ASSIGNMENT 3

Problem 1:

Solution has been submitted separately.

Problem 2:

Consider a relational schema R = (A, B, C, D, E, F, G) satisfying the following functional dependencies: F = {DF->A, E->A, FB->C, D->A, E->G, F->BG}

a. Derive all candidate keys for this schema.

Ans: The Candidate Key for this R will be (DEF).
The closure of $(DEF)^+$={ABCDEFG}.

b. Derive a canonical cover of the functional dependencies in F.

Ans: The functional dependencies are:
F: {DF→A, E→A, FB→C, D→A, E→G, F→BG}

As F → BG then F→B and F→G …….Decomposition
As FB→C and F→B then F→C

As FB→C and F→B then B is extraneous attribute in FB→C thus we will have F→C
As DF→A and D→A then F is extraneous attribute in DF→A

The canonical cover of functional dependencies F will be:

F_c={D→A, E→A, E→G, F→B, F→C, F→G }
i.e. F_c={ D→A, E→AG, F→BCG}

c. Is the above schema in BCNF? Prove or disprove. If it is not in BCNF, convert it into BCNF.

d. Ans: R is not in BCNF. Because for schema to be BCNF for each non-trivial dependency A→B, A should be a superkey which is not the case here.

Considering the canonical cover,

1. As D→A holds on R and D is not the super key we replace R in to two other relations R1=(D, A) and Relation1=(B, C, D, E, F, G) where R1 is in BCNF

2. As F→BCG holds on Relation1 and F is not the super key we replace Relation1 by in to two other relations as R2= (F, B, C, G) and R3=(D, E, F) where R2 and R3 are in BCNF.

Thus finally we get R1= (D, A), R2=(F, B, C, G) and R3=(D, E, F) where all are in BCNF.

e.Is the BCNF schema from c) dependency-preserving? Prove or disprove. If not, convert it into 3NF.

Ans:

F_c={ D→A, E→AG, F→BCG}
F1={D→A} is preserved by R1.
F3={ F→BCG } is preserved by R2.
But F2={E→AG} is not preserved in result of c thus the BCNF form in c is not dependency preserving

Converting into 3NF we get:
F_c={ D→A, E→AG, F→BCG} and Candidate key = {D, E, F}

Therefore, 3NF form is:
R1 = {D, A},
R2 = {F, B, C, G},
R3 = {A, E, G},
and R4 = {D, E, F}

Problem 3:

BakeryInformation(custid, custname, cphone, address, city, zip, cakeid, cakename, cakeprice, slices, status, ingredid, iname, iprice, qty, orderdate, pickupdate, pricepaid)

a. Explain why the above is not a good relational design. Name several reasons.
   Ans: It is not a good relation design because of the following reasons:
   1) Redundancy: For each order, all the details of Customer, Cake and their ingredients will be repeated.
   2) Chances of inconsistency increase as same data is repeated several times.
   3) NULL Values: In case of
      a) New cake recipe is found and no order is made null values will be assigned.
      b) New customer is added to the table.
      c) Bakery has some ingredients, which are not being used in any cake.
   4) Updating anomaly: If for a particular customer, address needs to be changed, then every row for that customer needs to be updated.

5) For accessing minimal data, entire table table has to check which makes it time consuming.
6) Lastly, this schema does not adhere to the normalization rules. As everything is under one table, it will make maintenance and querying from the database difficult and it will be hard to see relation between each attribute. The relational database system works better with several small tables than a big table.

b. Derive all candidate keys for this table.

Ans: There is only one candidate key: {custid, cakeid, ingredid, orderdate}.

c. Identify the set F of non-trivial functional dependencies for this schema. (It is enough to identify a subset E such that the closures of E and F are the same.)
Ans:

custid→{ custname, cphone, address, city}
cakeid→{ cakename, cakeprice, slices, status}
ingredid→{ iname, iprice}
cakeid, Ingredid→{ qty}
custid, cakeid, orderdate→{ pickupdate, pricepaid }

d. Derive a canonical cover of the functional dependencies in F
Ans: Since there is no extraneous attribute and there is no transitive dependency present, the canonical cover of the functional dependencies in F is same as part c.

e. Is the above schema in BCNF? Prove or disprove. If it is not in BCNF, convert it into BCNF
Ans:
Schema is not in BCNF. Because for schema to be BCNF for each non-trivial dependency A→B, A should be a superkey which is not the case here.

for nontrivial dependency custid→{ custname, cphone, address, city},
{custid} is not superkey or { custname, cphone, address, city} is not subset of {custid}. Therefore, decomposing to BCNF gives us:
CUSTOMER (<u>custid</u>, custname, cphone, address, city)

for nontrivial dependency cakeid→{ cakename, cakeprice, slices, status},
{cakeid} is not superkey or { cakename, cakeprice, slices, status } is not subset of {cakeid}. Therefore, decomposing to BCNF gives us:
CAKE(<u>cakeid</u>, cakename, cakeprice, slices, status)

for nontrivial dependency ingredid→{ iname, iprice},
{ingredid } is not superkey or {iname, iprice} is not subset of { ingredid }. Therefore, decomposing to BCNF gives us:
INGREDIENT(<u>ingredid</u> ,iname, iprice)

for nontrivial dependency cakeid, Ingredid→{ qty},
{cakeid, Ingredid } is not superkey or {qty} is not subset of { cakeid, Ingredid}. Therefore, decomposing to BCNF gives us:
RECIPE(<u>cakeid, Ingredid,</u> qty)

for nontrivial dependency custid,cakeid, orderdate→{ pickupdate, pricepaid },
{ custid,cakeid, orderdate } is not superkey or { pickupdate, pricepaid } is not subset of { custid,cakeid, orderdate }. Therefore, decomposing to BCNF gives us:
ORDER(<u>custid,cakeid, orderdate</u>, pickupdate, pricepaid)

f. Is the BCNF schema from e) dependency-preserving? Prove or disprove. If not, convert it into 3NF

Ans: BCNF Schema from e) preserves dependency

custid→{ custname, cphone, address, city} is preserved in CUSTOMER
cakeid→{ cakename, cakeprice, slices, status} is preserved in CAKE
ingredid→{ iname, iprice} is preserved in INGREDIENT
cakeid, Ingredid→{ qty} is preserved in RECIPE
custid,cakeid, orderdate→{ pickupdate, pricepaid } is preserved in ORDER

Thus there is no need to convert the schema into 3NF.

g. If we add an additional constraint that a certain kind of cakes ordered on a certain day have the same price(for example, customers paid the same price for all orders for a 'red bean cake' ordered on 2017-03-10). How would that change your answers in parts b) through f)?

Ans:

b. There will be no changes in candidate key . Thus candidate key: {custid, cakeid, ingreid, orderdate}.

c. There will be one more functional dependency as custid,orderdate→pricepaid

custid→{ custname, cphone, address, city}
cakeid→{ cakename, cakeprice, slices, status}
ingredid→{ iname, iprice}

cakeid, Ingredid→{ qty}
custid,cakeid, orderdate→{ pickupdate}
custid,orderdate → {pricepaid}

d. Canonical cover of the functional dependencies in F is same as part c.

e. BCNF will be:

CUSTOMER (<u>custid</u>, custname, cphone, address, city)

CAKE(<u>cakeid</u>, cakename, cakeprice, slices, status)

INGREDIENT(<u>ingredid</u> ,iname, iprice)

RECIPE(<u>cakeid, Ingredid, qty</u>)

Above all relations are in BCNF.

for nontrivial dependency custid,orderdate → {pricepaid} , we will decompose ORDER as it is not in BCNF.

PRICE_OF_DATE(<u>custid, orderdate</u>,pricepaid)

ORDER1(<u>custid, cakeid, orderdate</u>, pickupdate)

f. Part e preserves functional dependency.

custid→{ custname, cphone, address, city} is preserved in CUSTOMER
cakeid→{ cakename, cakeprice, slices, status} is preserved in CAKE
ingredid→{ iname, iprice} is preserved in INGREDIENT
cakeid, Ingredid→{ qty} is preserved in RECIPE
custid,cakeid, orderdate→{ pickupdate} is preserved in ORDER1
custid,orderdate → {pricepaid} is preserved in PRICE_OF_DATE

Problem 4:

Following queries are using the **MySQL** Database:

a. List all tables in the schema, and their number of attributes.

```
mysql> select table_name,count(*) as number_of_attributes
from information_schema.columns
where table_schema='food'
group by table_name;
+-------------+----------------------+
| table_name  | number_of_attributes |
+-------------+----------------------+
| CUSTOMER    |                    5 |
| MENUITEM    |                    3 |
| ORDER1      |                    7 |
| ORDERDETAIL |                    4 |
| RESTAURANT  |                    4 |
+-------------+----------------------+
5 rows in set (0.00 sec)
```

b. List the table with the most attributes.

```
mysql>select y.table_name from
(select table_name,count(*) as number_of_attributes
from information_schema.columns
where table_schema='food'
group by table_name
having number_of_attributes =
    (select max(x.counter)
    from (select table_name,count(*) as counter
        from information_schema.columns
        where table_schema='food'
        group by table_name)as x))as y;
+------------+
| table_name |
+------------+
| ORDER1     |
+------------+
1 row in set (0.01 sec)
```

c. List the attribute name and table name of any attribute of type int.

```
mysql> select table_name,column_name
from information_schema.columns
where table_schema='food' and data_type='int';
+-------------+-------------+
| table_name  | column_name |
+-------------+-------------+
| CUSTOMER    | cid         |
| MENUITEM    | rid         |
| ORDERDETAIL | oid         |
| ORDERDETAIL | rid         |
| ORDERDETAIL | quantity    |
| ORDER1      | oid         |
| ORDER1      | cid         |
| ORDER1      | rid         |
| RESTAURANT  | rid         |
+-------------+-------------+
9 rows in set (0.01 sec)
```

d. List any pairs of tables that have an attribute with the same name and same data type.

```
mysql> select t1.table_name,t2.table_name
from
(select table_name,column_name,data_type
from information_schema.columns
where table_schema='food' and data_type='int')as t1 ,
(select table_name,column_name,data_type
from information_schema.columns
where table_schema='food' and data_type='int')as t2
where t1.table_name<t2.table_name and t1.column_name=t2.column_name
and t1.data_type=t2.data_type;
+-------------+-------------+
| table_name  | table_name  |
+-------------+-------------+
| MENUITEM    | ORDERDETAIL |
| ORDERDETAIL | ORDER1      |
| CUSTOMER    | ORDER1      |
| MENUITEM    | ORDER1      |
| ORDERDETAIL | ORDER1      |
| MENUITEM    | RESTAURANT  |
| ORDERDETAIL | RESTAURANT  |
| ORDER1      | RESTAURANT  |
+-------------+-------------+
8 rows in set (0.00 sec)
```