Example: Summarize the Normal Forms by FDs

CS 4750
Database Systems

[C.M. Ricardo and S.D. Urban, "Database Illuminated," Ch.6]

Example

Let's consider the `work_project` table that stores information about projects, assignment, and budget and hour allocation

`work_project`

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima	Lee	E110	30	Rivera	200000	03/15/18	48000	Levine	10	
Maxima	Lee	E120	15	Tanaka	200000	03/15/18	45000	Jones	15	

Notice that the table consists of many attributes, some data are redundant.

Assumptions We Made

- Each project has a unique name
- Names of employees are managers are not unique
- Each project has one manager, whose name is stored in `prjMgr`
- Many employees can be assigned to work on each project
- An employee can be assigned to more than one project
- The attribute `hours` tell the number of hours per week a particular employee is assigned to work on a particular project
- `budget` stores the amount budgeted for a project
- `startDate` gives the starting date for a project
- `salary` gives the annual salary of an employee

Assumptions We Made (2)

- `empMgr` gives the name of the employee's manager, who might not be the same as the project manager
- `empDept` gives the employee's department
- Department names are unique
- Each department has only one manager
- The employee's manager is the manager of the employee's department
- `rating` gives the employee's rating for a particular project
- The project manager assigns the rating at the end of the employee's work on that project

List of FDs

Based on the assumptions, we can list the following FDs

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Since we assumed people's names were not unique, `empMgr` does not functionally determine `empDept`.

`prjMgr` does not determine `prjName`.

Is work_project Table in 1NF?

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima	Lee	E110	30	Rivera	200000	03/15/18	48000	Levine	10	
Maxima	Lee	E120	15	Tanaka	200000	03/15/18	45000	Jones	15	

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Given FDs

Check if there is a column that is not atomic.

With a composite key (`prjName, empId`), each column would be single valued. Thus, the work_project table is in 1NF

Is work_project Table in 2NF?

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima	Lee	E110	30	Rivera	200000	03/15/18	48000	Levine	10	
Maxima	Lee	E120	15	Tanaka	200000	03/15/18	45000	Jones	15	

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Given FDs

Check if there is a partial dependency.

The work_project has partial dependencies, violating 2NF

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
```

Use 2NF, Decompose work_project

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima	Lee	E110	30	Rivera	200000	03/15/18	48000	Levine	10	
Maxima	Lee	E120	15	Tanaka	200000	03/15/18	45000	Jones	15	

The work_project has partial dependencies, violating 2NF

prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept

Thus, transform the work_project table into an equivalent set of 2NF relations by projection, resulting in

Project(prjName, prjMgr, budget, startDate)
Emp(empId, empName, salary, empMgr, empDept)
Work(prjName, empID, hours, rating)

Is work_project Table in 3NF?

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima	Lee	E110	30	Rivera	200000	03/15/18	48000	Levine	10	
Maxima	Lee	E120	15	Tanaka	200000	03/15/18	45000	Jones	15	

2NF

```
Project(prjName, prjMgr, budget, startDate)
Emp(empId, empName, salary, empMgr, empDept)
Work(prjName, empID, hours, rating)
```

Use the set of tables `Project`, `Emp`, and `Work`. Check if there is a non-key attribute or combination of attribute that functionally determines another non-key attribute.

Check if there is a transitive dependency.

Is work_project Table in 3NF?

2NF

```
Project(prjName, prjMgr, budget, startDate)
Emp(empId, empName, salary, empMgr, empDept)
Work(prjName, empID, hours, rating)
```

Given FDs

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Check if there is a transitive dependency.

`Project`: No transitive dependency in `Project`. Thus, it is in 3NF.

`Emp`: There is a transitive dependency since `empDept` → `empMgr`.

Since `empDept` is not a superkey, nor is `empMgr` part of a candidate key, this violates 3NF. Thus, we need to decompose `Emp`

```
Emp(empId, empName, salary, empDept)
Department(empDept, empMgr)
```

`Work`: No transitive dependency in `Work`. Thus, it is in 3NF.

Use 3NF, Decompose work_project

2NF

```
Project(prjName, prjMgr, budget, startDate)
Emp(empId, empName, salary, empMgr, empDept)
Work(prjName, empID, hours, rating)
```

Given FDs

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Therefore, the new set of 3NF tables is

```
Project(prjName, prjMgr, budget, startDate)
Emp(empId, empName, salary, empDept)
Department(empDept, empMgr)
Work(prjName, empID, hours, rating)
```

Sample Data

Emp

empId	eName	salary	empDept
E101	Jones	60000	10
E105	Adams	55000	12
E110	Rivera	48000	10
E120	Tanaka	45000	15

Project

prjName	prjMgr	budget	startDate
Jupiter	Smith	100000	01/15/19
Maxima	Lee	200000	03/15/18

Work

prjName	empId	hours	rating
Jupiter	E101	25	9
Jupiter	E105	40	
Jupiter	E110	10	8
Maxima	E101	15	
Maxima	E110	30	
Maxima	E120	15	

Department

empDept	empMgr
10	Levine
12	Jones
15	Jones

Is work_project Table in BCNF?

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima										
Maxima										

```
prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr
```

Given FDs

Check if there is a determinant that is not a superkey.

Any one of `empId`, `empDept`, or `prjName` is sufficient to show that the `work_project` table is not BCNF. Thus, normalize it.

Use BCNF, Decompose work_project

work_project

prjName	prjMgr	empId	hours	eName	budget	startDate	salary	empMgr	empDept	rating
Jupiter	Smith	E101	25	Jones	100000	01/15/19	60000	Levine	10	9
Jupiter	Smith	E105	40	Adams	100000	01/15/19	55000	Jones	12	
Jupiter	Smith	E110	10	Rivera	100000	01/15/19	48000	Levine	10	8
Maxima	Lee	E101	15	Jones	200000	03/15/18	60000	Levine	10	
Maxima										
Maxima										

prjName → prjMrg, budget, startDate
empId → empName, salary, empMgr, empDept
prjName, empId → hours, rating
empDept → empMgr

Given FDs

Therefore, the new set of BCNF tables is

Project(<u>prjName</u>, prjMgr, budget, startDate)
Emp(<u>empId</u>, empName, salary, empDept)
Department(<u>empDept</u>, empMgr)
Work(<u>prjName</u>, <u>empID</u>, hours, rating)

Consider each table. For every nontrivial FD, a determinant determines everything and thus it is a superkey.

Take Away

- Be able to identify reasonable / applicable FDs
- Understand the characteristics of each normal form
- Given a set of tables, you should be able to analyze and determine whether the given tables are in a certain normal form.
- If they are not, which tables (or part of the tables) violate the properties. You should be able to fix all the violations such that the set of tables are in the desired normal form.
- If you choose to use 3NF, all tables must be in 3NF.
- If you choose to use BCNF, all tables must be in BCNF.