

# Chapter 5

## Normalization

## Database Design - Recap

- When we design a database, how do we know that the design is “correct”, i.e. it would not create problems when processing the database?
- Is there a way to verify the correctness of the design?

# Limitations of E-R Designs

## ER Design

- Provides a set of guidelines, does not result in a unique database schema
- Does not provide a way of evaluating alternative schemas

Normalization theory provides a mechanism for analyzing and refining the schema produced by an E-R design

# Normalization Theory

- Result of E-R analysis need further refinement
- Appropriate decomposition can solve problems
- The underlying theory is referred to as *normalization theory* and is based on *functional dependencies* (and other kinds, like *multivalued dependencies*)

# Database Tables and Normalization

## Normalization

- Process for evaluating and correcting table structures to minimize data redundancies
- Usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them
  - helps eliminate data anomalies
    - Anomaly : something that deviates from what is standard, normal, or expected
    - i.e. when you update the data in the database, you EXPECT to get the correct result but SOMETIMES you DON'T
    - We should not have a database that gives INCORRECT result even if it only happens SOMETIMES – integrity problem
- Works through a series of stages called normal forms:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)

## Database Tables and Normalization (continued)

- 2NF is better than 1NF; 3NF is better than 2NF
- For most business database design purposes, 3NF is highest we need to go in the normalization process
- The other normal forms are Boyce-Codd NF, 4 NF, 5 NF and Domain Key NF

# Normalization and Database Design (continued)

In the last chapter we use ERD to design database ERD

- Provides the big picture, or macro view, of an organization's data requirements and operations
  - Top-down approach to database design
- Created through an iterative process
  - Identifying relevant entities, their attributes and their relationship
  - Use results to identify additional entities and attributes

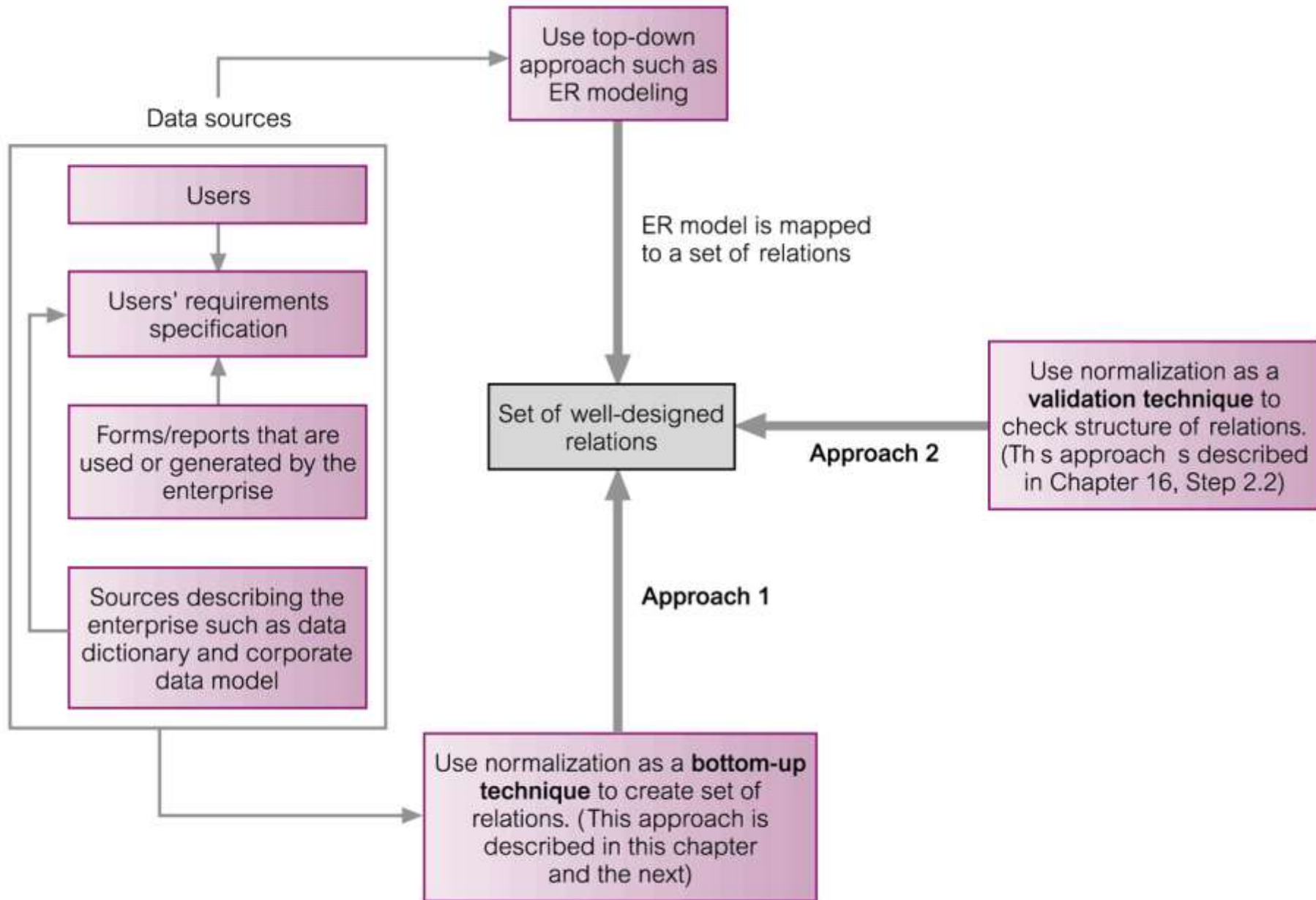
# Normalization and Database Design (continued)

## Normalization procedures

- Focus on the characteristics of specific entities
- A micro view of the entities within the ER diagram
  - Bottom-up approach to database design
- Difficult to separate normalization process from ER modeling process
- Two techniques should be used concurrently



# How Normalization Supports Database Design



**Figure 13.1** How normalization can be used to support database design.

# Unnormalized Sample Data

## Case Scenario:

- A cyber café wants to capture data about its members
- A log book is used to record down the following:
  - Entry No, Member email, password, name, phone, IN time and OUT time

MEMBERVISIT							
id	email	password	fname	lname	phone	date_time_in	date_time_out
001	dayj@ohio.edu	rocket	John	Day	592-0646	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	bullet	Thom	Luce	592-1111	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	rocket	John	Day	592-0646	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	bullet	Thom	Luce	592-1111	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	arrow	Sean	McGann	592-2222	27-JUN-02 16:00	27-JUN-02 18:00

Entry no

An example of a logbook entry

\*note the duplicate entries

# Unnormalized Design Causes Update Problems

MEMBERVISIT							
id	email	password	fname	lname	phone	date_time_in	date_time_out
001	dayj@ohio.edu	rocket	John	Day	592-0646	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	bullet	Thom	Luce	592-1111	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	rocket	John	Day	592-0646	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	cannon	Thom	Luce	592-1111	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	arrow	Sean	McGann	592-2222	27-JUN-02 16:00	27-JUN-02 18:00

When the password for Thom Luce was changed, it was not changed in both his entries.

Which password is the correct password?

# Normalized Design Eliminates Update Problems

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

VISIT			
<u>id</u>	MEMBER\$email	date_time_in	date_time_out
001	dayj@ohio.edu	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	27-JUN-02 16:00	27-JUN-02 18:00

If the database was normalized properly, there would only be one update done to the password, and the password would only appear in one place.

# Unnormalized Design Creates Insert Problems

MEMBERVISIT							
<u>visit_id</u>	email	password	fname	lname	phone	date_time_in	date_time_out
001	dayj@ohio.edu	rocket	John	Day	592-0646	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	bullet	Thom	Luce	592-1111	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	rocket	John	Day	592-0646	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	bullet	Thom	Luce	592-1111	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	arrow	Sean	McGann	592-2222	27-JUN-02 16:00	27-JUN-02 18:00
	frostr@ohio.edu	turtle	Raymond	Frost	597-2902		

A new member cannot be created unless they have already made a visit, otherwise there would be no primary key value.  
This field may not be left blank.

# Normalized Design Eliminates the Insert Problem

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111
mcgann@ohio.edu	arrow	Sean	McGann	592-2222
frostr@ohio.edu	turtle	Raymond	Frost	597-2902

VISIT			
<u>id</u>	MEMBER\$email	date_time_in	date_time_out
001	dayj@ohio.edu	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	27-JUN-02 16:00	27-JUN-02 18:00

In a normalized database , the addition of Raymond Frost as a new member is separate from the logging of his visits.



# Unnormalized Design Creates Delete Problems

MEMBERVISIT								
visit_id	email	password	fname	lname	phone	date_time_in		date_time_out
001	dayj@ohio.edu	rocket	John	Day	592-0646	25-JUN-02	14:00	25-JUN-02 17:30
002	luce@ohio.edu	bullet	Thom	Luce	592-1111	25-JUN-02	18:00	25-JUN-02 20:00
003	dayj@ohio.edu	rocket	John	Day	592-0646	26-JUN-02	10:00	26-JUN-02 11:30
004	luce@ohio.edu	bullet	Thom	Luce	592-1111	27-JUN-02	09:00	27-JUN-02 10:00
<del>005</del>	<del>megann@ohio.edu</del>	<del>arrow</del>	<del>Sean</del>	<del>McGann</del>	<del>592 2222</del>	<del>27 JUN 02</del>	<del>16:00</del>	<del>27 JUN 02 18:00</del>

If a member makes only one visit, deleting that record will cause the loss of the member data

For example, there is only one record to indicate Sean McGann's member data.

If we had to delete record 005 (lets say we got the date wrong), then we would lose all of Sean's data.

# Normalized Design Eliminates the Delete Problem

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

VISIT			
<u>id</u>	MEMBER\$email	date_time_in	date_time_out
001	dayj@ohio.edu	25-JUN-02 14:00	25-JUN-02 17:30
002	luce@ohio.edu	25-JUN-02 18:00	25-JUN-02 20:00
003	dayj@ohio.edu	26-JUN-02 10:00	26-JUN-02 11:30
004	luce@ohio.edu	27-JUN-02 09:00	27-JUN-02 10:00
005	mcgann@ohio.edu	27-JUN-02 16:00	27-JUN-02 18:00

Now, deleting visit 005 would not cause the loss of Sean McGann's member data because his visitation data is separate from his member data.



# First Normal Form (1NF)

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111, 593-0212
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

1NF: A table in which all fields contain a single value.

Lets assume that a member can have more than one phone number. The example above would be a violation of the 1NF rule; therefore this table would have to be redesigned.

# Fixing Normalization Violations

## Step 1: Tables

- *Create new table(s)*
- *Rename original table if necessary*

## Step 2: Relationships

- *Establish relationships between original and new table(s)*

## Step 3: Fields

- *Transfer fields and rename as needed*

## Step 4: Keys

- *Choose PK and FK for all tables*  
(PK and FK keys will be discussed in more details in a few slides later)

# Solving a 1NF Violation

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111, 593-0212
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

Step 1:

Tables

- Since the phone number column violated the 1NF rule, make a new table to hold phone numbers: DIRECTORY.

MEMBER
<u>email</u>
password
fname
lname
phone

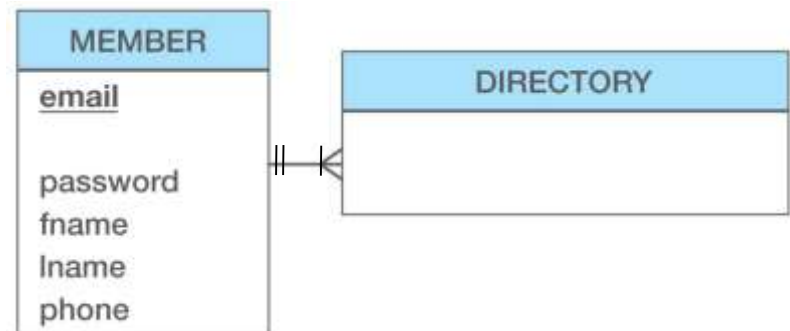
DIRECTORY

# Solving a 1NF Violation

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111, 593-0212
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

## Step 2: Relationships

- A member has multiple phone numbers and a phone number belongs to one member

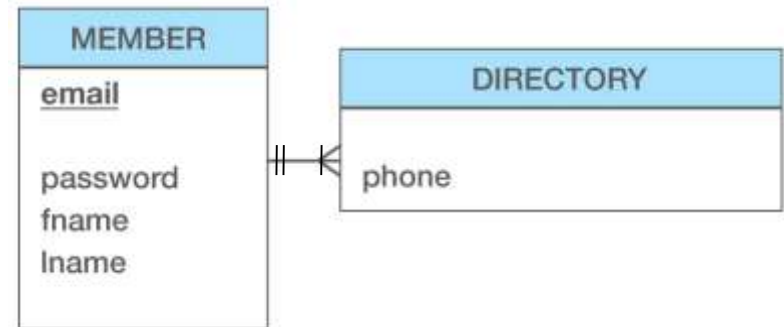


# Solving a 1NF Violation

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111, 593-0212
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

## Step 3: Fields

- The phone field is transferred to the *DIRECTORY* table



# Solving a 1NF Violation

MEMBER				
<u>email</u>	password	fname	lname	phone
dayj@ohio.edu	rocket	John	Day	592-0646
luce@ohio.edu	cannon	Thom	Luce	592-1111, 593-0212
mcgann@ohio.edu	arrow	Sean	McGann	592-2222

## Step 4: Keys

(keys will be discussed in more details in a few slides later)

- The email column in MEMBER becomes a FK (MEMBER\$email) in DIRECTORY
- The PK in DIRECTORY becomes MEMBER\$email and phone since two members could have the same phone number (e.g. two members from the same household).

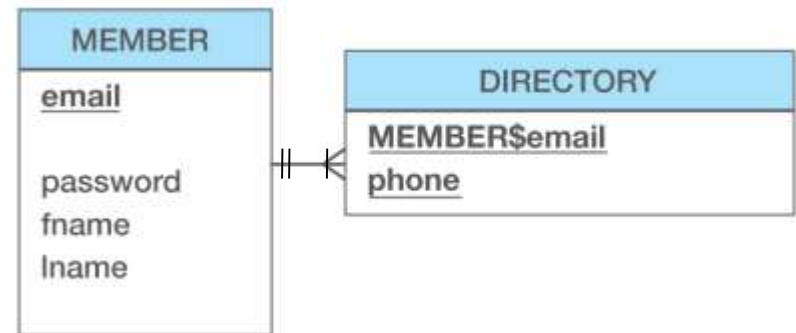


Exhibit 4-10: Arcade Database Solving the 1NF Violation

# Tables in 1NF Eliminate Repeating Data Problems

MEMBER			
<u>email</u>	password	fname	lname
dayj@ohio.edu	rocket	John	Day
luce@ohio.edu	cannon	Thom	Luce
mcgann@ohio.edu	arrow	Sean	McGann

PHONE_LIST	
<u>MEMBER\$email</u>	<u>phone</u>
dayj@ohio.edu	592-0646
luce@ohio.edu	592-1111
luce@ohio.edu	593-0212
mcgann@ohio.edu	592-2222

Now all tables have fields that contain only single values.

# Determinants

- An important concept in Normalization is something called a **determinant**.
- Determinant: a field or group of fields that controls or determines the values in another field.
- From the previous example, the value of email will determine the values in all the other fields.
- That is, if you know someone's email, you can determine the rest of their information.
  - Stated another way, given an email, you will be able to find/retrieve the **[EXACTLY ONE]** member's name, phone, etc.
  - A determinant **MUST return only ONE value**
    - *Given the StudentID 'ABC123456', how many [Name, DOB, IC-No] will you retrieve?*
    - *Given the Name 'Lim Ah Kau', how many [StudentID, DOB, IC-No] will you retrieve?*
    - *Which is the determinant : StudentID or Name ?*



# Functional Dependency

Determinants are based on the concepts of **functional dependency**

- A **functional dependency** occurs when the value of one (set of) attribute(s) determines (**arrow symbol**) the value of a second (set of) attribute(s):

**StudentID**  $\rightarrow$  **StudentName** (i.e. One studentID will return one name)

**StudentID**  $\rightarrow$  (**DormName, DormRoom, Fee**)

(i.e. One studentID will return one dorm name, one dorm room no and one fee)

- The attribute on the left side of the functional dependency is called the **determinant**.
- Functional dependencies may be based on equations:  
**ExtendedPrice = Quantity X UnitPrice**  
**(Quantity, UnitPrice)**  $\rightarrow$  **ExtendedPrice**
- Functional dependencies are not equations!

# Functional Dependencies Are Not Equations

Object Color	Weight	Shape
Red	5	Ball
Blue	5	Cube
Yellow	7	Cube

**ObjectColor  $\rightarrow$  Weight**

**ObjectColor  $\rightarrow$  Shape**

**ObjectColor  $\rightarrow$  (Weight, Shape)**

# Composite Determinants

- Composite determinant** = a determinant of a functional dependency that consists of more than one attribute

**(StudentID, CourseCode) → (Grade)**

StudentName	CourseCode	Grade
ABC1234	AACS3013	A
ABC1234	AACS2373	A
ABC1234	AACS1234	B
ABC4567	AACS3013	B+
ABC4567	AACS2373	A-
ABC4567	AACS1234	A

Not unique

Each combination is unique

## Functional Dependency Rules

- If  $A \twoheadrightarrow (B, C)$ , then  $A \twoheadrightarrow B$  and  $A \twoheadrightarrow C$ .
- If  $(A, B) \twoheadrightarrow C$ , then neither  $A$  *nor*  $B$  determines  $C$  by itself.

**ObjectColor  $\twoheadrightarrow$  (Weight, Shape)**

**ObjectColor  $\twoheadrightarrow$  Weight**

**ObjectColor  $\twoheadrightarrow$  Shape**

**(StudentID, CourseCode)  $\twoheadrightarrow$  (Grade)**

**(StudentID)  $\twoheadrightarrow$  (Grade) NOT TRUE**

**(CourseCode)  $\twoheadrightarrow$  (Grade) NOT TRUE**

# What Makes Determinant Values Unique?

- A determinant is unique in a relation if and only if, it determines every other column in the relation.
- You cannot find the determinants of all functional dependencies simply by looking for unique values in **ONE** column:
  - Data set limitations
    - Combination of columns may create a unique determinant
  - Must be logically a determinant
    - Make sense; according to business rules

## Relational Database and Keys

- In relational database design, the concept of KEYS are very important
- A **key** is a combination of one or more columns that is used to identify rows in a relation.
  - *A given key value will only identify one row of data*
- A **composite key** is a key that consists of two or more columns.
- A Key is a determinant

## Candidate and Primary Keys

- A **candidate key** is a key that determines all of the other columns in a relation.
  - There can be more than one candidate key
    - E.g. StudentID, C\_No, Driving\_License\_No as a row in a table
- A **primary key** is a candidate key selected as the primary means of identifying rows in a relation.
  - **There is only ONE primary key per relation.**
    - E.g. StudentID selected as the primary key
  - The primary key may be a composite key.
  - The ideal primary key is short, numeric, and never changes.

# Surrogate Keys

- A **surrogate key** is an artificial column added to a relation to serve as a primary key.
  - DBMS supplied
  - Short, numeric, and never changes—an ideal primary key
  - Has artificial values that are meaningless to users
    - Just a number to distinguish one record from another
    - Record '1001' compared to record '1100'
  - Normally hidden in forms and reports
- Reasons for using (refer additional pdf reading material)
  - Sometimes the combination of natural keys may be too large to use efficiently
  - E.g. Lets say a STAFF entity has a composite PK consisting of:
    - (Name, Address, DOB, Phone) ? Salary
    - May be more efficient to just create another attribute call Staff\_No (DBMS will auto generate and maintain a running number) to identify a staff record



## Foreign Keys

- A **foreign key** is the primary key of one relation that is placed in another relation to form a link between the relations.
  - A foreign key can be a single column or a composite key.
  - The term refers to the fact that key values are *foreign* to the relation in which they appear as foreign key values.

Example

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)

EMPLOYEE (EmployeeNumber, EmployeeLastName, EmployeeFirstName,  
DepartmentName\*)

The values of the DepartmentName in the EMPLOYEE table comes from the primary key of the DEPARTMENT table.

# The Referential Integrity Constraint

- A **referential integrity constraint** is a statement that limits the values of the foreign key to those already existing as primary key values in the corresponding relation.
- E.g.


```
CREATE TABLE TITLEAUTHOR (  
    au_id      VARCHAR(11) NOT NULL,  
    title_id   VARCHAR(6)  NOT NULL,  
    au_ord     NUMBER(1),  
    royaltyp   NUMBER(3),  
    Primary key (au_id,title_id),  
    Foreign Key (au_id) References AUTHORS(au_id),  
    Foreign Key (title_id) References TITLES(title_id)  
);
```

The values of au\_id and title\_id in the TITLEAUTHOR table must be from

# Determinants and Duplicate Data

When you have duplicate data, knowing someone's email may not allow you to determine the rest of the data as shown below.

*Exhibit 4-13: Email Acts as a Determinant*



MEMBER						
<u>email</u>	fname	lname	phone	jumps	equip	level
dayj@ohio.edu	John	Day	529-0646	5	Y	B
luce@ohio.edu	Thom	Luce	529-1111	12	N	I
luce@ohio.edu	Thom	Luce	529-1111	12	N	I
mcgann@ohio.edu	Sean	McGann	592-2222	20	Y	A

In this example, the duplicated email [luce@ohio.edu](mailto:luce@ohio.edu) will still return only one value for fname, lname, phone, etc

MEMBER						
<u>email</u>	fname	lname	phone	jumps	equip	level
dayj@ohio.edu	John	Day	529-0646	5	Y	B
luce@ohio.edu	Thom	Luce	529-1111	12	N	I
luce@ohio.edu	Thomas	Luce	597-8822	12	N	I
mcgann@ohio.edu	Sean	McGann	592-2222	20	Y	A

*Exhibit 4-14: Email Fails to Act as a Determinant*

In this example, the duplicated email [luce@ohio.edu](mailto:luce@ohio.edu) will return two different values for phone, therefore email is not a determinant

# Another Example

Order Log

Customer No	Name	Tel No	Order No	Order Date	Item No	Description	Order Qty	Unit Price
A101	Ali bin Ahmad	1234567	C1001	12/5/14	K123	Blue Pen	5	2.50
					M345	Stainless steel ruler	3	7.80
					P876	Mars Bars	20	1.25
A102	Lim Ah Kau	1122448	C1002	13/5/14	K124	Black Pen	12	2.50
					K325	Marker Pen	24	1.90
A101	Ali bin Ahmad	1234567	C1003	15/5/14	K123	Blue Pen	10	2.50
					M345	Stainless steel ruler	6	7.80
					P876	Mars Bars	25	1.25

An example of an Order Log, where each customer can make orders.  
Each order can have many items.  
An item can appear in more than one order.

Order Log is not in 1<sup>st</sup> NF because some of its columns contains **multiple values**: i.e. multiple values for ItemNo, Description, OrderQty and UnitPrice for each Order

# Normalization Process - First Normal Form (1NF)

1NF rules:

- No multivalued attributes
- Every attribute value is atomic – single value only

Transform the OrderLog to 1NF

Step 1: Tables

- *Create new table(s)*
- *Rename original table if necessary*

Make a new table to hold all the column  
with multiple values:

Two tables required:

- **Orders** table to hold data about orders  
by each customer
- **OrderItems** table to hold data about  
items in each order

Orders
CustomerNo
Name
TelNo
OrderNo
OrderDate
ItemNo
Description
OrderQty
UnitPrice

OrderItems

# Normalization Process - First Normal Form (1NF)

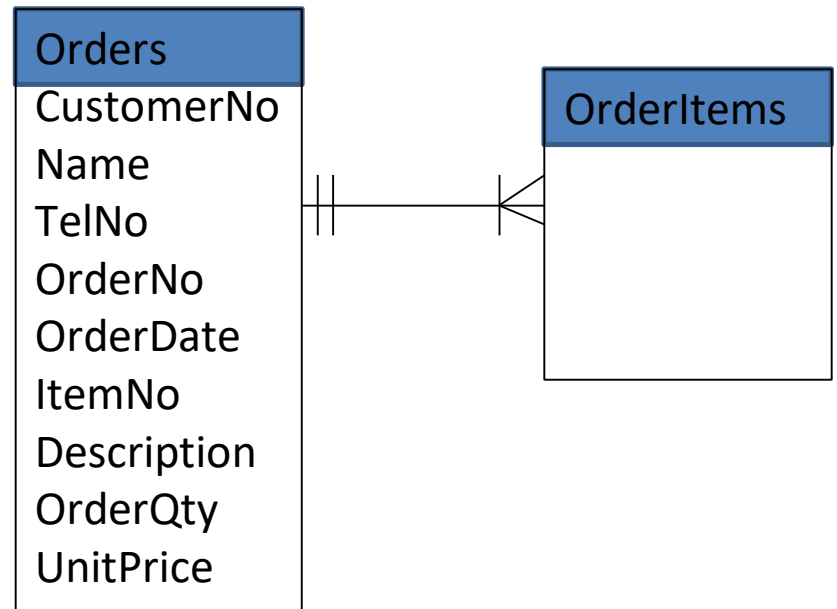
Transform the OrderLog to 1NF

Step 1: Tables

- *Create new table(s)*
- *Rename original table if necessary*

Step 2: Relationships

- *Establish relationships between original and new table(s)*
  - *An order can have multiple items*
  - *Each item in the OrderItems table belongs to one Order*



Step 3: Fields

- *Transfer fields and rename as needed*

Step 4: Keys

- *Choose PK and FK for all tables*

# Normalization Process - First Normal Form (1NF)

## Transform the OrderLog to 1NF

### Step 1: Tables

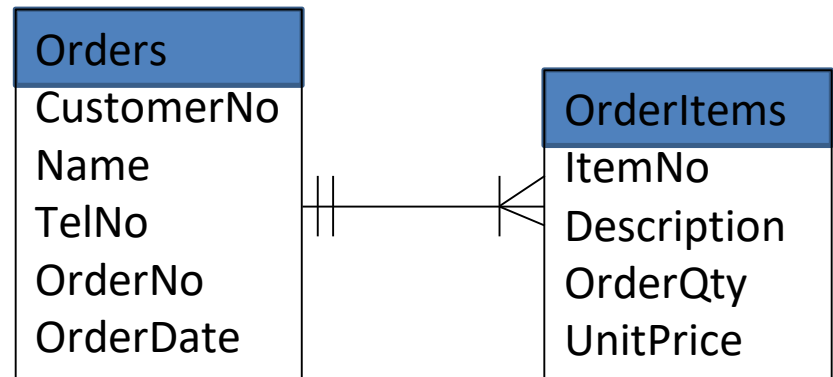
- *Create new table(s)*
- *Rename original table if necessary*

### Step 2: Relationships

- *Establish relationships between original and new table(s)*

### Step 3: Fields

- *Transfer fields and rename as needed*
  - *All the fields with multiple values transferred to the OrderItems table*



### Step 4: Keys

- *Choose PK and FK for all tables*

# Normalization Process - First Normal Form (1NF)

## Transform the OrderLog to 1NF

### Step 1: Tables

- *Create new table(s)*
- *Rename original table if necessary*

### Step 2: Relationships

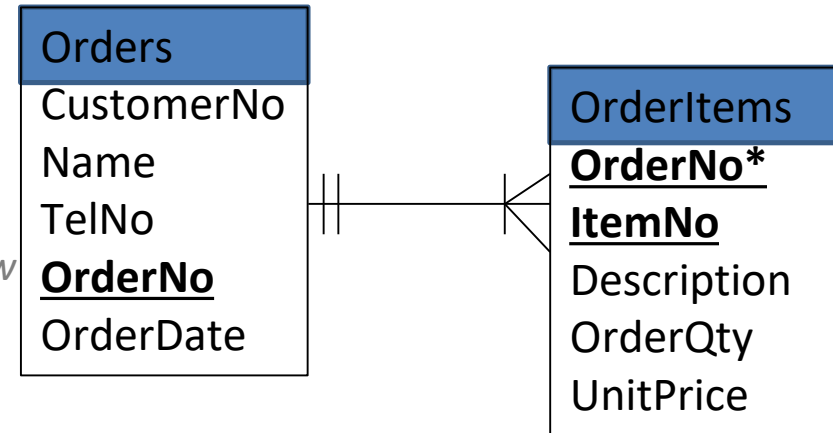
- *Establish relationships between original and new*

### Step 3: Fields

- *Transfer fields and rename as needed*

### Step 4: Keys

- *Choose PK and FK for all tables*



#### Orders table

OrderNo ? CustomerNo, Name, TelNo  
OrderDate

i.e. Given an OrderNo (e.g. C1001), you will  
be able to retrieve a row of related

data

#### OrderItems table

The **PK (OrderNo)** from the **ORDERS** table becomes a **FK** in the ORDERITEMS table to  
maintain a relationship between the two tables.

(OrderNo,ItemNo) ? Description, OrderQty, UnitPrice

The combination of OrderNo and ItemNo will return a row of related data



# Normalization Process - First Normal Form (1NF)

## Step 4: Keys

- *Choose PK and FK for all tables*

Orders table
OrderNo ? CustomerNo, Name, TelNo OrderDate
i.e. Given an OrderNo (e.g. C1001), you will be able to retrieve a row of related data

Order No	Customer No	Name	Tel No	Order Date
C1001	A101	Ali bin Ahmad	1234567	12/5/14
C1002	A102	Lim Ah Kau	1122448	13/5/14
C1003	A101	Ali bin Ahmad	1234567	15/5/14

OrderItems table
(OrderNo,ItemNo) ? Description, OrderQty, UnitPrice
The combination of OrderNo and ItemNo will return a row of related data

Order No	Item No	Description	Order Qty	Unit Price
C1001	K123	Blue Pen	5	2.50
C1001	M345	Stainless steel ruler	3	7.80
C1001	P876	Mars Bars	20	1.25
C1002	K124	Black Pen	12	2.50
C1002	K325	Marker Pen	24	1.90
C1003	K123	Blue Pen	10	2.50
C1003	M345	Stainless steel ruler	6	7.80
C1003	P876	Mars Bars	25	1.25

# Normalization Process - First Normal Form (1NF)

At the end of the 1NF transformation, we now have two relations:

Orders (OrderNo, CustomerNo, Name, TelNo, OrderDate)

OrderItems(OrderNo\*, ItemNo, Description, OrderQty, UnitPrice)

**Orders**

<u>Order No</u>	Customer No	Name	Tel No	Order Date
C1001	A101	Ali bin Ahmad	1234567	12/5/14
C1002	A102	Lim Ah Kau	1122448	13/5/14
C1003	A101	Ali bin Ahmad	1234567	15/5/14

**OrderItems**

<u>Order No</u>	<u>Item No</u>	Description	Order Qty	Unit Price
C1001	K123	Blue Pen	5	2.50
C1001	M345	Stainless steel ruler	3	7.80
C1001	P876	Mars Bars	20	1.25
C1002	K124	Black Pen	12	2.50
C1002	K325	Marker Pen	24	1.90
C1003	K123	Blue Pen	10	2.50
C1003	M345	Stainless steel ruler	6	7.80
C1003	P876	Mars Bars	25	1.25

Both the tables fulfill the 1NF requirements.

- The table has a key
- In every row, every column has only a single value

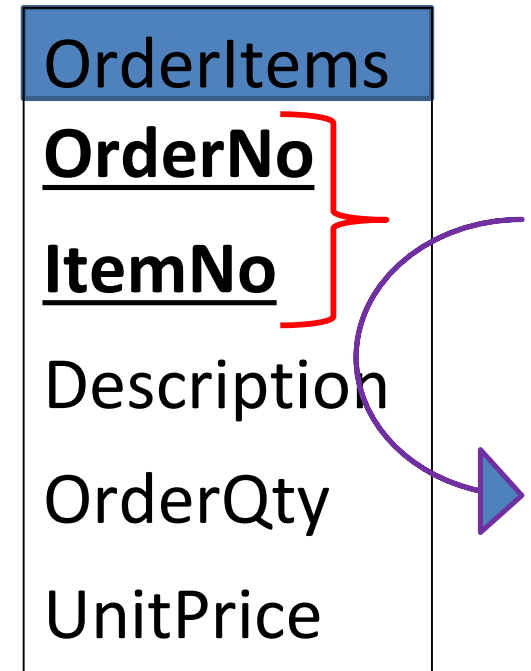
# Second Normal Form (2NF) transformation

2NF: A table in which each non-key field is determined by the **whole primary key** and **NOT part of the primary key** by itself (no partial dependency)

In the OrderItems relation, the composite key (OrderNo, ItemNo) will determine the OrderQty

**OrderItems**

Order No	Item No	Description	Order Qty	Unit Price
C1001	K123	Blue Pen	5	2.50
C1001	M345	Stainless steel ruler	3	7.80
C1001	P876	Mars Bars	20	1.25
C1002	K124	Black Pen	12	2.50
C1002	K325	Marker Pen	24	1.90
C1003	K123	Blue Pen	10	2.50
C1003	M345	Stainless steel ruler	6	7.80
C1003	P876	Mars Bars	25	1.25



*Exhibit 4-15: 2NF Violation*

# Second Normal Form (2NF) transformation

If you know an **ItemNo**, you can determine its **Description** and **UnitPrice**.

Therefore, the **Description** and **UnitPrice** non-key fields are determined by just part of the primary key: **ItemNo**.

Description, UnitPrice are **PARTIALLY dependent** on (OrderNo,ItemNo)

Therefore, **OrderItems** is **ONLY in 1NF** not 2NF

**OrderItems**

Order No	Item No	Description	Order Qty	Unit Price
C1001	K123	Blue Pen	5	2.50
C1001	M345	Stainless steel ruler	3	7.80
C1001	P876	Mars Bars	20	1.25
C1002	K124	Black Pen	12	2.50
C1002	K325	Marker Pen	24	1.90
C1003	K123	Blue Pen	10	2.50
C1003	M345	Stainless steel ruler	6	7.80
C1003	P876	Mars Bars	25	1.25

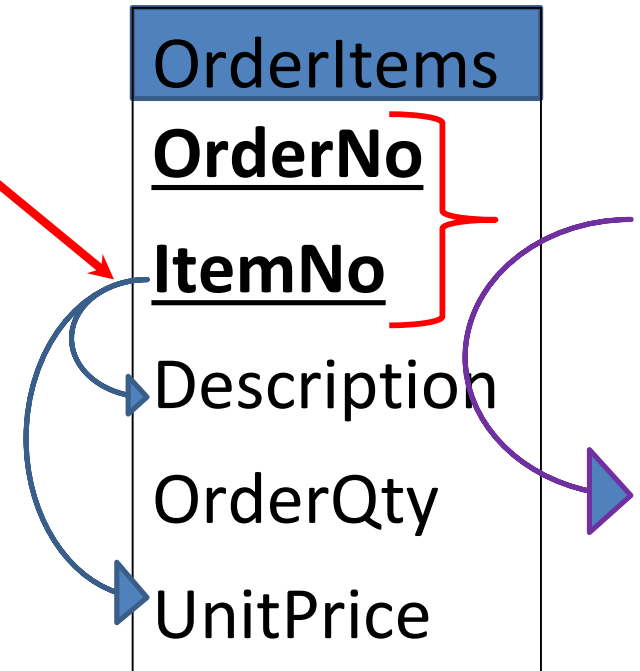


Exhibit 4-15: 2NF Violation

# Update Problem Caused by 2NF Violation

Order No	Item No	Description	Order Qty	Unit Price
C1001	K123	Blue Felt Pen	5	2.50
C1001	M345	Stainless steel ruler	3	7.80
C1001	P876	Mars Bars	20	1.25
C1002	K124	Black Pen	12	2.50
C1002	K325	Marker Pen	24	1.90
C1003	K123	Blue Pen	10	2.50
C1003	M345	Stainless steel ruler	6	7.80
C1003	P876	Mars Bars	25	1.25



The above table is ONLY in 1NF, it violates the 2NF rules. Data not determined by the whole primary key will be duplicated and any updates may not be made to all instances of duplicate data. (Update Anomaly)

In this example, we no longer know the correct description for ItemNo 'K123'. Do you think a relation in 1NF have Insertion Anomaly? Deletion Anomaly?

# Solving a 2NF Violation

## Step 1: Tables

- *Since only OrderQty belongs in the ORDERITEMS table, create a new table (ITEMS) for the item information.*

## Step 2: Relationships

OrderItems
OrderNo
ItemNo
ItemNo
Description
OrderQty
UnitPrice

Items

## Step 3: Fields

## Step 4: Keys

# Solving a 2NF Violation

## Step 1: Tables

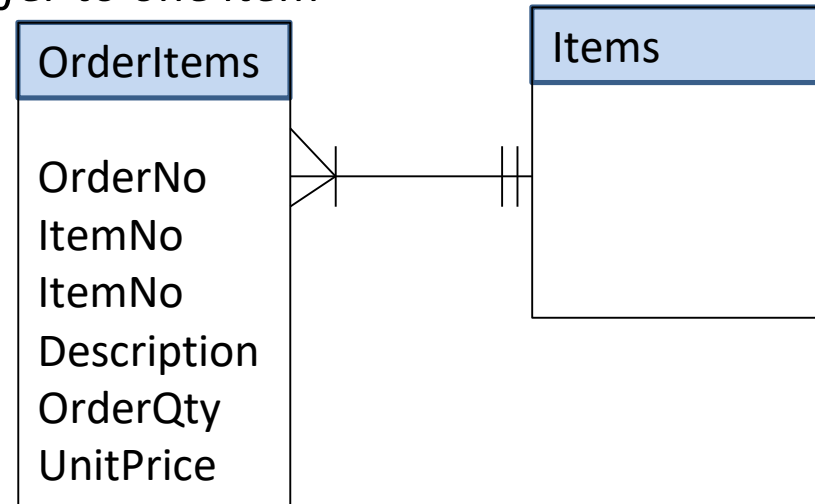
- *Since only OrderQty belongs in the ORDERITEMS table, create a new table (ITEMS) for the item information.*

## Step 2: Relationships

- *An ITEMS can appear in many ORDERITEMS table.*
- *A row in each ORDERITEMS table will only refer to one item*

## Step 3: Fields

## Step 4: Keys



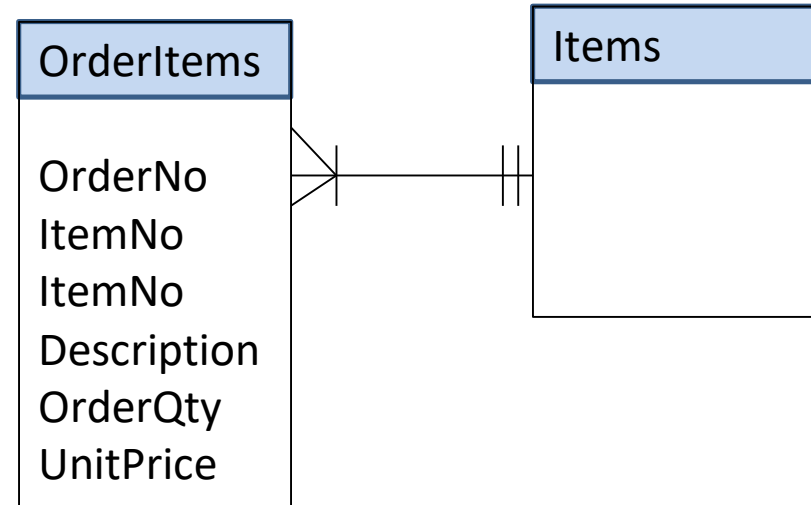
# Solving a 2NF Violation

## Step 1: Tables

- *Since only OrderQty belongs in the ORDERITEMS table, create a new table (ITEMS) for the item information.*

## Step 2: Relationships

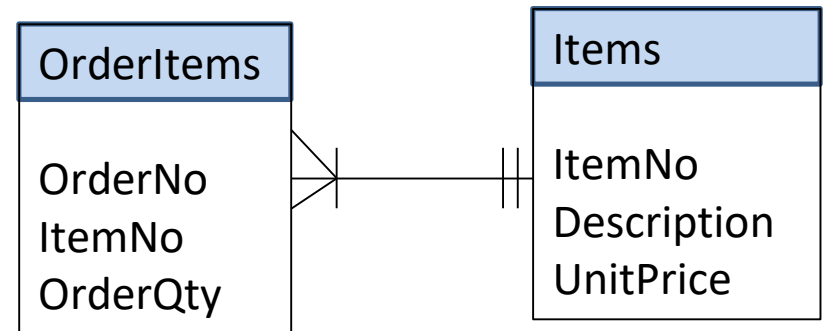
- *An ITEMS can appear in many ORDERITEMS table.*
- *A row in each ORDERITEMS table will only refer to one item*



## Step 3: Fields

- *The Description and UnitPrice are about ITEMS information*

## Step 4: Keys





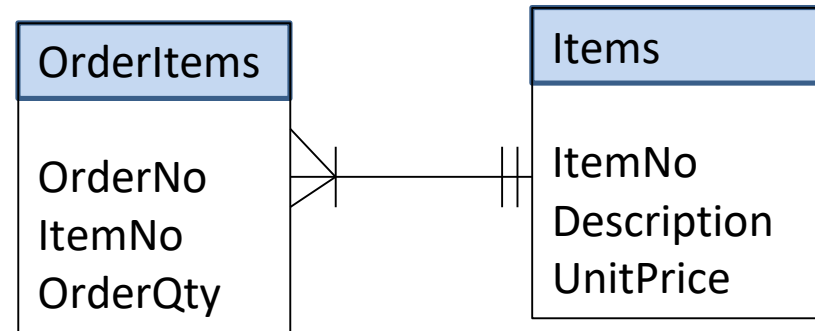
# Solving a 2NF Violation

## Step 1: Tables

- Since only *OrderQty* belongs in the *ORDERITEMS* table, create a new table (*ITEMS*) for the item information.

## Step 2: Relationships

- An *ITEMS* can appear in many *ORDERITEMS* table.
- A row in each *ORDERITEMS* table will only refer to one item

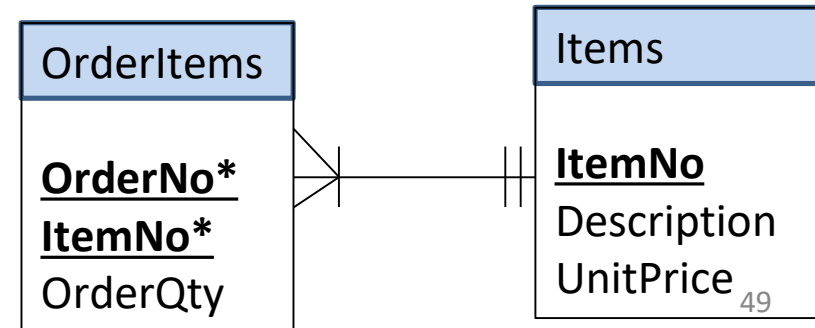


## Step 3: Fields

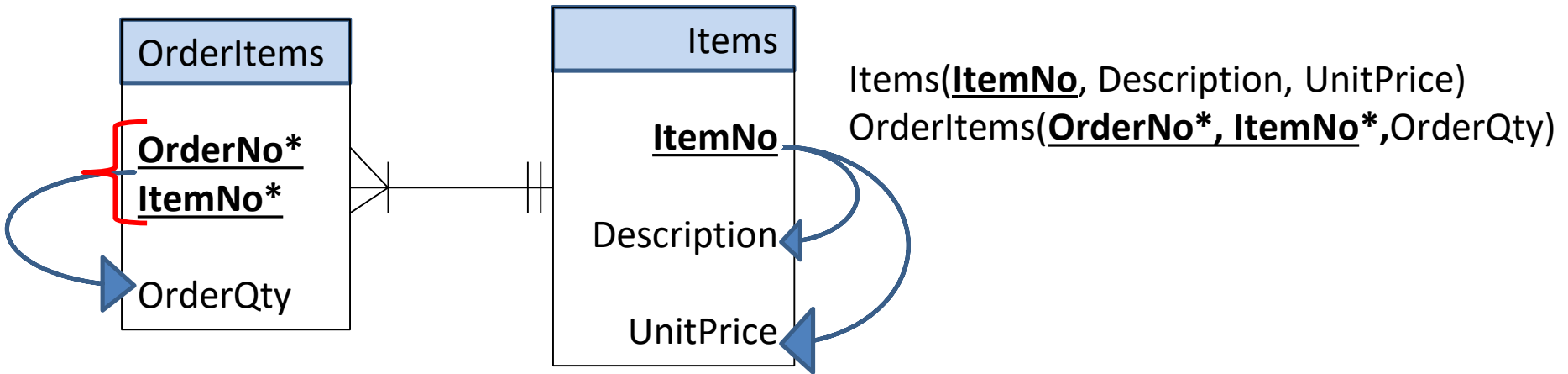
- The *Description* and *UnitPrice* are about *ITEMS* information

## Step 4: Keys

- The PK for **OrderItems** is a composite of (*OrderNo*, *ItemNo*).
- The PK for **Items** is *ItemNo*. This becomes the **FK** in *OrderItems*



# All Keys Are Now Determinants



All the fields in the ITEMS table are determined by the ItemNo.  
In ORDERITEMS, OrderQty is determined by the OrderNo + ItemNo.

OrderItems

Order No	Item No	Order Qty
C1001	K123	5
C1001	M345	3
C1001	P876	20
C1002	K124	12
C1002	K325	24
C1003	K123	10
C1003	M345	6
C1003	P876	25

Both the  
OrderItems and  
Items tables are  
**now in 2NF.**  
**There are no more  
partial  
dependency**

Items

Item No	Description	Unit Price
K123	Blue Felt Pen	2.50
M345	Stainless steel ruler	7.80
P876	Mars Bars	1.25
K124	Black Pen	2.50
K325	Marker Pen	1.90

# Third Normal Form (3NF) transformation

3NF: A table in which none of the non-key fields determine another non-key field (**no transitive dependency**)

The ORDERS table is already in 2NF as it fulfills the 2NF requirements:

- Already in 1NF (i.e. in each row, all columns contains atomic value)
- each non-key field is determined by the **whole primary key** and **NOT part of the primary key** by itself
  - OrderNo → CustomerNo, Name, TelNo, OrderDate

## Orders

Order No	Customer No	Name	Tel No	Order Date
C1001	A101	Ali bin Ahmad	1234567	12/5/14
C1002	A102	Lim Ah Kau	1122448	13/5/14
C1003	A101	Ali bin Ahmad	1234567	15/5/14

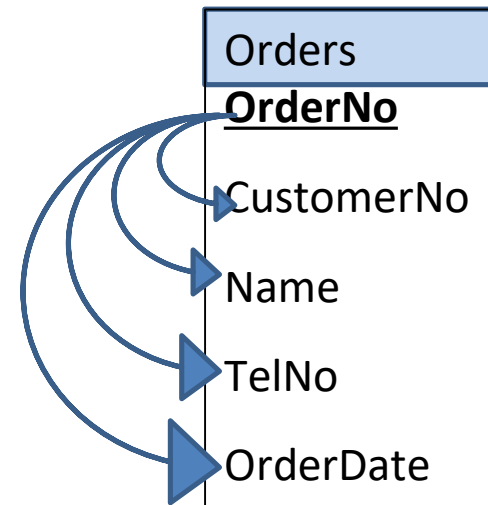


Exhibit 4-20: 3NF Violation

# Third Normal Form (3NF) transformation

3NF: A table in which none of the non-key fields determine another non-key field.

The ORDERS table is already in 2NF as it fulfill the 2NF requirements”

- Already in 1NF (i.e. in each row, all columns contains atomic value)
- each non-key field is determined by the **whole primary key** and **NOT part of the primary key** by itself
  - OrderNo → CustomerNo, Name, TelNo, OrderDate

However, once you know a CUSTOMER’S CustomerNo, you can determine his or her name and phone number.

**CustomerNo** is not a key field in the ORDERS relation. Therefore, the **Name, TelNo** are being determined by another non-key field. This is called a **transitive dependency**.

The **Name** and **TelNo** non-key fields are transitively dependent on the key field **OrderNo**.

To transform a relation to 3NF we must remove attributes Involved in the transitive dependency.

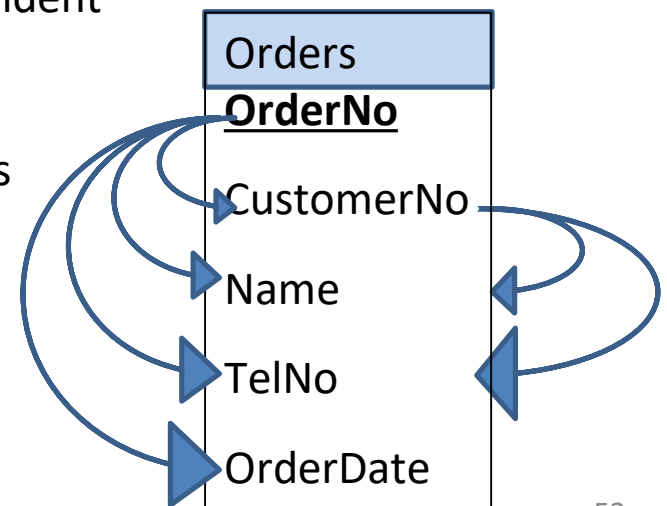


Exhibit 4-20: 3NF Violation

# Update Problem Caused by 3NF Violation

## Orders

Order No	Customer No	Name	Tel No	Order Date
C1001	A101	Ali bin Ahmad	1234321	12/5/14
C1002	A102	Lim Ah Kau	1122448	13/5/14
C1003	A101	Ali bin Ahmad	1234567	15/5/14

*Exhibit 4-21: 3NF Violation Creates Update Problem*

**Orders** table is **ONLY** in 2NF

Data not determined by the primary key will be duplicated and any updates may not be made to all instances of duplicate data.

In this example, we no longer know the correct TelNo for customer 'A101'. Will there be **Insertion** and **Deletion** anomalies for a **table in 2NF**?

# Solving a 3NF Violation

## Step 1: Tables

- *Create another table to store Name and TelNo that has transitive dependency. Call it CUSTOMERS table*

## Step 2: Relationships

## Step 3: Fields

## Step 4: Keys

Orders
<u>OrderNo</u>
CustomerNo
Name
TelNo
OrderDate

Customers

# Solving a 3NF Violation

## Step 1: Tables

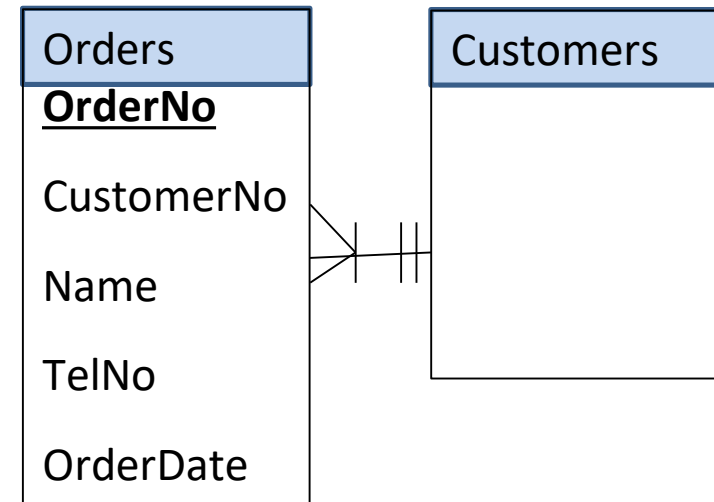
- *Create another table to store Name and TelNo that has transitive dependency. Call it CUSTOMERS table*

## Step 2: Relationships

- *A customer can have many orders*
- *An order belongs to only one customer*

## Step 3: Fields

## Step 4: Keys



# Solving a 3NF Violation

## Step 1: Tables

- *Create another table to store Name and TelNo that has transitive dependency. Call it CUSTOMERS table*

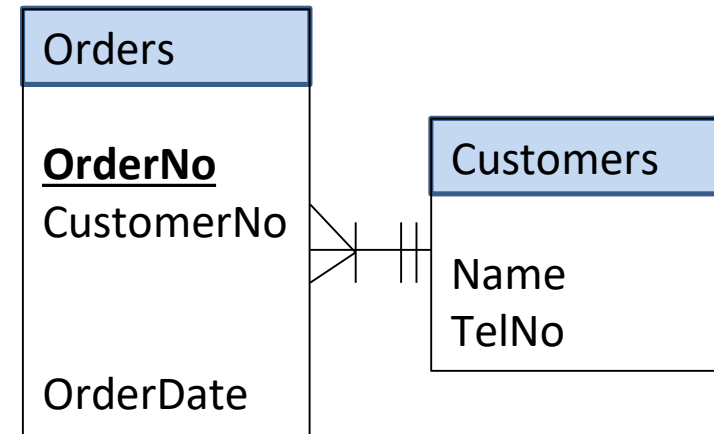
## Step 2: Relationships

- *A customer can have many orders*
- *An order belongs to only one customer*

## Step 3: Fields

- *Transfer Name and Telno to the*
- *CUSTOMERS table*

## Step 4: Keys





# Solving a 3NF Violation

## Step 1: Tables

- Create another table to store Name and TelNo that has transitive dependency. Call it CUSTOMERS table

## Step 2: Relationships

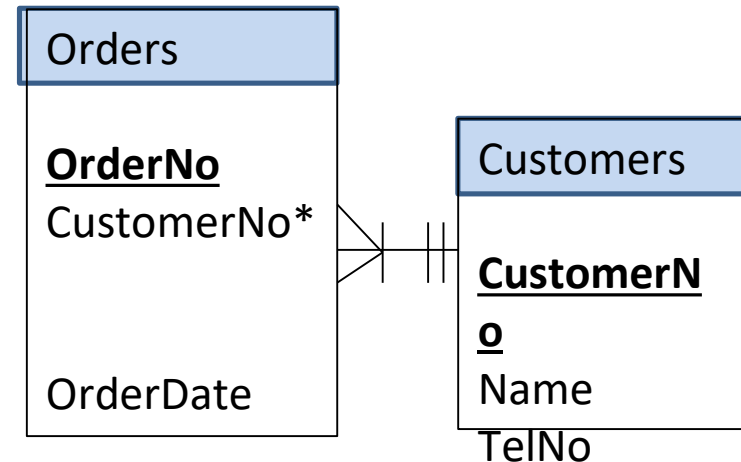
- A customer can have many orders
- An order belongs to only one customer

## Step 3: Fields

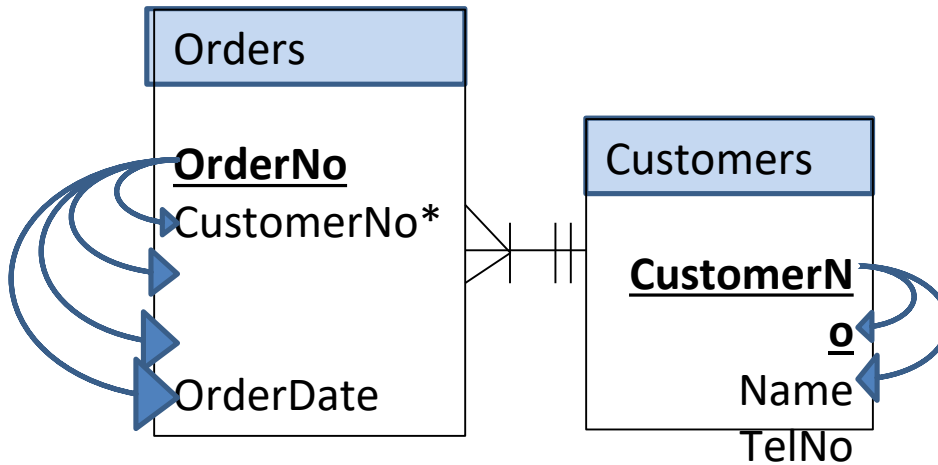
- Transfer Name and Telno to the CUSTOMERS table

## Step 4: Keys

- The PK for **Orders** is still OrderNo.
- The PK for **Customers** is CustomerNo. This becomes the **FK in Orders** to maintain a relation between **Customers** and **Orders**



# Keys Are Now Determinants



*Exhibit 4-23: 3NF Solution – Keys Are Now the Only Determinants*

All the fields in the **Orders** table are determined by **OrderNo**.

In **Customers**, all non-key fields are determined by **CustomerNo**.

# 3NF Solution With Sample Data

**Orders**

Order No	Customer No	Order Date
C1001	A101	12/5/14
C1002	A102	13/5/14
C1003	A101	15/5/14

**Customers**

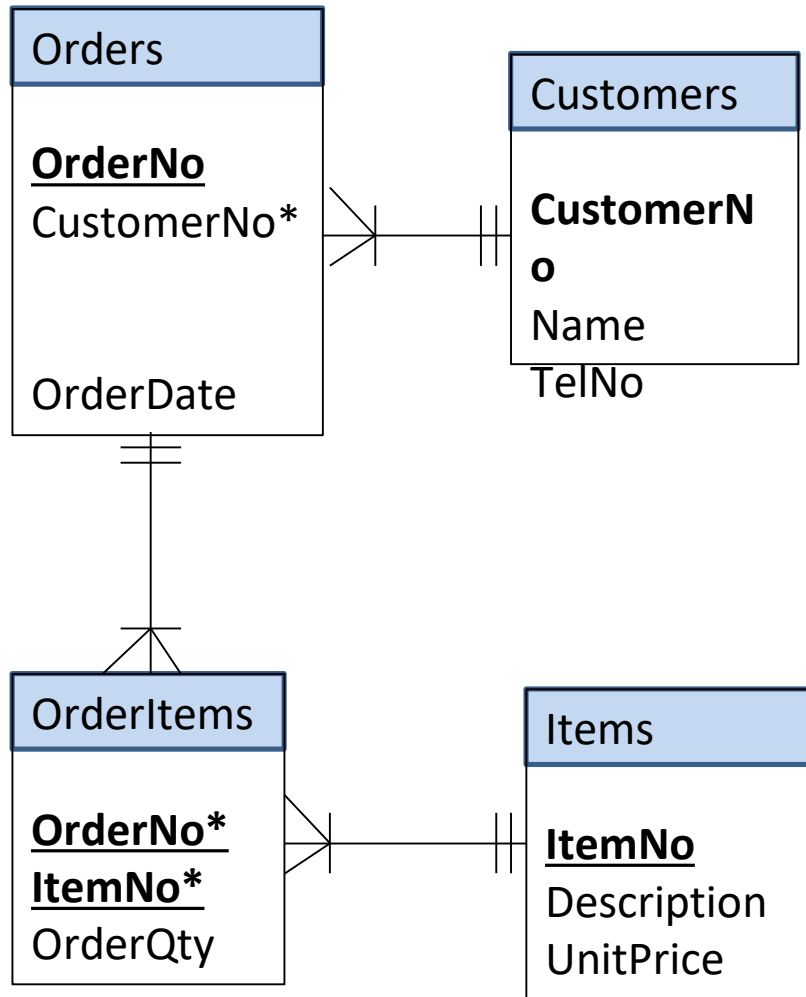
Customer No	Name	Tel No
A101	Ali bin Ahmad	1234321
A102	Lim Ah Kau	1122448

Orders (**OrderNo**, CustomerNo\*, OrderDate)

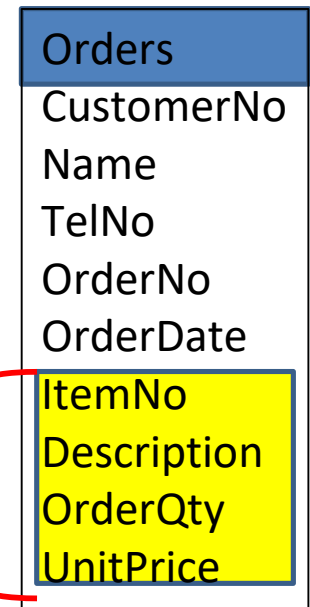
Customers(**CustomerNo**, Name, TelNo)

All the non-key fields are determined only by the primary key.

# Normalized vs. Unnormalized



Multiple values



All the relations are in 3NF:

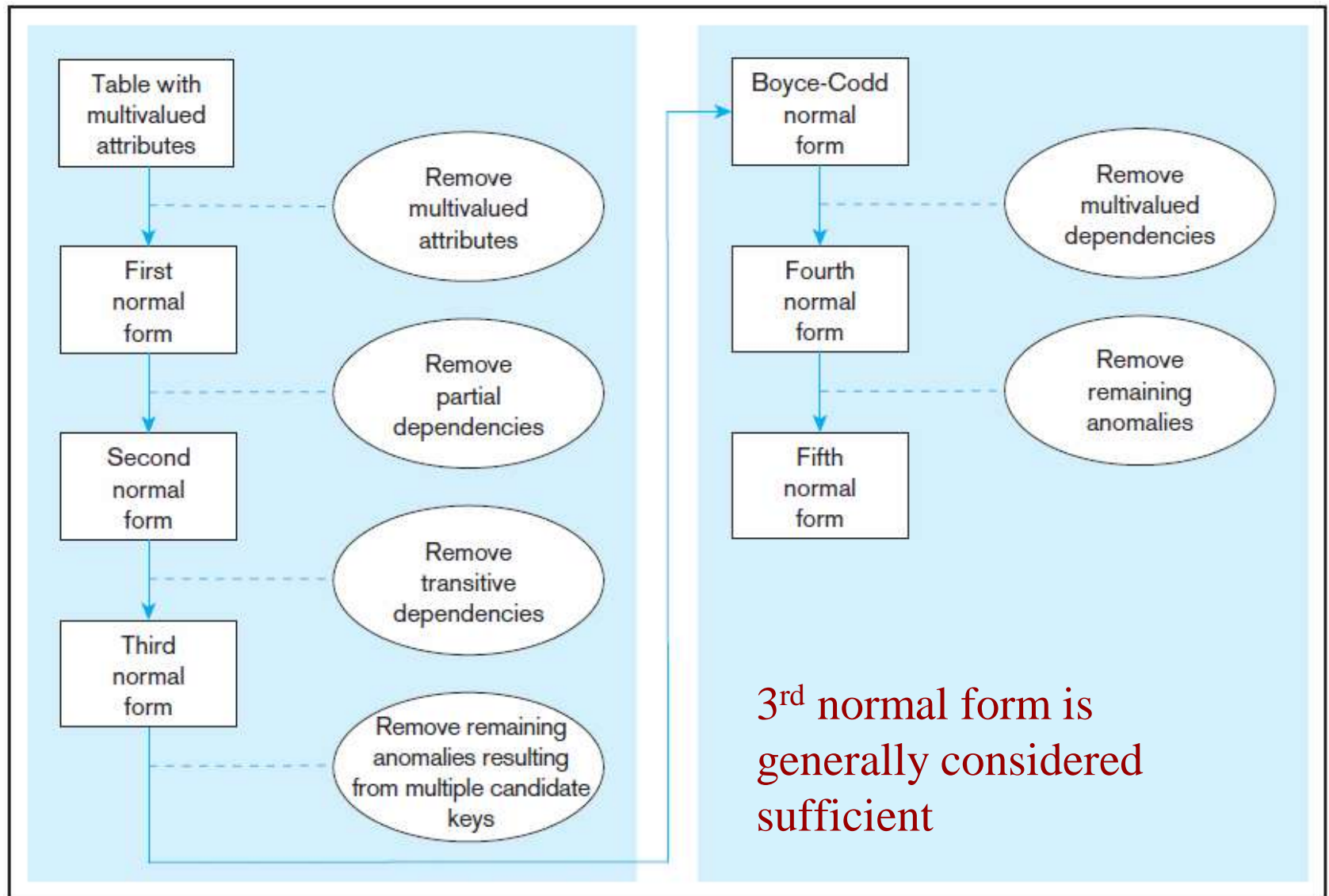
Customers(CustomerNo, Name, TelNo)

Items(ItemNo, Description, UnitPrice)

Orders (OrderNo, CustomerNo\*, OrderDate)

OrderItems(OrderNo\*, ItemNo\*, OrderQty)

Figure 4.22 Steps in normalization



# Normalisation - Definition

## 1NF rules:

- No multivalued attributes
- Every attribute value is atomic – single value only  
(no repeating group)
- 2NF: A table in which each non-key field is determined by the **whole primary key** and **NOT part of the primary key** by itself (no partial dependency)
- 3NF: A table in which none of the non-key fields determine another non-key field (no transitive dependency)

## 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

# 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

## 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

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



## First Normal Form

- No multivalued attributes
- Every attribute value is atomic

## 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 part of the key
  - No partial functional dependencies

## 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
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

# Denormalization

- Normalisation
  - Removes data redundancy
  - Solves INSERT, UPDATE, and DELETE anomalies
  - This makes it easier to maintain the information in the database in a consistent state
- However
  - It leads to many tables in the database
  - Often these need to be joined back together, which is expensive (requires more processing time) to do
  - So sometimes (not often) it is worth to 'denormalize'

# Denormalization

- You *might* want to denormalise if
  - Database processing speed are unacceptable (not just a bit slow)
  - There are going to be very few INSERTs, UPDATEs, or DELETEs
  - There are going to be lots of SELECTs that involve the joining of tables

Address

Number	Street	City	Postcode
--------	--------	------	----------

Not normalised since  
 $\{\text{Postcode}\} \rightarrow \{\text{City}\}$

Address1

Number	Street	Postcode
--------	--------	----------

Address2

Postcode	City
----------	------

# Denormalisation

- Database Design Goal : creation of normalized relations
- Problems of Normalization
  - Processing speed requirements should also be a goal
  - If tables decomposed to conform to normalization requirements, the number of database tables expands, which slows down processing.
  - Joining larger number of tables takes additional disk input/output (I/O) operations and processing logic (especially dealing with SQL) and reduces system speed
- Conflicts (Speed Vs Normalization)
  - Conflicts among design efficiency, information requirements, and processing speed are often resolved through compromises that may include denormalization.

# Summary

- Normalization is a process of optimizing databases to prevent update anomalies
- Normalization attempts to correct update issues by eliminating duplication
- Duplication also creates inconsistency
- Insertions can violate database integrity if the database is not normalized
- Deletions can violate database integrity if the database is not normalized

# Summary (con't.)

- Normal Forms – First (1NF), Second (2NF), Third(3NF)
- 1NF has no repeating groups
- 2NF is in 1NF and no non-key column is dependent on only a portion of the primary key
  - Every non-key attribute must be dependent on the ENTIRE primary key
- 3NF is in 2NF and the no transitive dependency
  - There are no “relationship” between the non-key attributes



## Summary

- Normalization is a table design technique aimed at minimizing data redundancies.
- First three normal forms (1NF, 2NF, and 3NF) are most commonly encountered.
- Tables are sometimes denormalized to yield less I/O which increases processing speed.