<http://jcsites.juniata.edu/faculty/rhodes/dbms/ermapping.html>

**COMPLAINTS**

COMPLAINTS(ID, UID, Text, Status, Filed-date-time, Handled-date-time, Employee ID,)

| ID | Text | Status | Filed-date-time | Handled-date-time | Employee-ID | UID |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Product works! | In-progress | 1-1-2020 | 2-1-2020 | 1 | 100 |
| 2 | Shop sucks | Resolved | 1-1-2020 | 2-1-2020 | 1 | 101 |

Non-Trivial FDs:

* ID -> Text, Status, Filed-date-time, Handled-date-time, Employee ID, UID
* {UID, Filed-date-time} -> ID

Closure Computation:

* {ID}+ ={ID, Text, Status, Filed-date-time, Handled-date-time, Employee ID, UID}
* {Filed-date-time}+ = {Filed-date-time}
* {Filed-date-time}+ = {Filed-date-time}
* {Handled-date-time}+ = {Handled-date-time}
* {Employee ID}+ = {Employee ID}
* {UID}+ = {UID}

Keys: ID, {UID, Filed-date-time}

The LHS of the non-trivial FDs is a key, So the relation is in 3NF.

*Assumptions:*

* May have same text from different complaints at different shop
* Different user can file complaints at the same time and be handled by different employees at the same time
* Assume that UID and Filed-date-time can determine the ID because normally we only can file one complaint at a time. And because ID is a key, UID and Filed-date-time which can determine ID should also be a key.
* Assume that one employee can handle multiple complaints at one time?????

***EXTRA------------------------------------------------------------------------------------***

\*\*\*\*(If assume one employee can only handle one complaint at a time(will not have same handled-date-time for one employee),then can put EmployeeID, handled-date-time, filed-date-time -> status into FDs and there will be 3NF violation,it can be done lah but a bit troublesome)

(to add or not to add????)

**--------------------------------------------------------------------------------------**

EmployeeID, handled-date-time, filed-date-time -> status

The LHS of this FD is not a key, and the RHS of this FD does not contain an attribute of the key, thus this FD violates 3NF.

***3NF Decomposition***

**Finding minimal basis:**

Step of finding minimal basis:

1. Make sure RHS of the FDs only have one attribute
2. Remove redundant FDs (*FD cannot derive from other FDs*)
3. Remove redundant FDs from the LHS (*not redundants means if remove one of the attribute from LHS then it will create a new FDs*)

1.

* ID -> Text
* ID -> Status
* ID-> Filed-date-time
* ID -> Handled-date-time
* ID -> Employee ID
* ID -> UID
* {UID, Filled-date-time} -> ID
* EmployeeID, Handled-date-time, Filed-date-time -> Status

2. No redundant FDs is found

3. No redundant FDs from LHS is found.

So minimal basis is:

* ID -> ID
* ID -> Text
* ID -> Status
* ID-> Filed-date-time
* ID -> Handled-date-time
* ID -> Employee ID
* ID -> UID
* {UID, Filled-date-time} -> ID
* EmployeeID, Handled-date-time, Filed-date-time -> Status

**Decomposition:**

Step I: Combine those FDs with same LHS

Result:

ID -> Text, Status, Filed-date-time, Handled-date-time, Employee ID

{UID, Filled-date-time} -> ID

EmployeeID, Handled-date-time, Filed-date-time -> Status

Step II:For each FDs create a relation schema that contains all the attributes in the FDs

Result:

R1(ID, Text, Status, Filed-date-time, Handled-date-time, Employee ID)

R2(UID, Filled-date-time, ID)

R3(EmployeeID, Handled-date-time, Filed-date-time, Status)

Step III: If no table contain a key of the original table, add a table that contains a key of the original table

Result:

R1(ID, Text, Status, Filed-date-time, Handled-date-time, Employee ID)

R2(UID, Filled-date-time, ID)

R3(EmployeeID, Handled-date-time, Filed-date-time, Status)

R4(ID,EmployeeID, Handled-date-time, Filed-date-time)

Step IV: Remove redundant tables (if any)

Final Result:

R1(ID, Text, Status, Filed-date-time, Handled-date-time, Employee ID)

R2(UID, Filled-date-time, ID)

R3(EmployeeID, Handled-date-time, Filed-date-time, Status)

R4(ID,EmployeeID, Handled-date-time, Filed-date-time)

\*\*\*If like that need to code 2 more table lah i think,might make things a little bit more complicated

***EXTRA------------------------------------------------------------------------------------------------***

**COMPLAINTS-ON-SHOPS**

COMPLAINTS-ON-SHOPS(ID, Sname, )

Keys: ID

Primary Keys: ID

Non-trivial FDs:

* ID -> Sname

The LHS of the non-trivial FDs is a key, So the relation is in 3NF.

(Two-attribute relation is always in BCNF, which is also in 3NF)

**COMPLAINTS-ON-ORDERS**

COMPLAINTS-ON-ORDERS(ID, OID)

Keys: ID

Primary Keys: ID

Non-trivial FDs:

* ID -> OID

The LHS of the non-trivial FDs is a key, So the relation is in 3NF.

(Two-attribute relation is always in BCNF, which is also in 3NF)

**EMPLOYEE**

EMPLOYEE(ID, Name, Salary)

Functional dependencies:

1. ID → Name, Salary

Closure computation:

{ID}+ = {ID, Name, Salary}

{Name}+ = {Name}

{Salary}+ = {Salary}

Key: ID

Checking if the relation is in 3NF:

A relation is in 3NF if for every non-trivial FDs:

1. LHS contains a key
2. Each attribute in RHS is contained in a key

The functional dependency ID → Name, Salary does not violate 3NF. LHS contains a key.

Therefore, the relation is in 3NF.

**PRODUCTS**

PRODUCTS(PName, Maker, Category)

Assumption:

1. Assume same product made by different maker has different names, phones made by Samsung is Galaxy X, phones made by Apple is IPhone

Functional dependencies:

1. PName → Maker, Category

Closure computation:

{PName}+ = {PName, Maker, Category}

{Maker}+ = {Maker}

{Category}+ = {Category}

Key: PName

Checking if the relation is in 3NF:

A relation is in 3NF if for every non-trivial FDs:

1. LHS contains a key
2. Each attribute in RHS is contained in a key

The functional dependency PName → Maker, Category does not violate 3NF. LHS contains a key.

Therefore, the relation is in 3NF.