**Database System Homework**

1. Consider the following relation schemas for the Retail Business Management System:

**Employees(eid, ename, address, telephone#)** /\* It is assumed that each telephone number is associated with exactly one address. \*/

**Clerks(eid, start\_date, hourly\_rate, hours)/\*** clerks who start to work on the same day have the same salary rate;\*/

**Managers(eid, start\_date, salary, prom\_date)/\*** managers who are promoted at the same time have the same salary rate;**\*/**

**Products(pid, pname, qoh, qoh\_threshold, original\_price, description, discnt\_category, discnt\_rate, discnt\_start\_date, length)** /\* Discount rate is determined by the discount category. \*/

**Suppliers(sid, sname, city, telephone#, email\_addr)/\*** sid is the primary key. A telephone number and email address are required. Different suppliers have different telephone numbers and different email addresses **\*/**

Do the following for each relation schema:

1. (30 points) Identify all non-trivial functional dependencies. Don’t make unrealistic assumptions about the data. Should use the union rule to combine the functional dependencies as much as possible. (You do not need to list redundant FDs. For example, if you already have A 🡪 BC, you don’t need to also list A 🡪 B and A 🡪 C. As another example, if you already have A 🡪 B and B 🡪 C, you don’t need to include A 🡪 C).

**Employees(eid, ename, address, telephone#)**

FDs: eid 🡪 ename telephone#

telephone# 🡪 address

**Clerks(eid, start\_date, hourly\_rate, hours)**

FDs: eid 🡪 start\_date hours

start\_date 🡪 hourly\_rate

**Managers(eid, start\_date, salary, prom\_date)**

FDs: eid 🡪 start\_date prom\_date

prom\_date 🡪 salary

**Products(pid, pname, qoh, qoh\_threshold, original\_price, description, discnt\_category, discnt\_rate, discnt\_start\_date, length)**

FDs: pid 🡪 pname qoh qoh\_threshold original\_price description discnt\_category discnt\_start\_date

discnt\_category 🡪 discnt\_rate

**Suppliers(sid, sname, city, telephone#, email\_addr)**

FDs: sid 🡪 sname city telephone#

telephone# 🡪 sid

email\_addr 🡪 sid

1. (30 points) Determine whether or not the schema is in 3NF or in BCNF. Need to provide justification.

Employees(eid, ename, address, telephone#)

eid is the only candidate key and the only prime attribute. This schema is not in 3NF because in telephone#  address, we have a non-prime (address) non-trivially depends on a non-superkey (telephone#). If a schema is not in 3NF, it cannot be in BCNF (this applies to the same cases below).

Clerks(eid, start\_date, hourly\_rate, hours)

eid is the only candidate key and the only prime attribute. This schema is not in 3NF because in start\_date  hourly\_rate, we have a non-prime (hourly\_rate) non-trivially depends on a non-superkey (start\_date)

Managers(eid, start\_date, salary, prom\_date)

eid is the only candidate key and the only prime attribute. This schema is in not in 3NF because in prom\_date  salary, we have a non-prime (salary) non-trivially depends on a non-superkey (prom\_date).

Products(pid, pname, qoh, qoh\_threshold, original\_price, description, discnt\_category, discnt\_rate, discnt\_start\_date, length)

pid is the only candidate key and the only prime attribute. This schema is not in 3NF because in discnt\_category  discnt\_rate, we have a non-prime (discnt\_rate) non-trivially depends on a non-superkey (discnt\_category). The functional dependency “discnt\_category discnt\_start\_date  length” also makes the schema not in 3NF.

Suppliers(sid, sname, city, telephone#, email\_addr)

sid, telephone# and email\_addr are candidate keys and prime attributes. This schema is in BCNF because for each non-trivial functional dependency, the left-hand side (either sid, telephone# or email\_addr) is a superkey.

1. (40) For each schema that is not in 3NF, decompose it into 3NF schemas using Algorithm LLJD-DPD-3NF. Show the result after each step of the algorithm. Are they decomposed schemas in BCNF? Justify your answer.

Decompose Employees into 3NF:

1. All candidate keys: eid
2. Minimal cover: eid 🡪 ename, eid 🡪 telephone#, telephone# 🡪 address
3. Decomposition: R1 = eid ename telephone#

R2 = telephone# address

BCNF: Both R1 and R2 are in BCNF as for each non-trivial functional dependency, the left is a super key.

Decompose Clerks into 3NF:

1. All candidate keys: eid
2. Minimal cover: eid 🡪 start\_date, eid 🡪 hours, start\_date 🡪 hourly\_rate
3. Decomposition: R1 = eid start\_date hours

R2 = start\_date hourly\_rate

BCNF: Both R1 and R2 are in BCNF as for each non-trivial functional dependency, the left is a super key.

Decompose Managers into 3NF:

1. All candidate keys: eid
2. Minimal cover: eid 🡪 start\_date, eid 🡪 prom\_date, prom\_date 🡪 salary
3. Decomposition: R1 = eid start\_date prom\_date

R2 = prom\_date salary

BCNF: Both R1 and R2 are in BCNF as for each non-trivial functional dependency, the left is a super key.

Decompose Products into 3NF:

1. All candidate keys: pid
2. Minimal cover: pid 🡪 pname, pid 🡪 qoh, pid 🡪 qoh\_threshold, pid 🡪 original\_price, pid 🡪 description, pid 🡪 discnt\_category, pid 🡪 discnt\_start\_date, discnt\_category 🡪 discnt\_rate,

Decomposition: R1 = pid pname qoh qoh\_threshold original\_price description discnt\_category discnt\_start\_date,length

R2 = discnt\_category discnt\_rate

BCNF: R1, R2 are all in BCNF as for each non-trivial functional dependency, the left is a superkey.