1

a) Employee (emp_id, name, surname, salary, gender)

Department (dept_id, name, location, emp_id)

Project (project_id, dept_id, state, due_date, budget)

WorksIn (dept_id, emp_id)

Reports_to (subordinate_emp_id, supervisor_emp_id)

```
CREATE TABLE Employee (
    emp_id VARCHAR(10),
    name    VARCHAR(20),
    surname VARCHAR(20),
    Salary  REAL,
    gender  VARCHAR(10),
    PRIMARY KEY (emp_id))

CREATE TABLE Department (
    dept_id VARCHAR(10),
    name    VARCHAR(20),
    location VARCHAR(20),
    emp_id VARCHAR(10) DEFAULT '101',
    PRIMARY KEY (dept_id),
    FOREIGN KEY (emp_id) REFERENCES Employee (emp_id) ON DELETE SET DEFAULT)

CREATE TABLE Project (
    project_id VARCHAR(10),
    dept_id  VARCHAR(10),
    State    VARCHAR(15),
    due_date DATETIME,
    budget   REAL,
    PRIMARY KEY (project_id, dept_id),
    FOREIGN KEY (dept_id) REFERENCES Department (dept_id))

CREATE TABLE WorksIn
    dept_id VARCHAR(10),
    emp_id VARCHAR(10),
    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 ReportsTo (
    subordinate_emp_id VARCHAR(10),
    supervisor_emp_id VARCHAR(10),
    PRIMARY KEY (subordinate_emp_id, supervisor_emp_id),
    FOREIGN KEY (subordinate_emp_id) REFERENCES Employee (emp_id),
    FOREIGN KEY (supervisor_emp_id) REFERENCES Employee (emp_id))
```

b) CREATE ASSERTION Total
CHECK (
NOT EXISTS (
SELECT Employee.emp-id
FROM Employee, WorksIn
WHERE Employee.emp-id = WorksIn.emp-id
GROUP BY Employee.emp-id
HAVING COUNT(*) < 1))

c) • CHECK (Employee.Salary > 36000)

• CHECK (Department.name LIKE '%' || Department.location || '%')

d) CREATE TRIGGER UnsuccessfulProjects
AFTER UPDATE OF budget ON Project
REFERENCING
OLD ROW AS oldRow
NEW ROW AS newRow
FOR EACH ROW
WHEN (newRow.budget < oldRow.budget)
UPDATE Project
SET state = 'Unsuccessful'
WHERE project-id = oldRow.project-id
AND dept-id = oldRow.dept-id

2 a) We can read the relation as "each ~~product~~ is sold at each store to at most one person.". Thus, the maximum number of tuples is equal to the number of rows of Product multiplied by the number of rows of Store.

$100 \times 5 = \underline{500 \text{ tuples}}$

b) We can consider it as 2 sub relation.

1) Each SalesPerson Sales each product to each customer in at most one store.

$\underbrace{10}_{\text{number of Salesperson}} \times \underbrace{100}_{\text{number of products}} \times \underbrace{1000}_{\text{number of customers}} = 1.000.000$

2) Each customer buys each product in each store to at most one SalesPerson.

$\underbrace{1000}_{\text{number of customers}} \times \underbrace{100}_{\text{number of products}} \times \underbrace{5}_{\text{number of Stores}} = 500.000$

Then, we take minimum of them:

$\min\{1.000.000, 500.000\} = \underline{500.000 \text{ tuples}}$

3

1. $A \rightarrow C$
2. $B \rightarrow E$
3. $CB \rightarrow F$
4. $FE \rightarrow G$
5. $FG \rightarrow AH$

a)

6. $CB \rightarrow B$ Trivial rule
7. $CB \rightarrow E$ Transitivity on 6,2
8. $CB \rightarrow FE$ Split/combine on 7,3
9. $CB \rightarrow G$ Transitivity on 8,4

b)

6. $AB \rightarrow A$ Trivial rule
7. $AB \rightarrow C$ Transitivity on 6,1
8. $AB \rightarrow B$ Trivial rule
9. $AB \rightarrow CB$ Split/combine on 7,8
10. $AB \rightarrow F$ Transitivity on 9,3
11. $AB \rightarrow E$ Transitivity on 8,2
12. $AB \rightarrow EF$ Split/combine on 11,10

Scanned with CamScanner

4

$A \to B$

$CD \to E$

$F \to D$

$E \to G$

$AC \to D$

$D \to C$

$R = \{A, B, C, D, E, F, G\}$

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

$\{B\}^+ = \{B\}$

$\{C\}^+ = \{C\}$

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

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

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

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

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

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

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

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

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

$\{A, G\}^+ = \{A, G, B\}$

$\{B, C\}^+ = \{B, C\}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A subset of all the elements in the relation is a key if its closure includes all the elements. Since I found a key with 2 elements, I don't need to go on with 3 elements as I would find a proper subset with 3 elements, it will be a superkey not a key.

$\{A, F\}$ is the only key of R.

b) R is not in BCNF, because if we compare left hand sides of all dependencies with the key $\{A, F\}$, we see that all the functional dependencies $A \to B$, $CD \to E$, $F \to D$, $E \to G$, $AC \to D$ and $D \to C$ violate BCNF condition (none of the left hand sides of all functional dependencies is equal to AF).

c) Decompose R into two relations such that $R1 = \{A, B\}$ and $R2 = \{A, C, D, E, F, G\}$

$R = \{A, B, C, D, E, F, G\}$

$A \to B$ violates

$R1 = \{A, B\}$ $\qquad$ $R2 = \{A, C, D, E, F, G\}$

R1 in BCNF since

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

$\{B\}^+ = \{B\}$

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

$\{A\}$ is key and $A \to B$ is the only non-trivial functional dependency.

for R2 : using the clasures from part a) by subtracting B from the sets,

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

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

$\{F\}^+ = \{F, D, C, E, G\}$ $\qquad$ $F \to DCEG$

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

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

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

⋮

There are lots of functional dependencies violating BCNF. Since R2 is not in BCNF, we have to decompose it into 2 new relations.

$\{C, D\}^+ = \{\cancel{C}, \cancel{D}, E, G\}$ from part a)

$CD \to EG$ violates since left hand side is not (super) key.

$R = \{A, B, C, D, E, F, G\}$

$R1 = \{A, B\}$ in BCNF

$R2 = \{A, C, D, E, F, G\}$ $CD \to EG$ violates

$R21 = \{C, D, E, G\}$ $E \to G$ violates

$R22 = \{C, D, A, F\}$ $D \to C$ violates

$R211 = \{C, D, E\}$ in BCNF

$R212 = \{E, G\}$ in BCNF

$R221 = \{D, A, F\}$ $F \to D$ violates

$R222 = \{D, C\}$ in BCNF

$R2211 = \{A, F\}$ in BCNF

$R2212 = \{D, F\}$ in BCNF

Scanned with CamScanner

d) i) It is not dependency preserving. For example, $AC \rightarrow D$ functional dependency can't be obtained.

ii) Since we decomparited R into the relations such that all in BCNF, and BCNF decomposition is always lossless, the above decomposition is lossless.

5)

a) I checked whether I got all 1's in all rows in below queries or not to find all dependencies.

For dependency a->e,

```sql
select
	count(distinct s.e)
from
	sample s
group by
	s.a
```

For dependency c->a,

```sql
select
	count(distinct s.a)
from
	sample s
group by
	s.c
```

For dependency c->b,

```sql
select
	count(distinct s.b)
from
	sample s
group by
	s.c
```

For dependency c->e,

```sql
select
	count(distinct s.e)
from
	sample s
group by
	s.c
```

For dependency e->a,

```sql
select
	count(distinct s.a)
from
	sample s
group by
	s.e
```

For dependency ab->c,

```sql
select
	count(distinct s.c)
from
	sample s
group by
	s.a, s.b
```

For dependency be->c,

```sql
select
	count(distinct s.c)
from
	sample s
group by
	s.b, s.e
```

b)

```sql
create table if not exists a_e_table(
	a varchar(5),
	e varchar(10),
	primary key(a)
);


create table if not exists a_c_table(
	a varchar(5),
	c int,
	primary key(c),
	foreign key(a) references a_e_table(a)
);


create table if not exists b_c_table(
	b varchar(5),
	c int,
	primary key(c),
	foreign key(c) references a_c_table(c)
);


create table if not exists c_d_table(
	c int,
	d int,
	primary key(c, d),
	foreign key(c) references a_c_table(c)
);
```

c)

```sql
insert into a_e_table(a,e)
	select distinct s.a, s.e
	from sample s


insert into a_c_table(a,c)
	select distinct s.a, s.c
	from sample s


insert into b_c_table(b,c)
	select distinct s.b, s.c
	from sample s


insert into c_d_table(c,d)
	select distinct s.c, s.d
	from sample s
```