CST3604 Supplemental Lecture Notes

Physical Database Design

Database Design – Review of Basic Database Design Principles

(Part 2 of 2)

(Lecture Notes 0B)

Prof. Abel Angel Rodriguez

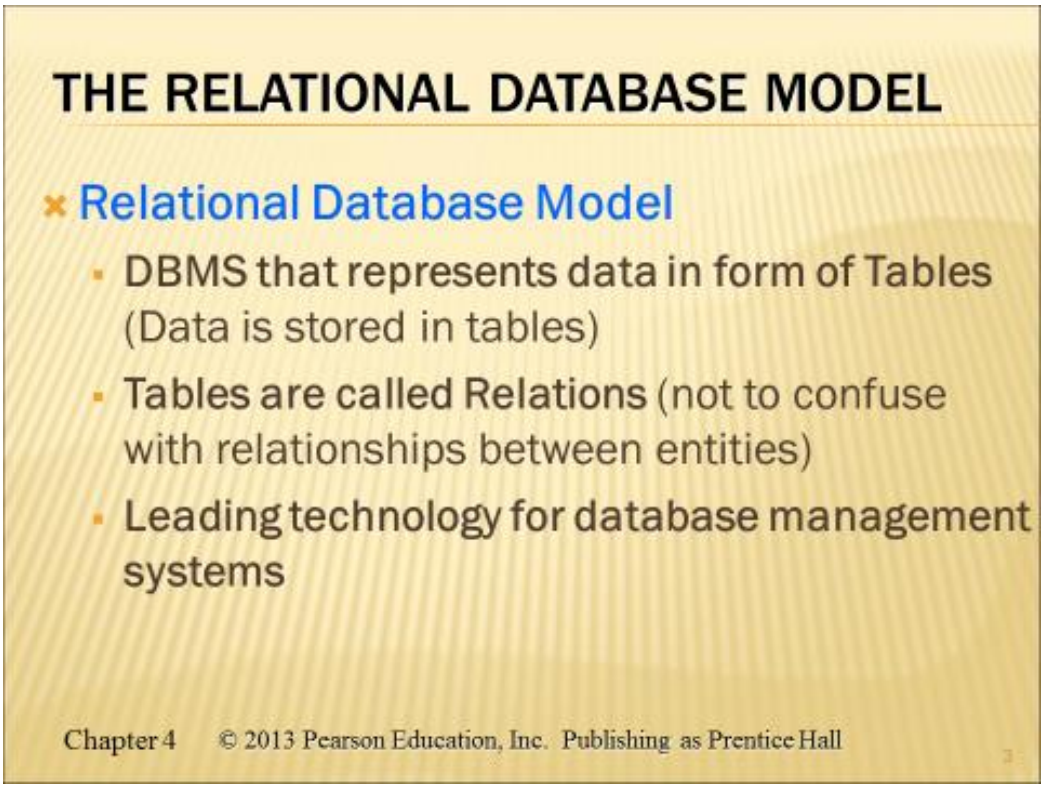
Lecture 1 The Relational Database Model

1 - The Relational Database Model

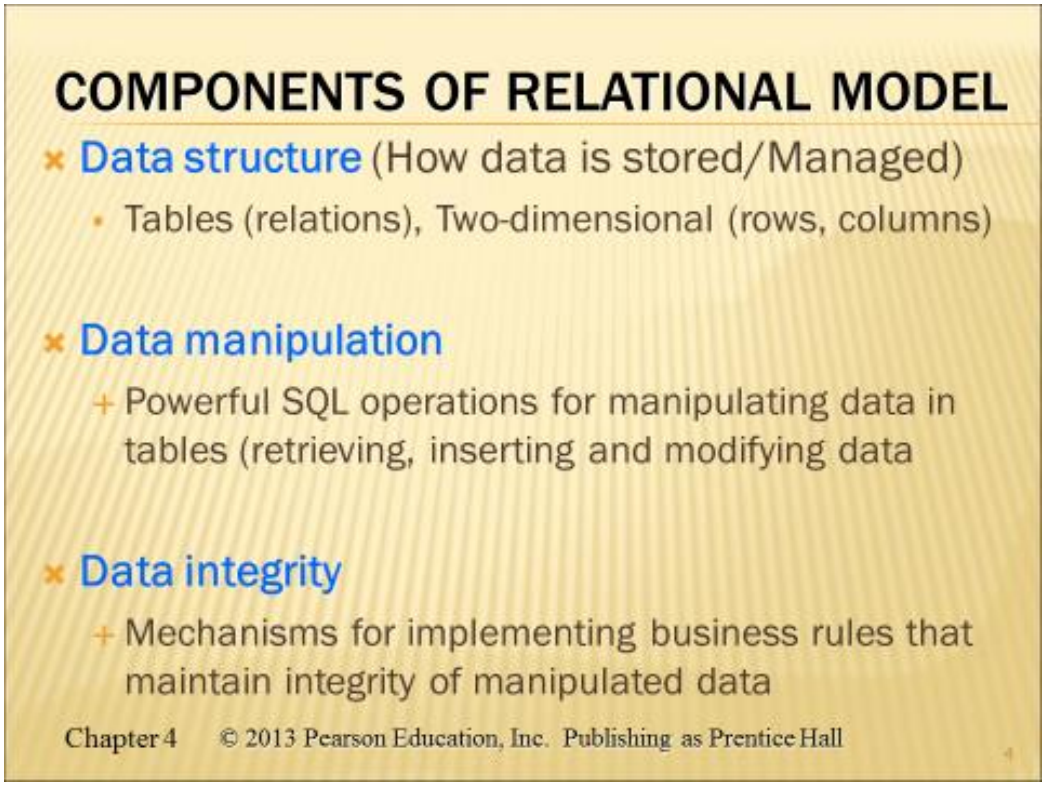
1.1.1 Components of the Relational Database Management System (DBMS)

Components of a Relational Database Model

- What is the Relational Database Model:



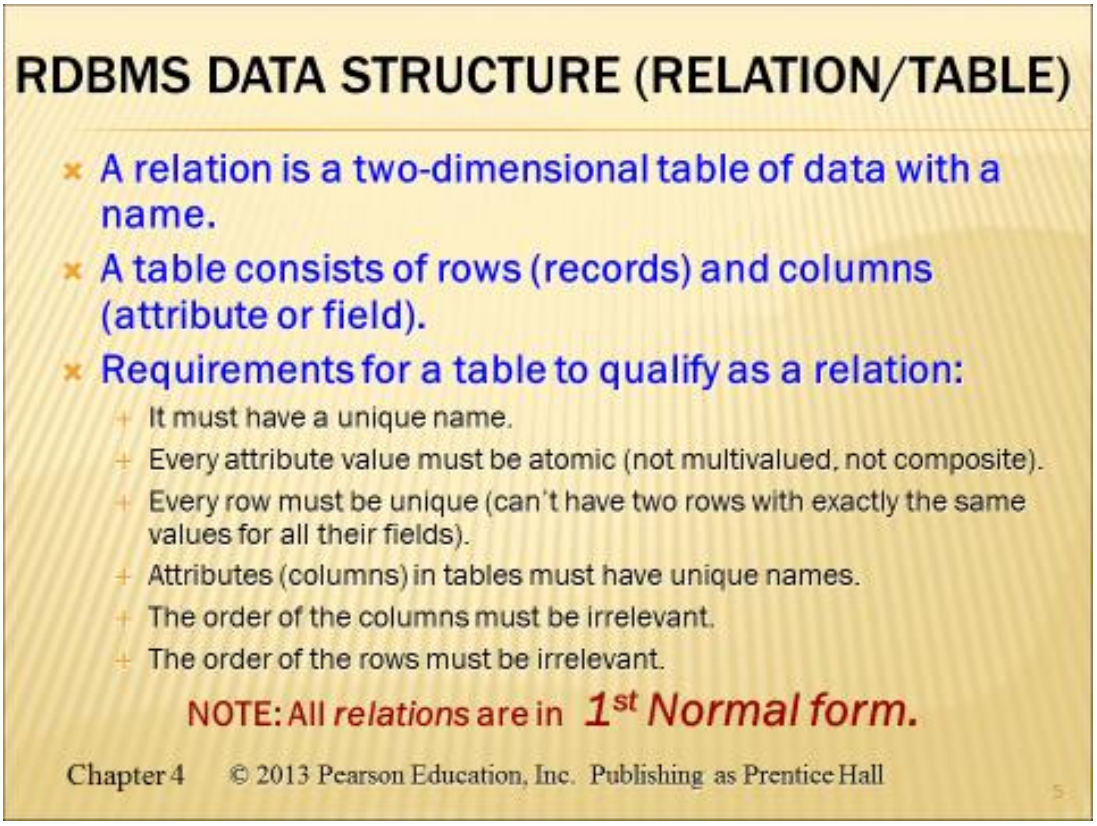
- What are the key features compared to other types of DBMS?:

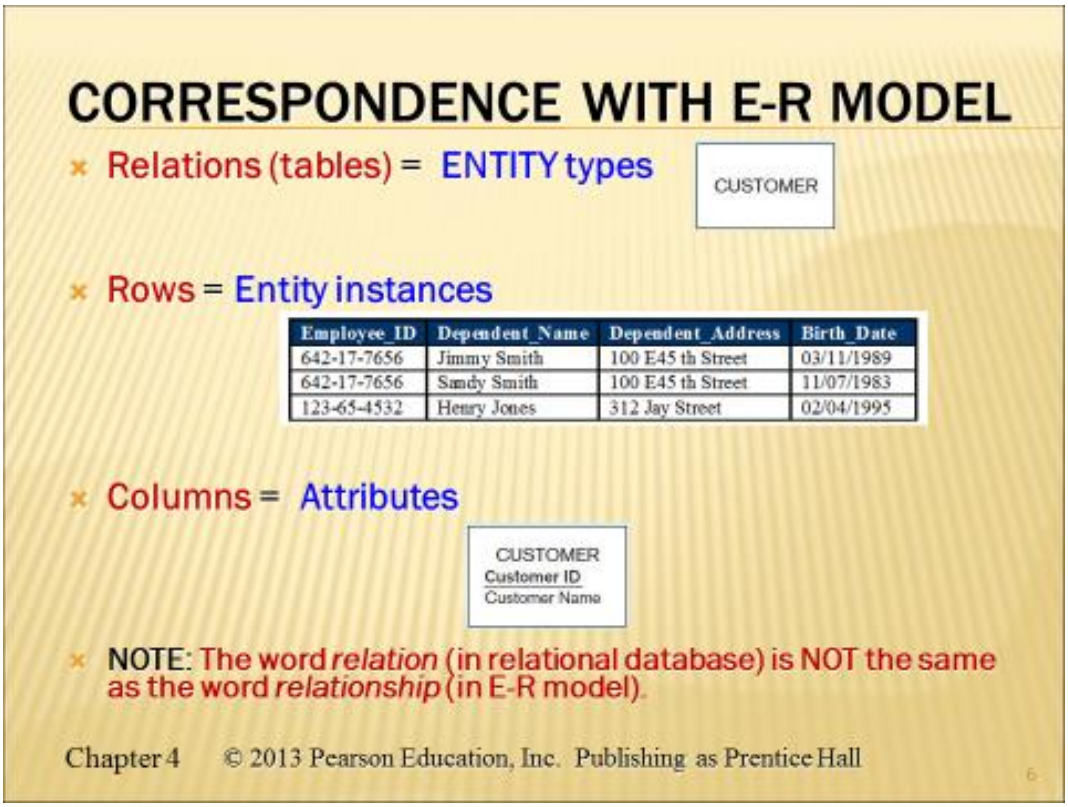


1.2. Data Structure in RDBMS – Storage Using Tables

Data Structure (Relation/Table)

- Data Structure component (How data is stored):



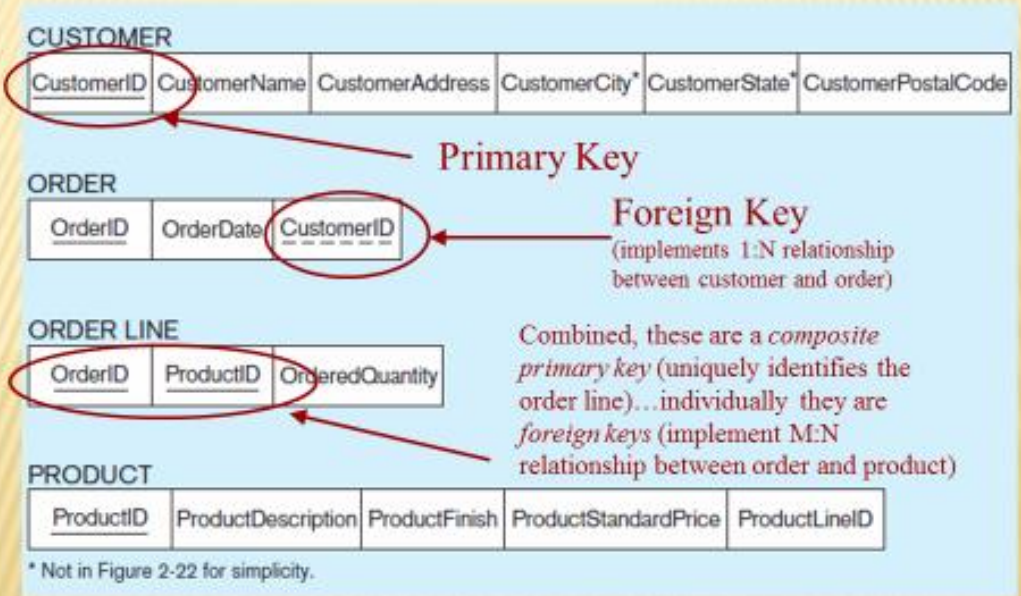


Relational Keys

RELATIONAL KEY FIELDS

- Every relation/table must have a primary key
- Keys are special columns that serve two main purposes:
  - *Primary keys* are unique identifiers of the relation.
    - Examples include employee numbers, social security numbers, etc. *This guarantees that all rows are unique.*
  - *Foreign keys* are (non-primary key) identifiers in a in a dependent (child) table in a relationship with another table parent table where it is a primary key
    - A column that is a primary key in another table, but a foreign key in this table

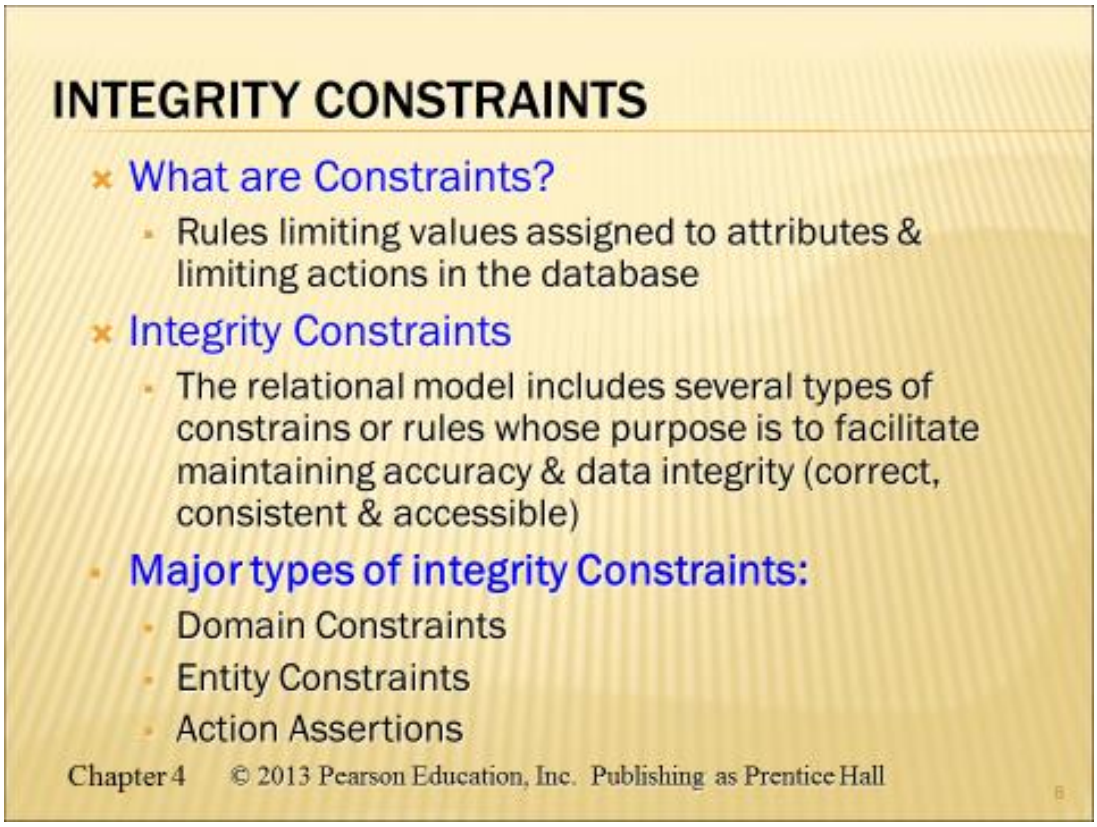
Figure 4-3 Schema for four relations (Pine Valley Furniture Company)



1.3 Integrity Constraints in RDBMS – Rules & Controls Provided by RDBMS

Integrity Constraints (Rules)

- Rules:



Domain Constraints:

INTEGRITY CONSTRAINTS – DOMAIN CONSTRAINTS

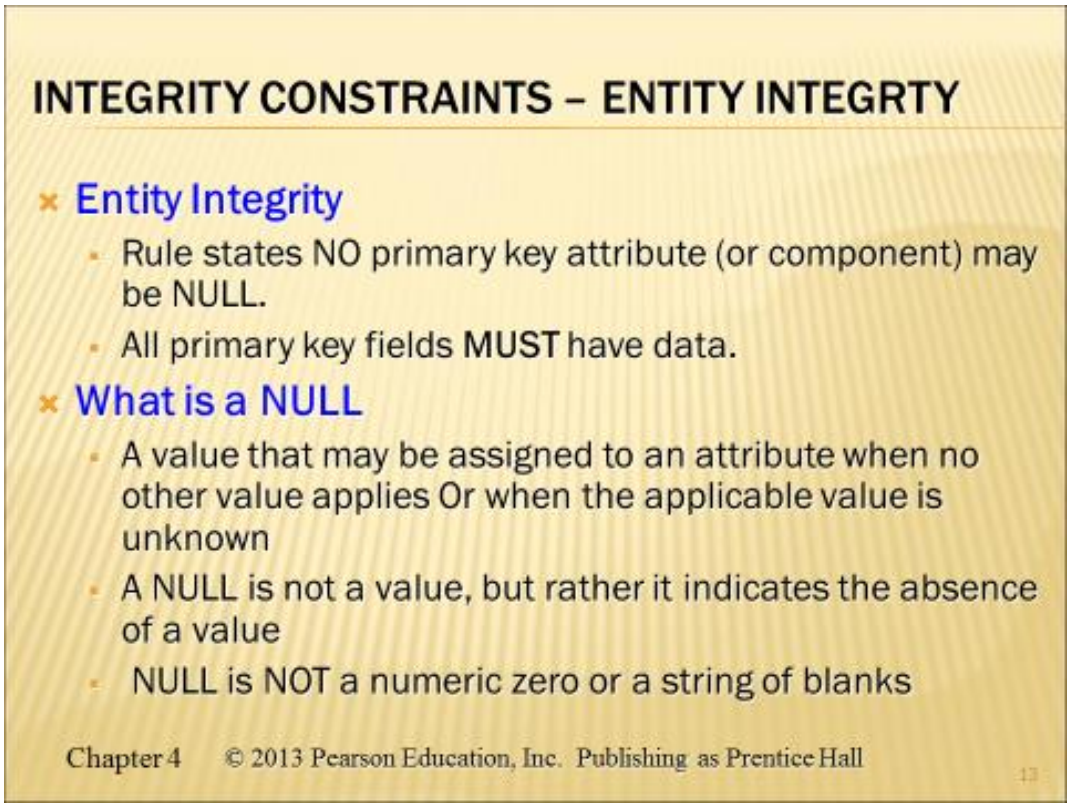
- Domain Constraints
  - The set or allowable values that can be assigned to an attribute
  - Can be listed in a domain definition table (See Table 4-1)
  - Domain definition table contains:
    - **Domain name** – name describing attribute domain
    - **Domain meaning** – description of attribute
    - **Data type** – type of data storage available in DBMS
    - **Data size** – length of data
    - **Value range** – number of allowable values

TABLE 4-1 Domain Definitions for INVOICE Attributes

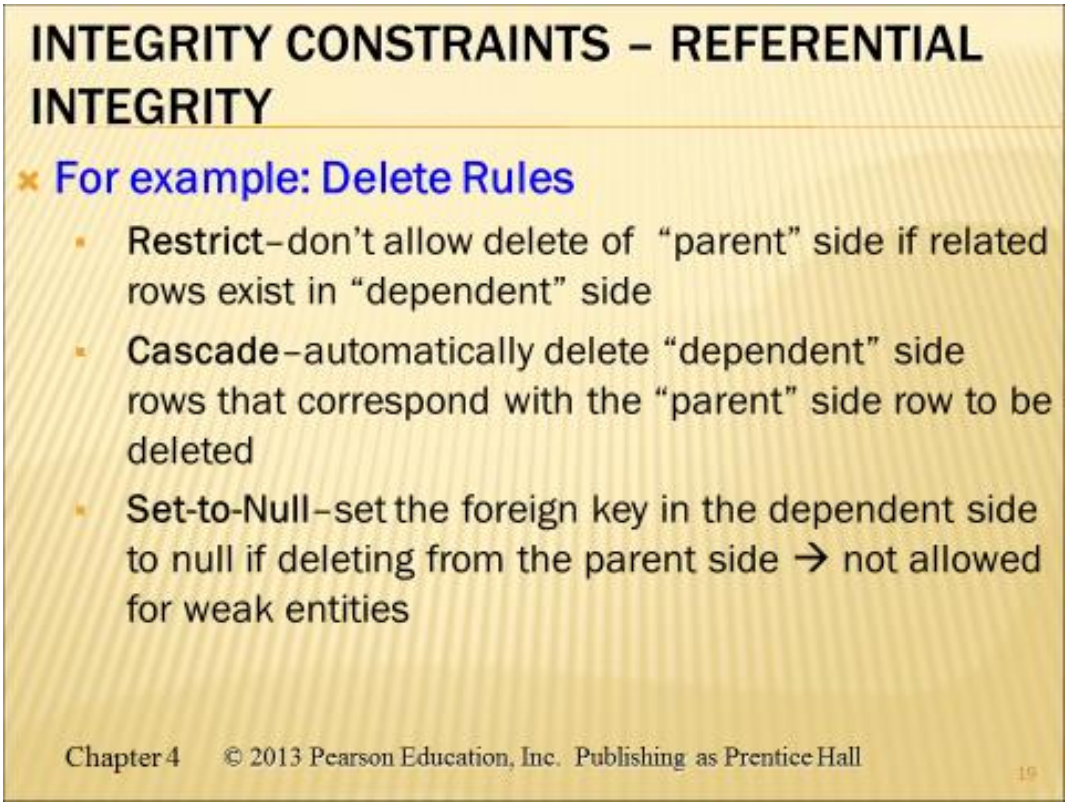
Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
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 definitions enforce domain integrity constraints.

Entity Constraints:

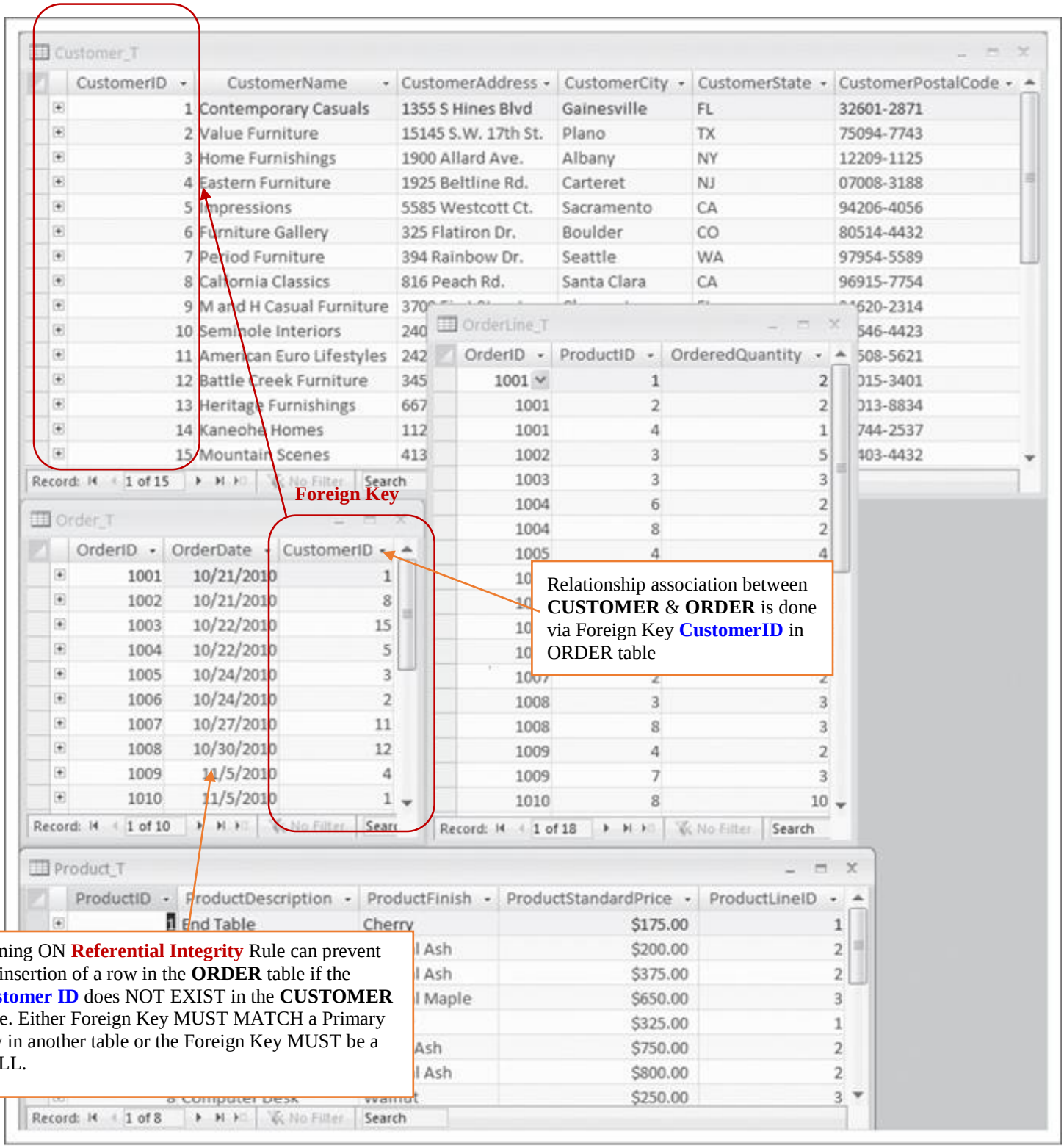


Referential Integrity:



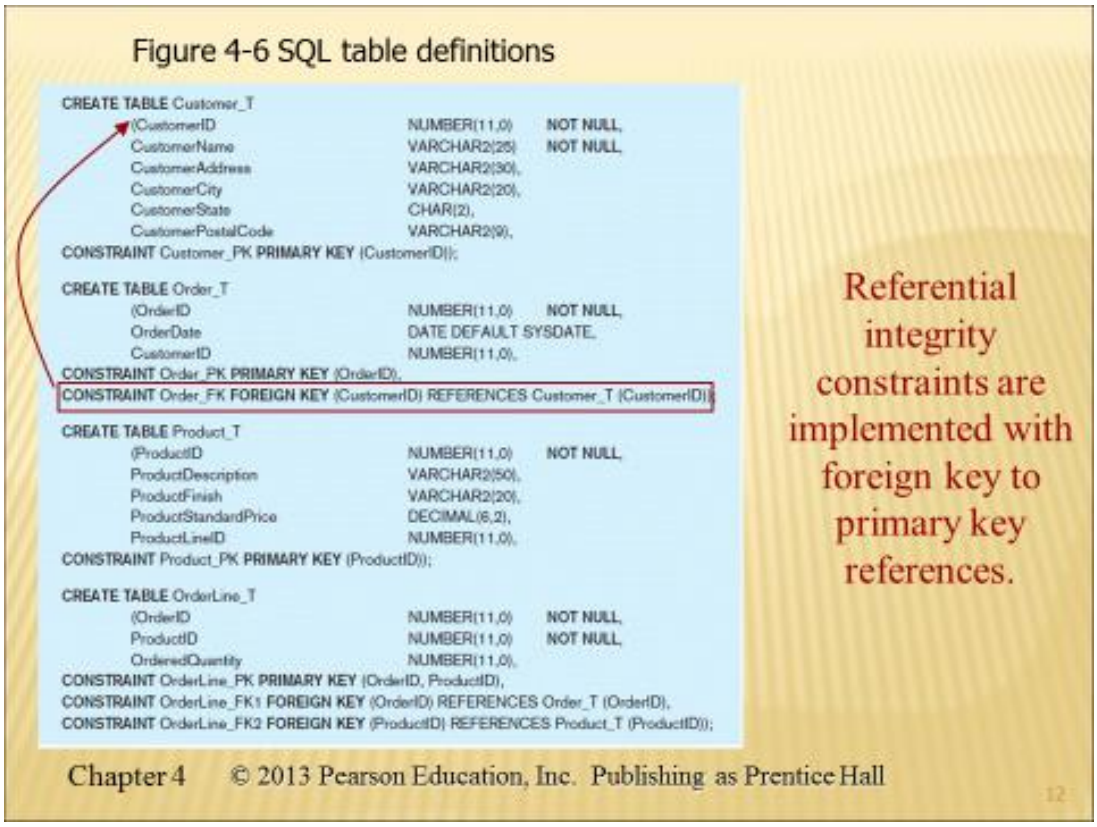
INTEGRITY CONSTRAINTS – REFERENTIAL INTEGRITY

- Pine Valley Furniture Relational Schema Example:
  - **Figure 4-3** – Example of table instance of the Pine Valley schema
  - The association between two tables is defined via Primary and Foreign Key.
  - Example association between CUSTOMER & ORDER tables is defined in the ORDER table by the Foreign Key CustomerID.
  - **Referential Integrity Enforcement** – If you insert a new row in the ORDER table, the customer record or row must already exist in the CUSTOMER table.



Creating Tables in the RDBMS & Referential Integrity

- Tables or relations can be created in the following ways in most RDBMS:
  - RDBMS provides a GUI – a graphical user-interface will enable you to create & define the table & column, etc.
  - RDBMS SQL Scripts (Best Practice) – You can also create a SQL script that will create the tables.
- Referential Integrity CONSTRAINTS can be turned on via the SQL SCRIPT used to create that tables as shown below:



1.3 Summary of Components of the Relational Database Management System (DBMS)

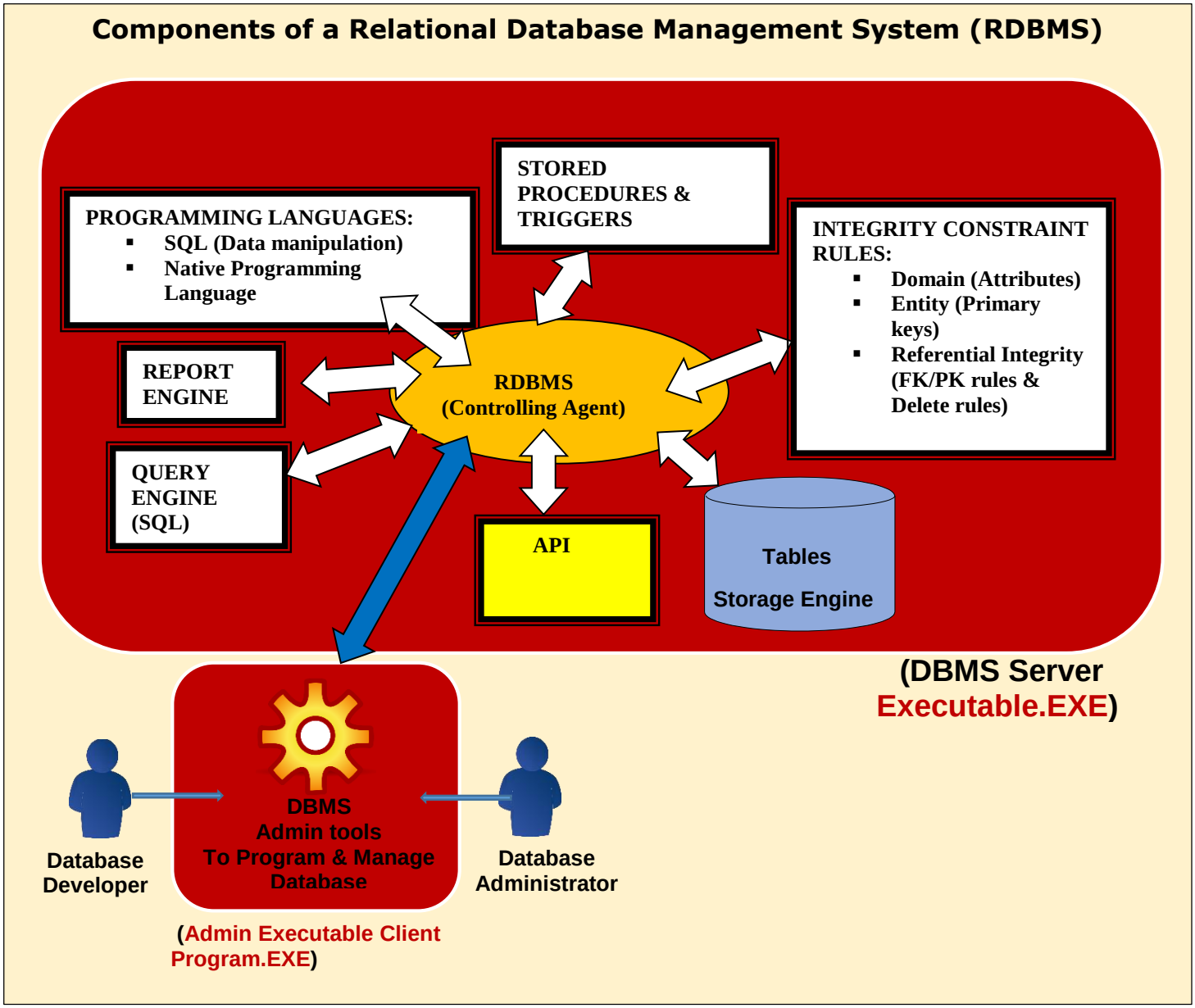
Summary of the Relational Database Model

Relational Database Management System (DBMS) Components

- RDBMS Components details:
  - A RDBMS is primarily composed of the following basic components:

Item	Component	Description
1.	**Storage System (Tables)**	▪ Storage System where data is kept ▪ Relations DBMS use tables to store data
2.	**Queries Engine**	▪ Engine to enable the querying (searching, updating, inserting & deleting) the data using SQL Language.
3.	**Reporting Services**	▪ Engine to enable the creation/generation of reports or formatted result of database queries and contains useful data for decision-making and analysis.
4.	**programming language:** ▪ **Data Manipulation Language: SQL** ▪ **Custom(native-Language)**	▪ Each DBMS comes with a data manipulation language to allow you to search, insert, update and delete records from the storage system. This language is known as SQL (Structured Query Language) ▪ In addition, each DBMS comes with specialized programming language (powerful language similar to Java, C#, C++ etc.) in addition to SQL to write complex Server-side processing (Performing most of the processing on the DATABASE SERVER and not on CLIENT application)
5.	**Programming Tool:** ▪ **Stored Procedures**	▪ Function-like executable BLOCKS of one or more SQL statement & Native-Code grouped together as one EXECUTABLE UNIT! These EXECUTABLE blocks can be called from CLIENT APPLICATIONS to perform the desired processing/query on the database
6.	**Programming Tools:** ▪ **Triggers**	▪ A special type of **Stored Procedure** that **EXECUTES AUTOMATICALLY** under some specified condition (Similar to Event-Handlers in programming languages)
7.	**Administrative Tools/Applications:** ▪ **Database Development User-Interface/Programs** ▪ **Database Administration Interface/program**	▪ **For Developers** – User-Interface Programs that allows the ***Database Developer*** to program, create databases & execute queries against the database. ▪ **IMPORTANT!** These are programs or **client executables** that allows user/developer to interface and program the DBMS ▪ **For Database Administrators** – User-Interface Program that allows the ***Database Administrator*** manage the DBMS network administration, security, backup & Recovery, etc. ▪ **IMPORTANT!** These are programs or **client executables** that allows user/system administrator to interface and manage the DBMS
8.	**Administrative Tools: Other**	▪ Other tools
9.	**Application Programming Interface (API)**	▪ A set of routines, protocols, and tools for integration by external programs into the existing Database Management Software application. ▪ Opens the door to all kinds or integration into the DBMS by other applications etc.
10.	**Data Integrity Mechanism** ▪ **Domain Constraints (Attributes Rules)** ▪ **Entity Constraints (Primary Key rules)** ▪ **Referential Integrity (Foreign Key/Primary Key rules & Delete rules)**	▪ Constraints are RULES limiting **values assigned to attributes** & limiting **actions in the database**. ▪ The relational model includes several types of constrains or rules whose purpose is to facilitate maintaining accuracy & data integrity (correct, consistent & accessible): ○ **Domain Constraints** – Rules on *value* assigned to attributes ○ **Entity Constraints** – Rules on entities *primary keys* (NOT NULL) ○ **Referential Integrity Constraints** – Business rules implemented in the database o Rules between to related tables: - Foreign Keys must have a matching Primary key on another table - Foreign Key can be NULL - Delete rules (Restrict, Cascade, Set-to-Null, etc)

- Illustration of components of a typical RDBMS:



- Examples of Database Management System (RDBMS) Servers/Programs are:

  - **Oracle**
  - **Microsoft SQL Server**
  - **Sybase SQL Server**
  - **MySQL**
  - **IBM DB2**

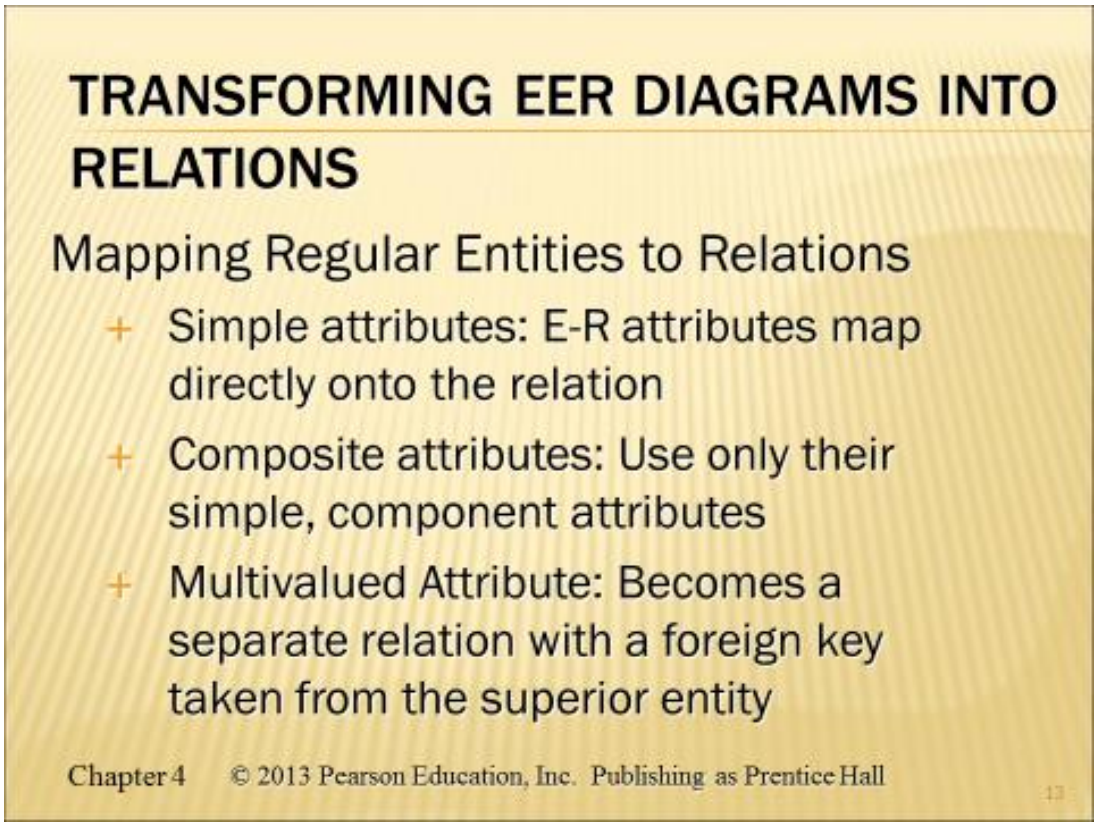
2.0 Transforming EER Diagrams into Logical Diagrams

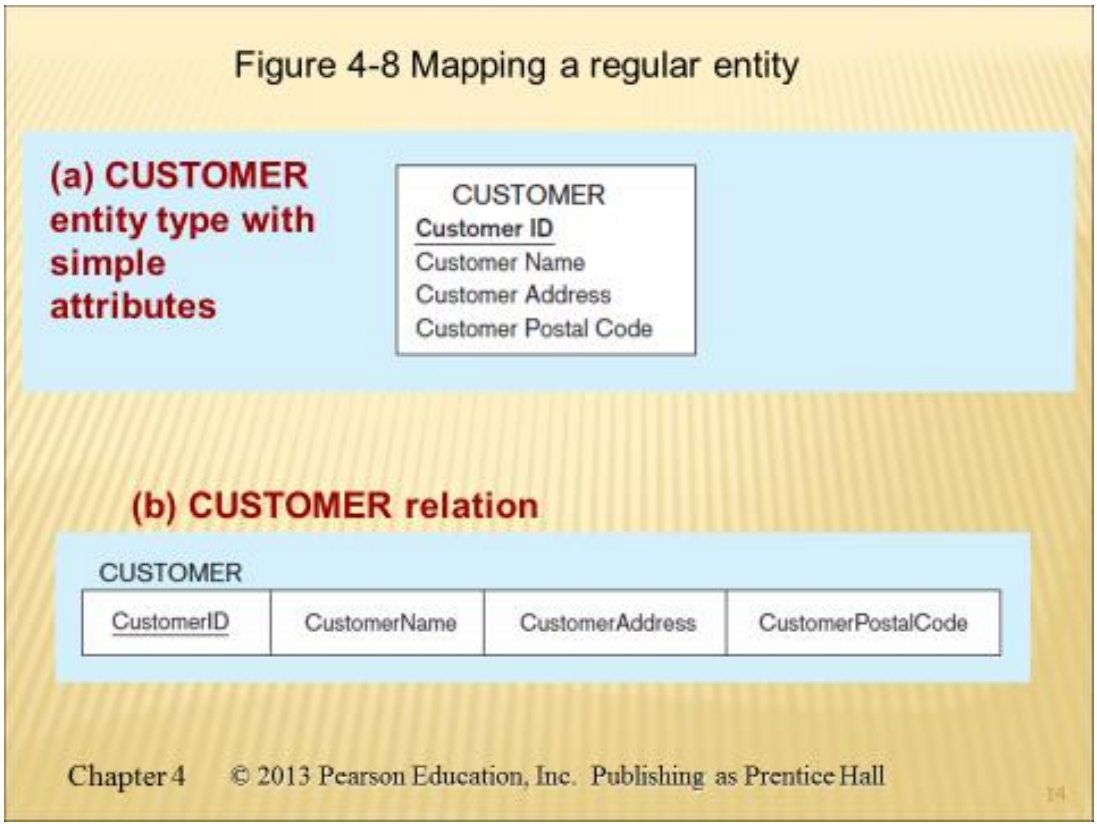
2.1 Transforming EER Diagrams

- Transforming:

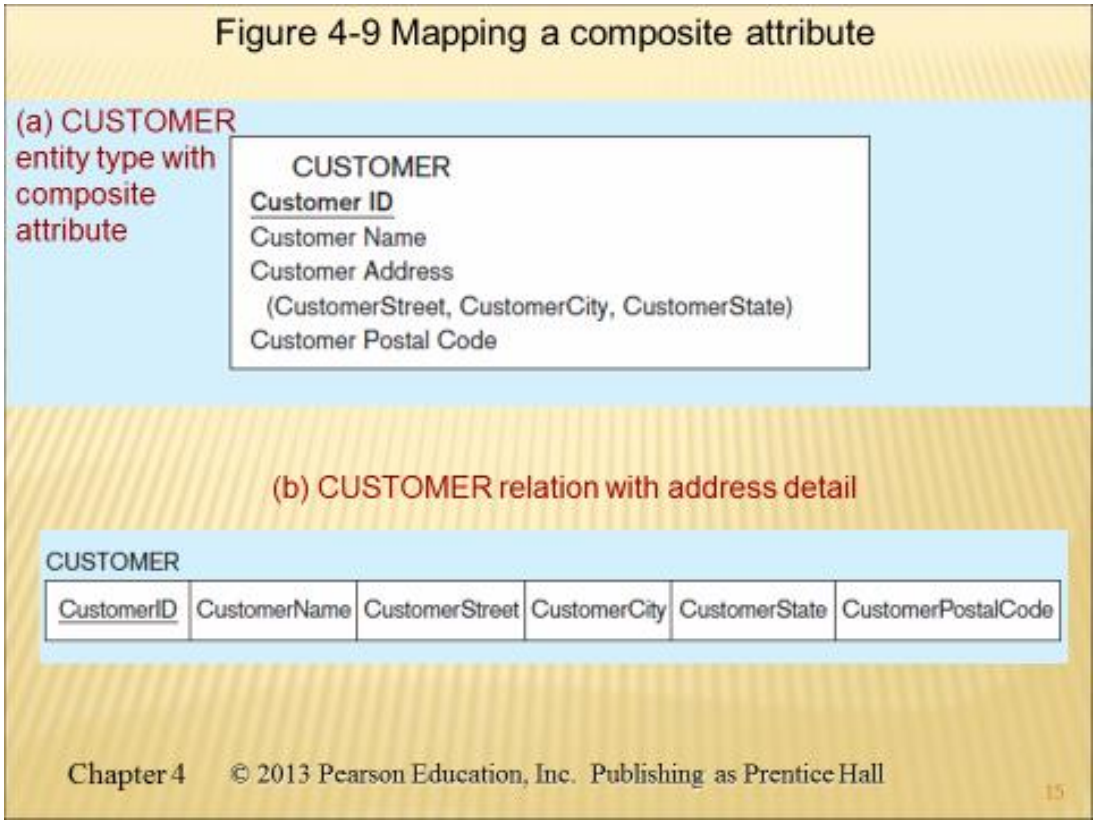
Mapping Regular Entities

Mapping The Entities with Simple Attributes:

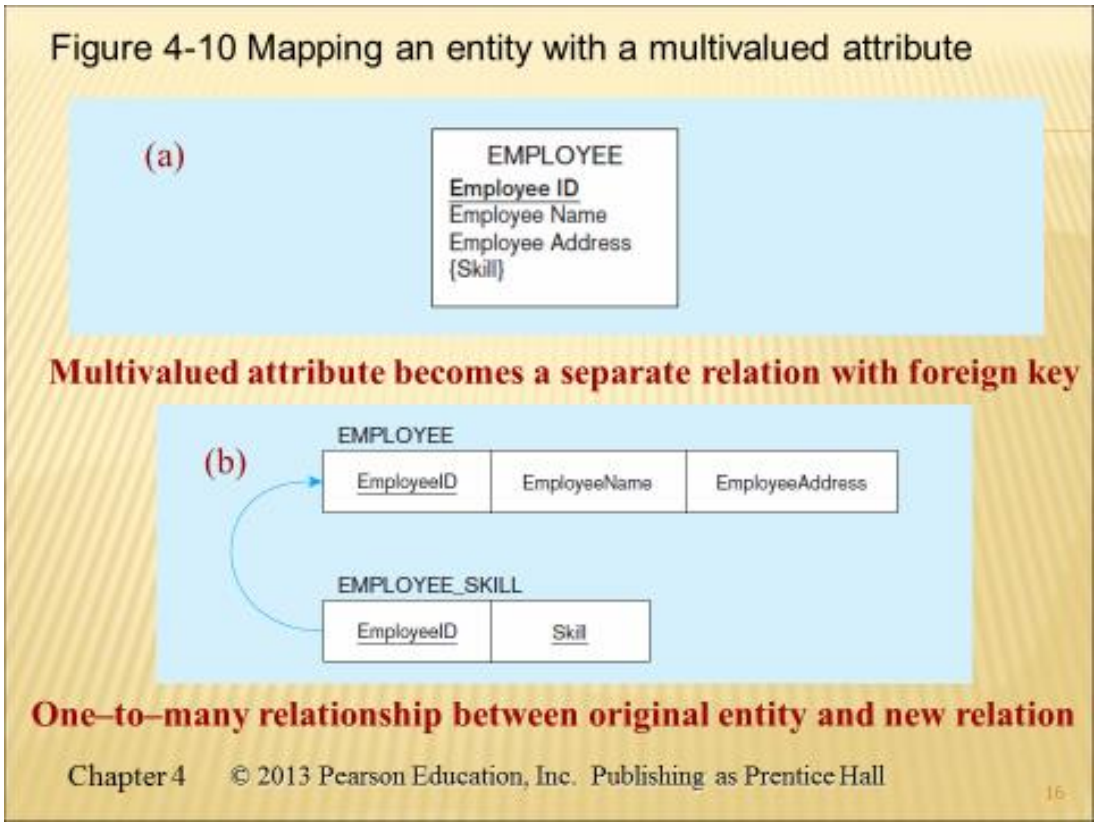




Mapping Entity Composite Attributes



Mapping Entity with Multivalued Attributes



Mapping Weak Entities

TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Weak Entities

- Becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Figure 4-11 Example of mapping a weak entity

a) Weak entity DEPENDENT

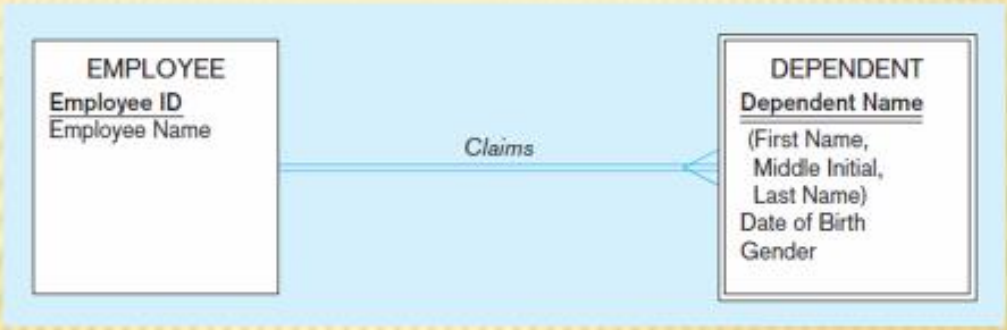
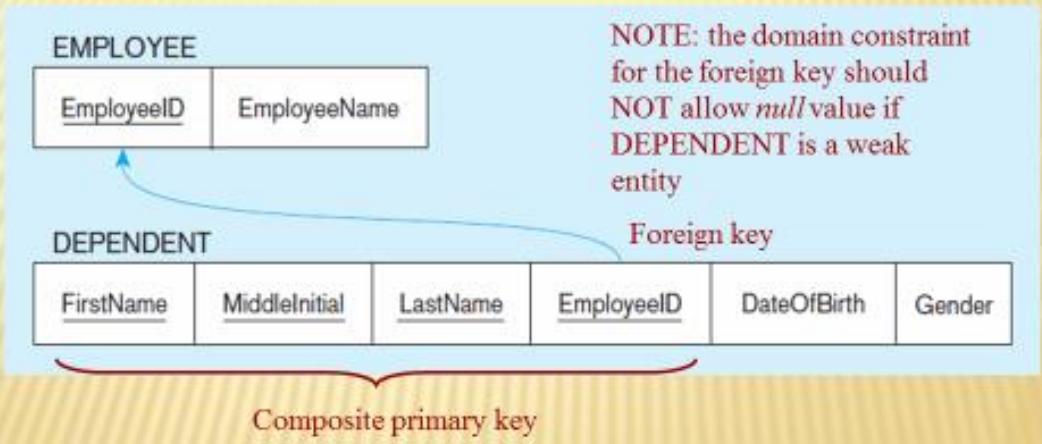


Figure 4-11 Example of mapping a weak entity (cont.)

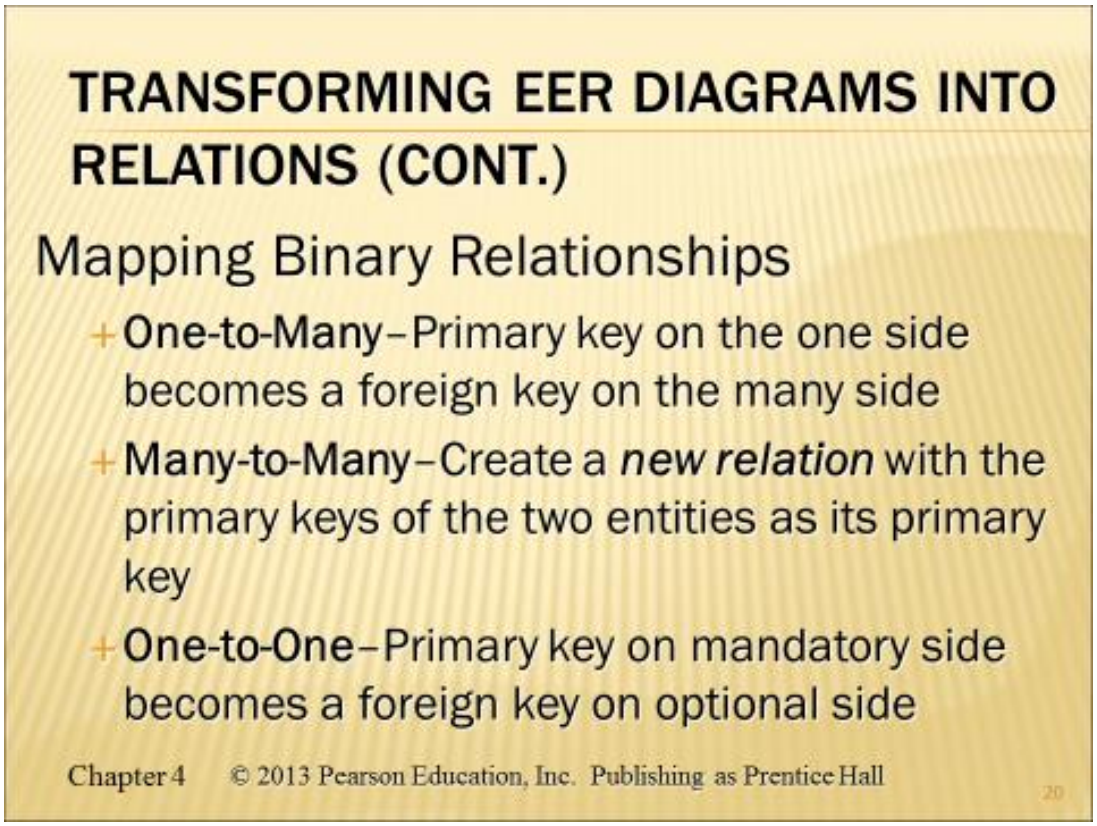
b) Relations resulting from weak entity



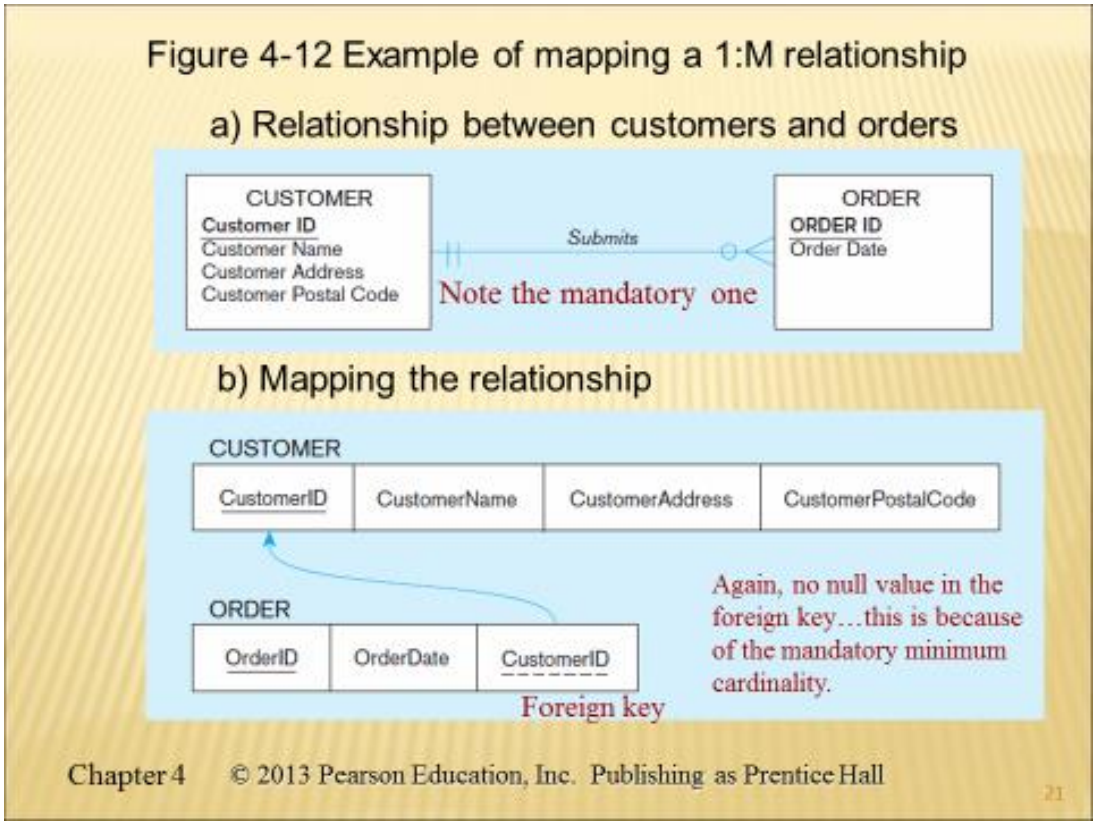
2.2 Transforming EER Diagrams – Mapping Relationships

Mapping Relationships from Conceptual Model to Logical Model

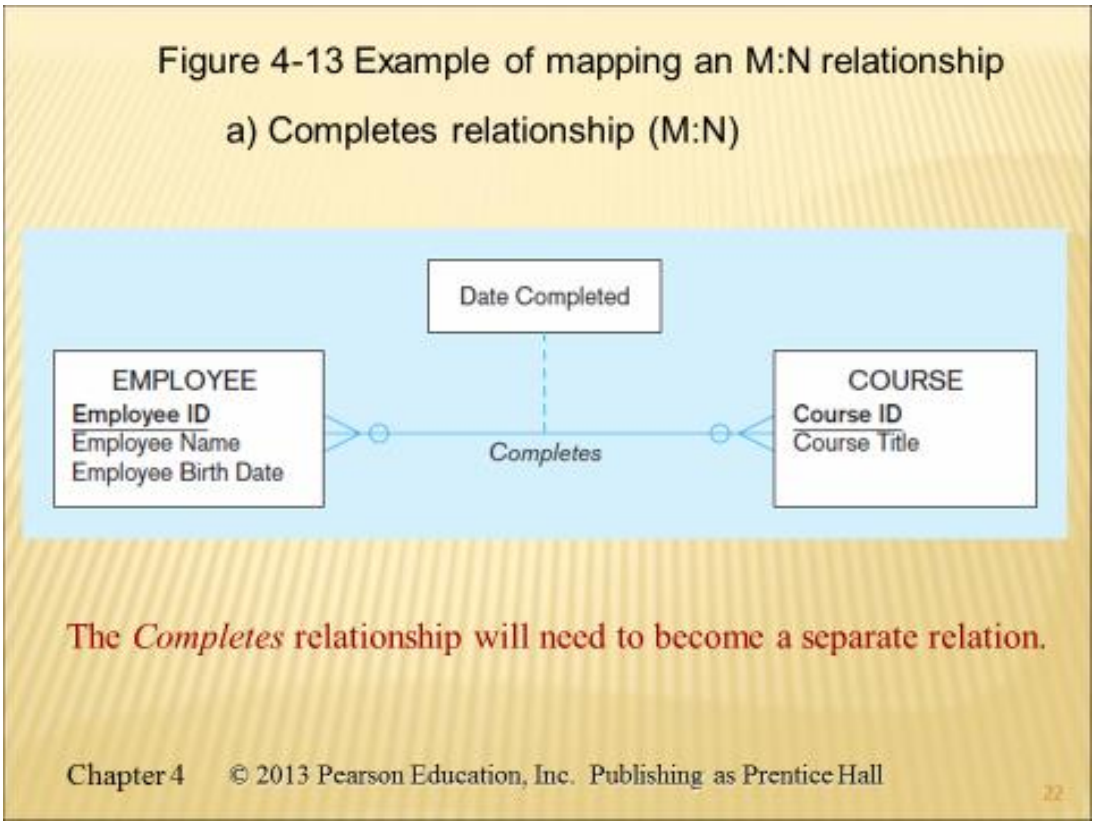
Mapping BINARY Relationships:

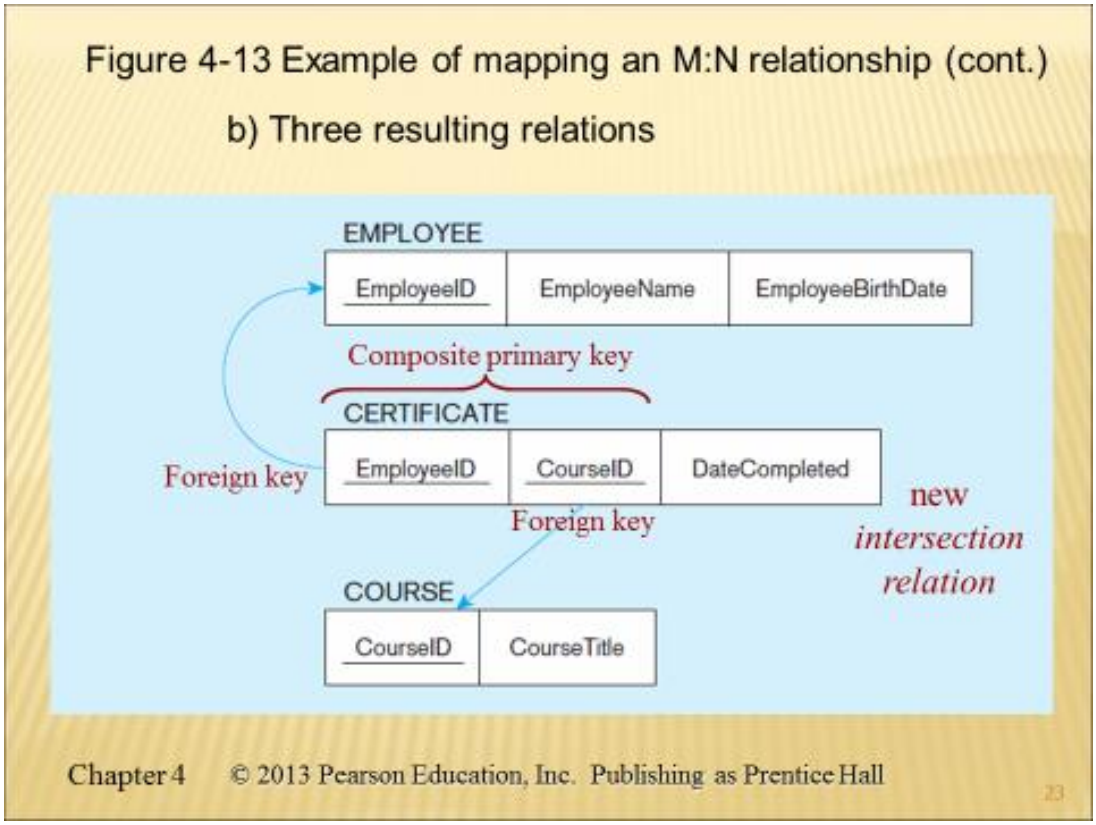


Mapping 1-to-Many (1:M) Binary Relationships:



Mapping Many-to-Many (M:N) BINARY Relationships





Mapping One-to-One (1:1) BINARY Relationships

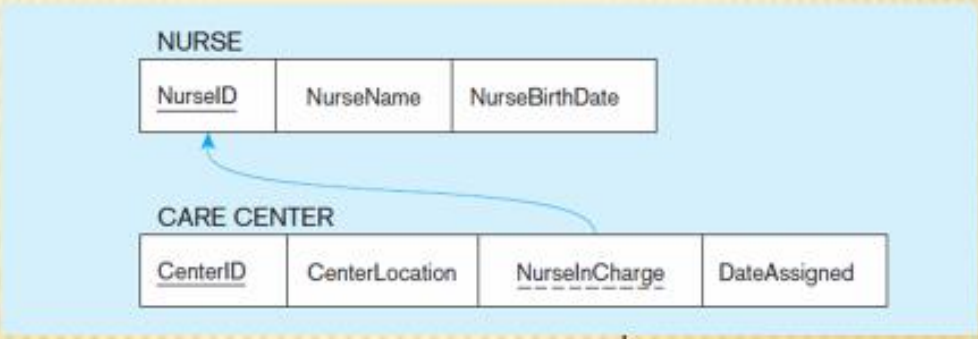
Figure 4-14 Example of mapping a binary 1:1 relationship

a) In charge relationship (1:1)

Often in 1:1 relationships, one direction is optional

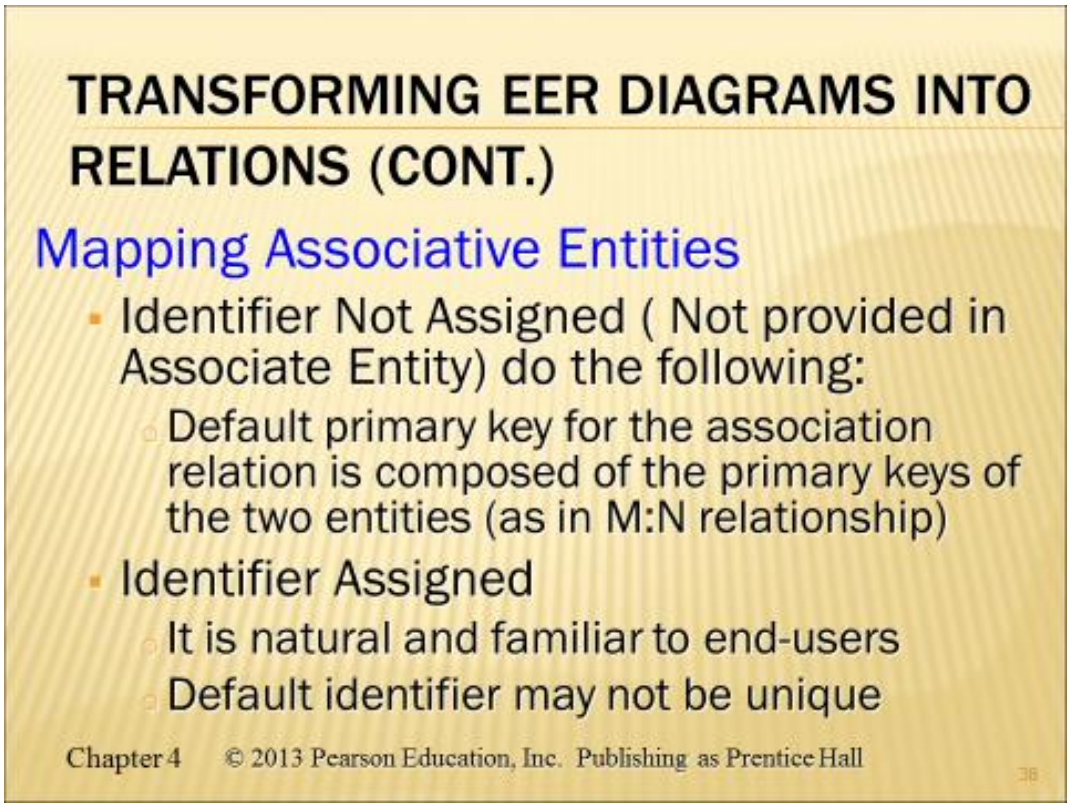
Figure 4-14 Example of mapping a binary 1:1 relationship (cont.)

b) Resulting relations



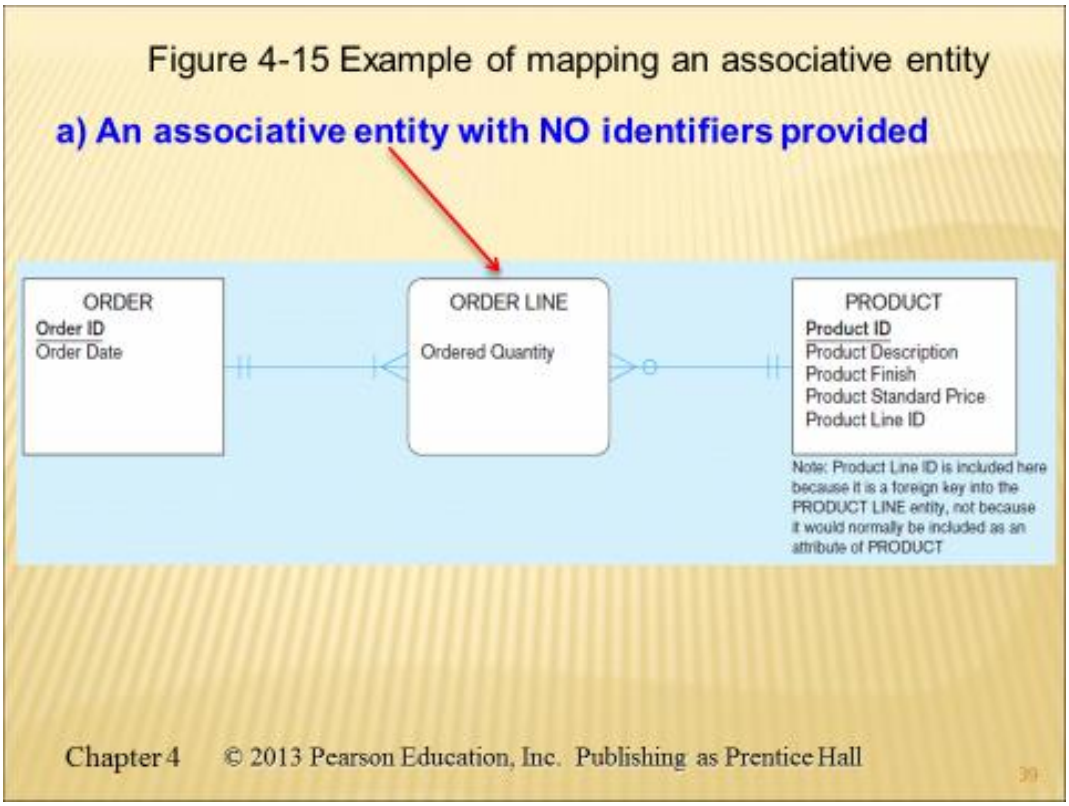
Foreign key goes in the relation on the optional side, matching the primary key on the mandatory side

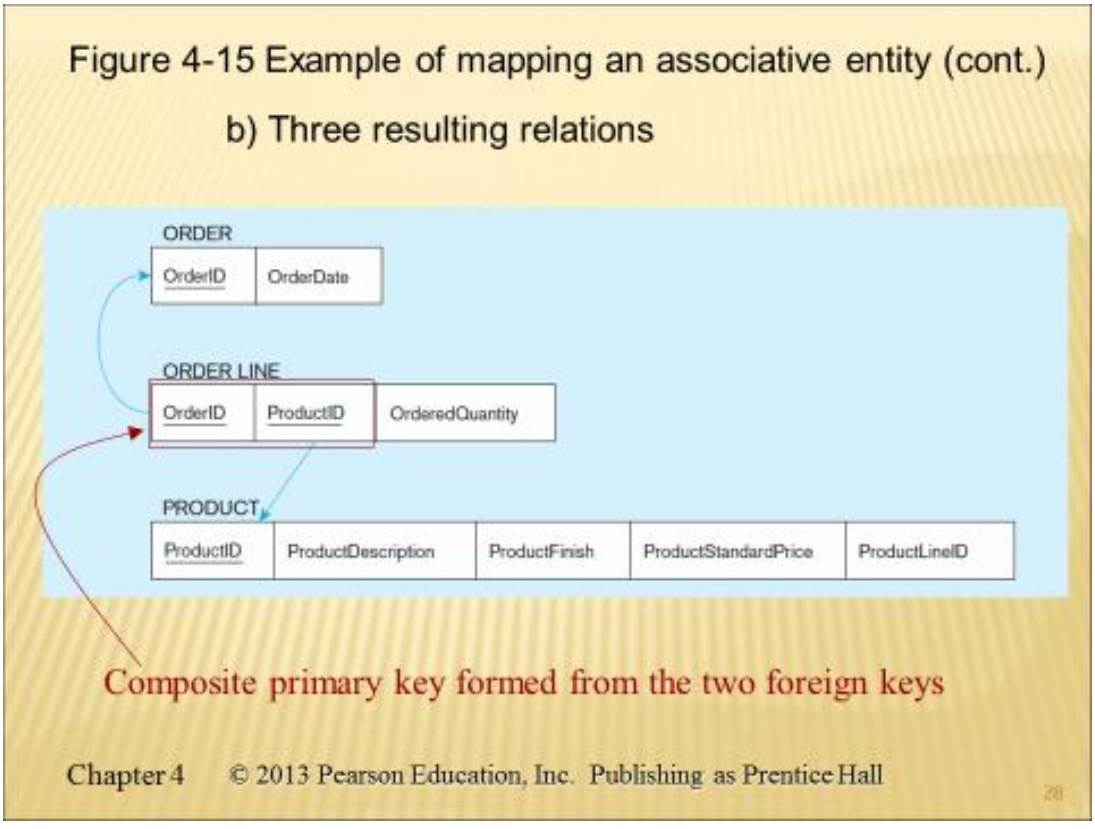
Mapping Associate BINARY Entities Relationships



Mapping Associate BINARY Entities Relationships – Associate Entity has NO Identifiers provided:

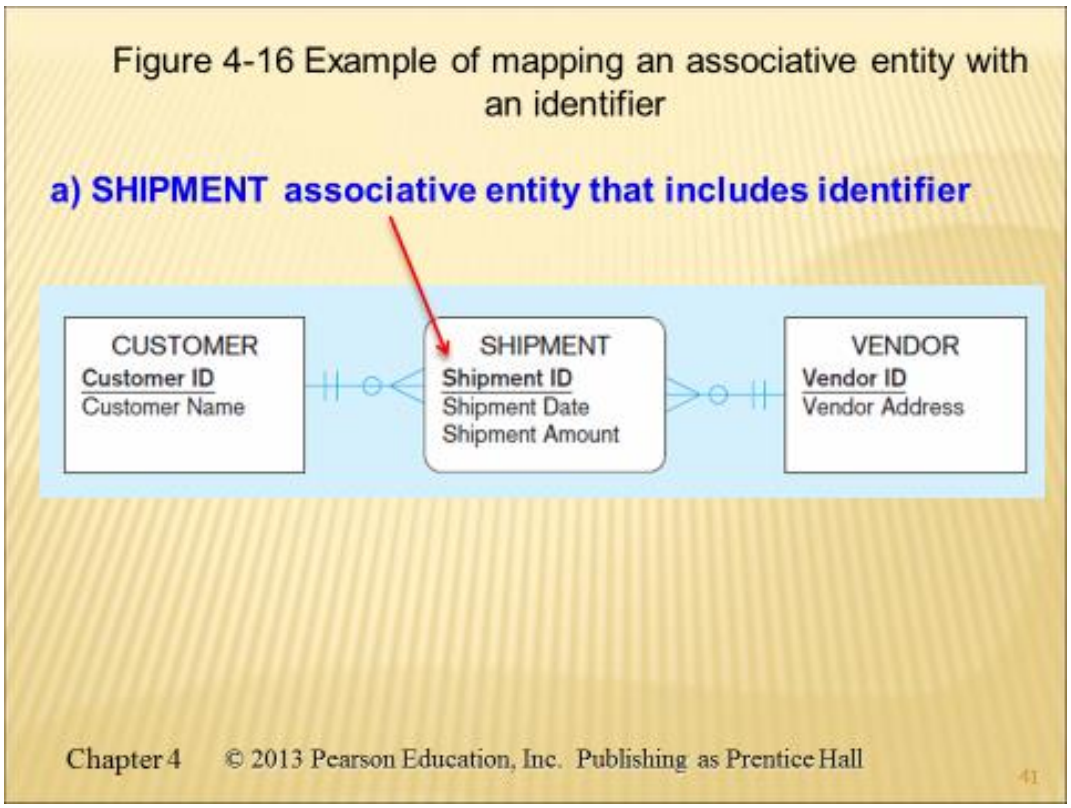
- Remember that Associate Entities are One-to-Many (1:M) relationships between each binary relationship pair
- In this case NO IDENTIFIERS are listed in the Conceptual Model to be translated

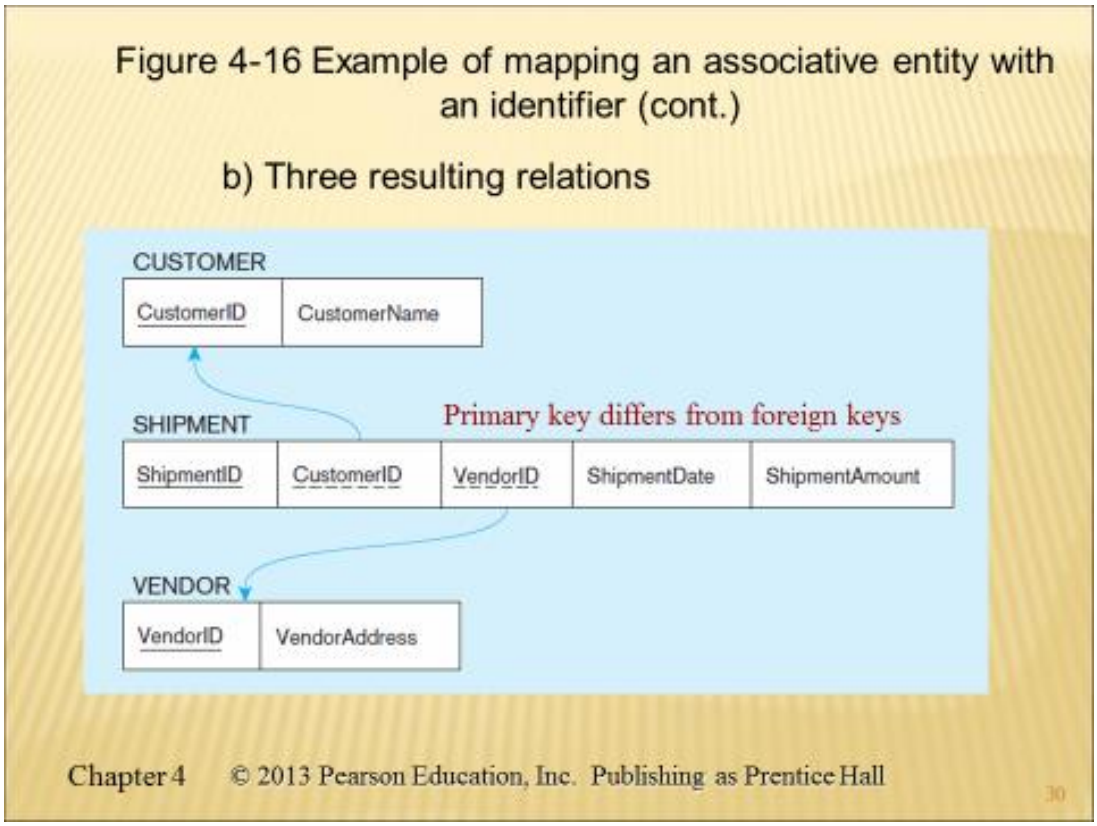




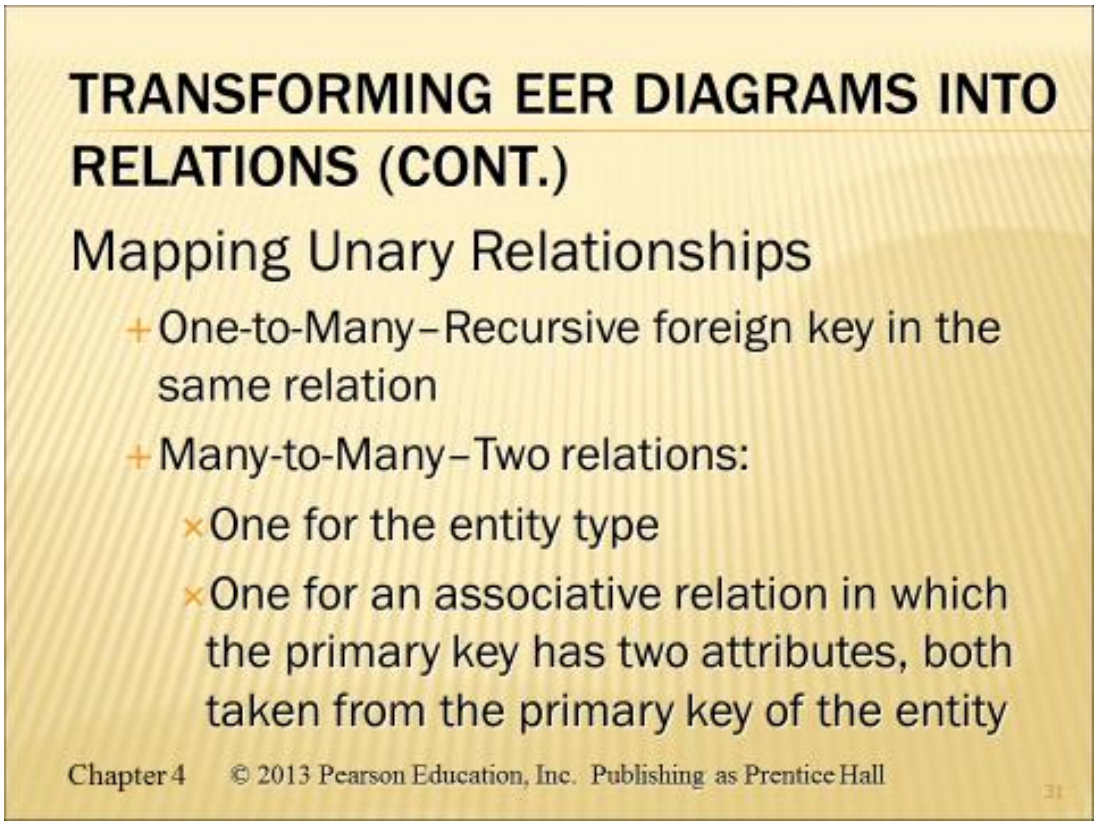
Mapping Associate BINARY Entities Relationships – Associate Entity has NO Identifiers provided:

- In this case an IDENTIFIERS is listed in the Conceptual Model to be translated

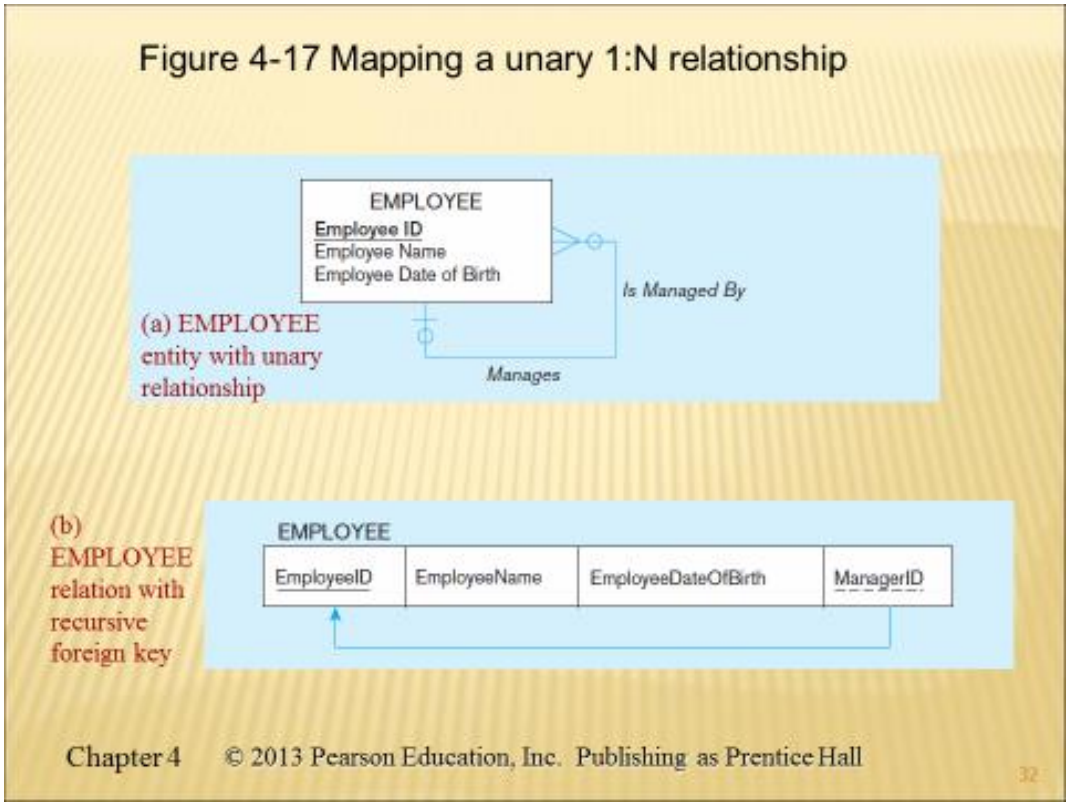




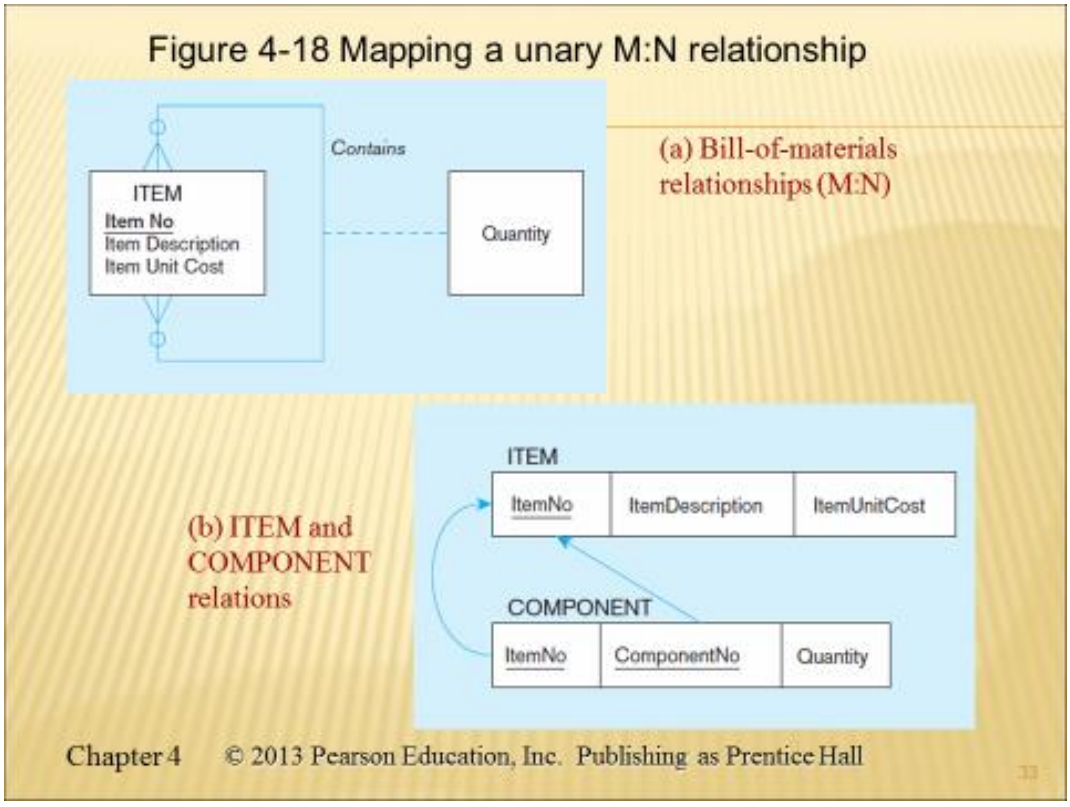
Mapping UNARY Relationships:



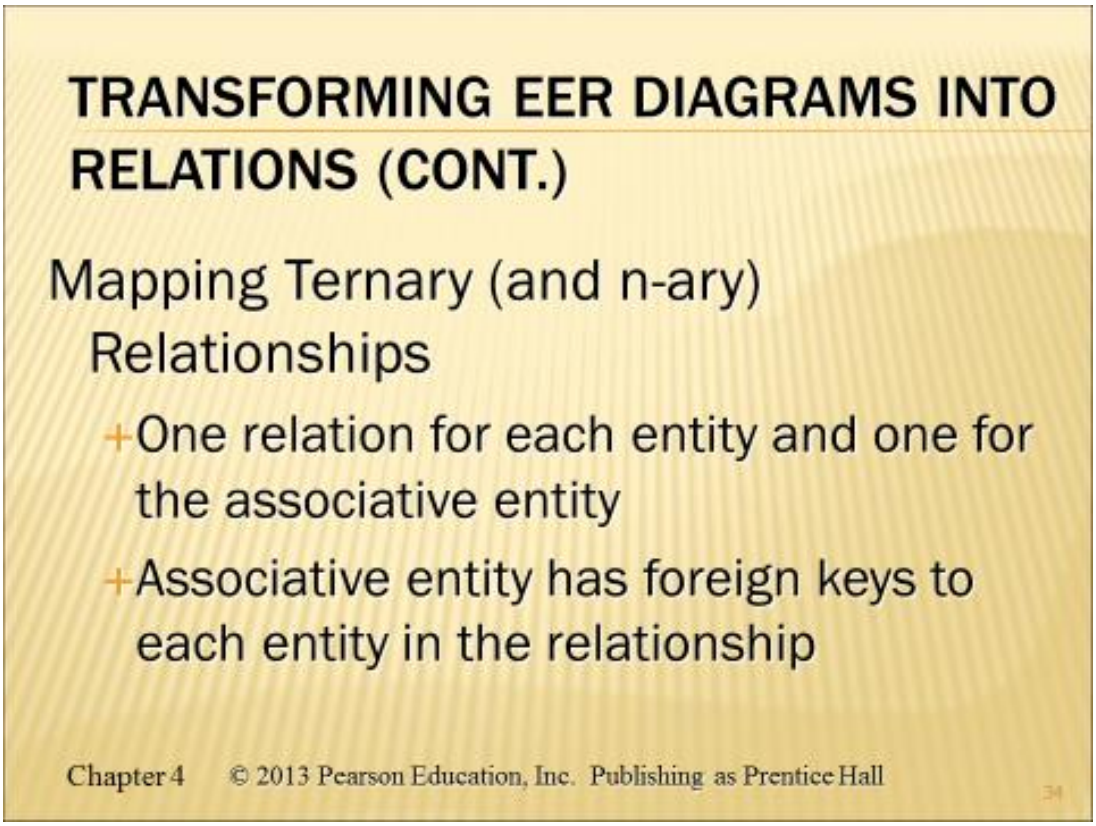
Mapping One-to-Many (1:M) UNARY Entities Relationships



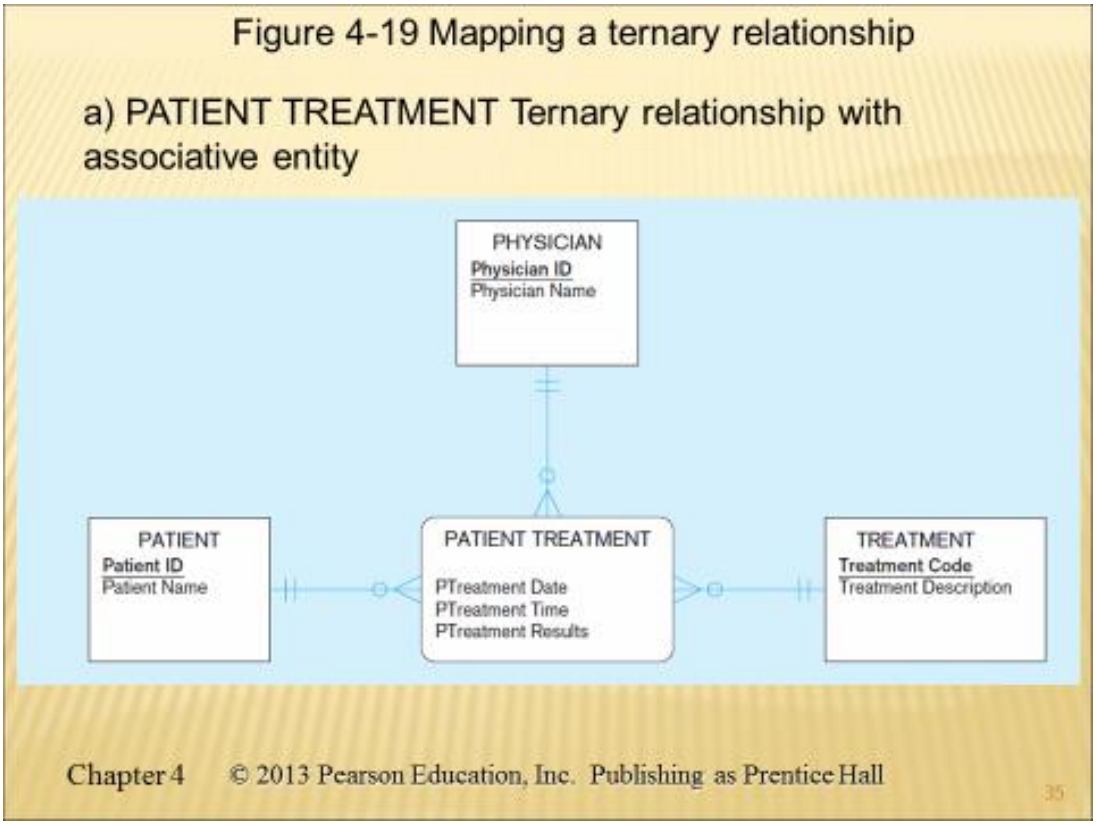
Mapping Many-to-Many (M:N) UNARY Entities Relationships

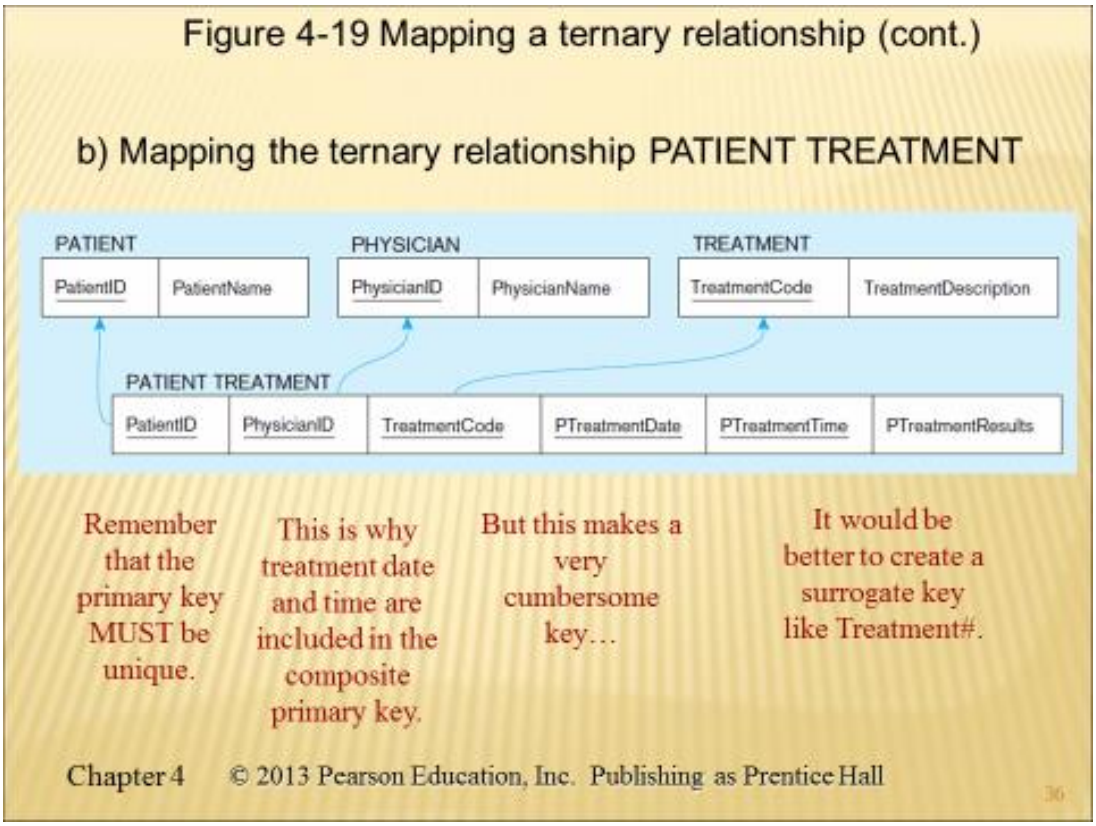


Mapping TERNARY Entities Relationships



Mapping TERNARY Entities Relationships with Associate Entity





2.1 Transforming EER Diagrams – Transforming Supertypes/Subtypes

- Transforming:

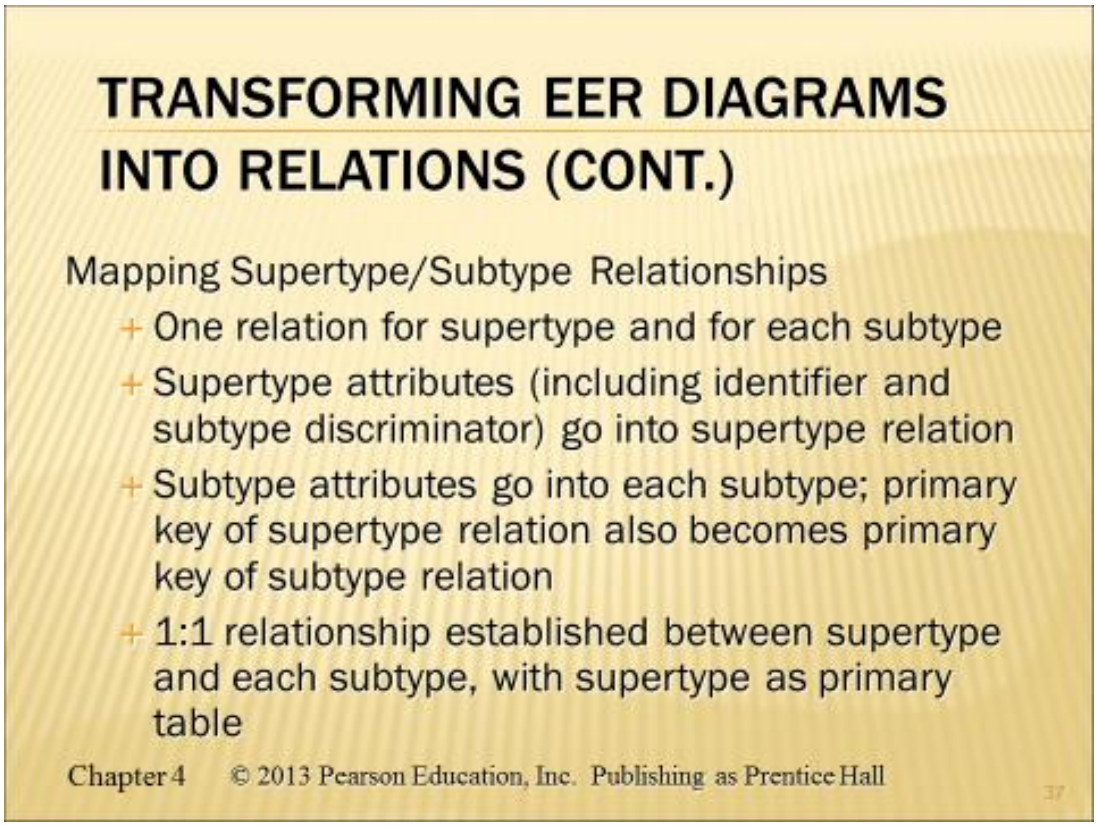


Figure 4-20 Supertype/subtype relationships

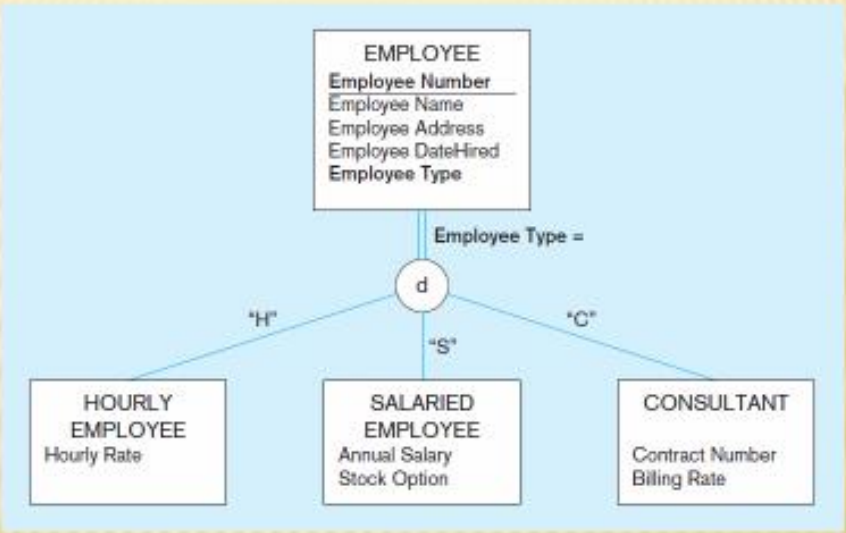
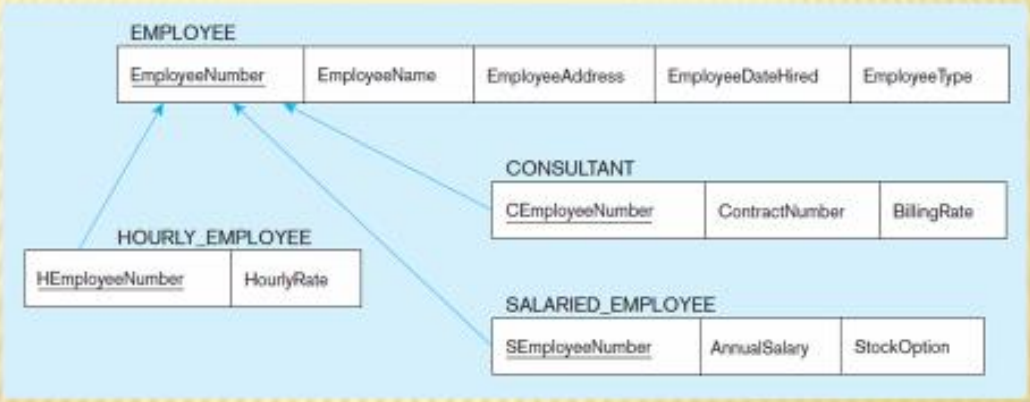


Figure 4-21
Mapping supertype/subtype relationships to relations

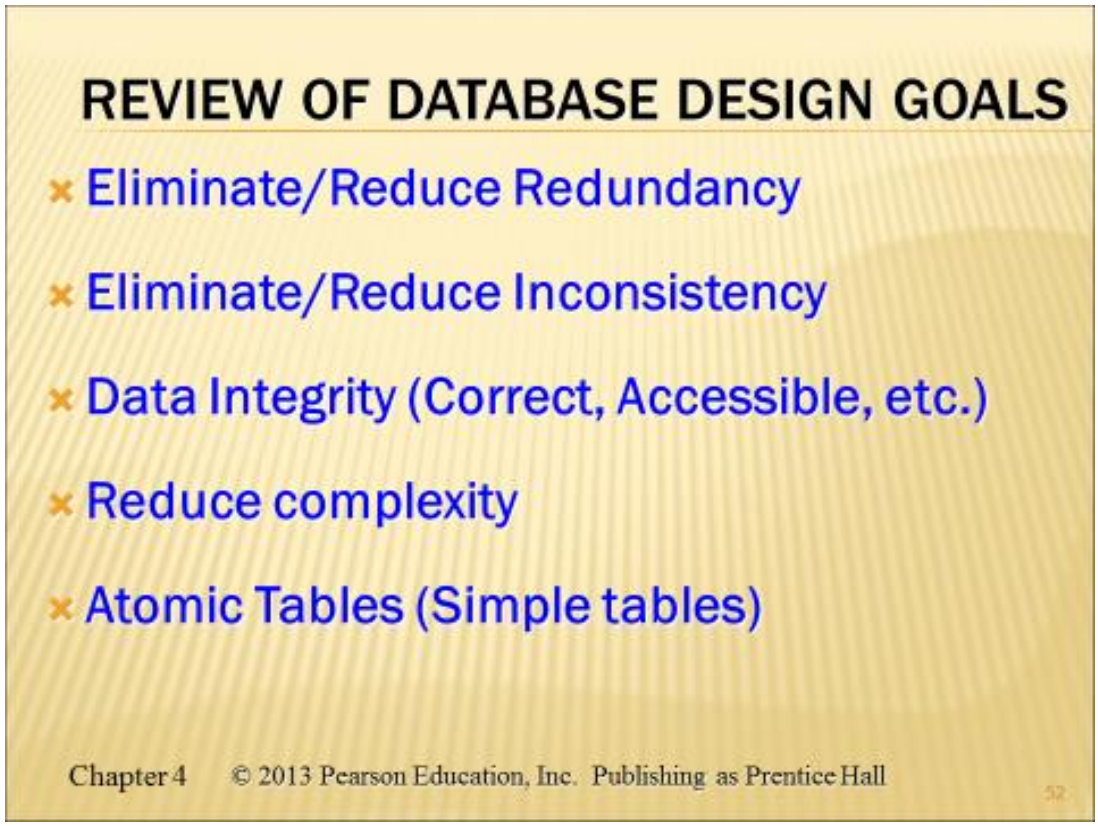


These are implemented as one-to-one relationships.

3.0 Database Design Phase – Normalization

3.1 Normalization Defined

- Normalization:



Data Normalization Defined

DATA NORMALIZATION

- Normalization:
  - The process of decomposing tables with anomalies to smaller, *well-structured* tables
- Normalization use:
  - Primarily a tool to validate and improve a logical design to *avoid unnecessary duplication of data (Redundancy)*

WHEN TO USE DATA NORMALIZATION

- During logical database design:
  - Use normalization concepts as a quality check for the relations that are obtained from mapping E-R conceptual diagrams
- When engaging in reverse engineering:
  - reverse-engineering older systems Many of the tables and user views for older systems are redundant and subject to the anomalies

DATA NORMALIZATION BENEFITS

- Minimize data redundancy, thereby avoiding anomalies and conserving storage space
- Simplify the enforcement of referential integrity constraints (Business Rules)
- Make it easier to maintain data (insert, update, and delete).
- Provide a better design that is an improved representation of the real world and a stronger basis for future growth (Scalable)

3.2 Normalization – Steps

Identifying Well Structure & Abnormal Relations (Tables)

WELL-STRUCTURED RELATIONS

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
  - **Insertion Anomaly**–adding new rows forces user to create duplicate data
  - **Deletion Anomaly**–deleting rows may cause a loss of data that would be needed for other future rows
  - **Modification Anomaly**–changing data in a row forces changes to other rows because of duplication

General rule of thumb: A table should not pertain to more than one entity type. It should be ATOMIC

EXAMPLE–FIGURE 4-2B

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

Question–Is this a relation?

Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key?

Answer–Composite: EmpID, CourseTitle

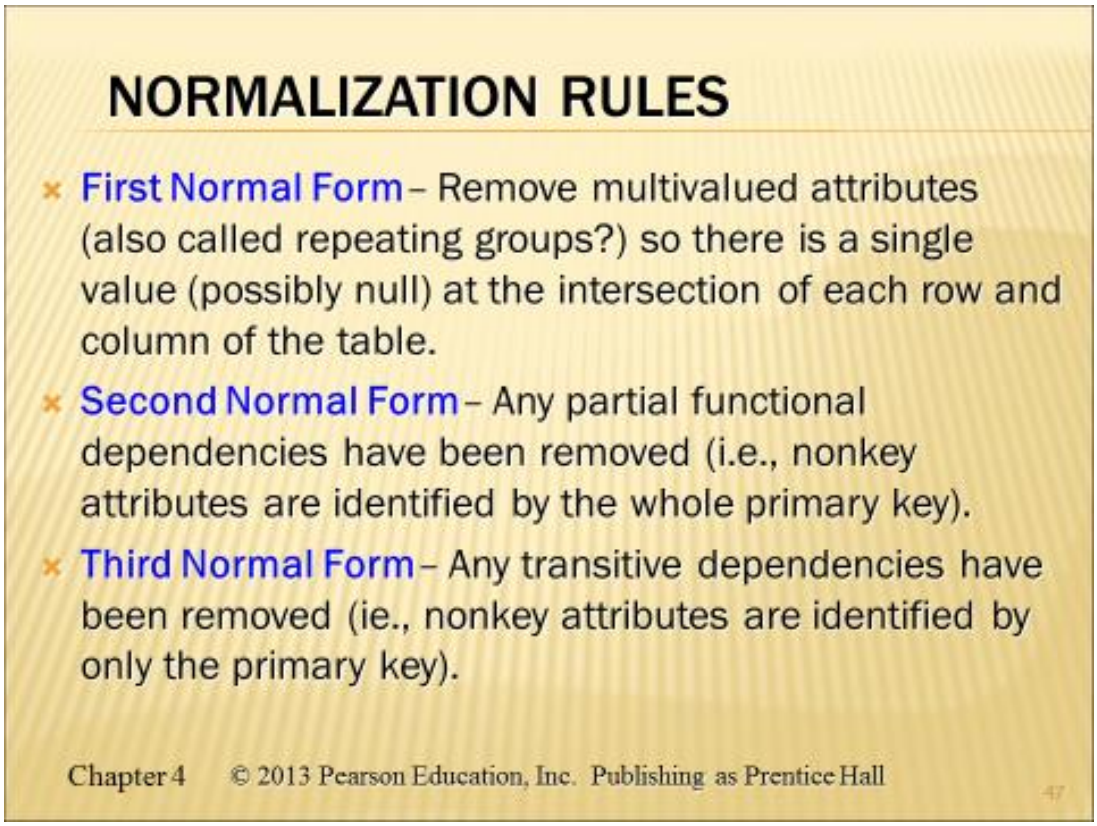
ANOMALIES IN THIS TABLE

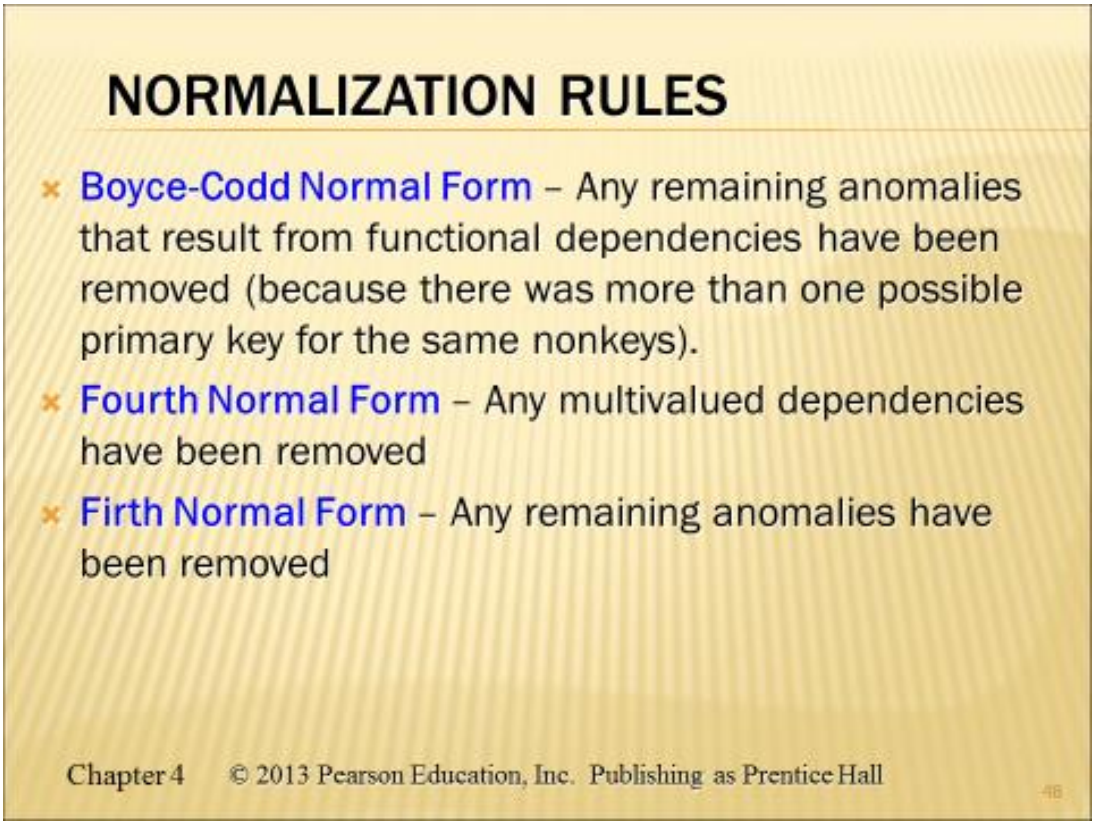
- **Insertion**–can't enter a new employee without having the employee take a class (or at least empty fields of class information)
- **Deletion**–if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification**–giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

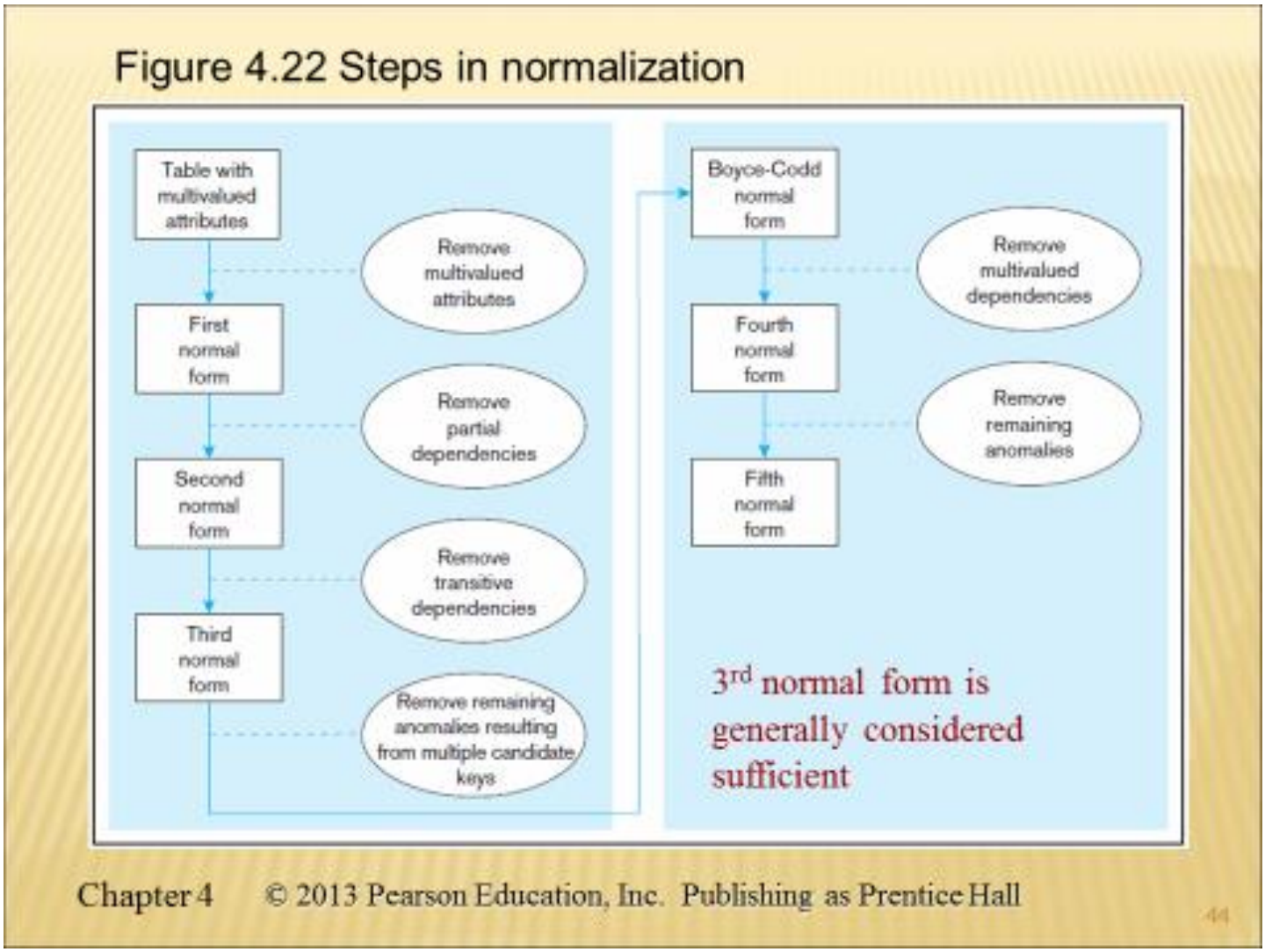
Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities.

Normalization Steps & Rules





Normalization High-level Steps Flow Diagram



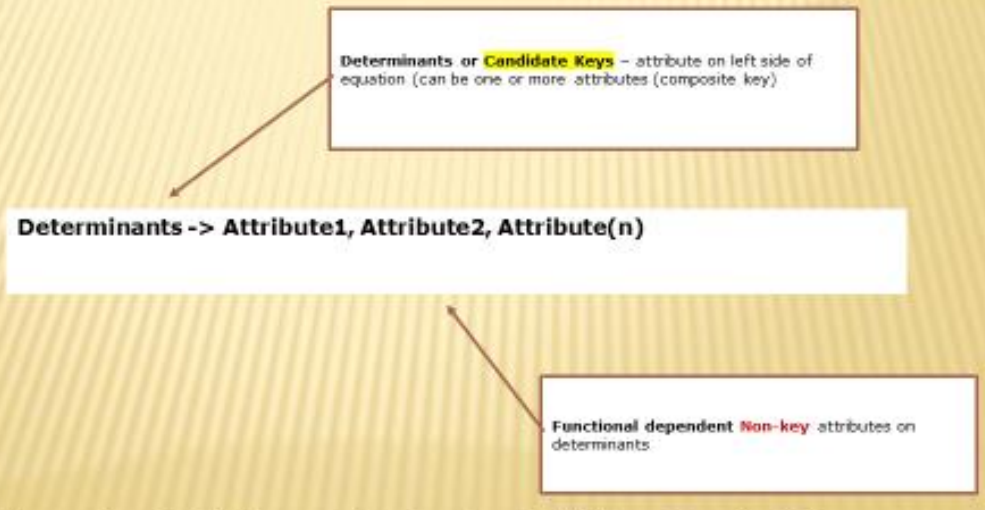
Normalization Foundamentals

FUNCTIONAL DEPENDENCIES AND KEYS

- Normalization is based on the analysis of functional dependency
- Functional Dependency: The value of one attribute (the *determinant*) determines the value of another attribute
- Candidate Key: An Attribute or combination of attributes that uniquely identifies a row in relationship.
  - Candidate keys will become the primary key
  - Each non-key field is functionally dependent on every candidate key.

FUNCTIONAL DEPENDENCIES AND KEYS

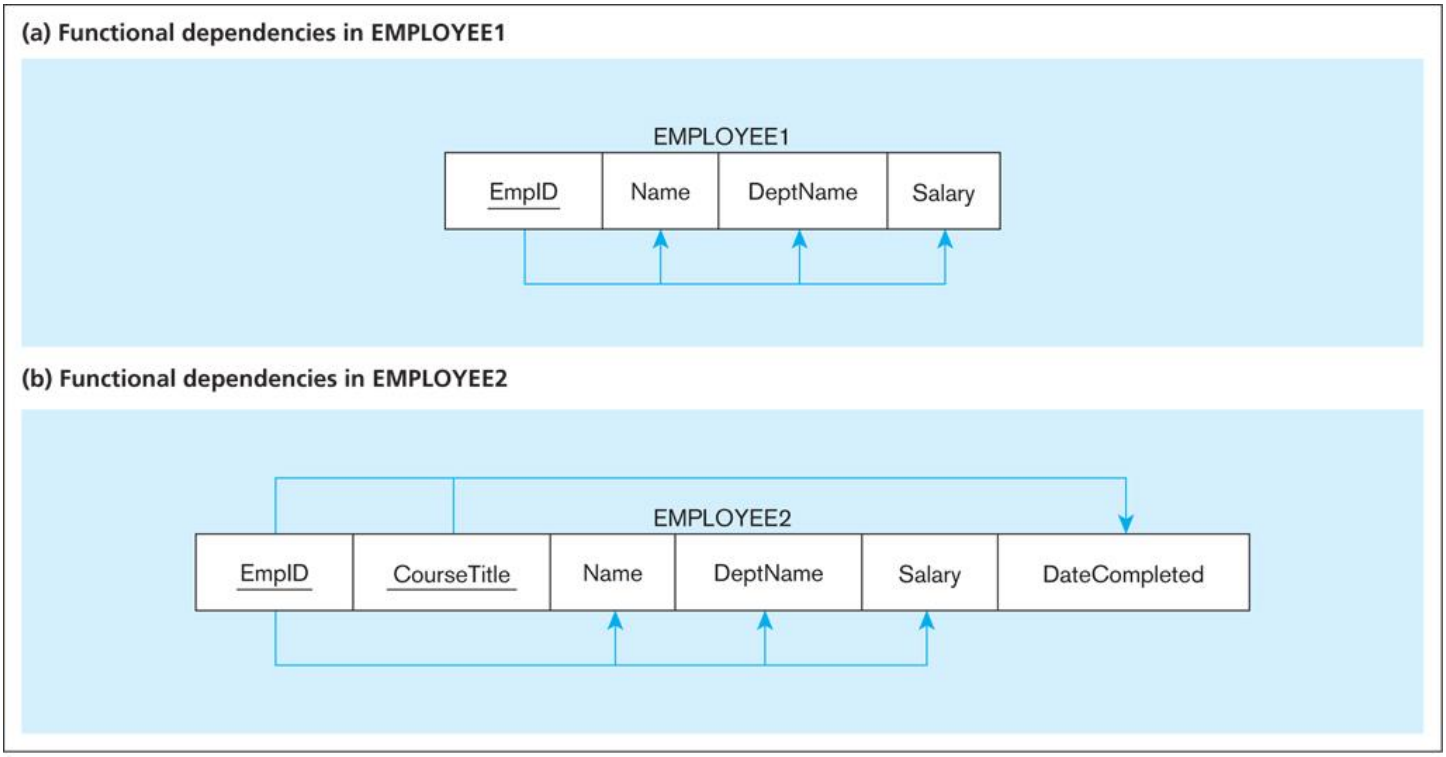
- Functional Dependency Equation:



FUNCTIONAL DEPENDENCIES AND KEYS

- Examples of Functional Dependency:
  - *EmpID, CourseTitle -> DateCompleted*
    - **DateCompleted** is functionally dependent on both *EmpID* & *CourseTitle* since DateCompleted is determined by the identity of employee & title of course.
  - *SSN -> Name, Address, Birthdate*
    - **Name, address & birthdate** are functionally dependent on **SSN**. There can only be one name, address & birthdate for each SSN.
  - *VIN -> Make, Model, Color*
    - **Make, model & color** are functionally dependent on **VIN**. There can only be one make, model & color for each VIN.

Identifying Candidate Keys by determining Functional Dependencies:



- **EMPLOYEE1(EmpiD, Name, DeptName, Salary). EmpiD is the only determinant in this relation. All of the other attributes are functionally dependent on EmpiD. Therefore, EmpiD is a candidate key and (because there are no other candidate keys) also is the primary key.**

- **The functional dependencies indicate that the combination of EmpiD and Course Title is the only candidate key (and therefore the primary key) for EMPLOYEE2. In other words, the primary key of EMPLOYEE2 is a composite key. Neither EmpiD nor CourseTitle uniquely identifies a row in this relation and therefore**

Relationship between DETERMINANTS & CANDIDATE KEYS

- Relationship between determinants and candidate keys as follows:
  - A candidate key is always a determinant, whereas a determinant may or may not be a candidate key.
  - A candidate key is a determinant that uniquely identifies the remaining (nonkey) attributes in a relation (Example Emp1 in EMPLOYEE1 above)
  - A determinant may be a candidate key (such as EmpiD in EMPLOYEE I), part of a composite candidate key (such as EmpiD in EMPLOYEE2), or a nonkey attribute

Normalization Example – Pine Valley Furniture Company

Customer Invoice at Pine Valley Furniture & its Form View we wish to create a database for:

PVFC Customer Invoice

Customer ID	2	**Order ID**	1006
Customer Name	Value Furniture	**Order Date**	10/24/2010
Address	15145 S.W. 17th St. Plano TX 75022		

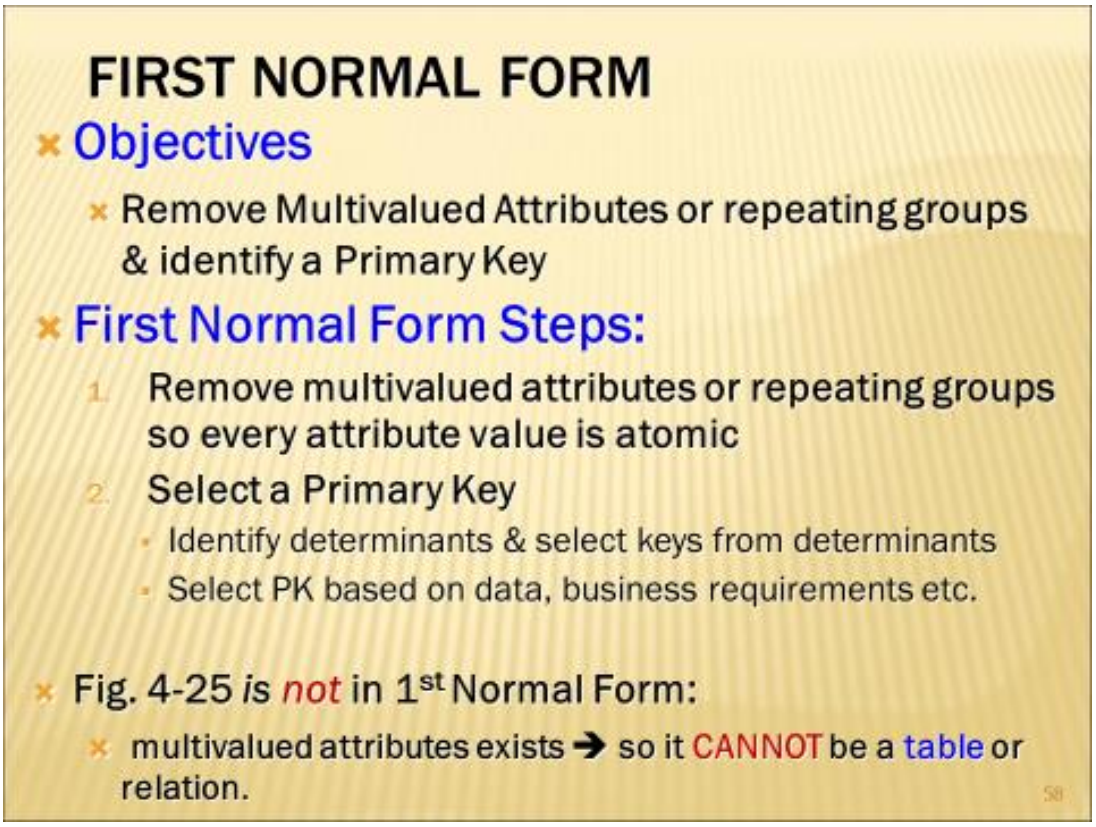
Product ID	Product Description	Finish	Quantity	Unit Price	Extended Price
7	Dining Table	Natural Ash	2	$800.00	$1,600.00
5	Writer's Desk	Cherry	2	$325.00	$650.00
4	Entertainment Center	Natural Maple	1	$650.00	$650.00
				Total	$2,900.00

First Attempt to put it all inside a Table:

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

STEP 1 – Convert First Normal Form

Step 1 – Convert to First Normal Form



Step 1(a) – Remove Repeating Groups/Multivalued Attributes:

- The current state of this non-table is that it contains repeating groups:

Multi-value Attributes or **Repeating Groups**

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Conclusion:
This cannot qualify as a TABLE because is NOT atomic (no single-values in each cell)

Multi-value Attributes or **Repeating Groups**

1) We remove the repeating groups or multivalued attributes by **filling in the relevant data values into the vacant cells**,
   - As result, we are left with single-valued attributes thus satisfies the atomic property of a relation or table:

All rows are now **Single-value** attributes

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Step 1(b) – Select a Primary Key:

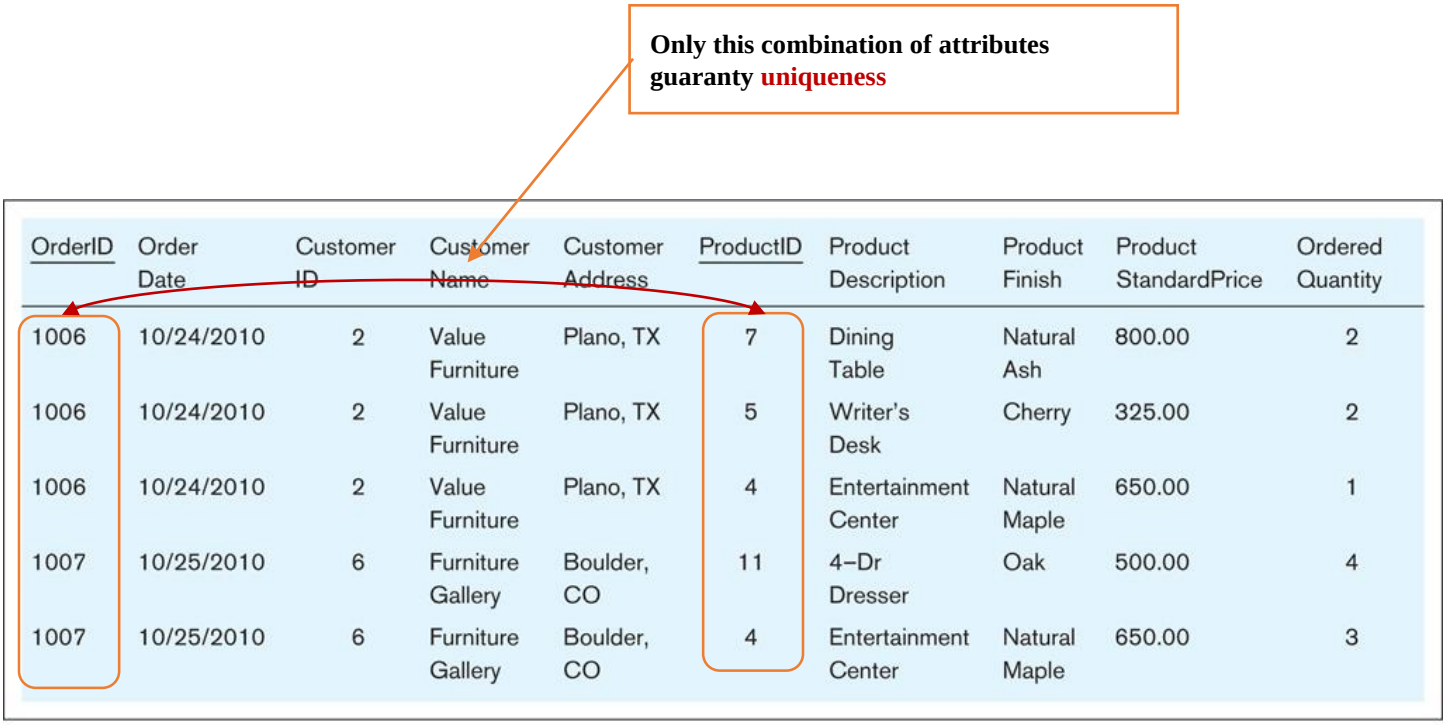
1) **First we identify Determinants based on functional dependencies of the attributes**.
   - We identify 4 determinants:

OrderID -> OrderDate, CustomeriD, CustomerName, CustomerAddress
CustomerID -> CustomerName, CustomerAddress
ProductID -> ProductDescription, ProductFinish, ProductStandardPrice
OrderID, ProductID -> OrderedQuantity

Determinants are in Right-Hand side of equation

2) Review the data in each row to identify which value or combination of value between these 4 Determinants identifies the row as UNIQUE.

- We notice that only the combination of **OrderID** and **ProductID** guarantees uniqueness.
- Therefore the combination or composite of **OrderID** and **ProductID** is the PRIMARY KEY



OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

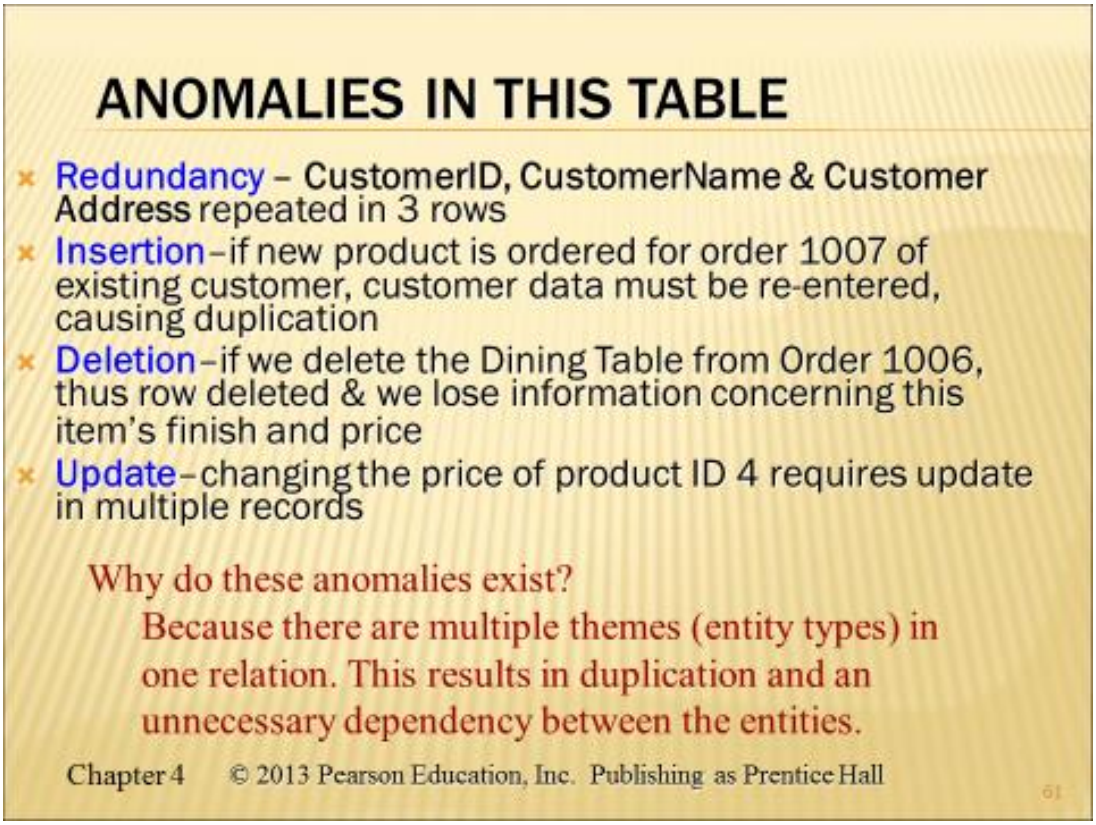
Results of 1st Normal Form – The new Invoice Table is created:

Invoice Table is now in First Normal Form (no repeating groups/Multivalued, Primary Key Identified)

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Status of Table after First Normal Form is applied

INVOICE Table Current Status – Abnormalities after Applying First Normal Form:



Delete this product, delete row, thus loose information on Dining Table

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

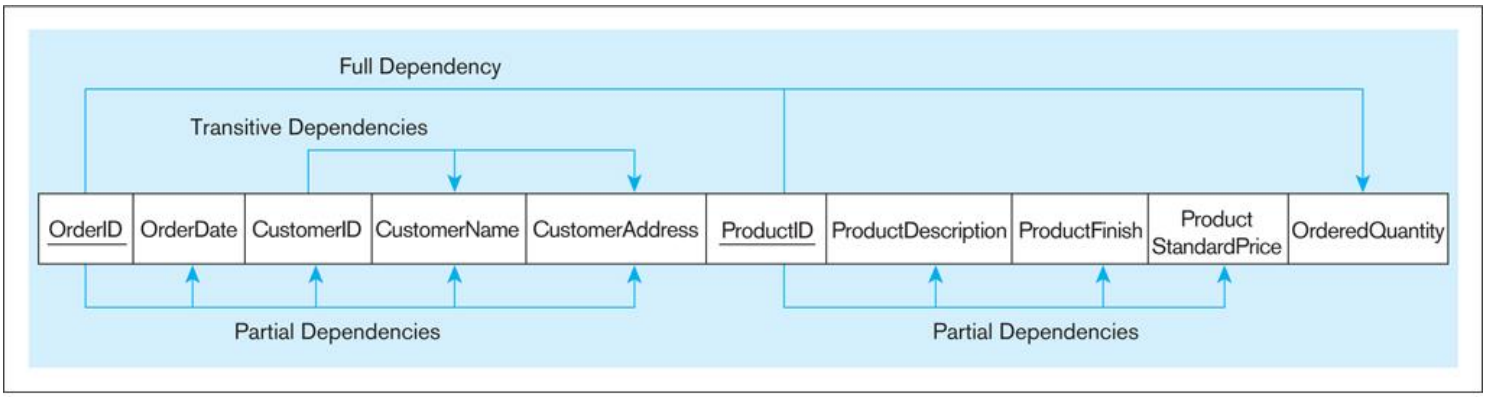
Redundancy

Update Price. You must update in two or more places

Analysis of Functional Dependencies in the INVOICE TABLE

First – identify from the First Normal Form, that the Primary Key is a Composite Key or Combination of OrderID & ProductID

Second – Analysis of each dependency:



First Functional Dependency – Attributes Dependent only on Part of the Primary Key:

OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

- Attributes, **OrderDate**, **CustomerID**, **CustomerName** & **CustomerAddress** are dependent on **OrderID** the **first** part of **Primary Key**) but have **NOTHING TO DO** with **ProductID** the **second** half of the **Primary Key**.

Second Functional Dependency – Non-Key Attributes Dependent on other non-key attribute:

CustomerID ➔ CustomerName, CustomerAddress

- Attributes, **CustomerName** & **CustomerAddress** are also dependent on **CustomerID** **WHICH IS NOT** a **Primary Key**).

Third Functional Dependency – Attributes Dependent only on Part of the Primary Key:

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

- Attributes, **ProductDescription**, **ProductFinish** & **ProductStandardPrice** are dependent on **ProductID** the **second** part of **Primary Key**) but have **NOTHING TO DO** with **OrderID** the **first** half of the **Primary Key**.

Fourth Functional Dependency – Attribute with FULL Dependency on Primary Key:

OrderID, ProductID ➔ OrderQuantity

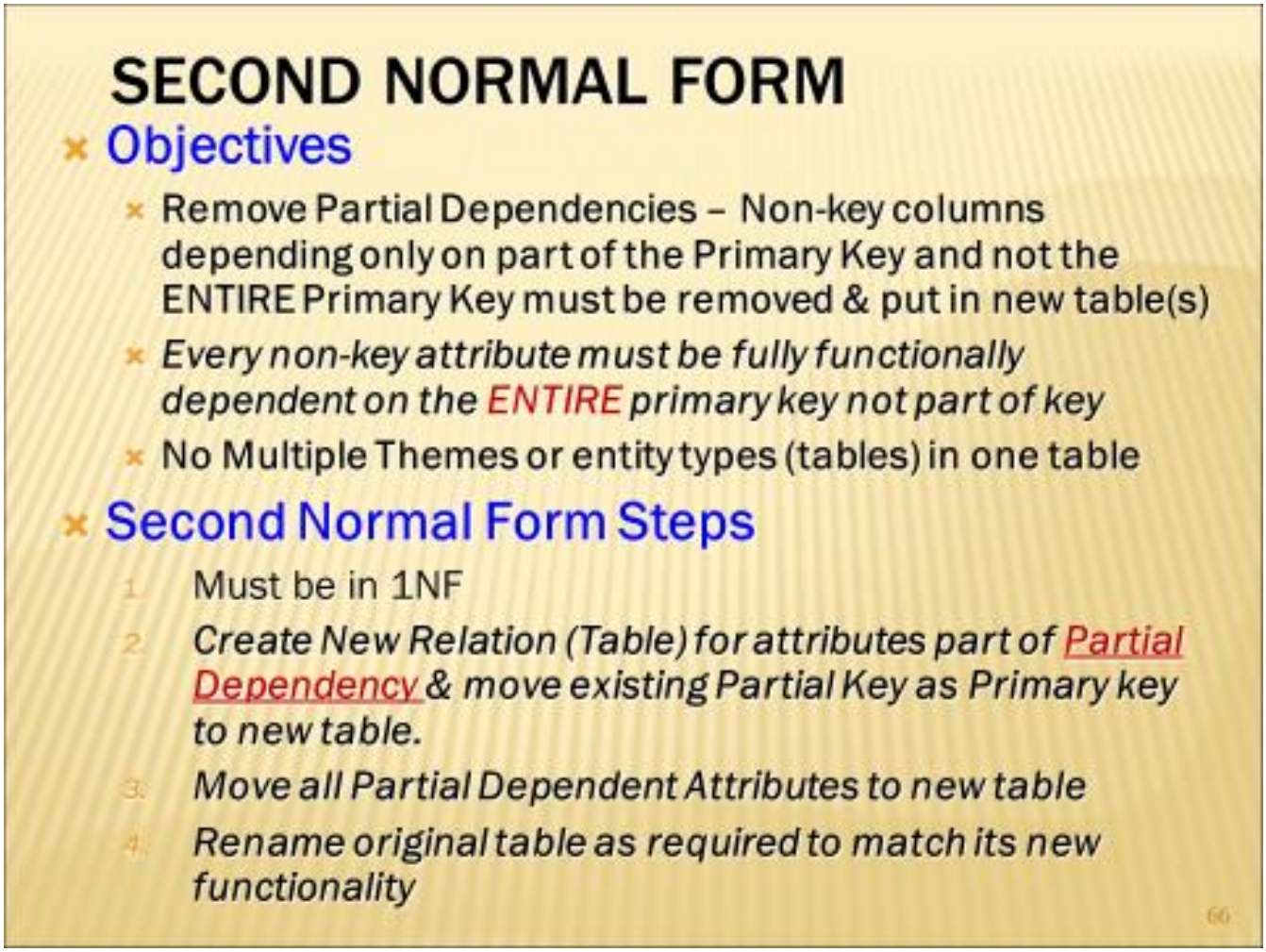
- Attributes, **OrderQuantity** is dependent on both **OrderID** & **ProductID** tor the **FULL Composite Primery Key**.
- **OrderQuantity** is the **ONLY ATTRIBUTE** that **FULLY DEPENDS ON THE PRIMARY KEY!!!!**

Third – Based on Functional dependencies above, we have the following types of dependencies after First Normal Forms:

- **Partial Dependencies** – Non-Key Columns/Attributes depending on PART of Primary Key not whole Primary Key
- **Transitive Dependencies** – Non-Key Columns/Attributes depending on other Non-Key Attributes
- **Full Dependency** – Non-Key Columns/Attributes fully depends on Primary Key

STEP 2 – Convert Second Normal Form

Step 2 – Second Normal Form Objectives



Functional Dependency – Partial Dependency Defined and Analyzed

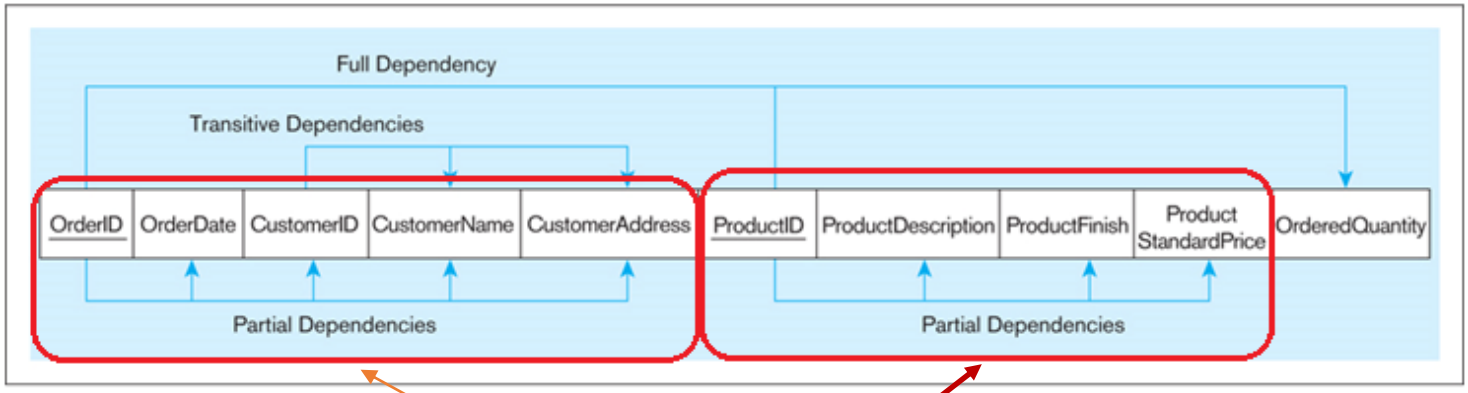
PARTIAL DEPENDENCY

- **Partial Functional Dependency:**
  - One or more non-key columns are functionally dependent on PART and NOT ENTIRE Primary Key
- **Current status of INVOICE table:**
  - NOT in **2nd Normal Form** since as shown in Figure 4-27 we have the following Partial Dependencies on table:

OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress

ProductID → ProductDescription, ProductFinish, ProductStandardPrice

67



Partial Dependencies – these attributes do not depend on both OrderID & ProductID combined (Primary Key) but only to one or the other.

Step 2 – Steps to Convert to 2nd Normal Form

Step 2(a) – Must Be in 1st Normal Form:

- **INVOICE Table is in 1st Normal Form:**

OrderID	OrderDate	CustomerID	CustomerName	CustomerAddress	ProductID	ProductDescription	ProductFinish	Product StandardPrice	OrderedQuantity

Step 2(b) – Create a New Relationship for all Attributes Participating in Partial Dependency:

1) Create a NEW Relation (Table) for each PRIMARY KEY Attribute (or combination of Attribute) that is the Determinant or a Partial Dependency
2) Make it Primary Key of the NEW Relation (Table)
3) Give the Table a Name

OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

Determinant (Primary key) of this Partial Dependency

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

Determinant (Primary key) of this Partial Dependency

Results of Step 2(b) – We have two new tables with their own Primary Key:

CUSTOMER ORDER

OrderID

PRODUCT

OrderID

Step 2(c) – MOVE the non-key Attributes that were part of Partial Dependency to the NEW Table:

1) **MOVE all Functionally Dependent Attribute part of the Partial Dependency to the NEW Relation (Table)**

OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

Non-Key Partial Dependent Attributes

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

Non-Key Partial Dependent Attributes

Results of Step 2(c) – Two New Tables Were Created from Invoice Table:

CUSTOMER ORDER

OrderID	OrderDate	CustomerID	CustomerName	CustomerAddress

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice

Step 2(d) – Rename the remaining INVOICE table if required:

1) **Rename original INVOICE table to match its new functionality.**

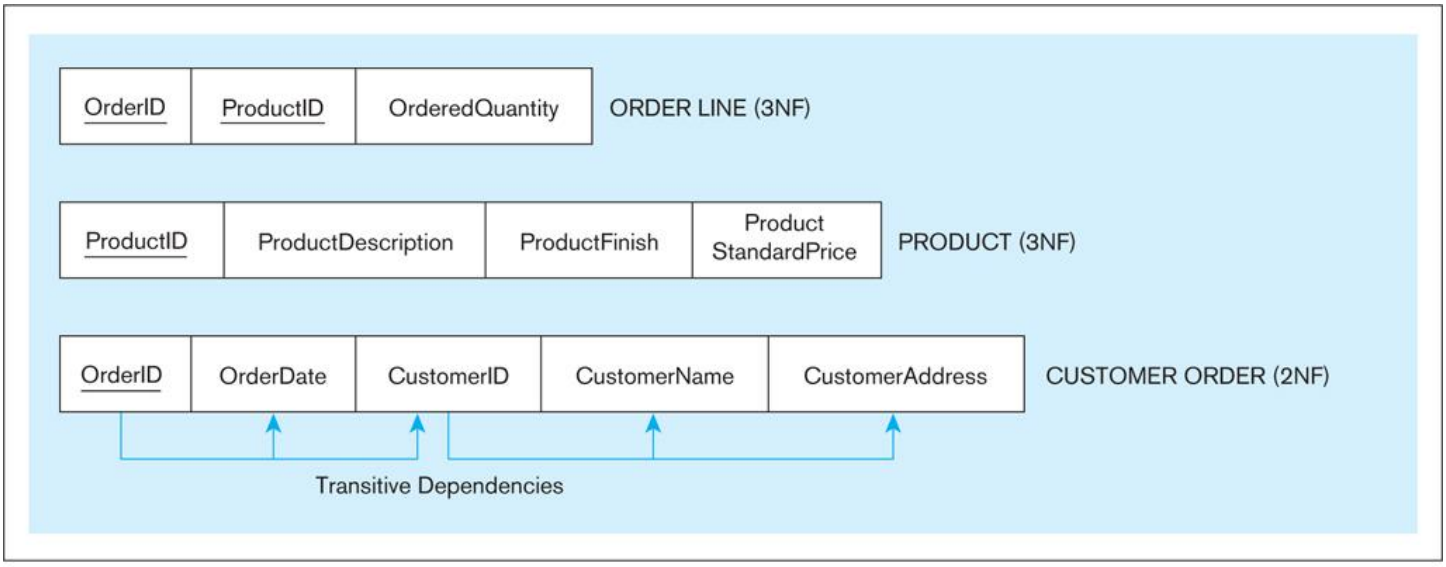
- Since what remains of the **INVOICE** table is just **OrderID**, **ProductID** & **OrderedQuantity**, we rename this table, **ORDER LINE** because each row of this table represents one line item of an order.

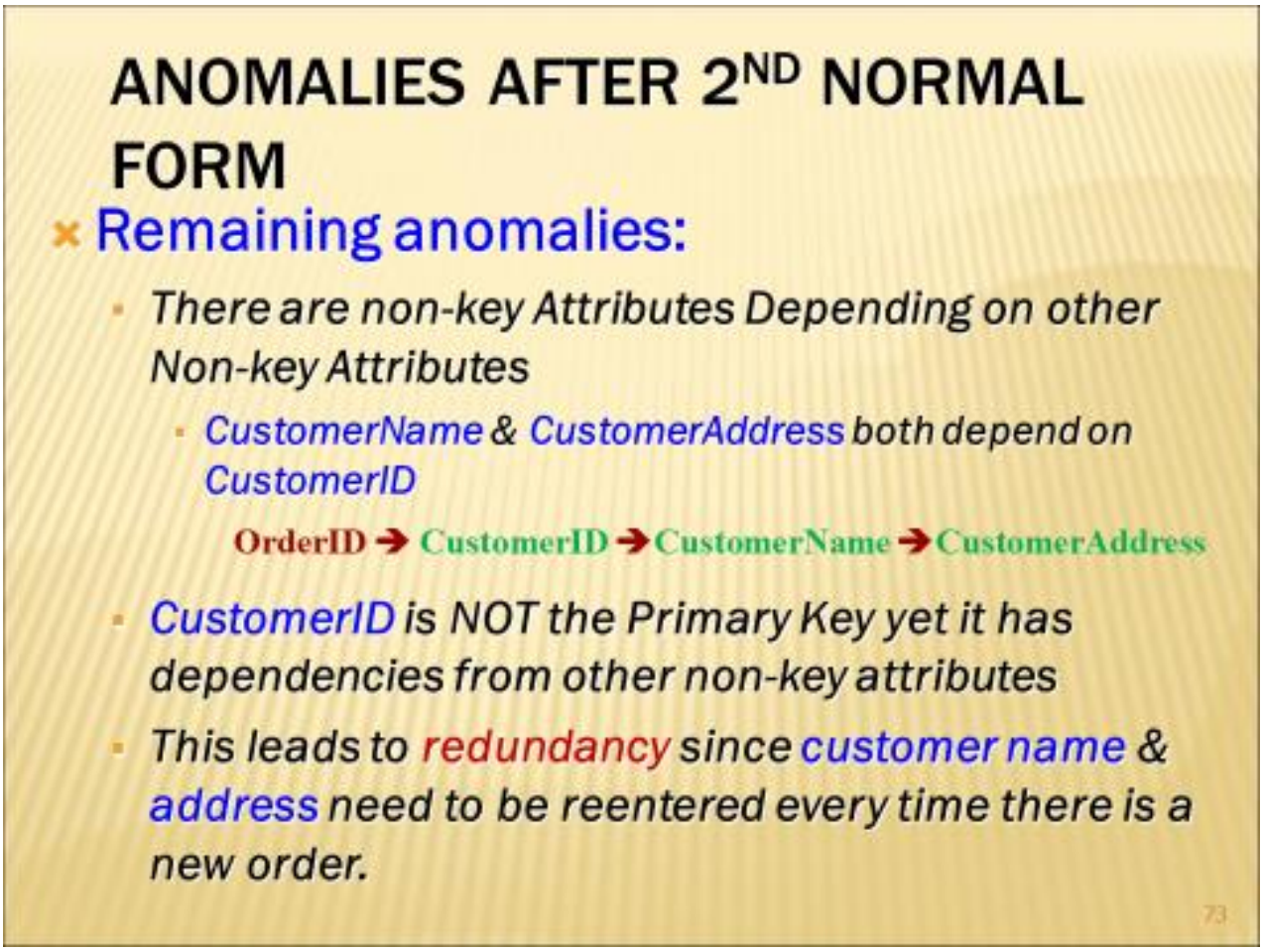
ORDER LINE

OrderID	ProductID	OrderedQuantity

Status of Tables after Second Normal Form is applied

INVOICE Table Divided into 3 Tables After Applying 2nd Normal Form:





STEP 3 – Convert to Third Normal Form

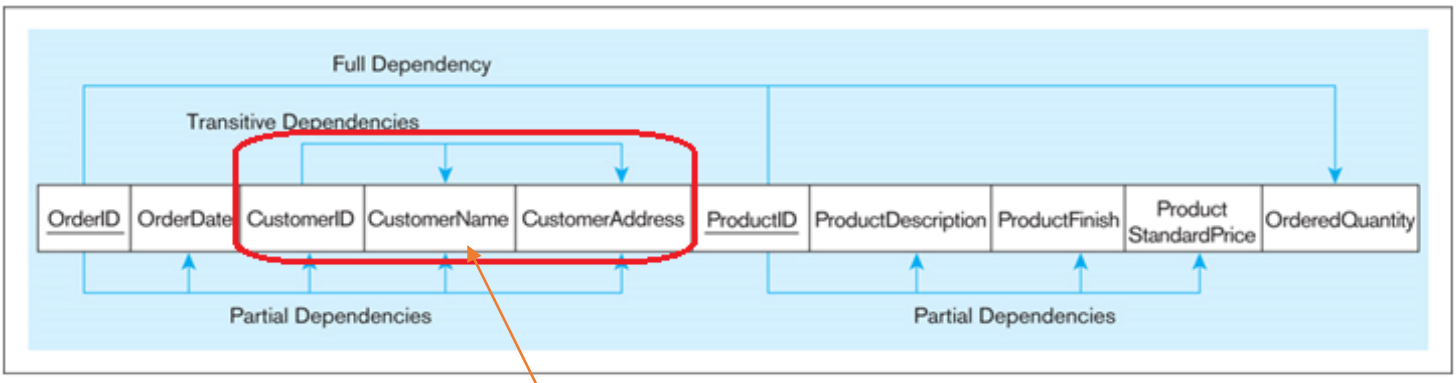
Step 2 – 3rd Normal Form Objectives

Functional Dependency – Transitive Dependency Defined and Analyzed

TRANSITIVE DEPENDENCY

- **Transitive Functional Dependency:**
  - One or more non-key columns are functionally dependent on Primary Key but via another non-key Attribute
  - In other words, non-key columns depend on another non-key column
- **Current status of INVOICE table:**
  - NOT in **3rd Normal Form** since as shown in Figure 4-28 the following Transitive Dependencies on table:
    OrderID → CustomerID → CustomerName → CustomerAddress
  - CustomerID is not part of Primary Key

74

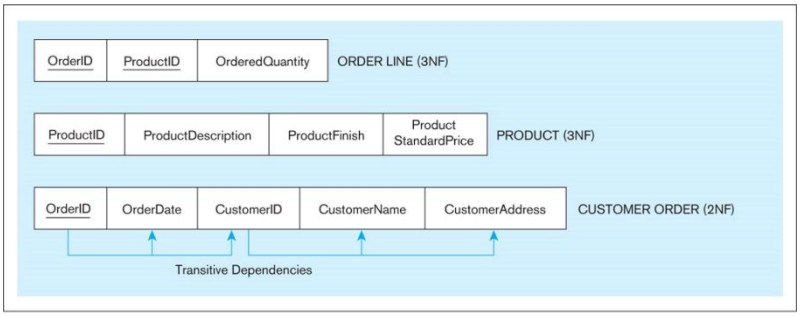


Transitive Dependencies – these attributes CustomerName & CustomerAddress ALSO depend on CustomerID (Non-key attribute) in addition to Primary Key OrderID .

Step 3 – Steps to Convert to 3rd Normal Form

Step 3(a) – Must Be in 2nd Normal Form:

- **All Tables are in 2nd Normal Form:**



Step 3(b) – Create a New Relationship for all Attributes Participating in Partial Dependency:

1) **Create a NEW Relation (Table) for Non-Key Attribute that is a Determinant or acting like a Primary key (when is not) to other Non-Key Attributes that are dependent on it**
2) **Make it Primary Key of the NEW Relation (Table)**
3) **Give the Table a Name**
4) **Move all dependent Non-Key Attribute in original table to the new table**
5) **Leave the original Non-Key Attribute or Determinant in the Old table to serve as a Foreign Key**

Results of Step 2(b) – We have a new tables with the non-key Determinant of old table as a Primary Key:

OrderID ➔ CustomerID ➔ CustomerName ➔ CustomerAddress

Determinant to other Non-key attributes who depend on it such as CustomerName & CustomerAddress

CustomerID is Primary Key in NEW table

CUSTOMER

CustomerID	CustomerName	CustomerAddress

CustomerID is Foreign Key in original table

ORDER

OrderID	OrderDate	CustomerID

Review & Status of INVOICE table after Second Normal Form is applied

Customer Invoice at Pine Valley Furniture & its Form View we wish to create a database for:

PVFC Customer Invoice

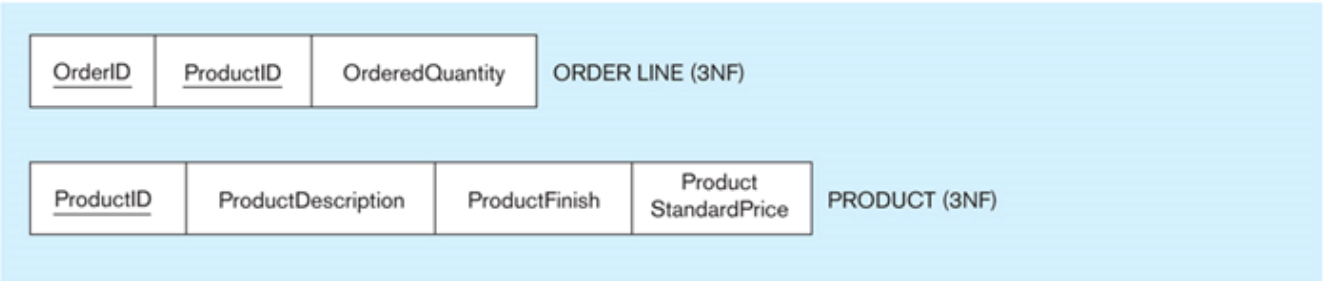
Customer ID	2	**Order ID**	1006
Customer Name	Value Furniture	**Order Date**	10/24/2010
Address	15145 S.W. 17th St. Plano TX 75022		

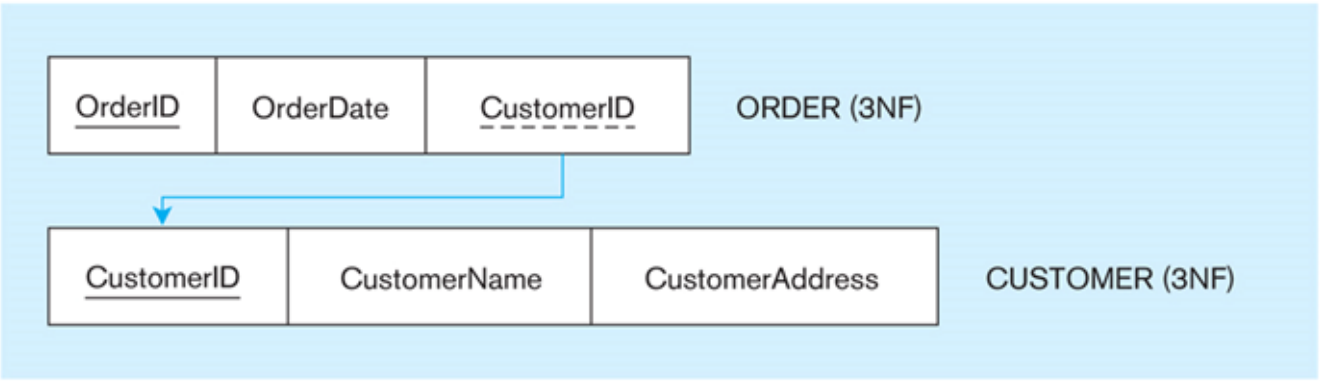
Product ID	Product Description	Finish	Quantity	Unit Price	Extended Price
7	Dining Table	Natural Ash	2	$800.00	$1,600.00
5	Writer's Desk	Cherry	2	$325.00	$650.00
4	Entertainment Center	Natural Maple	1	$650.00	$650.00
				Total	$2,900.00

First Attempt to put it all inside a Table:

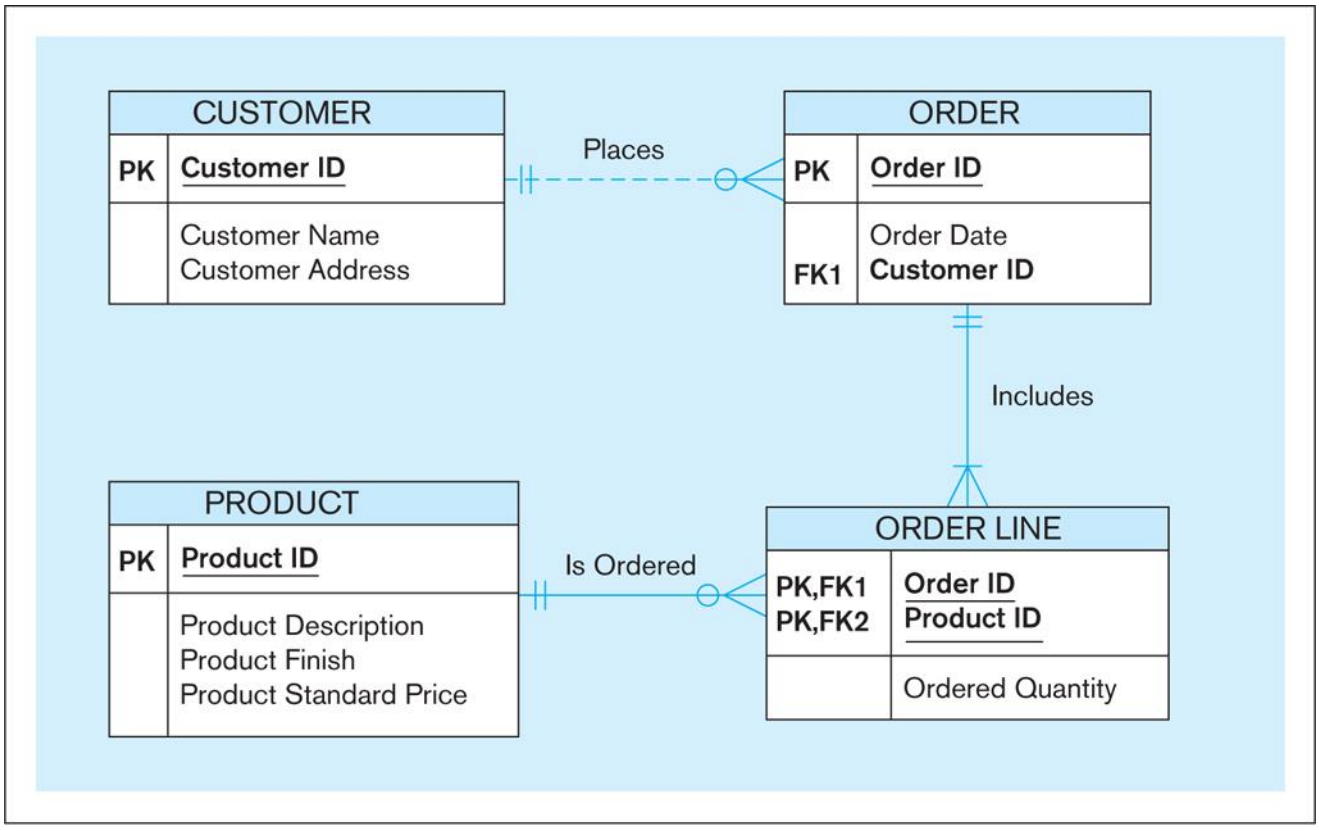
OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Invoice Table was broken down into 4 Table after 1st, 2nd & 3rd Normal Forms:





Relational Schema using MS Visio for the Invoice Data

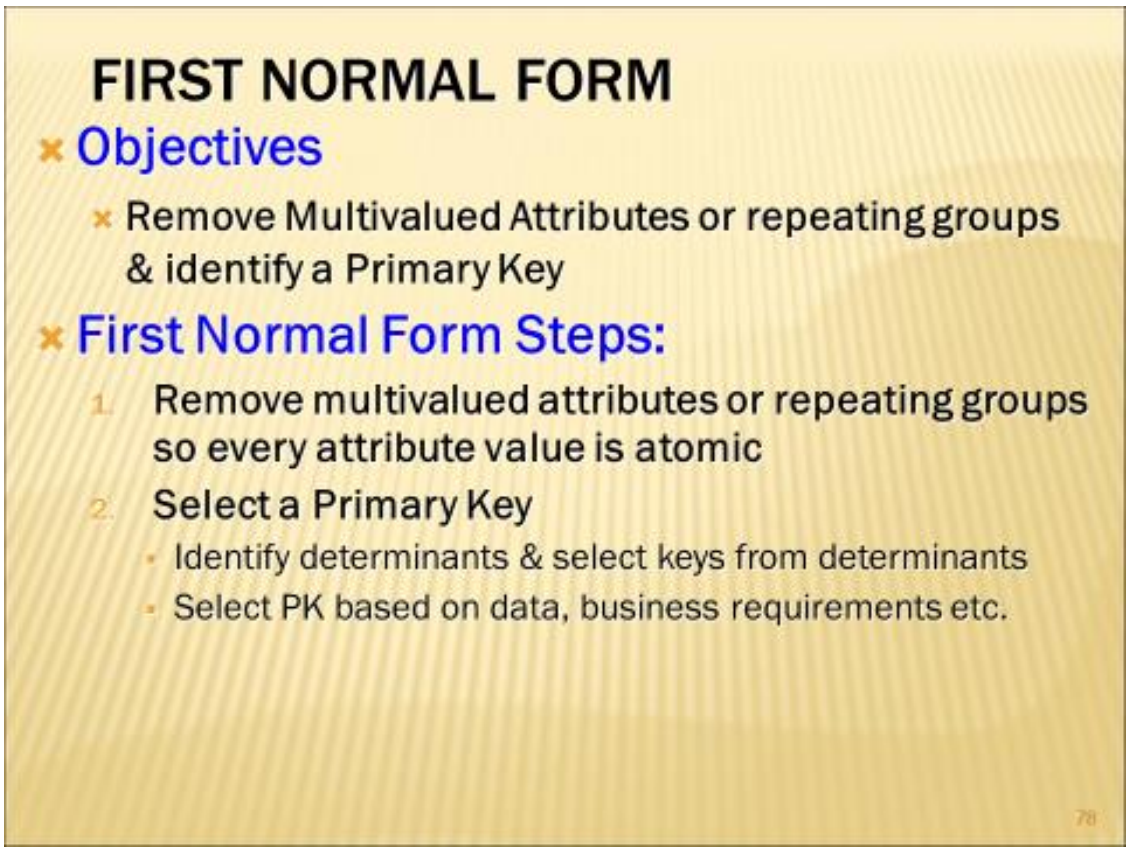


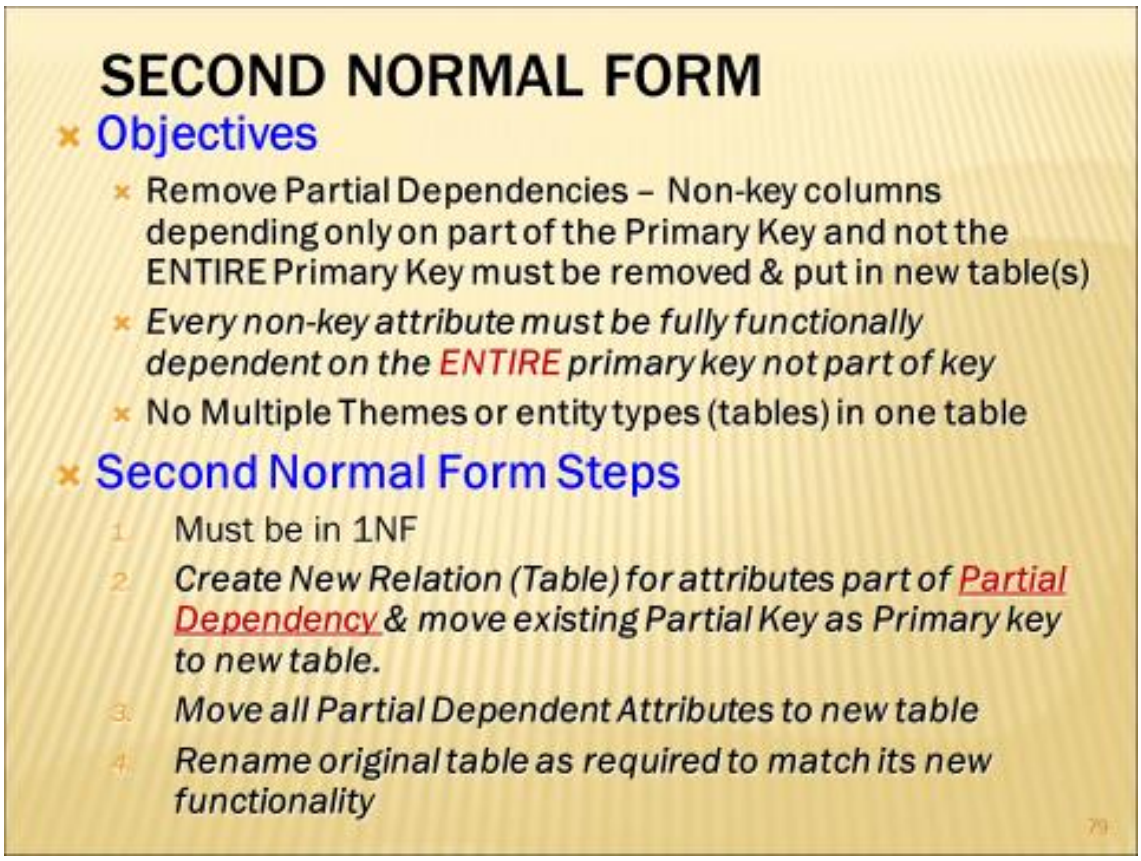
Boyce-Codd, Fourth, & Firth Normal Forms

- **These other types of Normal Forms will be Out of Scope for this Course**

3.2 Normalization – Summary

- Summary:





THRID NORMAL FORM

- Objectives
  - Remove Transitive Dependencies – Non-key attributes/columns depending on another non-key Attribute.
  - No Non-key columns should depend on another non-key Attribute, it should only depend on the Primary Key
  - *Any non-key attribute that depend on other non-key Attribute should be moved to a new table*
- Second Normal Form Steps
  1. Must be in 2NF
  2. *Create a NEW Relation (Table) for Non-Key Attribute that is a Determinant or acting like a Primary key (when is not) to other Non-Key Attributes that are dependent on it*
  3. *Make it Primary Key of the NEW Relation (Table)*
  4. *Give Table a name*
  5. *Move ALL dependent Non-Key Attribute in original table to the new table*
  6. *Leave the original Non-Key Attribute or Determinant in the Old table to serve as a Foreign Key*

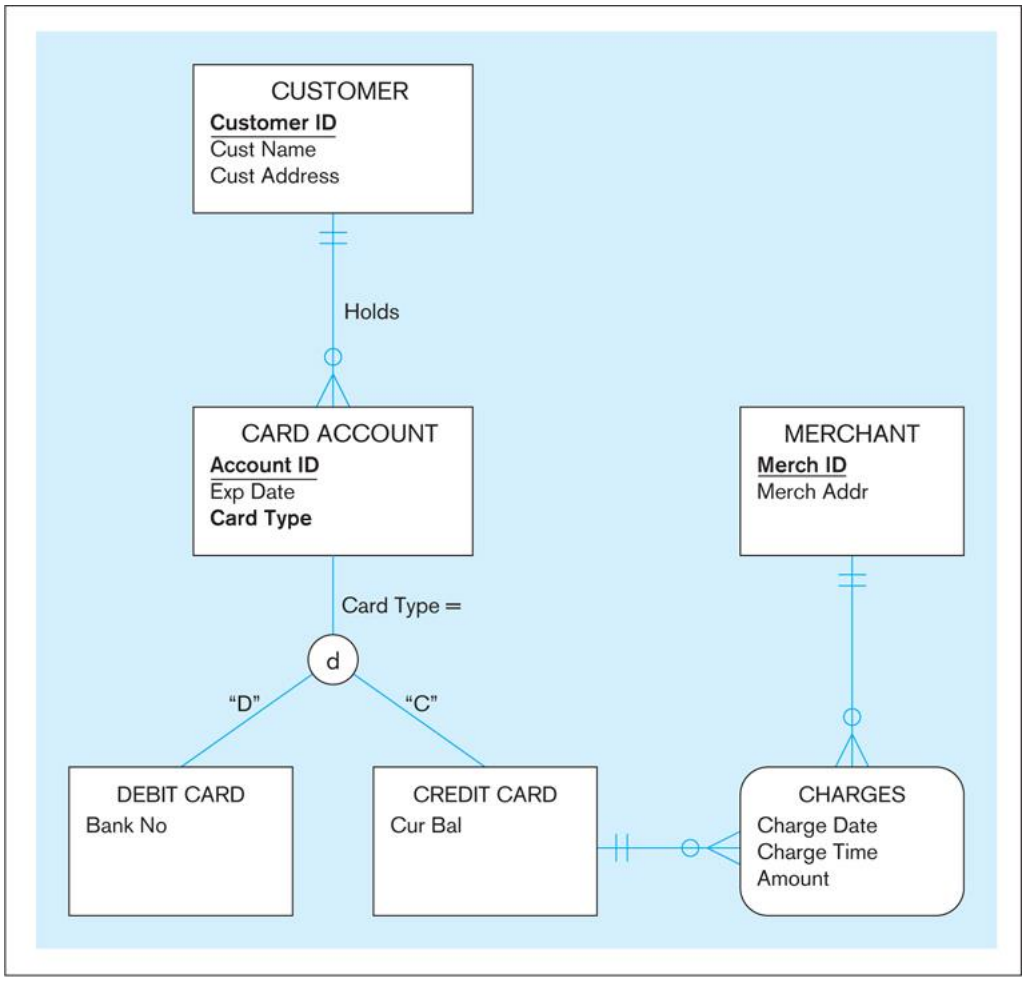
4.0 Database Design – Sample Problems

4.1 Book Problems

- Problems:

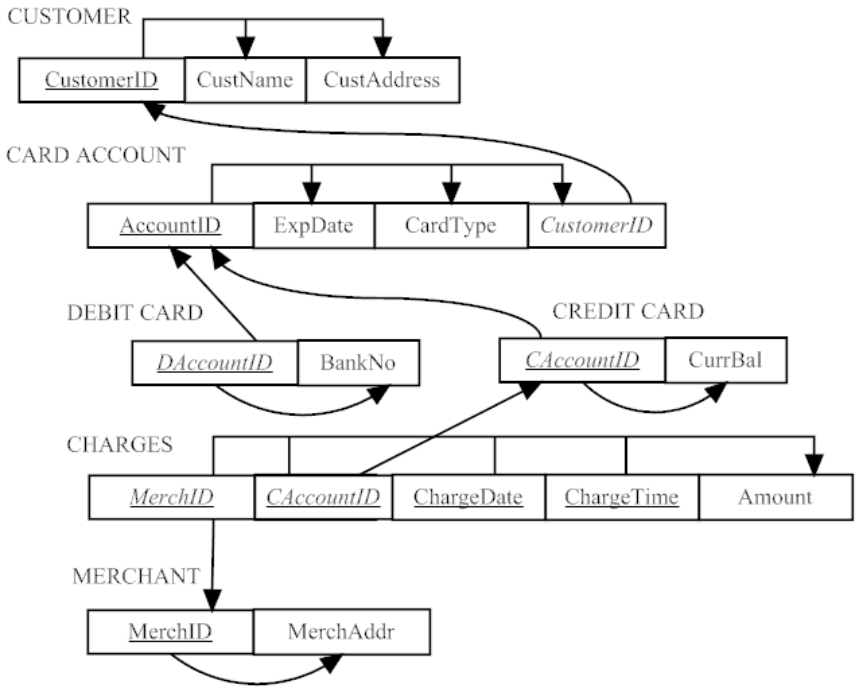
Problem #6

6. Figure 4-33 (page 194) shows an EER diagram for a simplified credit card environment. There are two types of card accounts: debit cards and credit cards. Credit card accounts accumulate charges with merchants. Each charge is identified by the date and time of the charge as well as the primary keys of merchant and credit card.
   a. Develop a relational schema.
   b. Show the functional dependencies.
   c. Develop a set of 3NF relations using an enterprise key.



Answer:

6 Develop a relational schema & Show the functional dependencies
-Transforming an E-R diagram to relations (parts a and b), Figure 33



*c. Using an enterprise key (*Foreign keys *shown in italics)*

OBJECT (OID, ObjectType)
CUSTOMER (OID, CustomerID, CustName, CustAddress)
CARD ACCOUNT (OID, AccountID, ExpDate, CardType, *CustomerID*)
DEBIT CARD (OID, DAccountID, BankNo)
CREDIT CARD (OID, CAccountID, CurrBal)
CHARGES (OID, *MerchID*, *CAccountID*, ChargeDate, ChargeTime, Amount)
MERCHANT (OID, MerchID, MerchAddr)

Problem #8

8. Table 4-4 shows a relation called GRADE REPORT for a university. Your assignment is as follows:
   a. Draw a relational schema and diagram the functional dependencies in the relation.
   b. In what normal form is this relation?
   c. Decompose GRADE REPORT into a set of 3NF relations.
   d. Draw a relational schema for your 3NF relations and show the referential integrity constraints.
   e. Draw your answer to part d using Microsoft Visio (or any other tool specified by your instructor).

9. Table 4-5 shows a shipping manifest. Your assignment is as

TABLE 4-4 Grade Report Relation

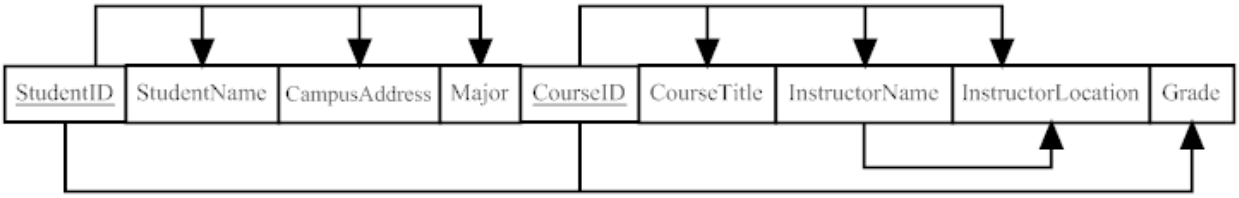
Grade Report

StudentID	StudentName	CampusAddress	Major	CourseID	CourseTitle	Instructor Name	Instructor Location	Grade
168300458	Williams	208 Brooks	IS	IS 350	Database Mgt	Codd	B 104	A
168300458	Williams	208 Brooks	IS	IS 465	Systems Analysis	Parsons	B 317	B
543291073	Baker	104 Phillips	Acctg	IS 350	Database Mgt	Codd	B 104	C
543291073	Baker	104 Phillips	Acctg	Acct 201	Fund Acctg	Miller	H 310	B
543291073	Baker	104 Phillips	Acctg	Mkgt 300	Intro Mktg	Bennett	B 212	A

Answer:

a. Draw a relational schema and diagram the functional dependencies in the relation:

Transforming Table 4 (GRADE REPORT) to relations:



b. **In what normal form is this relation?**
First normal form (1NF)

c. **Decompose GRADE REPORT into a set of 3NF relations:**
3NF relations for GRADE REPORT

STUDENT

StudentID	StudentName	CampusAddress	Major

REGISTRATION

StudentID	*CourseID*	Grade

COURSE

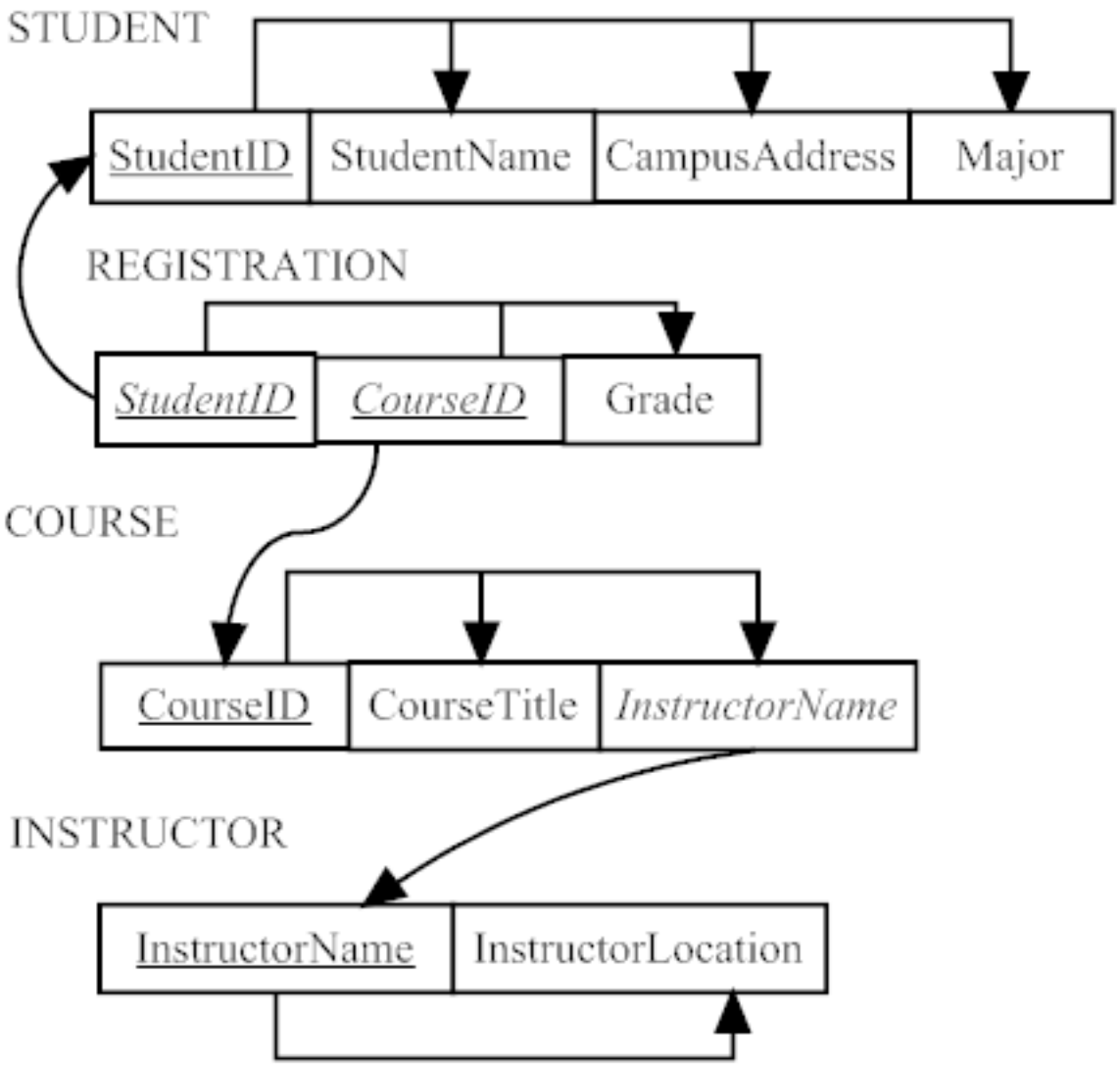
CourseID	CourseTitle	*InstructorName*

INSTRUCTOR

InstructorName	InstructorLocation

d. **Draw a relational schema for your 3NF relations and show the referential integrity constraints:**

Relational schema with functional dependencies



e. **Draw your answer to part d using Microsoft Visio (or any other tool specified by your instructor):**

An ERD to show the final version of the relations