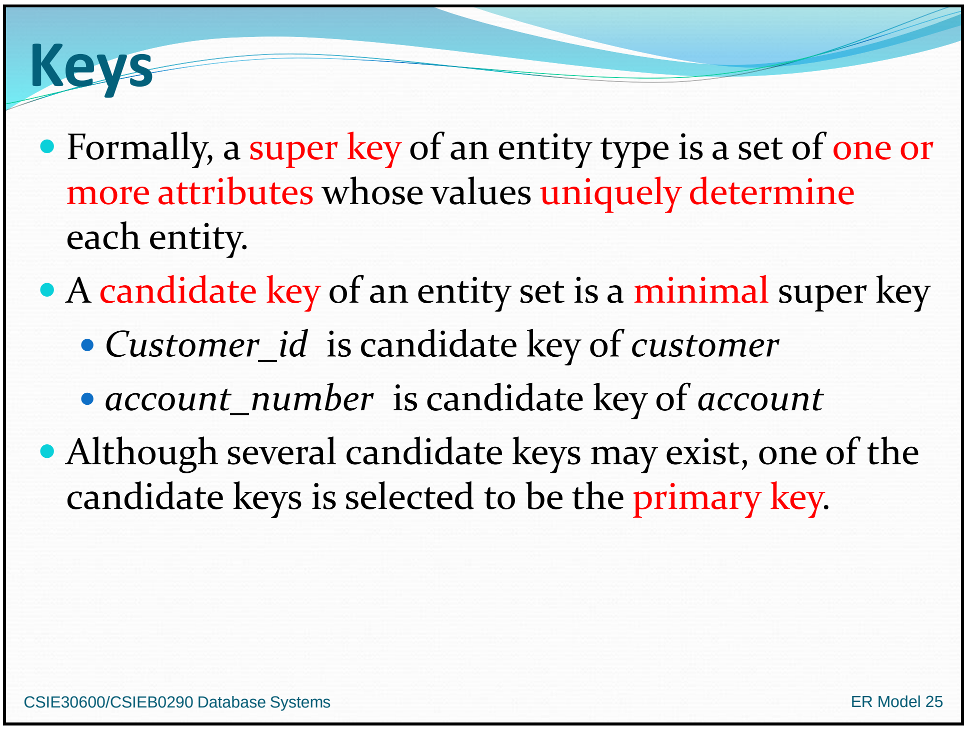
c_1 •
(Sunco Oil, Houston, John Smith)
c_2 •
(Fast Computer, Dallas, Bob King)
⋮

Figure 3.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.



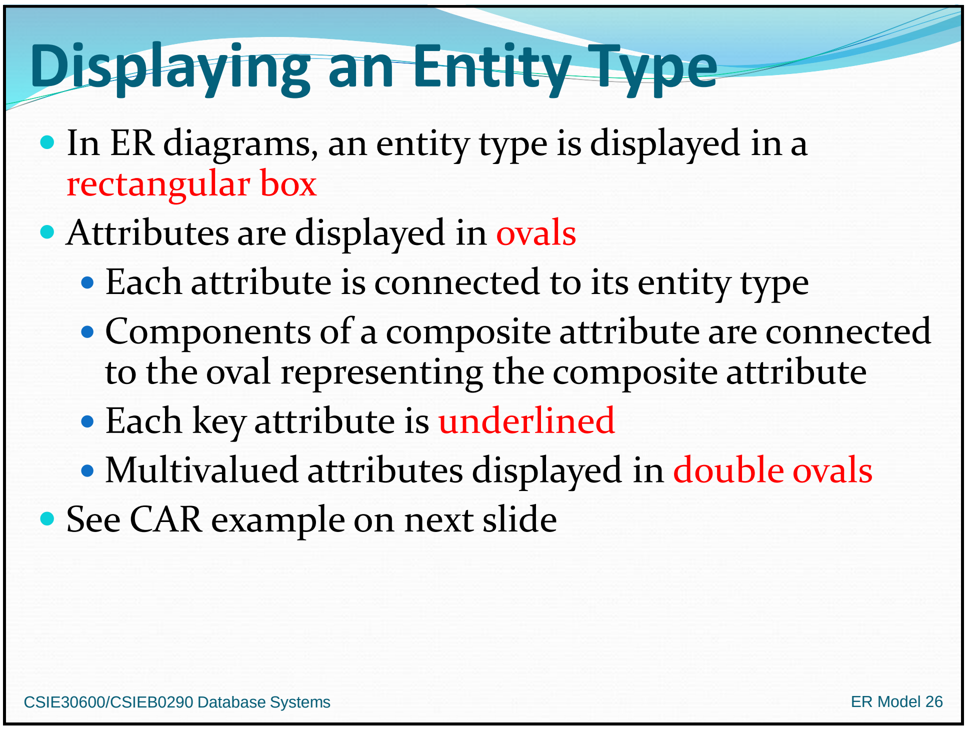
Entity Types and Key Attributes (2)

- 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



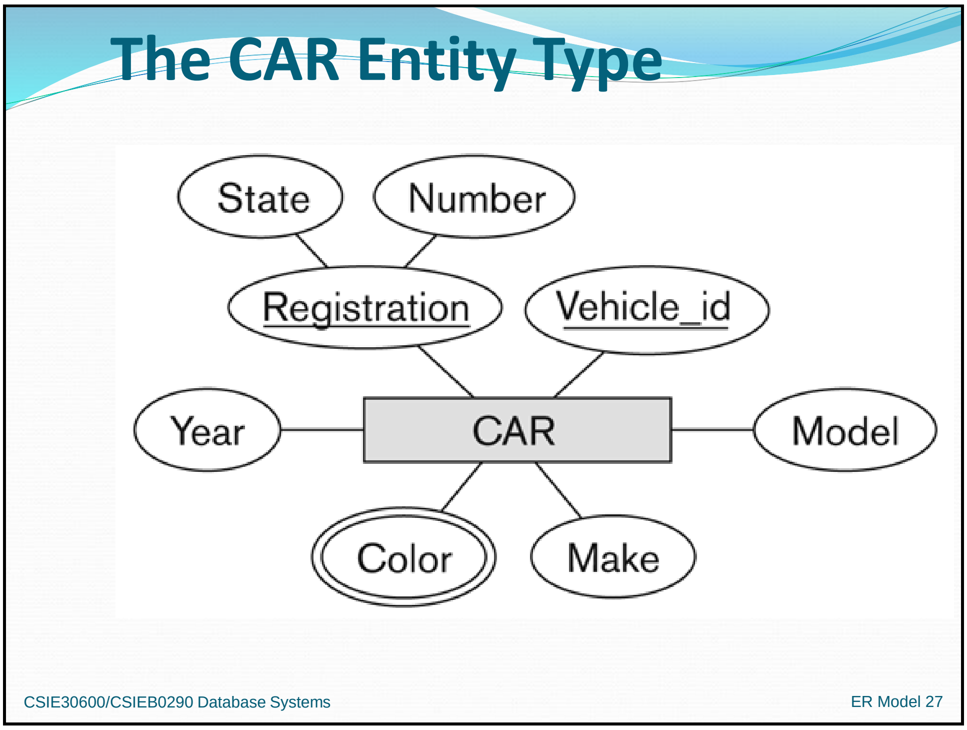
Keys

- Formally, a super key of an entity type is a set of one or more attributes whose values uniquely determine each entity.
- A candidate key of an entity set is a minimal super key
  - *Customer_id* is candidate key of *customer*
  - *account_number* is candidate key of *account*
- Although several candidate keys may exist, one of the candidate keys is selected to be the primary key.

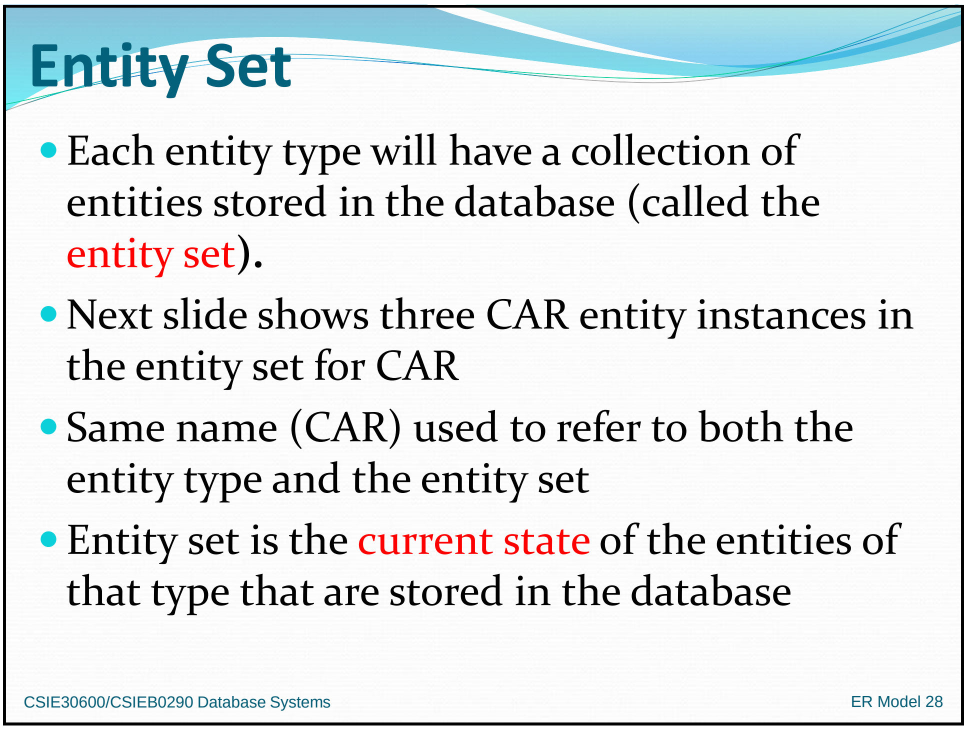


Displaying an Entity Type

- 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
- See CAR example on next slide

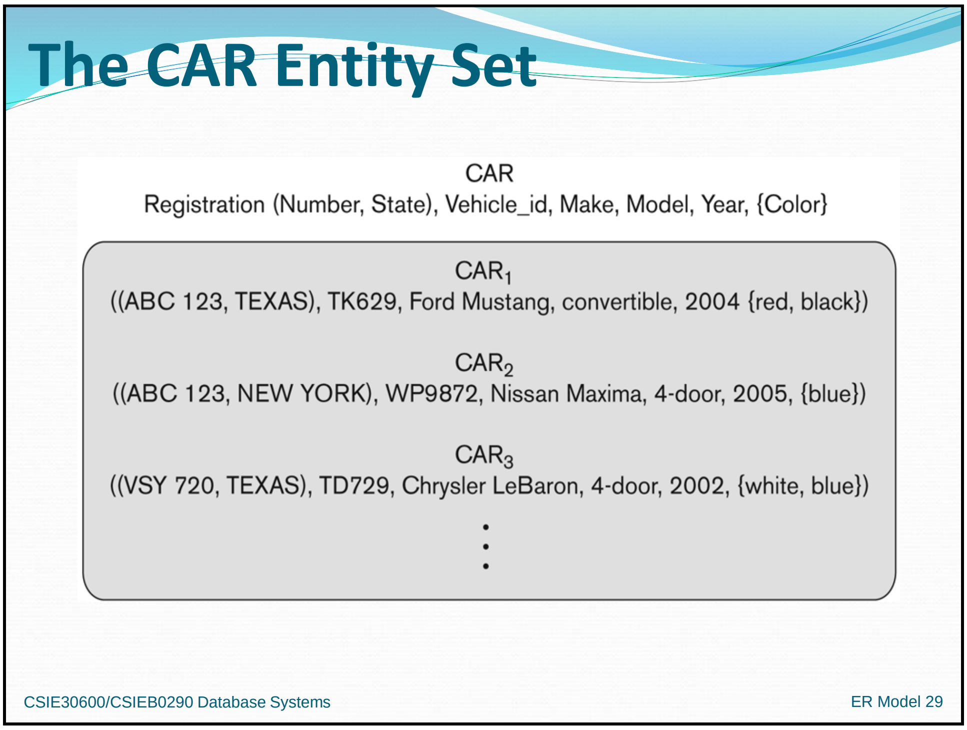


The CAR Entity Type



Entity Set

- Each entity type will have a collection of entities stored in the database (called the entity set).
- Next slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- Entity set is the current state of the entities of that type that are stored in the database



The CAR Entity Set

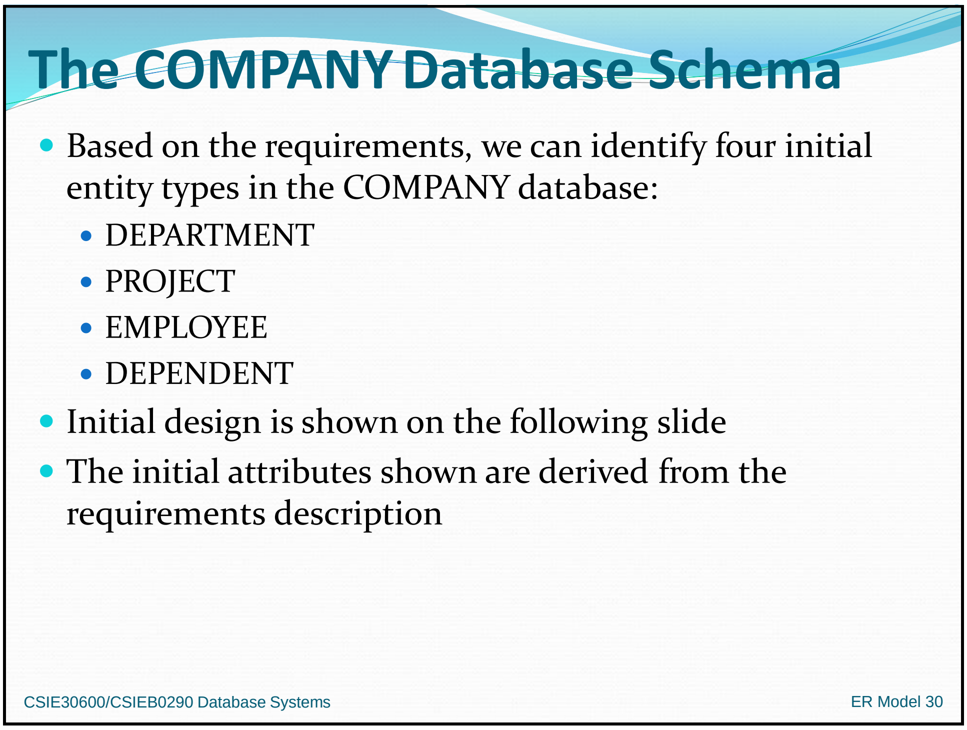
CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

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})

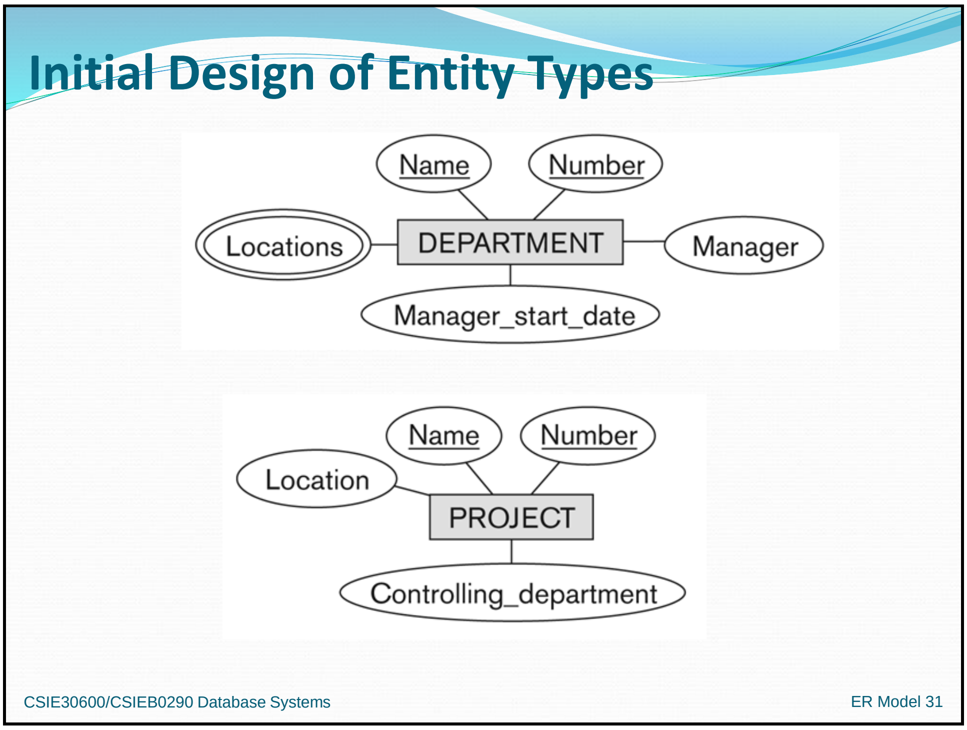
CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮



The COMPANY Database Schema

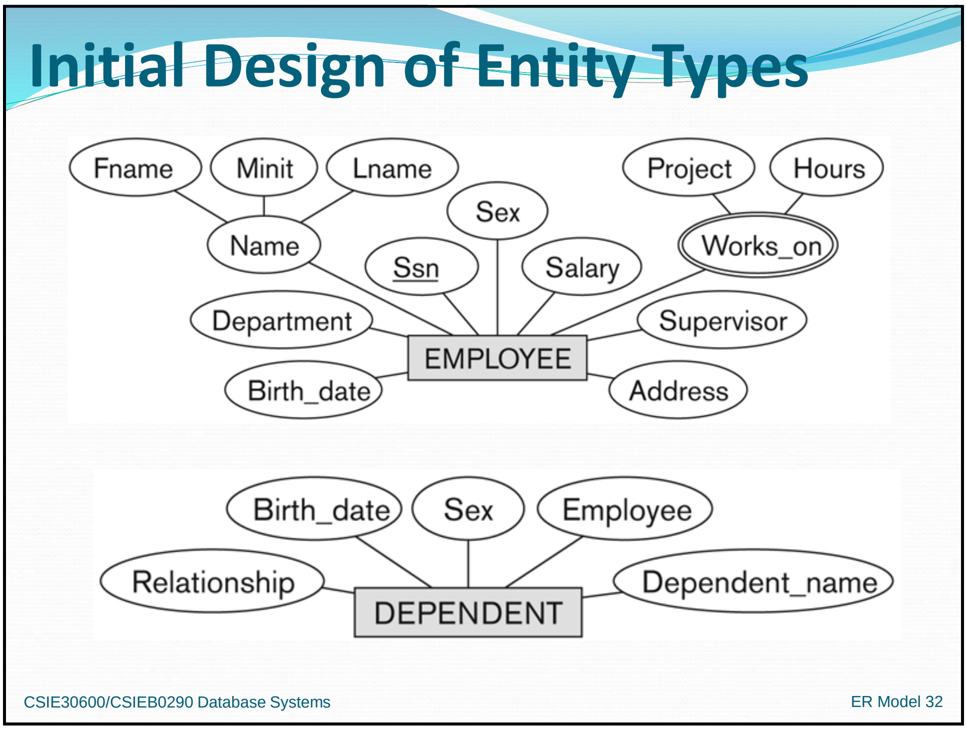
- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Initial design is shown on the following slide
- The initial attributes shown are derived from the requirements description



Initial Design of Entity Types

CSIE30600/CSIEB0290 Database Systems

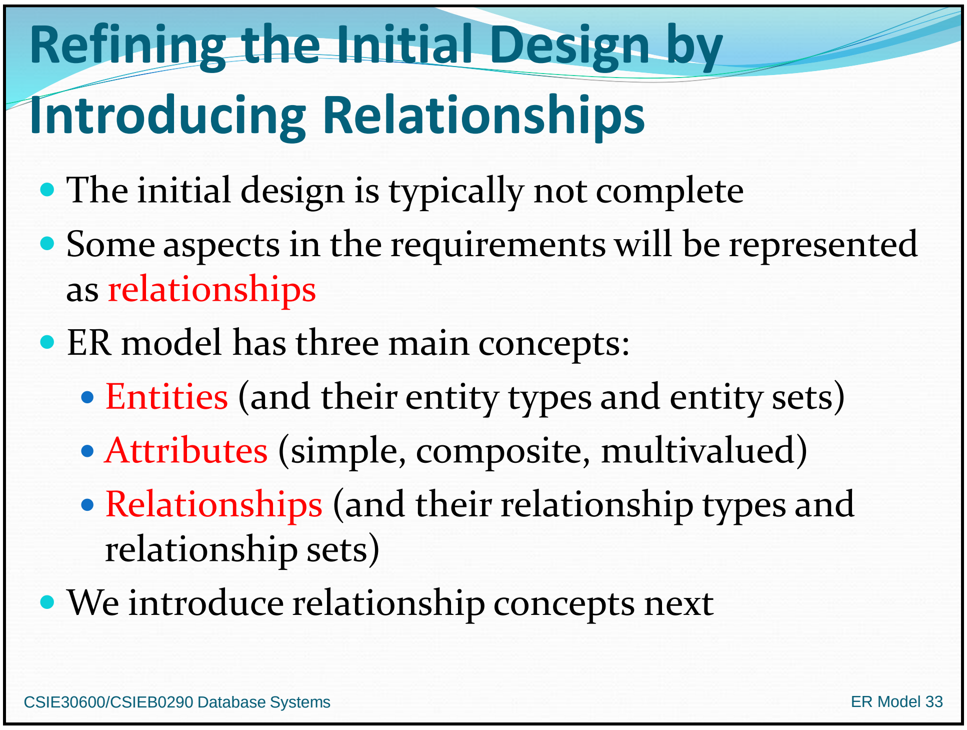
ER Model 31



Initial Design of Entity Types

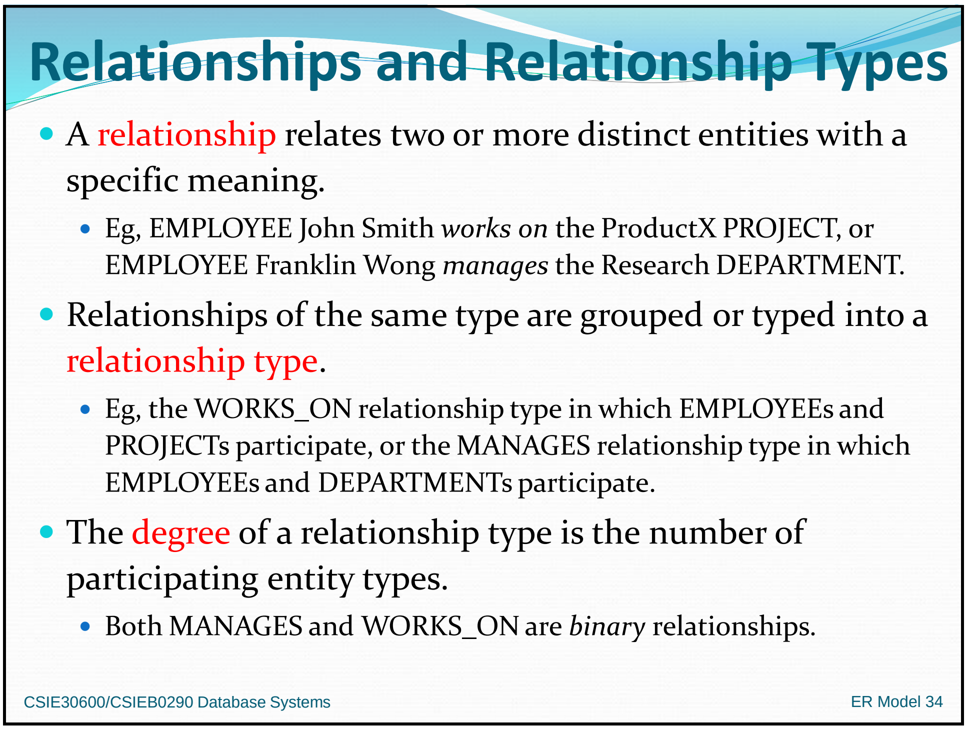
CSIE30600/CSIEB0290 Database Systems

ER Model 32



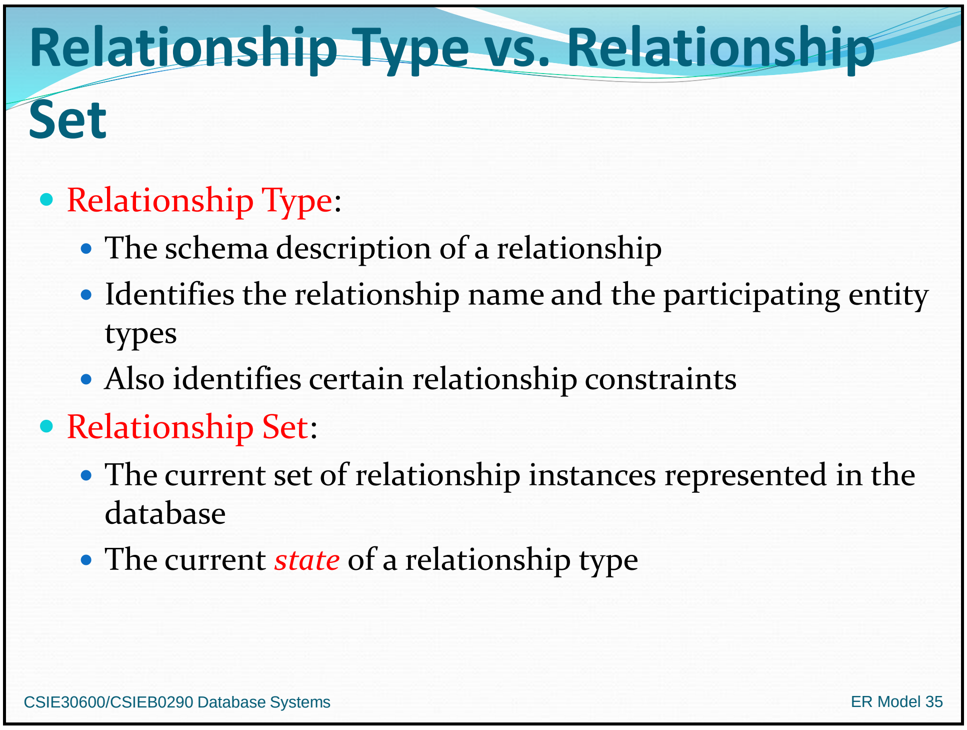
Refining the Initial Design by Introducing Relationships

- The initial design is typically not complete
- Some aspects in the requirements will be represented as relationships
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)
- We introduce relationship concepts next



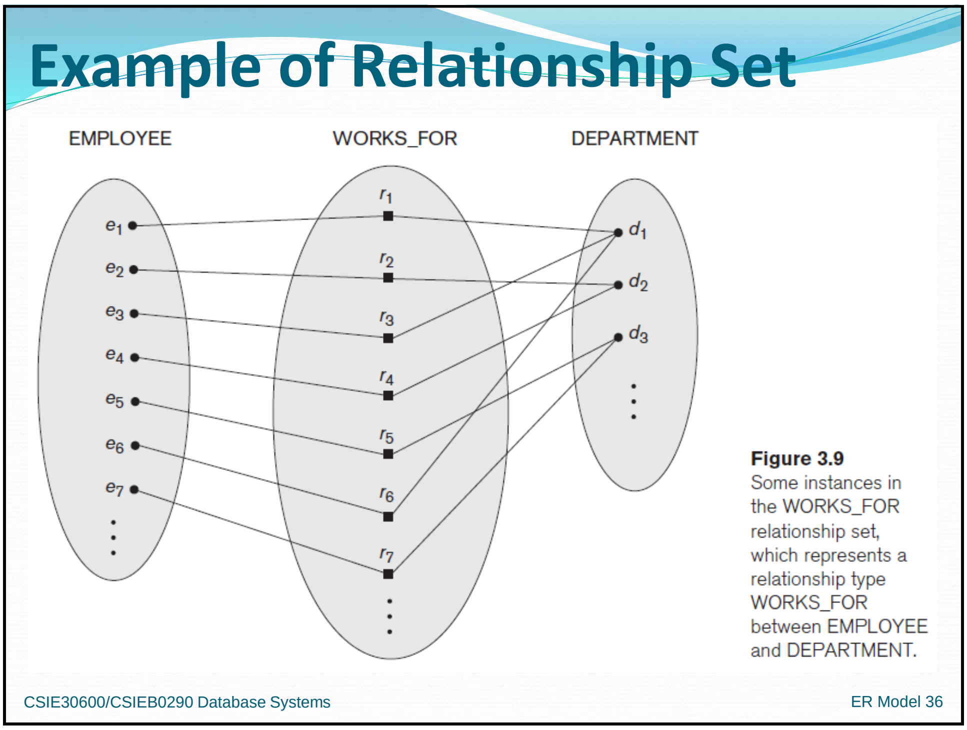
Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - Eg, 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.
  - Eg, 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.



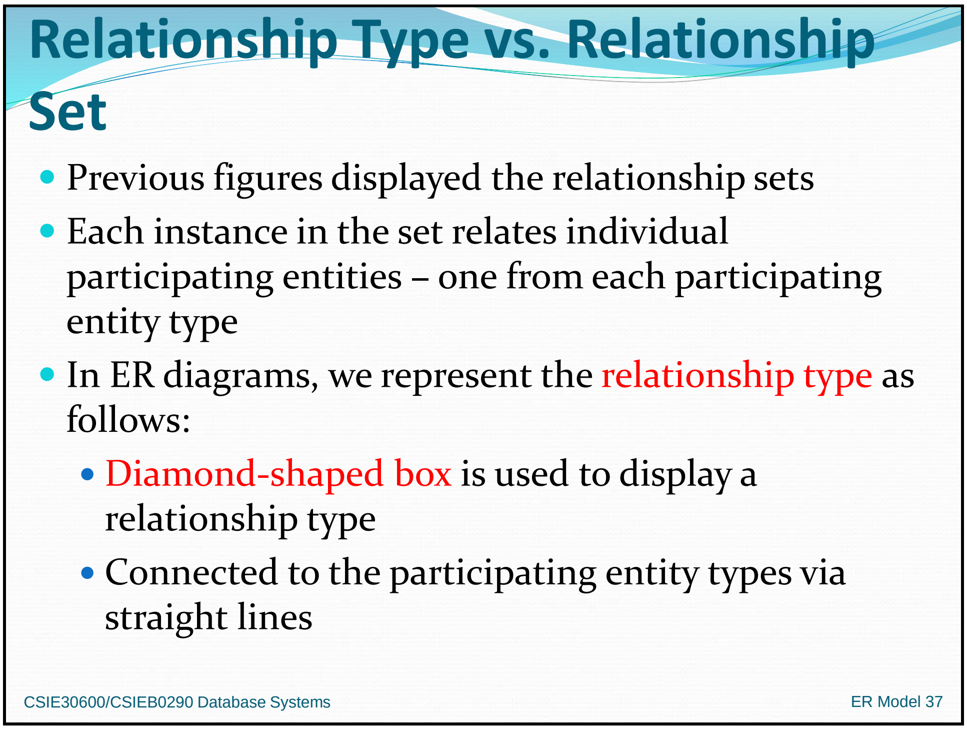
Relationship Type vs. Relationship Set

- Relationship Type:
  - 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



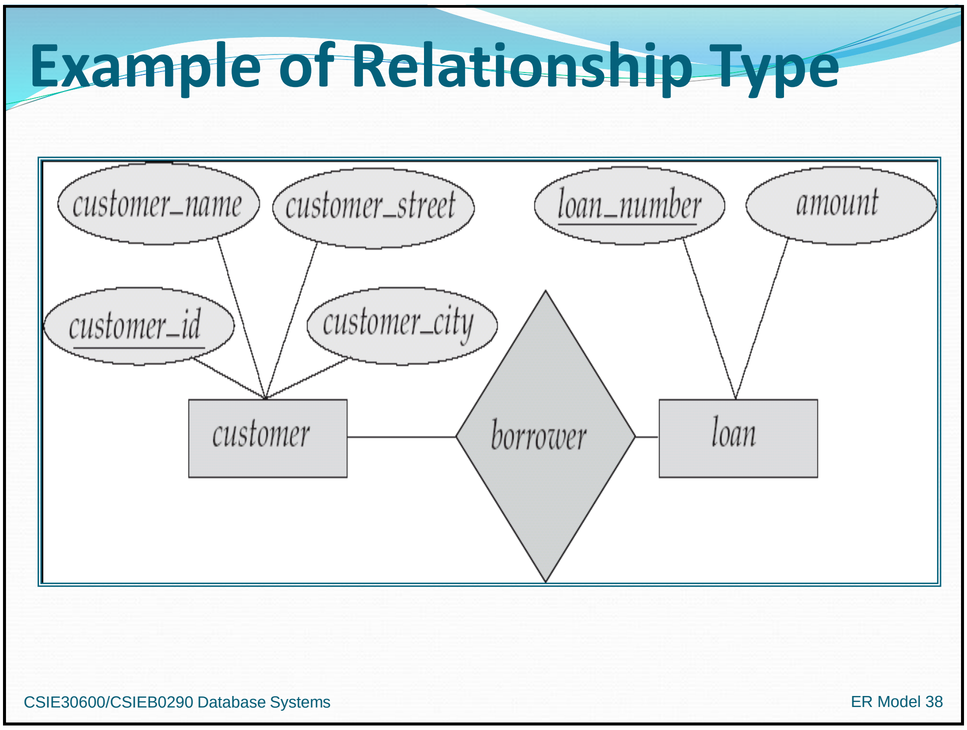
Example of Relationship Set

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



Relationship Type vs. Relationship Set

- Previous figures displayed the relationship sets
- 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



Example of Relationship Type

Refining the COMPANY Database Schema

- 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)

Relationship Set (Binary)

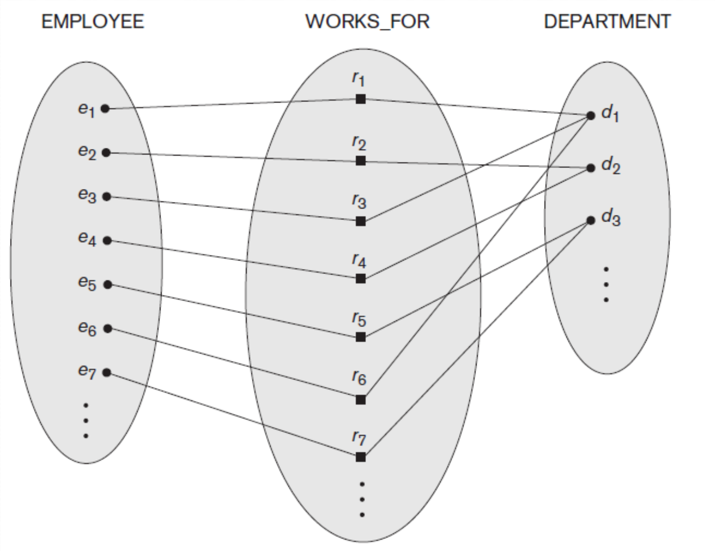


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship Set (Ternary)

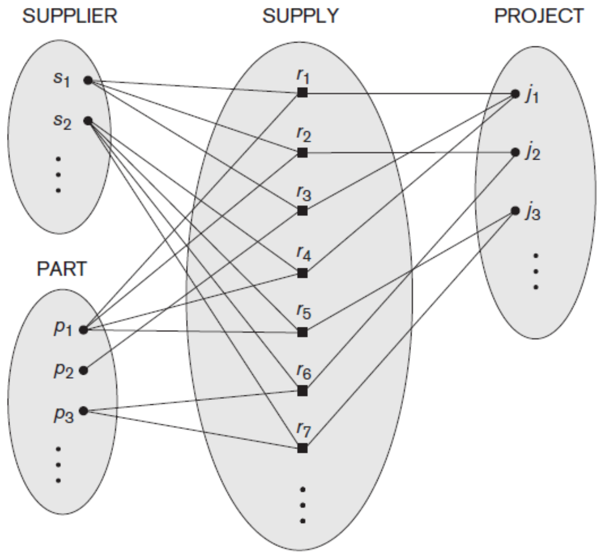


Figure 3.10
Some relationship instances in the SUPPLY ternary relationship set.

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.

Recursive Relationship Type

- A relationship type associating the same participating entity type in distinct roles
- 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

Recursive Relationship

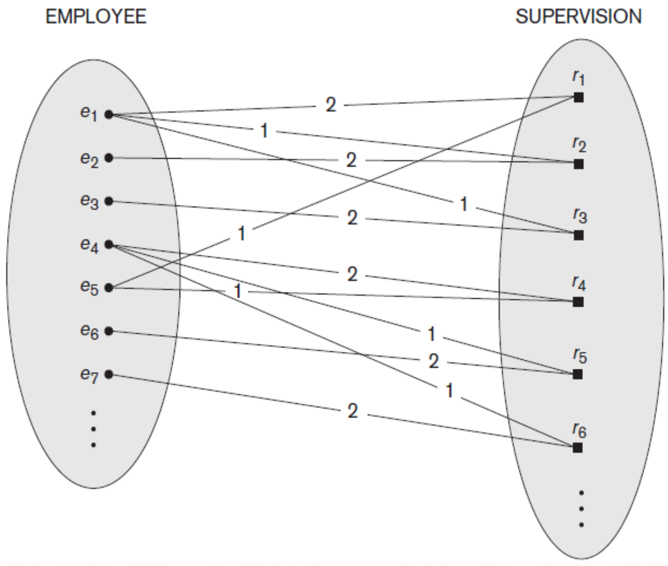
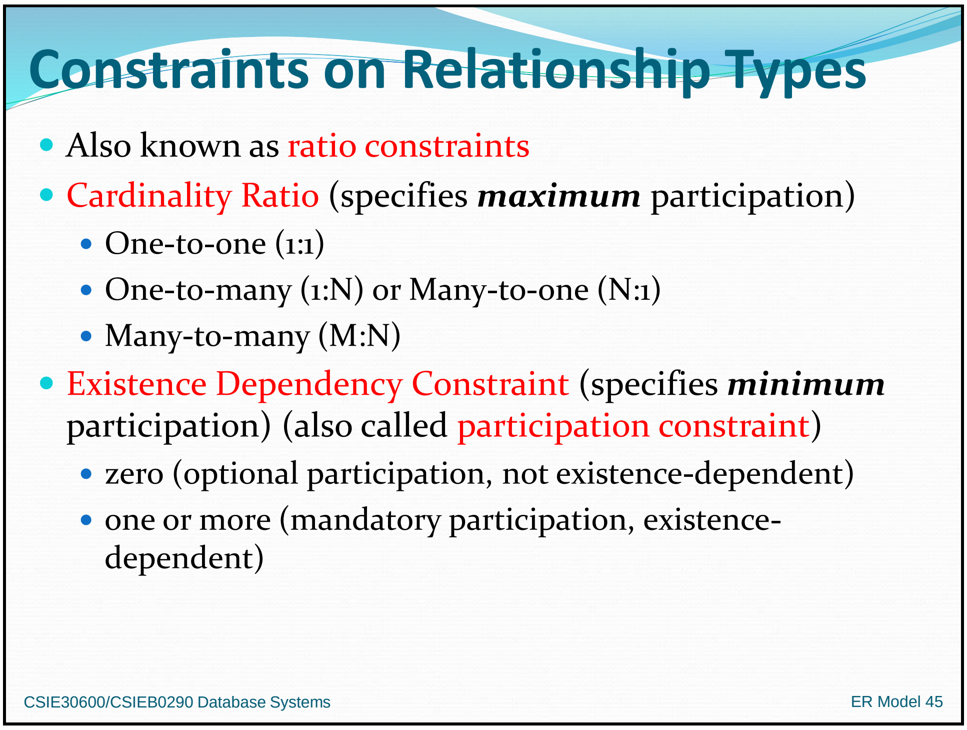
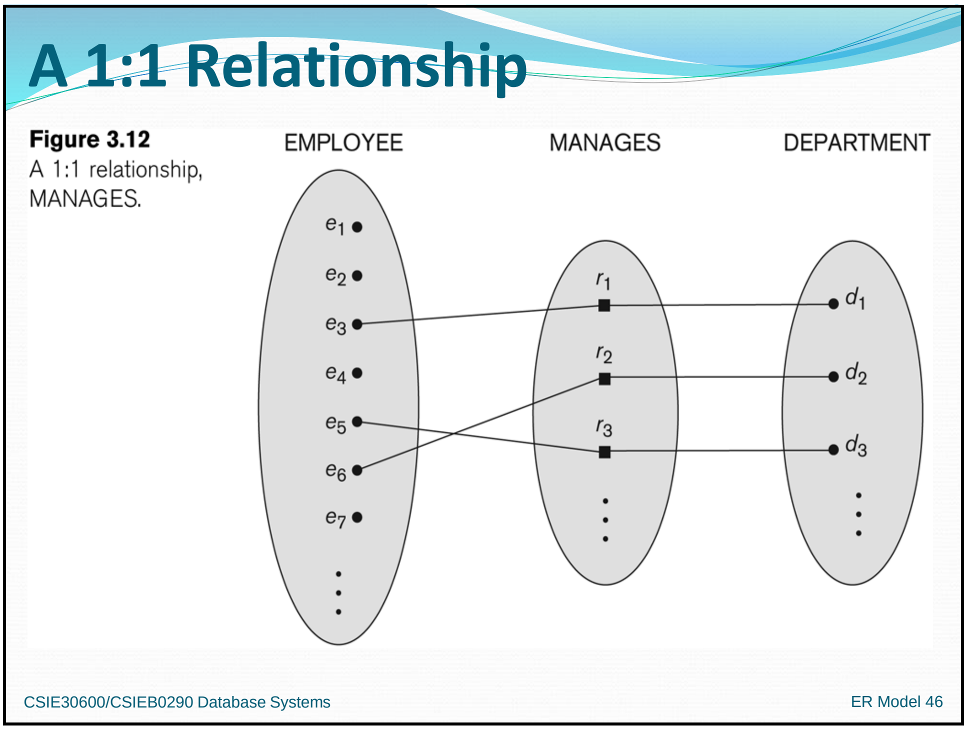


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).



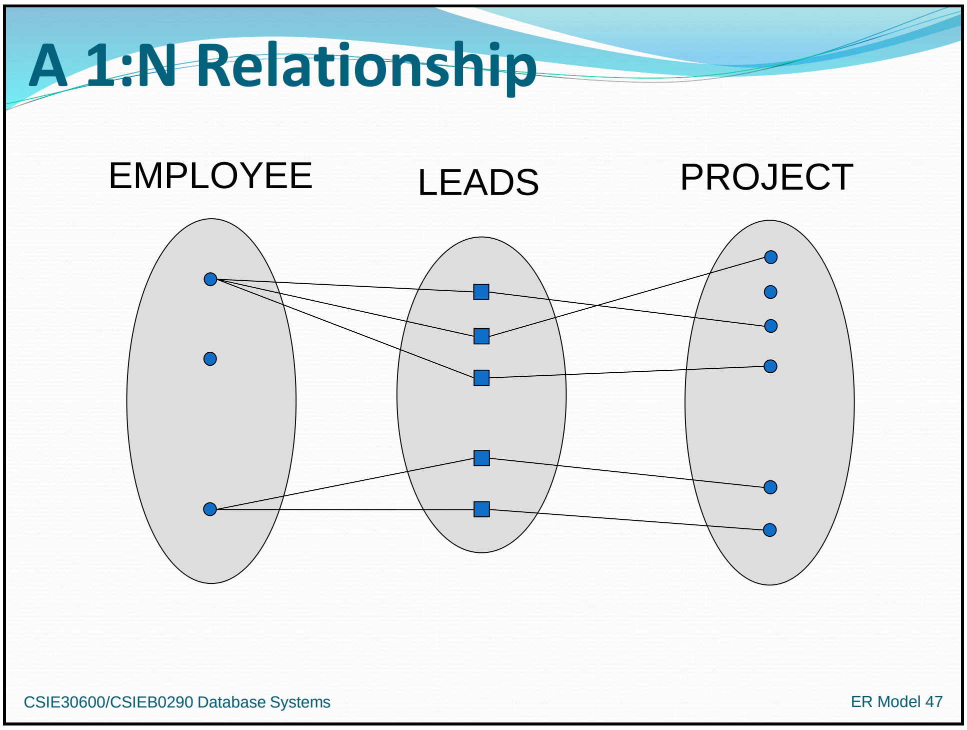
Constraints on Relationship Types

- 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)



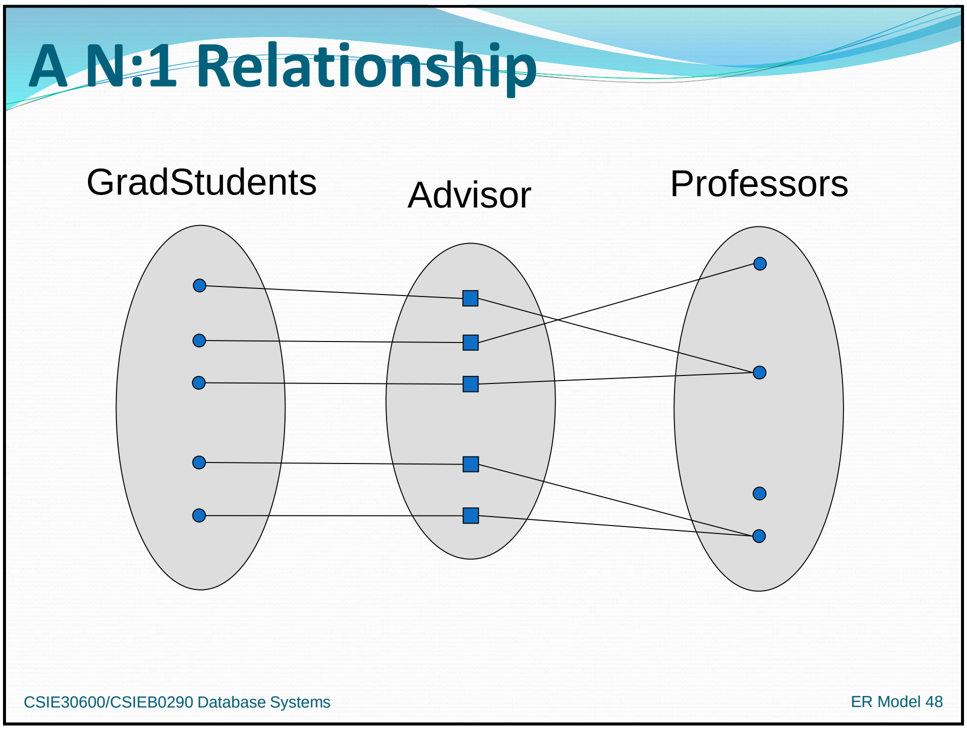
A 1:1 Relationship

Figure 3.12
A 1:1 relationship, MANAGES.



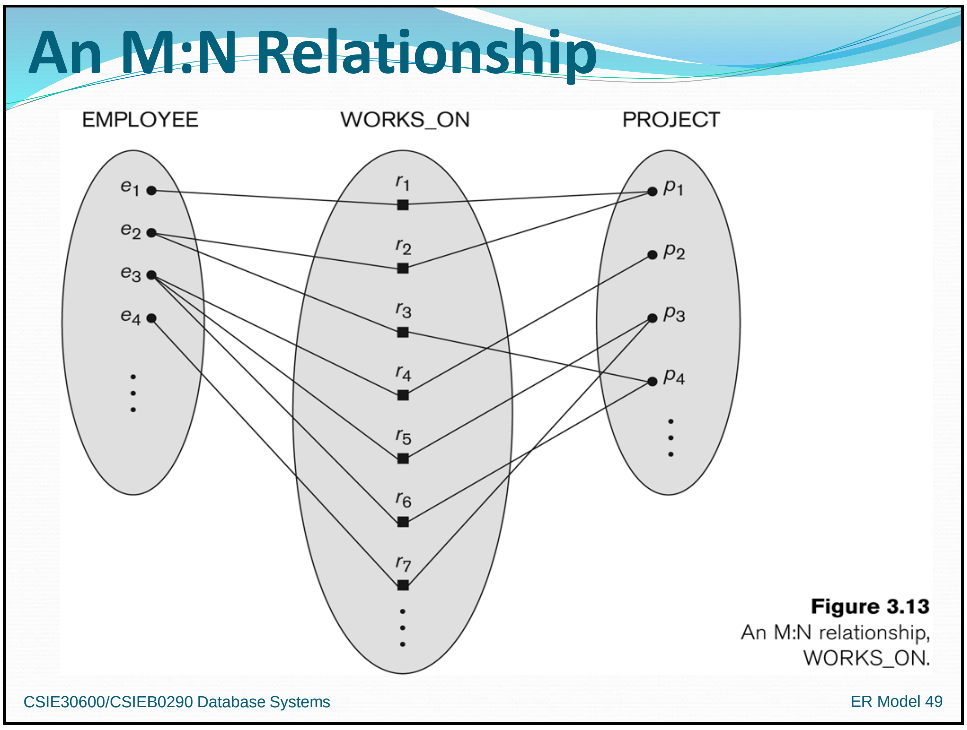
A 1:N Relationship

CSIE30600/CSIEB0290 Database Systems ER Model 47



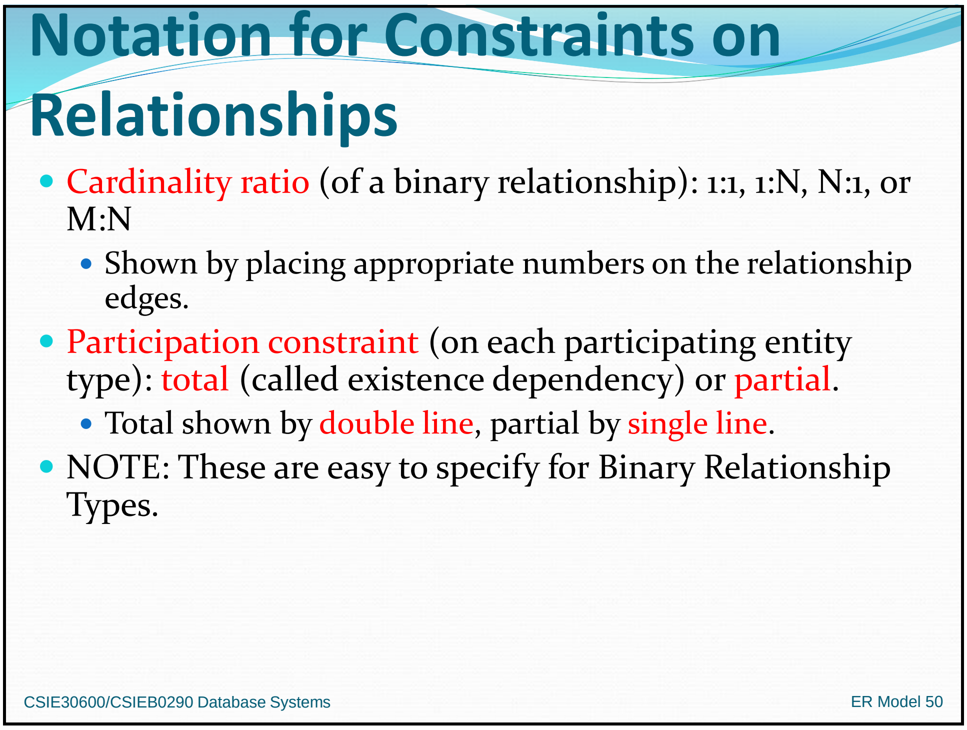
A N:1 Relationship

CSIE30600/CSIEB0290 Database Systems ER Model 48



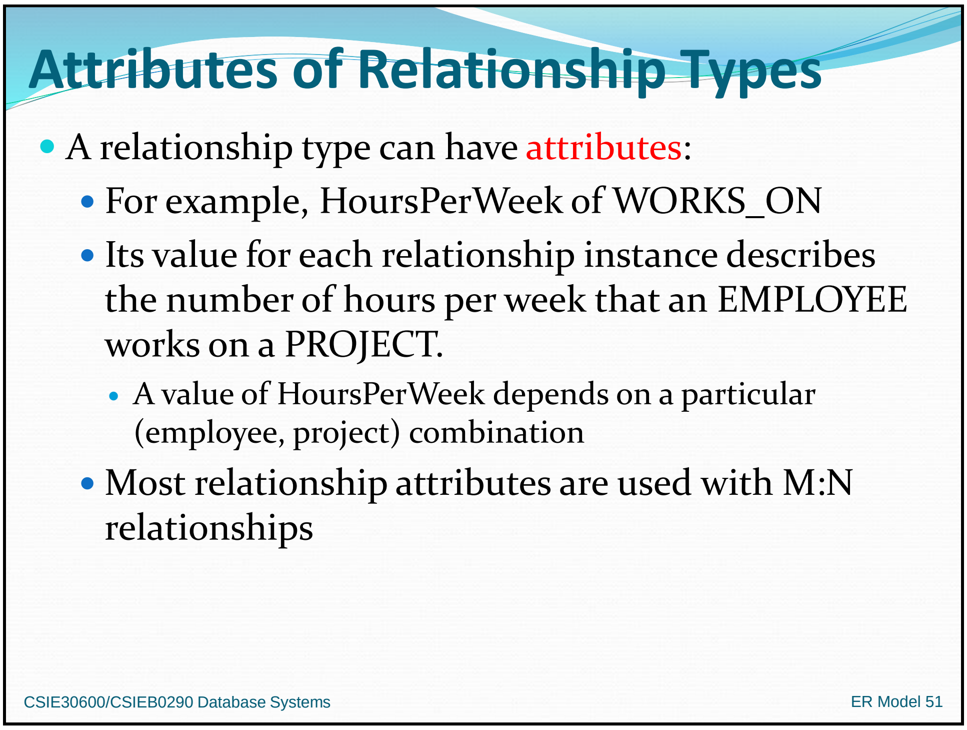
An M:N Relationship

Figure 3.13
An M:N relationship, WORKS_ON.



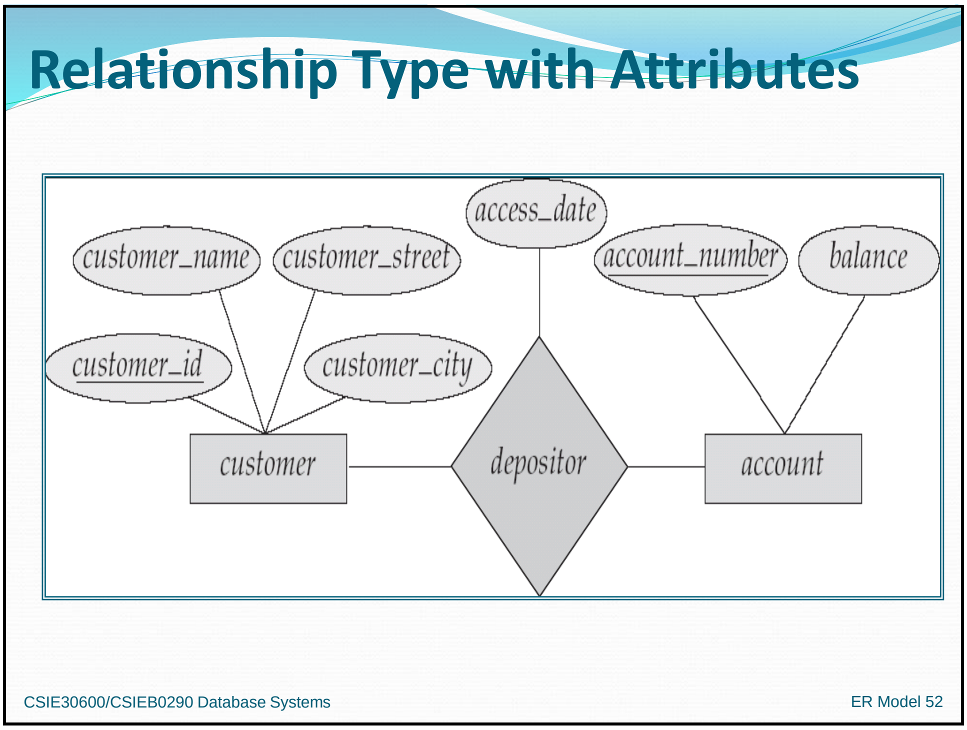
Notation for Constraints on Relationships

- 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.
- NOTE: These are easy to specify for Binary Relationship Types.



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

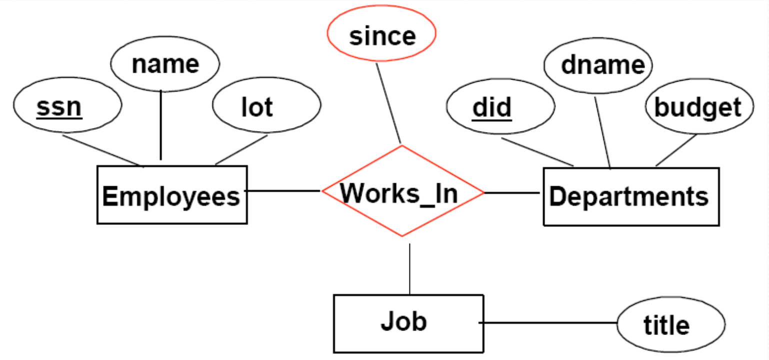


Relationship Type with Attributes

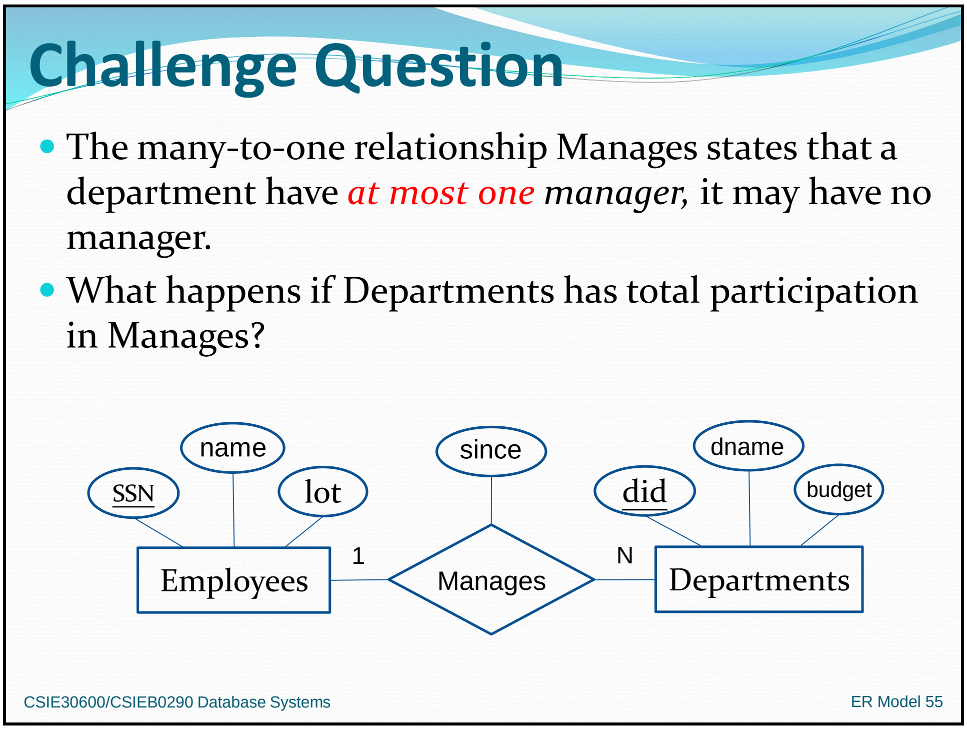
Attributes of Relationship Types

- Attributes of 1:1 relationship type can be migrated to one entity type
- For a 1:N relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For M:N relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes

Challenge Questions



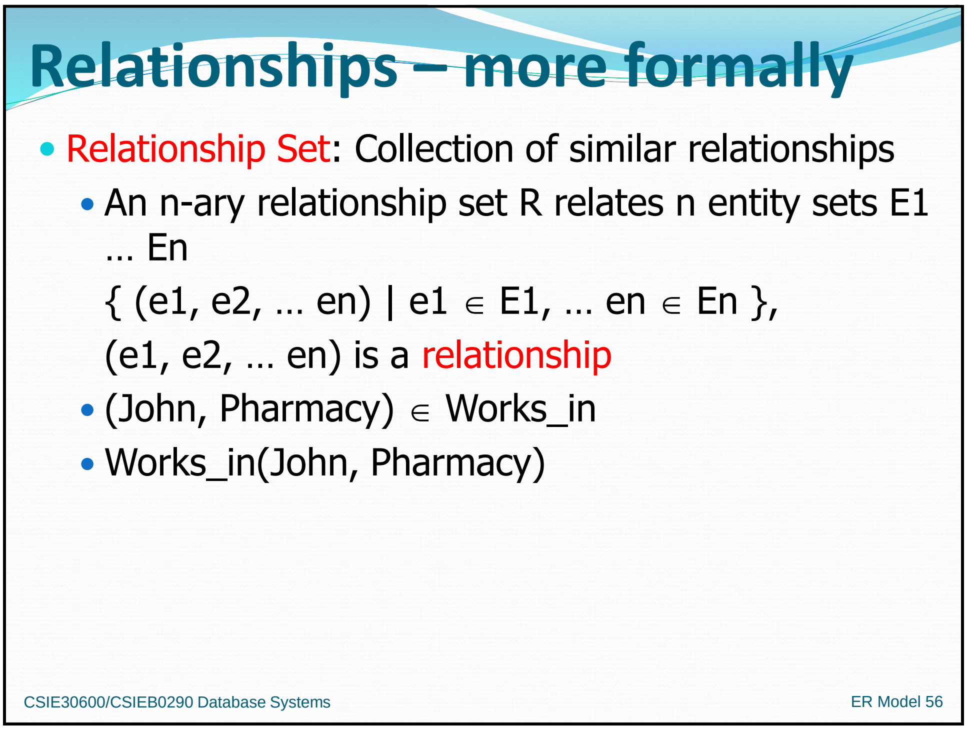
- Can we instead place “since” in the Job entity?
- Or place “since” in the Employee entity?



Challenge Question

- The many-to-one relationship Manages states that a department have *at most one manager*, it may have no manager.
- What happens if Departments has total participation in Manages?

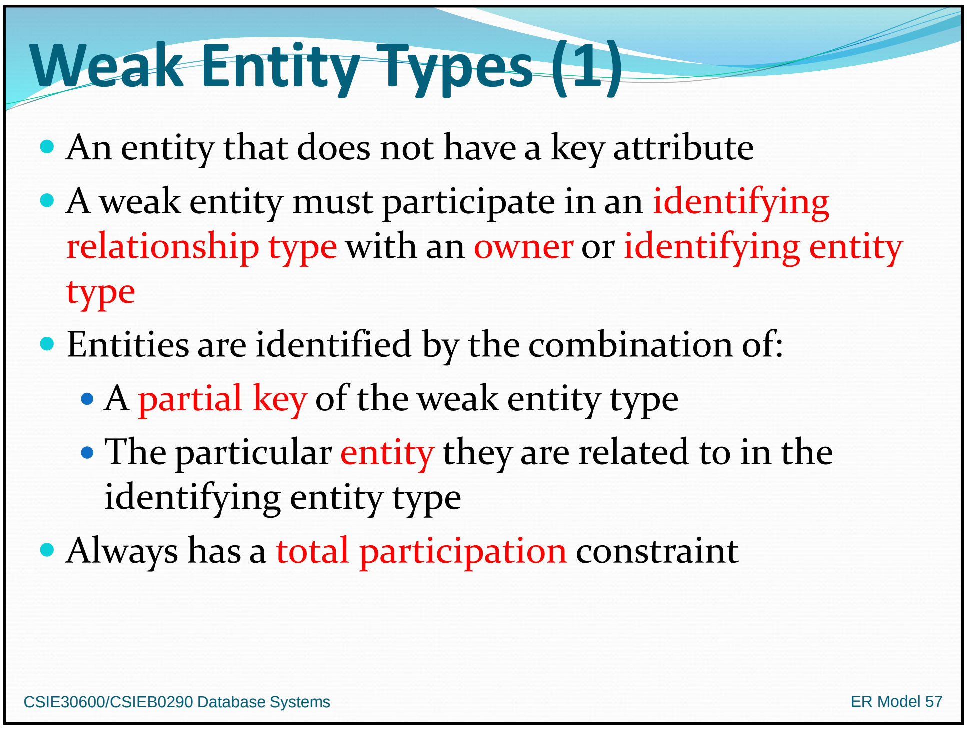
CSIE30600/CSIEB0290 Database Systems ER Model 55



Relationships – more formally

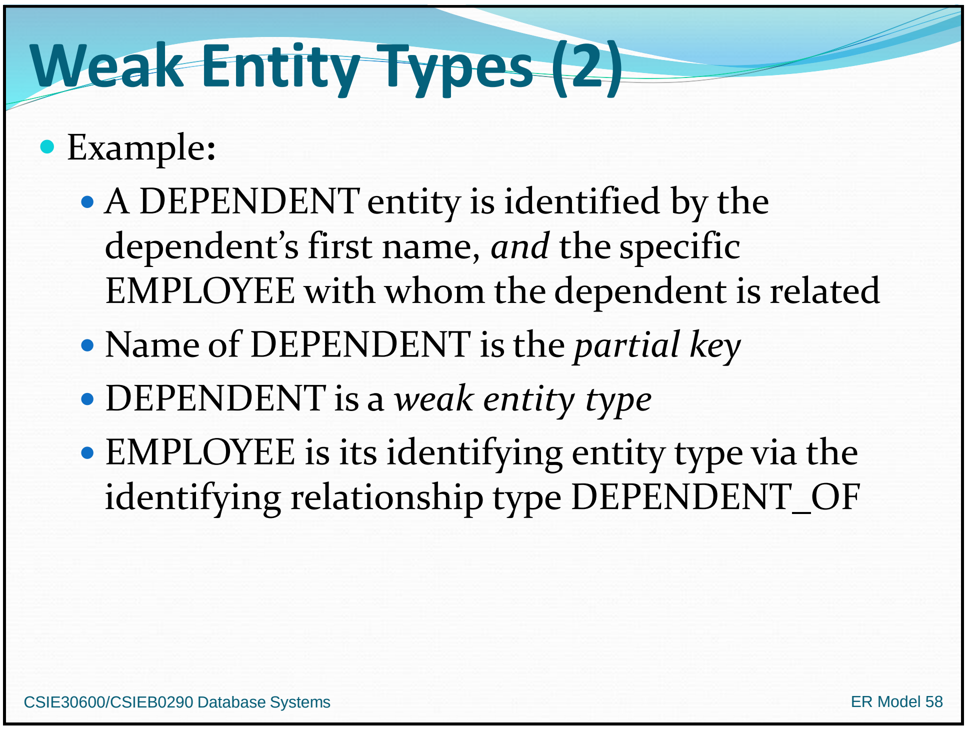
- Relationship Set: Collection of similar relationships
  - An n-ary relationship set R relates n entity sets E1 … En
    { (e1, e2, … en) | e1 ∈ E1, … en ∈ En },
    (e1, e2, … en) is a relationship
  - (John, Pharmacy) ∈ Works_in
  - Works_in(John, Pharmacy)

CSIE30600/CSIEB0290 Database Systems ER Model 56



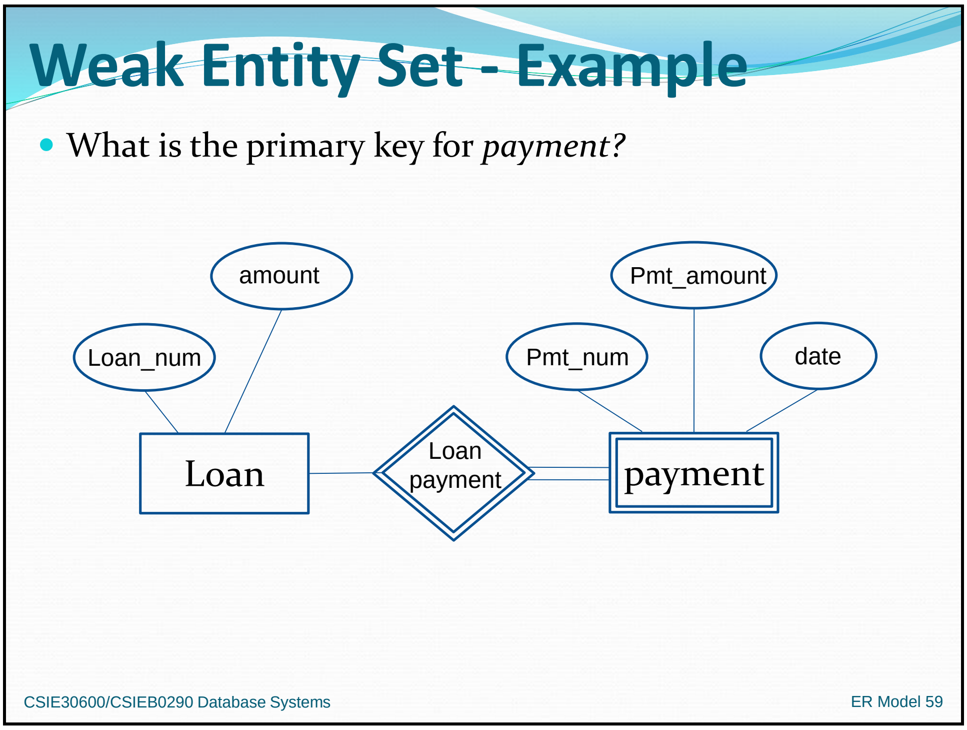
Weak Entity Types (1)

- An entity that does not have a key attribute
- 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 entity type
- Always has a total participation constraint



Weak Entity Types (2)

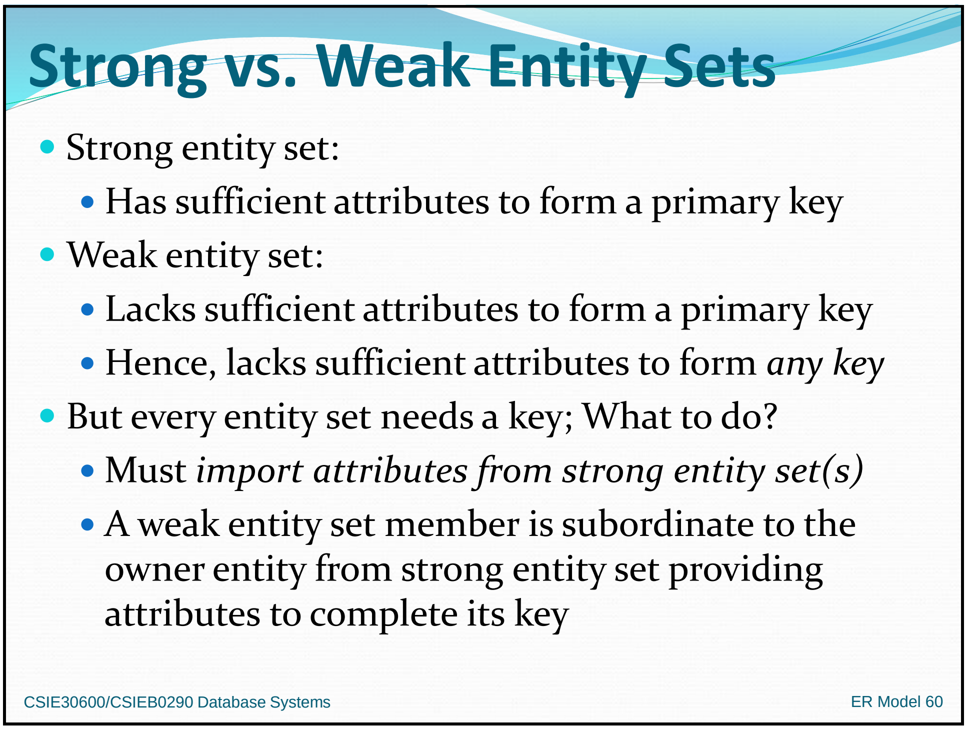
- Example:
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type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



Weak Entity Set - Example

- What is the primary key for *payment?*

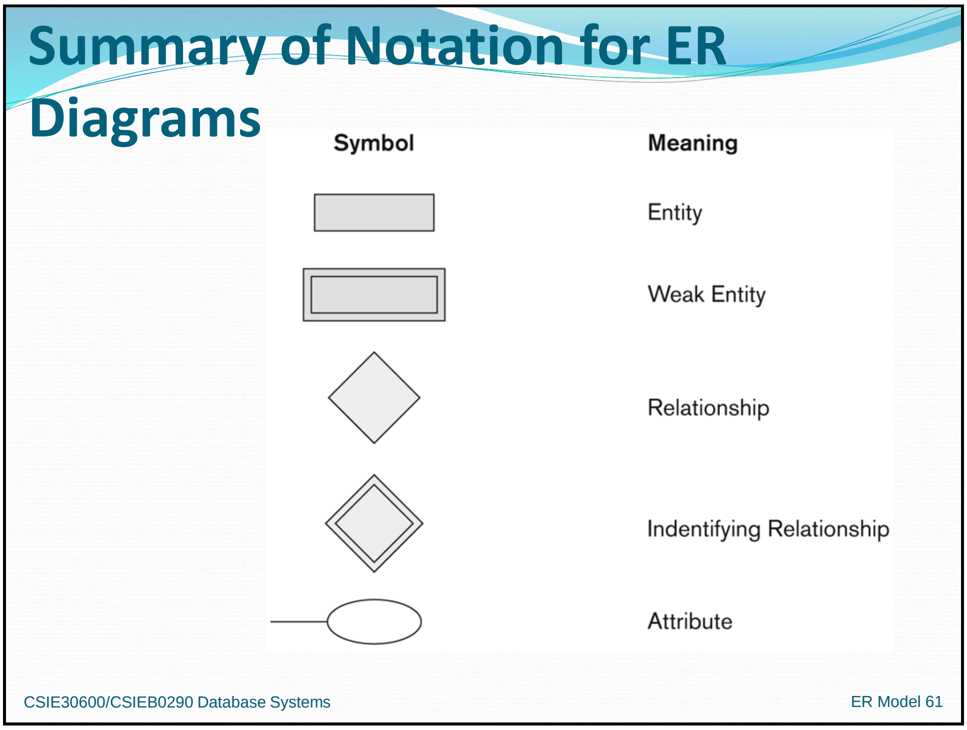
CSIE30600/CSIEB0290 Database Systems ER Model 59



Strong vs. Weak Entity Sets

- Strong entity set:
  - Has sufficient attributes to form a primary key
- Weak entity set:
  - Lacks sufficient attributes to form a primary key
  - Hence, lacks sufficient attributes to form *any key*
- But every entity set needs a key; What to do?
  - Must *import attributes from strong entity set(s)*
  - A weak entity set member is subordinate to the owner entity from strong entity set providing attributes to complete its key

CSIE30600/CSIEB0290 Database Systems ER Model 60



Summary of Notation for ER Diagrams

Symbol

Meaning

Entity

Weak Entity

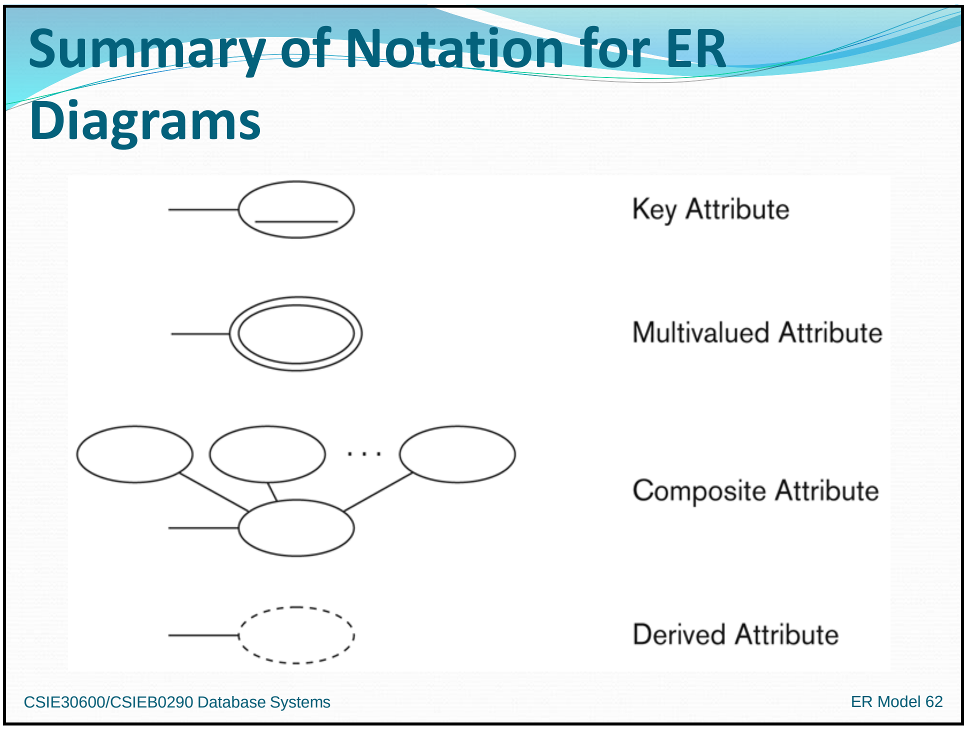
Relationship

Indentifying Relationship

Attribute

CSIE30600/CSIEB0290 Database Systems

ER Model 61



Summary of Notation for ER Diagrams

Key Attribute

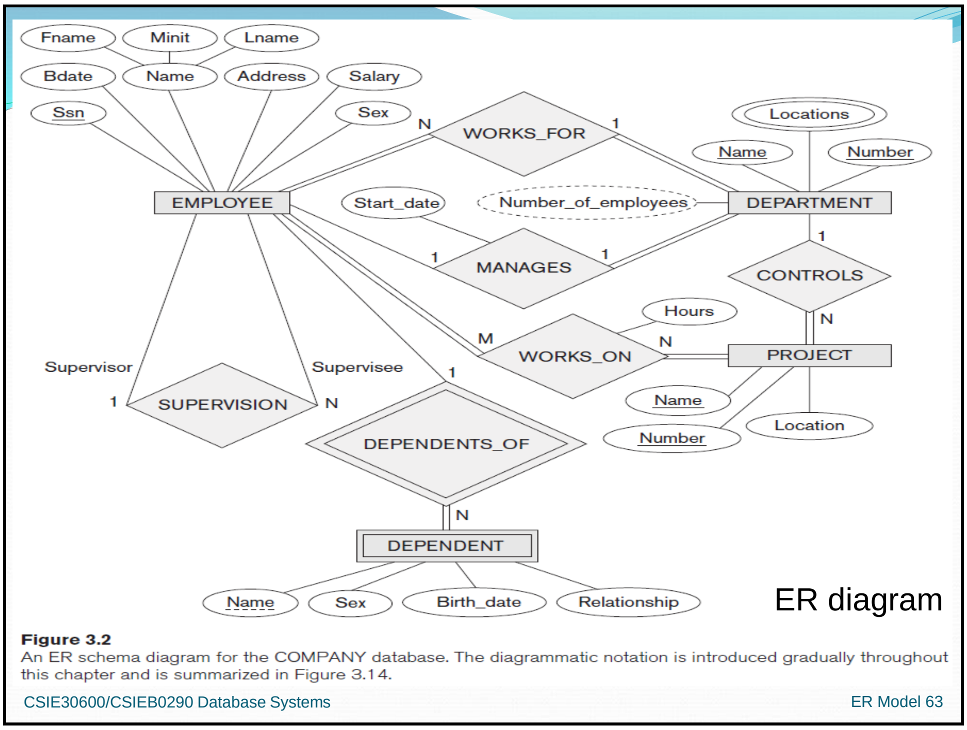
Multivalued Attribute

Composite Attribute

Derived Attribute

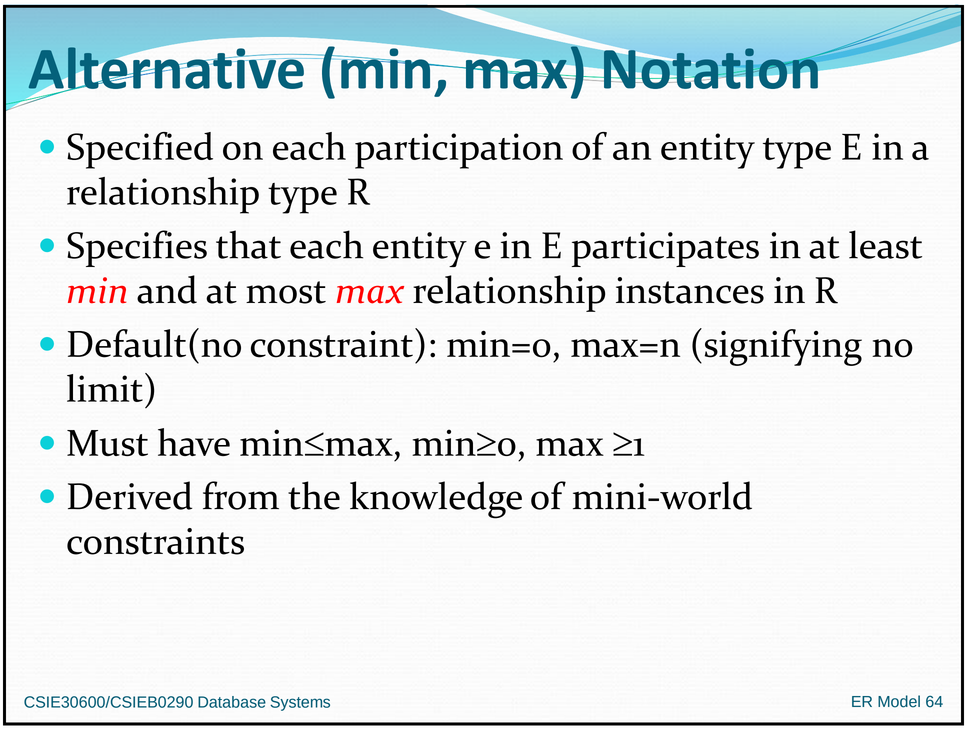
CSIE30600/CSIEB0290 Database Systems

ER Model 62



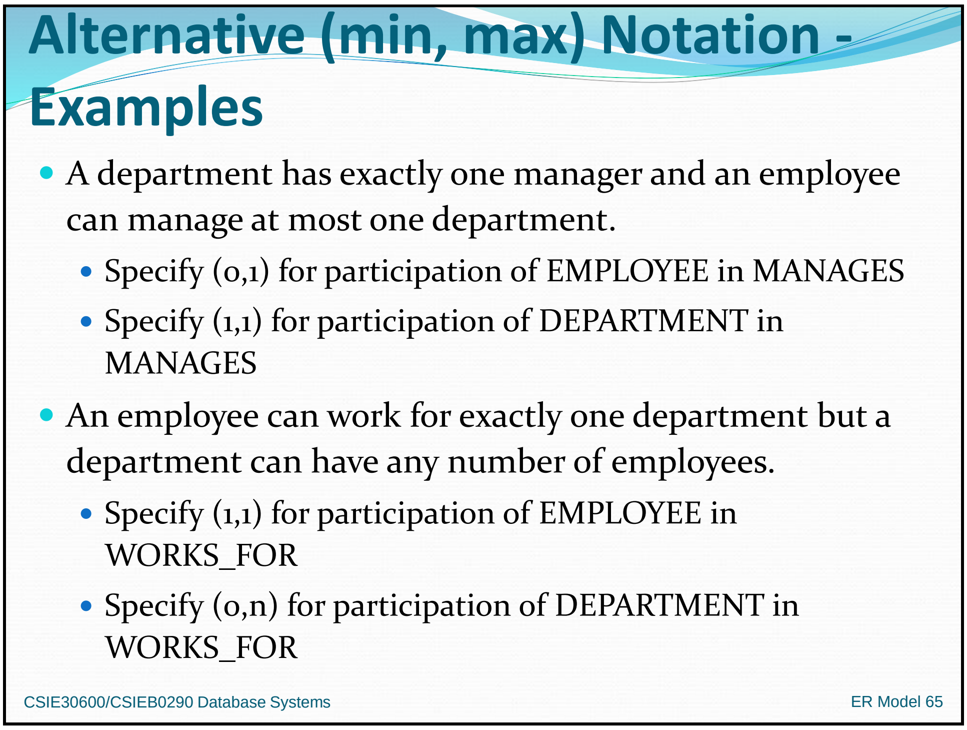
ER diagram

Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.



Alternative (min, max) Notation

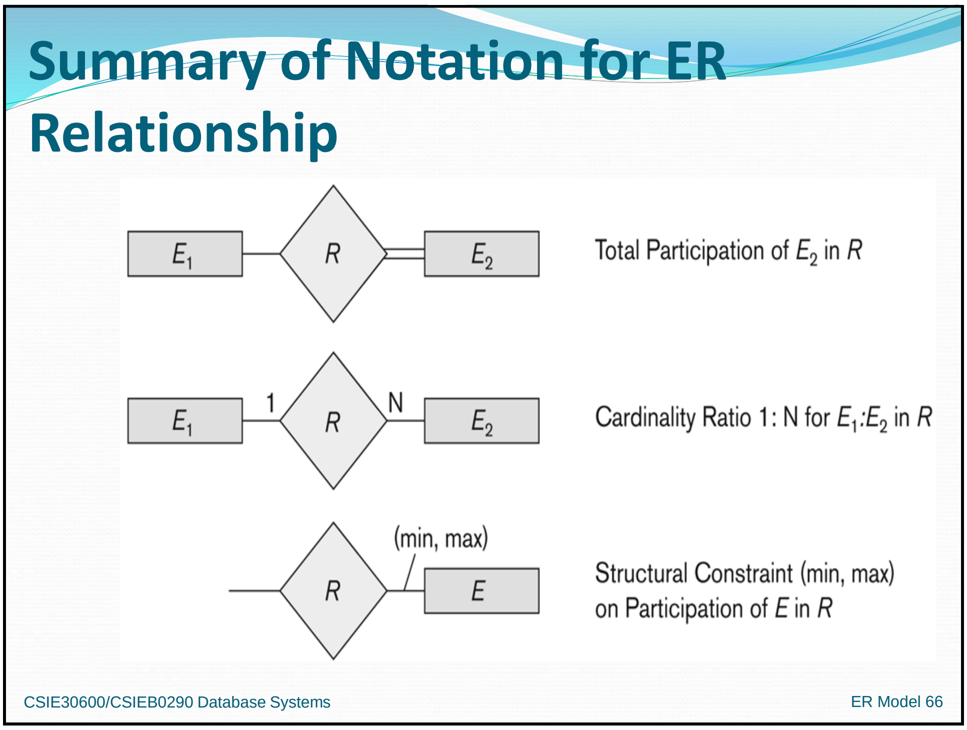
- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints



Alternative (min, max) Notation - 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

Summary of Notation for ER Relationship


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

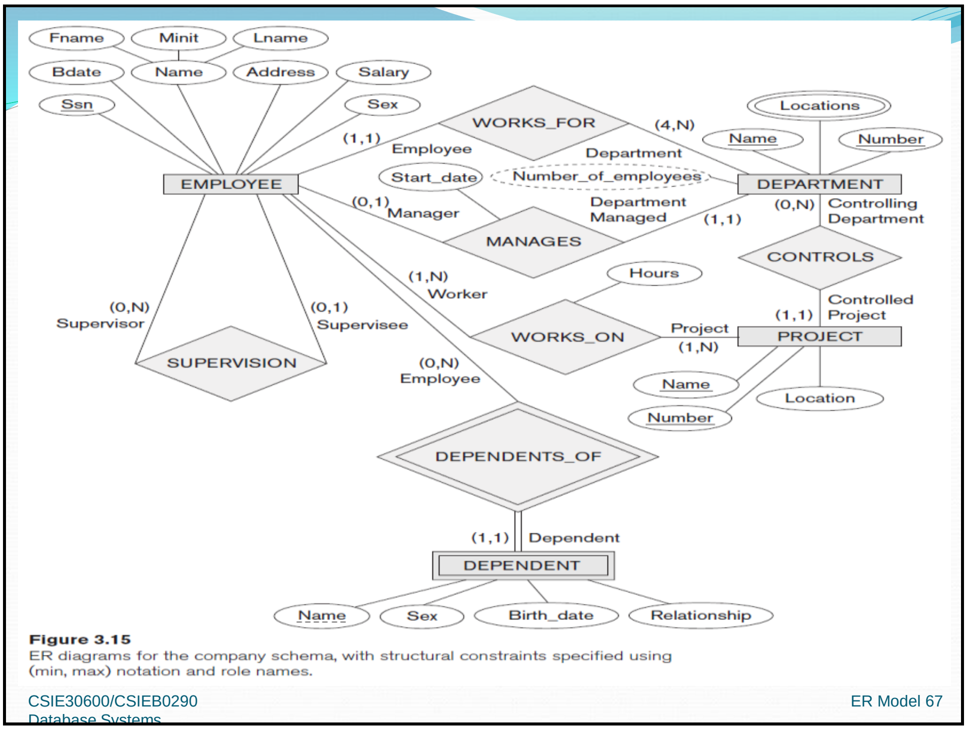
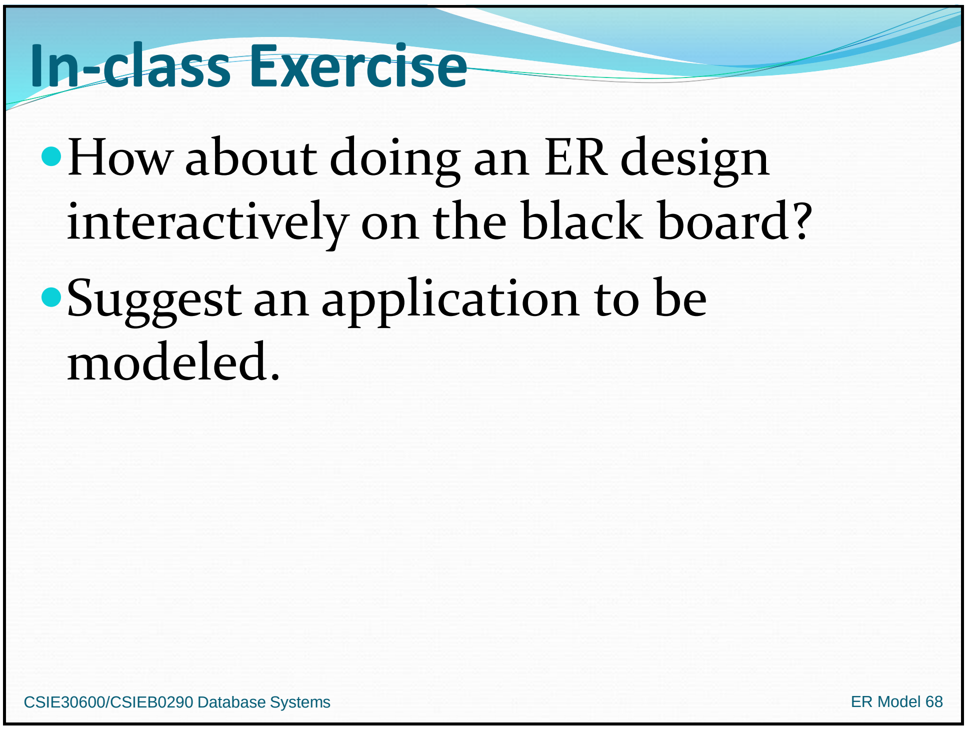
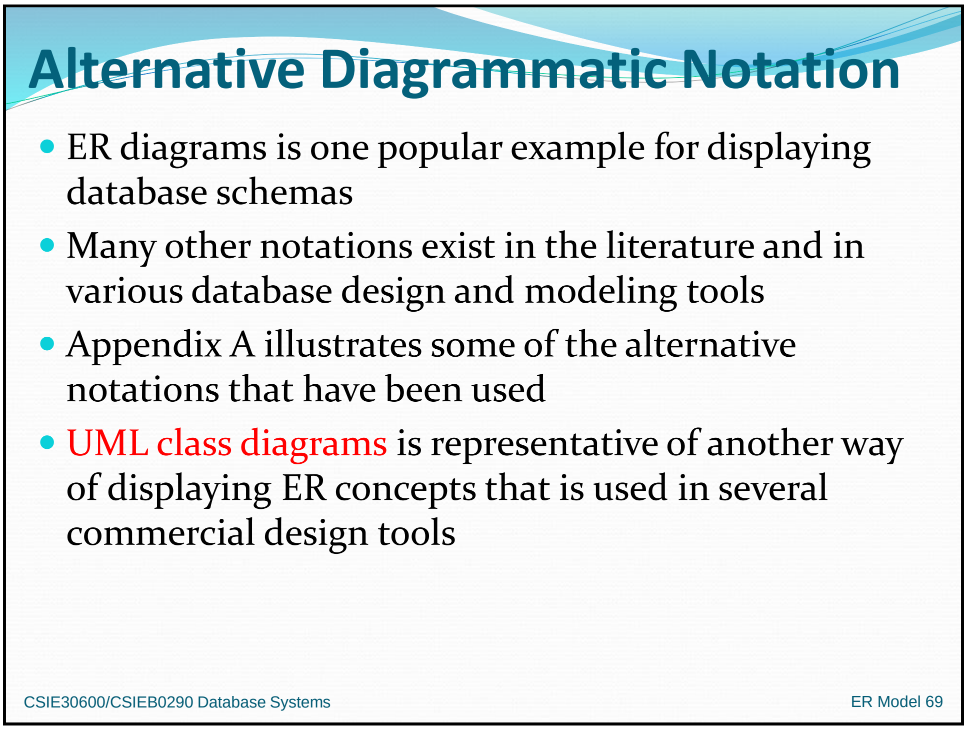


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.



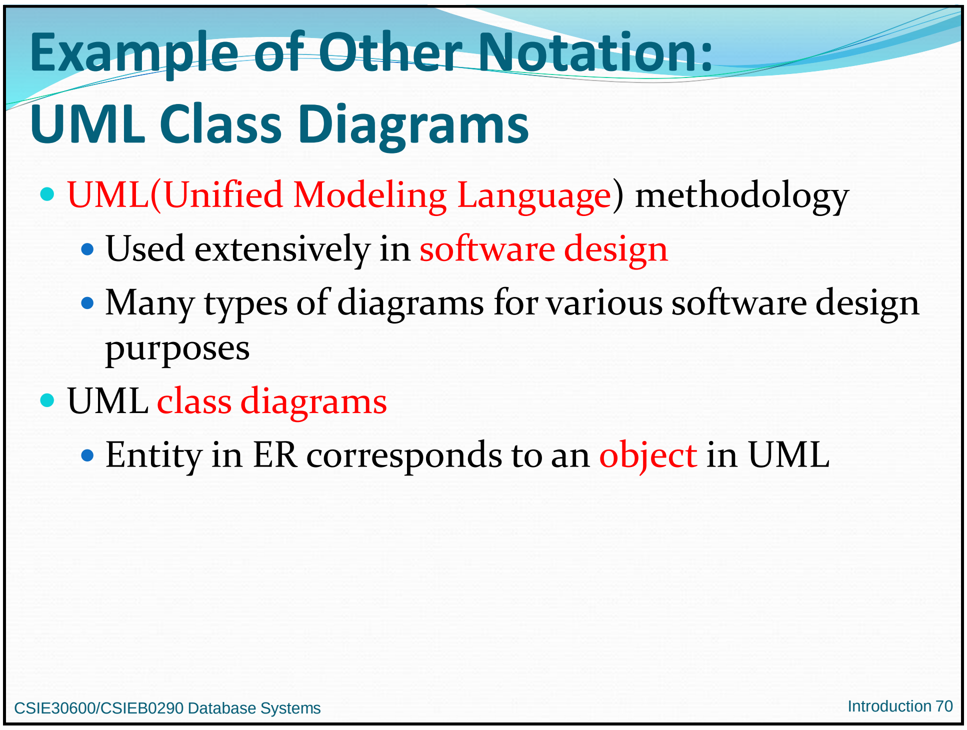
In-class Exercise

- How about doing an ER design interactively on the black board?
- Suggest an application to be modeled.



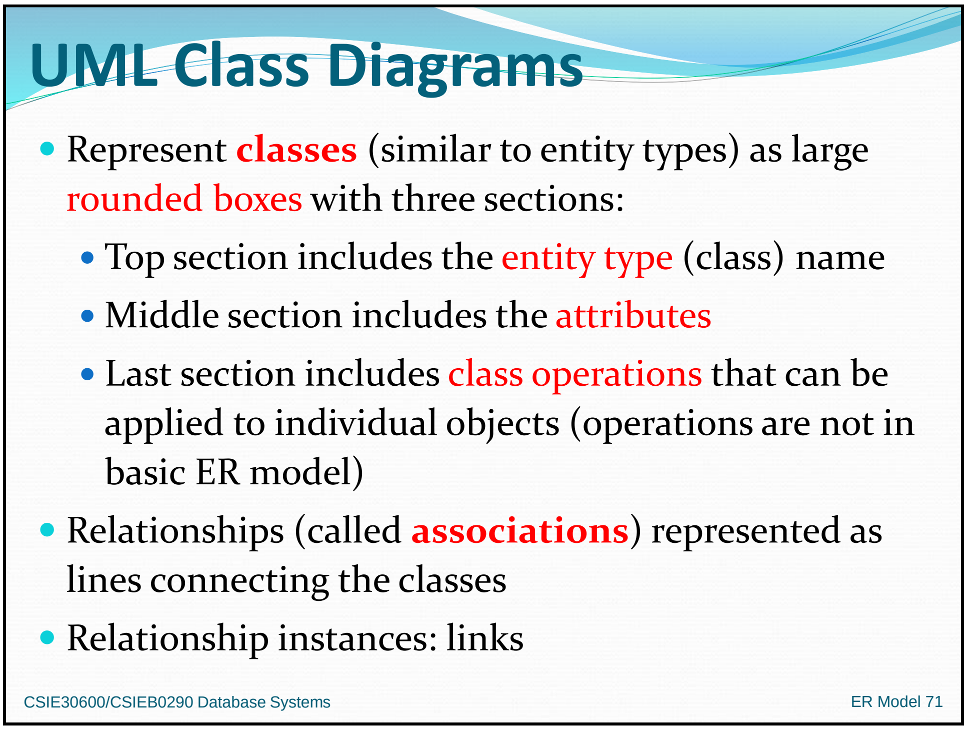
Alternative Diagrammatic Notation

- ER diagrams is one popular example for displaying database schemas
- Many other notations exist in the literature and in various database design and modeling tools
- Appendix A illustrates some of the alternative notations that have been used
- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools



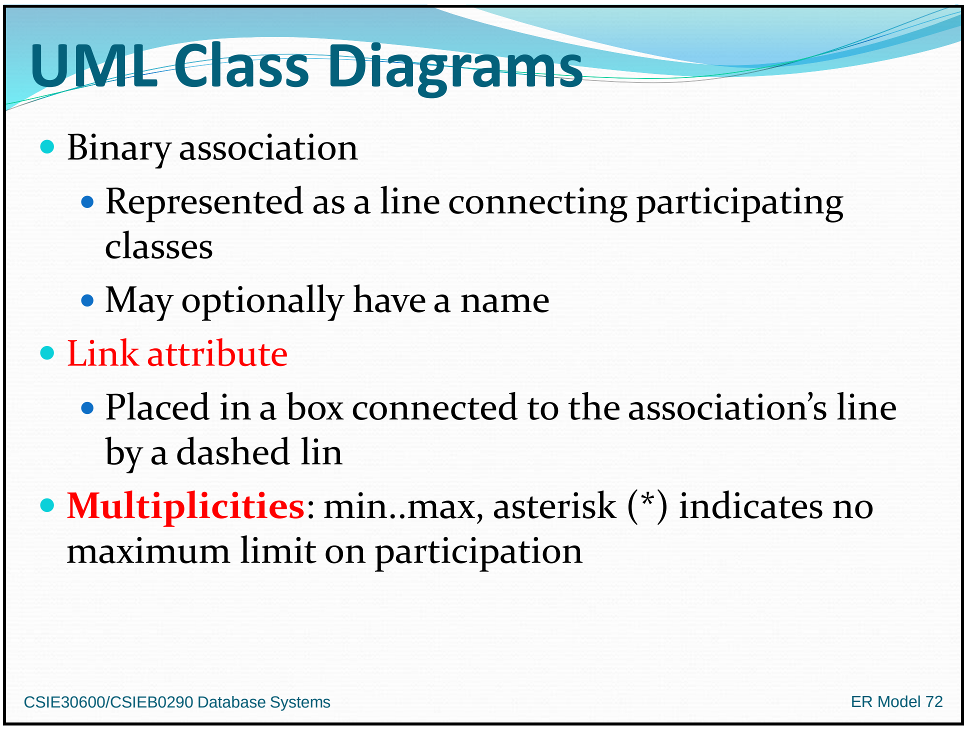
Example of Other Notation: UML Class Diagrams

- UML(Unified Modeling Language) methodology
  - Used extensively in software design
  - Many types of diagrams for various software design purposes
- UML class diagrams
  - Entity in ER corresponds to an object in UML



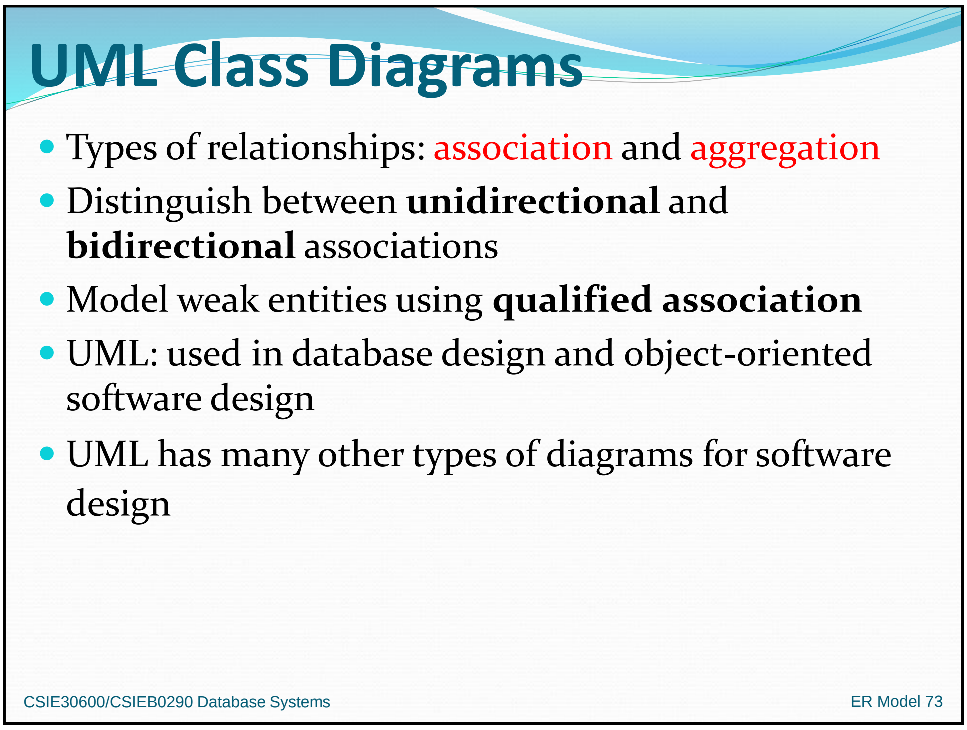
UML Class Diagrams

- Represent **classes** (similar to entity types) as large rounded boxes with three sections:
  - Top section includes the entity type (class) name
  - Middle section includes the attributes
  - Last section includes class operations that can be applied to individual objects (operations are not in basic ER model)
- Relationships (called **associations**) represented as lines connecting the classes
- Relationship instances: links



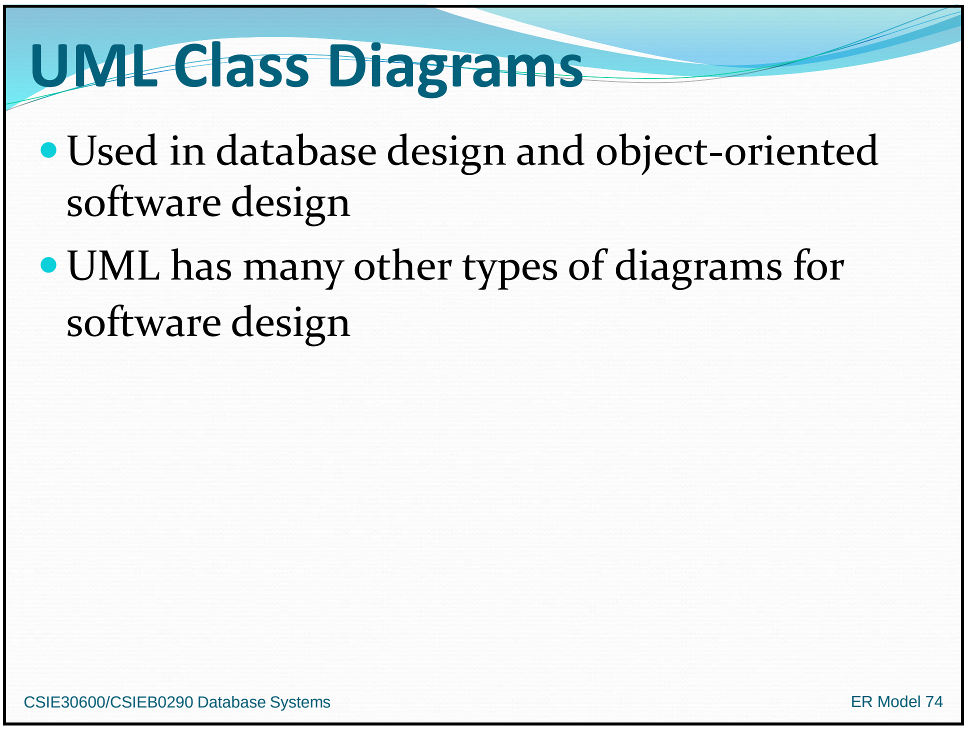
UML Class Diagrams

- Binary association
  - Represented as a line connecting participating classes
  - May optionally have a name
- Link attribute
  - Placed in a box connected to the association's line by a dashed lin
- **Multiplicities**: min..max, asterisk (*) indicates no maximum limit on participation



UML Class Diagrams

- Types of relationships: association and aggregation
- Distinguish between **unidirectional** and **bidirectional** associations
- Model weak entities using **qualified association**
- UML: used in database design and object-oriented software design
- UML has many other types of diagrams for software design



UML Class Diagrams

- Used in database design and object-oriented software design
- UML has many other types of diagrams for software design

COMPANY Database in UML

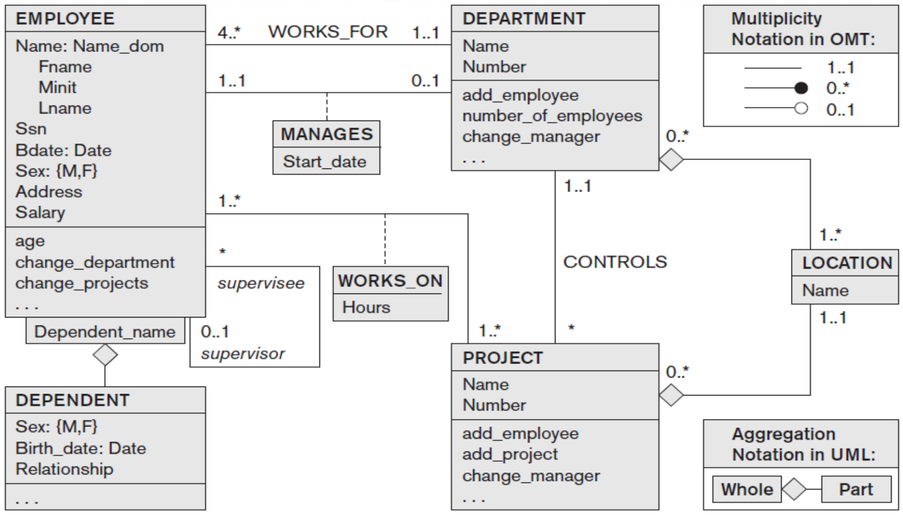
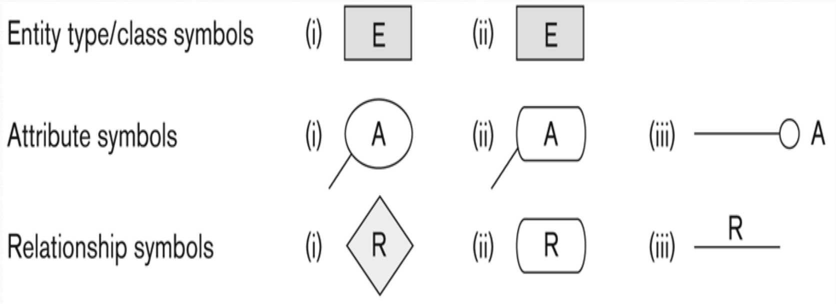
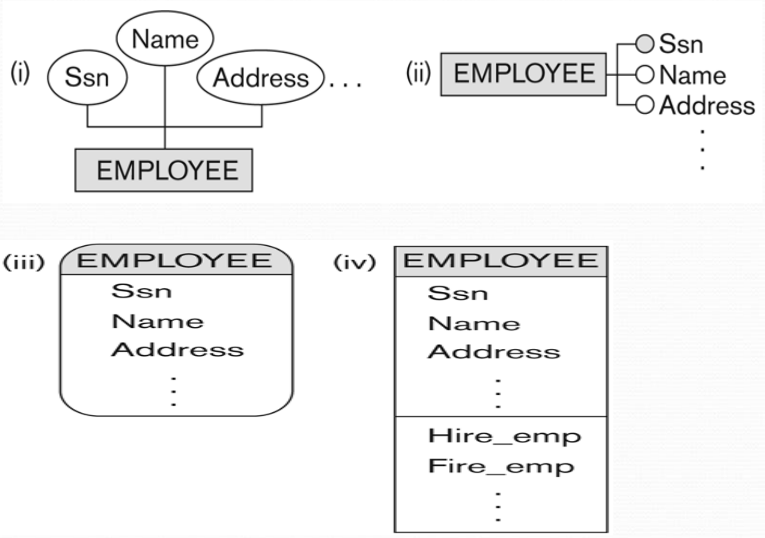


Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.

Other Alternative Diagrammatic Notations



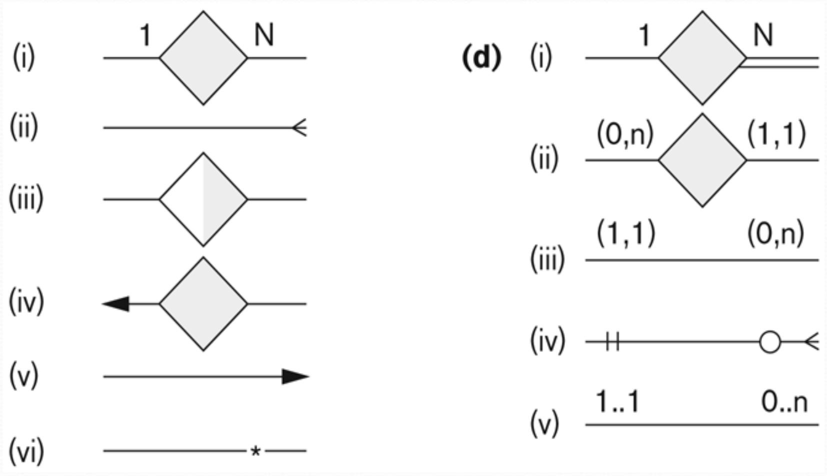
Other Alternative Diagrammatic Notations



CSIE30600/CSIEB0290 Database Systems

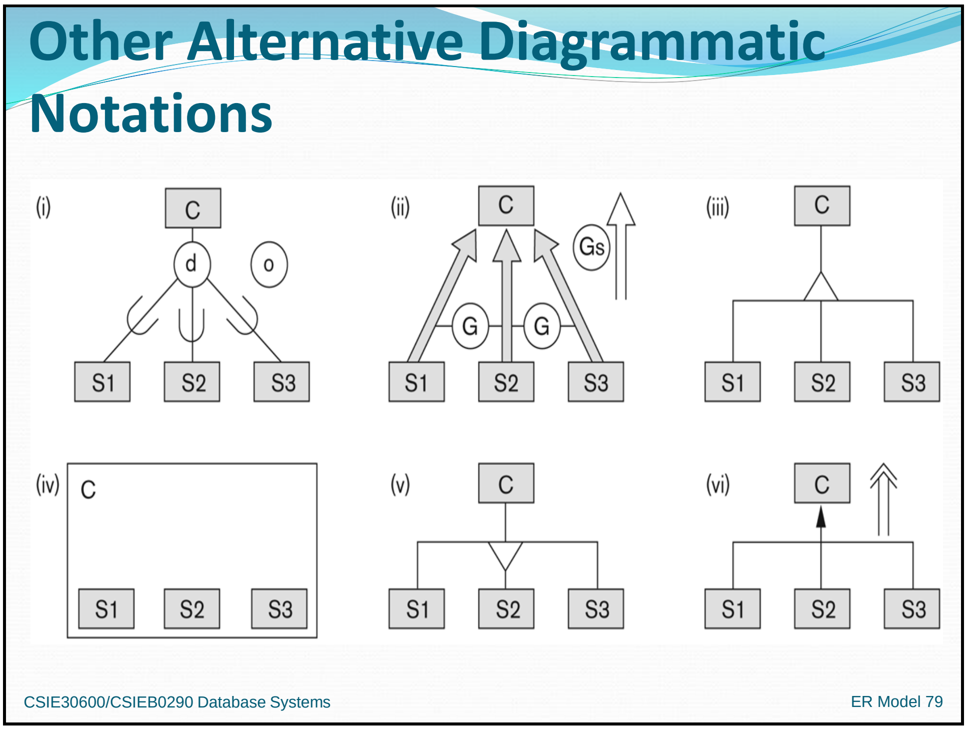
ER Model 77

Other Alternative Diagrammatic Notations

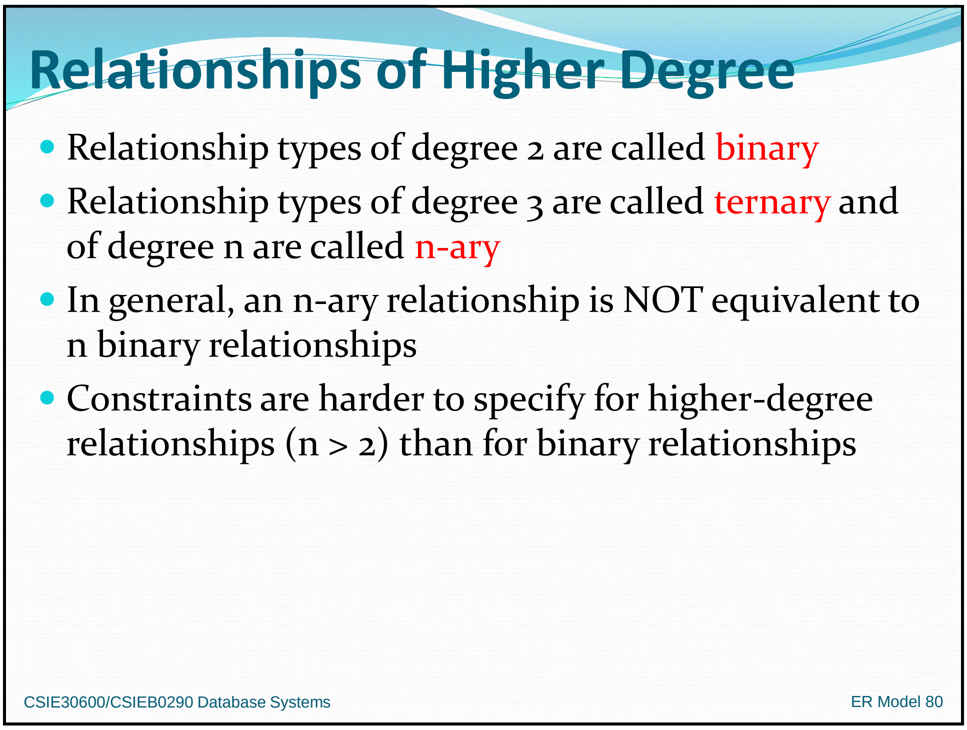


CSIE30600/CSIEB0290 Database Systems

ER Model 78



Other Alternative Diagrammatic Notations



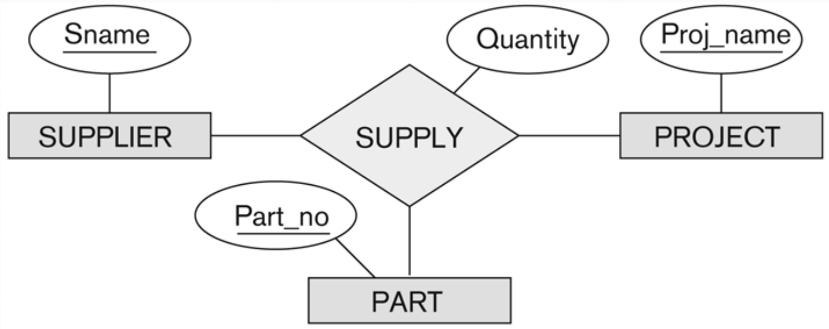
Relationships of Higher Degree

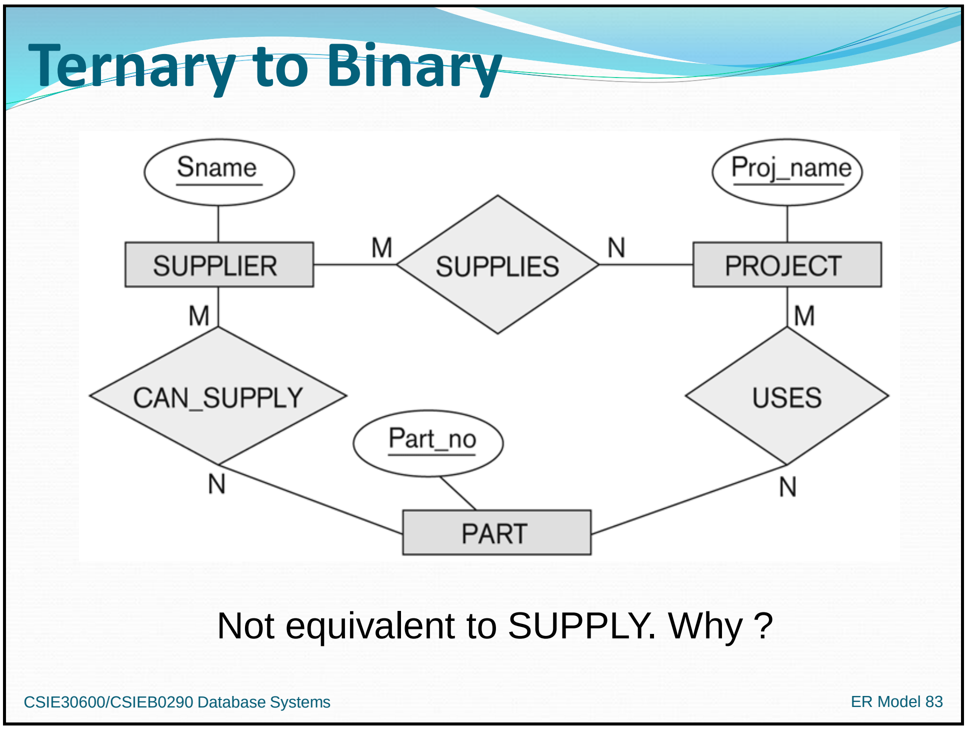
- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary and of degree n are called n-ary
- In general, an n-ary relationship is NOT equivalent to n binary relationships
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships

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 next 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)

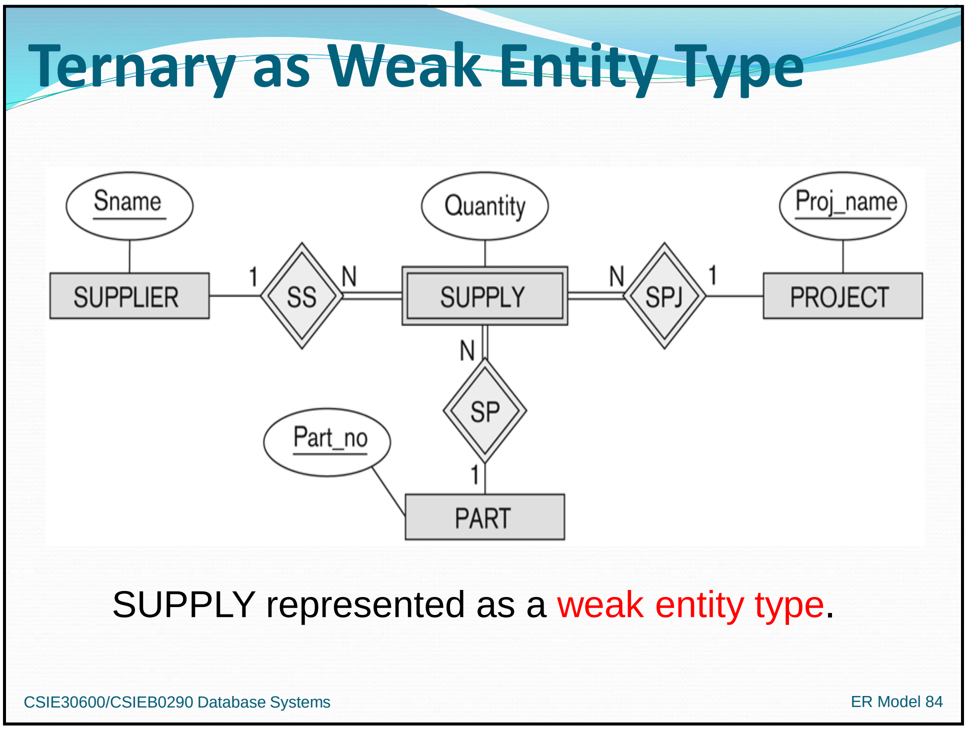
Example of a Ternary Relationship





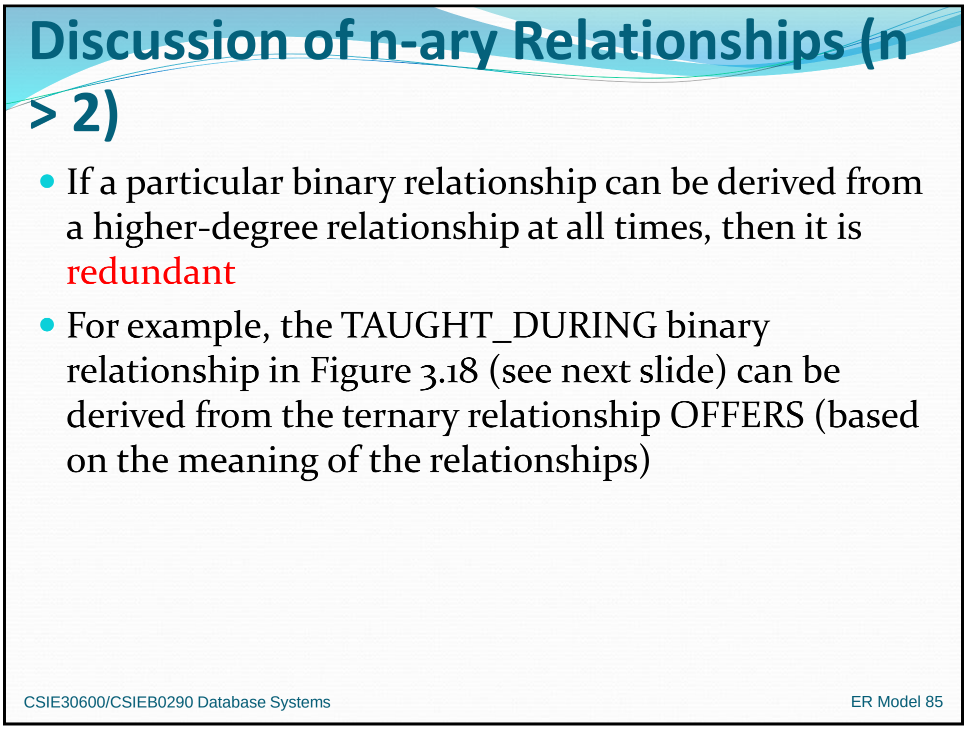
Ternary to Binary

Not equivalent to SUPPLY. Why ?



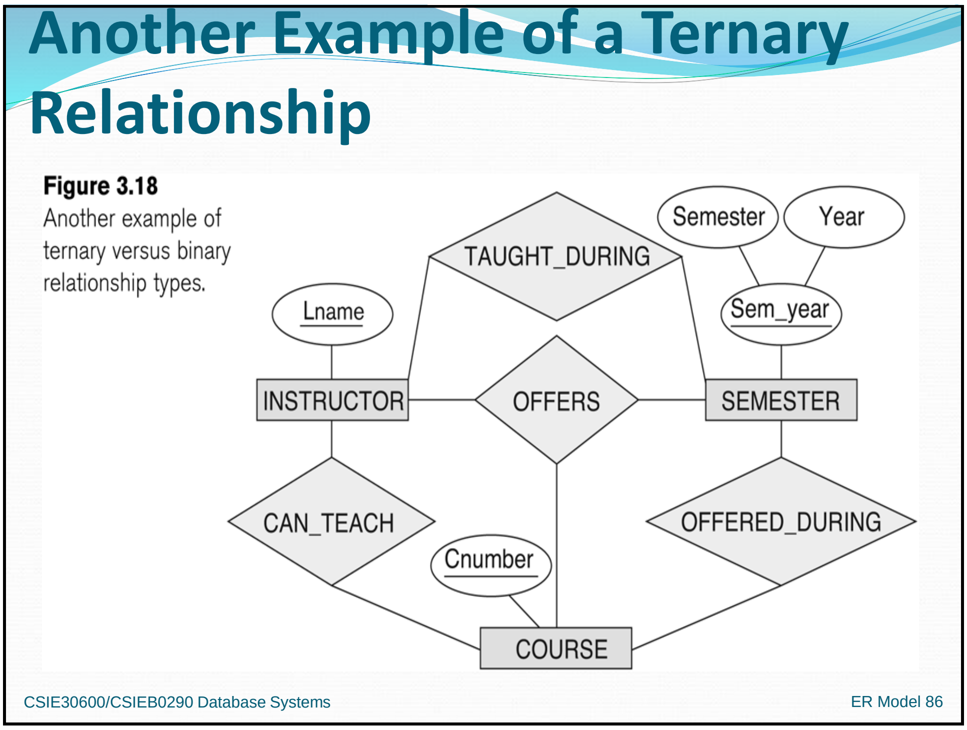
Ternary as Weak Entity Type

SUPPLY represented as a weak entity type.



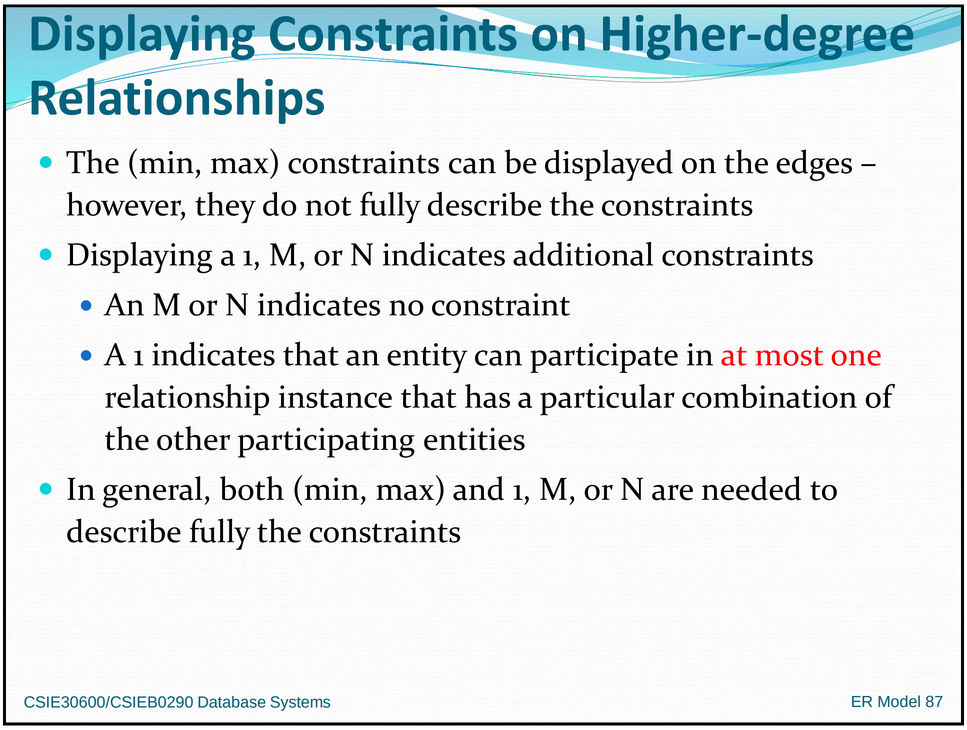
Discussion of n-ary Relationships (n > 2)

- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant
- For example, the TAUGHT_DURING binary relationship in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)



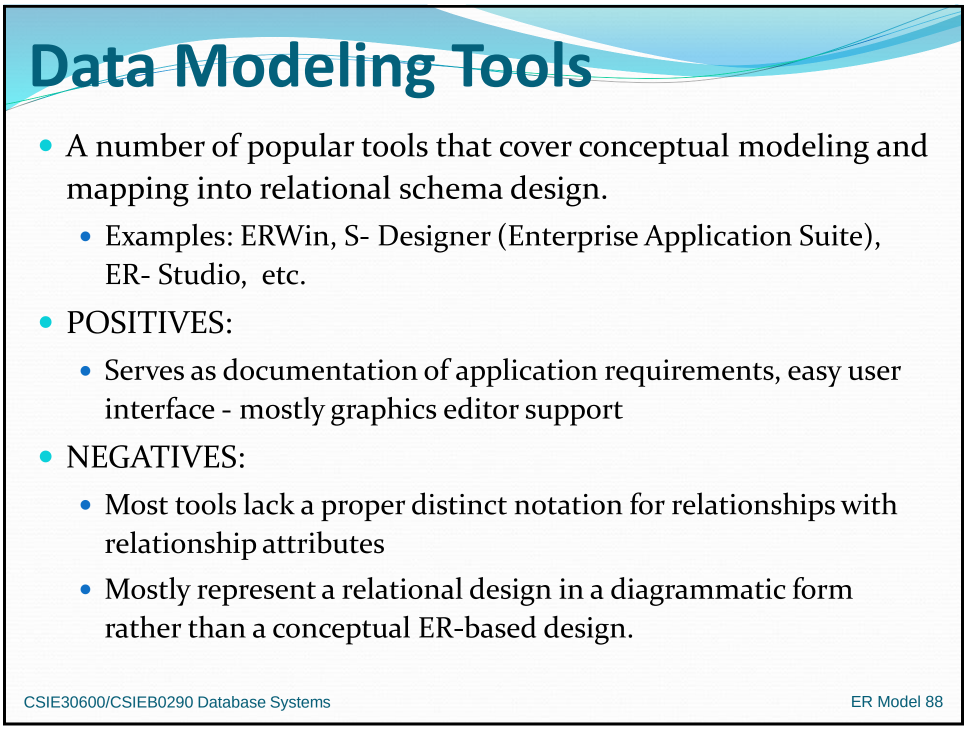
Another Example of a Ternary Relationship

Figure 3.18
Another example of ternary versus binary relationship types.



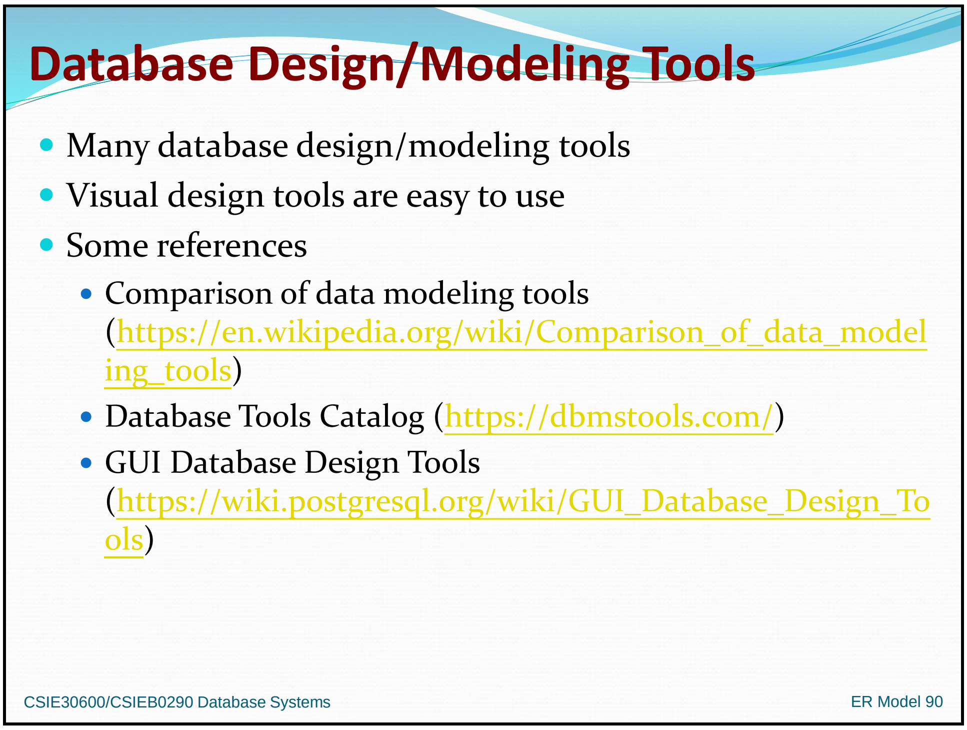
Displaying Constraints on Higher-degree Relationships

- The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints
- Displaying a 1, M, or N indicates additional constraints
  - An M or N indicates no constraint
  - A 1 indicates that an entity can participate in at most one relationship instance that has a particular combination of the other participating entities
- In general, both (min, max) and 1, M, or N are needed to describe fully the constraints



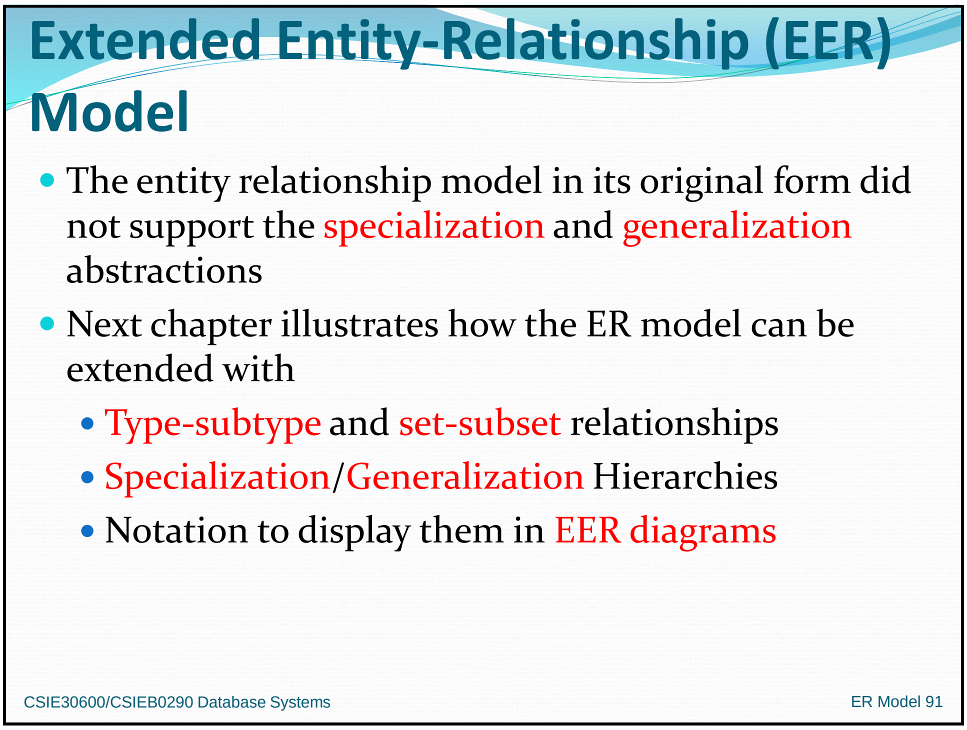
Data Modeling Tools

- A number of popular tools that cover conceptual modeling and mapping into relational schema design.
  - Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, etc.
- POSITIVES:
  - Serves as documentation of application requirements, easy user interface - mostly graphics editor support
- NEGATIVES:
  - Most tools lack a proper distinct notation for relationships with relationship attributes
  - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design.



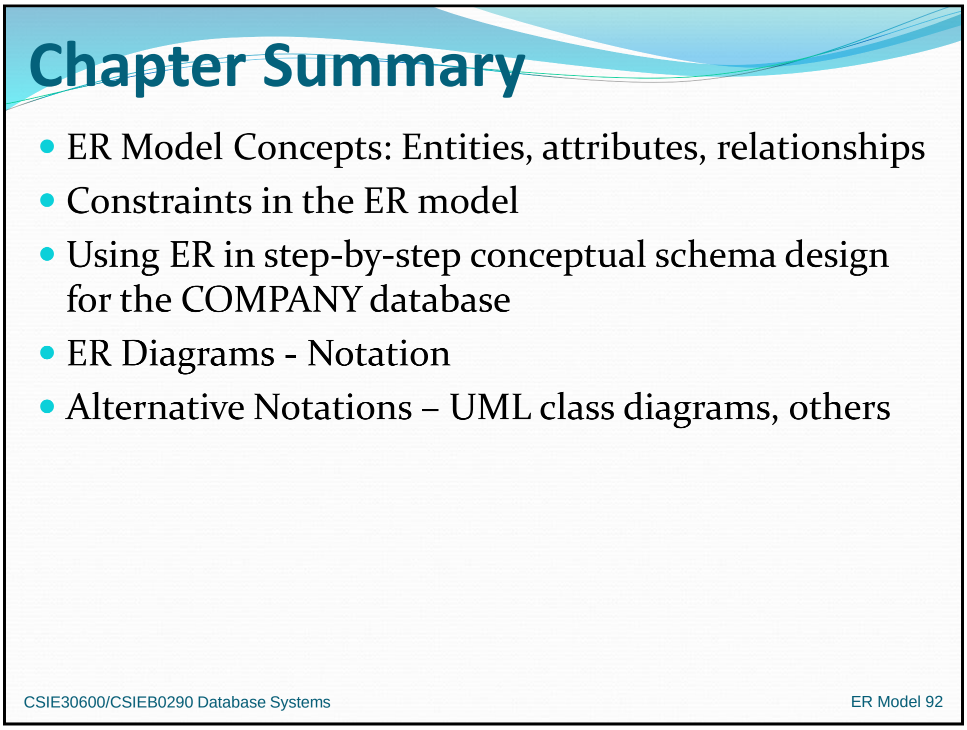
Database Design/Modeling Tools

- Many database design/modeling tools
- Visual design tools are easy to use
- Some references
  - Comparison of data modeling tools (https://en.wikipedia.org/wiki/Comparison_of_data_modeling_tools)
  - Database Tools Catalog (https://dbmstools.com/)
  - GUI Database Design Tools (https://wiki.postgresql.org/wiki/GUI_Database_Design_Tools)



Extended Entity-Relationship (EER) Model

- 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



Chapter Summary

- ER Model Concepts: Entities, attributes, relationships
- Constraints in the ER model
- Using ER in step-by-step conceptual schema design for the COMPANY database
- ER Diagrams - Notation
- Alternative Notations – UML class diagrams, others