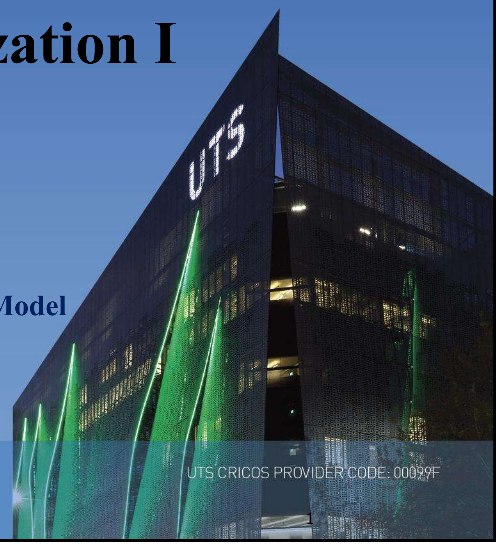
UTS:**ENGINEERING AND INFORMATION TECHNOLOGY**

UTS
UNIVERSITY OF TECHNOLOGY, SYDNEY

# lecture 5: Normalization I




**Modern Database Management**

**11th Edition, International Edition**

**Chapter 4: Logical Database Design and the Relational Model**

Jeffrey A. Hoffer, V. Ramesh, Heikki Topi

**Innovation in practice**
eng.uts.edu.au • it.uts.edu.au

UTS CRICOS PROVIDER CODE: 00099F

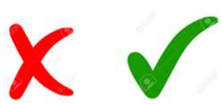


## Participations and Discussions

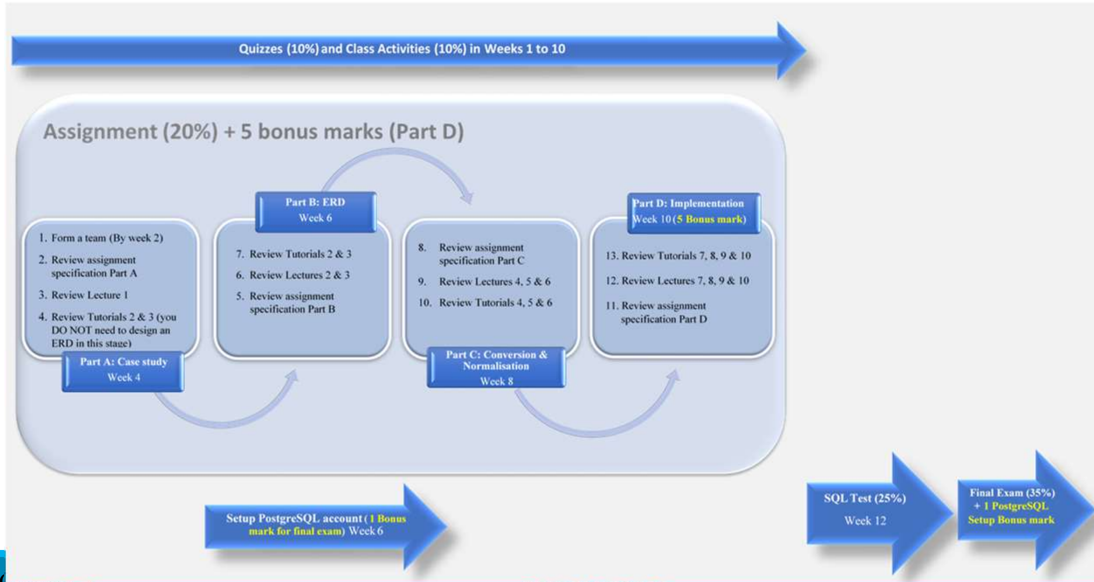
**If you have any question and you don't want to share it now, send it to us via UTSOnline/Discussion Board.**

**However, it is better to speak out ☺**

**Please follow the following signs in the lecture slide ☺**

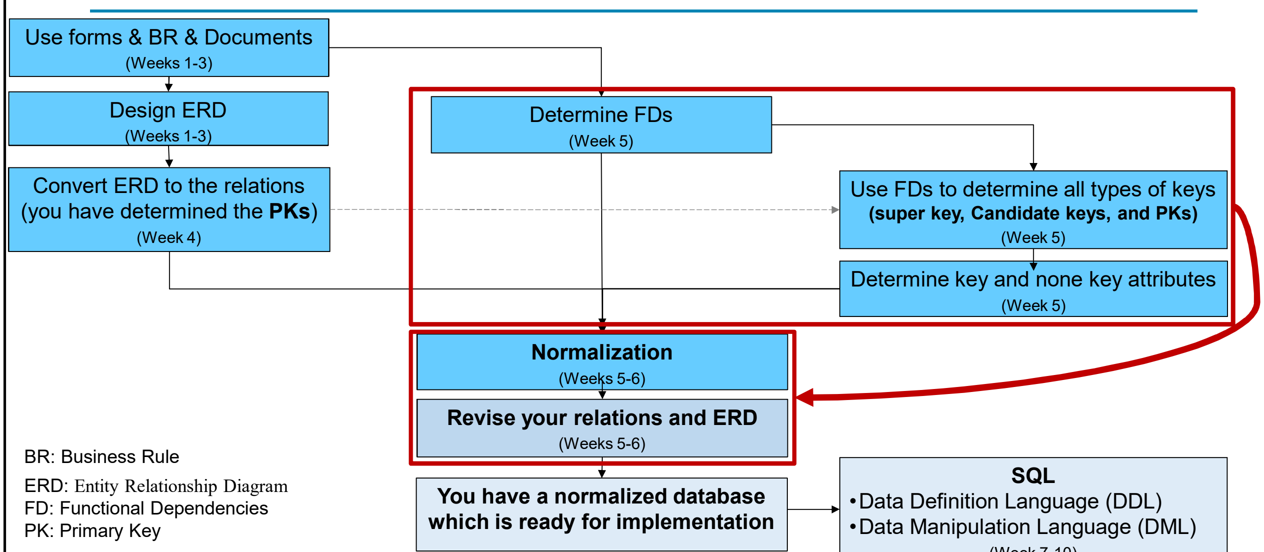


# Assessment Chart and Knowledge Guideline






# Subject Flowchart




BR: Business Rule
ERD: Entity Relationship Diagram
FD: Functional Dependencies
PK: Primary Key

# DF Learning Plan

**Description:** we will have collaborative lecture at the beginning of the class. You need to do some tasks during the lecture as part of your class activities. Then you will do a quiz of what you have learned, then the tutorial will start. you will work in groups during the class.

Please be aware that the lecture slides with Blue title are designed for your self study.

**Workshop Timetable:**

| Activity | Duration | Comments |
|---|---|---|
| **Lecture** | 1 hour and 40 minutes | You will have 3 tasks to complete that need to take 10 minutes in total |
| **Rest** | 10 minutes | Have fun |
| **Review** | 10 minutes | Please review the review questions and ask your questions if you have any |
| **Tutorial** | 50 Minutes | Have even more fun :D |
| **Quiz (Open Book)** | 5 minutes | On today's content. Will be run before or after the tutorial. Do your best ;) |
| **Leave the class** | 5 minutes | Don't forget to review what you have learn in this class, and check the information that is provided on **UTSOnline/Learning Material/Week 5** |

**NOTE:** next week we will have a short lecture and a long tutorial to practice normalization.



# Subject Overview

- **Design Entity Relationship Diagram (ERD)**
  - Week 1: Data Modelling I (Conceptual Level)
  - Week 2: Data Modelling II (Conceptual Level)
  - Week 3: Data Modelling III (Conceptual Level)
  - Week 4: Convert ERD to Relations (Logical Level)
  - **Week 5: Functional Dependencies**
  - **Week 5: Normalization I**
  - **Week 6: Normalization II**
- **Data manipulation**
  - **Week 7: Simple Query**
  - **Week 8: Multiple Table Queries**
  - **Week 9: Subquery**
  - **Week 10: Correlated Subquery**

## Lecture Five Objectives:

**Introduction: Why we need to do normalization and what are the anomalies?**

1. **Terms to Know to Do Normalization**
   - 1.1. Functional Dependencies
   - 1.2. Keys: Super-key, Candidate key and Primary Key
   - 1.3. Determining Candidate Keys from Functional Dependencies (FDs)
   - 1.4. Partial Functional Dependencies
   - 1.5. Transitive Functional Dependencies
   - 1.6 Why Partial and Transitive FDS cause the anomalies?
2. **Well-Structured Relations and Data Normalization**
3. **Steps in normalization**
4. **First Normal Form**
5. **Second Normal Form**
6. **Third Normal Form**



## Why we need to do normalization?

## By now:

- You have designed your ERD.
- You have converted your ERD to the relations.

➢ **Questions:**

1. Are your relations well-structured?
2. Will you have any redundant data in your relations?
3. Will you have any data inconsistency in your database?



## Example 1–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**

## **Is this relation (table) well-structured?** Have a look at the **anomalies** in this Table:

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 |

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

**Note:** The anomalies also happen after merging databases that are designed by the other database designers, or merging tables from different databases to create a new table.





## Example 2–Figure 4-26

| 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 |

FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

Question–Is this a relation? Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key? Answer–Composite: **OrderID, ProductID**

## **Is this relation (table) well-structured?** Have a look at the **anomalies** in this 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 |

FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

- **Insertion Anomaly**: if new product is ordered for order **1007** of existing customer, **customer data** must be re-entered, causing duplication
- **Deletion Anomaly** : if we delete the **Dining Table** from Order **1006**, we lose information concerning this item's finish and price
- **Update (Modification) Anomaly**: changing the **price** of **product ID 4** requires update in multiple records.

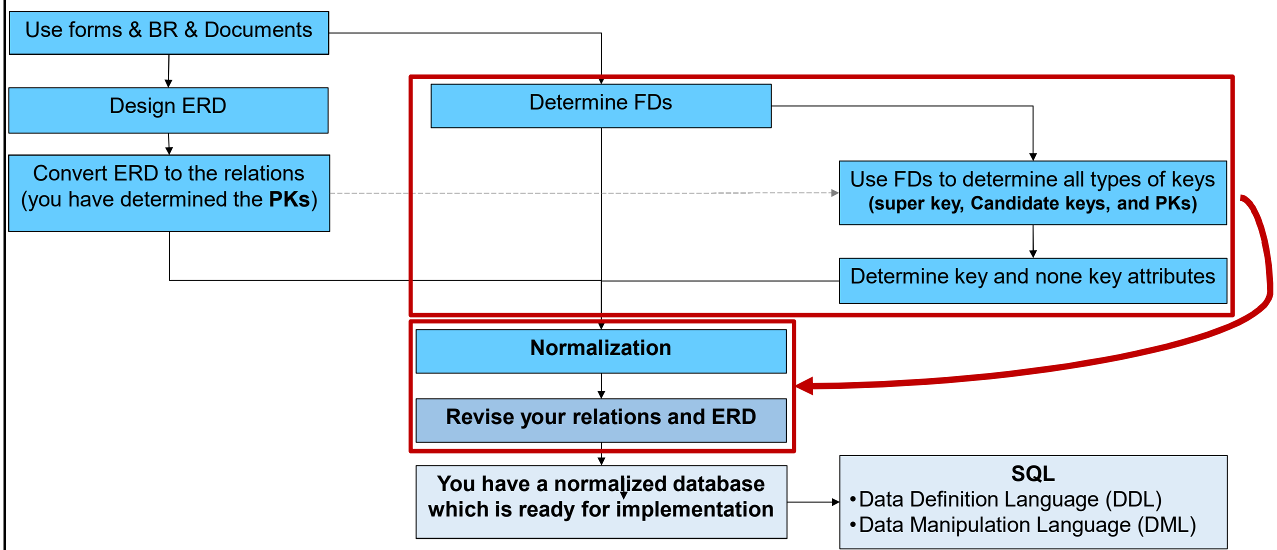
**Note:** The anomalies also happen after merging databases that are designed by the other database designers, or merging tables from different databases to create a new table.



## **Solution of these problems:**

**You need to normalize your relations to solve these problems**

## Subject Flowchart






# 1. Terms to Know for Normalization

## 1. Terms to Know for Normalization

1.1. Functional Dependencies

1.2. Keys: Super-key, Candidate key and Primary Key

1.3. Determining Candidate Keys from Functional Dependencies (FDs)

1.4. Partial Functional Dependencies

1.5. Transitive Functional Dependencies

1.6 Why Partial and Transitive FDS cause the anomalies?



## 1.1. Functional Dependencies

**Functional Dependency**: A constraint between two attributes in which the value of one attribute (**dependent**) is determined by the value of another attribute(s) (**determinant**).

- The value of one attribute or combinations of attributes (X) determines the value of another attribute(s) (Y)

  **X -> Y**

  where X determinant, Y dependent

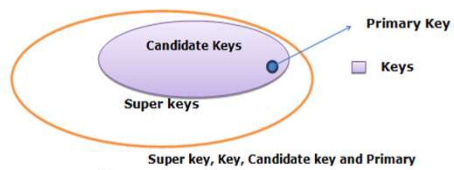
**Example:**

EmpID -> FName, Lname, DeptName, Salary

## 1.2. Keys of a Relation

- **Super-Key:**
  - Is a set of attributes within a table (relation) whose values can be used to uniquely identify a row in the relation.
- **Candidate Key:**
  - An attribute, or **minimal set** of attributes, that uniquely identifies a row in a relation (A unique identifier).
  - Each non-key field is functionally dependent on every candidate key.
  - One of the candidate keys will become the primary key

  E.g., perhaps there are both "credit card number" and "SS#" in a table. In this case both are candidate keys.
- **Primary Key:**
  - Is a unique identifier
  - It cannot contain null values
  - One of the candidate keys will become the primary key






## Example: Super key, Candidate Key and PK

One of the candidate keys will become the primary key

Candidate Keys

Super-Key

| StudentId | firstName | lastName | courseId |
|---|---|---|---|
| L0002345 | Jim | Black | C002 |
| L0001254 | James | McCloud | A004 |
| L0002349 | Peter | Black | C002 |
| L0001198 | Anne | McCloud | S042 |
| L0023487 | Peter | Murray | P301 |
| L0018453 | Anne | Norris | S042 |

**Note: Names are not a good choice to be a PK.**

## Class Activity 5.1 (3 minutes)

**Explain two properties that must be satisfied by candidate keys?**

a) **Unique identification:**

b) **Non-redundancy:**



## Class Activity 5.2 (2 minutes)

**Explain the difference between candidate key and primary key.**

## 1.3. Determining Candidate Keys Using Functional Dependencies (FDs)

- The candidate keys of a relation R can be defined using given FD set of the relation.
- To achieve this goal, the following concepts are discussed:
  - 1.3.1. Attribute Closure
  - 1.3.2. The Algorithm to Determine Candidate Keys Using FDs



## 1.3.1. Attribute Closure

- **Attribute Closure:**

  Attribute closure of an attribute set can be defined as a set of attributes which can be functionally determined from it.

  …

  Given FD set of a Relation R, If A is an attribute (or a combination of attributes), the set of attributes in relation R that are **functionally** dependent on A is called Attribute Closure of A and it can be represented as **$A^+$**.

- **Steps to Find the Attribute Closure of A**

  Given FD set of a Relation R:

  1- Add A to the attribute closure set of A (A+)

  2- Recursively add attributes which can be functionally determined from attributes of the set A+ until done.

## 1.3.1. Example: Find the Attribute Closure of A

**R (E-ID, E-Name, E-City, E-State)**

**FDs = { E-ID→ E-Name, E-ID → E-City, E-City → E-State }**

The attribute closure of E-ID can be calculated as:

1. Add E-ID to the set

**(E-ID)+ = {E-ID}**

2. Add Attributes which can be derived (functionally determined) from **any attribute of set**.
   - In this case, E-Name and E-City can be derived from E-ID.
   - In addition, E-State can be derived from E-City. So these are also a part of closure.

**(E-ID)+ = {E-ID, E-Name, E-City, E-State }**

Similarly:

**(E-Name)+ = {E-Name}**

**(E-City)+ = {E-City, E-State }**

Reference: http://www.geeksforgeeks.org/finding-attribute-closure-and-candidate-keys-using-functional-dependencies/



## 1.3.2. The Algorithm to Determine Candidate Keys Using FDs

**Step 1:** Collect all related FDs to the relation R

**Step 2:** Create a table with three columns

| Left | Middle | Right |
|---|---|---|
| | | |

**Step 3:** Write all attributes that only show up on the left side of some FDs under **Left** column

These attributes most be part of a key

**Step 4:** Write all attributes that only show up on the right side of some FDs under **Right** column

These attributes are not part of any key

**Step 5:** Write all attributes that show up on both left and right sides of some FDs under **Middle** column

These attributes may or may not be part of a key

**Step 6:** Determine the closure of attributes under **Left** and **Middle** columns to find which combination of those attributes **will functionally determine all other attributes**. Start from attributes under Left column.

**Step 6.1.** Add the attribute to the attribute closure set

**Step 6.2.** Add Attributes which can be derived from any attribute of the attribute closure set.

**Step 7:** The different combinations of attributes under Left and Middle columns that functionally determine **all other** attributes in relation R **are keys** for R i.e. **If A+ =R then A is a candidate key for R**

# 1.3.2. Example: Determining Candidate Keys Using FDs

**R (ABCD)**
**FDs={AB→C,**
**C→B,**
**C→ D}**

Steps 1, 2, 3, 4 and 5

| Left | Middle | Right |
|---|---|---|
| A | BC | D |

**Note: Super Key is the combination of attributes under the Left and Middle columns** (in this example ABC is the super key).

**Step 6.1.** **A+ ={A}**

**Step 6.2.** A+ is not equal to R. Therefore, we need to add another attribute under Middle column and find the attribute closure of the combination of A and B:

**AB+ ={AB}**
**AB+ ={AB}** and **AB→C** then **AB+ = {ABC}**
**AB+ ={ABC}** and **C→ D** then **AB+ = {ABCD} = R**

We need to try other combinations of attributes under Left and Middle columns to find all possible candidate key. Similarly:

**AC+ = {ACBD}**

**Step 7:** Determine the candidate keys:

**AB+ equals to R. So AB is a candidate key for R.**
**AC+ also equals to R. So AC is another candidate key for R.**



# More Examples for Determining keys Using FDs:

There are more examples presented in the related video that is uploaded on UTSonline in Week 5 folder.

## Class Activity 5.3: Determine the PK of the following relation (5 Minutes)

| |
|---|
| **INVOICE (OrderID, ProductID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)** |

- FDs:

| |
|---|
| **OrderID, ProductID → OrderQuantity** |
| **ProductID → ProductDescription, ProductFinish, ProductStandardPrice** |
| **OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress** |
| **CustomerID → CustomerName, CustomerAddress** |



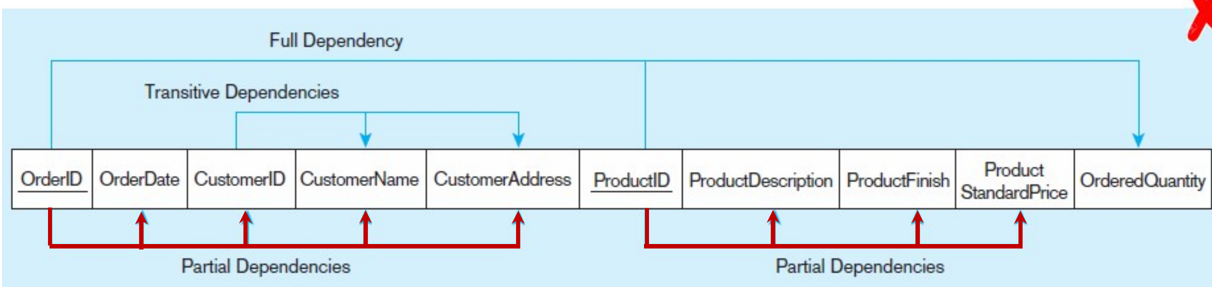
## Important Note:

**Partial** and **Transitive** Functional Dependencies in a relation cause the insertion, deletion and modification **anomalies.**

- **What is Partial Functional Dependency?**
- **What is Transitive Functional Dependency?**
- **Why they cause the anomalies?**

## 1.4. Partial Functional Dependencies (Book format)

A functional dependency in which one or more non-key attributes are functionally dependent **on part (but not all)** of the primary key (**Composite primary Key**).




**Composite primary Key of this relation is: OrderID, ProductID**



## 1.4. Partial Functional Dependencies (Tutorial format)

A functional dependency in which one or more non-key attributes are functionally dependent **on part (but not all)** of the primary key (**Composite primary Key**).

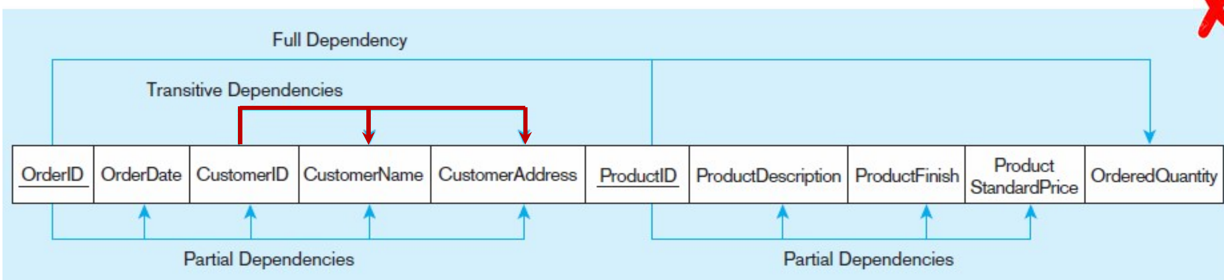
**INVOICE (OrderID, ProductID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)**

- **FDs:**

OrderID, ProductID → OrderQuantity
**ProductID → ProductDescription, ProductFinish, ProductStandardPrice**
**OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress**
CustomerID → CustomerName, CustomerAddress

## 1.5. Transitive Functional Dependencies (Book format)

A functional dependency between the primary key and one or more non-key attributes that are dependent on the primary key **via another non-key attribute**.




**Composite primary Key of this relation is: OrderID, ProductID**



## 1.5. Transitive Functional Dependencies (Tutorial format)

A functional dependency between the primary key and one or more non-key attributes that are dependent on the primary key **via another non-key attribute**.

**INVOICE (OrderID, ProductID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)**

- **FDs:**

OrderID, ProductID → OrderQuantity

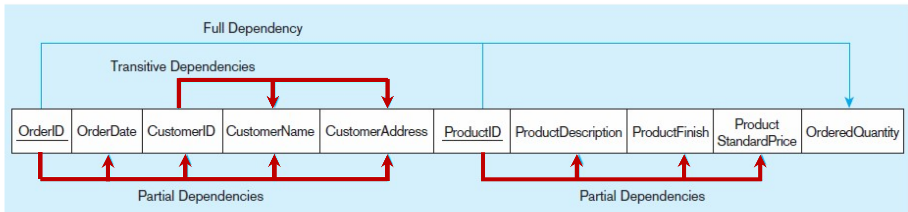
ProductID → ProductDescription, ProductFinish, ProductStandardPrice

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

**CustomerID → CustomerName, CustomerAddress**

## 1.6. Why Partial and Transitive FDS cause the anomalies?

Functional Dependency Diagram for INVOICE (Figure 4-27)




INVOICE Relation

**INVOICE (OrderID, ProductID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)**

FDs

OrderID, ProductID ➔ OrderQuantity

**ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice**

**OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress**

CustomerID → CustomerName, CustomerAddress



## Have a look at the anomalies in INVOICE relation (table) that has partial and transitive FDs:

| 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 |

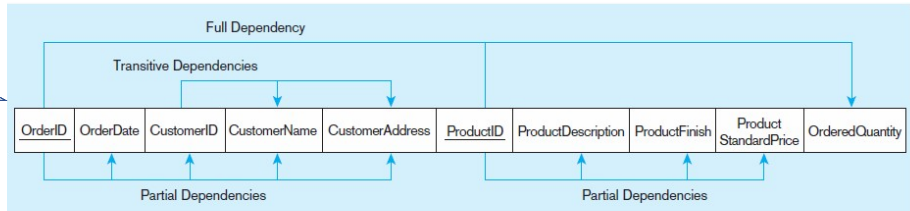
FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

- **Insertion Anomaly**: if new product is ordered for order **1007** of existing customer, **customer data** must be re-entered, causing duplication
- **Deletion Anomaly** : if we delete the **Dining Table** from Order **1006**, we lose information concerning this item's finish and price
- **Update (Modification) Anomaly**: 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.

## Have a look at the relation (in two formats) and review the answer to “Why Partial and Transitive FDS cause the anomalies?”

Functional Dependency Diagram for INVOICE (Figure 4-27)




INVOICE Relation

INVOICE (OrderID, ProductID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)

**Why do these anomalies exist?**

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

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



# 2. Well-Structured Relations and Data Normalization

## 2. 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 (update) Anomaly:** changing data in a row forces changes to other rows because of duplication

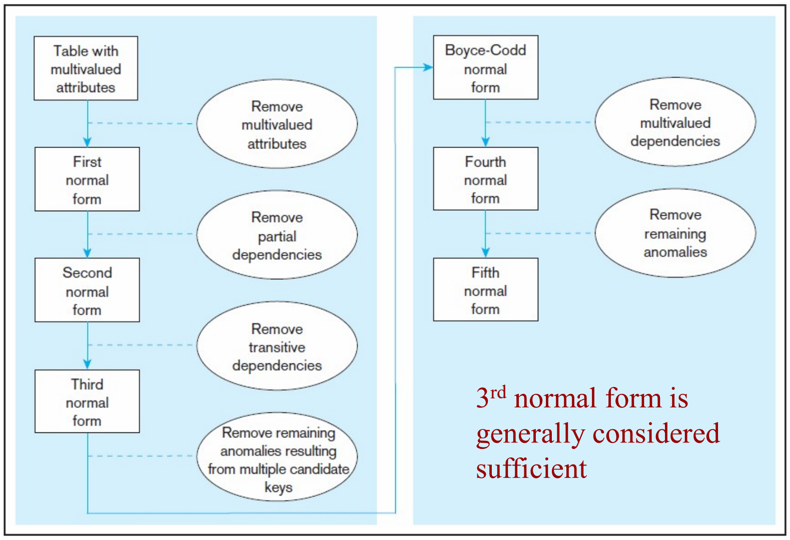
**General rule of thumb: A table should not pertain to more than one entity type.**



## 2. Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that **avoid unnecessary duplication of data**
- The process of decomposing relations with anomalies to produce smaller, **well-structured** relations

# 3. Steps in Normalization (Figure 4.22)






# 4. First Normal Form

I. **No derived attribute** (Derived attribute can be calculated or derived using some business rule from other attributes)

- **Example:** In the following relation StuAge is a derived attribute and should be removed from the relation
  - Student(StudentID, StuDateOfBirth, **StuAge**, StuAddress)
  - Student(StudentID, StuDateOfBirth, StuAddress)

II. **Every attribute value is atomic** (Atomic attributes can't be divided into subparts)

- **Example:** In the following relation StuAddress is a non-atomic attribute and should be divided to smaller parts
  - Student(StudentID, StuDateOfBirth, **StuAddress**)
  - Student(StudentID, StuDateOfBirth, **StuUnitNumber, StuStreet, StuSuburb, StuState**)

  **Note: in the following slides "Customer Address" has been ASSUMED as an atomic attribute.**

III. **No multivalued attributes** (Multivalued attributes can have more than one value at a time)

Based on this, a relation is in first normal form if:

- There are no repeating groups in the relation.
- A Primary key has been defined, which uniquely identifies each row in the relation

- **Example:** In the next slides

## Table with multivalued attributes, not in 1st normal form

- **Multivalued attributes:** Attributes that can have more than one value at a time.
- **A relation is in first normal form (1NF) if:**
  - There are no repeating groups in the relation.
  - A Primary key has been defined, which uniquely identifies each row in the relation

| 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 |

FIGURE 4-25 INVOICE data (Pine Valley Furniture Company)

**Note: This is NOT a relation.**



## Table with no multivalued attributes and unique rows, in 1st normal form (1NF)

| 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 |

FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

**Note: This is a relation and is in 1NF, but not a well-structured one.**

**This relation is still**

## Anomalies in this Table (Review)

| 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 |

FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

- **Insertion Anomaly**: if new product is ordered for order **1007** of existing customer, **customer data** must be re-entered, causing duplication
- **Deletion Anomaly** : if we delete the **Dining Table** from Order **1006**, we lose information concerning this item's finish and price
- **Update (Modification) Anomaly**: 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.



## 5. Second Normal Form

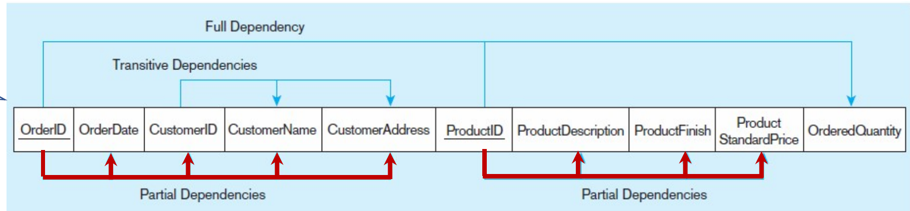
- **1NF PLUS every non-key attribute is fully functionally dependent on the <u>ENTIRE primary key</u>**
  - Every non-key attribute must be defined by the **entire key**, not by only part of the key
  - No partial functional dependencies

- **Solution:**

1. Create new relation for each Primary Key (PK) and move non-key attributes that are only dependent on this PK.
2. Consider this PK as a Foreign Key (FK) in the original table (relation)

## Partial Functional Dependencies in INVOICE (Book and Tutorial Formats)

Functional Dependency Diagram for INVOICE (Figure 4-27)




INVOICE Relation

**INVOICE (<u>OrderID</u>, <u>ProductID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)**

FDs

OrderID, ProductID → OrderQuantity

**ProductID → ProductDescription, ProductFinish, ProductStandardPrice**

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

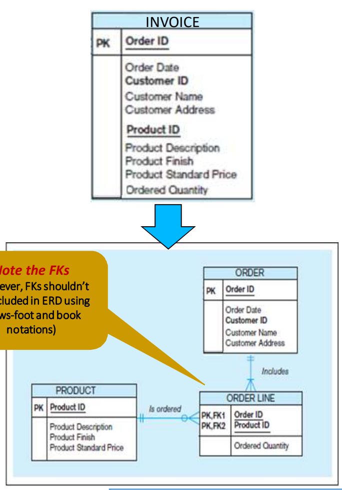
CustomerID → CustomerName, CustomerAddress

**Based on the FDs there are partial functional dependencies in this relation → Therefore, This Relation (INVOICE) is NOT in 2nd Normal Form**

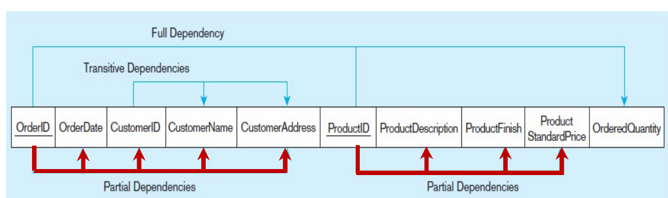


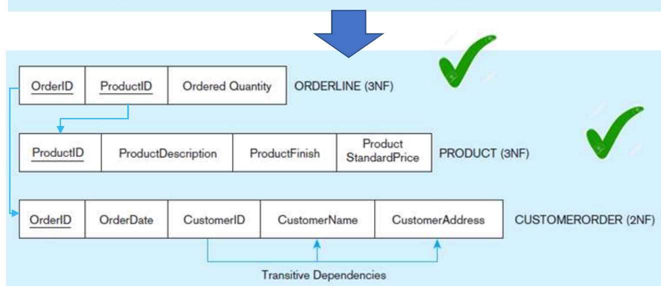
## Removing Partial Dependencies (using Book Format in Figure 4-28)

1. Create new relation for each Primary Key (PK) and move non-key attributes that are only dependent on this PK.
2. Consider this PK as a Foreign Key (FK) in the original table (relation)




*Note the FKs*
(However, FKs shouldn't be included in ERD using crows-foot and book notations)



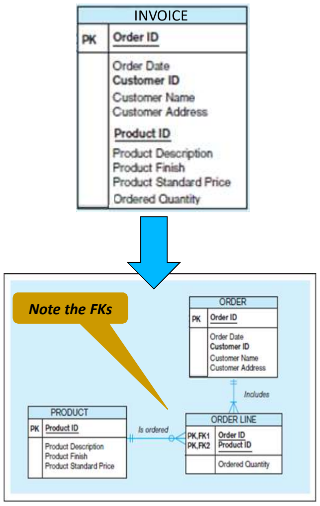




Getting it into Second Normal Form

**Partial dependencies are removed, but there are still transitive dependencies**

## Removing Partial Dependencies (using Tutorial Format for Relations)

1. Create new relation for each Primary Key (PK) and move non-key attributes that are only dependent on this PK.
2. Consider this PK as a Foreign Key (FK) in the original table (relation)




INVOICE (<u>OrderID</u>, <u>ProductID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity) ✗

- OrderID, ProductID ➔ OrderQuantity
- ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice
- OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress
- CustomerID → CustomerName, CustomerAddress

➢ **New Relations:** Getting it into Second Normal Form

Note the FKs

- ORDER-LINE (<u>OrderID*</u>, <u>ProductID*</u>, OrderQuantity) ✓
- PRODUCT (<u>ProductID</u>, ProductDescription, ProductFinish, ProductStandardPrice) ✓
- ORDER (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress) ✗

**Partial dependencies are removed, but there are still transitive dependencies in ORDER relation.**

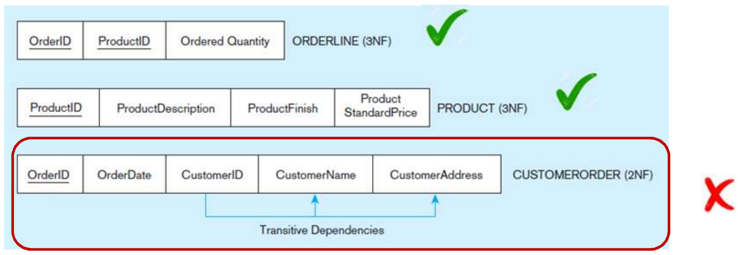


# 6. Third Normal Form

- ➢ **2NF PLUS no transitive dependencies** (functional dependencies on non-primary-key attributes)
- ➢ Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third.
- ➢ (If it is **non-transitive** then each non-key attribute is not dependent on, or a determinant for, any other non-key attributes).

- ➢ **Solution:**
  1. Non-key determinant with transitive dependencies go into a new table;
  2. Non-key determinant becomes primary key in the new table, and
  3. Stays as foreign key in the old table (relation)

# By now …

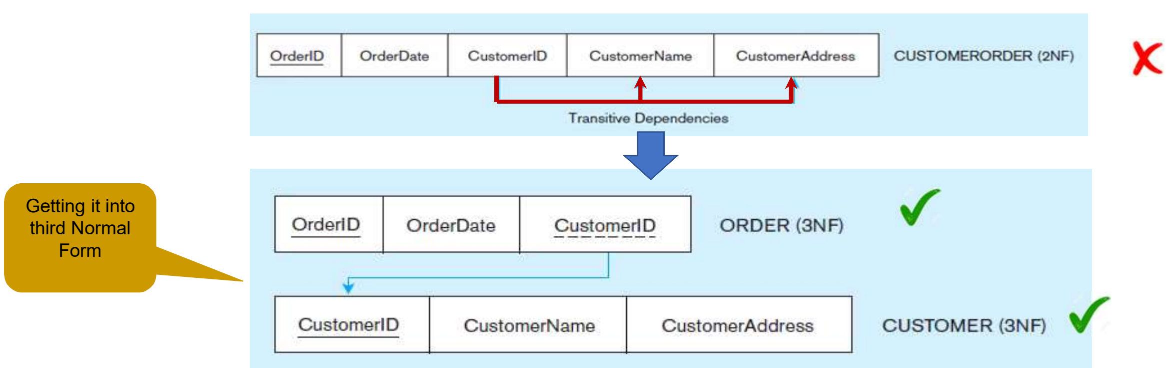



OrderID, ProductID ➔ OrderQuantity
**ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice**
**OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress**
CustomerID → CustomerName, CustomerAddress



# **Removing Transitive Dependencies** (Book Format in Figure 4-29)

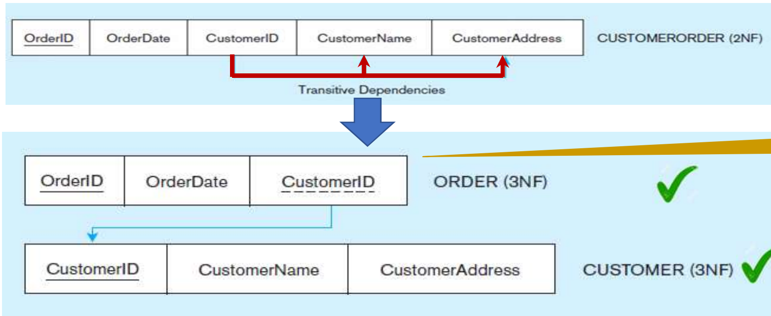
- **Solution:**
  1. Non-key determinant with transitive dependencies go into a new table;
  2. Non-key determinant becomes primary key in the new table, and
  3. Stays as foreign key in the old table




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

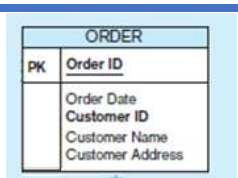
## Removing Transitive Dependencies (Using Book Format in Figure 4-29)

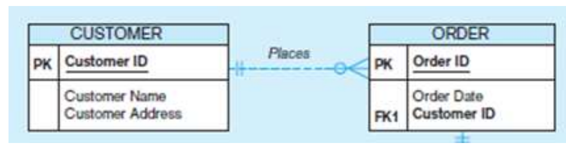
1. Non-key determinant with transitive dependencies go into a new table;
2. Non-key determinant becomes primary key in the new table, and
3. Stays as foreign key in the old table




Getting it into third Normal Form

*Note the FKs*






## Removing Transitive Dependencies (Using Tutorial Format for Relations)

1. Non-key determinant with transitive dependencies go into a new table;
2. Non-key determinant becomes primary key in the new table, and
3. Stays as foreign key in the old table

Original Relation

**ORDER (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress)**

- **FDs:**

**OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress**

**CustomerID → CustomerName, CustomerAddress**

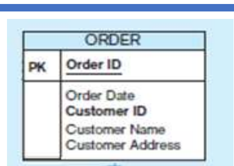
Getting it into third Normal Form

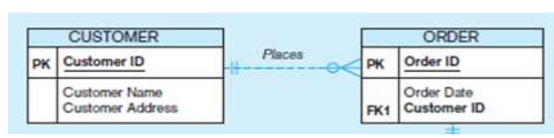
- **New Relations:**

**ORDER (<u>OrderID</u>, OrderDate, CustomerID*)**

**CUSTOMER (<u>CustomerID</u>, CustomerName, CustomerAddress)**

*Note the FKs*



## Normalized logical Design

| 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 |

FIGURE 4-25 INVOICE data (Pine Valley Furniture Company)

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

CustomerID ➔ CustomerName, CustomerAddress

OrderID ➔ OrderDate, CustomerID

OrderID, ProductID ➔ OrderQuantity

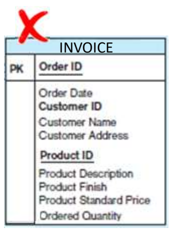
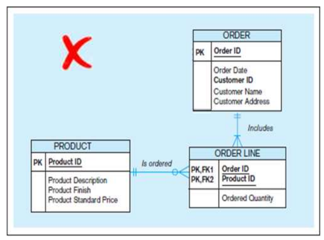
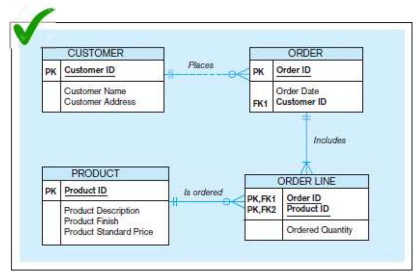



Figure 4-30 shows the result of normalization, yielding four separate relations where initially there was only one.






## Summary

- State two properties of candidate keys
- Determining keys from FDs
- Define first, second, and third normal form
- Use normalization to decompose anomalous relations to well-structured relations

## Next Lecture…

1. Review of 1NF, 2NF and 3NF.
2. Boyce Codd Normal Form (BCNF) → Optional
   1.1. BCNF Example 1
   1.2. BCNF Example 2
3. Creating New Relations in a Higher Normal Form → Optional
4. Role of Normalization → Optional
5. Advantages of Refinement (Top-Down) Approach → Optional
6. Tutorial 6 – Section one