

IST 210 Section 2 Group B Project #2

Relation: People(Pers. ID, Name, Zip, City, State, Address)

FDs: PersID \rightarrow Name, Zip, City, State, Address

Address, City, State \rightarrow Zip

Zip \rightarrow City, State

New FDs: None

Closures: (PersID)⁺ = Pers. ID, Zip, City, State, Address

This would make Pers ID a candidate key, and its combination with any other attribute results in a superkey. Based on the physical workings of cities, states and the zip code system, we conclude that no combination of those attributes yields a candidate key.

People is not in BCNF; we could break it down into: People1(City, State, Zip), People2(Pers. ID, Name, Address, Zip). However, if we later rejoined People1 and People2, we would lose the FD Address, City, State \rightarrow Zip. Noticing that the attribute Zip is prime, we conclude that People is in 3NF and leave it unaltered.

Relation: Customers(Cust. ID, email)

FDs: Cust. ID \rightarrow email

New FDs: None

Closures: (Cust. ID)⁺ = Cust. ID, email

This would make Cust. ID a candidate key, and combined with email it is a superkey. We also consider:

email: The same email address could be used by two different Cust. ID's.

Since the closure of the function dependency yields all the attributes of the relation, Customers is BCNF.

Relation: CreditCard(Type, Number, Exp. Date)

FDs: Type, Number -> Exp. Date

New FDs: None

Closures: (Type, Number)+ = Type, Number, Exp. Date

Since the closure of (Type, Number) yields all the attributes of CreditCard, we know it is a candidate key. We also consider:

Type: Two cards from the same company (i.e. same type) could have different numbers.

Number: Two cards could share the same number, but they would be from different companies (i.e. be of different type.)

Type, Exp. Date: Two cards from the same company could have the same expiration date, but each would have a different number.

Number, Exp. Date: Two cards from different companies could have the same number and expiration date.

Since the closure of the dependency yields all attributes, we conclude that CreditCard is in BCNF.

Relation: Techs(Pers. ID, availability, cert level, min. rate)

FDs: Pers. ID -> availability, cert level, min. rate

New FDs: None

Closures: (Pers. ID)+ = Pers. ID, availability, cert level, min. rate

Since the closure of Pers. ID yields all attributes, we know that Pers. ID is a candidate key, and that any combination of Pers. ID and any other attribute would be a superkey. So we consider:

availability: Any two Techs with the same availability could have different certification levels.

cert. level: Any two Techs with the same certification level could have different availability.

min. rate: Any two Techs with the same minimum rate of pay could have different availability.

avail + cert.: It is possible for two Techs to have the same availability and certification level yet have a different minimum pay rate.

avail + min rate: It is possible for two Techs to have the same availability and minimum pay rate yet have a different certification level.

cert + min. rate: It is possible for two Techs to have the same certification level and minimum pay rate yet have different availability.

Since the closure of the functional dependency yields all the attributes of the relation, Techs is in BCNF.

Relation: Admin(Pers. ID, availability, fixed wage, department)

FDs: Pers. ID \rightarrow availability, fixed wage, department

New FDs: None

Closures: (Pers. ID)⁺ = Pers. ID, availability, fixed wage, department.

Since the closure of Pers. ID yields all attributes, we know that Pers. ID is a candidate key, and that any combination of Pers. ID and any other attribute would be a superkey. So we consider:

availability: Any two Admin with the same availability could have a different fixed wage.

fixed wage: Any two Admin with the same fixed wage could have a different availability.

department: Any two Admin with the same department could have different fixed wages.

availability+fixed wage: Any two Admin with the same availability and fixed wage could have different departments.

availability+department: Any two Admin with the same availability and department could have different fixed wages.

department+fixed wage: Any two employees with the same department and fixed wage could have different availability.

Since the closure of the functional dependency yields all the attributes of the relation, Admin is BCNF.

Relation: Computers(serial#, OS, proc. speed, RAM)

FDs: serial# \rightarrow OS, proc. speed, RAM

New FDs: None

Closures: (serial#)⁺ = serial#, OS, proc. speed, RAM

Since the closure of serial# yields all attributes, we know that serial# is a candidate key, and that any combination of serial# and any other attribute would be a superkey. So we consider:

OS: Any two computers with the same OS could have a different proc. speed.

proc. speed: Any two computers with the same proc. speed could have a different OS.

RAM: Any two computers with the same RAM could have a different proc. speed.

OS+proc. speed: Any two computers with the same OS and proc. speed could have different RAM.

OS+RAM: Any two computers with the same OS and RAM could have different proc. speed.

RAM+proc. speed: Any two computers with the same RAM and proc. speed could have a different OS.

Since the closure of the functional dependency yields all the attributes of the relation, Computers is in BCNF.

Relation: Mac(serial#, cert type)

FDs: serial# \rightarrow cert type

New FDs: None

Closures: (serial#)⁺ = serial#, cert type

Since the closure of serial# yields all attributes, we know that serial# is a candidate key, and that

any combination of serial# and any other attribute would be a superkey. So we consider:

cert. type: Any two Macs with the same certification requirement could have different serial#'s.

Since the closure of the functional dependency yields all the attributes of the relation, Mac is in BCNF.

Relation: PC(serial#, proc. manf., manf.)

FDs: serial# \rightarrow proc. manf., manf.

New FDs: None

Closures: (serial#)⁺ = serial#, proc. manf., manf.

Since the closure of serial# yields all attributes, we know that serial# is a candidate key, and that any combination of serial# and any other attribute would be a superkey. So we consider:

proc. manf.: Any two PCs could have the same proc. manf. but different manf.

manf.: Any two PCs could have the same manf. but different proc. manf.

proc. manf.+manf.: Any two PCs could have the same proc. manf. and manf. but could have different serial#'s.

Since the closure of the functional dependency yields all the attributes of the relation, PC is in BCNF.

Relation: Problems(Prob ID, description, solution, critical)

FDs: Prob ID \rightarrow description, solution, critical
solution \rightarrow critical

New FDs: None

Closures: (Prob. ID)⁺ = Prob. ID, description, solution, critical
(solution)⁺ = solution, critical

Therefore, Problems is not in BCNF and must be decomposed. This makes sense because the

criticality of a problem depends on what solution must be taken. We decompose Problems into:
Criticality(solution, critical) and Prob/Solution(Prob. ID, description, solution)

Relation: Criticality(solution, critical)

FDs: solution \rightarrow critical

New FDs: None

Closures: (solution)⁺ = solution, critical

Since the closure of solution yields all attributes, it is a candidate key, and any other attribute combined with solution is a superkey. We also consider:

critical: different solutions may have the same Boolean critical value.

Since the closure of the FD yields all attributes, we now conclude that Criticality is in BCNF

Relation: Prob/Solution(Prob. ID, description, solution)

FDs: Prob. ID \rightarrow description, solution

description, solution \rightarrow Prob. ID

New FDS: None

Closures: (Prob. ID)⁺ = Prob. ID, description, solution

(description, solution)⁺ = Prob. ID, description, solution

Since the closure of Prob. ID yields all attributes, it is a candidate key, and any other attribute combined with solution is a superkey. The same is true for (description, solution). We also consider:

description: There could be two different solutions for the same description.

solution: Two different problems could have the same solution.

Since the closure of each FD yields all attributes of the relation, we conclude that Prob/Solution is now in BCNF.

Relation: Have(Pers. ID, Type, Number)

FDs: None; Multiple people might use the same credit card (a husband and a wife, for example) and the some people might use multiple credit cards.

New FDs: None

Closures: None

Since there are no FDs in this relation, we conclude that is in BCNF.

Relation: Belongs-To(Serial#, Pers. ID.)

FDs: serial# -> Pers. ID We know this because Belongs-To is an open-arrow many to one relationship from Computers to Customers; each computer belongs to only one person.

New FDs: None

Closures: (serial#)+ -> serial#, Pers. ID

Since the closure of serial# yields all attributes, we know that serial# is a candidate key, and that any combination of serial# and any other attribute would be a superkey. So we consider:

Pers. ID: One person (i.e. personal ID) might own more than one computer.

Since the closure of the functional dependency yields all the attributes of the relation, Belongs-To is in BCNF.

Relation: Diagnosis(serial#, Prob ID)

FDs: None; it is a many-many relationship

New FDs: None

Closures: None

Since there no FDs in this relation, we conclude that Diagnosis is in BCNF.

Relation: Works-On(Pers. ID, serial#, rate)

FDs: Pers. ID, serial# \rightarrow rate

New FDs: None

Closures: (Pers. ID, serial#)⁺ = Pers. ID, serial#, rate

Since the closure of (Pers. ID, serial#) yields all attributes, we know that (Pers. ID, serial#) is a candidate key, and that any combination of (Pers. ID, serial#) and any other attribute would be a superkey. So we consider:

Pers. ID: One employee could work on multiple computers.

serial#: One computer could be worked on by multiple employees.

rate: A particular rate might be associated with more than one instance of Pers. ID and serial#

rate + serial#: Multiple employees might work on the same computer for the same rate.

Pers. ID + rate: One employee might work on multiple computers for the same rate.

Since the closure of the functional dependency yields all the attributes of the relation, Works-On is in BCNF.