

## Relational Modelling

### Objectives

- Illustrate techniques to describe information in terms of table definitions and occurrences
- Guard against loss of information consistency
- How do we know our tables are correct ?

## Key Points In Rel. Modelling

1. Ordering of rows is not significant
2. Ordering of columns is not significant
3. Each row/column intersection contains a single attribute value. Multiple values are not allowed
4. Each row in a table must be distinct.  
=> row can always be uniquely identified by quoting an appropriate combination of attribute values

*A table conforming to these restrictions is called a normalised table*

## Exercise

Suggest a relation structure for a company wishing to manage its sales/customer records.

Each customer has a name and is serviced by one salesman. The structure must contain customer numbers, salesman numbers, customer names and salesman names.

## Duplicated vs Redundant Data

- Must be careful to distinguish between redundant and duplicated data
- Duplicated data:- occurs where an attribute (column) has two or more identical values
- Redundant data:- occurs if you can delete a value without information being lost

=> redundancy is unnecessary duplication

## Duplication vs. Redundancy (2)

### Example

Part#	Part Desc.		Part#	Part Desc.
P2	nut	<i>delete nut</i>  <i>loss of information</i>	P2	.....
P1	bolt		P1	bolt
P3	washer		P3	washer
P4	nut		P4	nut

=> nut was duplicate but NOT redundant!

Logical DB Design - Section 4  
©VW

5

## Duplicated vs. Redundant (3)

S#	Part#	Part Desc.		S#	Part#	Part Desc.
S2	P1	bolt	<i>Delete bolt</i>  <i>no loss of information</i>	S2	P1	bolt
S7	P6	bolt		S7	P6	bolt
S2	P4	nut		S2	P4	nut
S5	P1	bolt		S5	P1	.....

=> duplicated data was redundant

Logical DB Design - Section 4  
©VW

6

## Eliminating Redundancy

- We cannot just delete values from the table in the pervious example!
- Preferable to split table into 2 tables

Part#	Part Desc.	S#	Part#
P1	bolt	S2	P1
P6	bolt	S7	P6
P4	nut	S2	P4
		S5	P1

Logical DB Design - Section 4  
©VW

7

## Eliminating Redundancy (2)

- Eliminated redundancy by table splitting
  - P1 description only appears once
  - relationship is made by including part# in two tables
- So far we've assumed that table structures which permit redundancy can be recognised by inspection of table occurrence
- This is not entirely accurate since attribute values are subject to insertion / change / deletion

Logical DB Design - Section 4  
©VW

8

## Eliminating Redundancy (3)

SP

S#	P#	Desc
S2	P1	bolt
S7	P6	bolt
S2	P4	nut

Inspection of table SP does not reveal any redundancy

Could even suggest that  
"no two suppliers may supply same part#"

Logical DB Design - Section 4  
©VW

9

## Repeating Groups

- We stated earlier that:  
"An attribute must only have one value in each row"

S#	SName	P#
S5	Wells	P1
S2	Heath	P1, P4
S7	Barron	P6
S9	Edwards	P8, P2, P6

Logical DB Design - Section 4  
©VW

10

## Repeating Groups (2)

Problems:

- Table is asymmetric representation of symmetrical data
- Rows can be sorted into s# but not into p#
- Rows are different length 'cos of variation in number of p#'s
- If rows were fixed length, they would need to be padded with null values

Logical DB Design - Section 4  
©VW

11

## Elimination of Repeating Groups

- Easiest way to eliminate repeating groups is to write out the table occurrence using a vertical layout and fill in the blanks by duplicating the non-repeating data necessary

S#	SName	P#
S5	Wells	P1
S2	Heath	P1
S2	Heath	P4
S7	Edwards	P8
S7	Edwards	P2
S7	Edwards	P6



S#	SName	P#
S5	Wells	P1
S2	Heath	P1
S2	Heath	P4
S7	Edwards	P8
S7	Edwards	P2
S7	Edwards	P6

But this can lead to 'redundancy' of information!  
(Does the right hand table contain redundant information?)

Logical DB Design - Section 4  
©VW

12

## Elimination of Repeating Groups(2)

### Alternate method:

- Split table into two tables so that repeating group appears in one table and rest of attributes in another
- Need to provide correspondance between tables by including a key attribute with the repeating group table

S#	SName
S5	Wells
S2	Heath
S7	Barron
S9	Edwards

S#	P#
S5	P1
S2	P1
S2	P4
S7	P6
S9	P8
S9	P2
S9	P6

Logical DB Design - Section 4  
©VW

13

## Eliminating Repeating Groups & Redundancy

- Snapshot of table is inadequate guide to presence / absence of redundant data
- Need to know underlying rules
- DBA must discover rules which apply to conceptual model

Logical DB Design - Section 4  
©VW

14

## Determinants

- If there are rules such that duplicate values of attribute A are always associated with the same value of attribute B (within any given occurrence of the table) then attribute A is a determinant of attribute B

Note: A is determinant of B can be written as

$$A \longrightarrow B$$

**Note:** In the special case where duplicate values of A are not allowed in a table ( i.e. A is a key) then A is obviously a determinant on B

Logical DB Design - Section 4  
©VW

15

## Determinants (2)

### **Example:**

- If each possible p# value has precisely one associated part description value (i.e. P4 has just one description nut) then we can say that p# is a determinant of part description.

$$p\# \longrightarrow p\_desc$$

- Similarly if each possible p# value has precisely one quantity in stock then we can say p# is a determinant of quantity in stock

$$p\# \longrightarrow Qty\_in\_Stock$$

Logical DB Design - Section 4  
©VW

16

## Determinants (3)

Stock

P#	P_Desc	Qty
P2	nut	5000
P1	bolt	8300
P3	washer	9750
P4	nut	2326

Question:

is P\_Desc a determinant of P# ?

is P\_Desc a determinant of Qty ?

Logical DB Design - Section 4  
©VW

17

## Superfluous Attribute

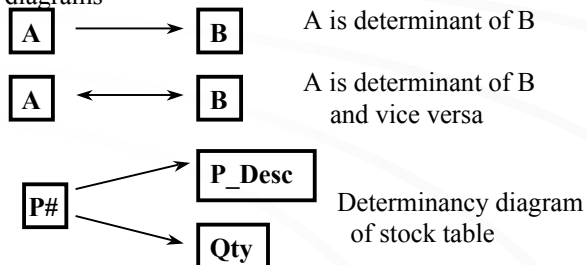
- If P# determines Qty then composite attribute {P#, P\_Desc} also determines Qty, but P\_Desc is superfluous
- We assume determinants do not contain any superfluous attributes

Logical DB Design - Section 4  
©VW

18

## Determinancy Diagrams

- We need a notation to express where one attribute determines another => we use determinancy diagrams



Logical DB Design - Section 4  
©VW

19

## Determinancy Diagrams (2)

### EXAMPLE

- Supplier identified by single S# & a part is identified by single P#
- Each supplier has only one SName but different suppliers may have same names
- A supplier may supply many different parts in many different pack sizes
- A part may be supplied by many different suppliers in many different pack sizes
- A given supplier supplies a given part in just one pack size

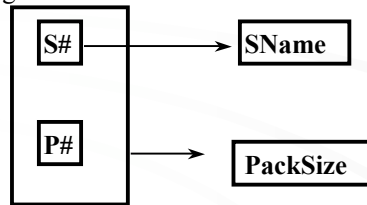
Logical DB Design - Section 4  
©VW

20

## Determinancy Diagrams (3)

Supp-Part (S#, Sname, P#, PackSize)

- Can recognise determinants by drawing determinancy diagrams



**{ S#, P# } is determinant for packsize**

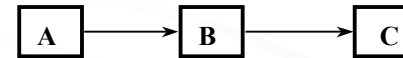
Logical DB Design - Section 4  
©VW

21

## Transitive Determinants

- If A is determinant of B & B is determinant of C

Then A is determinant of C



Logical DB Design - Section 4  
©VW

22

## Identifiers

- Because of the rule that 'no two rows in a table can have identical values throughout'

therefore

individual row can always be identified by quoting the values of all its attributes. However some values may not be needed.

Logical DB Design - Section 4  
©VW

23

## Identifiers (2)

Example:

Employee ( Employee#, Employee\_name, Salary)

Rules:

- No two rows should have the same value for Employee#

=> Employee# is a row identifier of the table

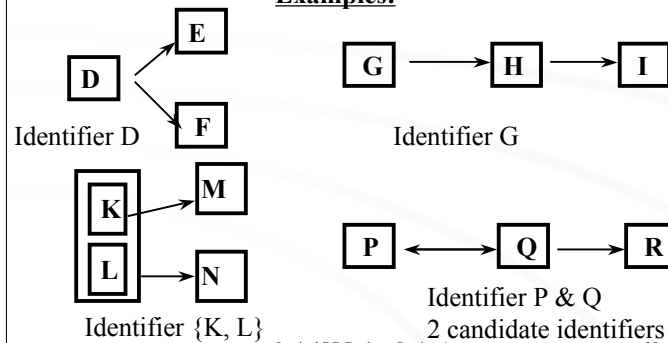
- Where a composed attribute forms the identifier  
=> no component part (of identifier) can be null  
(entity constraint)

Logical DB Design - Section 4  
©VW

24

### Identifiers (3)

#### Examples:

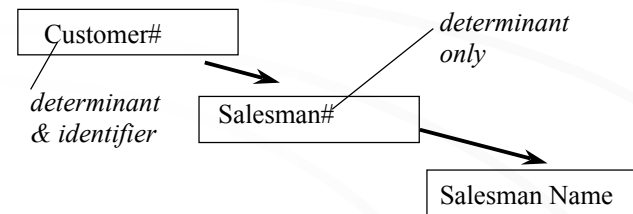


Logical DB Design - Section 4  
©VW

25

### Determinancy & Redundancy

- Given a determinancy diagram (developed from enterprise rules) we can detect and eliminate table structures which could contain redundant data



Logical DB Design - Section 4  
©VW

26

### Determinancy & Redundancy(2)

- Each customer# is associated with one salesman# but a salesman# may be associated with several different customer#
- Therefore salesman# could have duplicate values
- But salesman# is a determinant of salesman name
- Therefore each occurrence of a duplicate salesman# values will be associated with the same salesman name  
=> table can contain redundant values of salesman

Logical DB Design - Section 4  
©VW

27

### Determinancy & Redundancy (3)

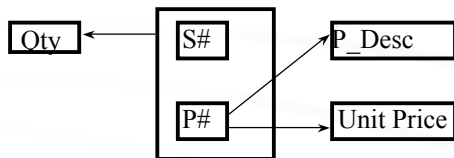
- But customer# values cannot be duplicated ( because customer# is the identifier for our table) so we cannot allow redundant values of salesman#
- Potential redundancy arises because salesman# is a determinant but not a candidate identifier

Logical DB Design - Section 4  
©VW

28

## Determinancy & Redundancy (4)

### Example 2



Potential redundancy in values of P\_desc and Unit\_Price  
 P# is a determinant but not a candidate identifier!  
 Gives rise to Boyce-Codd Rule for detecting redundancy

Logical DB Design - Section 4  
 ©VW

29

## Transforming tables in well-normalised table

- Boyce/Codd rule for determining redundancy is rule "Every determinant must be a candidate identifier"
- A table which obeys this rule is said to be in Boyce / Codd normal form (BCNF)

to put it another way:

"all attributes in a relation should be dependent on the key, the whole key and nothing but the key"

Logical DB Design - Section 4  
 ©VW

30

## Transformation into Well N.F.

- A determinant which is not a candidate identifier is called a non identifying determinant
- To transform a badly normalised (non BCNF) table into well normalised tables:

*Create new tables such that each non identifying determinant in the old table becomes a candidate identifier in a new table*

Logical DB Design - Section 4  
 ©VW

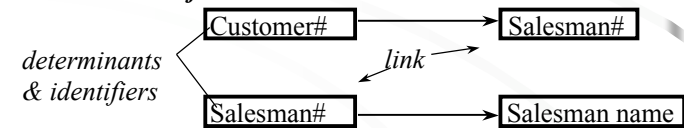
31

## Example of BCNF Normalisation

### Example 1



in normalised form is:

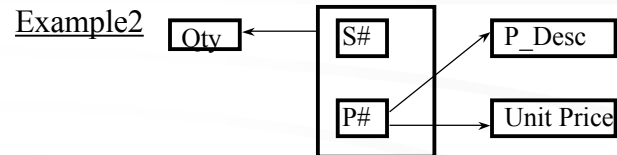


Logical DB Design - Section 4  
 ©VW

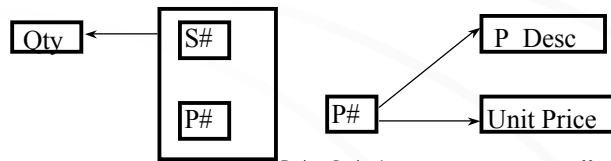
32



## Example of BCNF Normalisation (2)



in normalised form is:



Logical DB Design - Section 4  
©VW

33

## Fully Normalised Tables

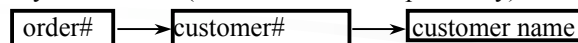
- Fully normalised tables are structured in such a way that they cannot contain redundant data
- Generally a well normalised table (i.e. one in which each determinant is a candidate identifier) is also fully normalised, but not always! - so further normalisation may be desirable.

Logical DB Design - Section 4  
©VW

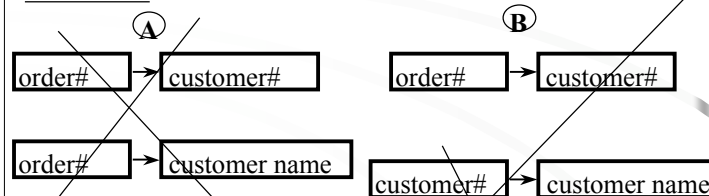
34

## Fully Normalised Tables (2)

Badly Normalised (hidden transitive dependency)



Normalised



Logical DB Design - Section 4  
©VW

35

## Fully Normalised Tables (3)

Ⓐ contains redundant data e.g.

order#	customer#
1	C1
2	C2
3	C1

order#	customer name
1	Smith
2	Jones
3	Smith

- If delete smith from row 1 of customer name table
  - can still use order#(1) to find corresponding customer# in order cust table
  - search order cust table for another order# placed by that customer (order# 3 or 4)
  - uses order# to search customer name table for corresponding customer name

Logical DB Design - Section 4  
©VW

36

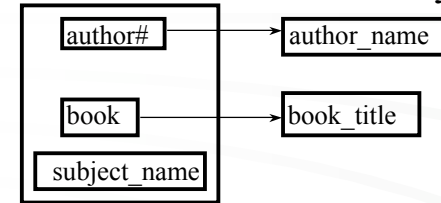
## Fully Normalised Tables (3)

- Basic error: associate in customer name table a determinant (order#) with the transitively dependant attribute customer name

Logical DB Design - Section 4  
©VW

37

## Multi-valued Determinacy



Well normalised tables:

Author (author#, author\_name) ✓

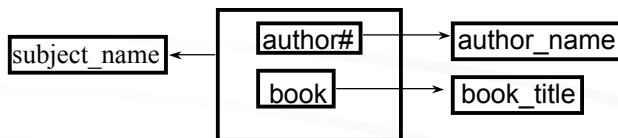
Book (book#, book\_title) ✓

Author\_Book\_Subject(author#, book#, subject\_name) ✗

Logical DB Design - Section 4  
©VW

38

## Multi-valued Determinacy(2)



If author is responsible for particular subject content of book

Author\_book\_subject2(author#, book#, subject\_name) ✓

Logical DB Design - Section 4  
©VW

39

## Enterprise Rules for Book example

- each book has a unique book#
- each author has a unique author #
- each subject classification has a unique subject\_name
- book# does not distinguish an individual copy of a book, only an individual work
- A book may be written by several authors and be classified under several subject\_names
- an author may write several books
- a subject\_name may be applied to several books

Logical DB Design - Section 4  
©VW

40

### Book example (2)

- Book#B15 is jointly authored by A2 and A5 and is classified under subject names biology & physics.
- If every author of a given book is always associated with all the subject-names under which the book is classified, then the attribute subject-name can contain certain redundant values.

### Book example (3)

- If subject names biology & physics were deleted from rows 1 and 2, it would be possible to deduce those values from row 3 and 4

*Therefore*  
author-book-subject  
is well but  
not fully  
normalised

author	book	subject
A1	B1	biology
A1	B1	physics
A2	B15	biology
A2	B15	physics
A5	B15	biology
A5	B15	physics

### Book example (4)

- The table is not fully normalised because the same set of subject-names is associated with each author of the same book.  
Book# is said to *multi-determine* author# & subject\_name.

#### Note

This would not be true if a different rule were assumed. i.e. that subject-name refers to a subject area within a book for which an individual author was responsible.

eg. delete biology from row 3 => you cannot deduce the info. from elsewhere in the table.

### Book example (5)

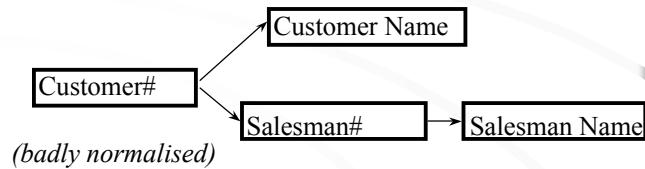
- Full normalisation can be achieved by splitting the table into two parts:

author	book	subject
A1	B1	biology
A1	B1	physics
A2	B15	biology
A2	B15	physics
A5	B15	biology
A5	B15	physics

## Advantages of Full Normalisation

- So far emphasis has been placed on eliminating redundancy. Further benefits relate to deletion, insertion operations.

### Deletion Side Effect



Logical DB Design - Section 4  
©VW

45

## Advantages of Normalisation(2)

C#	CName	S#	SName
C1	Brown	S4	Jones
C2	Carter	S7	Samson
C3	Cross	S4	Jones
C4	Barns	S8	Baker

- Delete C2 => delete whole tuple  
=> lose salesman information
- Deleting C# on its own is not allowed as it is an identifier and cannot be null

Logical DB Design - Section 4  
©VW

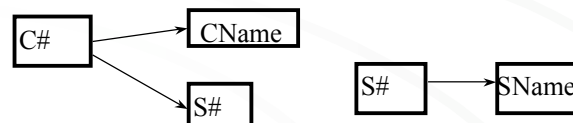
46

## Advantages of Normalisation(3)

### Insertion Side Effect

- Add S3 whose name is Hall
- You cannot do this until that salesman is associated with a customer, otherwise identifier C# will be null

Should be modelled as:



Logical DB Design - Section 4  
©VW

47

## Review

Logical DB Design - Section 4  
©VW

48