CHAPTER 4:
LOGICAL DATABASE DESIGN & THE RELATIONAL MODEL
(*PART II - NORMALIZATION*)

Shakeel Ahmad

Modern Database Management
12th Edition
Jeff Hoffer, Ramesh Venkataraman, Heikki Topi

New 2017:
1. "Anatomy" of func dep
2. Common errors in normalization

We learned in Chap 2
Database **Analysis**;
We are moving into
Database **Design**

OBJECTIVES – PART 2

- Define terms
- List five properties of relations
- State two properties of candidate keys
- Define first, second, and third normal form
- Describe problems from merging relations
- Transform E-R and EER diagrams to relations
- Create tables with entity and relational integrity constraints
- Use normalization to convert anomalous tables to well-structured relations

RELATION [REVIEW FROM PART I]

- A relation is a named, two-dimensional table of data.
- A table consists of rows (records) and columns (attributes or fields).
- 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 ***1st Normal form***

CORRESPONDENCE WITH E-R MODEL (FROM PART 1 W NEW ANNOTATION)

- **Relations (tables)** correspond with **entity types** or with many-to-many relationship types.
- **Rows** correspond with **entity instances** or with many-to-many relationship instances.
- **Columns** correspond with **attributes.**

- NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).
  - Relations: tables;
  - relationships: interaction among tables (PK-FK pair)

(START OF BRAND NEW SLIDES)

DATA NORMALIZATION

Which columns go to which table

- Normalization is a formal process for deciding which attributes should be grouped together in a relation
  - so that all anomalies are removed
- Normalization is based on *normal forms* and *functional dependencies*
  - Functional dependencies:
    - full, partial, transitive
  - Normal form:
    - 1st, 2nd, 3rd

2nd normal form: no partial dependency
3rd normal form: no transitive dependency

DATA NORMALIZATION

- The process of **decomposing** relations with anomalies
  - to produce **smaller**, ***well-structured*** relations ("anomalies are removed" – last slide)
- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that
  - ***avoid unnecessary duplication of data***
- **Four-point goals of normalization: P. 177**

WELL-STRUCTURED RELATIONS

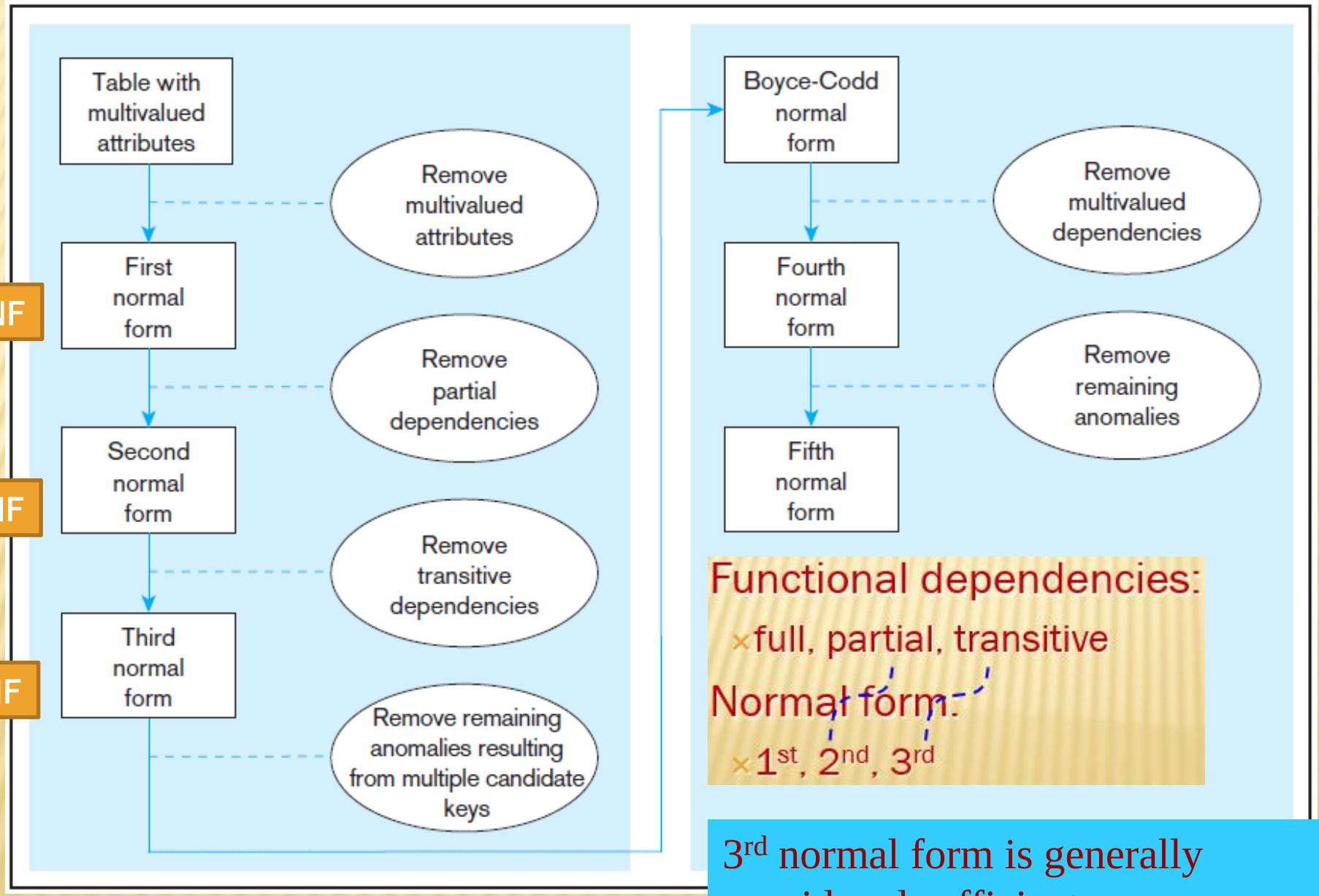
Source of anomalies: one table contains N entity types

- 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 pertain to only one entity type

Make one table contain ONLY ONE entity type

Figure 4.22 Steps in normalization



PURPOSE OF NORMALIZATION (2017 NEW)

- Normalization takes a "big", clumsy table (containing multiple themes/multiple logical dependencies/multiple entity types), and decompose it into several smaller tables that are:

1. One theme/logical dependency in each table;
2. Each "smaller" table is in full dependency;
3. With referential integrity among the related tables

Above three: Final status of normalization

FUNCTIONAL DEPENDENCIES AND KEYS

- Functional Dependency: The value of one attribute (the **determin*ant***) determines the value of another attribute (the dependant)
  - A determinant in a database table is any **attribute** that you can use to determine the values assigned to other attribute(s) ***in the same row***.
  - If we call the determinant "D", and other attributes "A", then
    - Knowing D (value), we can know A (values)
      - Denoted as: D ➔ A
    - Or: "Given a D-value, we can have a specific A-value"

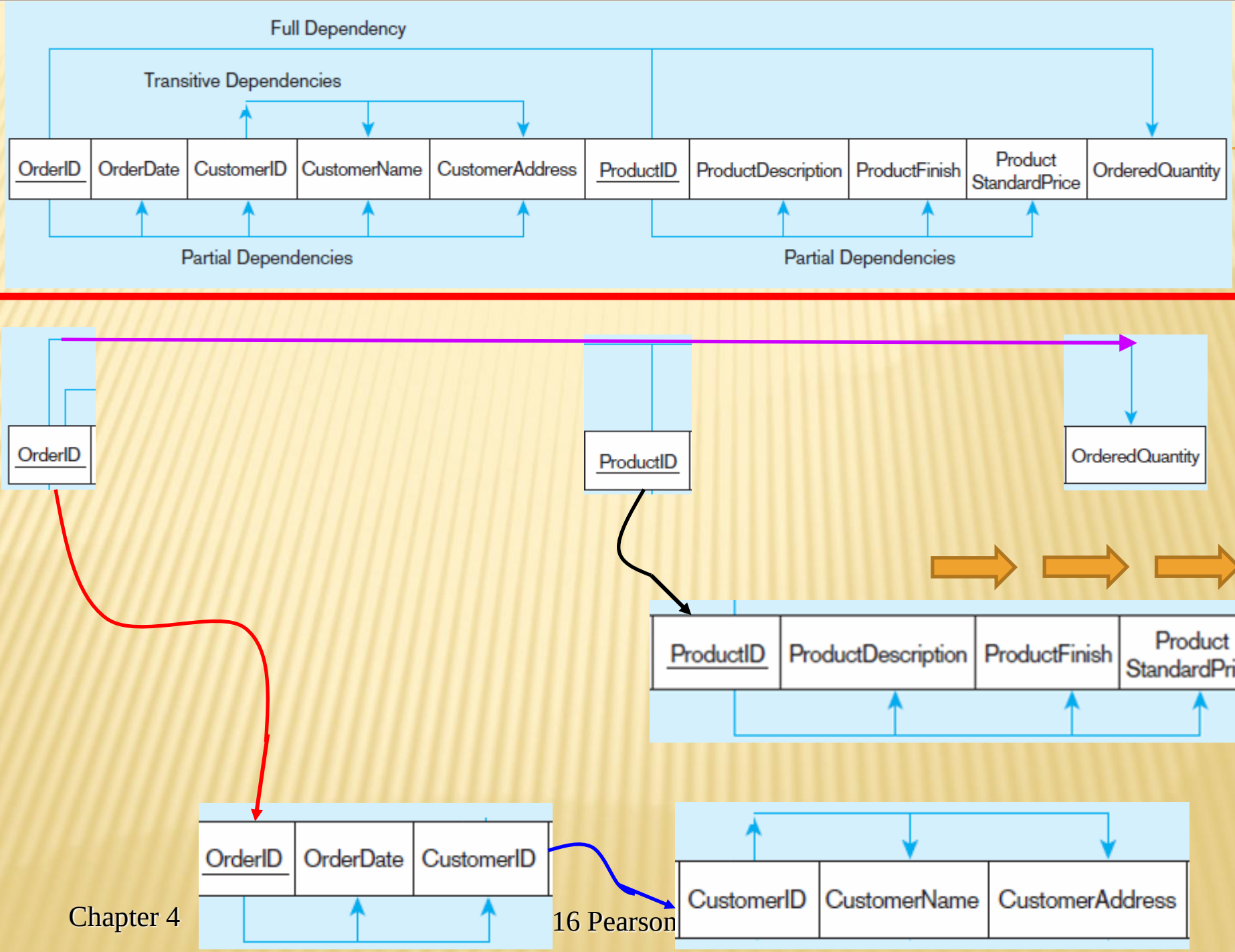
Example: PP. 177~178

Notation

FUNCTIONAL DEPENDENCIES AND KEYS (CONT)

- Candidate Key:
  - A unique identifier. One of the candidate keys will become the primary key
    - E.g. perhaps there is both credit card number and SS# in a table…in this case both can be candidate keys
- The determinant in a relation is a …
- Notation of functional dependencies (Func. Dep.): P. 179, illustrated w Fig 4-23
- Note: Distinguish "curvy arrows" (Ref. Integrity) and "straight arrows" (Func Dep)

Example:


Full Dependency
Transitive Dependencies
OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderedQuantity
Partial Dependencies
Partial Dependencies
OrderID
ProductID
OrderedQuantity
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderID
OrderDate
CustomerID
CustomerID
CustomerName
CustomerAddress

DIFFERENTIAE FUNC. DEPEND. ARROWS FROM REF INTEGRITY ARROWS (ADDED 2017)

Perspective	Func. Dep.	Ref Integrity
Scope	Within relation	Across relations
Shape	Straight line	Curvy line
Connection & direction	Determinant → determinee	FK (one table) → PK (another table)
End points	Every field/ attribute – ALL fields must be involved	ONLY between the PK/FK; only TWO fields (FK, PK) involved

EXAMPLE–FIGURE 4-2B

Example of Anomalies

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
110	Chris Lucero	Info Systems	43,000	C++	4/22/2015
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/2015
150	Susan Martin	Marketing	42,000	Java	8/12/2015

Question–Is this a relation?

Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key?

Answer–Composite: EmpID, CourseTitle

ANOMALIES IN THIS TABLE

1. **Insertion**–can't enter a new employee without having the employee take a class
2. **Deletion**–if we remove employee 140, we lose information about the existence of a Tax Acc class
3. **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: __EMPLOYEE_ and COURSE

This results in data duplication and an unnecessary dependency between the entities

EXAMPLE–FIGURE 4-2B

EMPLOYEE2

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

FULL DEPENDENCY ("FUNC DEP")

- The dependency that departs from the primary key (could be single-field or composite) and ends at non-key attributes
  - The legitimate dependency (the "Good", the "ALWAYS HAVE")
  - Relations ALWAYS have a full dependency – every relation MUST have
    - There are not always partial or transitive dependencies

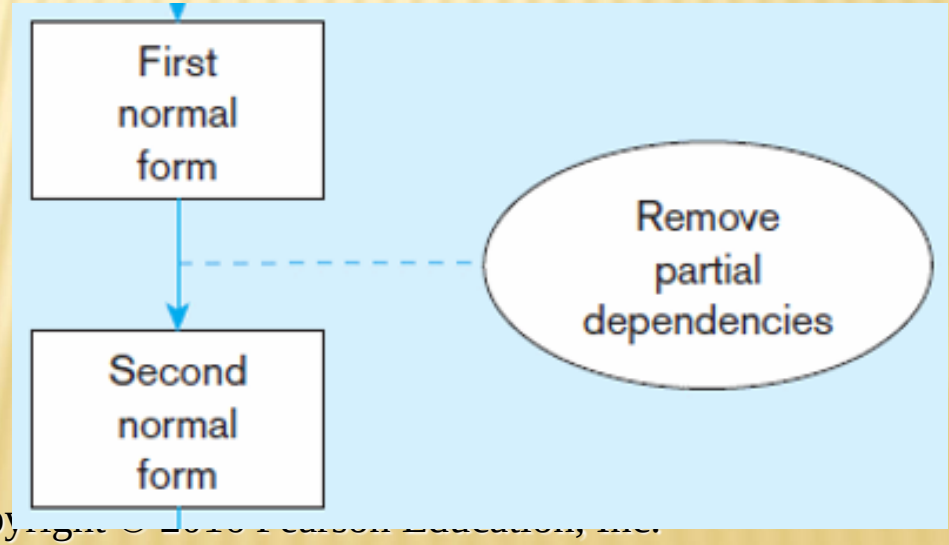
Basic concept

Figure 4.22 Steps in normalization

A **partial functional dependency** exists when a nonkey attribute is functionally dependent on part (but not all) of the primary key. As you can see, the following partial dependencies exist.

- From 1st normal form (1NF) to 2nd normal form (2NF):
  - Remove partial dependencies ➔ 2nd normal form
  - Partial dependency ("PD"): when there are two attributes working as the determinant (_____ key), some determinees depend ONLY on one PART of the key

PD



★ A general definition of 2NF: PP. 183~4

Not always exist

Second normal form (2NF)
A relation in first normal form in which every nonkey attribute is fully functionally dependent on the primary key.

Partial functional dependency
A functional dependency in which one or more nonkey attributes are functionally dependent on part (but not all) of the primary key.

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

The first of these partial dependencies (for example) states that the date on an order is uniquely determined by the order number and has nothing to do with the ProductID.

To convert a relation with partial dependencies to second normal form, the following steps are required:

1. Create a new relation for each primary key attribute (or combination of attributes) that is a determinant in a partial dependency. That attribute is the primary key in the new relation.
2. Move the nonkey attributes that are dependent on this primary key attribute (or attributes) from the old relation to the new relation.

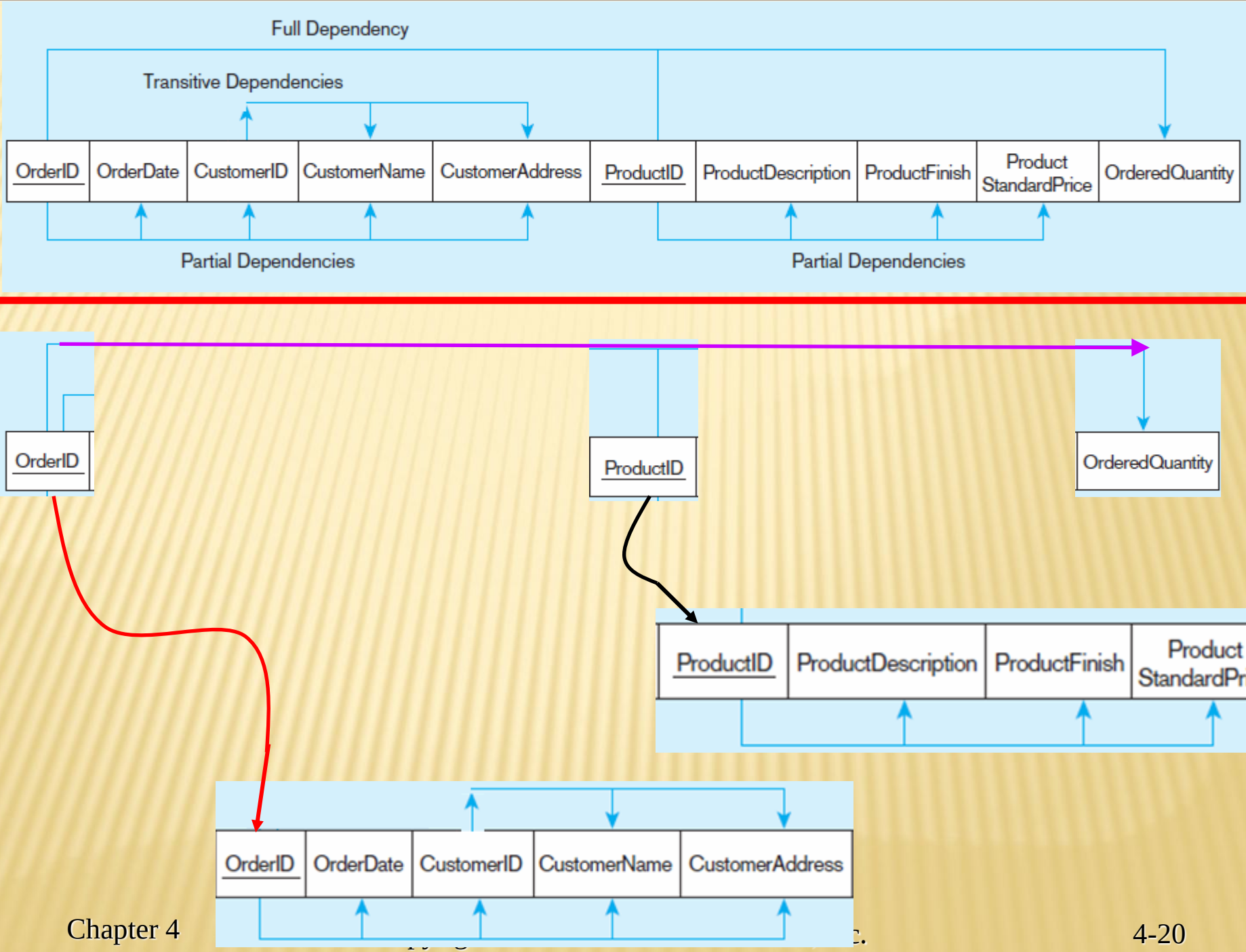
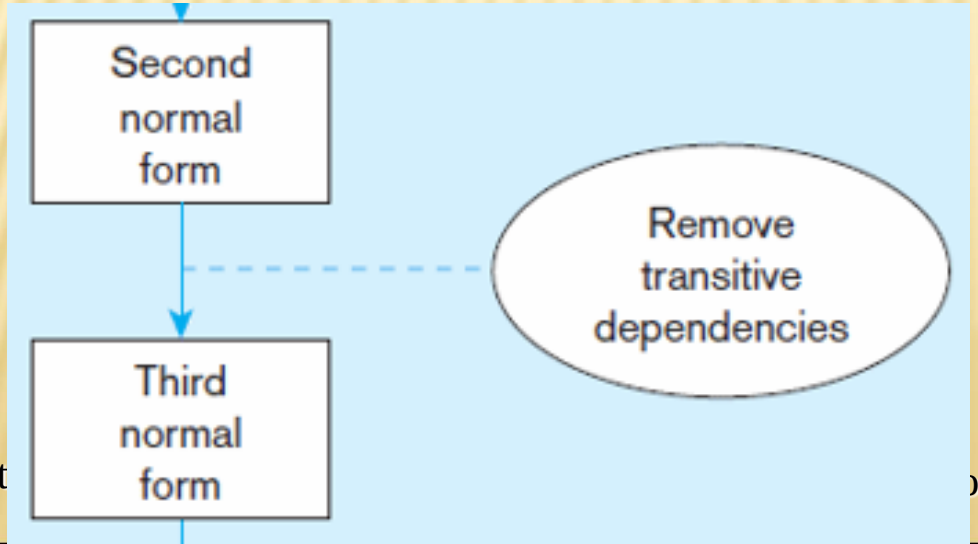

Full Dependency
Transitive Dependencies
OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderedQuantity
Partial Dependencies
Partial Dependencies
OrderID
ProductID
OrderedQuantity
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress

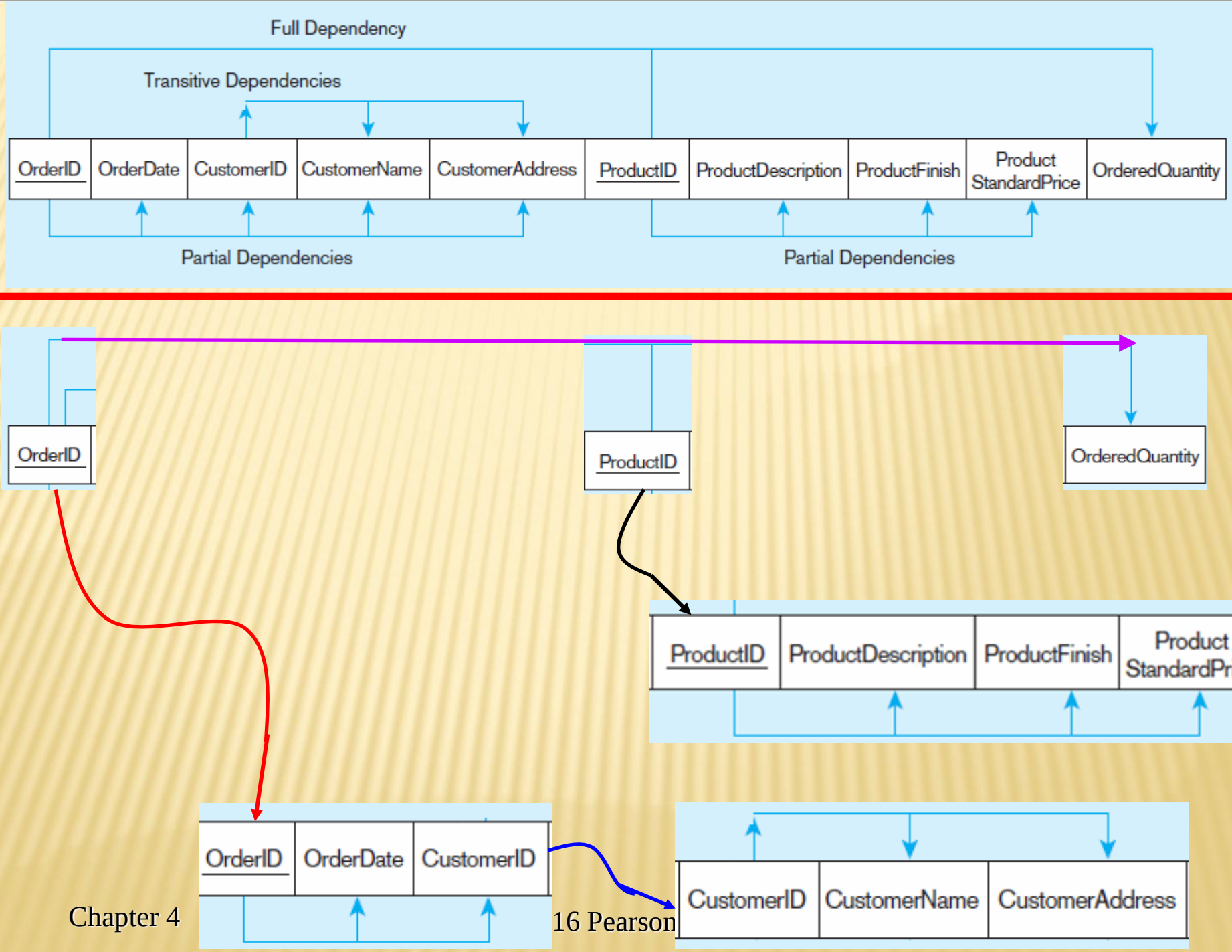
Figure 4.22 Steps in normalization

- From 2nd normal form (2NF) to 3rd normal form (3NF):
  - Remove transitive dependencies ➔ 3rd normal form
  - Transitive dependency ("TD"): when there are non-key attributes acting/behaving as a determinant (of other attributes) - determinees depend on a non-key attribute

TD



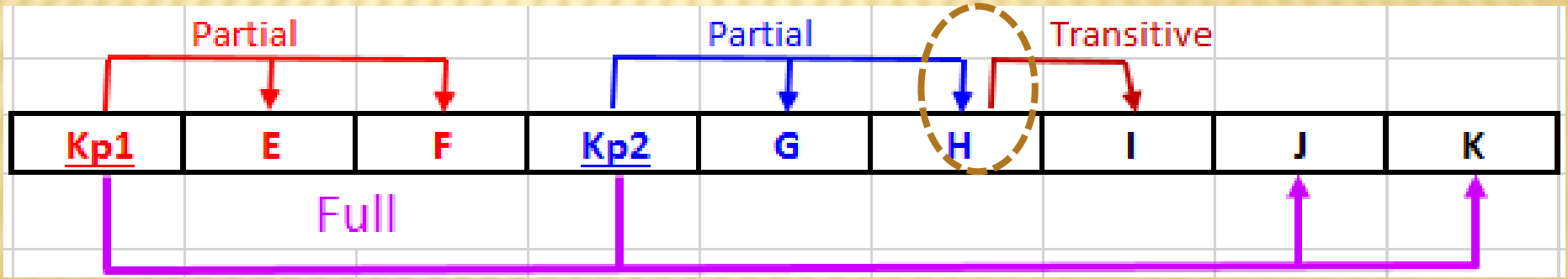
Not always exist


Full Dependency
Transitive Dependencies
OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderedQuantity
Partial Dependencies
Partial Dependencies
OrderID
ProductID
OrderedQuantity
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderID
OrderDate
CustomerID
CustomerID
CustomerName
CustomerAddress

PARTIAL, TRANSITIVE, AND FULL FUNCTIONAL DEPENDENCIES

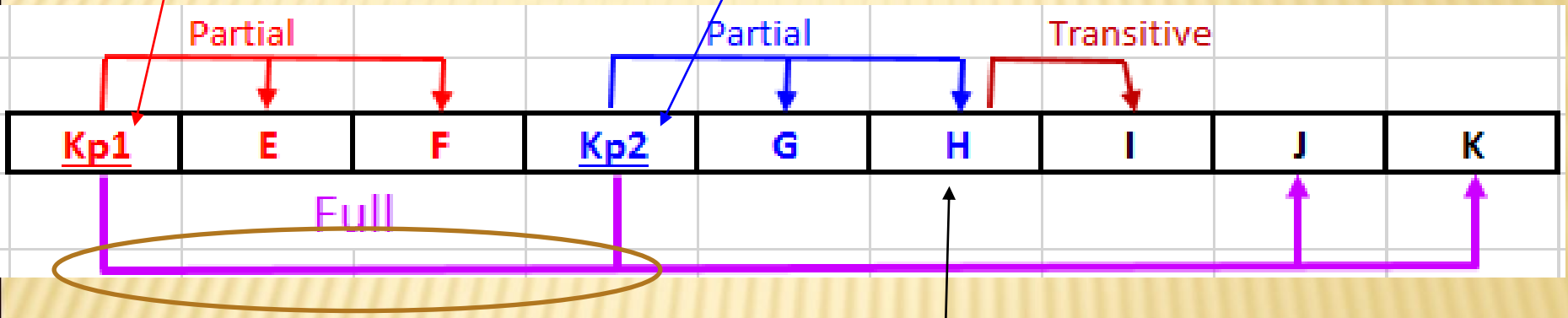
1. Partial: Note the PK here has TWO PARTS
2. Transitive: Like skipping a stone

3. Full: non-key attributes FULLY depend on whole set pf key attributes



FULL, PARTIAL, AND TRANSITIVE DEPENDENCY CLOSE-UPS

- "Kp1" – Key part 1; "Kp2" – key part 2



- Kp1 + Kp2 is the whole key
-

H here is non-key

EXISTENCE OF PARTIAL, TRANSITIVE, AND FULL FUNCTIONAL DEPENDENCIES (NEW 2017)

- Full: non-key attributes FULLY depend on whole set of key attributes (can be one field, can be 2+)
  - ALL relations have full dependencies – always exist
- Partial: Note the PK here has TWO PARTS
  - Only when there is a _____ key need we examine for PD
  - → not always exist
- Transitive: Like skipping a stone – "PK determines non-key (PK → attr 1, attr1→ attr2, etc); → not always exist

Use as a check

FIRST NORMAL FORM

- No multivalued attributes
  - Every attribute value is atomic
- Fig. 4-25 *is not* in 1st Normal Form (multivalued attributes) ➔ it is not a relation.
- Fig. 4-26 *is* in 1st Normal form.
- ***All relations* are in 1st Normal Form.**

Fig 4-25 Table with multivalued attributes, not in 1 NF

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

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Descriptio n	Product F inish	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

Note: this is NOT a relation

Fig 4-26 Table w no multivalued attributes & w unique rows, in 1NF

Repeat

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

Repeat

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

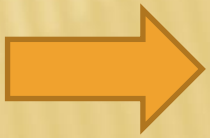
Note: this is a relation, but not a well-structured one

ANOMALIES IN THIS TABLE (4-26)

- **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 customer 1006, we lose information concerning dining table'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



Intuitive examination for the "themes" in the following table

Attributes in 1st theme

Attributes in 2nd theme

Attributes in 3rd theme

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

FIGU 4-26 INVOICE relation (1NF) (Pine Valley Furniture mpany)

Attributes in 4th theme

SECOND NORMAL FORM

- 1NF, PLUS ***every non-key attribute is fully functionally dependent on the ENTIRE primary key***
  - Every non-key attribute must be defined by the entire key, not by only one part of the key
    - ➔ *No partial functional dependencies*
- Discussion: "Entire primary key" implies that this key is a _____ key?
- Therefore, 2NF is relevant only when the original relationship is _____-to-_____? (hint: cardinality)

Figure 4-27 Functional dependency diagram for INVOICE

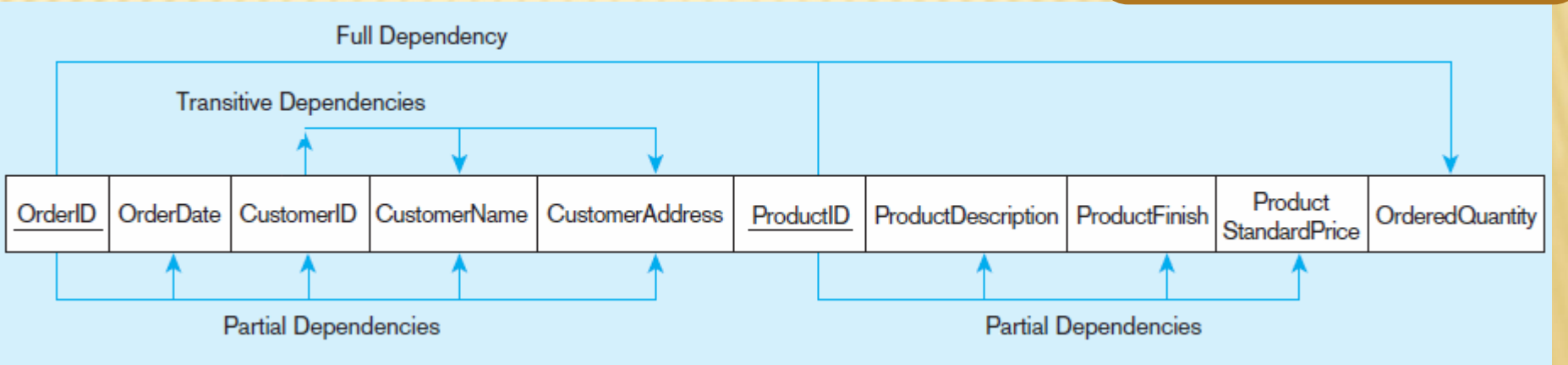
OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

OrderID, ProductID ➔ OrderQuantity

CustomerID ➔ CustomerName, CustomerAddress

Need to know not only func deps, but also the **TYPES** of func deps



Therefore, NOT in 2nd Normal Form

So, what is PD? TD? FD?

SECOND NORMAL FORM

Exercise 1

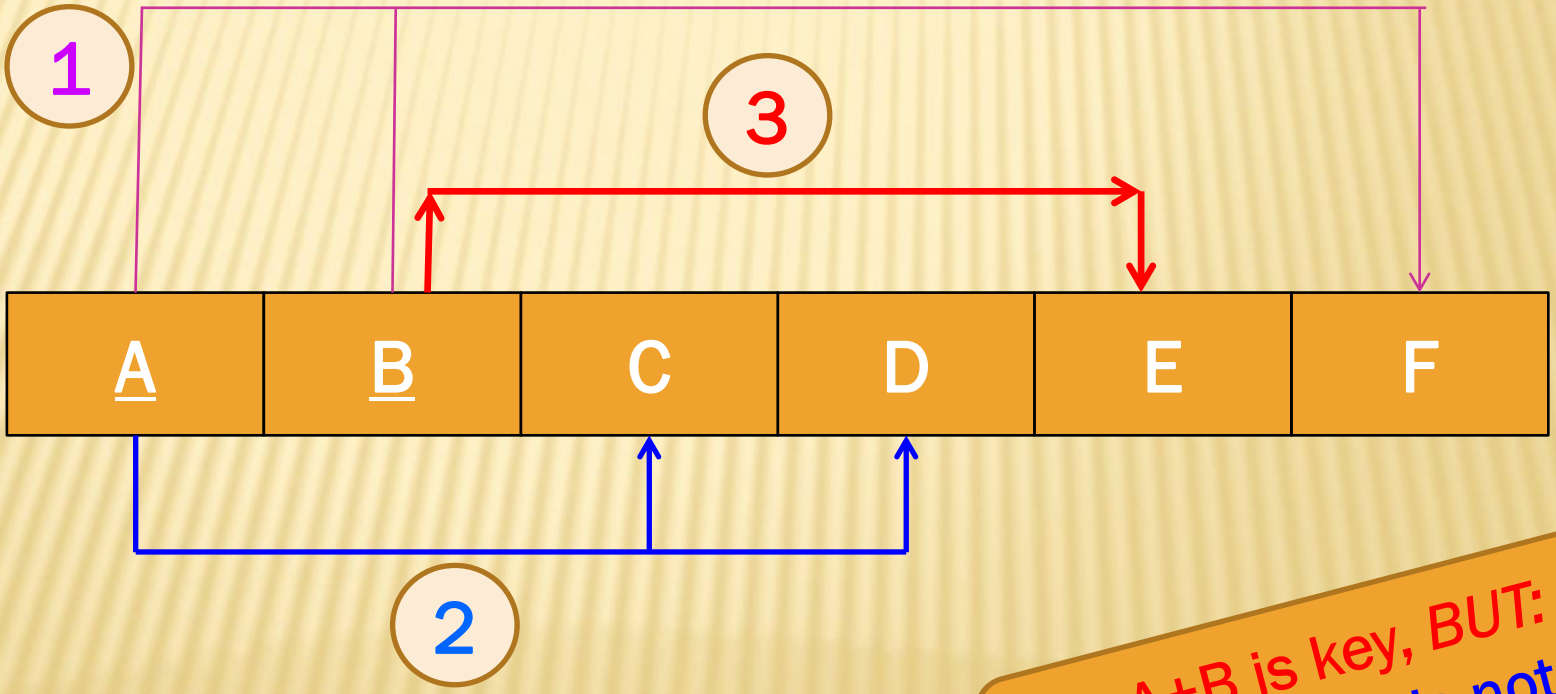
1. Solution (p.183): create a new relation for each primary key component attribute that is determinant in a partial dependency
   - × That attribute is the primary key in the new relation
2. Move the non-key attributes that are dependent on this primary key attribute (“part”) from the old relation to the new relation (**remove them [!] from the old relation**)

+ When A+B is key, (A+B) should ➔c, d, e, f
+ *IF* (A+B)➔f, and A➔c,d; B➔e - violation

SECOND NORMAL FORM

Print 4-33 and 4-34 on ONE sheet of paper helps to visualize

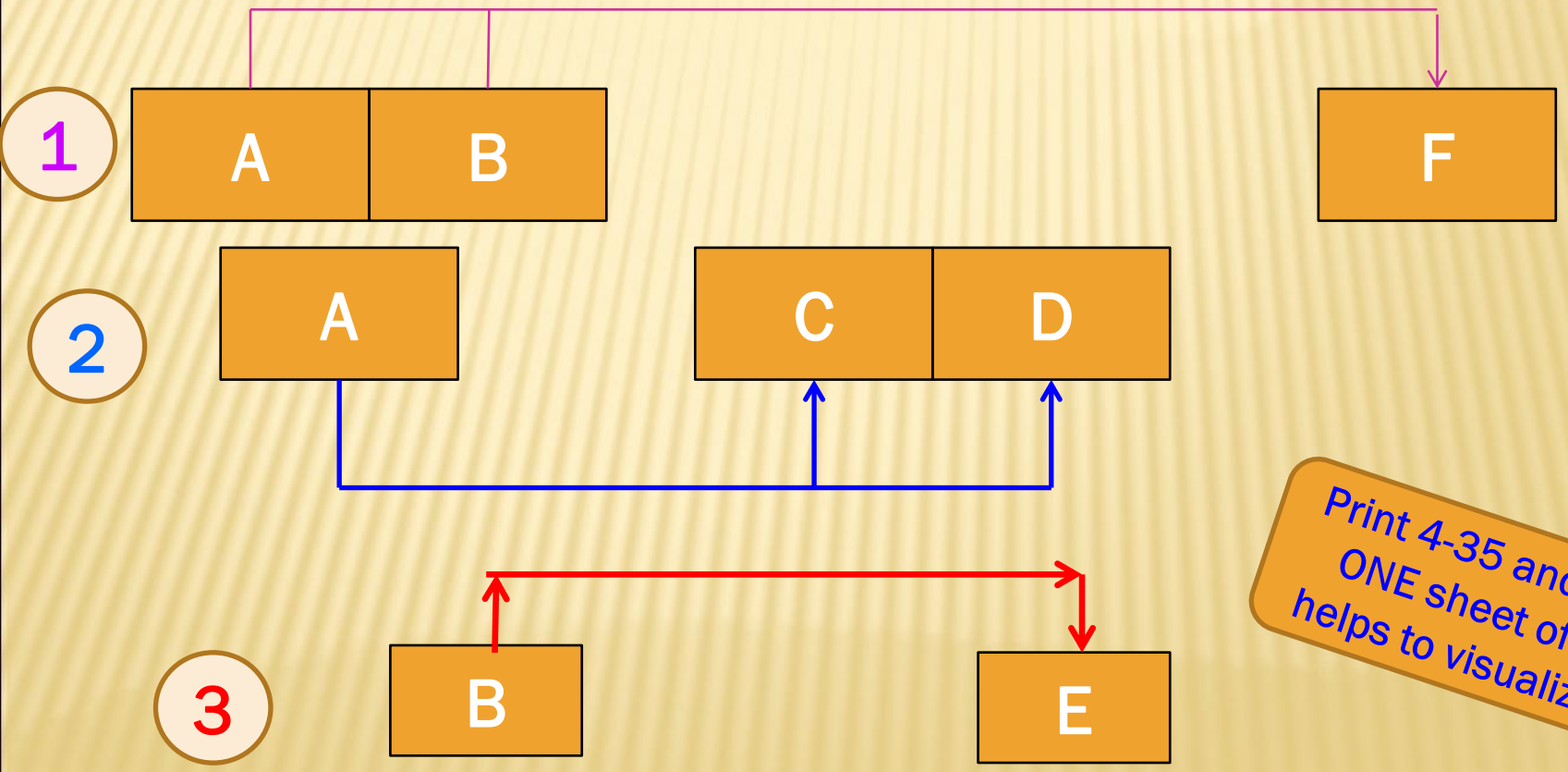
- When A+B is key, then (A+B) ***should*** ➔c, d, e, f
- IF (A+B)➔f, and A➔c,d; B➔e - violation



A+B is key, BUT:
A is not key, B is not key;

SECOND NORMAL FORM - SOLUTION

- IF (A+B)➔f, and A➔c,d; B➔e – violation
- Solution: decompose to Relation (1), (2) , and relation (3)



Print 4-35 and 4-36 on ONE sheet of paper helps to visualize ➔➔

Print 4-32 and 4-33 on ONE sheet of paper helps to visualize

Figure 4-27 Functional dependency diagram for INVOICE

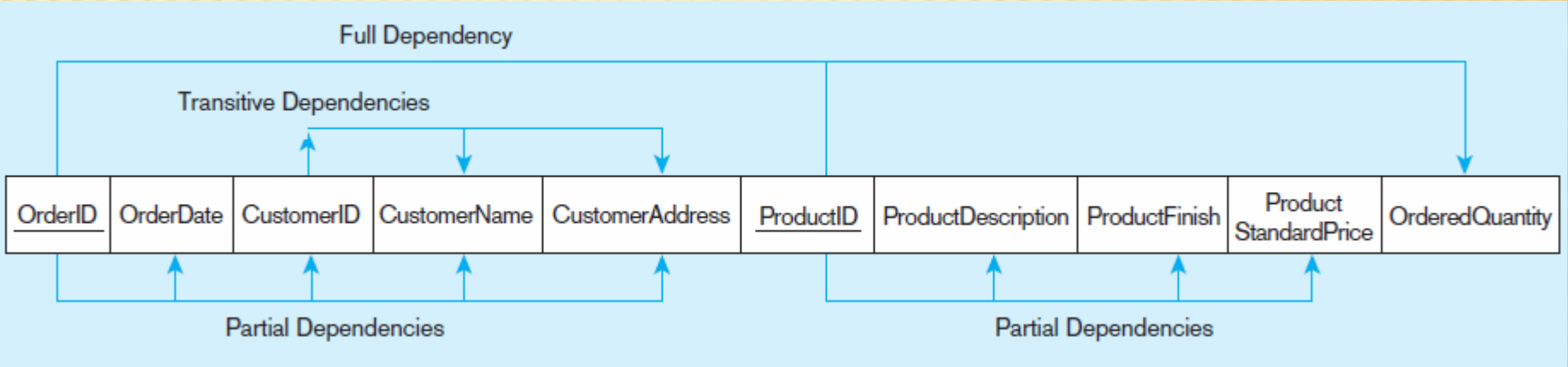
OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

Partial dependency (PD)

OrderID, ProductID ➔ OrderQuantity

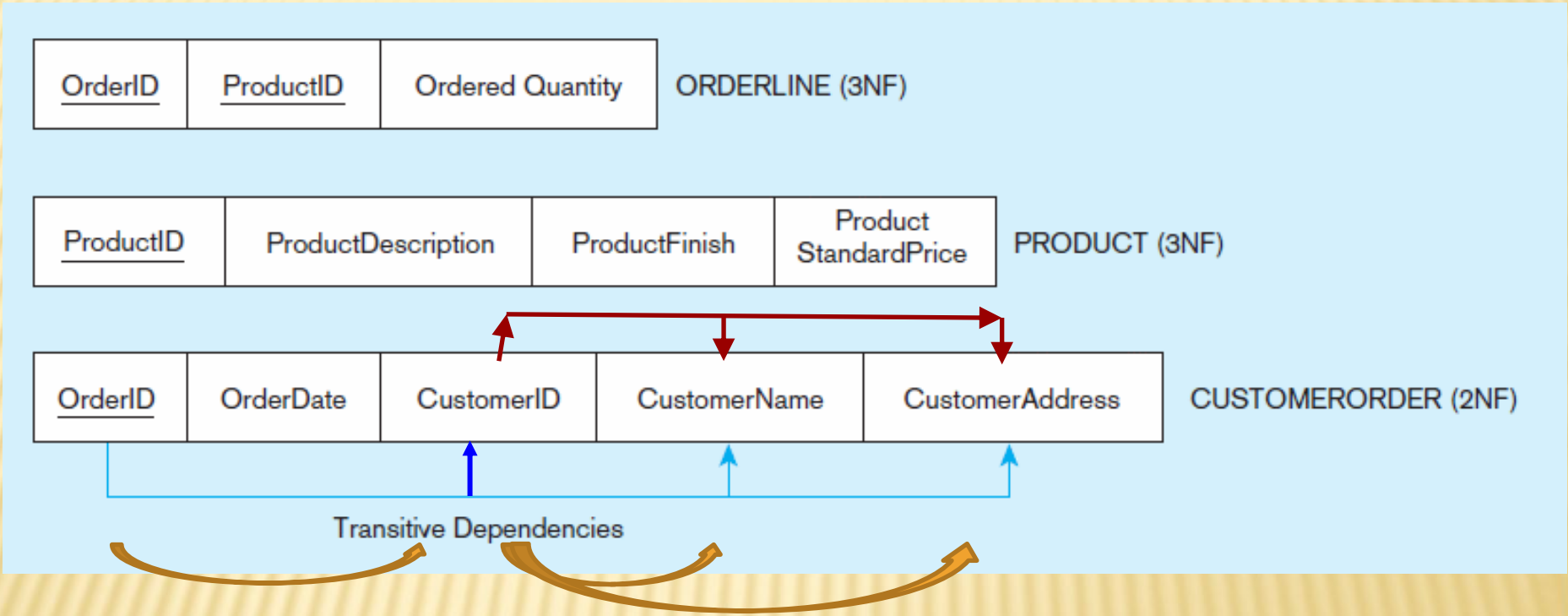
CustomerID ➔ CustomerName, CustomerAddress



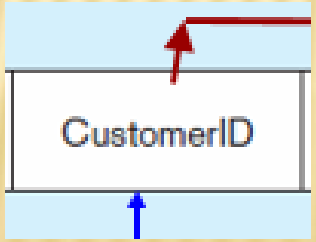
NOT in 2nd Normal Form

Remove PD

Figure 4-28 Removing partial dependencies (PD)



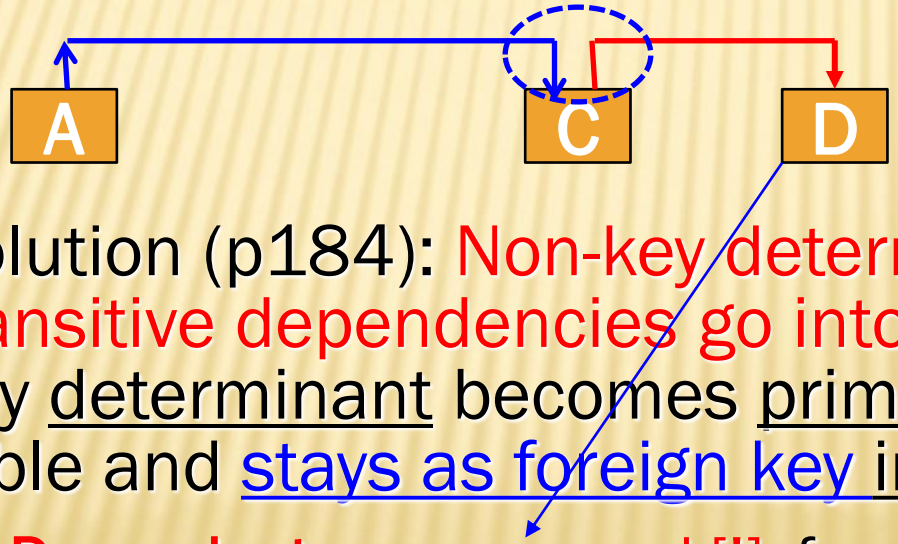
Partial dependencies are removed, but there are still transitive dependencies



"In, and then out" - **Transitive**

THIRD NORMAL FORM

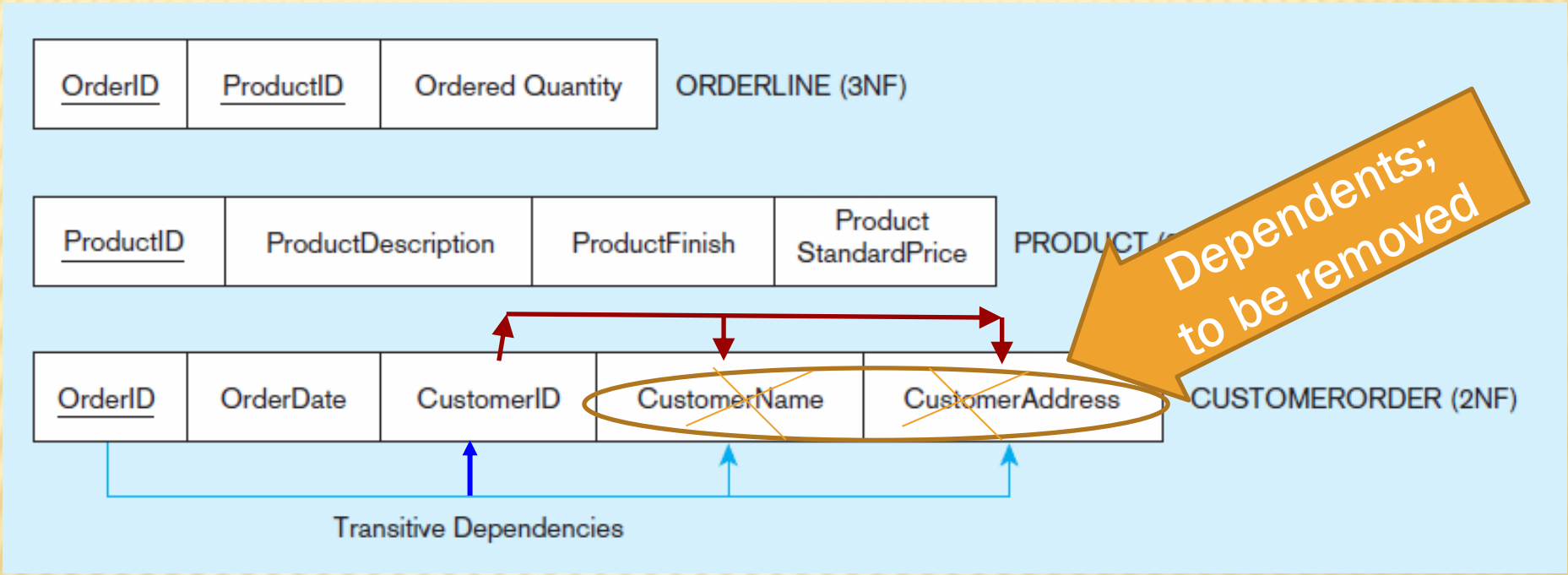
- 2NF PLUS ***no transitive dependencies*** (NO func deps on non-primary-key attributes)
- Note: called transitive because the primary key is a determinant for another attribute, which in turn is a determinant for a third
  - TD: A➔c and c ➔ d in a relation ("in, and then out")



- Solution (p184): Non-key determinant ("c"here) with transitive dependencies go into a new relation; non-key determinant becomes primary key in the new table and stays as foreign key in the old table
  - **Dependents** are removed [!] from the old relation

Print 4-38 and 4-39 on ONE sheet of paper helps to visualize

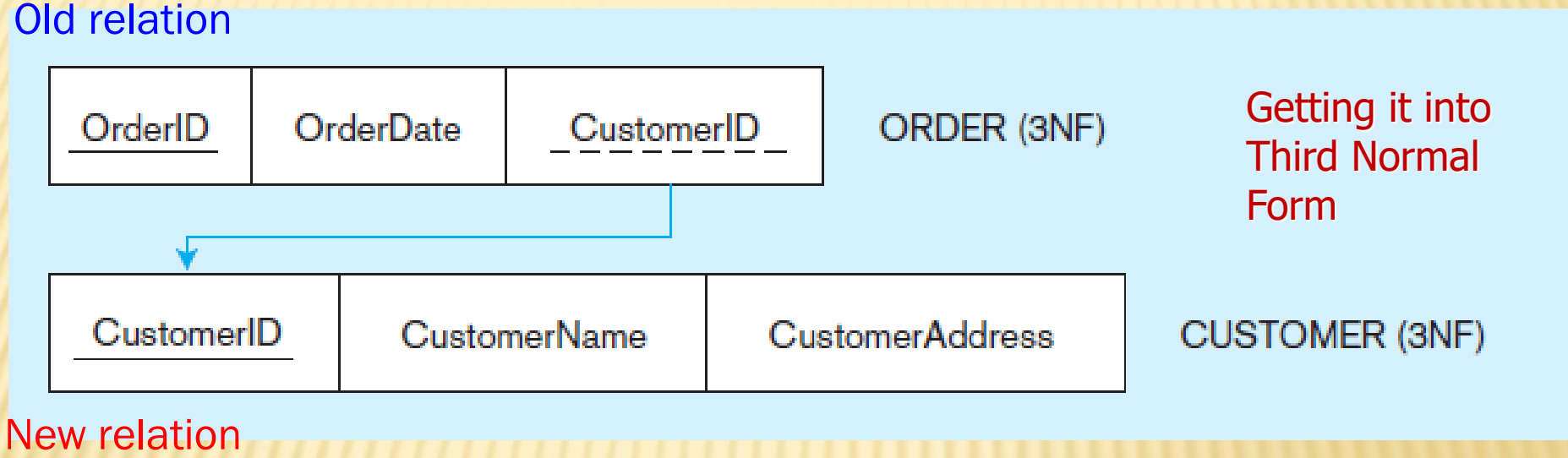
Figure 4-28 Removing transitive dependencies (TD)



Creating:

CustomerID	CustomerName	CustomerAddress

Figure 4-29 Removing partial dependencies



Transitive dependencies are removed

Transitive dependency: A➔c➔d.
Solution: A➔c, and also
c➔d , a separate, different relation

To avoid 3 anomalies

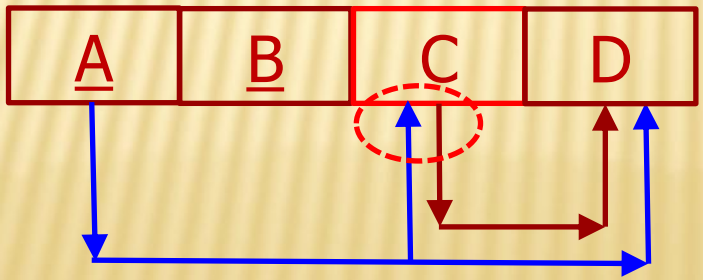
PARTIAL VS TRANSITIVE DEPENDENCY (1)

- Partial: "part" of the key determines others;
  - So only happens when there is a _____ key
  - (A+B) → f; yet also exists B → e
  - Func dep arrows: two arrows out from the composite key should JOINTLY enter all non-key fields; but in violation there are arrows coming out from ONLY ONE PART of the composite key

"Part B of key" determines E: partial

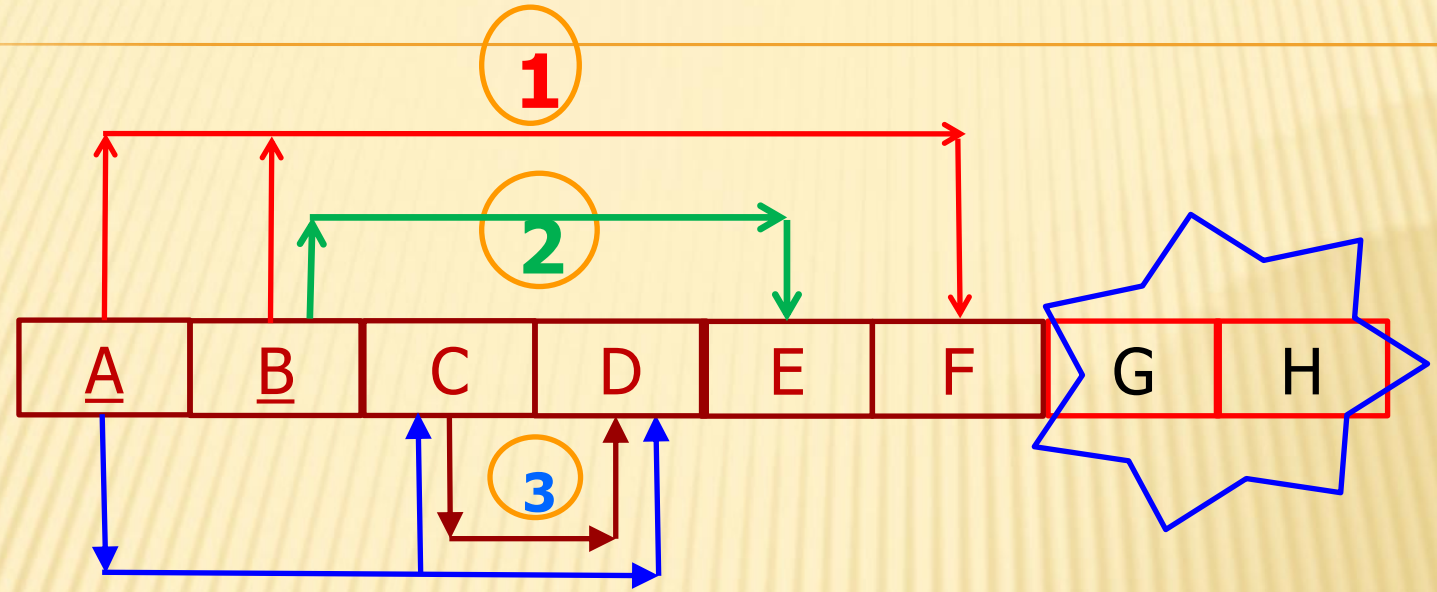
PARTIAL VS TRANSITIVE DEPENDENCY

- Transitive: Key determines non-key-1, and non-key-1 determines non-key-2
  - "You are NOT a key! So you can't determine others!"
  - A (key) → c (non-key-1) → d (non-key-2)
  - Func dep arrows: arrows come out ONLY from the key; but in violation there are arrows also coming out from a non-key attribute



Non-key C determines D: Transitive

PARTIAL VS TRANSITIVE DEPENDENCY

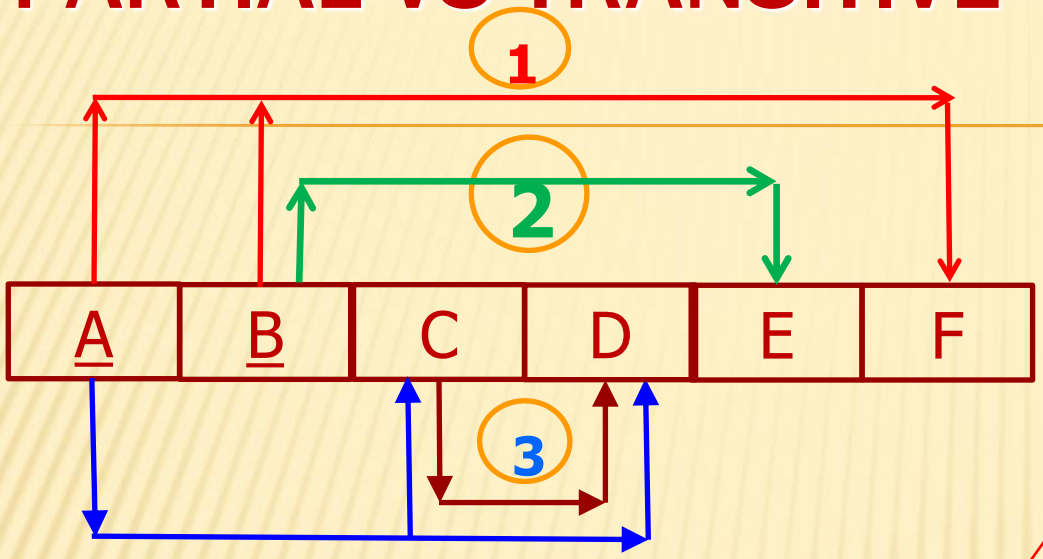


1: _____ dependency?

2: _____ dependency?

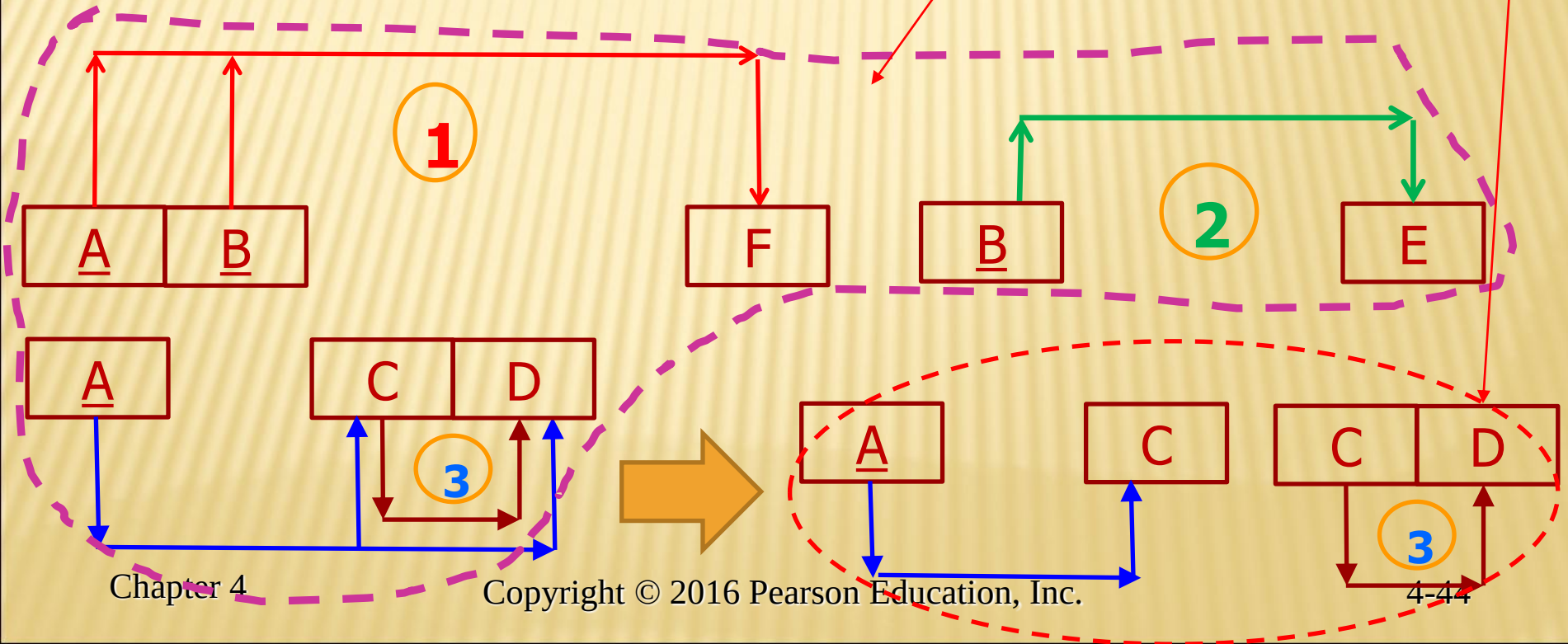
3: _____ dependency?

PARTIAL VS TRANSITIVE - NORMALIZED

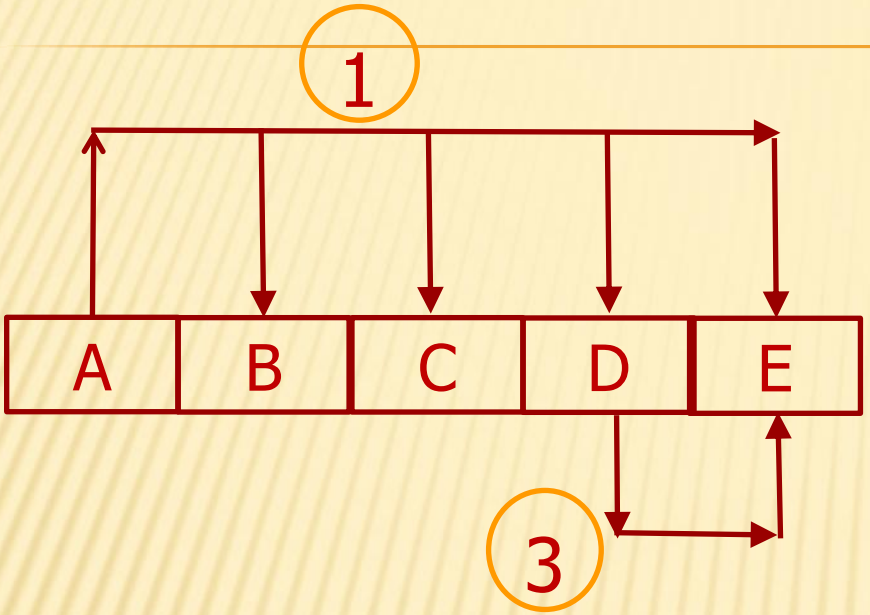


2NF

The central part of 3NF



PARTIAL VS TRANSITIVE DEPENDENCY



There is ALWAYS a _____ dependency; There may or may not be _____ or _____ dependencies.

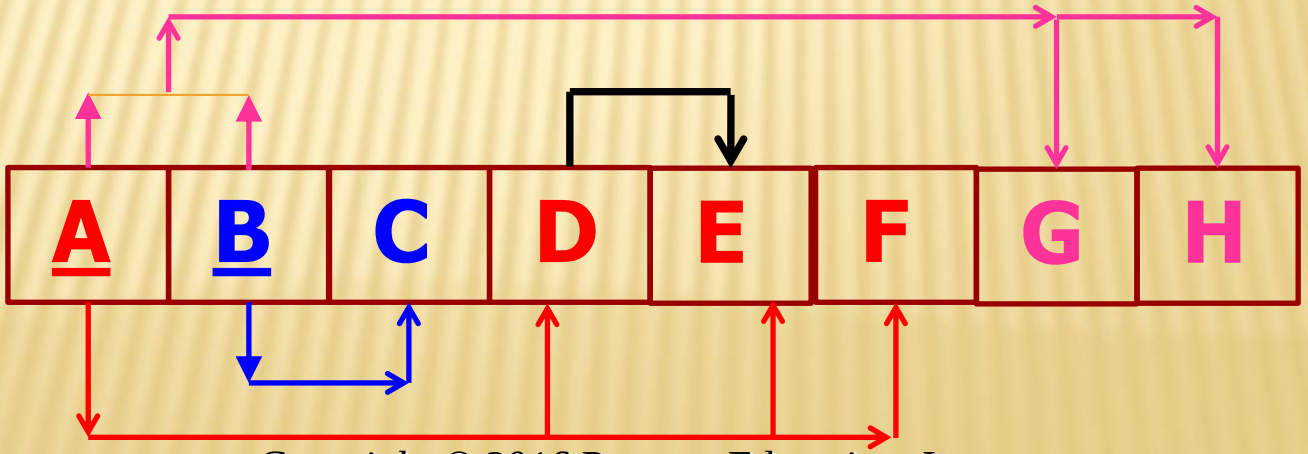
1: _____ dependency?

3: _____ dependency?

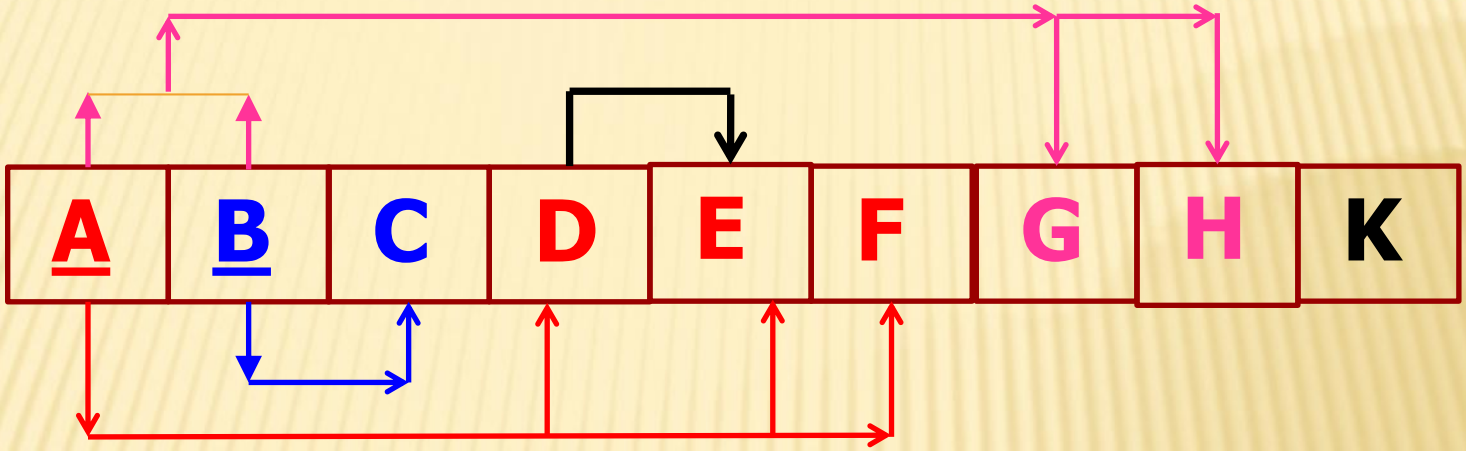
Q: Will we have the issue of partial dependency in this case? ➔ conclusion re where would each exist

RECAP: IDENTIFY PD AND TD

- 2NF: No partial dependency
  - Implies: composite key
  - Dependencies with the determinant being PART of the composite key ⊘
- 3NF: No transitive dependency
  - Dependencies with the determinant being non-key ⊘



ANATOMY OF FUNCTIONAL DEPENDENCIES (1)

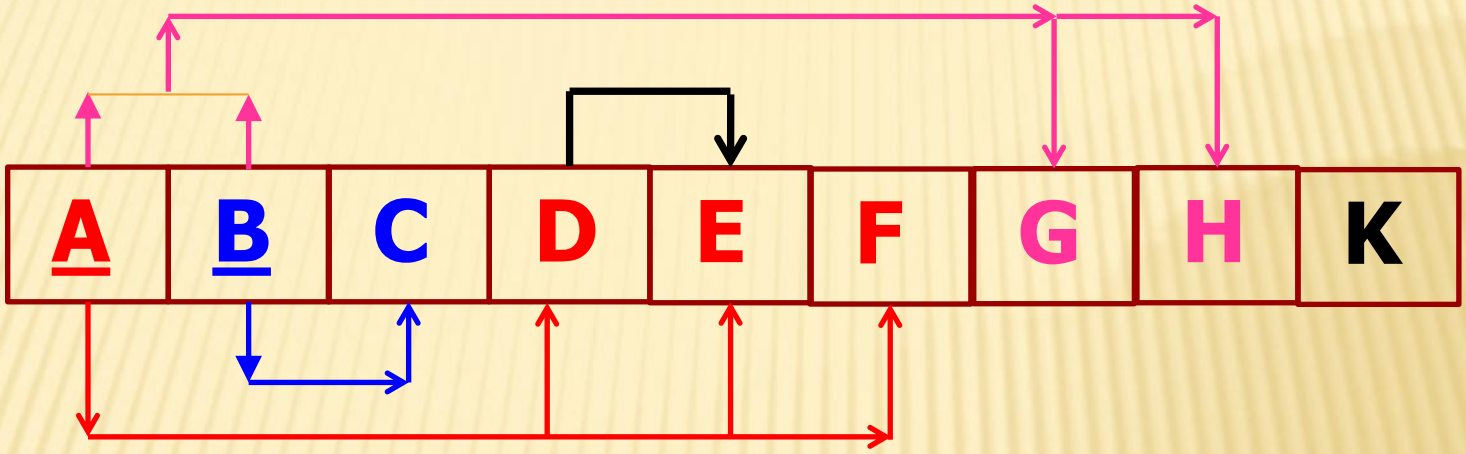


- [A+B] ➔ G, H - ?
- B ➔ C - ?
- A ➔ D, E, F - ?
- D ➔ E - ?

Types?

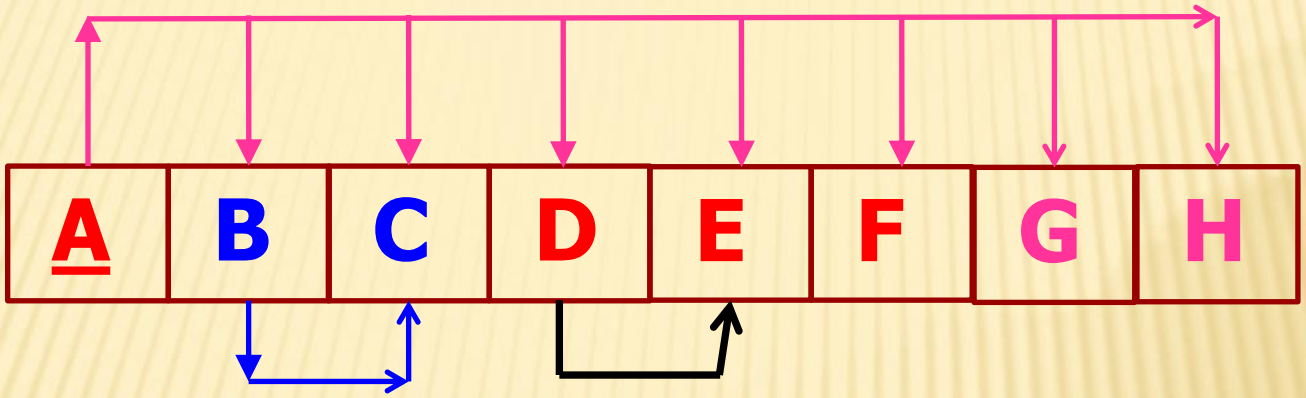
- What can we say about K???

ANATOMY OF FUNCTIONAL DEPENDENCIES (2)



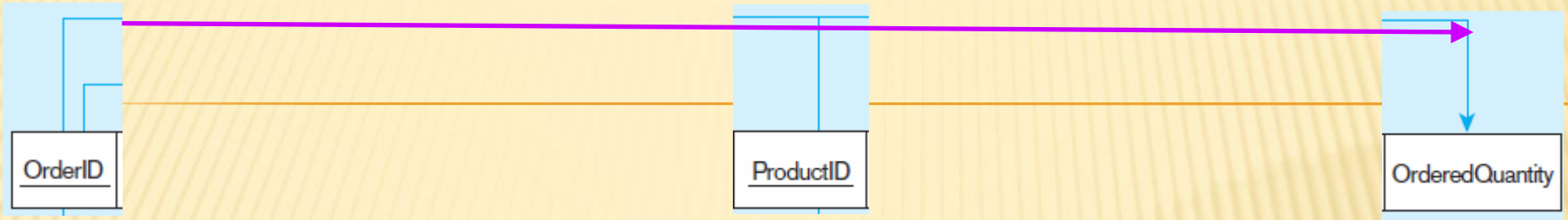
- A+B: 2 fields jointly have 1 arrow out
- A: 2 arrows out
- B: 2 arrows out
- D: 1 arrow in, 1 arrow out
- E: 2 arrows in

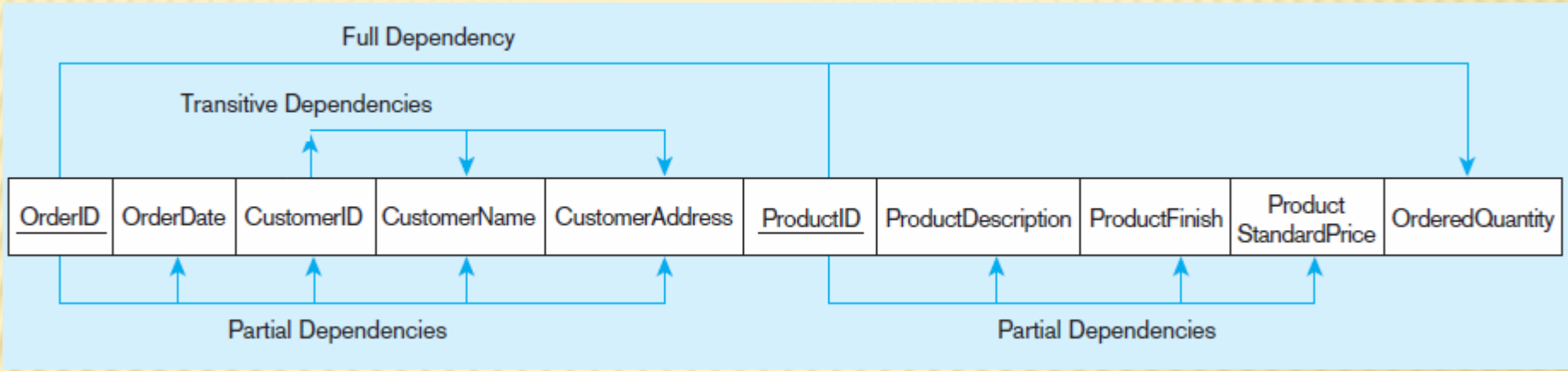
ANATOMY OF FUNCTIONAL DEPENDENCIES (3)

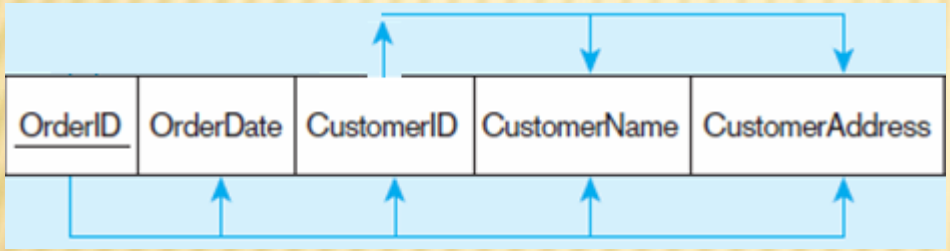


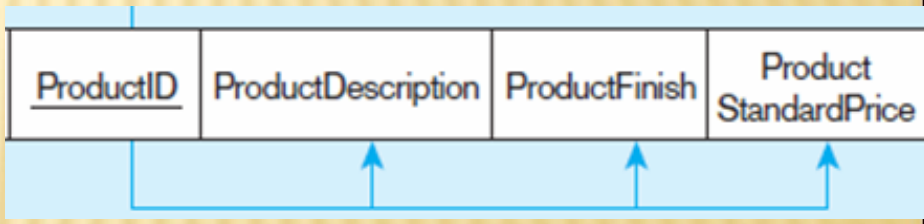
- A: 1 arrow out
- B: 1 arrow in, 1 arrow out
- D: 1 arrow in, 1 arrow out
- C: 2 arrows in
- E: 2 arrows in

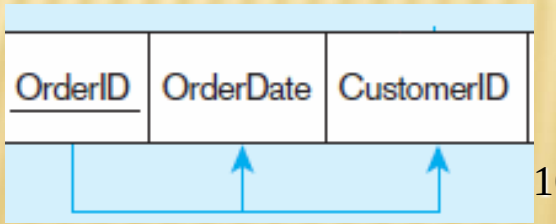
Do we have PD here? Why?

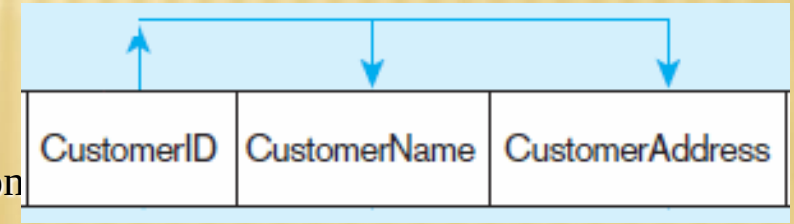

OrderID
ProductID
OrderedQuantity


Full Dependency
Transitive Dependencies
OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress
ProductID
ProductDescription
ProductFinish
Product StandardPrice
OrderedQuantity
Partial Dependencies
Partial Dependencies


OrderID
OrderDate
CustomerID
CustomerName
CustomerAddress


ProductID
ProductDescription
ProductFinish
Product StandardPrice


OrderID
OrderDate
CustomerID

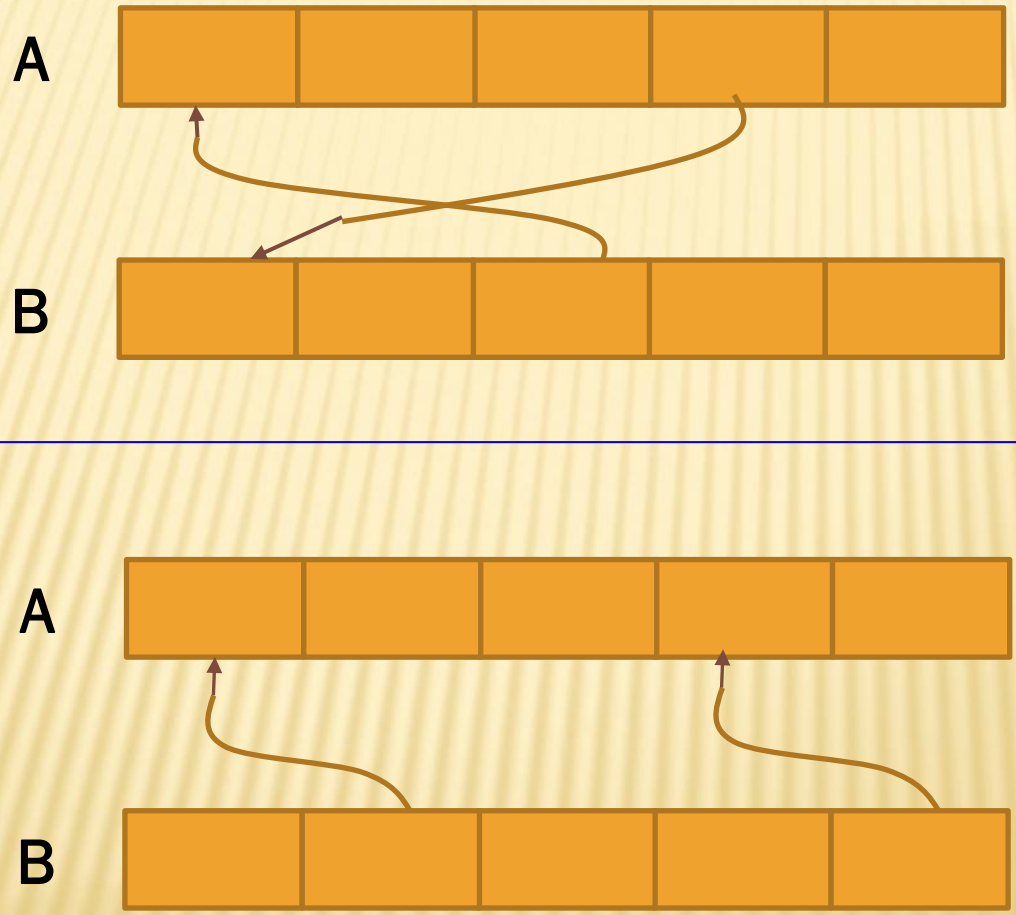

CustomerID
CustomerName
CustomerAddress

RECAP REF INTEGRITY VS FUNC DEP: THIS ARROW IS NOT "THAT ARROW"

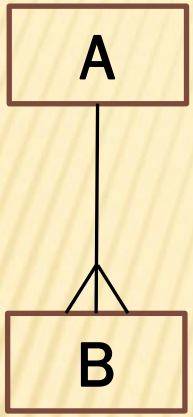
1. The arrows we're discussing here are about _____ _____
2. In **Part I** of the chapter we also had curvy arrows; those were about _____ _____

- In "1" above, arrow starts from _____ and ends at _____ (in a _____ _____)
- In "2" above, arrow starts from _____ and ends at _____ (in a _____ _____)
- -- so, don't confuse the two!

SEVERE ERRORS IN NORMALIZATION (1)



COMMON ERRORS IN NORMALIZATION (2)



A

A1	A2	A3	A4	B1

B

B1	B2	B3	B4	B5

W/o arrows, can you tell the mistakes?

A

A1	A2	A3	A4

B

B1	B2	B3	A1	A4

COMMON ERRORS IN NORMALIZATION (3) (4)

- Common error #3:
  - The number of functional dependencies in 1NF does not match the number of relations in 3NF
  - Example: In 1NF, 5 func. deps were identified; after normalization, in 3NF there were only 4 relations (tables)
- Common error #4:
  - The fields in the relations in 3NF are not organized in the same relations (tables) as the fields were in their corresponding func. deps in 1NF
  - Example: Say in the original 1NF, the fields were identified to belong to several func deps that they were grouped as: A,C,D; B,E,F; A,B,G,H
    - But in the final 3NFs, the relations contain fields as: A,C,D,H; B,E,F; A,B,G

MERGING RELATIONS

- View Integration–Combining entities from multiple ER models into common relations
- Issues to watch out for when merging entities from different ER models:
  - Synonyms–two or more attributes with different names but same meaning
  - Homonyms–attributes with same name but different meanings
  - Transitive dependencies–even if relations are in 3NF prior to merging, they may not be after merging
  - Supertype/subtype relationships–may be hidden prior to merging

ENTERPRISE KEYS

- Primary keys that are unique in the whole database, not just within a single relation
- Corresponds with the concept of an object ID in object-oriented systems

Figure 4-31 Enterprise keys

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, EmpName, DeptName, Salary)
CUSTOMER (OID, CustID, CustName, Address)

a) Relations with enterprise key

b) Sample data with enterprise key

OBJECT

OID	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

EMPLOYEE

OID	EmpID	EmpName	DeptName	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

OID	CustID	CustName	Address
2	100	Fred's Warehouse	Greensboro, NC
3	101	Bargain Bonanza	Moscow, ID
6	102	Jasper's	Tallahassee, FL
7	103	Desks 'R Us	Kettering, OH

TO TELL A TD FROM A PD; FD ALWAYS EXISTS

Watch the field where the arrow originates:

- If the arrow goes out from a NON-key field, this func dep is TD;
  - "From non-key to non-key"
- If the arrow goes out from PART of a composite key, this func dep is PD (Partial);
  - "From PART of a key to non-key"
  - PD ONLY exists when/where there is a composite key!
- There ALWAYS exists a Full Dependency
  - "From key to non-key"