Database Normalization

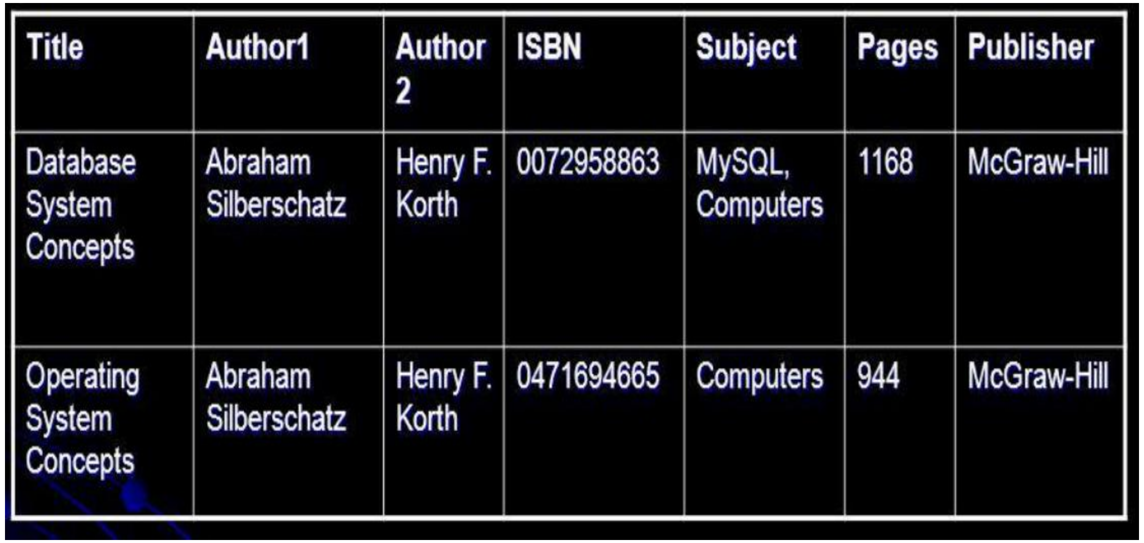
Amir Dirin 2022

Normalization

- Database normalization is the process of removing redundant data from your tables to improve storage efficiency, data integrity, and scalability.
- In the relational model, a method exists for quantifying how efficient a database is. These classifications are called normal forms (or NF), and there are algorithms for converting a given database between them.

- Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued.

Table 1



Title	Author1	Author 2	ISBN	Subject	Pages	Publisher
Database System Concepts	Abraham Silberschatz	Henry F. Korth	0072958863	MySQL, Computers	1168	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	Henry F. Korth	0471694665	Computers	944	McGraw-Hill

- This table is not very efficient with storage
- This design does not protect data integrity
- Third, this table does not scale well.

First Normal Form

- There are two violations of the first normal form:
    - The subject field contains more than one piece of information with more than one value in a single field, it would be very difficult to search for all books on a given subjct

Exercise

Student_ID	Name	Subject	HoD	Office No.
1	Timo	ICT	Simo	0934945
2	Satu	ICT	Simo	0934945
3	Mika	ICT	Simo	0934945
4	Outi	ICT	Simo	0934945

Three major problem with this table

1. Insertion anomaly
2. Deletion anomaly
3. Updating anomaly

Insertion anommaly

Student_ID	Name	Subject	HoD	Office No.
1	Timo	ICT	Simo	0934945
2	Satu	ICT	Simo	0934945
3	Mika	ICT	Simo	0934945
4	Outi	ICT	Simo	0934945
5	**Matti**	**ICT**	**Simo**	**0934945**

Imagine if we would have added 100 more students, we must insert redundant data for every row.

Delete anomally

Student_ID	Name	Subject	HoD	Office No.

Here we not only remove the student we also remove the details about the subject information, hod, office no.

Update Modification anomaly

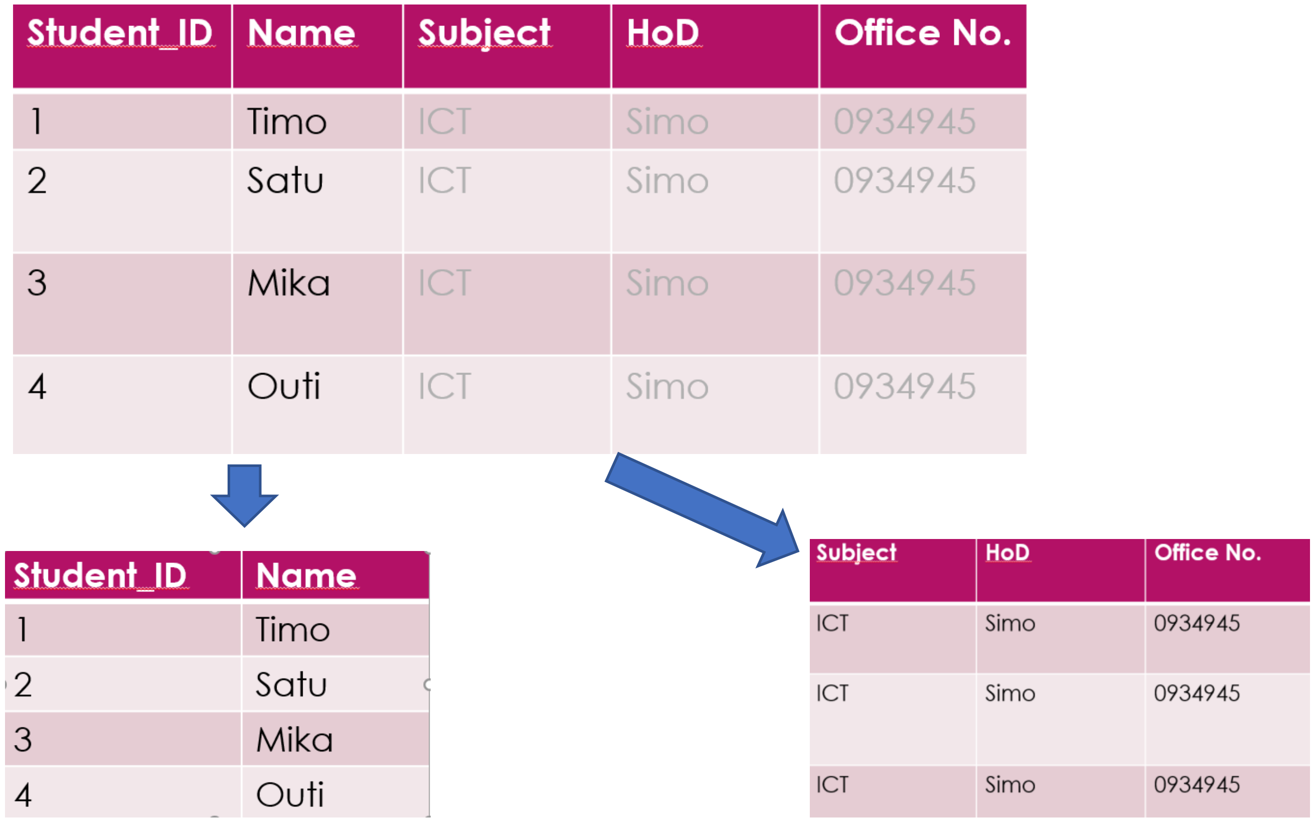
Student_ID	Name	Subject	HoD	Office No.
4	Timo	ICT	~~Simo~~ Vesa	0934945
2	Satu	ICT	~~Simo~~ Vesa	0934945
1	Mika	ICT	~~Simo~~ Vesa	0934945
3	Outi	ICT	~~Simo~~ Vesa	0934945

In case we need to change one field we have to update the entire table.

- In our table the subject information is common for all students. Therefore, the subject-related field can be separated from the student’s name table.

Questions

- How Normalization would solve this problem?
- Divide the table into two tables



Student_ID	Name	Subject	HoD	Office No.
1	Timo	ICT	Simo	0934945
2	Satu	ICT	Simo	0934945
3	Mika	ICT	Simo	0934945
4	Outi	ICT	Simo	0934945

Student_ID	Name
1	Timo
2	Satu
3	Mika
4	Outi

Subject	HoD	Office No.
ICT	Simo	0934945
ICT	Simo	0934945
ICT	Simo	0934945

- So for our table the subject table would have contain only one row and that is

Student_ID	Name	Subject
1	Timo	ICT
2	Satu	ICT
3	Mika	ICT
4	Outi	ICT

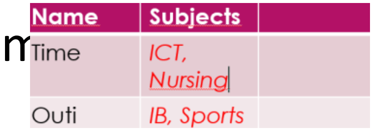
Subject	HoD	Office No.
ICT	Simo	0934945

Normalization is about minimizing data redundancy.
Here only the subject has been repeated, any changes into a subject table impact all student.

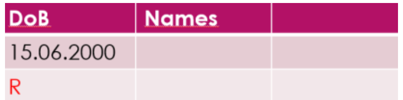
In this solution, we do not have problems such as
Insert, delete, update.

Normalization techniques.

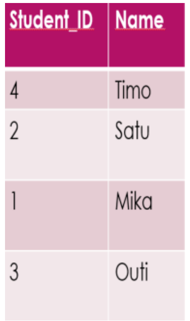
- Normalization process:
  - Scalable table design which can be easily extended
  - If the table is not even the first data normal forms then that is not a good database design
  - 1N rule
    1. Each column should contain atomic values
    2. In each column the value must be the sam
    3. Each column must have a unique name
    4. Order in which data is saved does not matter



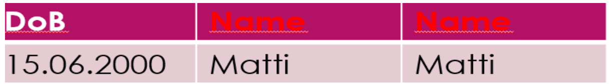
Name	Subjects	
Time	*ICT,* *Nursing*	
Outi	*IB, Sports*	



DoB	Names	
15.06.2000		
R		



Student_ID	Name
4	Timo
2	Satu
1	Mika
3	Outi



DoB	Name	Name
15.06.2000	Matti	Matti

Exercise

- Is this a first N table based on the rules defined in the previous slides?

Student_ID	Name	Subject
4	Timo	OS, DB
2	Satu	Java
1	Mika	C,C++
3	Outi	ICT

No! rule number1 is broken since the subject column does not contain an atomic value

How to solve the problem?

- Make the subject values atomic

Student_ID	Name	Subject
4	Timo	OS
2	Satu	Java
1	Mika	C
3	Outi	ICT
5	**Timo**	**DB**
6	**Mika**	**C++**

Does this a 1N table?
Check the rule!

2N

- For the table to be in second normal form it must satisfy two conditions
  - It should be in 1st Normal form
  - Should not be partial dependency in the table
  - Dependency VS Partial-dependency
    - In the example with student-ID we are able to get all the relevant data from the table e.g,
      - Give the major for student 2.

Student Table

pk

student_ID	Name	Reg-no	Major	Address
1	Timo	CSE-18	CSE	TN
2	Timo	IT-18	IT	AP
3	Matti	CSE-18	CSE	HR
4	Satu	CSE-18	CSE	MH
5				

- What about partial-dependency

Score Table

sore_ID	student-ID	subject-ID	marks	
1	1	1	82	
2	1	2	77	
3	2	1	85	
4	3	3	82	

If I want to get the student's number 1 score can we get it?

Subject Table

subject_ID	subject_Name	teacher

Student_ID	Name	Reg-no	Major	address

No, Why? Since it will be difficult to know which students number 1 we are referring to? Therefore we need student-ID and subject_ID together to make it unique for identifying the data. Here teachers are not dependent on the students so it is partially dependent on the subject. In second N this should not exist

- So we have to remove the teacher’s name from the score table because it is partially dependent we can do this in many different ways
  - 1. move the teachers’ names to the Subject table
  - 2. Add a new table for teachers and use the teacher-ID wherever we want

3Nf

- In the score table the exam name and total marks are missing

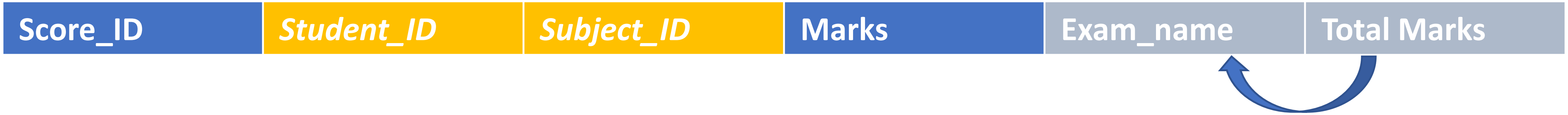
totalMarks depends to the examName, e.g, KMM vs. Math which total marks does not depend on primery keys

sore_ID	student-ID	subject-ID	marks	teacher	ExamName	TotalMarks
1	1	1	82	Amir		
2	1	2	77	Simo		
3	2	1	85	Vesa		
4	3	3	82	Auvo		

For a table to be in 3rd Normal form:

1. It has to be in 2nd Normal Form
2. And it should be *transitive dependency*

Transitive dependency



Score_ID	*Student_ID*	*Subject_ID*	Marks	Exam_name	Total Marks

The primary key for the score table is a composite key (students_ID and subject_ID).

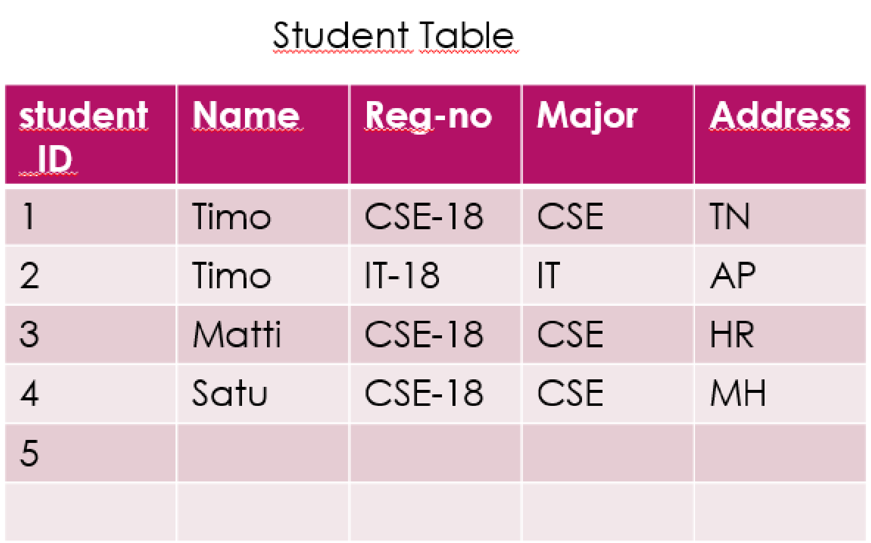
The exam_Name now depends on students_ID and subject_ID. For example, for some courses (Java) we have a practical exam and for some e,g, Math we do not. So we can not say that the exam name depends on the student's name and subject both.

What about the total_marks, does it depend on the student or subject?Total_marks depends on the exam name. Like the exam, names define how the points are calculated for the total, like lecture 40 points, assignment 10 points exam 50 points for DB course.

But for Math written exam gives the total marks.

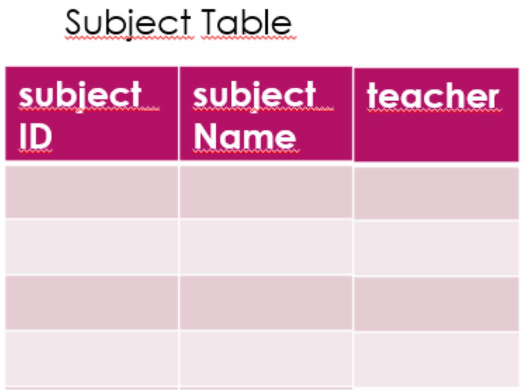
So in the table, the total marks and exam are related this is called transitive dependency. The solution is that the Exam name and total_marks must be in the new table.

Solution for transitive dependency



Student Table

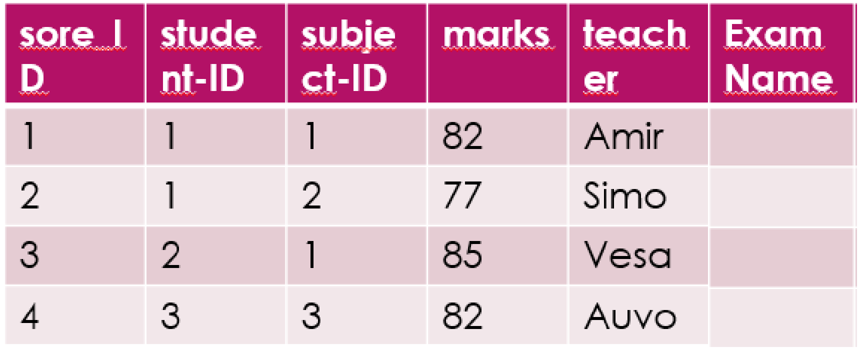
student_ID	Name	Reg-no	Major	Address
1	Timo	CSE-18	CSE	TN
2	Timo	IT-18	IT	AP
3	Matti	CSE-18	CSE	HR
4	Satu	CSE-18	CSE	MH
5				



Subject Table

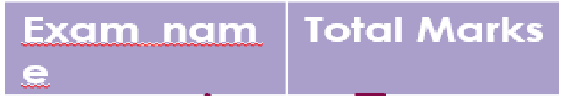
subject_ID	subject_Name	teacher

Score Table



sore_ID	student-ID	subject-ID	marks	teacher	Exam Name
1	1	1	82	Amir	
2	1	2	77	Simo	
3	2	1	85	Vesa	
4	3	3	82	Auvo	

Exam Table



Exam_name	Total Marks

Boyce-Codd Normal Form (3.5N)

- The table should be in 2N form
- For any dependency A—>B then A should be a super key
  - Or A not be a prime attribute while B is a prime attribute; meaning a non. prime attribute drives the prime attribute which BCNF does not allow this type of dependency

Student_ID	Subject	teacher
101	Java	Amir
101	C++	Simo
102	Java	Vesa
103	C#	Hannu

One student may participate in multiple courses which has a different teacher. Multiple teachers teach one subject e,g. java

Here we can use the teacher’s name to find the subject as well. The subject is a primery key.

Solution BCNF

Student Table

Student_ID	Teacher_ID

Teacher Table

Teacher_ID	Teacher	subject

4N

- It should satisfy BCNF
- IT should not have
  - Multi-Valued dependency
- In 2NF we removed ***partial dependency*** and,
- In 3NF we removed ***transitive dependency***.
- In 4NF ***Multi-valued dependency***

A—>B, is a multi-valued dependency
if A1 depends on B1 and B2.

A table should have at least 3 columns to have a multi-valued dependency

X

OK

B and C should be independent from each other

Example

Student_ID	Course	hobby
1	Physic	Aikido
1	Math	Ski
2	C#	football

There is no relation between course and hobby of students so it is advised it should be in a separate table of the multi-value dependency.

Student and hobby table

Student_ID	Course

Student_ID	hobby

5N(PJNF)-Project Join Normal Form

- It should be in 4NF
- It should not have *Join Dependency*
  - *If so then we can make it to a small relation*

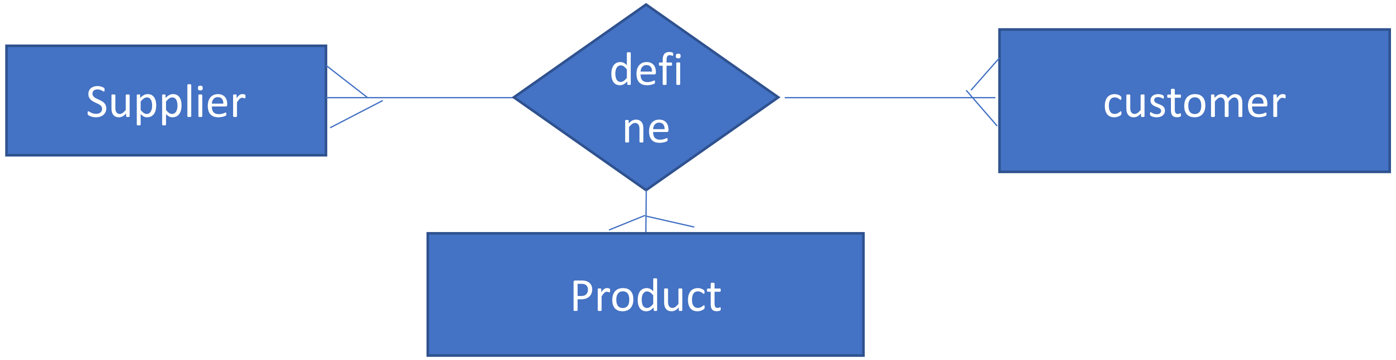
Supplier	Product	customer

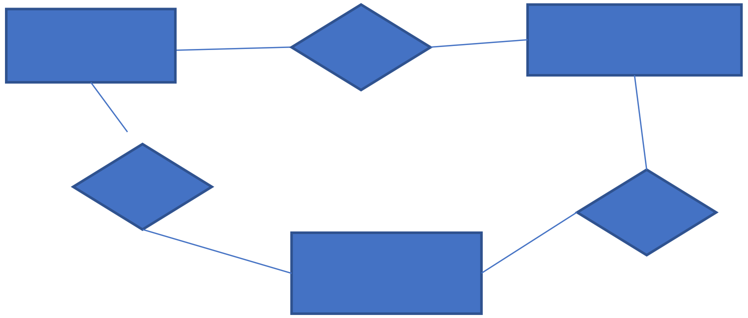
Supplier	product

Supplied	Product	customer
ABC	X1	Giganti
ABC	Y2	Ikea
DFCH	M25	metropolia

Supplier	customer

Customer	product





- In this table solution some data are missed for example selling the X1 by ABC to Giganti. The 5N says if you lose the joint to the data then you should not decompose the data.

Exercise 1

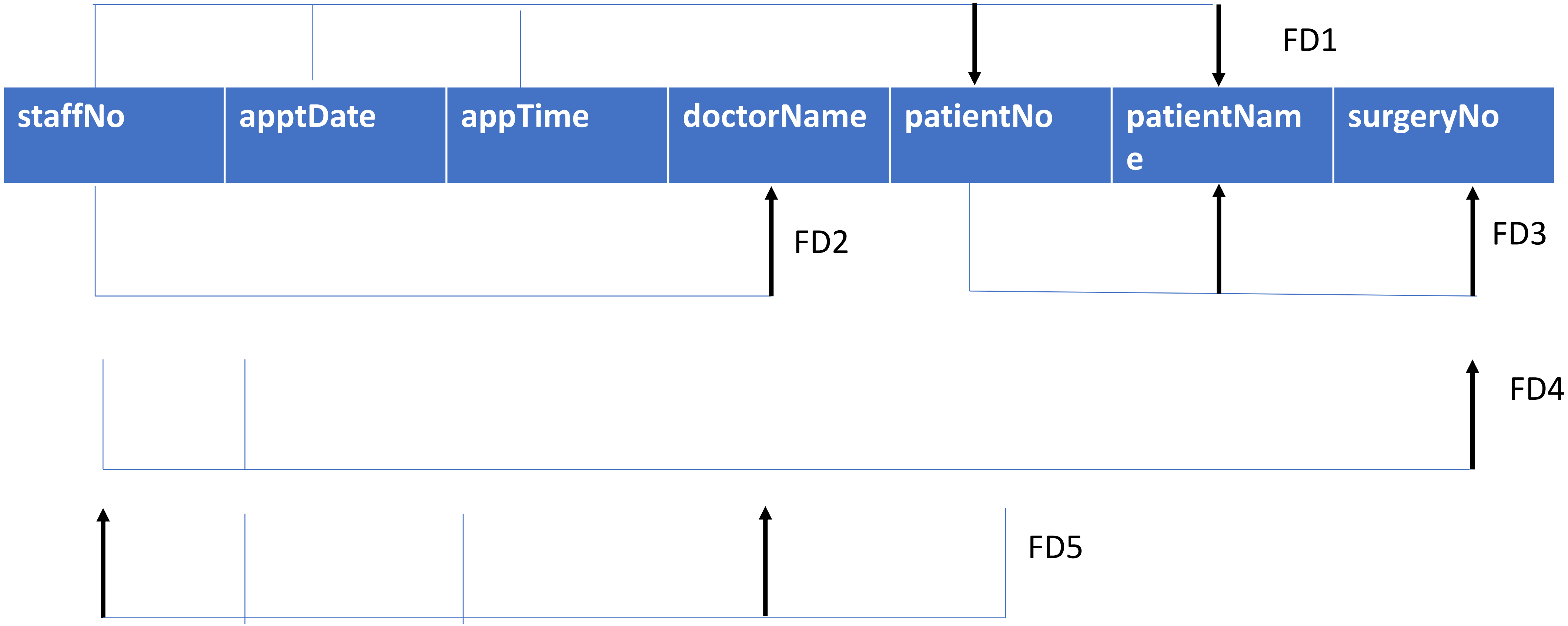
staffNo	doctorName	patientNo	patientName	appointmet		surgeryNo
				date	time	
S101	Timo O	P100	Amir D	12.08	10:00	S10
S102	Outi K	P106	Marko H	23.08	14:00	S15
S103	Robin D	P200	Amir D	14.09	16:00	S30
S104	Magnus H	P201	Marko H	15.10	12:00	S13

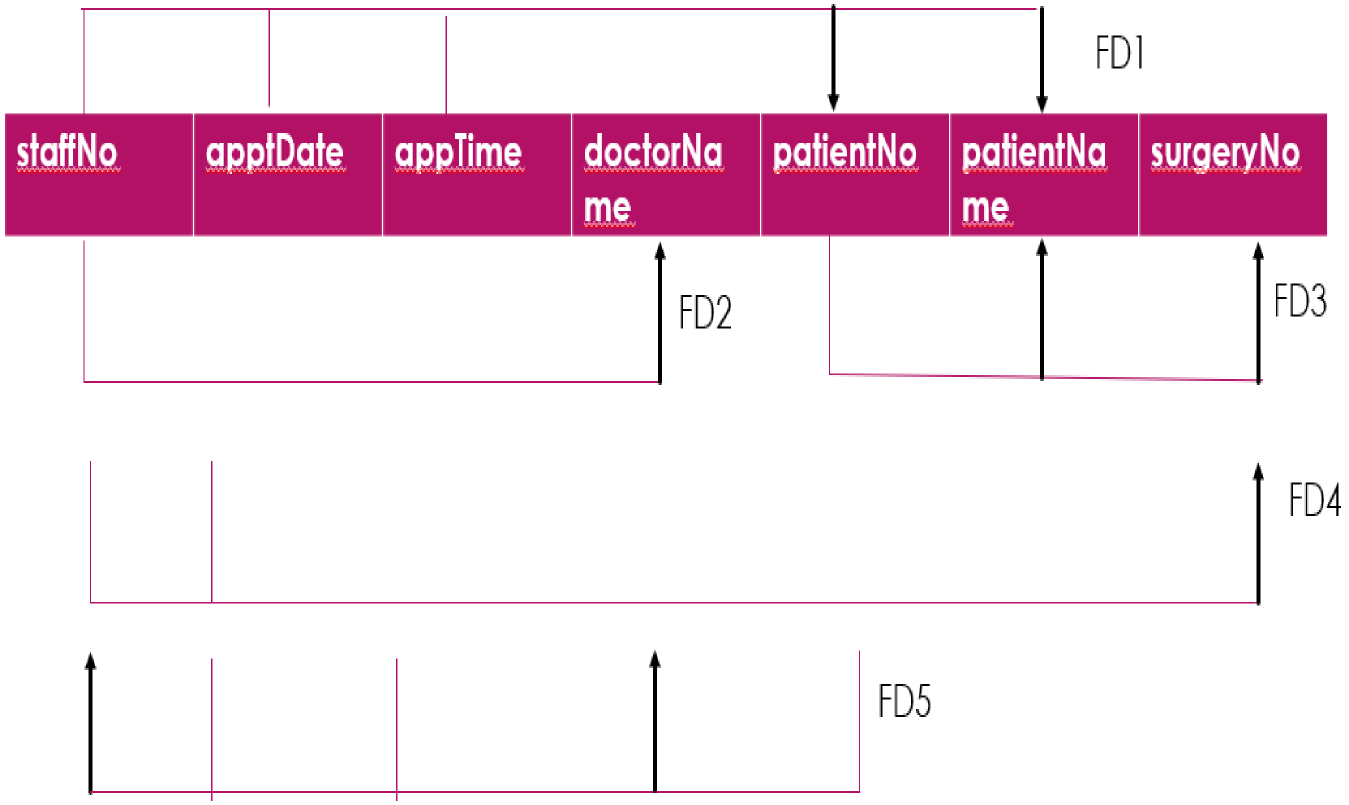
Make sure the table is normalized

- This not normalized since in the appointment column there are multiple values that violate the 1NF.
- Considering the staff and pationetNo as candidate keys there are many anomalies that exist.

  1. Insertion anomalies:
     - To insert a new patient to make an appointment with doctor we need to enter the correct detail for the staff, e.g., to add a new patient we need to add patientNo, patientName and am an appointment, we must enter the corrrect details of the doctor (staffNo, doctorName) so the detail consistent with values for the designated doctor.
  2. Deletion anomalies:
     - If we want to delete a patient named Amir D for example, two records need to be deleted. This anomaly is also obvious when we want to delete the doctor, multiple recodes needs to be deleted to maintain the data integrity. When we delete a doctor recod for exam Magnus H about his patients are also lost from the database
  3. Modification anolamalies
     - With redundant data, when we change the value of one column of a doctor, doctorName we must update all the Doctor records that assign to a particula patient otherwise the data would inconsistent. We also need to to modify the appointmet schedules because differe Doctor has different schedules.

- Let’s assume that a patient is registered at only one surgery and he or she has more than one appointment on a given day. All the schedules have been fixed for the whole days and week.
- In 1NF
  - We remove all the repeating group’s appointment and assign new column(appDate and appTime) and assigned primary keys (candidate keys) Then think about functional dependency


FD1
staffNo
apptDate
appTime
doctorName
patientNo
patientNam
e
surgeryNo
FD2
FD3
FD4
FD5



FD1 is already 2NF which depends

staffNo	doctorName

*Doctor (**staffNo**,doctorName)*

FK		
staffNo	appDate	surgeryNo

Surgery (**staffNo**, appDate, surgeryNo)

patientNo	patientName

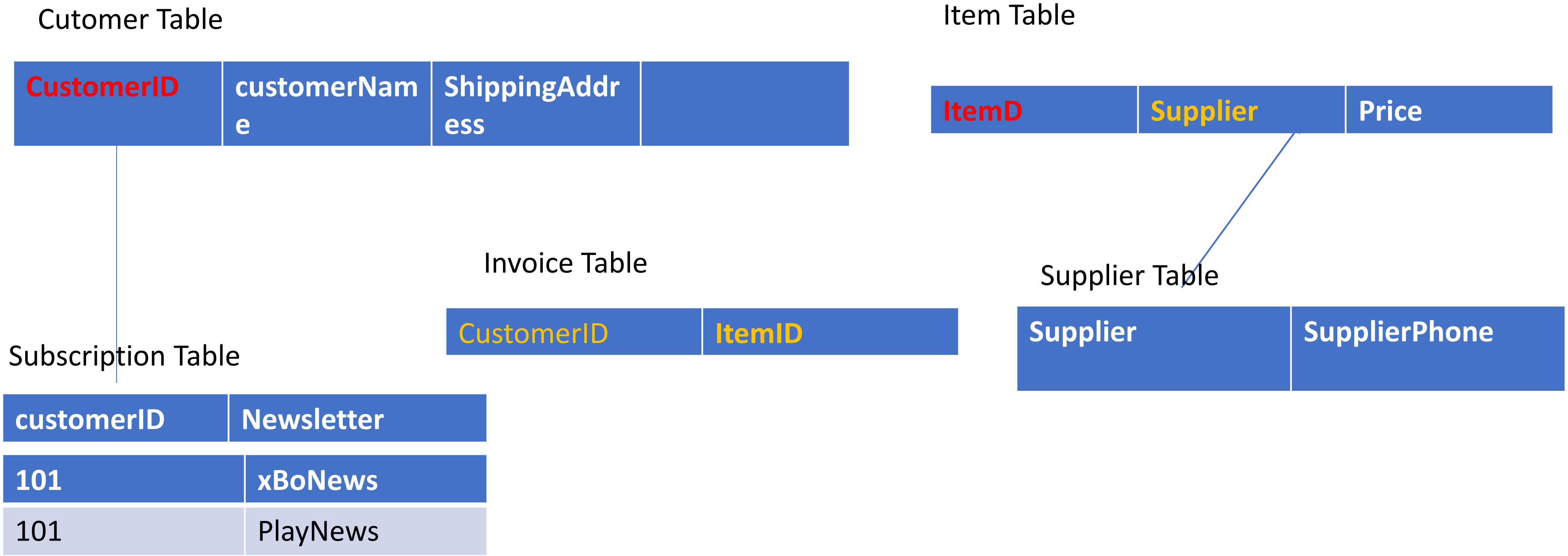
Patient (**patientNo**, patientName)

			FK
staffNo	appDate	appTime	patientNo

*Appointment (**staffNo,appDate, appTime**, patientNo)*

Exercise 2

Customer Name	Item	ShippingAddress	NewsLetter	supplier	supplierPhone	price
Mikko	xBox	Ahventie, Vaasa	XboxNews	Microsoft	0800112	250
Outi	PlayStation	Oikotie, TRE	xboxNews, PlayStation	tukku	080012	300
Timo	PSP, xBOX	Ahventie, Vaasa	Play Station	sony	080012	450



Cutomer Table

CustomerID	customerName	ShippingAddress	

Item Table

ItemD	Supplier	Price

Invoice Table

CustomerID	ItemID

Supplier Table

Supplier	SupplierPhone

Subscription Table

customerID	Newsletter
101	xBoNews
101	PlayNews

- 3N