# Database Management

COMP 3010E FALL 2025

LECTURE 4 RELATIONAL MODEL AND LOGICAL DESIGN

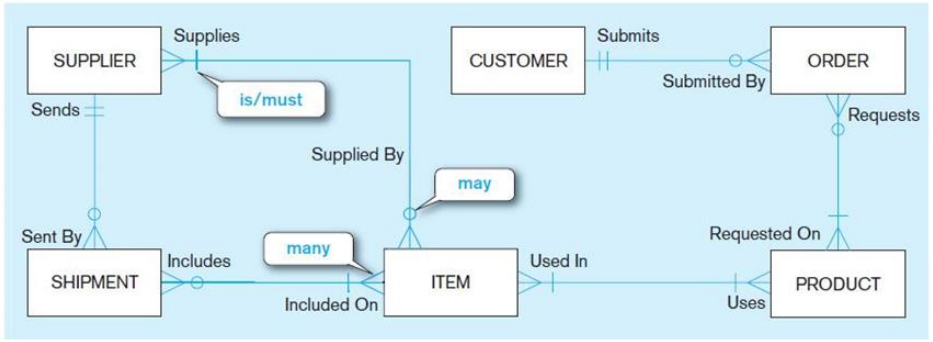
# Agenda

- Relational Model
- Logical Design: converting EERD to relational schema

# Steps of Data Modeling

## Conceptual Data Model

- Capture the nature of relationships among data/objects
- It is about understanding the business rules
- A symbolic model – easy to understand by human
- E-R/EER model




## Logical Data Model

- Transform the conceptual model into a logical representation
- It is about creating the database structure
- Logically well-defined and can be understood by computers
- Focus of data analysts

**CUSTOMER**

| Cust_ID | Cust_Name | Address |
|---|---|---|
| 101 | David Hills | … |
| 102 | Mary Lee | … |

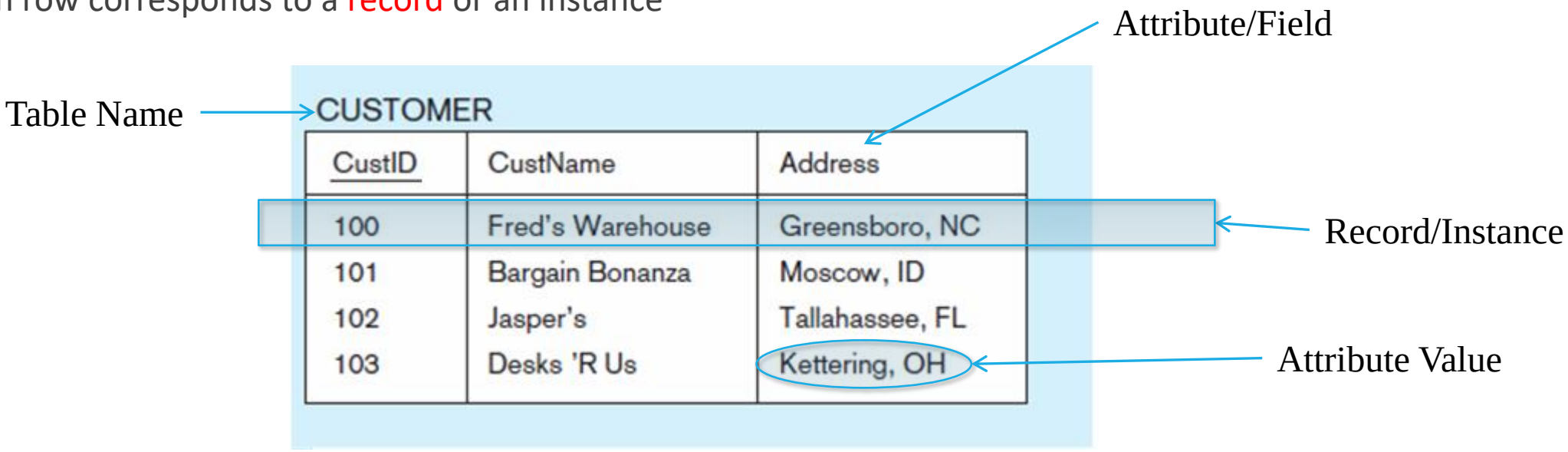
## Physical Data Model

- Translate the logical model into technical specifications for storing/retrieving data
- It is about storing data to achieve data processing efficiency and data quality
- Database users and data analysts usually do not worry about this step

Index and file organization on disk

# Logical Data Modeling

- **Relational Data Model**
  - Represent data in the form of relations (vs. object-oriented classes or XML tags in other models)
- **Relation**
  - A named, two-dimensional table of data
  - Each column represents an attribute or a field
  - Each row corresponds to a record or an instance



# Relational Databases

- **Relational database**
  - Represent data as a collection of tables/relations
  - Establish relationships by the means of common fields in related tables

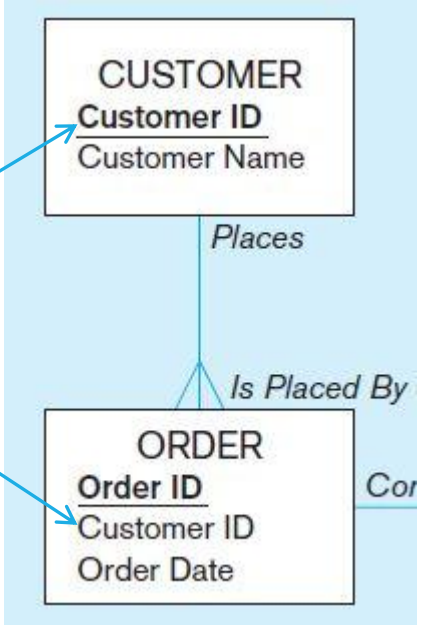
**CUSTOMER**

| Cust_ID | Cust_Name | Address |
|---|---|---|
| 101 | David Hills | … |
| 102 | Mary Lee | … |

**ORDER**

| Order_ID | Cust_ID | Order_date |
|---|---|---|
| 1100001 | 101 | 05/02/2012 |
| 1100002 | 102 | 05/03/2012 |

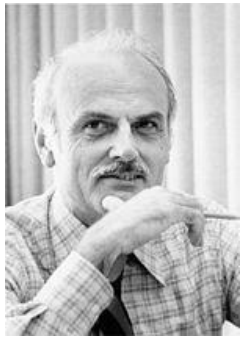
Common Field



# Relational Data Model

## Historical Background

- First introduced in 1970 by E. F. Codd, an IBM research fellow
- Has a solid theoretical foundation (set theory)
- Widespread commercial success in 1980s
- By far the most commonly used in various database applications



## Three Components

- **Data structure**
  - Represent data in the form of tables (with rows and columns)
- **Data manipulation**
  - Powerful operations (typically implemented using SQL)
- **Data integrity**
  - Mechanisms to enforce business rules that maintain the integrity of data

# Relation

## Definition

- A named, two-dimensional table of data
- Consist of a set of named columns (attributes/fields) and unnamed rows (records/instances)
- Typically contains information about one specific entity/relationship type

## Properties

- Unique table name
- Each entry is atomic (or single valued), i.e., no multivalued attributes
- Each column has a unique name
- Each row is unique (no duplicate rows)
- The sequence/order of columns (left to right) is insignificant
- The sequence/order of rows (top to bottom) is insignificant

**CERTIFICATE**

| EmpID | CourseTitle | DateCompleted |
|---|---|---|
| 100 | SPSS | 6/19/201X |
| 100 | Surveys | 10/7/201X |
| 140 | Tax Acc | 12/8/201X |
| 110 | Visual Basic | 1/12/201X |
| 110 | C++ | 4/22/201X |
| 150 | SPSS | 6/19/201X |
| 150 | Java | 8/12/201X |

# Multivalued Attributes

- Remove multivalued attributes from tables

EMPLOYEE1

| EmpID | Name | DeptName | Salary | CourseTitle | DateCompleted |
|---|---|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 | SPSS<br>Surveys | 6/19/201X<br>10/7/201X |
| 140 | | | | | |
| 110 | | | | | |
| 190 | | | | | |
| 150 | | | | | |

EMPLOYEE2

| EmpID | Name | DeptName | Salary | CourseTitle | DateCompleted |
|---|---|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 | SPSS | 6/19/201X |
| 100 | Margaret Simpson | Marketing | 48,000 | Surveys | 10/7/201X |
| 140 | Alan Beeton | Accounting | 52,000 | Tax Acc | 12/8/201X |
| 110 | Chris Lucero | Info Systems | 43,000 | Visual Basic | 1/12/201X |
| 110 | Chris Lucero | Info Systems | 43,000 | C++ | 4/22/201X |
| 190 | Lorenzo Davis | Finance | 55,000 | | |
| 150 | Susan Martin | Marketing | 42,000 | SPSS | 6/19/201X |
| 150 | Susan Martin | Marketing | 42,000 | Java | 8/12/201X |

# Relational Keys (1)

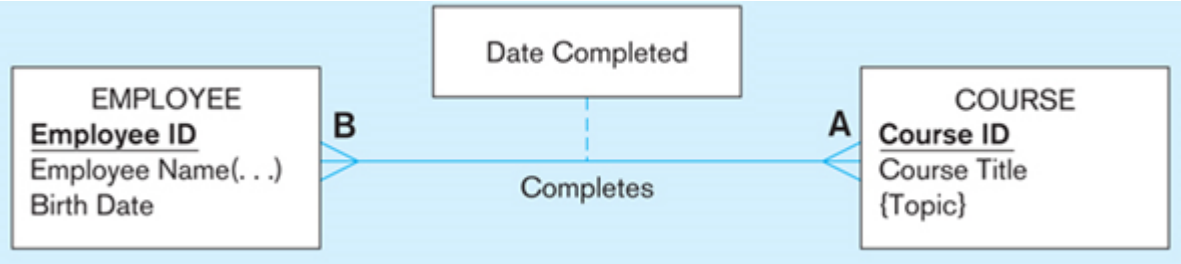
- **Primary Key (PK)**
  - An attribute or a combination of attributes used to uniquely identifies each row
  - Contains no redundant information.
  - Each relation should have a designated primary key
  - May be the same as an entity's identifier, but there are exceptions
    e.g., associative entities, weak entities, and relationships
- Candidate Keys: all the keys that can serve as the PK.

EMPLOYEE1

| EmpID | Name | DeptName | Salary |
|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 |
| 140 | Allen Beeton | Accounting | 52,000 |
| 110 | Chris Lucero | Info Systems | 43,000 |
| 190 | Lorenzo Davis | Finance | 55,000 |
| 150 | Susan Martin | Marketing | 42,000 |

# Relational Keys (2)

- **Composite key**
  - A primary key
  - Consist of more than one attribute
  - Commonly seen in relations of associative entities, weak entities, and relationships




**Completes**

| EmpID | CourseTitle | DateCompleted |
|---|---|---|
| 100 | SPSS | 6/19/201X |
| 100 | Surveys | 10/7/201X |
| 140 | Tax Acc | 12/8/201X |
| 110 | Visual Basic | 1/12/201X |
| 110 | C++ | 4/22/201X |
| 150 | SPSS | 6/19/201X |
| 150 | Java | 8/12/201X |

# Relational Keys (3)

- **Foreign Key (FK)**
  - An attribute (possibly composite) in a relation
  - Serve as the primary key of another relation
  - Allow us to associate various relations/tables
  - Provide an important mechanism to deal with relationships and enforce referential integrity

**ORDER**

| OrderID | CustID | OrderDate |
|---|---|---|
| 1100001 | 101 | 05/02/2012 |
| 1100002 | 102 | 05/03/2012 |

**Foreign Key column**

**CUSTOMER**

| CustID | CustName | Address |
|---|---|---|
| 101 | David Hills | … |
| 102 | Mary Lee | … |

# Relational Keys (4)

- **SuperKey**
  - An attribute or a combination of attributes used to uniquely identifies each row
  - May contains no redundant information
  - The combination of all the columns is always a superkey (but not necessarily and candidate key)

EMPLOYEE1

| EmpID | Name | DeptName | Salary |
|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 |
| 140 | Allen Beeton | Accounting | 52,000 |
| 110 | Chris Lucero | Info Systems | 43,000 |
| 190 | Lorenzo Davis | Finance | 55,000 |
| 150 | Susan Martin | Marketing | 42,000 |

# Schema (1)

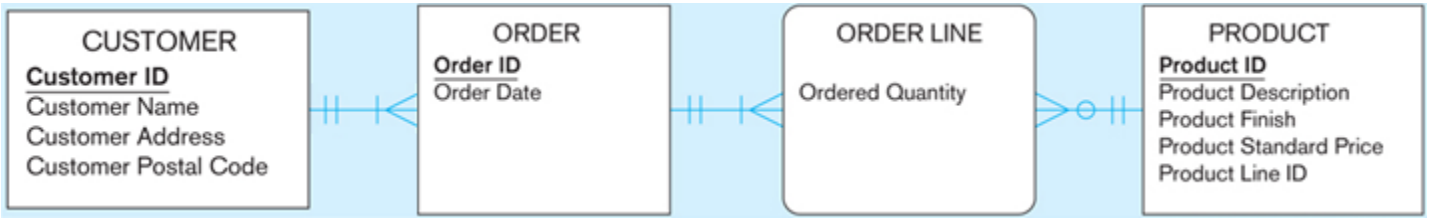
- **Definition**
  - A description of the overall logical structure of the database
  - A relational schema consists of a set of relations

- **Two Common Ways**
  - Short text statements
    - Each relation is represented by its name and followed by the names of its attributes in parentheses
    - Advantage: simple
  - A graphical representation
    - Each relation is represented by a name and a rectangle containing the attributes
    - Advantage: better expressing referential integrity constrains

# Schema (2)

## Short Text Statements

- Format: RELATION1(Attribute1, Attribute2,…)
- The primary key is underlined
- The foreign key is indicated by a dashed underline




**Example**: a schema for four relations --

CUSTOMER(CustomerID, CustomerName, CustomerAddress, CustomerPostalCode)

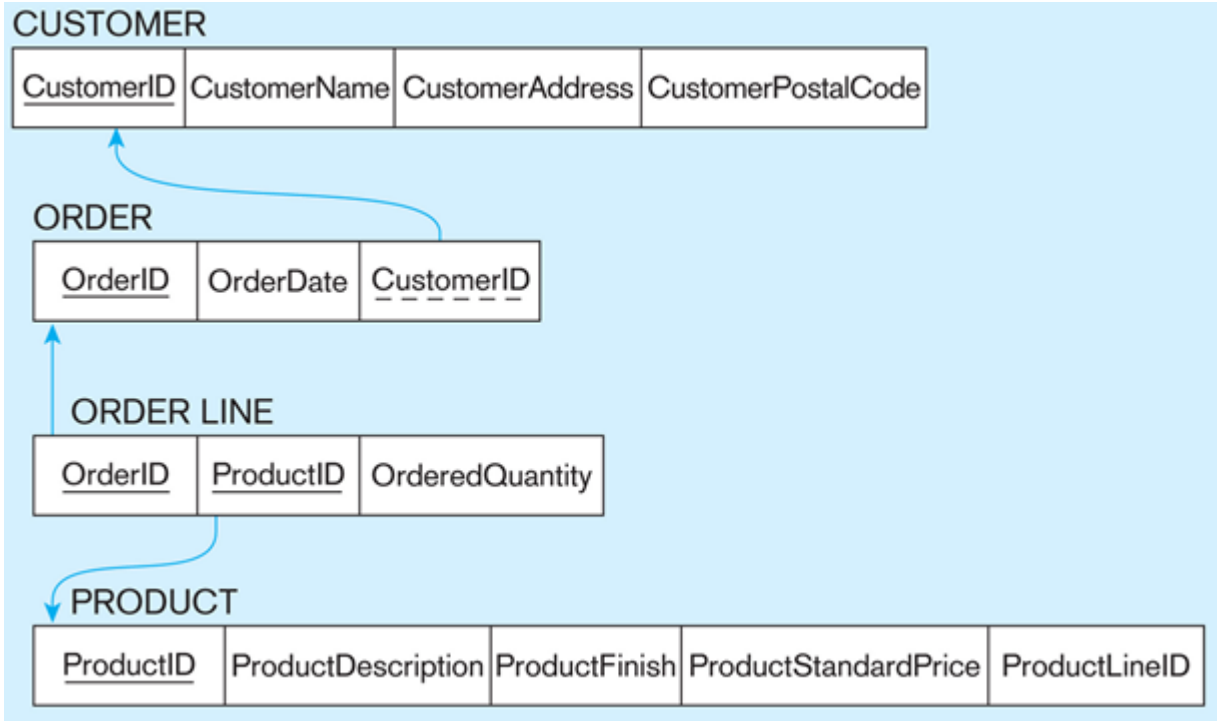
ORDER(OrderID, OrderDate, CustomerID)

ORDER LINE(OrderID, ProductID, OrderQuantity)

PRODUCT(ProductID, ProductDescription, ProductFinish, ProductStandardPrice, ProductLineID)

# Schema (3)

- **Graphical Representation**
  - Format: RelationName + rectangle
  - The primary key is underlined
  - The foreign key is indicated by a dashed underline
  - Draw an arrow from each FK to the associated PK



# Integrity Constraints

- Constraints or rules that limit acceptable values and actions, ensuring data integrity
- Major Types
  - Domain Constraints
  - Entity Integrity
  - Referential Integrity

# Domain Constraints

- All of the values that appear in a column must be from the *same domain*
- Domain definition: domain name, description, data type, size, etc.

**TABLE 4-1 Domain Definitions for INVOICE Attributes**

| Attribute | Domain Name | Description | Domain |
|---|---|---|---|
| CustomerID | Customer IDs | Set of all possible customer IDs | character: size 5 |
| CustomerName | Customer Names | Set of all possible customer names | character: size 25 |
| CustomerAddress | Customer Addresses | Set of all possible customer addresses | character: size 30 |
| OrderID | Order IDs | Set of all possible order IDs | character: size 5 |
| OrderDate | Order Dates | Set of all possible order dates | date: format mm/dd/yy |
| ProductID | Product IDs | Set of all possible product IDs | character: size 5 |
| ProductDescription | Product Descriptions | Set of all possible product descriptions | character: size 25 |
| ProductFinish | Product Finishes | Set of all possible product finishes | character: size 15 |
| ProductStandardPrice | Unit Prices | Set of all possible unit prices | monetary: 6 digits |
| ProductLineID | Product Line IDs | Set of all possible product line IDs | integer: 3 digits |
| OrderedQuantity | Quantities | Set of all possible ordered quantities | integer: 3 digits |

# Domain Constraints

- All of the values that appear in a column must be from the *same domain*
- Domain definition: domain name, description, data type, size, etc.

- Domain of CustomerID: {four digit numbers starting from 1001}
- Domain of Name: {any text strings}
- Domain of OrderDate: {all possible dates in yyyy-mm-dd format}
- ….

- A relation is a subset of the cross-product (the set of all possible combinations) of all the domains

# Entity Integrity

- Ensure every relation has a primary key with valid data values
- No primary key attribute may be null
- What is a null?
  - A value that may be assigned to an attribute with no or unknown applicable value
  - It is not a numeric “0”
  - It is not an empty string “ ”
  - A special marker indicates the absence of a data value

**Entity integrity rule:**

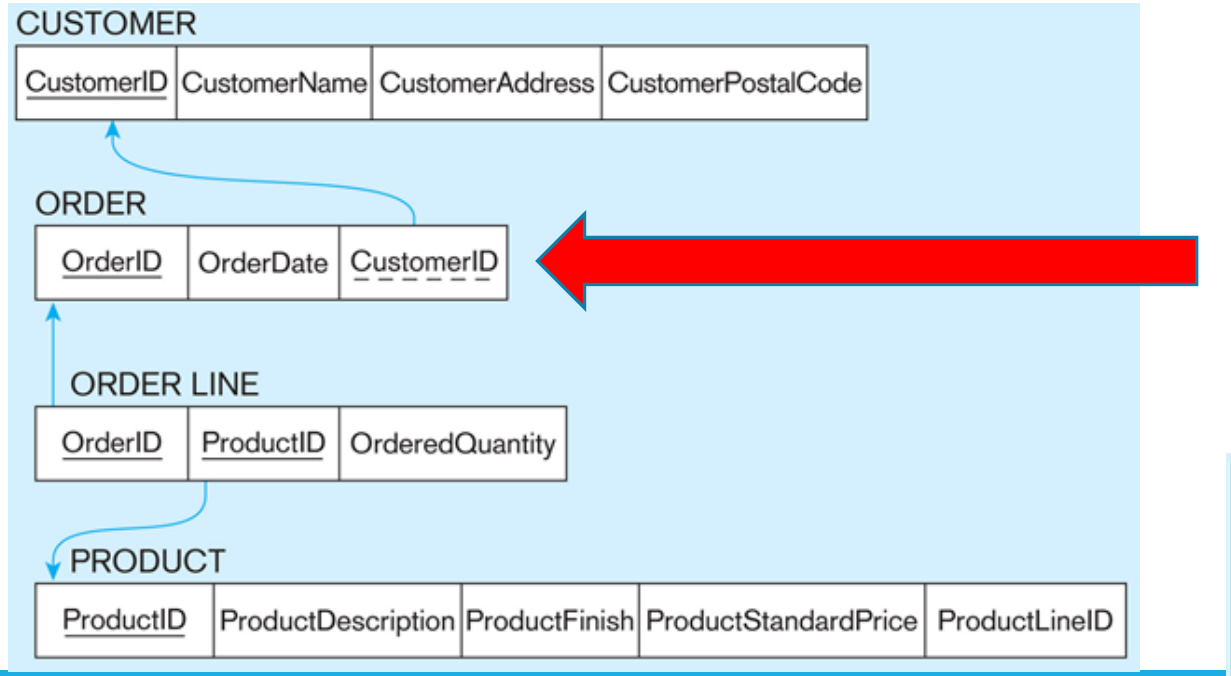
**every primary key attribute must be non-null**

DEPARTMENT

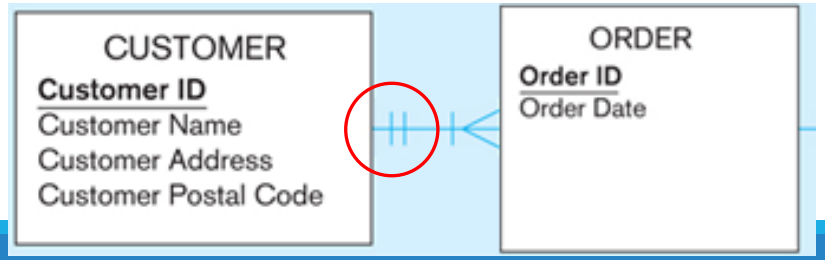
| DeptName | Location | Fax |
|---|---|---|
| Sales | A building | 3011912 |
| Marketing | B building | NULL |
| Research | C building | 3100988 |

# Referential Integrity

- **Referential integrity constraint**
  - Each foreign key value must either match a primary key value in another relation or be null
  - Maintain data consistency among the rows of two associated relations
  - Draw an arrow from each foreign key to the associated primary key

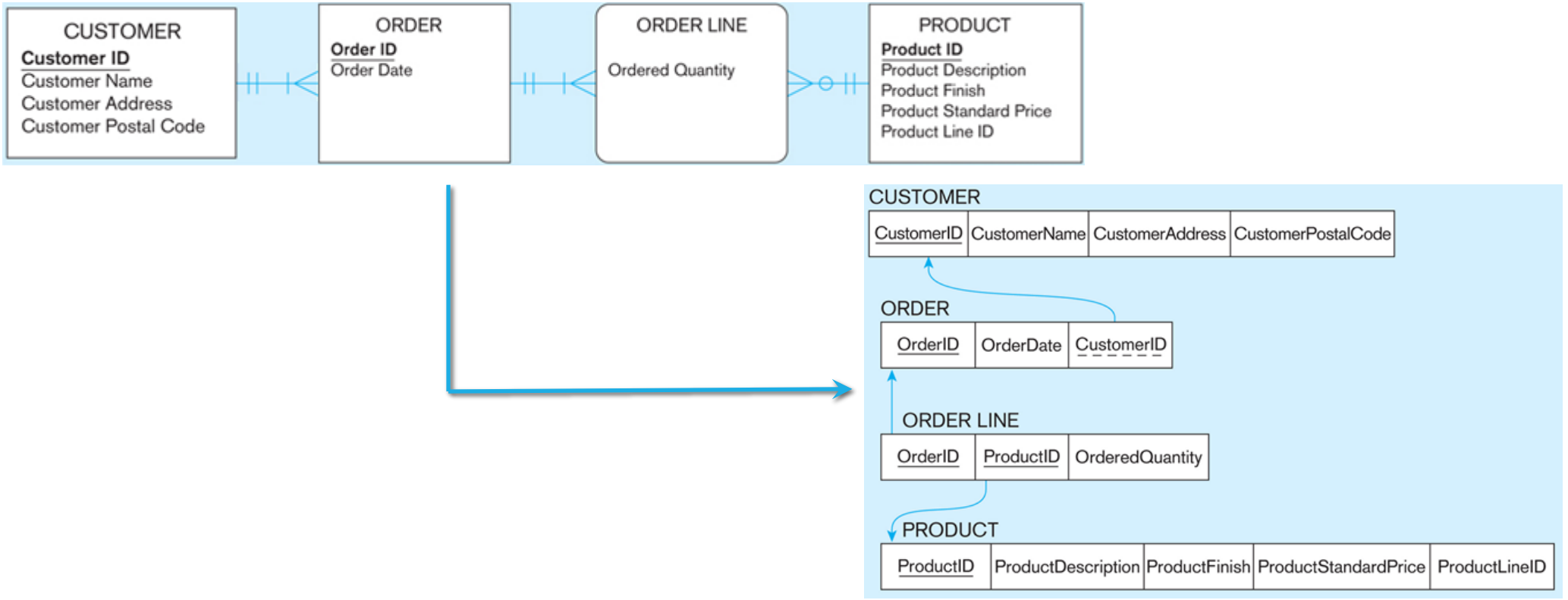



**Note: if the relationship is mandatory, the foreign key cannot be null, e.g. CustomerID in ORDER relation if each order must be placed by a customer.**



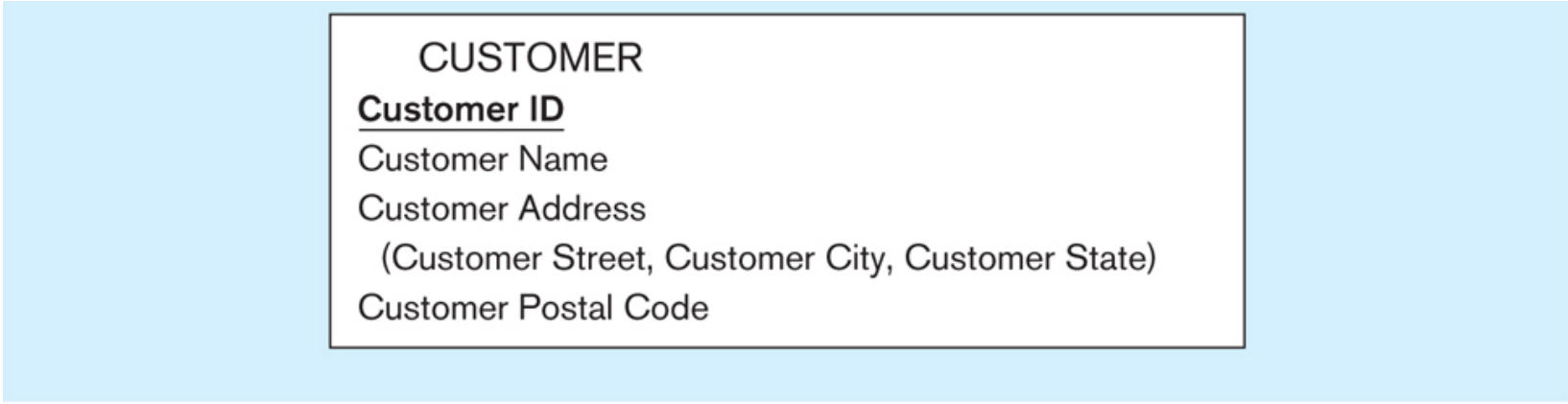
# Transform ERD into Relations

- Input: entity-relationship diagrams (ERD) or EER diagrams
- Output: relational schemas



# Map Regular Entities (1)

- Entity-type name → relation name
- Simple attribute → relation attribute
- Composite attribute → simple-component attributes (improve data accessibility)
- Identifier attribute → primary key

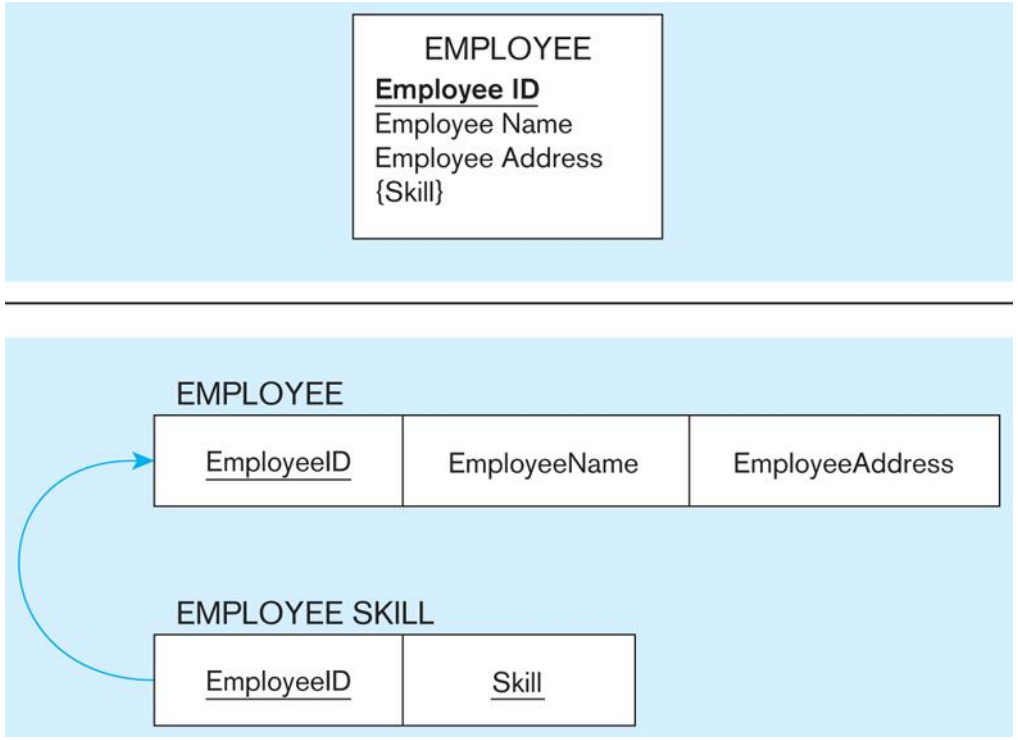



CUSTOMER

| CustomerID | CustomerName | CustomerStreet | CustomerCity | CustomerState | CustomerPostalCode |
|---|---|---|---|---|---|

# Map Regular Entities (2)

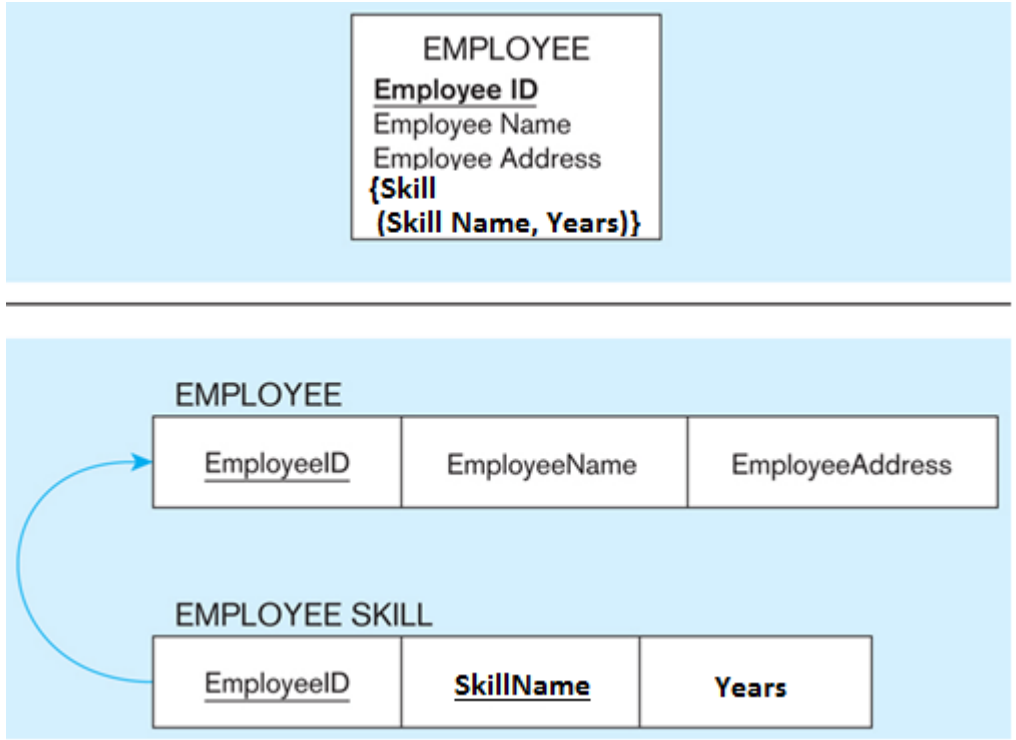
- Multivalued attribute → two relations
  - The 1st contains all attributes except the multivalued attribute
  - The 2nd contains two attributes: the PK from the 1st relation + the multivalued attribute




| EmployeeID | Skill |
|---|---|
| E001 | C++ |
| E002 | Visual Basic |
| E002 | R |
| E003 | SQL |

# Map Regular Entities (3)

- Multivalued composite attribute → two relations with additional attributes

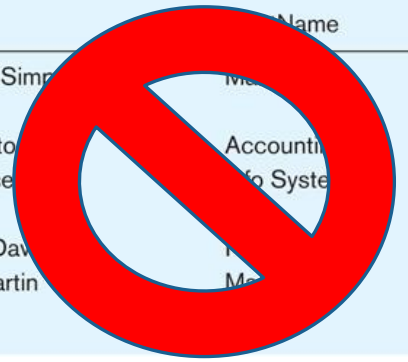


# Multi-valued Attributes

- When you have multi-valued attributes in a table and you want to convert the table to relation …

**(a) Table with repeating groups**

| EmpID | Name | Name | Salary | CourseTitle | DateCompleted |
|---|---|---|---|---|---|
| 100 | Margaret Simp | Ma | 48,000 | SPSS | 6/19/201X |
| | | | | Surveys | 10/7/201X |
| 140 | Alan Beeto | Accounti | 52,000 | Tax Acc | 12/8/201X |
| 110 | Chris Luce | fo Syste | | | |
| 190 | Lorenzo Dav | F | | | |
| 150 | Susan Martin | Ma | | | |



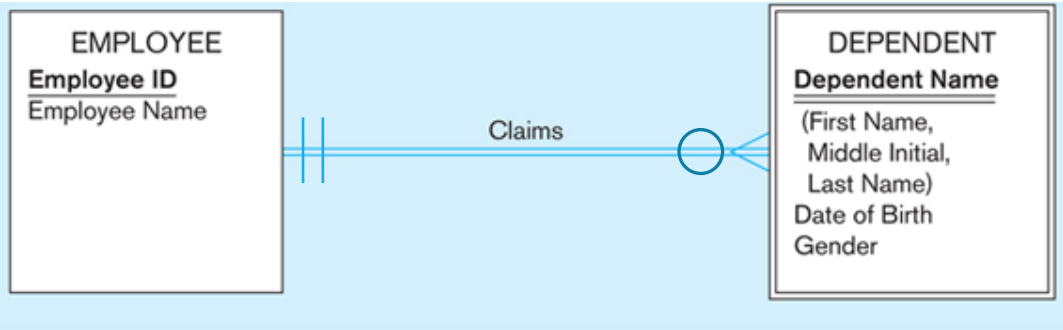
**(b) EMPLOYEE2 relation**

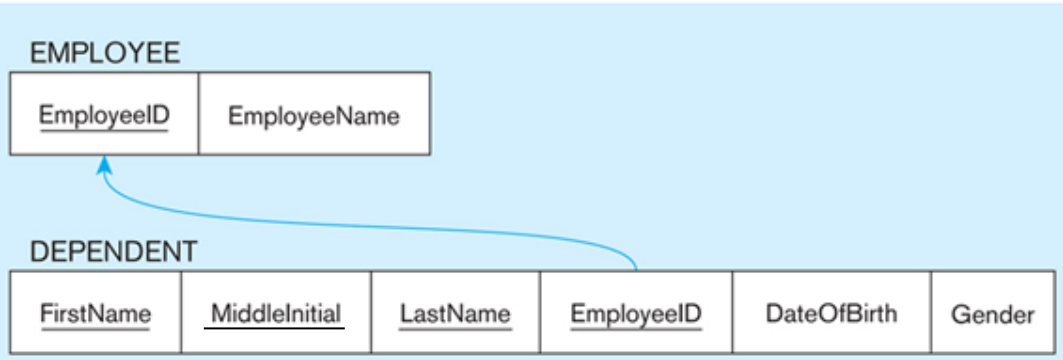
EMPLOYEE2

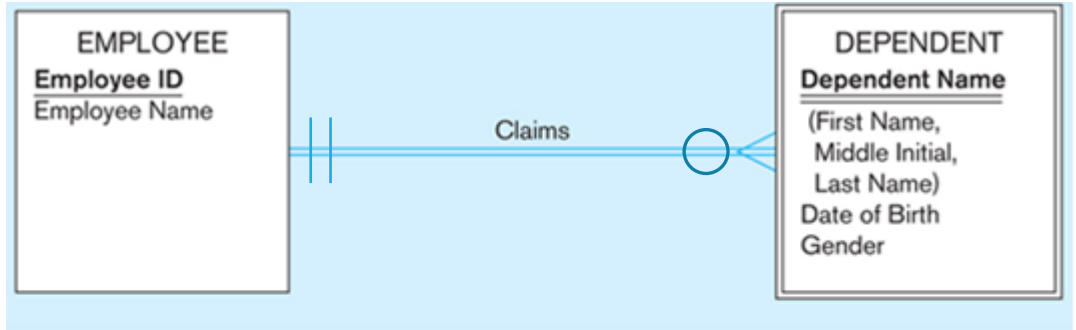
| EmpID | Name | DeptName | Salary | CourseTitle | DateCompleted |
|---|---|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 | SPSS | 6/19/201X |
| 100 | Margaret Simpson | Marketing | 48,000 | Surveys | 10/7/201X |
| 140 | Alan Beeton | Accounting | 52,000 | Tax Acc | 12/8/201X |
| 110 | Chris Lucero | Info Systems | 43,000 | Visual Basic | 1/12/201X |
| 110 | Chris Lucero | Info Systems | 43,000 | C++ | 4/22/201X |
| 190 | Lorenzo Davis | Finance | 55,000 | | |
| 150 | Susan Martin | Marketing | 42,000 | SPSS | 6/19/201X |
| 150 | Susan Martin | Marketing | 42,000 | Java | 8/12/201X |

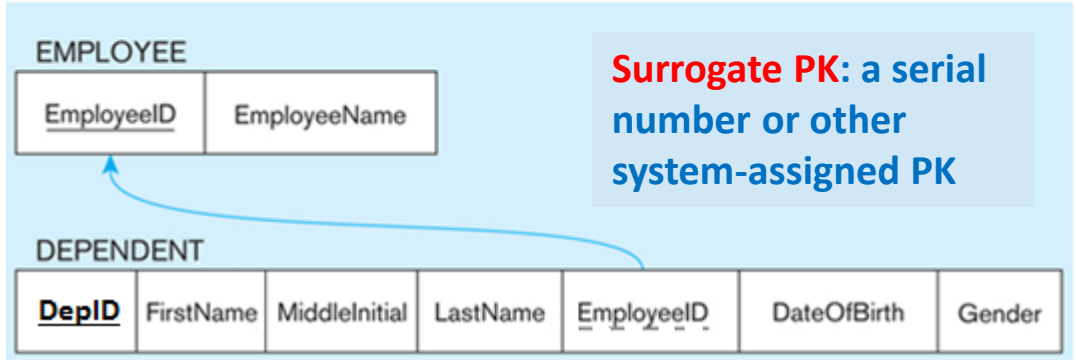
# Map Weak Entities

- Weak entity → a relation
  - Include the PK of the identifying relation as a FK attribute
  - PK: the partial identifier of the weak entity + the PK of the identifying relation



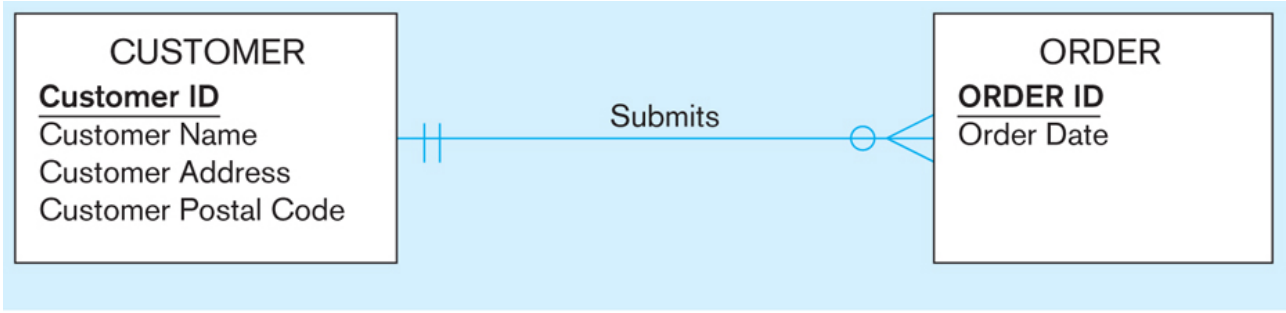



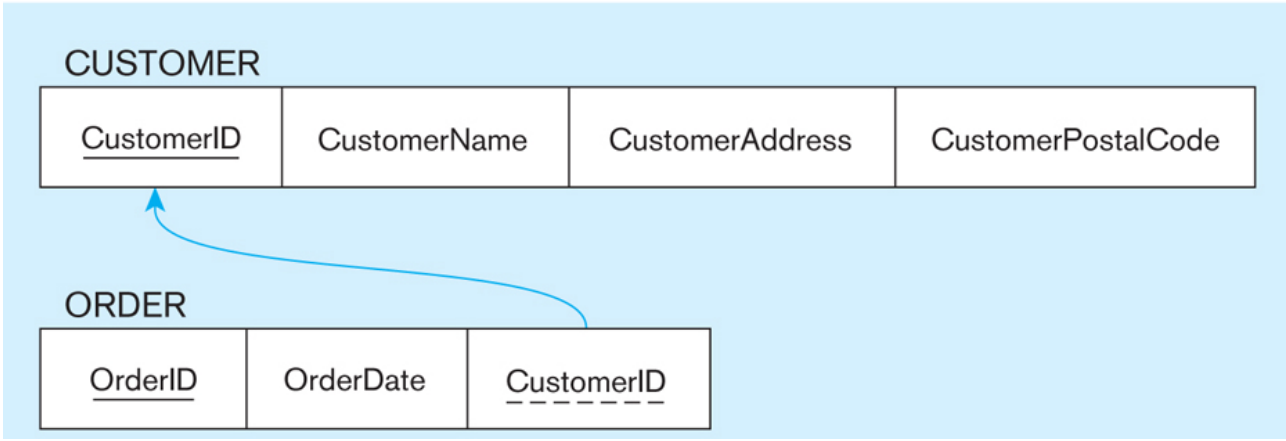



# Map Binary 1:M Relationship

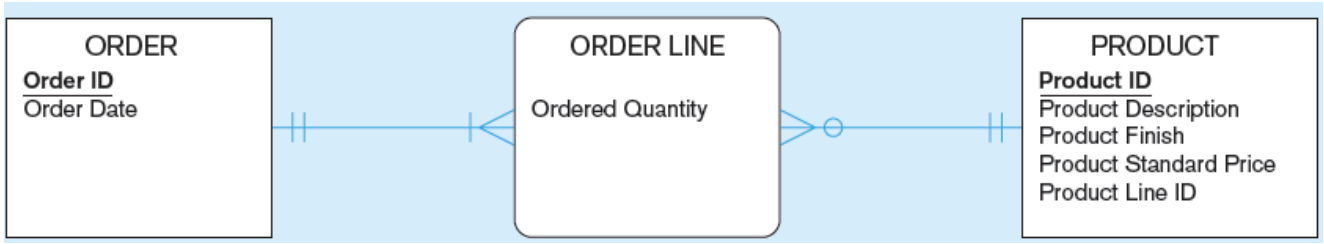
- Two entity types → two relations
  - Include the PK attribute(s) of the one-side entity/relation as a FK in the many-side entity/relation

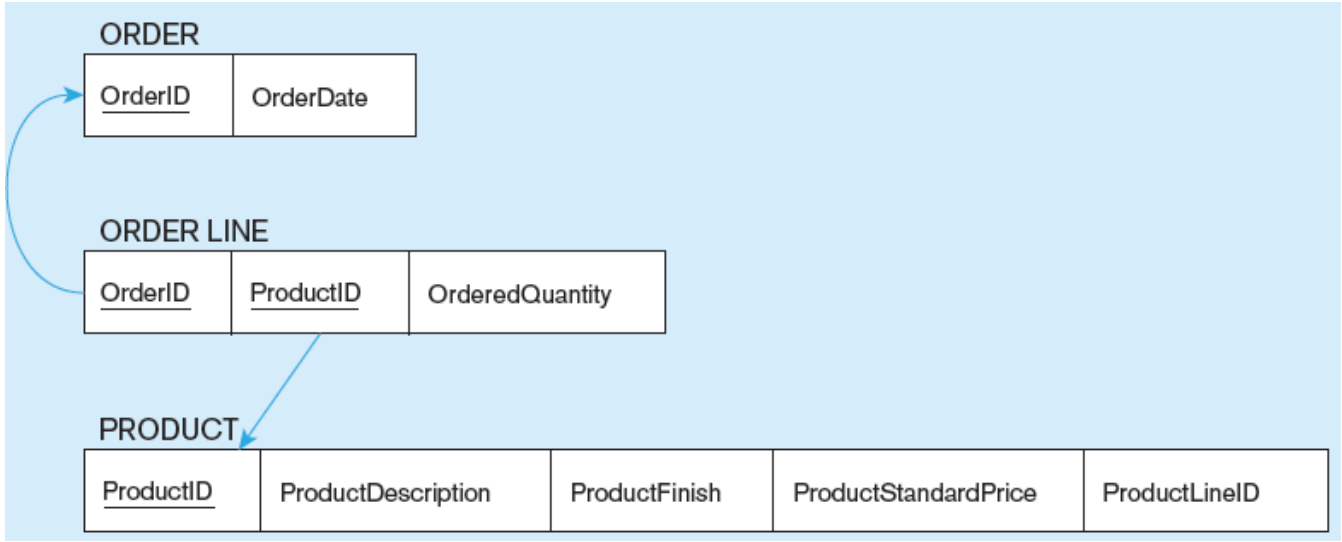


# Mapping Associative Entities

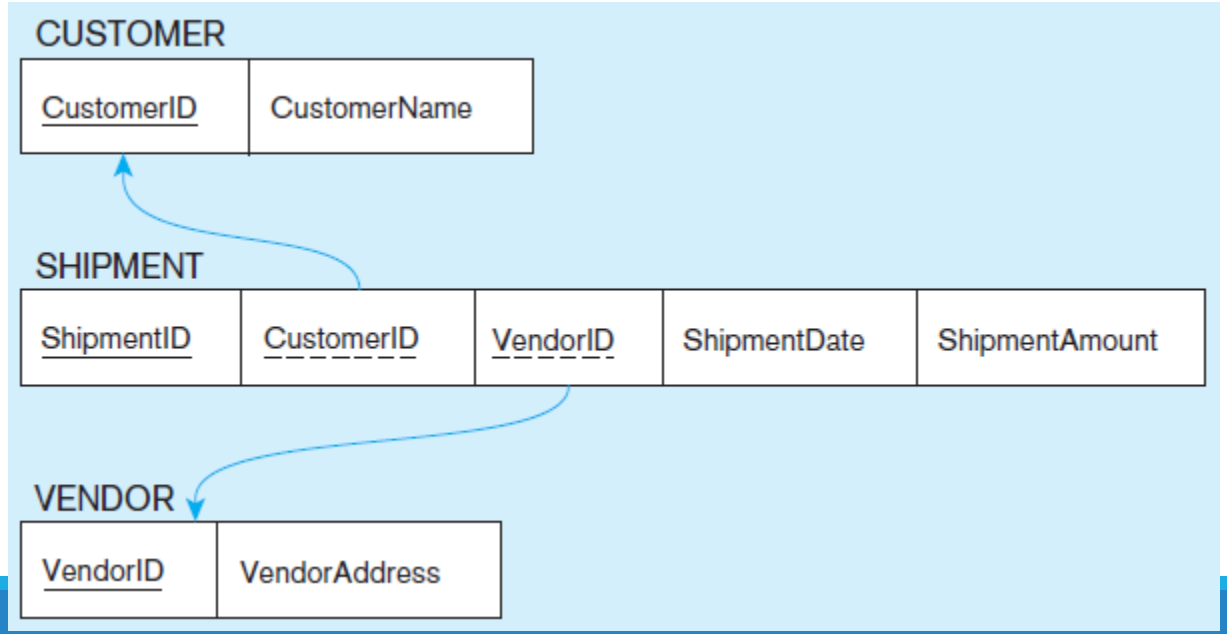
When no new keys are needed



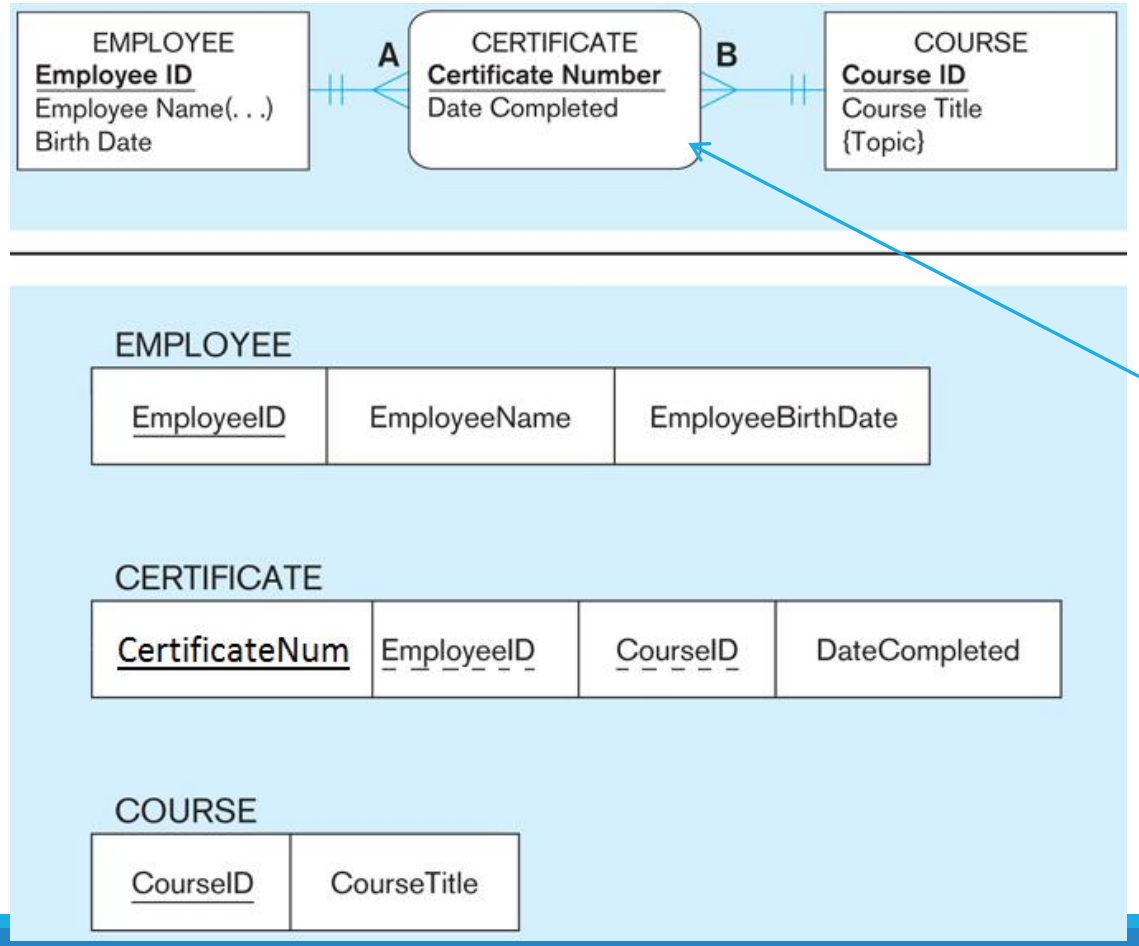



# Mapping Associative Entities (cont.)

When new keys are needed

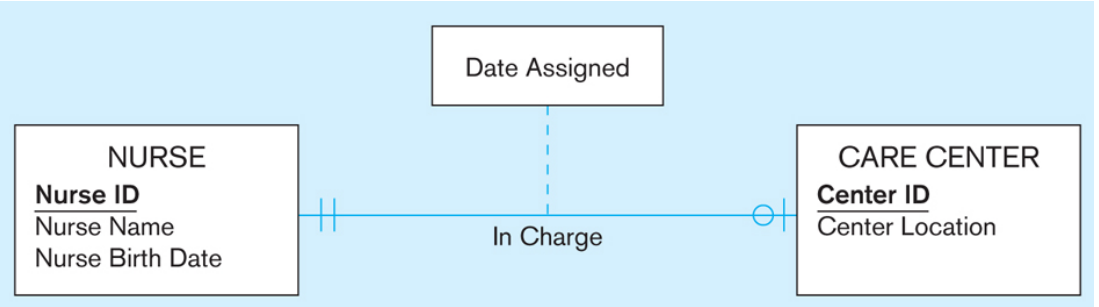
# Mapping Associative Entities (cont.)

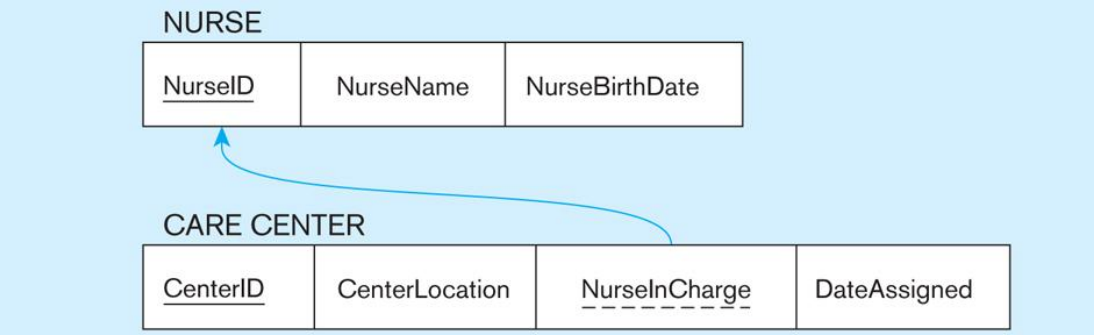



**Surrogate PK: a serial number or other system-assigned PK**

# Map Binary 1:1 Relationship

- Two entity types → two relations
  - Include the PK attribute(s) of the mandatory-side relation as a FK in the optional-side relation
  - The relationship attributes are also included in the optional-side relation



# Answers

Player

Not NULL

| **PlayerID** | **Name** | **DOB** | **Position** | **TeamID** |
|---|---|---|---|---|

Team

Not NULL

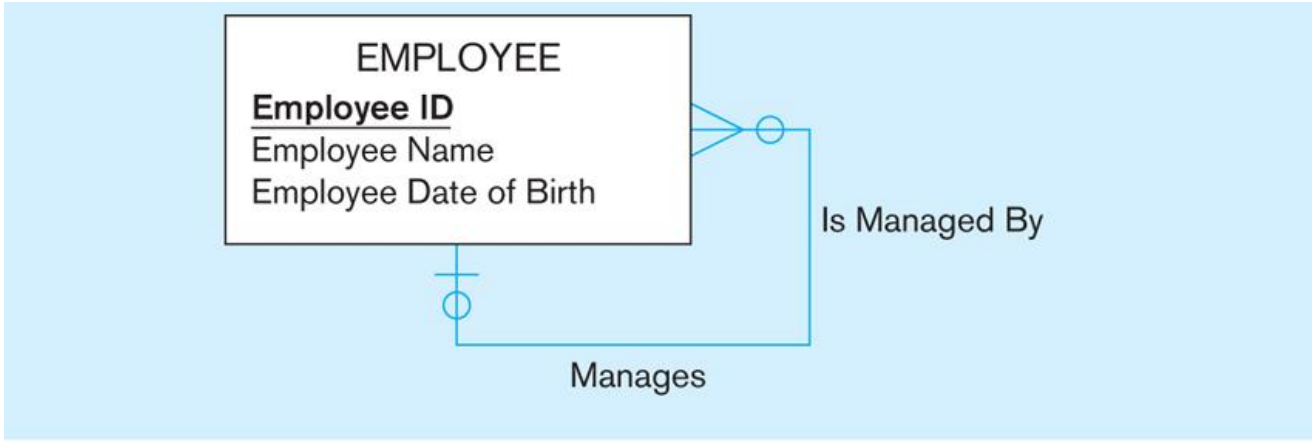
| **TeamID** | **Name** | **Home** | **Year_Founded** | **HeadCoach** |
|---|---|---|---|---|

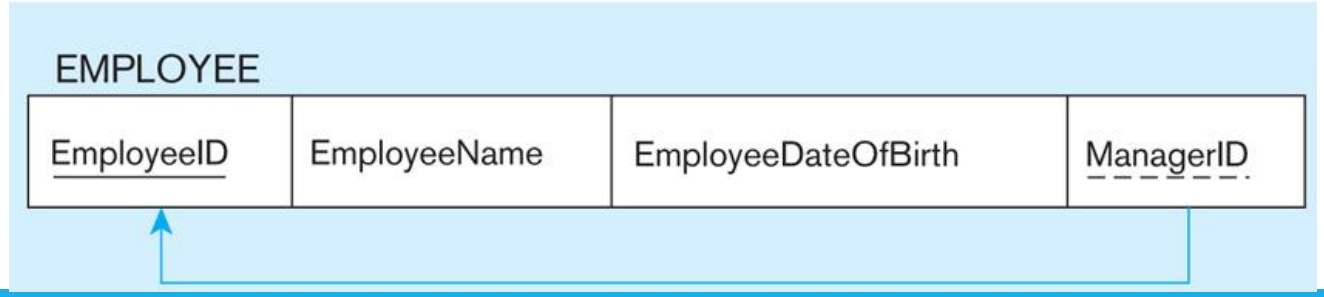
Head Coach

| **CoachID** | **Name** | **Career_Wins** | **Career_Losses** |
|---|---|---|---|

# Map Unary 1:M Relationship

- A single entity type → A singe relation
  - Add a (recursive) FK attribute to the relation, which references the PK values in the same relation

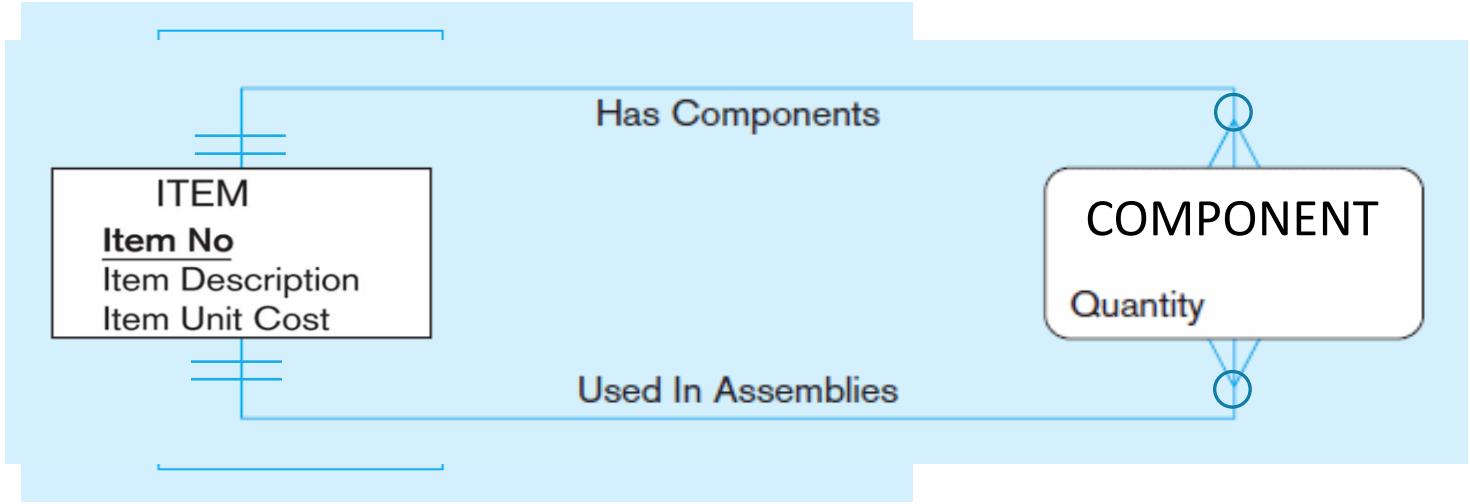


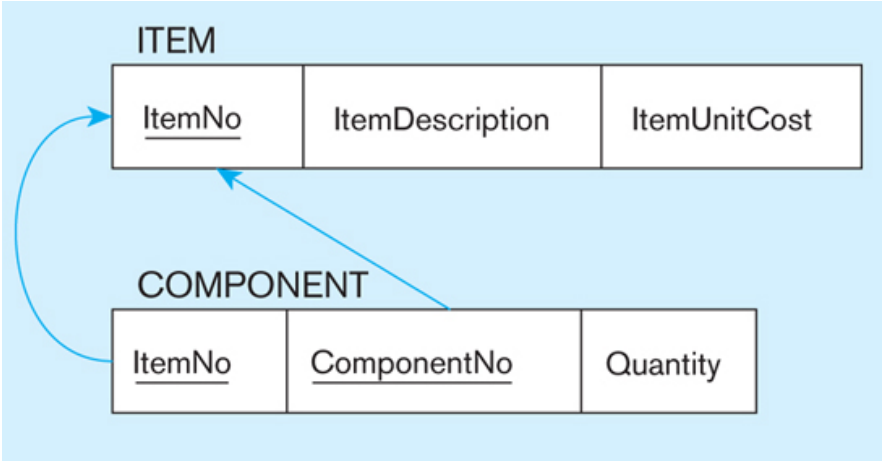




| EmployeeID | Employee Name | Employee DOB | ManagerID |
|---|---|---|---|
| 001 | James | 02/10/1990 | 002 |
| 002 | Alice | 10/20/1985 | 003 |
| 003 | Mike | 04/26/1980 | null |

# Map Unary M:N Relationship

- The single entity type → one relation
  - Contain all attributes of the entity type
- The relationship → a new relation
  - The PK consists of two attributes, both are also FK attributes referencing the PK of the other relation

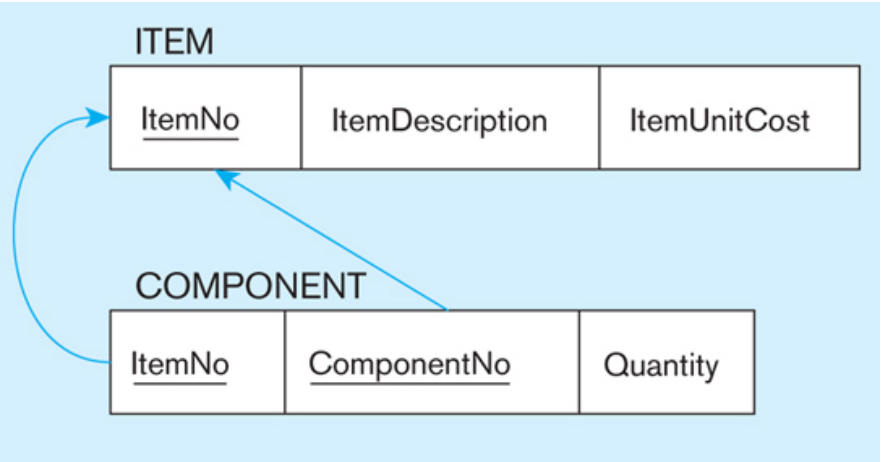


# Example

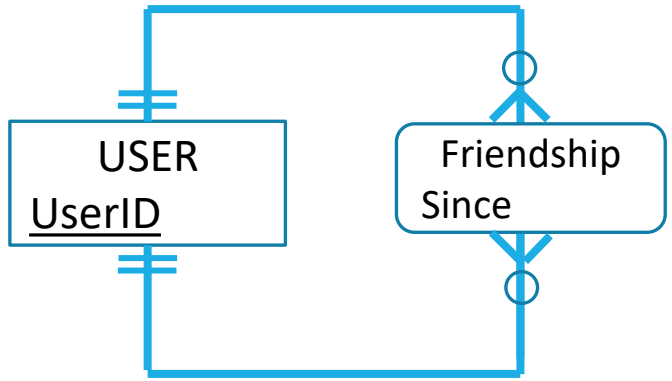
| ItemNo | ItemDescription | ItemUnitCost |
|---|---|---|
| 001 | Table Leg | 10 |
| 002 | Table Top | 30 |
| 003 | Dining Table | 80 |
| 004 | Dining Chair | 30 |
| 005 | Dining Set | 200 |

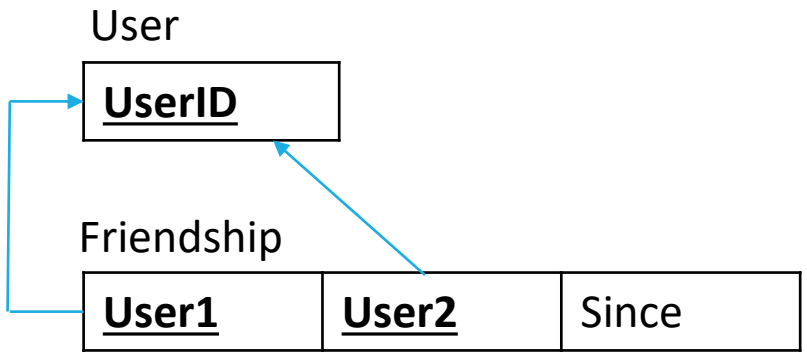
| ItemNo | ComponentNo | Quantity |
|---|---|---|
| 003 | 001 | 4 |
| 003 | 002 | 1 |
| 005 | 003 | 1 |
| 005 | 004 | 4 |



# Map Unary M:N Relationship

- The single entity type → one relation
  - Contain all attributes of the entity type
- The relationship → a new relation
  - The PK consists of two attributes, both are also FK attributes referencing the PK of the other relation



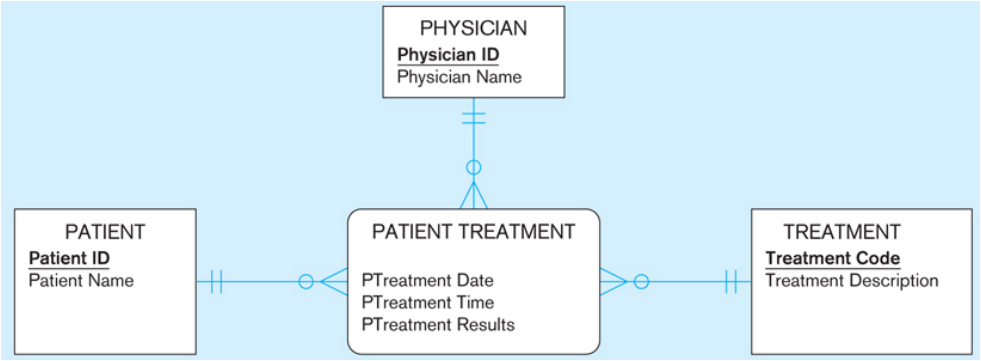



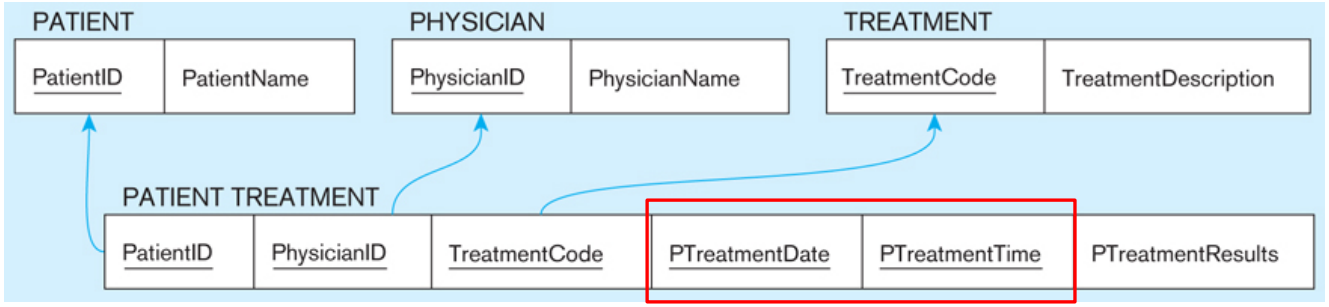

| **UserID** |
|---|
| James |
| John |
| Alice |
| Mary |

| **User1** | **User2** | Since |
|---|---|---|
| James | Alice | 2015 |
| James | John | 2016 |
| Alice | Mary | 2016 |

# Map Ternary Relationship

- Three participating entity types → three relations
- Relationship → an associative relation
  - Include the PK attribute(s) of the three entities/relations as the PK/FK attributes



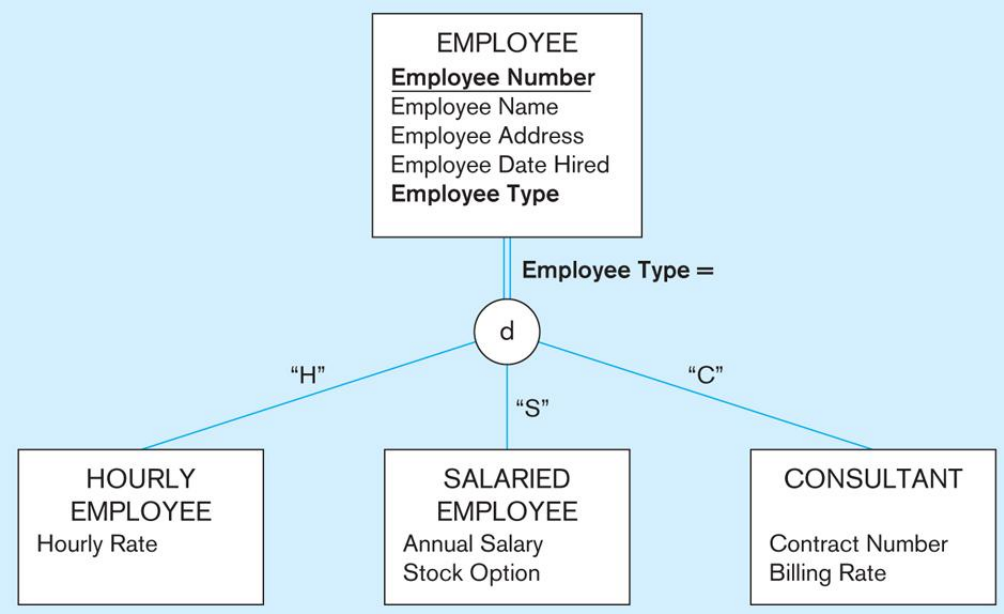



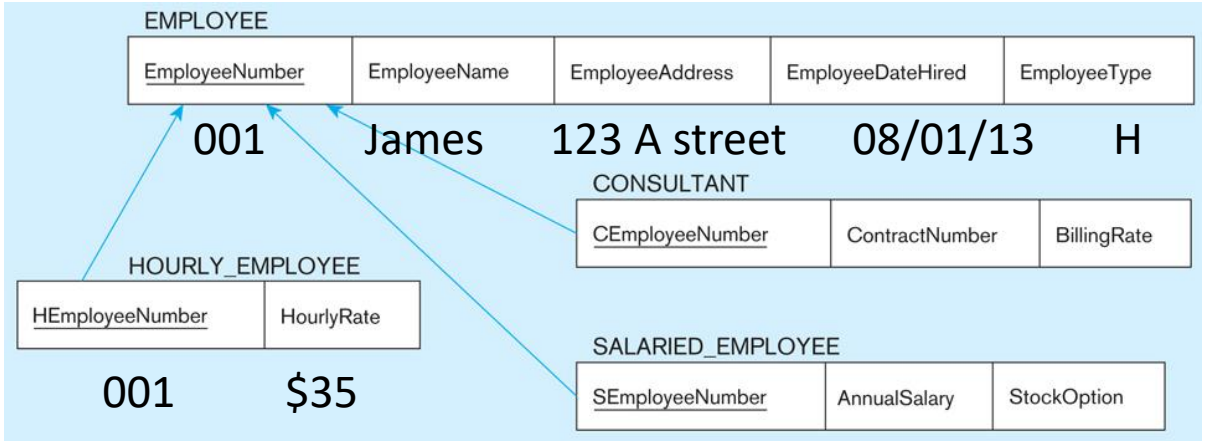

Why do we have to include PTreatmentDate and PTreatmentTime as two components of the PK?

# Map Supertype/Subtype Relationship

- Supertype → one relation
  - Contain all common attributes and the subtype-discriminator attribute
- Each subtype → an individual relation
  - Include the PK attribute of the supertype relation as the PK/FK attribute
  - Contain those attributes unique to the subtype only

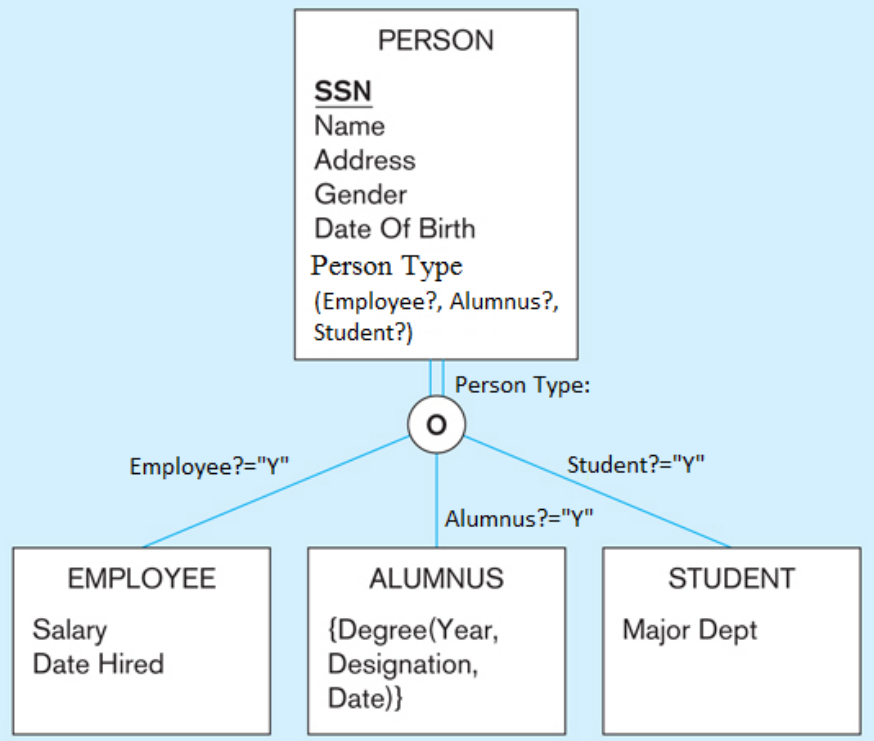
What if the disjointness constraint is overlap rule?

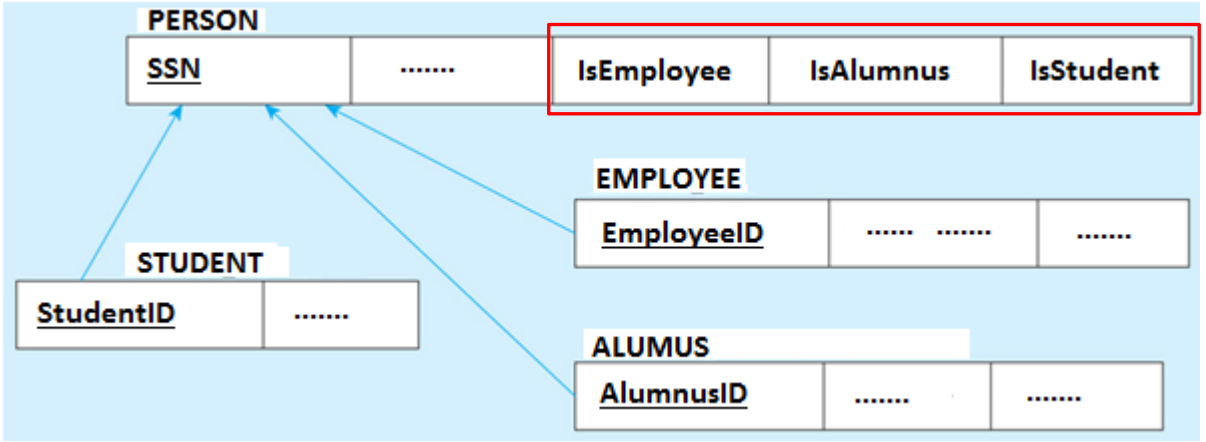


# Map Supertype/Subtype Relationship

- subtype-discriminator: composite attribute for overlap rule



# Delete Rules

- What happens if a referenced row is deleted?

| CustomerID | CustomerName | Address |
|---|---|---|
| 1 | Jack | 123 A street |
| ~~2~~ | ~~Mark~~ | ~~245 B street~~ |
| 3 | Jane | 123 C ave |

| OrderID | Date | CustomerID |
|---|---|---|
| 1 | Oct-1 | 1 |
| 2 | Nov-1 | 2 |
| 3 | Jan-1 | 2 |

# Delete Rules

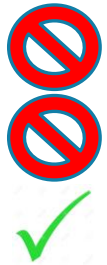
- What happens if a referenced row is deleted?
- Restrict: don't allow delete of "parent" side if related rows exist in "dependent" side
- Cascade: Automatically delete "dependent" side rows that correspond with the "parent" side row to be deleted
- Set-to-Null: set the foreign key in the dependent side to null if deleting from the parent side

The choice of these options is based on your business rules.

# Delete Rules: Option 1

- Restrict: don't allow delete of "parent" side if related rows exist in "dependent" side

Can delete?



| CustomerID | CustomerName | Address |
|---|---|---|
| 1 | Jack | 123 A street |
| 2 | Mark | 245 B street |
| 3 | Jane | 123 C Ave |

| OrderID | Date | CustomerID |
|---|---|---|
| 1 | Oct-1 | 1 |
| 2 | Nov-1 | 2 |
| 3 | Jan-1 | 2 |

# Delete Rules: Option 2

- Cascade: Automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted

| CustomerID | CustomerName | Address |
|---|---|---|
| 1 | Jack | 123 A street |
| ~~2~~ | ~~Mark~~ | ~~245 B street~~ |
| 3 | Jane | 123 C Ave |

| OrderID | Date | CustomerID |
|---|---|---|
| 1 | Oct-1 | 1 |
| ~~2~~ | ~~Nov-1~~ | ~~2~~ |
| ~~3~~ | ~~Jan-1~~ | ~~2~~ |

# Delete Rules: Option 3

- Set-to-Null: set the foreign key in the dependent side to null if deleting from the parent side

| CustomerID | CustomerName | Address |
|---|---|---|
| 1 | Jack | 123 A street |
| ~~2~~ | ~~Mark~~ | ~~245 B street~~ |
| 3 | Jane | 123 C Ave |

| OrderID | Date | CustomerID |
|---|---|---|
| 1 | Oct-1 | 1 |
| 2 | Nov-1 | NULL |
| 3 | Jan-1 | NULL |

# Summary

1. Entities become tables. Multi-valued attributes become new tables with composite PK.
2. 1:M relationship, add  PK of 1 side into M side as FK
3. 1:1 add PK of mandatory side as FK into optional side.
4. For associative entity, follow 2. The FKs also become a composite PK if no surrogate key defined
5. For Unary relationship, follow 2-4. The only difference is you reduce the number of tables by 1
6. Ternary: follow 4. Also add necessary attributes into the PK.
7. Subtype: each becomes a new table. Use supertype's PK as its own PK (also FK). Don't copy Supertype's other attributes.