

8.1.2 Normalisation from UNF to 3NF

Name: _____ Class: _____ Date: _____ **Total: 35 marks**

KEY WORDS normalisation • functional dependency • partial dependency • transitive dependency

Objective

Turn repeated order data into linked tables in Third Normal Form. Explain each change using the fields and their dependencies. Use the keys and relationships from sheets 8.1 and 8.1.1.

1 Key knowledge and worked examples

■ Why split a table?

A school shop records orders. If a product name is copied into many order rows, renaming that product requires many updates. Missing one update creates inconsistent names. **Normalisation** organises data into related tables to reduce unnecessary duplication and update problems. Repeated foreign keys remain necessary.

■ The rules for our worked example

Each OrderID identifies one order, its date and one customer. Each CustomerID identifies one customer name. Each ProductID identifies one product name. A product appears at most once in an order, with a Quantity for that order-product pair. Names need not be unique.

A **functional dependency** means one value determines another. Write **ProductID** - > **ProductName**: knowing the ProductID determines one ProductName under the rules. This is not a claim that the name uniquely identifies the product.

■ Step 1: remove the repeating group for 1NF

The starting record contains a list of products and a matching list of quantities. This is **unnormalised form (UNF)** . The lists are a **repeating group** .

UNF order

OrderID O1; Date 09/10; CustomerID C1; CustomerName Mei
Products: {P1, Pen, 2}; {P2, Pad, 1}

One row per product in the order:

1NF intermediate order table						
<u>OrderID</u>	<u>Date</u>	CustomerID	CustomerName	<u>ProductID</u>	ProductName	Qty
O1	09/10	C1	Mei	P1	Pen	2
O1	09/10	C1	Mei	P2	Pad	1

Date means OrderDate. Qty means Quantity. Both underlined fields form the key.

First Normal Form (1NF) requires single, atomic values and no repeating groups. Replace the lists with one row per order-product pair. In this intermediate table the composite key is (OrderID, ProductID): an order may contain several products, and a product may occur in several orders. The key is underlined in the diagram. The table is now 1NF, but it still repeats customer and product details.

■ Step 2: remove partial dependencies for 2NF

Dependencies in the 1NF table

OrderID → OrderDate, CustomerID, CustomerName
ProductID → ProductName
(OrderID, ProductID) → Quantity

ORDER

OrderID
OrderDate
CustomerID,
CustomerName

ORDER_LINE

OrderID (FK)
ProductID (FK)
Quantity

PRODUCT

ProductID
ProductName

2NF: move fields determined by only part of the pair to their own tables.

A **partial dependency** occurs when a non-key field depends on only part of a composite candidate key. ProductName depends on ProductID alone. OrderDate, CustomerID and CustomerName depend on OrderID alone. Quantity needs the whole pair: the same product can have different quantities in different orders.

Second Normal Form (2NF) requires 1NF and no partial dependencies. Split the intermediate table into ORDER, PRODUCT and ORDER_LINE as shown. Keep both IDs in ORDER_LINE as its composite primary key and as foreign keys. No order-product quantities are lost.

■ Step 3: remove the transitive dependency for 3NF

Dependency still inside ORDER
OrderID → CustomerID → CustomerName

ORDER
OrderID
OrderDate
CustomerID (FK)

CUSTOMER
CustomerID
CustomerName

3NF: CustomerName moves to CUSTOMER; CustomerID stays in ORDER to link it.

ORDER still stores a customer name for every order. OrderID -> CustomerID -> CustomerName is a **transitive dependency** : CustomerName depends on the non-key CustomerID. Move customer details to CUSTOMER and keep CustomerID in ORDER as a foreign key.

Third Normal Form (3NF) requires 2NF and no transitive dependency of non-key fields on the key. For these examples, every non-key field depends on the key, the whole key, and no other non-key field. Use this as a reminder after checking the dependencies, not as a substitute for naming them.

■ Check the final design

CUSTOMER	
<u>CustomerID</u>	CustomerName
C1	Mei

ORDER		
<u>OrderID</u>	OrderDate	CustomerID (FK)
O1	09/10	C1
O2	10/10	C1

PRODUCT	
<u>ProductID</u>	ProductName
P1	Pen
P2	Pad

ORDER_LINE		
<u>OrderID (FK)</u>	<u>ProductID (FK)</u>	Quantity
O1	P1	2
O1	P2	1
O2	P1	4

Underlined: primary keys. ORDER LINE uses the pair. FK: foreign key.



Trace order O1: its CustomerID finds Mei; its two ORDER_LINE rows find Pen and Pad and their quantities. Each descriptive fact has one home. The linked design still answers the original question.

To justify 3NF, check **all three stages**: atomic values and no repeating groups; no partial dependencies; no transitive dependencies. Name the relevant fields. Repeated IDs alone do not prove a violation. Different valid decompositions may reach 1NF; separating a repeating group into its own table is also possible.

2 Practice

2.1 The EVENT record contains lists of MemberID and corresponding Hours. Each member appears once per event. Identify the repeating group and rewrite the data as 1NF rows. State the composite key. [4]

EVENT (UNF)			
EventID	EventName	MemberID list	Hours list
E1	Fair	M1, M2	2, 3

2.2 ORDER_LINE has primary key (OrderID, ProductID). ProductID determines ProductName, while Quantity depends on the order-product pair. Name the dependency that prevents 2NF, identify its fields, and complete the split: PRODUCT(_____) and ORDER_LINE(_____) . State both primary keys. [4]

ORDER_LINE
OrderID, ProductID
ProductName, Quantity

2.3 EMPLOYEE has primary key EmployeeID. Each employee belongs to one department; DepartmentID determines DepartmentName. Name the dependency preventing 3NF, then give the two resulting tables and identify the foreign key. [4]

EMPLOYEE
EmployeeID, Name
DepartmentID, DepartmentName

2.4 The table below has primary key (StudentID, TestID). All values are atomic, a pair occurs once, and Mark depends on the full pair. No other dependencies exist. State its highest normal form up to 3NF and justify your answer. [3]

STUDENT_TEST		
<u>StudentID</u>	<u>TestID</u>	Mark
S1	T1	8
S1	T2	7
S2	T1	9

3 Exam-style questions

3.1 Each BOOK has one ISBN, Title and PublisherID. ISBN uniquely identifies a book. Each publisher has one PublisherName. All values in the table are atomic. Explain why the table is in 2NF but not 3NF, then give a 3NF design with primary and foreign keys.[5]

BOOK

ISBN, Title

PublisherID, PublisherName

3.2 A shop uses the unnormalised DELIVERY record shown. Each DeliveryID determines its Date and DriverID. Each DriverID determines DriverName. Each ItemID determines ItemName. An item appears at most once per delivery, and Quantity depends on the delivery-item pair. Produce a 3NF design. Underline every primary key, state every foreign key, and explain one partial and one transitive dependency removed. [8]

DELIVERY (UNF)

DeliveryID, Date, DriverID, DriverName

{ItemID, ItemName, Quantity}

3.3 A vet records pets, owners, visits and treatments. Each pet has one owner; an owner may have several pets. Each visit is for one pet and may include several treatments. Each treatment has a unique ID, name and fixed cost; the same treatment is recorded at most once per visit. Names are not unique. Store the owner’s name and telephone, the pet’s name and species, and each visit’s date. Produce a 3NF design with all primary and foreign keys. Explain why the visit-treatment table needs a composite key. [7]

Visit card

Pet: name, species

Owner: name, telephone

Date; list of treatments and costs