## Question1

Suppose you are given a relation R = (A, B, C, D, E) with the following functional dependencies: {CE → D, D → B, C → A}.

1. Find all candidate keys.
2. Identify the best normal form that R satisﬁes (1NF, 2NF, 3NF).
3. If the relation is not in 3NF, decompose it until it becomes 3NF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

**Answer**

1. The only key is {C, E}
2. The relation is in 1NF
3. Decompose into R1 = (A, C) and R2 = (B, C, D, E).

R1 is in 3NF, R2 is in 2NF.

Decompose R2 into, R21 = (C, D, E) and R22 = (B, D).

Both relations are in 3NF.

**Question 2**

Consider R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies:

{AB 🡪 C, A 🡪 DE, B 🡪 F, F 🡪 GH, D 🡪 IJ}

1. Find all candidate keys.
2. Identify the best normal form that R satisﬁes (1NF, 2NF, 3NF).
3. If the relation is not in 3NF, decompose it until it becomes 3NF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

**Answer**

1. The key is {A, B}.
2. The relation is in 1NF
3. Decompose into

{A, B, C} (3NF), and {A, D, E, I, J} and {B, F, G, H} (both in 2NF)

Decompose into

{A, B, C} {A, D, E} {D, I, J} {B, F} {F, G, H} (all in 3NF)

**Question 3**

Consider R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies:

{AB 🡪 C, BD 🡪 EF, AD 🡪 GH, A 🡪 I, H 🡪 J}

1. Find all candidate keys.
2. Identify the best normal form that R satisﬁes (1NF, 2NF, 3NF).
3. If the relation is not in 3NF, decompose it until it becomes 3NF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

**Answer:**

1. The key is {A, B, D}
2. The relation is in 1NF
3. Decompose into

{A, B, C} {B, D, E, F} {A, I} (all in 3NF) and {A, D, G, H, J} (in 2NF)

Decompose into

{A, D, G, H} and {H, J} (now 3NF)

**Question 4:**

Given the sample order form and description below, create the un-normalized form of the data and transform through the steps needed to move into 3NF.

**Gallery Customer History Form**

Customer Name

Jackson, Elizabeth Phone (206) 284-6783

123 – 4th Avenue

Fonthill, ON, L3J 4S4

Purchases Made

Artist Title Purchase Date Sales Price

03 - Carol Channing Laugh with Teeth 09/17/2000 7000.00

15 - Dennis Frings South toward Emerald Sea 05/11/2000 1800.00

03 - Carol Channing At the Movies 02/14/2002 5550.00

15 - Dennis Frings South toward Emerald Sea 07/15/2003 2200.00

The Gill Art Gallery wishes to maintain data on their customers, artists and paintings. They may have several paintings by each artist in the gallery at one time. Paintings may be bought and sold several times. In other words, the gallery may sell a painting, then buy it back at a later date and sell it to another customer.

**Solution:**

**UNF:**

customer [ custno, cust\_name, cust\_addr, cust\_phone, ( artist\_id, artist\_name, art\_title, pur\_date, price) ]

**1NF:**

customer [ custno, cust\_name, cust\_addr, cust\_phone]

cust\_art [ custno, art\_code, pur\_date, artist\_id, artist\_name, art\_title, price ]

**note:** the key chosen for the repeating group is the piece of art itself (a code was assigned), however because a piece of art may be bought by a customer more than once, the purchase date was added as part of the key to make the rows unique.

**2NF:**

customer [ custno, cust\_name, cust\_addr, cust\_phone]

cust\_art [ custno, art\_code, pur\_date, price ]

art [ art\_code, art\_title, artist\_id, artist\_name ]

**3NF:**

customer [ custno, cust\_name, cust\_street, cust\_city, cust\_prov, cust\_pstlcd, cust\_phone]

cust\_art [ custno, art\_code, pur\_date, price ]

art [ art\_code, art\_title, artist\_id(FK) ]

artist [ artist\_id, artist\_fname, artist\_lname ]

**Question 5:**

Given the sample forms and description below, create the un-normalized form of the data and transform through the steps needed to move into 3NF. Finally, merge the 2 solutions into a unified schema:

**Good News Grocers**

**User View 1 - Price Update List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Department** | **Product**  **Code** | **Aisle**  **Number** | **Price** | **Unit of Measure** |
| Produce | 4081 | 1 | 0.35 | lb |
| Produce | 4027 | 1 | 0.90 | ea |
| Produce | 4108 | 1 | 1.99 | lb |
|  |  |  |  |  |
| Butcher | 331100 | 5 | 1.50 | lb |
| Butcher | 331105 | 5 | 2.40 | lb |
| Butcher | 332110 | 5 | 5.00 | lb |
|  |  |  |  |  |
| Freezer | 411100 | 6 | 1.00 | ea |
| Freezer | 521101 | 6 | 1.00 | ea |
| Freezer | 866503 | 6 | 5.00 | ea |
| Freezer | 866504 | 6 | 5.00 | ea |

This report is used by the department managers to update the prices that are displayed in the grocery store for these products.

# User View 2: Product Cost Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Supplier** | **Product** | **Cost** | **Markup** | **Price** | **Dept**  **Code** |
| 21 – Very Veggie | 4108 – tomatoes, plum | 1.89 | 5% | 1.99 | PR |
| 32 – Fab Fruits | 4081 – bananas | 0.20 | 75% | 0.35 | PR |
| 32 – Fab Fruits | 4027 – grapefruit | 0.45 | 100% | 0.90 | PR |
| 32 – Fab Fruits | 4851 – celery | 1.00 | 100% | 2.00 | PR |
| 08 – Meats R Us | 331100 – chicken wings | 0.50 | 300% | 1.50 | BU |
| 08 – Meats R Us | 331105 – lean ground beef | 0.60 | 400% | 2.40 | BU |
| 08 – Meats R Us | 332110 – boneless chicken breasts | 2.50 | 100% | 5.00 | BU |
| 10 – Jerry’s Juice | 411100 – orange juice | 0.25 | 400% | 1.00 | FR |
| 10 – Jerry’s Juice | 521101 – apple juice | 0.25 | 400% | 1.00 | FR |
| 45 – Icey Creams | 866503 – vanilla ice cream | 2.50 | 100% | 5.00 | FR |
| 45 – Icey Creams | 866504 – chocolate ice cream | 2.50 | 100% | 5.00 | FR |

This report is used by the grocery store manager to determine the final selling price of his products.

**Example Solution 1:**

UNF:

product [ prod\_code, dept, aisle\_no, price, um ]   
  
1NF:

same

2NF:

same

3NF:

product [ prod\_code, dept\_cd (FK), price, um ]   
 dept [ dept\_cd, dept\_name, aisle\_no ]

**Note**: it may be debatable whether department actually determines aisle number. This may be true in a small grocery store but in a large grocery store, products from a department may be found in multiple aisles. This should be discussed with your database client to determine their exact needs.

UNF:

product [ prod\_code, prod\_desc, supplier\_id, supplier\_name, cost, markup, dept\_cd ]

1NF:

same (because there were no repeating groups)

2NF:

same

3NF:

product [ prod\_code, prod\_desc, supplier\_id (FK), cost, markup, dept\_cd ]

supplier [ supplier\_id, supplier\_name ]

**Merged 3NF solution:**

product [ prod\_code, prod\_desc, um, dept\_cd (FK), supplier\_id (FK), cost, markup ]

dept [ dept\_cd, dept\_name, aisle\_no ]

supplier [ supplier\_id, supplier\_name ]

**Example Solution 2:**

UNF:

dept [dept, aisle\_no (prod\_code, , price, um) ]   
1NF:

dept [dept\_id, dept\_name, aisle\_no]

dept\_product [dept, prod\_code, price, um ]   
2NF:

dept [dept\_id, dept\_name, aisle\_no ]

product [prod\_code, price, um, dept\_id (FK) ]  
  
3NF:

**Notes**: **1.** examing the relationship between department and product, we discover that it is a 1:M, therefore we do not need the composite table, dept\_product. So it is eliminated and the foreign key placed in the product table. **2**. it may be debatable whether department actually determines aisle number. This may be true in a small grocery store but in a large grocery store, products from a department may be found in multiple aisles. This should be discussed with your database client to determine their exact needs.

UNF:

supplier [supplier\_id, supplier\_name, (prod\_code, prod\_desc, cost, markup, dept\_cd ) ]

1NF:

supplier [supplier\_id, supplier\_name]

supplier\_product [supplier\_id, prod\_code, prod\_desc, cost, markup, dept\_cd ]

2NF:

supplier [supplier\_id, supplier\_name]

supplier\_product [supplier\_id, prod\_code]

product [prod\_code, prod\_desc, cost, markup, dept\_cd ]

note: if we were getting a product from more than 1 supplier, then the cost attribute would go ino the supplier\_product table.

3NF:

supplier [supplier\_id, supplier\_name]

product [prod\_code, prod\_desc, cost, markup, dept\_cd, supplier\_id (FK) ]

**note**: examing the relationship between supplier and product, we discover that it is a 1:M, therefore we do not need the composite table, supplier\_product. So it is eliminated and the foreign key placed in the product table.

**Merged 3NF solution:**

product [ prod\_code, prod\_desc, um, dept\_cd (FK), supplier\_id (FK), cost, markup ]

dept [ dept\_cd, dept\_name, aisle\_no ]

supplier [ supplier\_id, supplier\_name ]

**NOTE: Even with two different approaches to this question, we end up with the same 3NF solution.**

**Question 6:**

**For the following problems, identify what normal form the table is in: 0NF, 1NF, 2NF or 3NF. If it’s not in 3NF, bring it up to 3NF.**

**Table 1:**

Ship(BoatId, NbrOfPools, YearMade, Tonage, CompanyId, CompanyName)

The only functional dependencies for this table are:   
BoatId --> NbrOfPools, YearMade, Tonage, CompanyId, CompanyName   
CompanyId --> CompanyName   
CompanyName --> CompanyId

**Solution:**

2NF

Ship(BoatId, NbrOfPools, YearMade, Tonage, CompanyId)

Company(CompanyId, CompanyName)

**Table 2:**

Inventory(PartNbr, {Warehouse, Location}, QOH, Weight, PartColor)

PartNbr --> Weight, PartColor   
PartNbr + Warehouse --> QOH     QOH is Quantity On hand   
Warehouse --> Location

**Sample Data**   
**PartNbr  Warehouse Location  QOH  Weight  PartColor**   
01       500       NW        135   11.75  Blue   
01       600       SW        210   11.75  Blue   
01       800       East      192   11.75  Blue   
02       500       NW         75    2.50  Red   
02       800       East       45    2.50  Red   
03       500       NW        290   21.35  Green   
03       600       SW         83   21.35  Green

**Solution:**

0NF

Parts(PartNbr, Weight, PartColor)

InStock(PartNbr, Warehouse, QOH)

Warehouse(Warehouse, Location)

Another layout for the Part data:

**PartNbr  Weight  PartColor Warehouse Location  QOH**   
01        11.75  Blue      500       NW        135   
                           600       SW        210   
                           800       East      192   
02         2.50  Red       500       NW         75   
                           800       East       45   
03        21.35  Green     500       NW        290   
                           600       SW         83

This format makes it easier to see the repeating groups.

**Table 3:**

Hospital(Patient, Insurance, Doctor, {Test, Result})

Patient --> Insurance, Doctor   
Patient + Test --> Result

**Sample Data**   
**Patient    Insurance  Doctor     Test        Result**   
Tweety     Red Cross  Livingston Brain Scan  Not Found   
Tweety     Red Cross  Livingston Blood work  Yes and red   
Sylvester  Red Shield Kilder     Cat Scan    Yes he is a Cat   
Sylvester  Red Shield Kilder     X Rays      No broken bones   
Sylvester  Red Shield Kilder     Flea check  None

**Solution:**

0NF

Patient(Patient, Insurance, Doctor)

Test(Patient, Test, Result)

**Table 4:**

Cars(Vin, Make, Style, Price, Year, Color)

Vin --> Make, Style, Price, Year, Color

**Sample Data**   
Vin          Make   Style      Price Year  Color   
1FD56GH8901X Ford   SUV       55,000 1965  Red   
1HC21KQ0733G Chevy  Sports    18,200 1998  Green   
3AS88JV9203K Dodge  Truck     13,200 1997  Silver   
4PL72XD5431M Ford   Truck      5,500 1996  Blue   
8BE24RU7385Y Ford   Sedan      5,500 1996  Blue   
7JU69TD2485R Dodge  SUV       18,200 1998  Black   
0LF83SA6540Z Dodge  Sedan        500 1990  Rust

**Solution:**

3NF

Sources:

<http://www.gc.maricopa.edu/business/sylvester/cis164/norm-pr1.htm>

<http://www.cs.rpi.edu/~sibel/dbs/FALL2002/notes/normalizationqa.pdf>

<https://scs.senecac.on.ca/~dbs201/pages/Normalization_Practice.htm>

Our book