COURSE: CS/DSA- 4513 – DATABASE MANAGEMENT
SECTION: 001
SEMESTER: FALL 2019
INSTRUCTOR: DR. LE GRUENWALD

GROUP NUMBER:
43

GROUP MEMBERS:

MASON DRUCTOR
LINCE RUMAINUM
MASON SCHMIDT
TUAN VU

SCORE:

Problem 1

2.14

a) $\Pi_{\text{person\_name}}\left(\sigma_{\text{company\_name="BigBank"}}(works)\right)$

b) $\Pi_{\text{person\_name,city}}\left(\sigma_{\text{company\_name="BigBank"}}(employee \bowtie works)\right)$

c) $\Pi_{\text{person\_name,street,city}}\left(\sigma_{\text{company\_name="BigBank"} \wedge \text{salary} > 10000}(employee \bowtie works)\right)$

d) $\Pi_{\text{person\_name}}(employee \bowtie works \bowtie company)$

27.8

a)
$\{t \mid \exists\ s\ \varepsilon\ works\ (t[person\_name] = s[person\_name] \wedge s[company\_name] = "FBC")\}$

b)
$\{t \mid \exists\ s\ \varepsilon\ works\ (t[person\_name] = s[person\_name] \wedge s[company\_name] = \text{"FBC"}$
$\wedge\ \exists\ u\ \varepsilon\ employee\ (t[city] = u[city] \wedge t[person\_name] = u[person\_name]))\}$

c)
$\{t \mid \exists\ t\ \varepsilon\ \text{employee}\ (\exists\ s\ \varepsilon\ works\ (t[person\_name] = s[person\_name]$
$\wedge\ s[company\_name] = "FBC" \wedge s[salary] > 10000\ ))\}$

d)
$\{t \mid \exists\ t\ \varepsilon\ \text{employee}\ (\exists\ s\ \varepsilon\ works\ (t[person\_name] = s[person\_name]$
$\wedge\ \exists\ u\ \varepsilon\ employee\ (\ s[company\_name] = u[company\_name]\ \hat{}\ t[city] = u[city]))))\}$

Problem 2

a)
(classid, id, gender) → (salary, manager)
id → name
Name → (age, id)

Combine with transitivity and augmentation and trivial functional dependencies (FD):
(classid, id, gender) → (classid, id, gender, salary, manager, name, age) = R
So, (classid, id, gender) is a superkey.

Now we check if (classid, id, gender) is minimal.
Since classid and gender does not determine any other attributes in FD (not on the left hand side (LHS) of any other FD), we only need to check if id deteremines (salary, manager).
Without classid and gender, by transivity, id → name, name → (age, id), so id → age ≠ R is the only other FD and so (classid, id, gender) is a candidate key.

name → (age, id) ≠ R and since that is the only FD, name is not a superkey.

manager → (gender, age, classid, id)
Since this FD determines the previous candidate key (classid, id, gender), manager is a superkey and since it is atomic so it is also a candidate key.

Candidate keys are {(classid, id, gender), manager}

b)

1NF Pass
All attributes are atomic assuming attributes all of the attributes are not divisible and has no composite value (i.e: name cannot be split into first/last name)

2NF Fail
Violation: id is a part of a candidate key, but id → name, which is a partial dependency because name is non-prime attribute that is determined by a subset of a candidate key, id.

3NF Fail
Violation: not in 2NF
Violation: Not all attributes on the LHS of FD are superkeys and not all attributes on the RHS of FD are prime attributes.
name → (age, id) and name is not a super key and age is a non-prime attribute.
id → name because name is a non-prime attribute although id is on LHS is a superkey.

BCNF Fail
Violation: not in 3NF
Violation: name → (age, id) and name on the LHS is not a superkey.

c)
Decompose to 2NF

id $\rightarrow$ (name, age)
A violation of 2NF so we decompose the schema to:

EMPLOYEE1(id, classid, gender, manager, salary)

EMPLOYEE1 Functional Dependencies =
{(classid, id, gender) $\rightarrow$ (salary, manager),
Manager $\rightarrow$ (gender, classid, id)}

Candidate Keys for EMPLOYEE1
{(classid, id, gender), manager}

EMPLOYEE is in 2NF because (classid, id, and gender) are all required to for the functional dependency to determine (salary, manager), manager is atomic, there is no partial dependencies, so it cannot violate 2NF.

EMPLOYEE2(id, name, age)
EMPLOYEE2 Functional Dependencies =
{(name $\rightarrow$ (age, id),
Id $\rightarrow$ name}

Candidate Keys for EMPLOYEE2
{name, id}

EMPLOYEE2 is in 2NF because each candidate key is atomic and there is no partial dependencies, it cannot violate 2NF.

Problem 3

Application Details

This application is for a made-up security application for an office rental company that keeps track of the companies that use office-space in their real-estate. Each building needs to be kept track of with the number of floors, number of rooms, its physical address, and a unique building name. In each building there will be a number of rooms we need to keep track of with a unique room number, the size of the room, and the name of the building it is located in. Each building can have a number of companies working in them, each company kept track of with a unique company name, their mission statement, and the building name that they are located in. Each company can have a number of employees which have a unique id, a name, and the company name that they work for.

Functional Dependencies

Employee
id → name, company_name

Company
company_name → building_name, mission_statement

Building
building_name → number_of_floors, number_of_rooms, address

Room
room_num → room_size, building_name

Schema

employee(id, name, company_name)
company(company_name, building_name, mission_statement)
building(building_name, number_of floors, number_of_rooms, address)
room(room_num, room_size, building_name)