Assignment 8
CPS 510
November 18th, 2021

We will be using Bernstein's algorithm to normalize the following tables in 3NF.

3NF vs BCNF
3NF: A partial key (prime attribute) can *also* depend on an attribute that is ***not*** a superkey
BCNF: Every partial key (prime attribute) can *only* depend on a superkey,

Table 11 in 3NF Form (using Bernstein's algorithm)

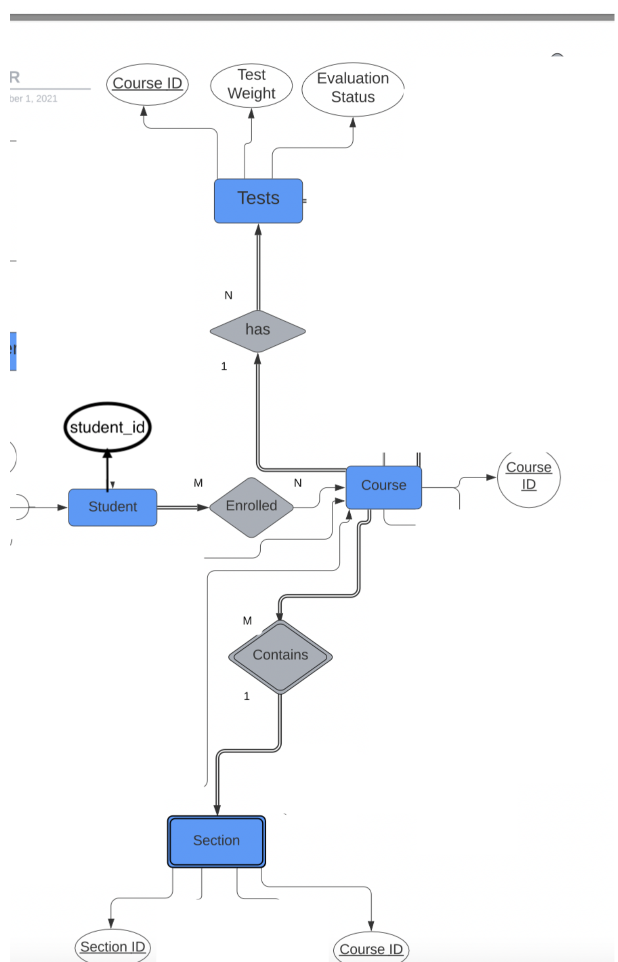
Why is this not in 3NF?
testName → testEvaluationStatus
testEvaluationStatus is transitively dependant on testName

For 3NF we need to make sure any non prime attributes are dependent on a prime attribute. In this case we have a non prime attribute dependent on another non prime attribute.

Step 1. Determine Functional Dependencies

testID → testWeight
courseID → sectionID
sectionID → courseID
studentID, sectionID → testEvaluationStatus
sectionID → testName
testName → testID
testName → testEvaluationStatus

testID = t
testWeight = w
courseID = c
sectionID = s
studentID = i
testEvaluationStatus = e
testName = n



Step 2. Break RHS and find redundancies

FD = {
t → w,
c → s,
s → c,
si → e,
s → n,
n → t
n → e
}

t → w: t += {t} we do not get w, so not redundant
c → s: c+= {c} we do not get s, so not redundant
s → c: s += {s,n,t,w,e} we do not get c, so not redundant
si → e: si += {s,i,c,n,t,w} we do not get e, so not redundant
s → n: s += {s,c} we do not get n, so not redundant
n → t: n+= {n} we do not get t so not redundant
n → e: e+= {n,t,w} we do not n so not redundant
So after removing redundancies, FD = {t → w, c → s, s → c, si → e, s → n, n → t, n → e}

Step 2b. Find and remove partial dependencies (or minimizing LHS)
si → e: s+= {c,s,n,t,w} i+= {i} fully dependant since we do not get e

If we find a partial dependency, we can remove one of the attributes from the LHS. In this case we checked all the LHS with composite attributes and we couldn't minimize LHS so they are 2NF. For the rest of FDs because they have one attribute on LHS they are already minimized. So FDs are:

t → w
c → s
s → c
si → e
s → n
n → t
n → e

Step 3: Find keys

A formula to make this process simple is:
If an attribute is not in LHS and is not in LHR: it is part of the key
If an attribute is only in LHS but not in RHS: it is part of the key
If an attribute is only in RHS and not in LHS: it is NOT part of the key

Firstly, find which one are not keys

→ w, n are on RHS but not in LHS
→ therefore candidate keys are: testID, courseID, student_ID

Step 4: Make Tables

- **Here we can make this 3nf by creating a separate exam table for the functional dependency** testEvaluationStatus → testName

We have FD = {t → w, c → s, s → c, si → e, s → n, e → n, n → t}

We make a relation for each functional dependency:
t → w : R1(t,w)
c → s : R2 (c, s)
s → c : R3 (s, c)
si → n : R4 (s,n,i)
s → n: R5 (s,n)
n → t : R6 (n,t)
n → e: R7 (e,n)

Now to make it lossless join and dependency preserving if there is no key in the Relations R1 to R4 we need to add at least one key. So we can add any of the HSRT or HSRC to decomposition so we have

R1(t,w) with FD: t → w
R2 (c, s) with FD: c → s
R3 (s, c) with FD: s → c
R4 (s,n,i) with FD: si → n
R5 (s,n) with FD:s → n
R6 (n,t) with FD : n → t
R7 (n,e) with FD: n → e
R8 (t,s,c) with no FD

BCNF DECOMPOSITION

Check if table is in BCNF
t → w,
c → s,
s → c,
si → e,
s → n,
n → t

A relation is in BCNF if and only if every non trivial, left irreducible FD has a candidate key as its determinant

So we need to test if the test if the hand sides are the keys are not

t+ = {t}   testID is a key
c+ = {c}   courseID is a key
s+ = {s,n,t,w)   sectionID is a key
si+ = {s,i,n,t,w,c}   sectionID and studentID are both keys
s+ = {s,c}   sectionID is a key
n+= {n,t}   testName is not a key. So it is not BCNF with respect to n → t

Since R is not BCNF with respect to n → t and
n+ = {n,t) decompose R in R11 and R12

R11 = {w,c,s,e,i}
R12 = {n,t} which is BCNF

Now the question is if R11 is in BCNF with FD11
FD11 = { t → w, c → s, s → c, si ---> e, s---> n}
si += {s,i,e,c,n,t,w} it is key
s+= {s,n,c,t,w} it is key

t += {t,w} is not key so split R11 to R111 and R112

R111 = (c,s,e,i)
With FD111 {c → s, si---> e, s ---> c}

R112 = (t,w) is BCNF

With F112 = ($t \rightarrow w$)

Final BCNF schema for R IS

R1 = (n,t)
R2 = (t,w)
R3 = (c,s,e,i)

Table 12: assignments

assignmentID → assignmentWeight, assignmentEvaluationStatus, sectionID, studentID, courseID

FDs: { assignmentID, evaluationStatus → assignmentWeight
assignmentID → evulationStatus
assignmentID → studentID
assignmentID, studentID → courseID
courseID → sectionID
}

R1: (assignmentID, evaluationStatus, assignmentWeight)
R2: (assignmentID, courseID, studentID, sectionID)