# CENG-352 WRITTEN ASSIGNMENT 1

Q1/ a)

```
create table Department (
    dept_id int,
    location char[55],
    name char[100],
    emp_id IS NOT NULL,
    primary key (dept_id),
    foreign key (emp_id)
        references Employee(emp_id)
        on delete set 101 );


create table Works_in (
    dept_id int,
    emp_id int,
    primary key (dept_id, emp_id),
    Foreign key (dept_id)
        references Department (dept_id)
        on delete no action,
    Foreign key (emp_id)
        references Employee (emp_id)
        on delete cascade );
```

```
create table Employee (
    emp_id int,
    name char [100],
    surname char [50],
    salary double,
    gender char [1]);


create table runs_project (
    project_id int,
    state char [50],
    due_date date,
    budget float,
    dept_id int,
    primary key (project_id, dept_id),
    foreign key (dept_id)
        references Department (dept_id)
        on delete cascade);
```

```
create table Reports_to(
   sup_emp_id int,
  sub_emp_id int,
   primary key (sup_emp_id, sub_emp_id),
  Foreign key (sup_emp_id)
        references Employee(emp_id),
   foreign key (sub_emp_id)
        references Employee(emp_id))
```

b)
```
create assertion every1_works(
    check (
        not exists(
            select emp_id
            from Works_in
            group by emp_id
            having count(emp_id)=0)
    )
)
```

```
c) create table Employee (

        — —
        — — ,

        salary double
             check (salary >= 10000),

        — — )


   create table Department (

        — —
        — — ,

        name char[100]
             check (name like
                  concat('%', location) or
                       name like
                  concat(location, '%')),

        — — — )

d) create trigger update_budget
   on runs_project after update
   referencing new table as inserted
               old table as deleted

   begin
     update runs_project
     set state = 'Unsuccessful' where
     project_id in
```

```
(select  i.project_id
 from    inserted i, deleted d
 where   i.project_id = d.project_id
         and
         d.budget > i.budget)

end
```

Q2/ a) Since product and store entities have many-to-many relation, there are 5x100 = 500 tuples between them.

Since (product, store) tuples can be related with at most one person, the max. number of tuples can be $\boxed{500}$.

b) Since Product and Store entities has many-to-one relation, there are 100 tuples can be obtained as (product, store).

Every (Product, store) tuple can be sold by at most one SalesPerson. So we can obtain (Product, Store, SalesPerson) only 100 tuples.

Every customer can buy every product: 990x100 = 99000

So, the answer is 99000+100 = 99100.

Q3/ b)

$$\left.\begin{array}{l} A \rightarrow C \\ CB \rightarrow F \end{array}\right\} AB \rightarrow F \quad \text{(Pseudo transivity)} \; ①$$

$B \rightarrow E$ ②

$AB \rightarrow EF$ (Union of ① and ②)

a) $\left.\begin{array}{l} CB \rightarrow F \\ B \rightarrow E \end{array}\right\}$ $CB \rightarrow FE$ (union) ①

$FE \rightarrow G$ ②

$CB \rightarrow G$ (transitivity of ① and ②)

Q4/

a) $\{A\}^+ = \{A, B\}$

$\{C, D\}^+ = \{C, D, E, G\}$

$\{F\}^+ = \{F, D, C\}$

$\{E\}^+ = \{E, G\}$

$\{A, C\}^+ = \{A, B, C, D, E, G\}$

$\{D\}^+ = \{D, C, E, G\}$

Since none of the LHS's of given FD's is the key and F is not in the closures of LHS's, F must be subset of the key.

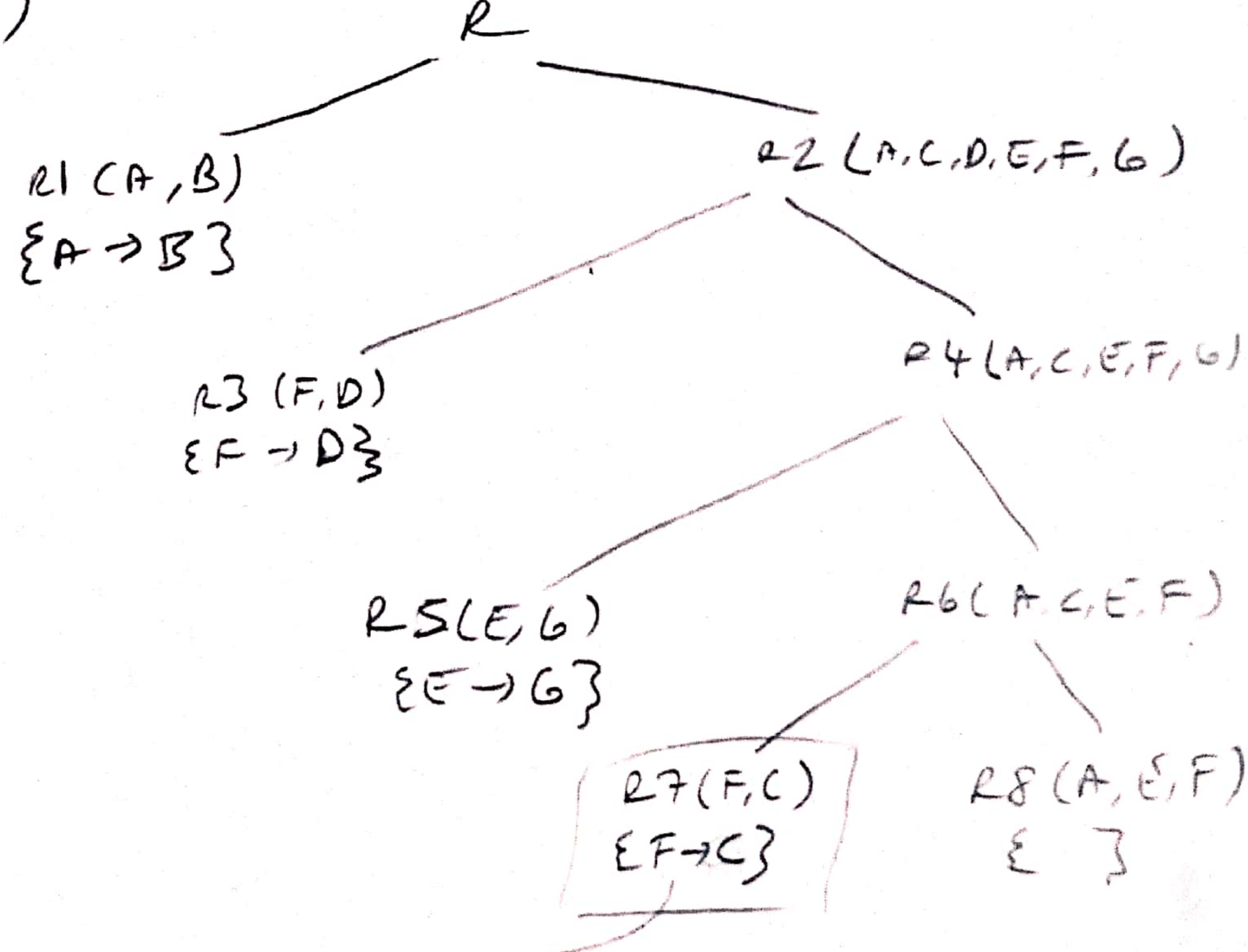
$\{A, F\}^+ = \{A, B, F, D, C, E, G\} \Rightarrow$ AF is the key!

So, we don't need to search the closures of attributes that includes A & F.

Since other combines with F attribute are not the key, the only key is AF.

b) Since all LHS's of FD's are not key, R is not in BCNF.

c)



this is non-trivial FD comes from the closure of F.

d) i) Since some of FD's are lost when we decomposed R into a collection of BCNF, it is not dependency-preserving.

ii) BCNF decomposition is always lossless.

Q5/

a) A → E

C → A

C → B

C → E

AB → C

BE → C

E → A

**NOTE:** Given table is named as "example" in DBeaver and columns are named as A, B, C, D and E respectively.

SQL Statements

A → E

```
select count (distinct e)
from example
group by a ;
```

⇒ If all counts are equal to 1, then this FD holds. (It is suitable for every FD's.

You can replace attributes for other FD's.

```
b) create  table  R1 (
       A    varchar (20),
       E    varchar (20),
       primary  key (A) );

   create  table  R2 (
       C    int,
       B  varchar (20),
       primary  key (C)
   );

   create  table R3 (
        C int,
        A varchar (20),
        primary  key (C),
        foreign  key (C)  references  R2(C));

   create  table  R4 (
        c int, D int );
```

c)

```
insert into R1
Select distinct A, E from example;

insert into R2
Select distinct C, B from example;

insert into R3
Select distinct C, A from example;

insert into R4
select c, d from example;
```