

# **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**

**CHAPTER 1 THE DATABASE ENVIRONMENT ..... 3**

<b>1.1 Database Introduction .....</b>	<b>3</b>
1.1.1 Introduction to Database .....	3

# 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:

## THE RELATIONAL DATABASE MODEL

**★ Relational Database Model**

- DBMS that represents data in form of Tables  
(Data is stored in tables)
- Tables are called Relations (not to confuse with relationships between entities)
- Leading technology for database management systems

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

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

## COMPONENTS OF RELATIONAL MODEL

### ✖ Data structure (How data is stored/Managed)

- Tables (relations), Two-dimensional (rows, columns)

### ✖ Data manipulation

- ✚ Powerful SQL operations for manipulating data in tables (retrieving, inserting and modifying data)

### ✖ Data integrity

- ✚ Mechanisms for implementing business rules that maintain integrity of manipulated data

## 1.2. Data Structure in RDBMS – Storage Using Tables

### Data Structure (Relation/Table)

- Data Structure component (How data is stored):

### RDBMS DATA STRUCTURE (RELATION/TABLE)

- ✖ A relation is a two-dimensional table of data with a name.
- ✖ A table consists of rows (records) and columns (attribute or field).
- ✖ Requirements for a table to qualify as a relation:
  - + It must have a unique name.
  - + Every attribute value must be atomic (not multivalued, not composite).
  - + Every row must be unique (can't have two rows with exactly the same values for all their fields).
  - + Attributes (columns) in tables must have unique names.
  - + The order of the columns must be irrelevant.
  - + The order of the rows must be irrelevant.

**NOTE: All relations are in 1<sup>st</sup> Normal form.**

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

5

### CORRESPONDENCE WITH E-R MODEL

- ✖ Relations (tables) = ENTITY types



- ✖ Rows = Entity instances

Employee_ID	Dependent_Name	Dependent_Address	Birth_Date
642-17-7656	Jimmy Smith	100 E45 th Street	03/11/1989
642-17-7656	Sandy Smith	100 E45 th Street	11/07/1983
123-65-4532	Henry Jones	312 Jay Street	02/04/1995

- ✖ Columns = Attributes



- ✖ NOTE: The word *relation* (in relational database) is NOT the same as the word *relationship* (in E-R model).

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

6

## 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 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

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

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

CUSTOMER
CustomerID CustomerName CustomerAddress CustomerCity* CustomerState* CustomerPostalCode

Primary Key

ORDER
OrderID OrderDate CustomerID

Foreign Key  
(implements 1:N relationship between customer and order)

ORDER LINE
OrderID ProductID OrderedQuantity

Combined, these are a *composite primary key* (uniquely identifies the order line)... individually they are *foreign keys* (implement M:N relationship between order and product)

PRODUCT
ProductID ProductDescription ProductFinish ProductStandardPrice ProductLineID

\* Not in Figure 2-22 for simplicity.

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

### Integrity Constraints (Rules)

□ Rules:

## INTEGRITY CONSTRAINTS

### ★ What are Constraints?

- Rules limiting values assigned to attributes & limiting actions in the database

### ★ Integrity Constraints

- The relational model includes several types of constraints or rules whose purpose is to facilitate maintaining accuracy & data integrity (correct, consistent & accessible)

### ▪ Major types of integrity Constraints:

- Domain Constraints
- Entity Constraints
- Action Assertions

## 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

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

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.

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

## Entity Constraints:

### INTEGRITY CONSTRAINTS – ENTITY INTEGRITY

#### ✖ Entity Integrity

- Rule states NO primary key attribute (or component) may be NULL.
- All primary key fields **MUST** have data.

#### ✖ What is a NULL

- A value that may be assigned to an attribute when no other value applies Or when the applicable value is unknown
- A NULL is not a value, but rather it indicates the absence of a value
- NULL is NOT a numeric zero or a string of blanks

## Referential Integrity:

### INTEGRITY CONSTRAINTS – REFERENTIAL INTEGRITY

#### ✖ Referential Integrity

- Business rules implemented in the database
- Rules between two 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)
- Referential Integrity Goals:
  - Consistency among rows between two tables!

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

18

### INTEGRITY CONSTRAINTS – REFERENTIAL INTEGRITY

#### ✖ For example: Delete Rules

- 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 → not allowed for weak entities

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

19

## 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.

**Primary Key**

Customer_T					
CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
9	M and H Casual Furniture	3700 E. 22nd St.	Oklahoma City	OK	73102-2314
10	Seminole Interiors	240 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-5423
11	American Euro Lifestyles	242 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-5621
12	Battle Creek Furniture	345 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-3401
13	Heritage Furnishings	667 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-8834
14	Kaneohe Homes	112 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-2537
15	Mountain Scenes	413 W. Cypress Creek Rd.	Fort Lauderdale	FL	33309-4432

Order_T		
OrderID	OrderDate	CustomerID
1001	10/21/2010	1
1002	10/21/2010	8
1003	10/22/2010	15
1004	10/22/2010	5
1005	10/24/2010	3
1006	10/24/2010	2
1007	10/27/2010	11
1008	10/30/2010	12
1009	11/5/2010	4
1010	11/5/2010	1

OrderLine_T					
OrderID	ProductID	OrderedQuantity	UnitPrice	LineTotal	ProductLineID
1001	1001	1	\$175.00	\$175.00	1
1001	1001	2	\$175.00	\$350.00	1
1001	1001	4	\$175.00	\$700.00	1
1002	1002	3	\$200.00	\$600.00	2
1002	1002	6	\$200.00	\$1,200.00	2
1003	1003	8	\$375.00	\$3,000.00	2
1003	1003	10	\$375.00	\$3,750.00	2
1004	1004	4	\$650.00	\$2,600.00	3
1004	1004	8	\$650.00	\$5,200.00	3
1005	1005	3	\$325.00	\$975.00	1
1006	1006	2	\$750.00	\$1,500.00	2
1007	1007	7	\$800.00	\$5,600.00	2
1008	1008	8	\$250.00	\$2,000.00	3
1009	1009	10	\$250.00	\$2,500.00	3
1010	1010	8	\$250.00	\$2,000.00	3

Product_T					
ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID	Category
1	End Table	Cherry	\$175.00	1	Computer Desks
2	End Table	Ash	\$200.00	2	Computer Desks
3	End Table	Ash	\$375.00	2	Computer Desks
4	End Table	Maple	\$650.00	3	Computer Desks
5	Computer Monitor	Cherry	\$325.00	1	Monitors
6	Computer Monitor	Ash	\$750.00	2	Monitors
7	Computer Monitor	Ash	\$800.00	2	Monitors
8	Keyboard	Cherry	\$250.00	3	Peripherals

Turning ON **Referential Integrity** Rule can prevent the insertion of a row in the **ORDER** table if the **Customer ID** does NOT EXIST in the **CUSTOMER** table. Either Foreign Key MUST MATCH a Primary Key in another table or the Foreign Key MUST be a NULL.

Relationship association between **CUSTOMER & ORDER** is done via Foreign Key **CustomerID** in **ORDER** 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:

**Figure 4-6 SQL table definitions**

```
CREATE TABLE Customer_T
  (CustomerID          NUMBER(11,0)    NOT NULL,
   CustomerName        VARCHAR2(25)   NOT NULL,
   CustomerAddress     VARCHAR2(30),
   CustomerCity        VARCHAR2(20),
   CustomerState       CHAR(2),
   CustomerPostalCode  VARCHAR2(9),
   CONSTRAINT Customer_PK PRIMARY KEY (CustomerID);

CREATE TABLE Order_T
  (OrderID             NUMBER(11,0)    NOT NULL,
   OrderDate            DATE DEFAULT SYSDATE,
   CustomerID          NUMBER(11,0),
   CONSTRAINT Order_PK PRIMARY KEY (OrderID),
   CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID);

CREATE TABLE Product_T
  (ProductID           NUMBER(11,0)    NOT NULL,
   ProductDescription   VARCHAR2(50),
   ProductFinish        VARCHAR2(20),
   ProductStandardPrice DECIMAL(6,2),
   ProductLineID       NUMBER(11,0),
   CONSTRAINT Product_PK PRIMARY KEY (ProductID);

CREATE TABLE OrderLine_T
  (OrderID              NUMBER(11,0)    NOT NULL,
   ProductID            NUMBER(11,0)    NOT NULL,
   OrderedQuantity      NUMBER(11,0),
   CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
   CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
   CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID);
```

Referential integrity constraints are implemented with foreign key to primary key references.

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

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

### Summary of the Relational Database Model

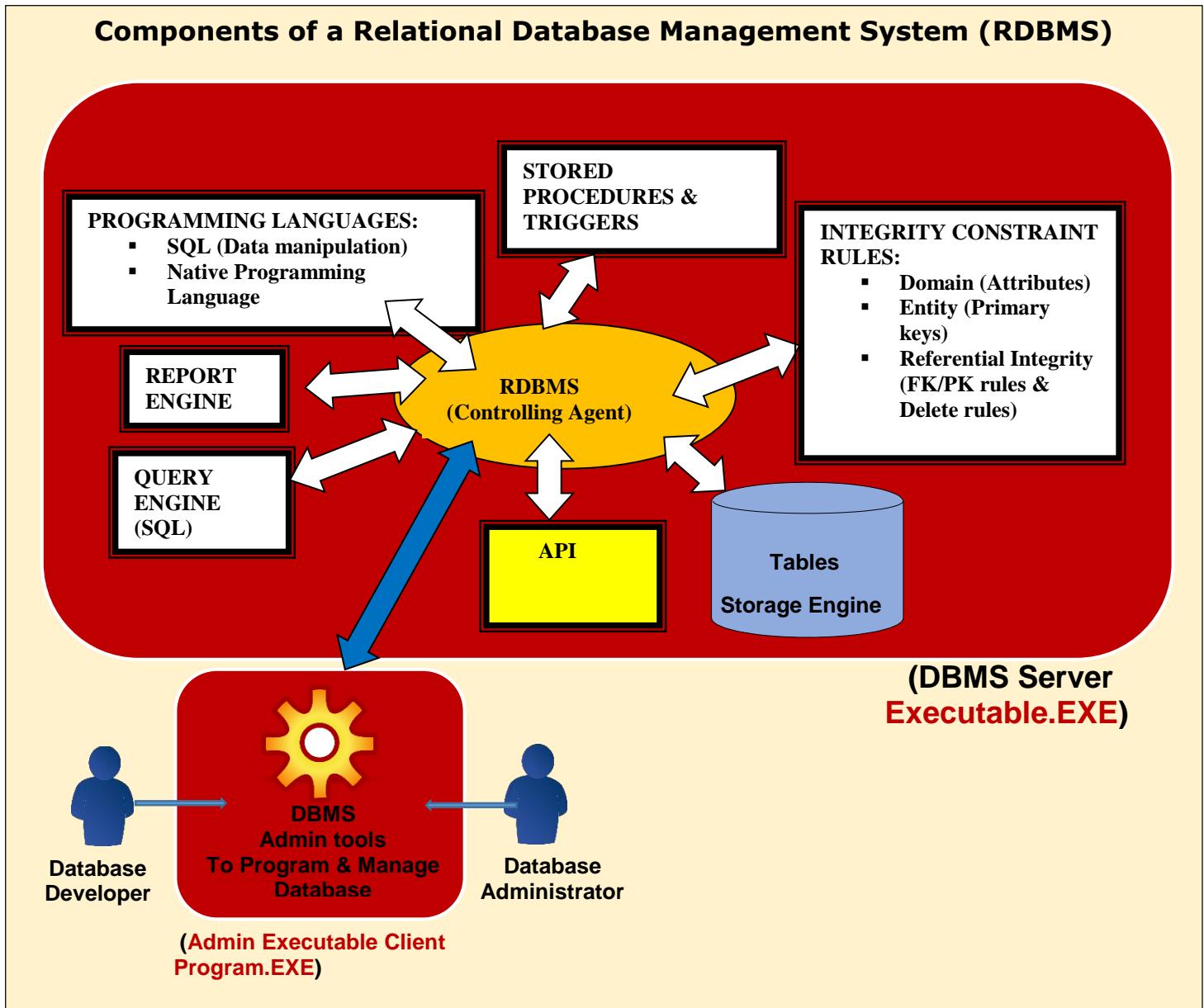
#### Relational Database Management System (DBMS) Components

□ RDBMS Components details:

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

Item	Component	Description
1.	<b>Storage System (Tables)</b>	<ul style="list-style-type: none"><li>▪ Storage System where data is kept</li><li>▪ Relations DBMS use tables to store data</li></ul>
2.	<b>Queries Engine</b>	<ul style="list-style-type: none"><li>▪ Engine to enable the querying (searching, updating, inserting &amp; deleting) the data using SQL Language.</li></ul>
3.	<b>Reporting Services</b>	<ul style="list-style-type: none"><li>▪ Engine to enable the creation/generation of reports or formatted result of database queries and contains useful data for decision-making and analysis.</li></ul>
4.	<b>programming language:</b> <ul style="list-style-type: none"><li>▪ <b>Data Manipulation Language: SQL</b></li><li>▪ <b>Custom(native-Language)</b></li></ul>	<ul style="list-style-type: none"><li>▪ 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)</li><li>▪ 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)</li></ul>
5.	<b>Programming Tool:</b> <ul style="list-style-type: none"><li>▪ <b>Stored Procedures</b></li></ul>	<ul style="list-style-type: none"><li>▪ Function-like executable BLOCKS of one or more SQL statement &amp; 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</li></ul>
6.	<b>Programming Tools:</b> <ul style="list-style-type: none"><li>▪ <b>Triggers</b></li></ul>	<ul style="list-style-type: none"><li>▪ A special type of <b>Stored Procedure</b> that <b>EXECUTES AUTOMATICALLY</b> under some specified condition (Similar to Event-Handlers in programming languages)</li></ul>
7.	<b>Administrative Tools/Applications:</b> <ul style="list-style-type: none"><li>▪ <b>Database Development User-Interface/Programs</b></li><li>▪ <b>Database Administration Interface/program</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>For Developers</b> – User-Interface Programs that allows the <i>Database Developer</i> to program, create databases &amp; execute queries against the database.</li><li>▪ <b>IMPORTANT!</b> These are programs or <b>client executables</b> that allows user/developer to interface and program the DBMS</li><li>▪ <b>For Database Administrators</b> – User-Interface Program that allows the <i>Database Administrator</i> manage the DBMS network administration, security, backup &amp; Recovery, etc.</li><li>▪ <b>IMPORTANT!</b> These are programs or <b>client executables</b> that allows user/system administrator to interface and manage the DBMS</li></ul>
8.	<b>Administrative Tools:</b> <ul style="list-style-type: none"><li>▪ <b>Other</b></li></ul>	<ul style="list-style-type: none"><li>▪ Other tools</li></ul>
9.	<b>Application Programming Interface (API)</b>	<ul style="list-style-type: none"><li>▪ A set of routines, protocols, and tools for integration by external programs into the existing Database Management Software application.</li><li>▪ Opens the door to all kinds of integration into the DBMS by other applications etc.</li></ul>
10.	<b>Data Integrity Mechanism</b> <ul style="list-style-type: none"><li>▪ <b>Domain Constraints (Attributes Rules)</b></li><li>▪ <b>Entity Constraints (Primary Key rules)</b></li><li>▪ <b>Referential Integrity (Foreign Key/Primary Key rules &amp; Delete rules)</b></li></ul>	<ul style="list-style-type: none"><li>▪ Constraints are RULES limiting <u>values assigned to attributes</u> &amp; limiting <u>actions in the database</u>.</li><li>▪ The relational model includes several types of constraints or rules whose purpose is to facilitate maintaining accuracy &amp; data integrity (correct, consistent &amp; accessible):<ul style="list-style-type: none"><li>○ <b>Domain Constraints</b> – Rules on <u>value</u> assigned to attributes</li><li>○ <b>Entity Constraints</b> – Rules on entities <u>primary keys</u> (NOT NULL)</li><li>○ <b>Referential Integrity Constraints</b> – Business rules implemented in the database<ul style="list-style-type: none"><li>○ Rules between related tables:<ul style="list-style-type: none"><li>- Foreign Keys must have a matching Primary key on another table</li><li>- Foreign Key can be NULL</li><li>- Delete rules (Restrict, Cascade, Set-to-Null, etc)</li></ul></li></ul></li></ul></li></ul>

- 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:

## TRANSFORMING EER DIAGRAMS INTO RELATIONS

### Mapping Regular Entities to Relations

- + Simple attributes: E-R attributes map directly onto the relation
- + Composite attributes: Use only their simple, component attributes
- + Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

Figure 4-8 Mapping a regular entity

(a) CUSTOMER entity type with simple attributes

CUSTOMER  
Customer ID  
Customer Name  
Customer Address  
Customer Postal Code

(b) CUSTOMER relation

CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode
------------	--------------	-----------------	--------------------

## Mapping Entity Composite Attributes

Figure 4-9 Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

CUSTOMER  
Customer ID  
Customer Name  
Customer Address  
(CustomerStreet, CustomerCity, CustomerState)  
Customer Postal Code

(b) CUSTOMER relation with address detail

CUSTOMER

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

## Mapping Entity with Multivalued Attributes

Figure 4-10 Mapping an entity with a multivalued attribute

(a)

EMPLOYEE
Employee ID
Employee Name
Employee Address
[Skill]

Multivalued attribute becomes a separate relation with foreign key

(b)

EMPLOYEE
EmployeeID   EmployeeName   EmployeeAddress

EMPLOYEE_SKILL
EmployeeID   Skill

One-to-many relationship between original entity and new relation

## 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

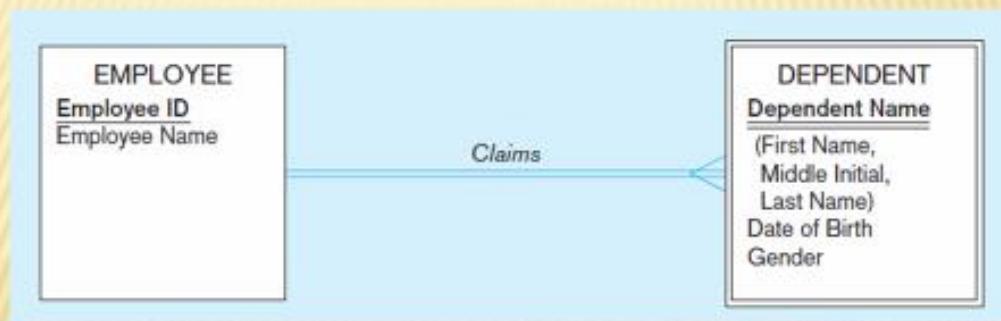
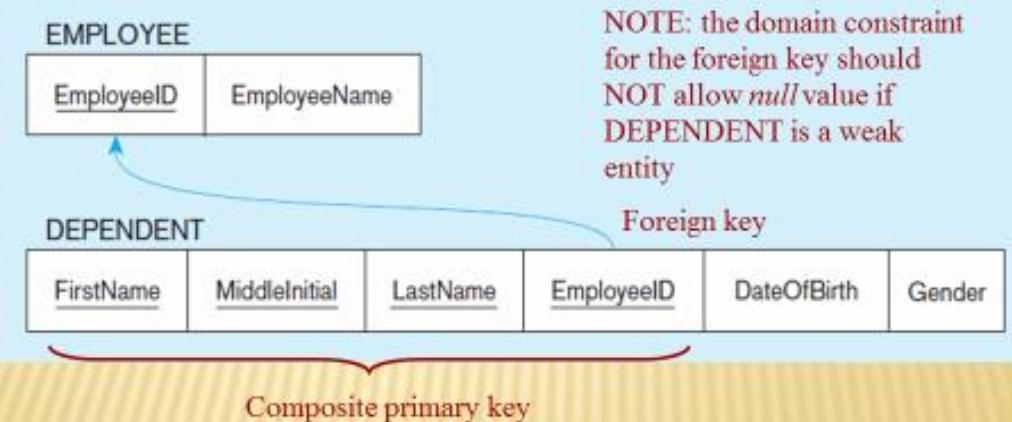


Figure 4-11 Example of mapping a weak entity (cont.)

b) Relations resulting from weak entity



### Mapping Relationships from Conceptual Model to Logical Model

Mapping BINARY Relationships:

## TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

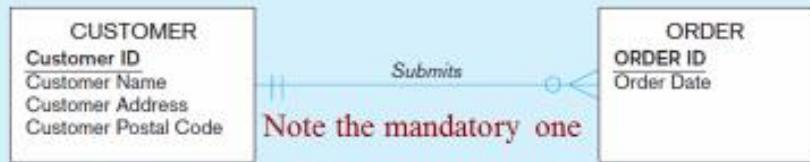
### Mapping Binary Relationships

- + One-to-Many – Primary key on the one side becomes a foreign key on the many side
- + Many-to-Many – Create a *new relation* with the primary keys of the two entities as its primary key
- + One-to-One – Primary key on mandatory side becomes a foreign key on optional side

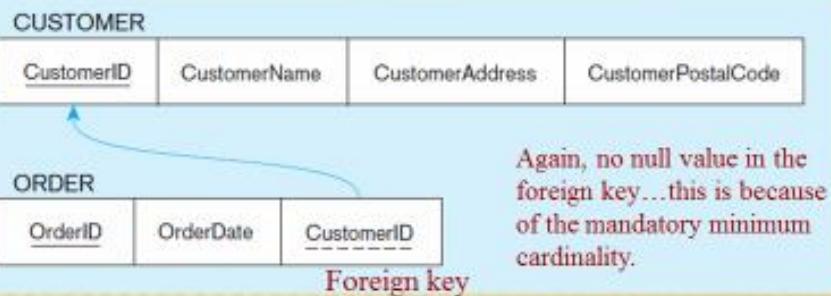
## Mapping 1-to-Many (1:M) Binary Relationships:

Figure 4-12 Example of mapping a 1:M relationship

### a) Relationship between customers and orders



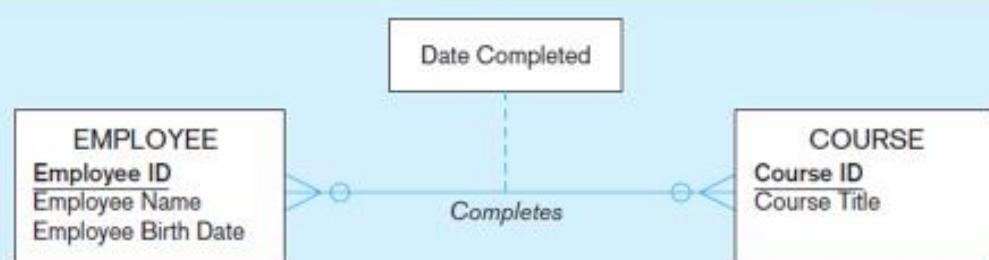
### b) Mapping the relationship



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

Figure 4-13 Example of mapping an M:N relationship

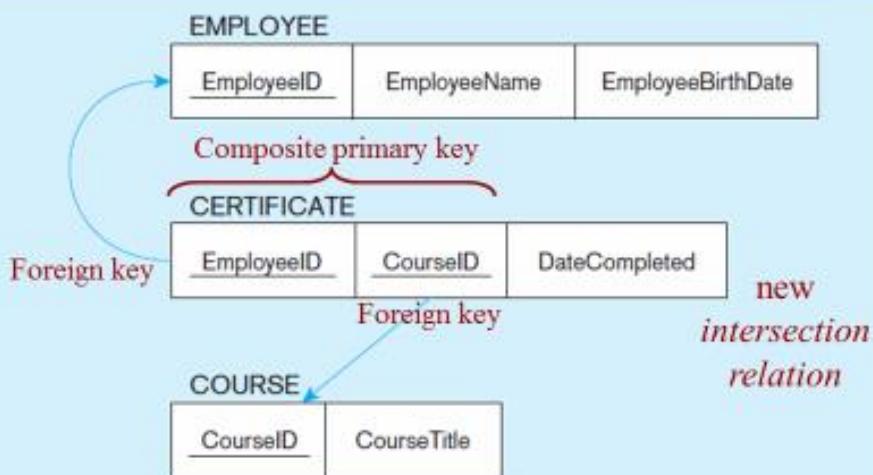
a) Completes relationship (M:N)



The *Completes* relationship will need to become a separate relation.

Figure 4-13 Example of mapping an M:N relationship (cont.)

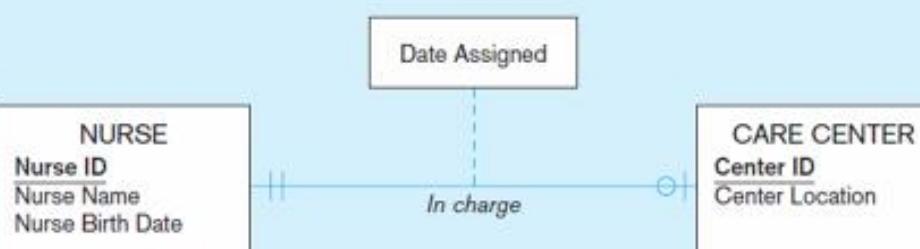
b) Three resulting relations



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

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

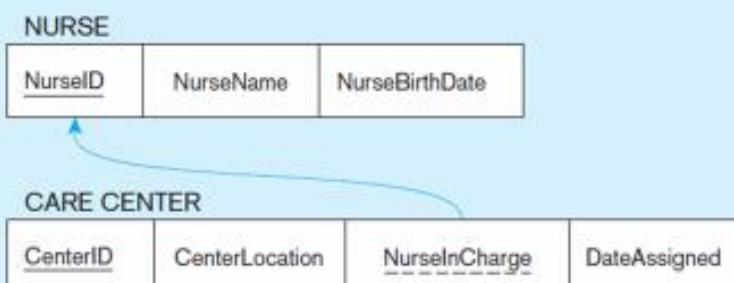
a) In charge relationship (1:1)



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

## TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

### Mapping Associative Entities

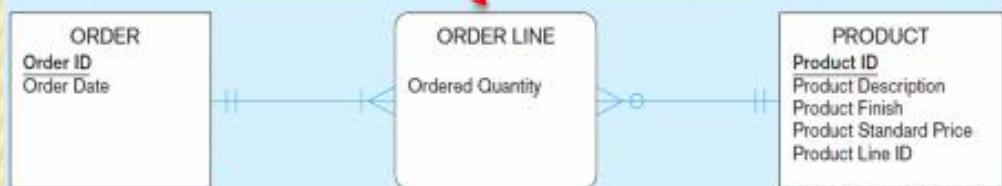
- Identifier Not Assigned ( Not provided in Associate Entity) do the following:
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)
- Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

## 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

Figure 4-15 Example of mapping an associative entity

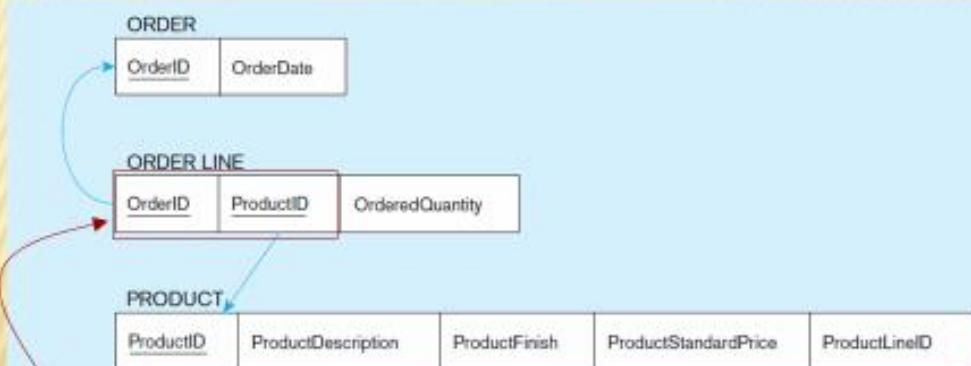
### a) An associative entity with NO identifiers provided



Note: Product Line ID is included here because it is a foreign key into the PRODUCT LINE entity, not because it would normally be included as an attribute of PRODUCT

Figure 4-15 Example of mapping an associative entity (cont.)

### b) Three resulting relations



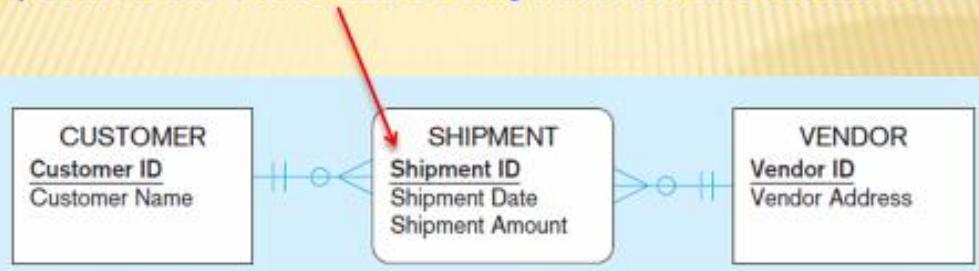
Composite primary key formed from the two foreign keys

## 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

Figure 4-16 Example of mapping an associative entity with an identifier

### a) SHIPMENT associative entity that includes identifier

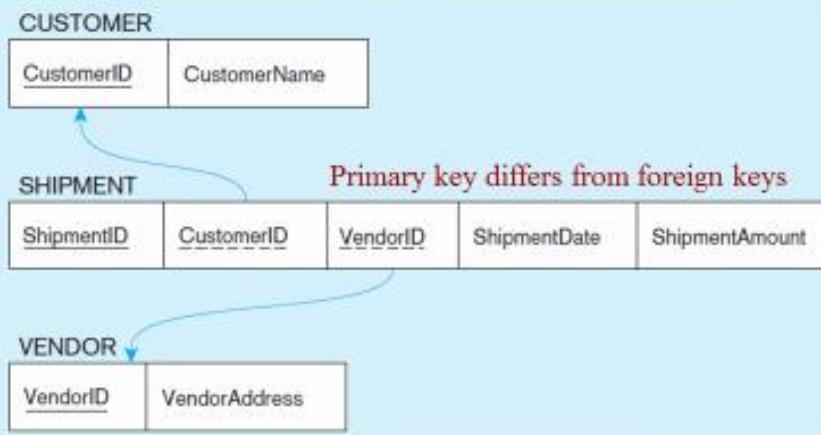


Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

41

Figure 4-16 Example of mapping an associative entity with an identifier (cont.)

### b) Three resulting relations



Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

30

**Mapping UNARY Relationships:**

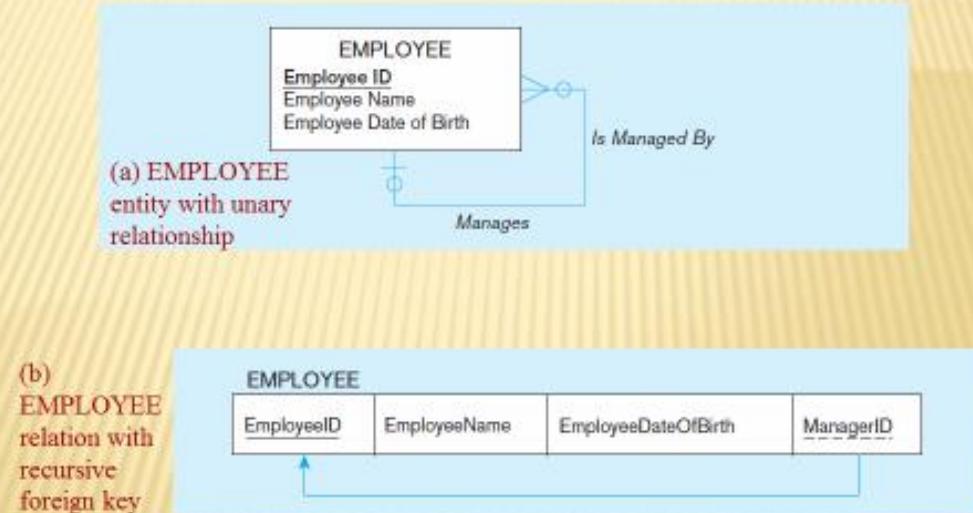
## TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

### Mapping Unary Relationships

- + One-to-Many – Recursive foreign key in the same relation
- + Many-to-Many – Two relations:
  - ✗ One for the entity type
  - ✗ One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

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

Figure 4-17 Mapping a unary 1:N relationship

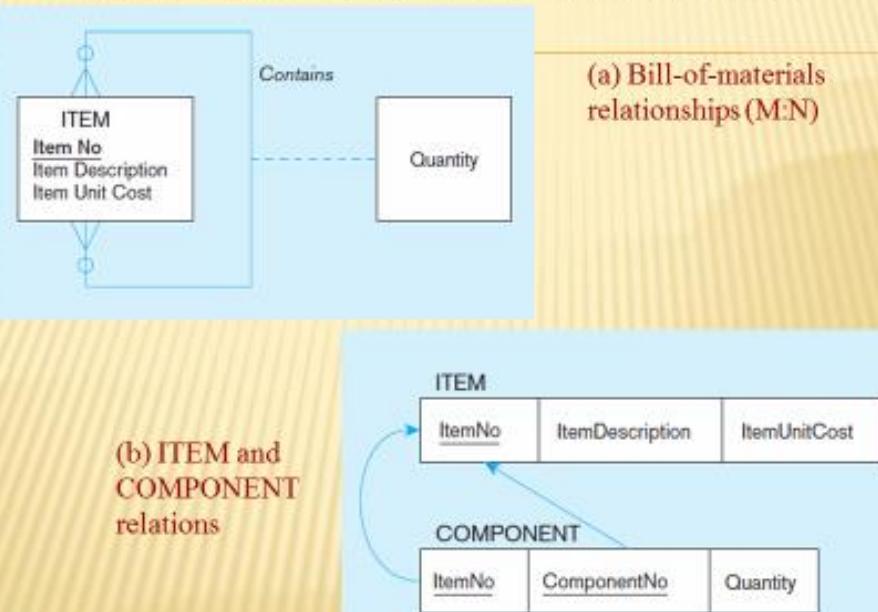


Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

32

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

Figure 4-18 Mapping a unary M:N relationship



Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

33

## TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

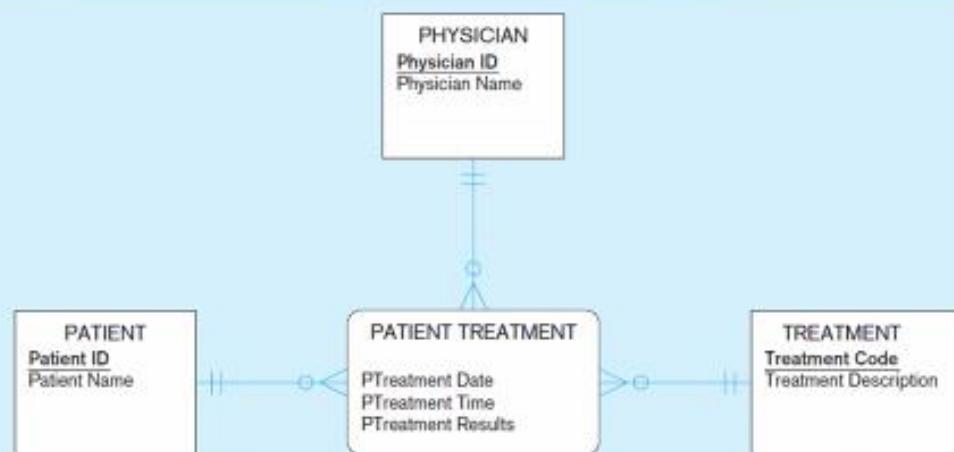
### Mapping Ternary (and n-ary) Relationships

- + One relation for each entity and one for the associative entity
- + Associative entity has foreign keys to each entity in the relationship

## Mapping TERNARY Entities Relationships with Associate Entity

Figure 4-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity

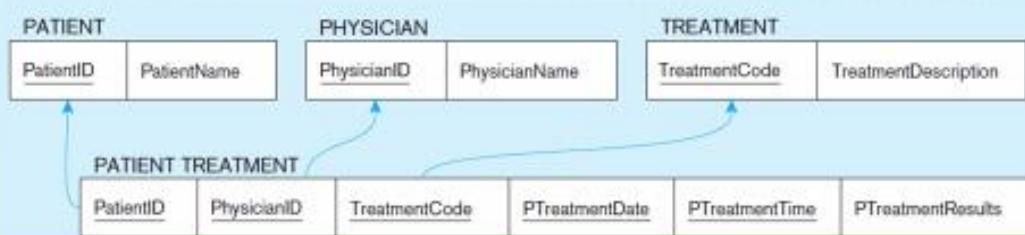


Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

35

Figure 4-19 Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT



Remember  
that the  
primary key  
MUST be  
unique.

This is why  
treatment date  
and time are  
included in the  
composite  
primary key.

But this makes a  
very  
cumbersome  
key...

It would be  
better to create a  
surrogate key  
like Treatment#.

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

36

## 2.1 Transforming EER Diagrams – Transforming Supertypes/Subtypes

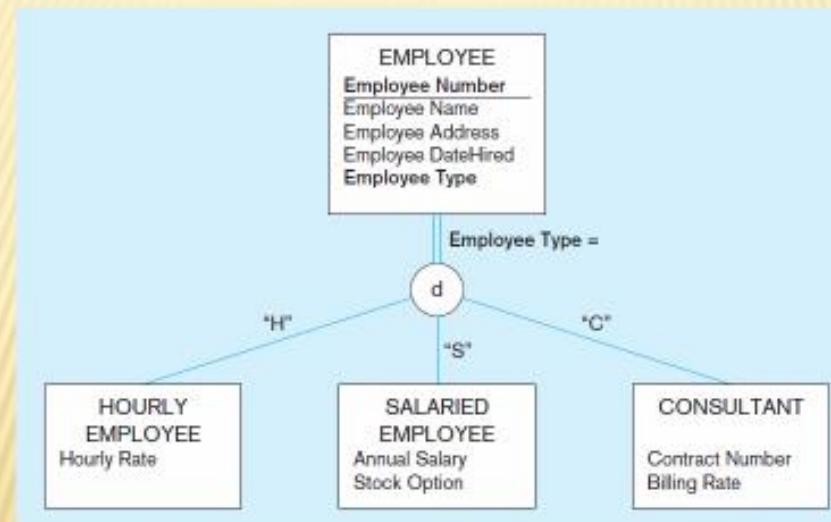
□ Transforming:

# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Supertype/Subtype Relationships

- + One relation for supertype and for each subtype
- + Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- + Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- + 1:1 relationship established between supertype and each subtype, with supertype as primary table

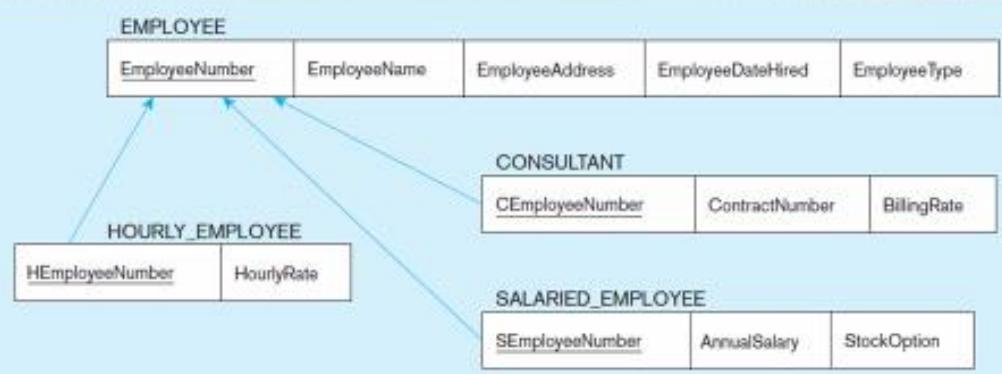
Figure 4-20 Supertype/subtype relationships



Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

38

Figure 4-21  
Mapping supertype/subtype relationships to relations



These are implemented as one-to-one  
relationships.

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

39

## 3.0 Database Design Phase – Normalization

### 3.1 Normalization Defined

□ Normalization:

#### REVIEW OF DATABASE DESIGN GOALS

- ✖ Eliminate/Reduce Redundancy
- ✖ Eliminate/Reduce Inconsistency
- ✖ Data Integrity (Correct, Accessible, etc.)
- ✖ Reduce complexity
- ✖ Atomic Tables (Simple tables)

## 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**

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

57

## EXAMPLE-FIGURE 4-2B

EMPLOYEE2

EmplID	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: EmplID, CourseTitle

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

42

## 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 RULES

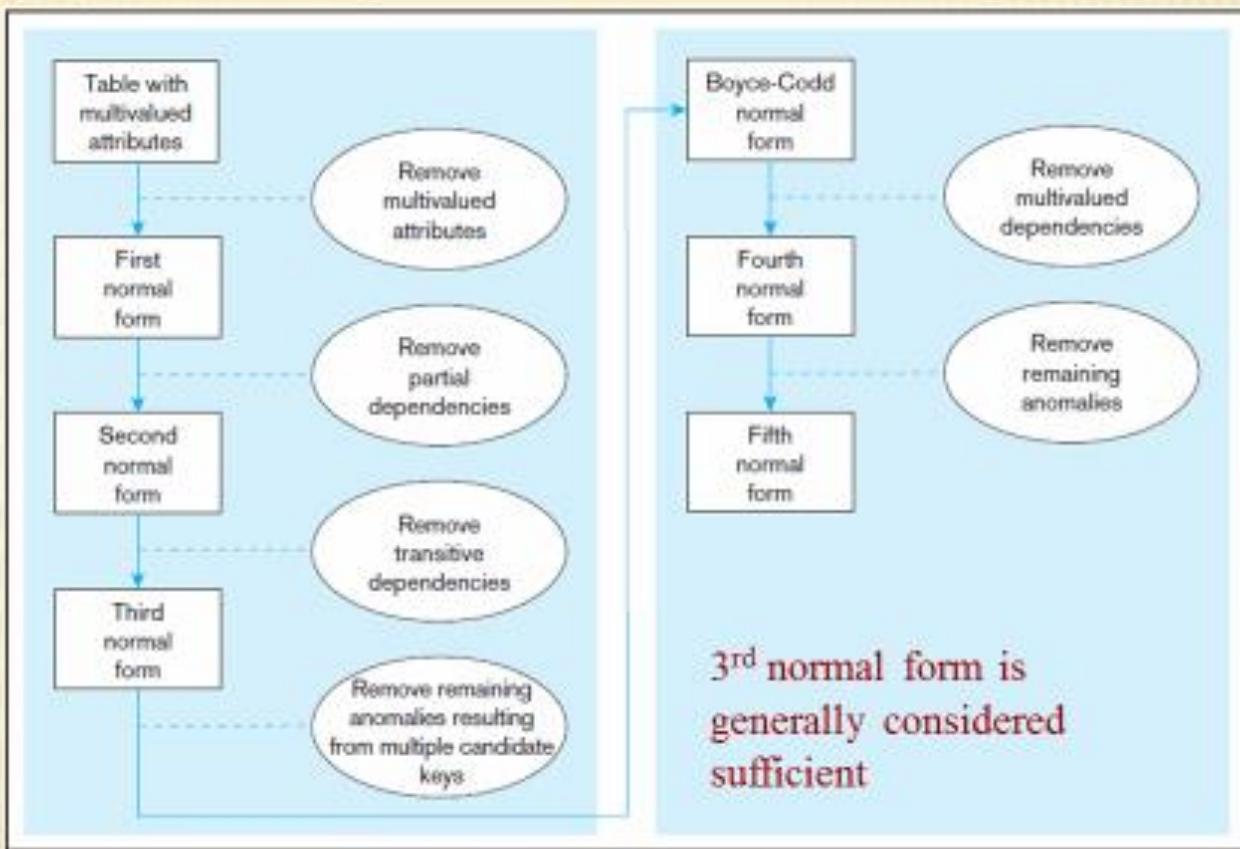
- ✖ **First Normal Form** – Remove multivalued attributes (also called repeating groups?) so there is a single value (possibly null) at the intersection of each row and column of the table.
- ✖ **Second Normal Form** – Any partial functional dependencies have been removed (i.e., nonkey attributes are identified by the whole primary key).
- ✖ **Third Normal Form** – Any transitive dependencies have been removed (ie., nonkey attributes are identified by only the primary key).

### NORMALIZATION RULES

- ✖ **Boyce-Codd Normal Form** – Any remaining anomalies that result from functional dependencies have been removed (because there was more than one possible primary key for the same nonkeys).
- ✖ **Fourth Normal Form** – Any multivalued dependencies have been removed
- ✖ **Fifth Normal Form** – Any remaining anomalies have been removed

## Normalization High-level Steps Flow Diagram

Figure 4.22 Steps in normalization



## FUNCTIONAL DEPENDENCIES AND KEYS

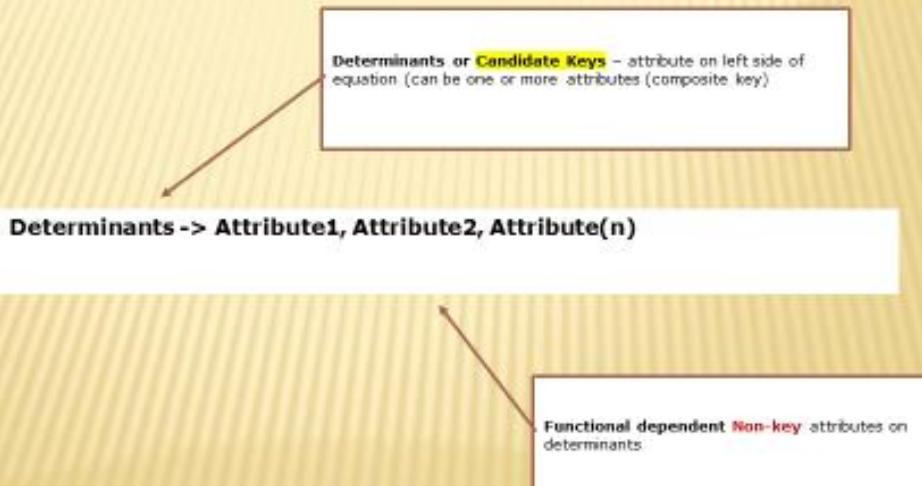
- ✖ Normalization is based on the analysis of functional dependency
- ✖ **Functional Dependency:** The value of one attribute (the *determinant*) determines the value of another attribute
- ✖ **Candidate Key:** An Attribute or combination of attributes that uniquely identifies a row in relationship.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

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

51

## FUNCTIONAL DEPENDENCIES AND KEYS

### ✖ Functional Dependency Equation:



Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

51

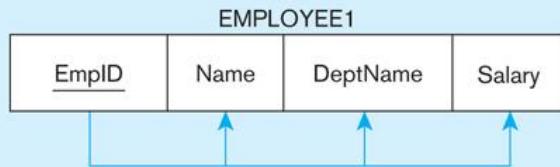
## FUNCTIONAL DEPENDENCIES AND KEYS

### ❖ Examples of Functional Dependency:

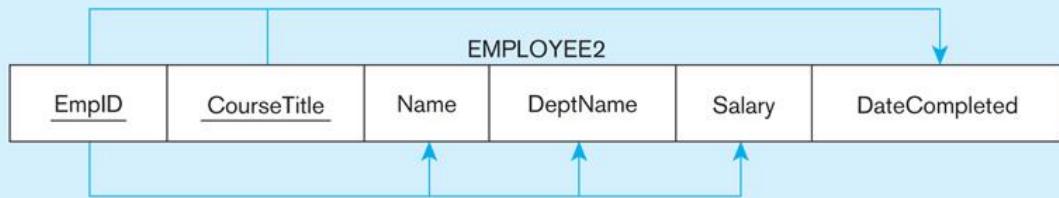
- *EmpID, CourseTitle -> DateCompleted*
  - DateCompleted is functionally dependent on both *EmpID & CourseTitle* since DateCompleted is determined by the identity of employee & title of course.
- *SSN -> Name, Address, Birthdate*
  - Name, address & birthdate are functionally dependent on SSN. There 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:

### (a) Functional dependencies in EMPLOYEE1



### (b) Functional dependencies in EMPLOYEE2



Copyright ©2013 Pearson Education, publishing as Prentice Hall

- **EMPLOYEE1(EmplID, Name, DeptName, Salary).** EmplID is the only determinant in this relation. All of the other attributes are functionally dependent on EmplID. Therefore, EmplID 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 EmplID 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 EmplID 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

### 3.3 Normalization – Example

## 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				
<hr/>					
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
<hr/>				Total	\$2,900.00
<hr/>				<hr/>	

Copyright ©2013 Pearson Education, publishing as Prentice Hall

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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

## STEP 1 – Convert First Normal Form

### Step 1 – Convert to First Normal Form

## FIRST NORMAL FORM

- ✖ Objectives
  - ✖ Remove Multivalued Attributes or repeating groups & identify a Primary Key
- ✖ First Normal Form Steps:
  1. Remove multivalued attributes or repeating groups so every attribute value is atomic
  2. Select a Primary Key
    - Identify determinants & select keys from determinants
    - Select PK based on data, business requirements etc.
- ✖ Fig. 4-25 is **not** in 1<sup>st</sup> Normal Form:
  - ✖ multivalued attributes exists → so it **CANNOT** be a **table** or relation.

58

### 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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

#### 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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

### 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

Only this combination of attributes  
guaranty uniqueness

<u>OrderID</u>	Order Date	Customer ID	Customer Name	Customer Address	<u>ProductID</u>	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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

Results of 1<sup>st</sup> Normal Form – The new Invoice Table is created:

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

<u>OrderID</u>	Order Date	Customer ID	Customer Name	Customer Address	<u>ProductID</u>	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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

## Status of Table after First Normal Form is applied

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

## ANOMALIES IN THIS TABLE

- ✖ **Redundancy** – CustomerID, CustomerName & Customer Address repeated in 3 rows
- ✖ **Insertion** – if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- ✖ **Deletion** – if we delete the Dining Table from Order 1006, thus row deleted & we lose information concerning this item's finish and price
- ✖ **Update** – changing the price of product ID 4 requires update in multiple records

Why do these anomalies exist?

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

Chapter 4 © 2013 Pearson Education, Inc. Publishing as Prentice Hall

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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

Redundancy

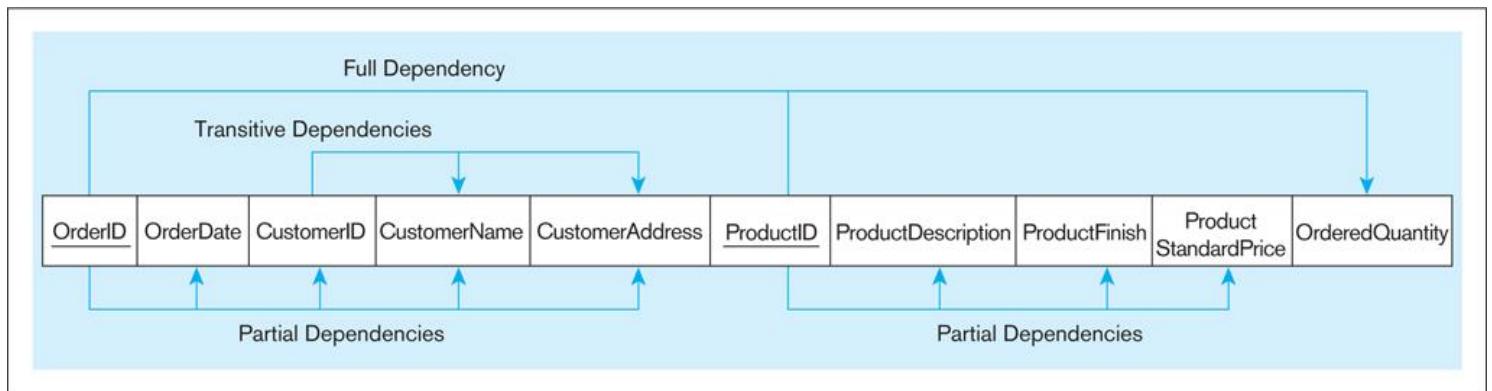
Delete this product, delete row, thus loose information on Dining Table

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:



Copyright ©2013 Pearson Education, publishing as Prentice Hall

**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** for the **FULL Composite Primary 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

## SECOND NORMAL FORM

### ★ Objectives

- Remove Partial Dependencies – Non-key columns depending only on part of the Primary Key and not the ENTIRE Primary Key must be removed & put in new table(s)
- Every non-key attribute must be fully functionally dependent on the **ENTIRE** primary key not part of key
- No Multiple Themes or entity types (tables) in one table

### ★ Second Normal Form Steps

- 1 Must be in 1NF
- 2 Create New Relation (Table) for attributes part of Partial Dependency & move existing Partial Key as Primary key to new table.
- 3 Move all Partial Dependent Attributes to new table
- 4 Rename original table as required to match its new functionality

## 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

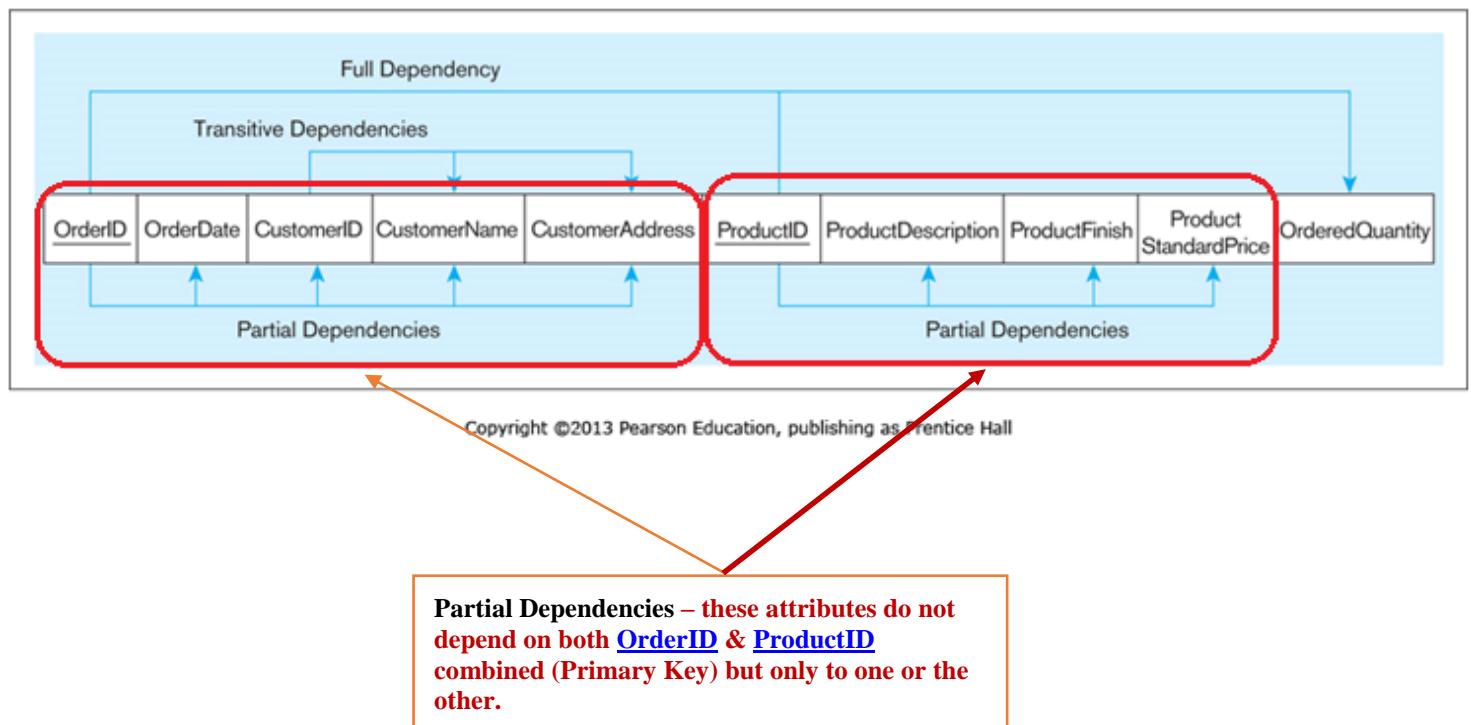
#### ★ Current status of INVOICE table:

- NOT in **2<sup>nd</sup> 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



## Step 2 – Steps to Convert to 2<sup>nd</sup> Normal Form

### Step 2(a) – Must Be in 1<sup>st</sup> Normal Form:

- INVOICE Table is in 1<sup>st</sup> Normal Form:

OrderID	OrderDate	CustomerID	CustomerName	CustomerAddress	ProductID	ProductDescription	ProductFinish	ProductStandardPrice	OrderedQuantity
1	2023-01-01	1	John Doe	123 Main St	1	Smartphone	Black	100.00	10

### 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

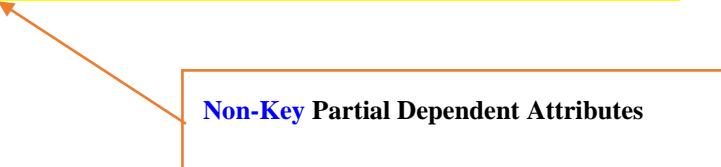
**Step 2(c) – MOVE the non-key Attributes that were part of Partial Dependency to the NEW Table:**

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

**OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress**



**ProductID → ProductDescription, ProductFinish, ProductStandardPrice**



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

**CUSTOMER ORDER**

<u>OrderID</u>	OrderDate	CustomerID	CustomerName	CustomerAddress

**PRODUCT**

<u>ProductID</u>	ProductDescription	ProductFinish	ProductStandardPrice

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

- Rename original INVOICE table to match its new functionality.

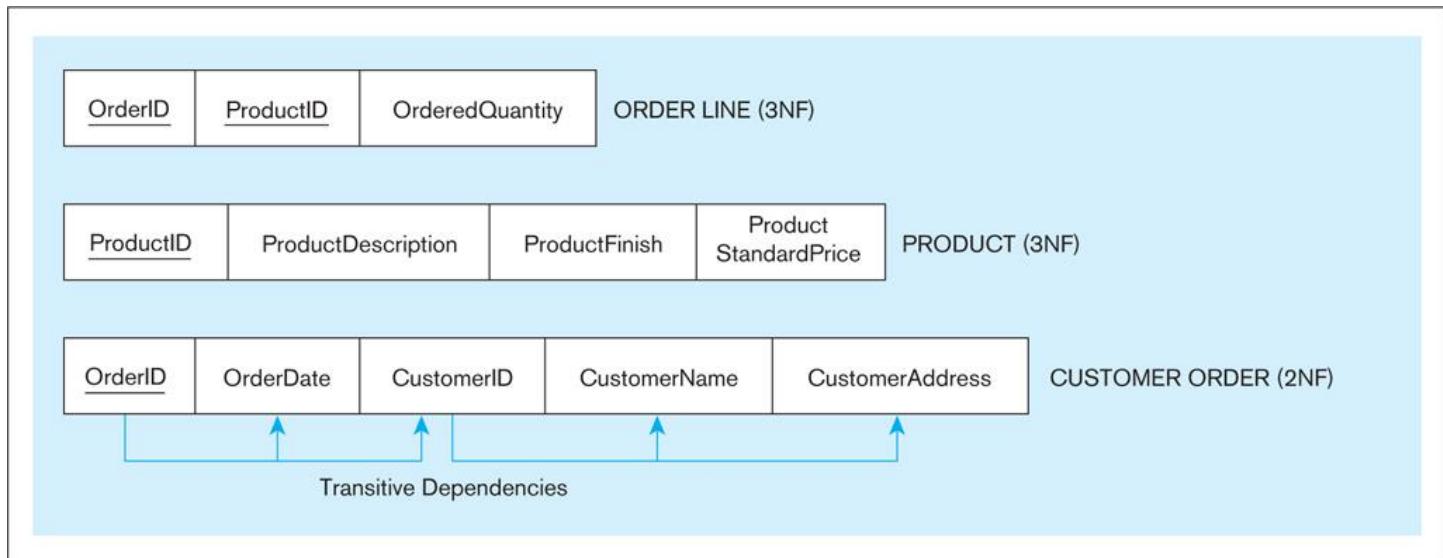
- Since what remains of the INVOICE table is just OrderID, ProductID & OrderedQuantity, we rename this table, **ORDER LINE** because each row of this table represents one line item of an order.

**ORDER LINE**

<u>OrderID</u>	<u>ProductID</u>	OrderedQuantity

## Status of Tables after Second Normal Form is applied

INVOICE Table Divided into 3 Tables After Applying 2<sup>nd</sup> Normal Form:



Copyright ©2013 Pearson Education, publishing as Prentice Hall

## ANOMALIES AFTER 2<sup>ND</sup> NORMAL FORM

### ✖ Remaining anomalies:

- There are non-key Attributes Depending on other Non-key Attributes
  - *CustomerName & CustomerAddress both depend on CustomerID*  
 $\text{OrderID} \rightarrow \text{CustomerID} \rightarrow \text{CustomerName} \rightarrow \text{CustomerAddress}$
- *CustomerID is NOT the Primary Key yet it has dependencies from other non-key attributes*
- *This leads to redundancy since customer name & address need to be reentered every time there is a new order.*

## STEP 3 – Convert to Third Normal Form

### Step 2 – 3rd Normal Form Objectives

## 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*

## Functional Dependency – Transitive Dependency Defined and Analyzed

### TRANSITIVE DEPENDENCY

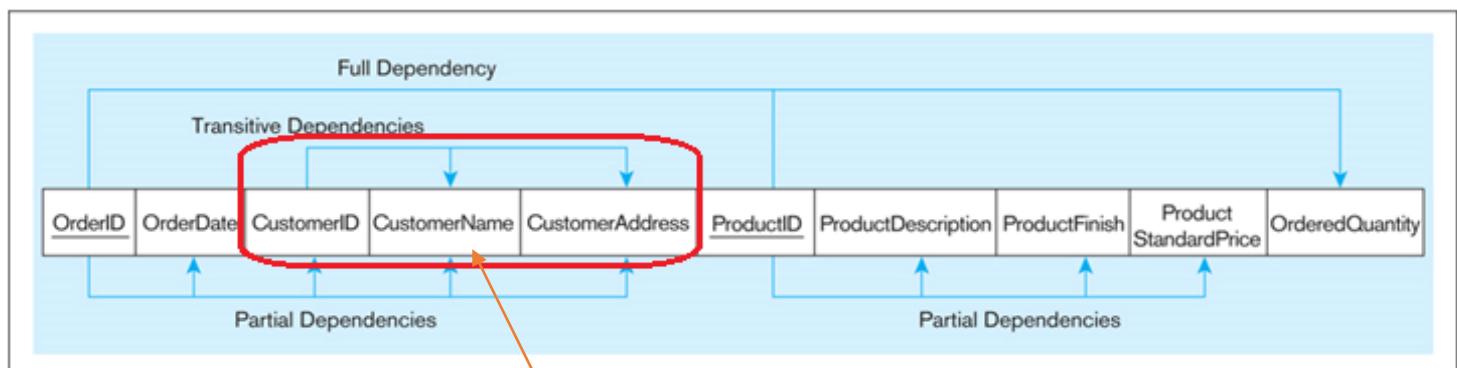
#### ★ Transitive Functional Dependency:

- ✖ 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 **3<sup>rd</sup> 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



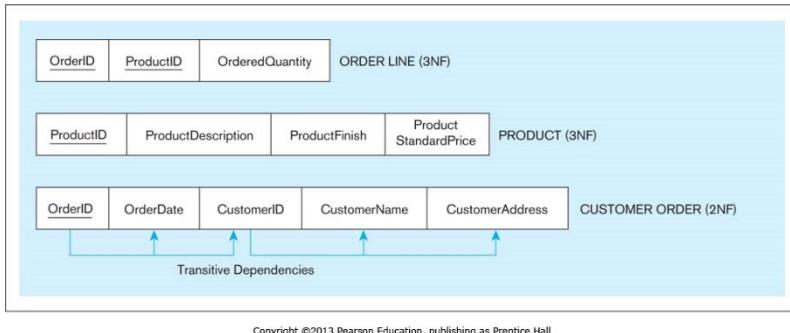
Copyright ©2013 Pearson Education, publishing as Prentice Hall

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 3<sup>rd</sup> Normal Form

### Step 3(a) – Must Be in 2<sup>nd</sup> Normal Form:

- All Tables are in 2<sup>nd</sup> Normal Form:



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

- 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
- Make it **Primary Key** of the NEW Relation (Table)
- Give the Table a Name
- Move all dependent Non-Key Attribute in original table to the new table
- 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**

CustomerID is Primary Key in NEW table

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

CUSTOMER

<u>CustomerID</u>	CustomerName	CustomerAddress
-------------------	--------------	-----------------

CustomerID is Foreign Key in original table

ORDER

<u>OrderID</u>	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 <th data-cs="2" data-kind="parent"></th> <th data-kind="ghost"></th>		
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

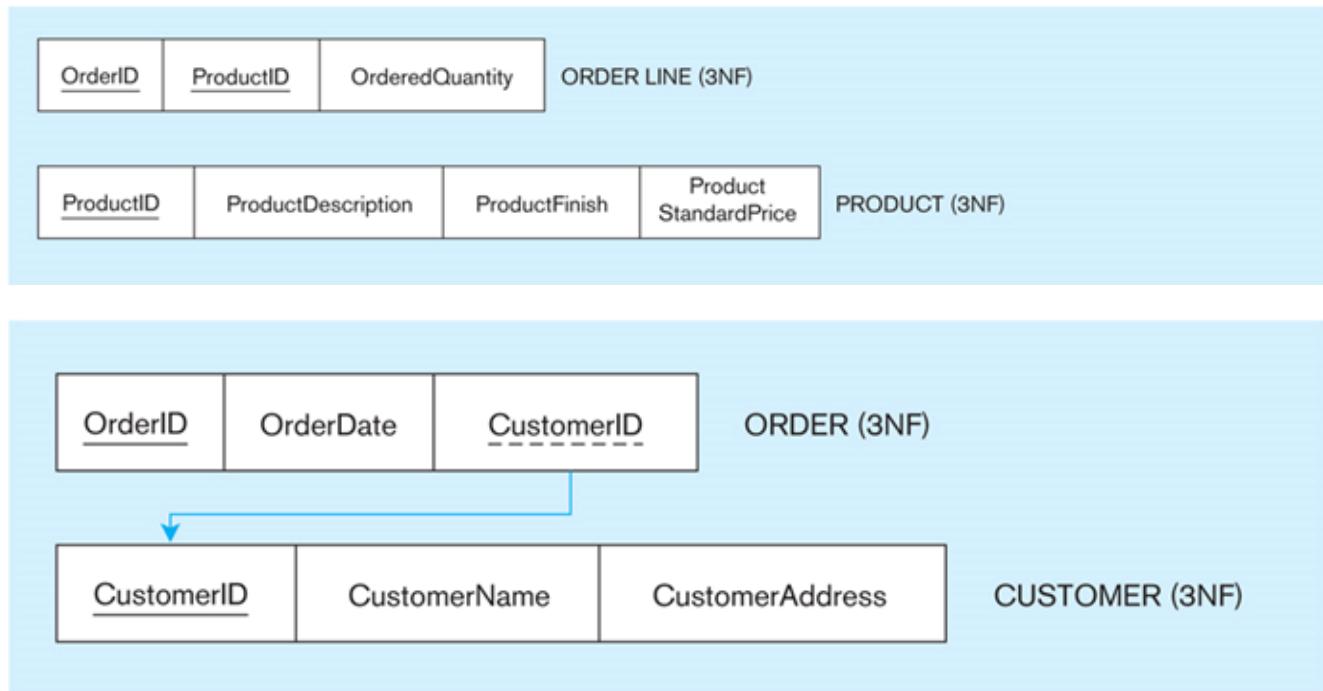
Copyright ©2013 Pearson Education, publishing as Prentice Hall

First Attempt to put it all inside a Table:

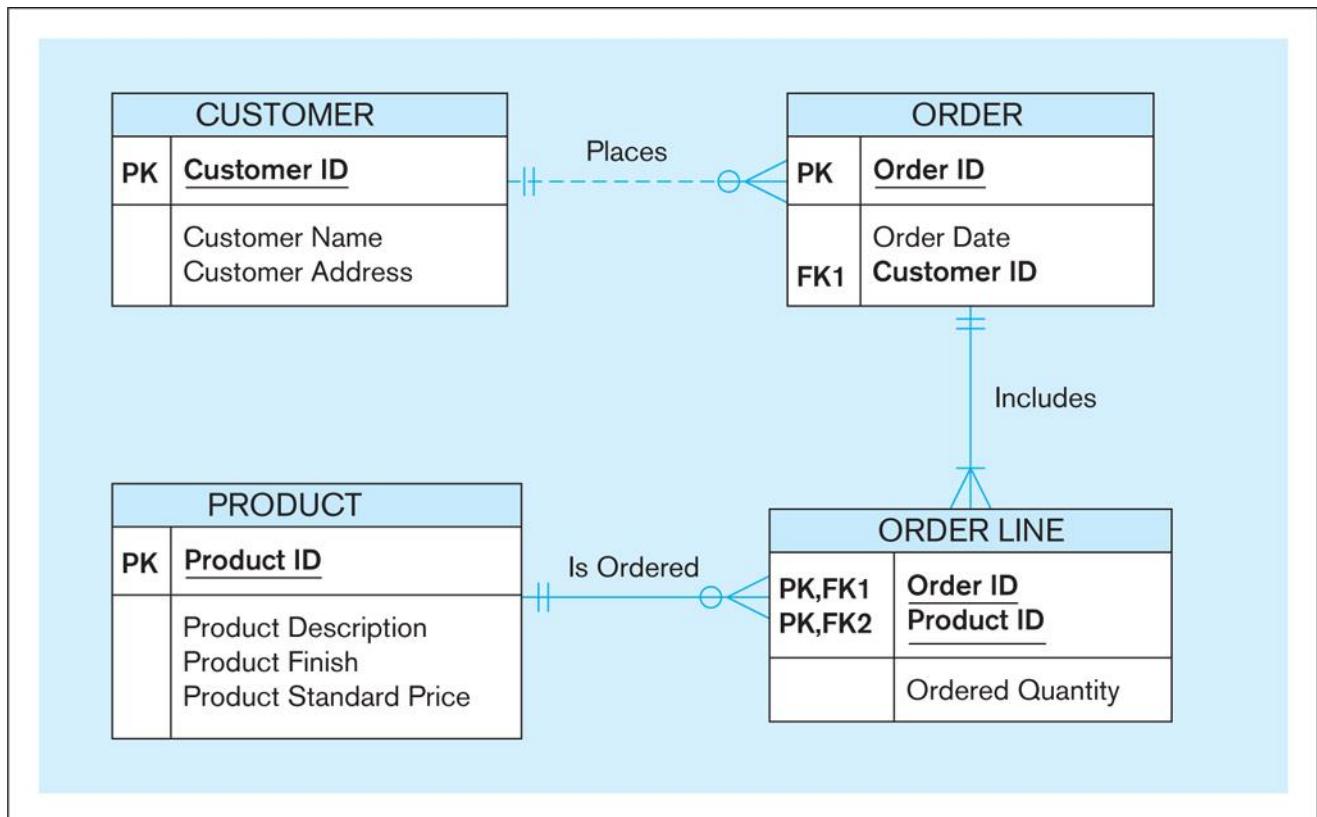
OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	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

Copyright ©2013 Pearson Education, publishing as Prentice Hall

**Invoice Table was broken down into 4 Table after 1<sup>st</sup>, 2<sup>nd</sup> & 3<sup>rd</sup> Normal Forms:**



**Relational Schema using MS Visio for the Invoice Data**



---

## **Boyce-Codd, Fourth, & Fifth Normal Forms**

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

### 3.2 Normalization – Summary

Summary:

## FIRST NORMAL FORM

### Objectives

- Remove Multivalued Attributes or repeating groups & identify a Primary Key

### First Normal Form Steps:

1. Remove multivalued attributes or repeating groups so every attribute value is atomic
2. Select a Primary Key
  - Identify determinants & select keys from determinants
  - Select PK based on data, business requirements etc.

78

## SECOND NORMAL FORM

### Objectives

- Remove Partial Dependencies – Non-key columns depending only on part of the Primary Key and not the ENTIRE Primary Key must be removed & put in new table(s)
- Every non-key attribute must be fully functionally dependent on the **ENTIRE** primary key not part of key
- No Multiple Themes or entity types (tables) in one table

### Second Normal Form Steps

1. Must be in 1NF
2. Create New Relation (Table) for attributes part of **Partial Dependency** & move existing Partial Key as Primary key to new table.
3. Move all Partial Dependent Attributes to new table
4. Rename original table as required to match its new functionality

79

## 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*

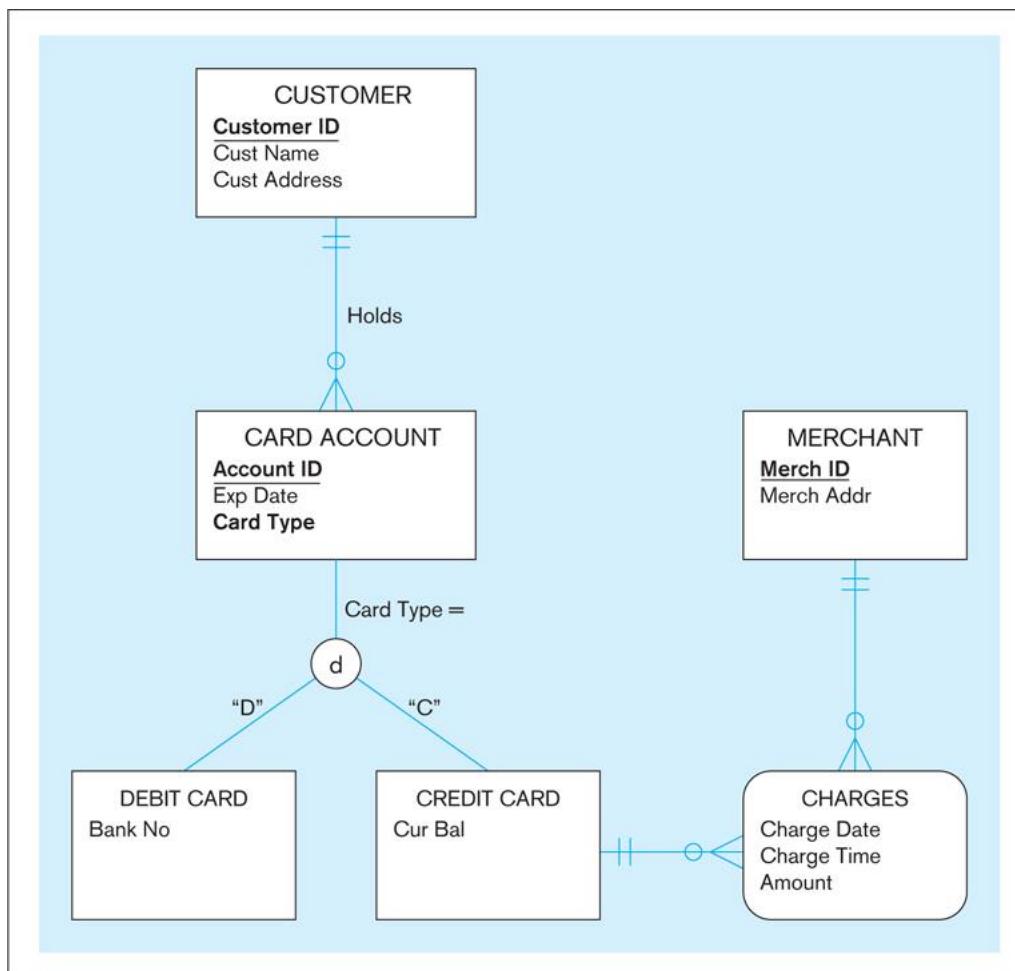
## 4.0 Database Design – Sample Problems

### 4.1 Book Problems

□ Problems:

#### Problem #6

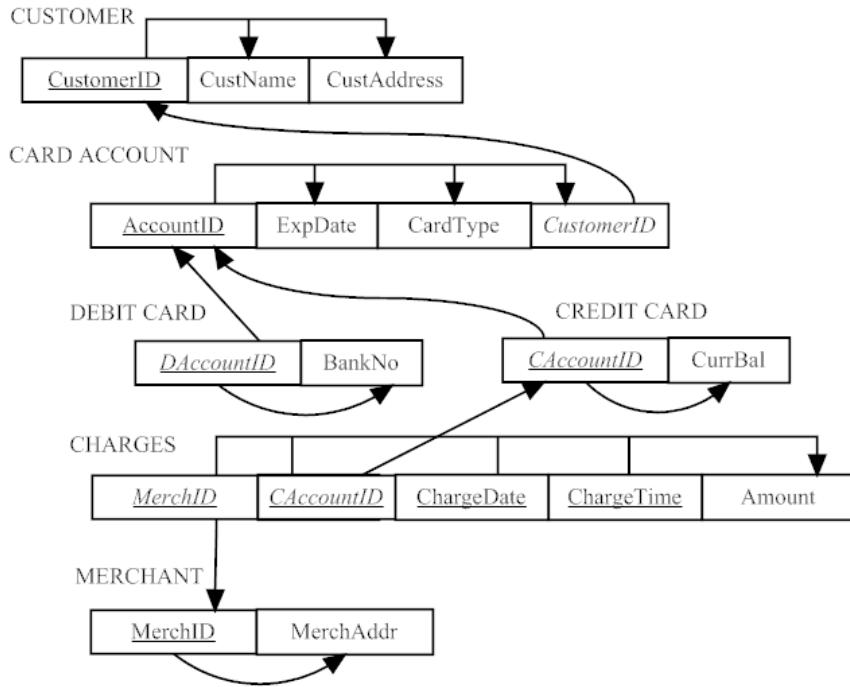
6. Figure 4-33 (page 194) shows an EER diagram for a simplified credit card environment. There are two types of card accounts: debit cards and credit cards. Credit card accounts accumulate charges with merchants. Each charge is identified by the date and time of the charge as well as the primary keys of merchant and credit card.
- Develop a relational schema.
  - Show the functional dependencies.
  - 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:
- Draw a relational schema and diagram the functional dependencies in the relation.
  - In what normal form is this relation?
  - Decompose GRADE REPORT into a set of 3NF relations.
  - Draw a relational schema for your 3NF relations and show the referential integrity constraints.
  - 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 follows:

**TABLE 4-4** Grade Report Relation

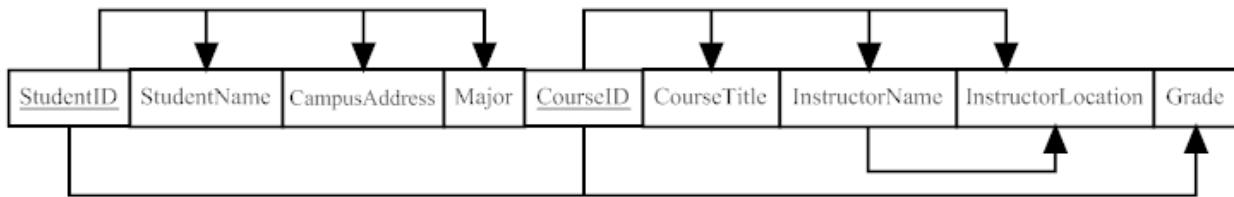
Grade Report

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	Mktg 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**

<u>StudentID</u>	StudentName	CampusAddress	Major
------------------	-------------	---------------	-------

**REGISTRATION**

<u>StudentID</u>	<u>CourseID</u>	Grade
------------------	-----------------	-------

**COURSE**

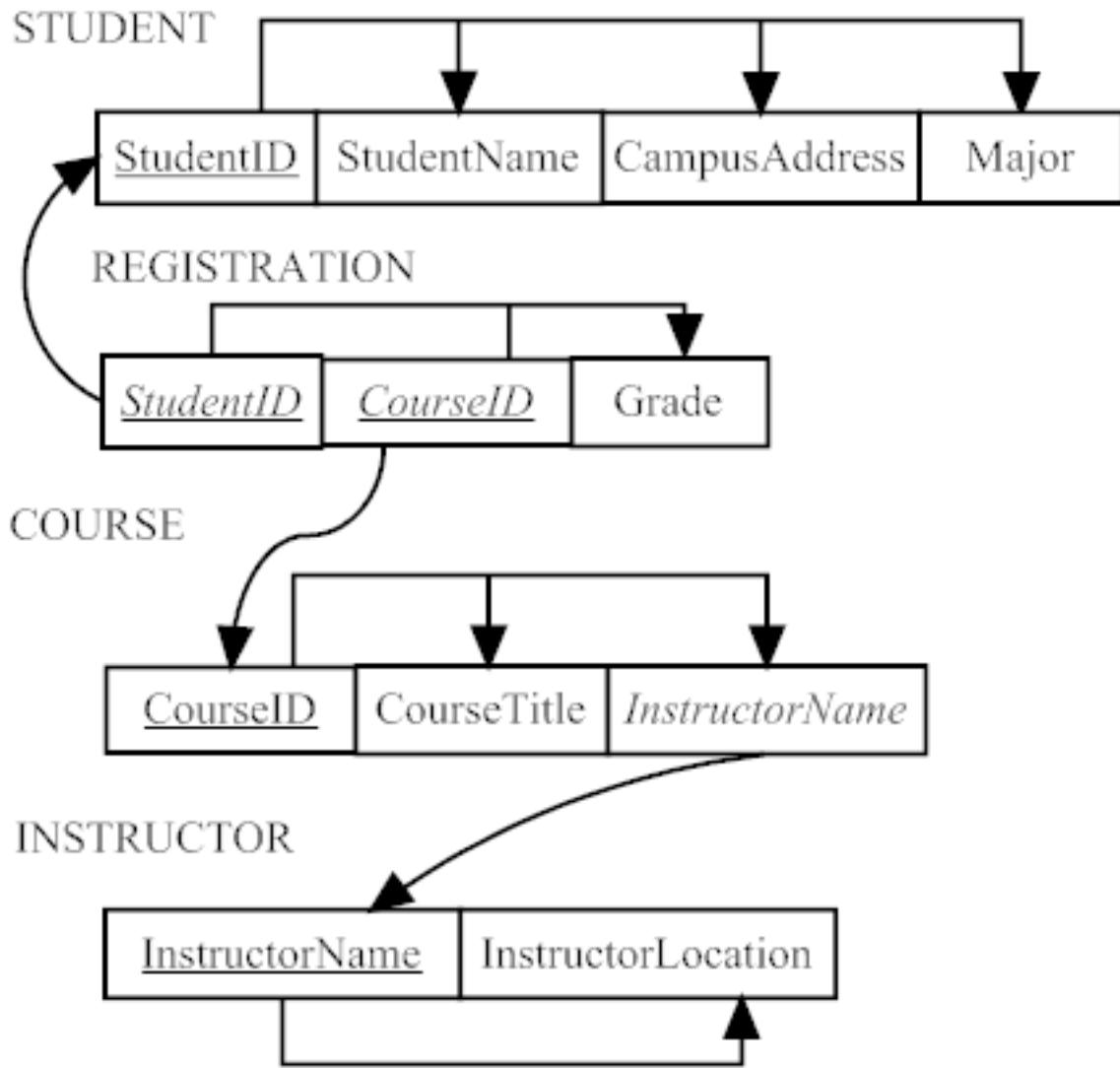
<u>CourseID</u>	CourseTitle	<i>InstructorName</i>
-----------------	-------------	-----------------------

**INSTRUCTOR**

<u>InstructorName</u>	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

