Ceng352 Written Assignment 1

Bilge Eren 2171587

March 5, 2020

1

1.1

```
create table departments (
     dept_id int not null,
     location varchar(100),
     name varchar(100),
     manager_id int not null,
     primary key (dept_id),
     foreign key (manager_id)
         references employees(emp_id)
         on delete set default
);

create table project (
    project_id int not null,
    dept_id int not null,
    due_date date,
    state varchar(100),
    budget int,
    primary key (project_id, dept_id),
    foreign key (dept_id)
        references departments(dept_id)
);

create table employees (
    emp_id int not null default 101,
    name varchar(100),
    surname varchar(100),
    salary int,
    gender varchar(100),
    primary key (emp_id)
);

create table works_in (
    emp_id int not null,
    dept_id int not null,
    primary key (emp_id, dept_id),
    foreign key (dept_id)
        references departments(dept_id)
        on delete restrict,
    foreign key (emp_id)
        references employees(emp_id)
        on delete cascade
);

create table reports_to (
    subord_id int not null,
    super_id int not null,

    foreign key (subord_id)
        references employees(emp_id),
    foreign key (super_id)
        references employees(emp_id)

);
```

1.2

```
CREATE ASSERTION Total CHECK (
    (SELECT COUNT(*)
    FROM employees
    group by emp_id ) =
    (SELECT COUNT(distinct emp_id)
    FROM works_in )

);
```

1.3

```
create table employees (
    emp_id int not null default 101,
    name varchar(100),
    surname varchar(100),
    salary int,
    gender varchar(100),
    primary key (emp_id),
    CHECK (salary >= 36000 )
);
```

```
create table departments (
    dept_id int not null,
    location varchar(100),
    name varchar(100),
    manager_id int not null,
    primary key (dept_id),
    foreign key (manager_id)
        references employees(emp_id)
        on delete set default,
    CHECK (name = "%location"
        or name = "location%")
);
```

1.4

```
CREATE TRIGGER ChangeState
    AFTER UPDATE OF budget ON project
    REFERENCING
    OLD ROW AS OldTuple
    NEW ROW AS NewTuple
    FOR EACH ROW
    WHEN (OldTuple.price > NewTuple.price)
     UPDATE project
     SET state = 'Unsuccesfull'
     WHERE project_id = OldTuple.project_id
```

2

2.1

Given E/R means that:
A given product from a given store can be R(Lets say this Purchase) by at most one person.
So that;
There are 100 product and 5 stores. To take greatest value of tuples we can calculate that A person can purchase 500 different ways. (100(product)*5(store)) since any product-store tuple should be purchase by at most one person.
The answer is **500**.

2.2

Given E/R means that:
A given product can be purchase by a given person from at most one store.
And a given product can be purchase by a given person from at most one sales person.
So that;

For the first argument, 100*1000*10
For the second argument, 100*1000*5
To calculate maximum number of tuples we should take minimum of two argument given above.
The answer is **500000**.

3

3.1

1	$CB \to F$	Given
2	$B \to E$	Given
3	$CB \to CE$	Augmentation of 2 and C
4	$CB \to E$	Decomposition of 3
5	$CB \to FE$	Union of 1 and 4
6	$FE \to G$	Given
7	$CB \to G$	Transitivity of 5 and 6

3.2

1	$A \to C$	Given
2	$CB \to F$	Given
3	$AB \to CB$	Augmentation of 1 and B
4	$AB \to F$	Transitivity of 2 and 3
5	$B \to E$	Given
6	$AB \to AE$	Augmentation of 5 and A
7	$AB \to E$	Decomposition of 6
8	$AB \to EF$	Union of 4 and 7

4

4.1

Since A and F does not appear in the right side of given functional dependencies $\{A, F\}$ is the key and $\forall X$ while $\{A, F\} \subset X$ are superkey.

4.2

No since in BCNF;
lemma: $\forall X$ attribute should be either $X+ = \{X\}$ or $X+ = \{all\_attributes\}$.

There are multiple violations of BCNF like $A+ = \{A, B\}$ acoocrding to the lemma given above.

4.3

.— **R1:** $A \to B$ $\{A, B\}$ BCNF —
.— **R2:** $\{A, D, E, F, G\}$ not BCNF —
. — **R2.1:** $F \to D$ $\{F, D\}$ BCNF —
. — **R2.2:** $\{A, F, E, G\}$ not BCNF —
. — **R2.2.1:** $E \to G$ $\{E, G\}$ BCNF —
. — **R2.2.2:** $\{A, F, E\}$ not BCNF —
. — **R2.2.2.1:** $\{F \to E\}$ $\{F, E\}$ BCNF —

. — **R2.2.2.2:** $\{A, F\}$ BCNF there is no FD in this set and all closure sets are trivial —

4.4

i. No since we cannot preserve $AC \rightarrow D$.
ii. No all BCNF decompositions are loseless decompositions.

5

5.1

Functional dependencies i have found and used for BCNF decomposition given in below with an order.
$A \rightarrow E$ $C \rightarrow A$ $C \rightarrow B$

```
Select count(distinct E)
from bigTable
group by A;
```

```
Select count(distinct A)
from bigTable
group by C;
```

```
Select count(distinct B)
from bigTable
group by C;
```

These all are bad dependencies since the only key is $\{C, D\}$. Since they are sufficient to decompose DB according to BCNF i didn't try other FDs.

5.2

For the first table given. Loaded data inside it with the help of DBeaver.

```
create table if not exists bigTable (
    A varchar(100) not null,
    B varchar(100) not null,
    C int not null,
    D int not null,
    E varchar(100) not null,
    primary key (C,D));
```

```
CREATE TABLE if not exists AE (
    A varchar(100) not null,
    E varchar(100) not null,
    primary key (A));
```

```
CREATE TABLE if not exists CA (
    C int not null,
    A varchar(100) not null,
    primary key (C)
    );
```

```
CREATE TABLE if not exists CB (
    C int not null,
    B varchar(100) not null,
```

```
    primary key (C)
);

CREATE TABLE if not exists CD (
    C int not null,
    D int not null
);
```

5.3

```
INSERT INTO AE
SELECT distinct A, E FROM bigTable;

INSERT INTO CA
SELECT distinct C, A FROM bigTable;

INSERT INTO CB
SELECT distinct C, B FROM bigTable;

INSERT INTO CD
SELECT C, D FROM bigTable;
```