Homework #7 S14
Functional Dependencies and Normalization, CSC3287 Database Systems Concepts

Due by email in pdf format to debra.parcheta@ucdenver.edu by the beginning of class on Thursday, April 17th, 2014

At an orthodontist's office, an engineer from the NeverStudied School of Database Design created a database that is not working very well. Each time a patient comes in to one of their offices (they have several offices to choose from), the staff must be able to record the treatment given and the costs for billing. Given the engineer's schema and functional dependencies shown below:

1. determine if the relation is 1NF. (If it is, tell how you know. If not put it in 1NF form.)
2. then, determine if the relation is 2NF. (If it is, tell how you know. If not put it in 2NF form.)
3. then, determine if the relation is 3NF. (If it is, tell how you know. If not put it in 3NF form.)
4. What is the key of the original ORTHODONTIST_VISIT table?

ORTHODONTIC_VISITS

Doc#	DocName	Office#	TreatmentGiven	OAddress	PName	PAddress	TCost	Treatment#	Patient#
ApptDate	ApptTime	DocDeg							

Office#→{Doc#, OAddress}
Patient# →{PName}
Doc# →{DocName, DocDeg}
Treatment# →{TCost, TreatmentGiven}
{Office#, Treatment#, Patient#} →{ApptDate, ApptTime}
TCost is a composite attribute that accounts for LaborCosts and SuppliesCosts.
Visit is a multivalued, composite attribute that consists of the atoms ApptDate, ApptTime, TreatmentGiven, Treatment# and TCost.

Make assumptions and follow them if necessary.
Full credit will not be given unless you *show all Functional dependency diagrams for all normal forms*.

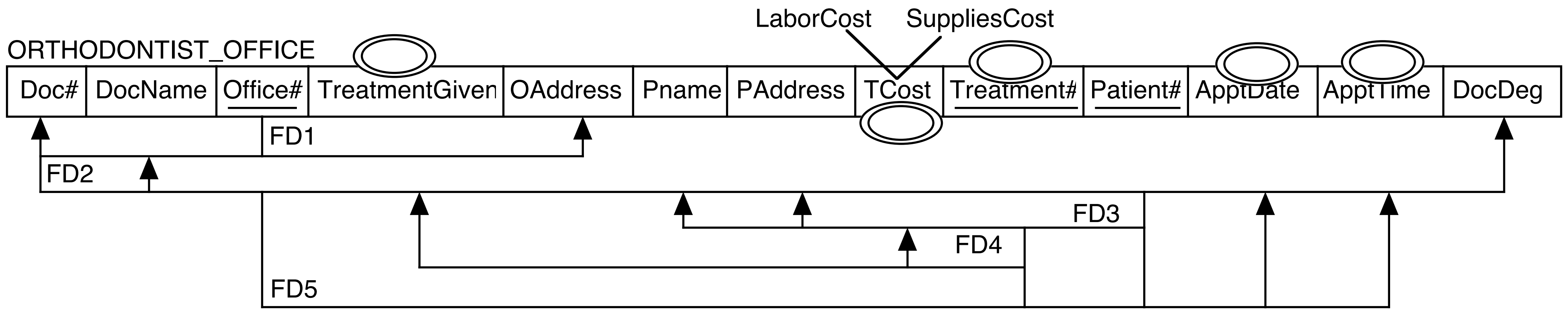
HW 7 Solution S14

To get to 1NF, we will remove multivalued and composite attributes. Separate them to their own relations and keep a copy of th original key(s) with them.

To get to 2NF, I will remove FDs that are not fully functionally dependent on the key of the relation they are in. Separate them tc own relations and keep a copy of the original key(s) with them.

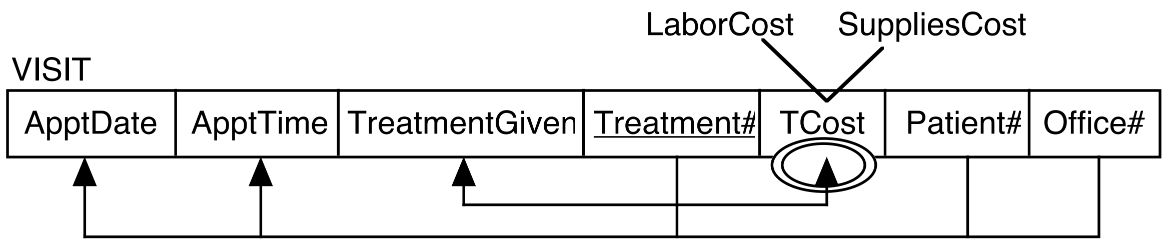
To get to 3NF, I will remove any remaining transitive dependencies. I will call the key of the new relation whatever the key of the non-prime attribute was that the transitive dependency was relying on. I will leave a copy of that key value(s) in the relation they came from.

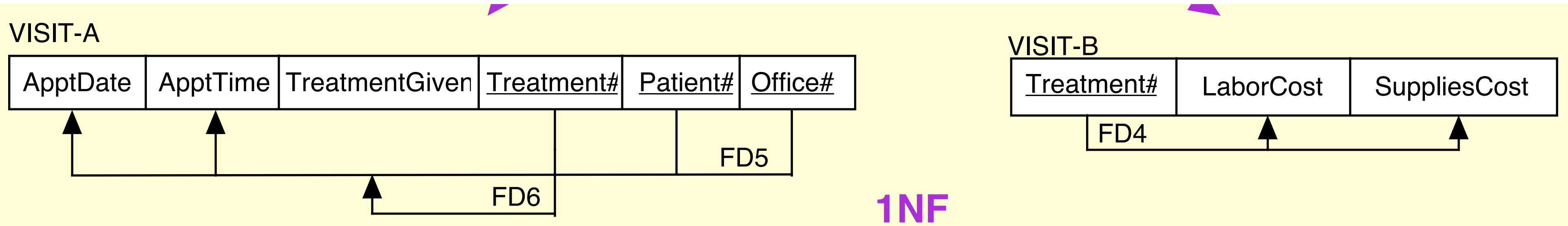
First, I must draw the functional dependencies and constraints given in the prob



The PAddress attribute had been neglected; I made it dependent on Patient

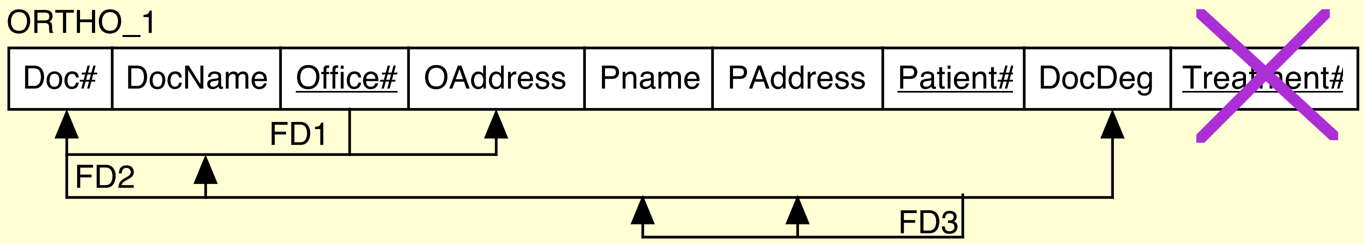
Now I will remove VISIT, then the TCost composite can be addressed from ther





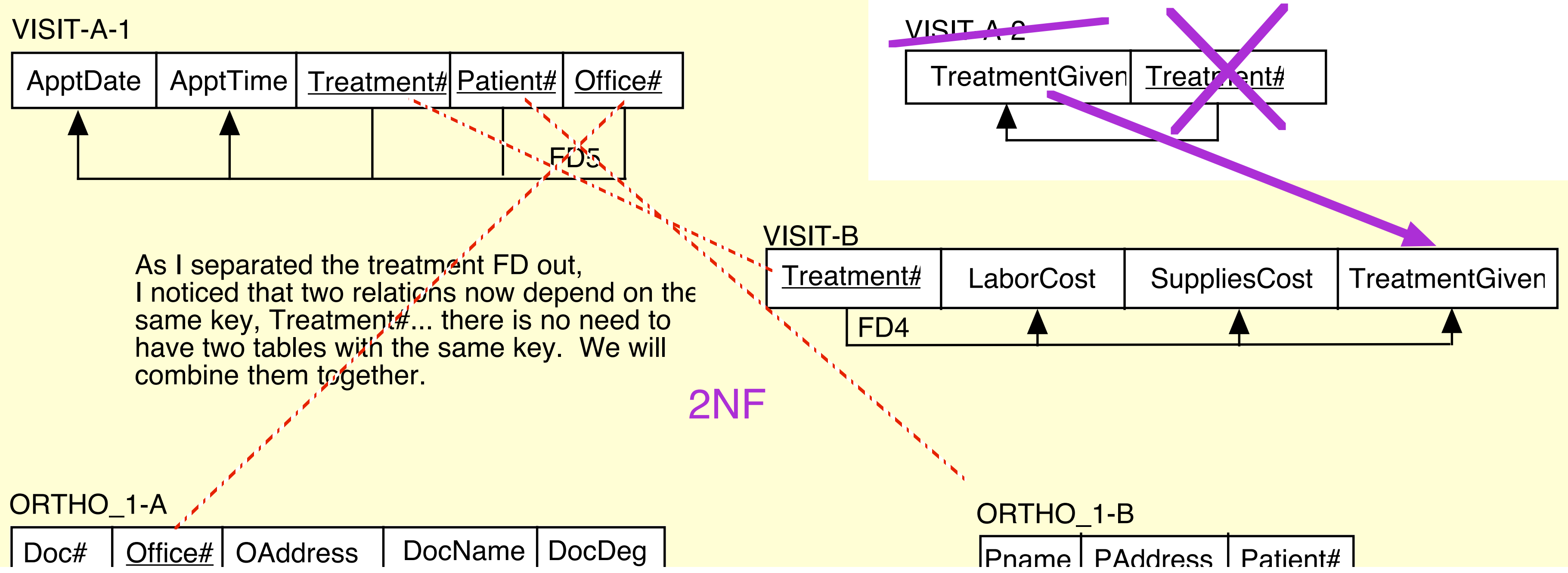
1NF

... and here is what is left of the original relati



I remove treatment from the original relation because I have represented and preserved it in the two new relations for VISIT and it no longer ha a role in and FD shown in ORTHO_1.

Now I must resolve dependencies that are not fully functionally dependent on the key of a relat

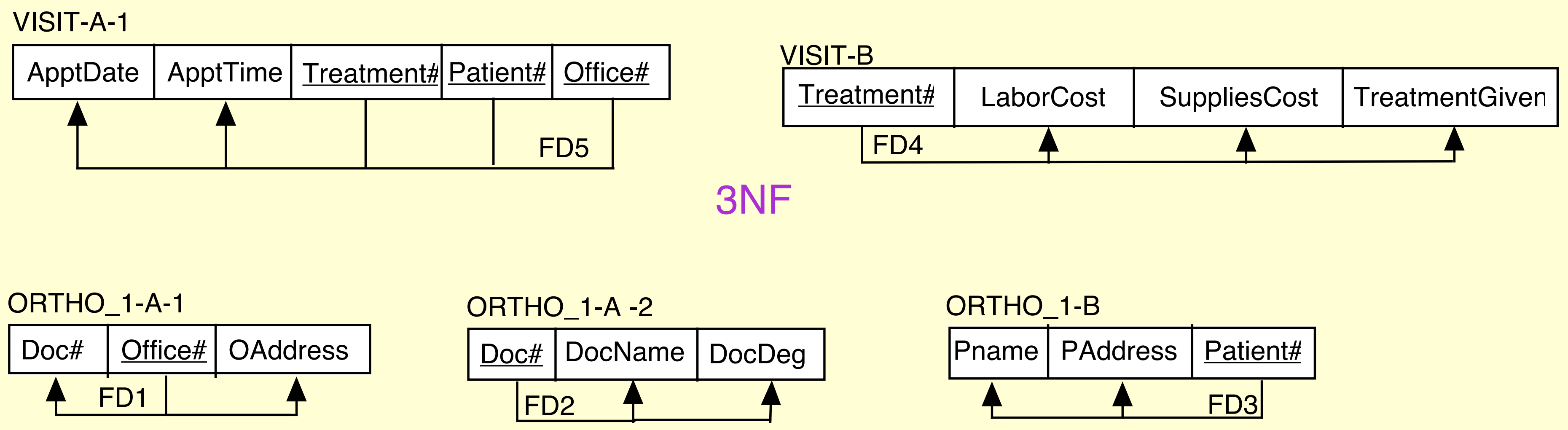


As I separated the treatment FD out,
I noticed that two relations now depend on the same key, Treatment#... there is no need to have two tables with the same key. We will combine them together.

2NF

Notice that I can still find all the attributes and relate all to one table that has the original key val

The only transitive dependency involves the Doc# attribute above


VISIT-A-1
ApptDate
ApptTime
Treatment#
Patient#
Office#
FD5
VISIT-B
Treatment#
LaborCost
SuppliesCost
TreatmentGiven
FD4
3NF
ORTHO_1-A-1
Doc#
Office#
OAddress
FD1
ORTHO_1-A -2
Doc#
DocName
DocDeg
FD2
ORTHO_1-B
Pname
PAddress
Patient#
FD3