CS2102 Final Exam (AY2019/20 Semester 2)

Question 10

(3 marks) Consider an application about the courses that are taught by professors.

- Each **professor** is identified by **pid** and has a name given by **pname**. Each professor specializes in one or more areas.
- Each **area** is identified by **aid** and has a name given by **aname**.
- Each course is identified by **cid** and has a name given by **cname**. Each course belongs to exactly one area (identified by **aid**).
- Each professor teaches 0 or more courses.
- If a professor P teaches a course C, then the area associated with C must be an area that P specializes in.
- Each course is taught by 0 or more professors.
- All identifiers (i.e., pid, aid, and cid) have integer domains.
- All names (i.e., pname, aname, and cname) have text domains.
- The values of attribute **aid** must be non-null

The following is a database schema designed for this application.

```
CREATE TABLE Profs (
    pid     integer,
    pname   text,
    PRIMARY KEY (pid)
);

CREATE TABLE Areas (
    aid     integer,
    aname   text,
    PRIMARY KEY (aid)
);

CREATE TABLE Courses (
    cid     integer,
    cname   text,
    aid     integer NOT NULL,
    PRIMARY KEY (cid),
    UNIQUE (cid, aid),
    FOREIGN KEY (aid) REFERENCES Areas
);

CREATE TABLE Specializes (
    pid     integer,
    aid     integer,
    PRIMARY KEY (pid, aid),
```

```
    FOREIGN KEY (pid) REFERENCES Profs,
    FOREIGN KEY (aid) REFERENCES Areas
);

CREATE TABLE Teaches (
    pid     integer,
    cid     integer,
    aid     integer NOT NULL,
    PRIMARY KEY (pid, cid),
    FOREIGN KEY (pid) REFERENCES Profs,
    FOREIGN KEY (pid, aid) REFERENCES Specializes
);
```

Identify all the application constraints that **are not** enforced by the above database schema.

If your answer is that all the constraints are enforced by the database schema, write down "None"; otherwise, write down each constraint that is not enforced.

Question 11

(3 marks) Consider the following database schema consisting of three tables.

```
CREATE TABLE Employees (
    eid     integer, -- employee identifier
    ename   text, -- employee name
    PRIMARY KEY (eid)
);

CREATE TABLE Projects (
    pid     integer, -- project identifier
    budget  integer, -- project budget
    PRIMARY KEY (pid)
);

CREATE TABLE Works (
    eid     integer, -- employee identifier
    pid     integer, -- project identifier
    week    integer  -- week that eid works on pid
            CHECK (week >= 1 AND week <= 52),
    hours   integer  -- hours that eid works on pid during specific week
            NOT NULL CHECK (hours > 0),
    PRIMARY KEY (eid,pid,week),
    FOREIGN KEY (eid) REFERENCES Employees
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    FOREIGN KEY (pid) REFERENCES Projects
```

```
            ON DELETE CASCADE
            ON UPDATE CASCADE
);
```

- The **Employees** table records information about employees (identified by **eid** with name given by **ename**).
- The **Projects** table records information about projects (identified by **pid** with budget given by **budget**).
- The **Works** table records the number of hours (given by attribute **hours**) worked by an employee (identified by **eid**) on a project (identified by **pid**) during a specific week (given by **week**).
- Each employee works on 0 or more projects.
- Each project is worked on by 0 or more employees.

The following trigger and trigger function are created to enforce the **constraint** that each employee must not exceed 40 working hours for each week (over all projects that the employee worked on for each week).

```
CREATE OR REPLACE function enforce_total_weekly_hours() RETURNS TRIGGER  AS $trigger$
DECLARE
    week_num  integer;
BEGIN
    SELECT week  INTO week_num
        FROM    Works W
        WHERE   eid = NEW.eid
        AND     (SELECT SUM(hours)
                FROM    Works
                WHERE   eid = W.eid
                AND week = W.week
                ) > 40;
    IF FOUND THEN
        RAISE exception 'Employee % exceeds 40 work hours in week %', NEW.eid, week_num;
    END IF;
    RETURN NULL;
END;
$trigger$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS works_trigger ON Works CASCADE;
CREATE CONSTRAINT TRIGGER works_trigger
    AFTER INSERT OR UPDATE OF hours ON Works
    DEFERRABLE INITIALLY DEFERRED
    FOR EACH ROW
    EXECUTE FUNCTION
    enforce_total_weekly_hours();
```

This question consists of two parts.

First Part: State whether the above trigger code correctly enforces the specified

constraint. Your answer must be either "Yes" or "No".

Second Part: If your answer to the first part is "No", describe what needs to be modified in the code to correctly enforce the specified constraint. If your answer to the first part is "Yes", leave your answer for the second part blank.

ANSWER SYNTAX: The two parts of your answer must be separated by a line consisting of only the hex character (i.e., #).

Question 12

(3 marks) Questions 12 to 15 are based on the schema **R(A,B,C,D,E)** with FDs **F**:

```
F = {
    A -> C,
    B -> A,
    C -> B,
    D -> A
}
```

Consider the following decomposition of R: **{ R1(A,B,C), R2(B,D,E), R3(C,D) }**.

This question consists of two parts.

First Part: State whether the decomposition is a lossless-join decomposition. Your answer should be either "Yes" or "No".

Second Part: Show the working for your answer.

ANSWER SYNTAX: The two parts of your answer must be separated by a line consisting of only the hex character (i.e., #).

Question 13

(3 marks) Questions 12 to 15 are based on the schema **R(A,B,C,D,E)** with FDs **F**:

```
F = {
    A -> C,
    B -> A,
    C -> B,
    D -> A
}
```

Consider the following decomposition of R: **{ R1(A,B,C), R2(B,D,E), R3(C,D) }**.

This question consists of two parts.

First Part: State whether the decomposition is a dependency-preserving decomposition. Your answer should be either "Yes" or "No".

Second Part: Show the working for your answer.

ANSWER SYNTAX: The two parts of your answer must be separated by a line consisting of only the hex character (i.e., #).

Question 14

(3 marks) Questions 12 to 15 are based on the schema **R(A,B,C,D,E)** with FDs **F**:

```
F = {
    A -> C,
    B -> A,
    C -> B,
    D -> A
}
```

Consider the following decomposition of R: **{ R1(A,B,C), R2(B,D,E), R3(C,D) }**.

This question consists of two parts.

First Part: State whether the decomposition is in BCNF. Your answer should be either "Yes" or "No".

Second Part: Show the working for your answer.

ANSWER SYNTAX: The two parts of your answer must be separated by a line consisting of only the hex character (i.e., #).

Question 15

(3 marks) Questions 12 to 15 are based on the schema **R(A,B,C,D,E)** with FDs **F**:

```
F = {
    A -> C,
    B -> A,
    C -> B,
    D -> A
}
```

This question consists of two parts.

First Part: Write down a BCNF decomposition of R. If your answer for question 14 is "Yes", your answer for this question must be different from **{ R1(A,B,C), R2(B,D,E), R3(C,D) }**.

Second Part: Show the working for your answer.

ANSWER SYNTAX: You should enter your answer and working as two separate parts as follows:

- First, type your answer. If your answer is the decomposition {R1(A,B), R2(A,C), R3(A,D,E)}, type the answer on a single line as
  ab; ac; ade
  You may also type the answer with one schema per line as follows:
  ab;
  ac;
  ade
  Note that for each schema, you just need to type its attributes (without the curly brackets, parentheses and table names). Separate two schemas with a semicolon character between them.
- Second, type a new line with just the hex character (i.e., #). This line serves to separate your answer and your working.
- Finally, type in your working for the answer (after the line with the hex character).

Question 16

(3 marks) Questions 16 and 17 are based on the following database schema for a shopping application.

```
CREATE TABLE Customers (
    cid             integer PRIMARY KEY,
    cname           text NOT NULL
);

CREATE TABLE Items (
    iid             integer PRIMARY KEY,
    iname           text NOT NULL,
    price           integer NOT NULL CHECK (price > 0)
);


CREATE TABLE Orders (
    oid             integer PRIMARY KEY,
    cid             integer NOT NULL REFERENCES Customers,
    month           integer NOT NULL CHECK (month >= 1 AND month <= 12)
);

CREATE TABLE OrderItems (
    oid             integer REFERENCES Orders,
    iid             integer REFERENCES Items,
    qty             integer NOT NULL CHECK (qty > 0),
    PRIMARY KEY (oid,iid)
);
```

- **Customers** records information about customers (identified by **cid**) with name given by **cname**.
- **Items** records information about shopping items (identified by **iid**) with name given by **iname** and unit price given by **price**.
- **Orders** records orders (identified by **oid**) made by customers (identified by **cid**) in the month given by **month**.
- Each record in **OrderItems** specifies the quantity (given by **qty**) of an item (identified by **iid**) bought in an order (identified by **oid**).
- A customer may place multiple orders in a month.
- Each order consists of 1 or more items.
- The quantity for each ordered item must be at least one.

Given an order, its **order price** is computed by the sum of (price X qty) over all items bought in that order.

Write a single CREATE VIEW SQL statement to find all customers who have placed some order where the order price is at least $200. Your answer view **must not** contain any duplicate record.

ANSWER SYNTAX: Your SQL view must conform to the schema shown below, where cid is a customer identifier. Your answer view may use a single CTE statement (defining possibly multiple temporary tables).

CREATE VIEW v16 (cid) AS ;

Question 17

(3 marks) Questions 16 and 17 are based on the following database schema for a shopping application.

```
CREATE TABLE Customers (
    cid             integer PRIMARY KEY,
    cname           text NOT NULL
);

CREATE TABLE Items (
    iid             integer PRIMARY KEY,
    iname           text NOT NULL,
    price           integer NOT NULL CHECK (price > 0)
);


CREATE TABLE Orders (
    oid             integer PRIMARY KEY,
    cid             integer NOT NULL REFERENCES Customers,
    month           integer NOT NULL CHECK (month >= 1 AND month <= 12)
);
```

```
CREATE TABLE OrderItems (
    oid             integer REFERENCES Orders,
    iid             integer REFERENCES Items,
    qty             integer NOT NULL CHECK (qty > 0),
    PRIMARY KEY (oid,iid)
);
```

- **Customers** records information about customers (identified by **cid**) with name given by **cname**.
- **Items** records information about shopping items (identified by **iid**) with name given by **iname** and unit price given by **price**.
- **Orders** records orders (identified by **oid**) made by customers (identified by **cid**) in the month given by **month**.
- Each record in **OrderItems** specifies the quantity (given by **qty**) of an item (identified by **iid**) bought in an order (identified by **oid**).
- A customer may place multiple orders in a month.
- Each order consists of 1 or more items.
- The quantity for each ordered item must be at least one.

Given an order, its **order price** is computed by the sum of (price X qty) over all items bought in that order.

Let **TotalOrderPrice(C,M)** denote the sum of the order prices of all the orders placed by a customer C in month M. Note that TotalOrderPrice(C,M) = 0 if C has no order in month M.

We say that a customer C is a **good customer** if C satisfies the following three conditions:

- C has at least one order,
- For each month M where TotalOrderPrice(C,M) > 0, TotalOrderPrice(C,M) >= 200, and
- For each month M where TotalOrderPrice(C,M) > 0, TotalOrderPrice(C,M) >= TotalOrderPrice(C',M) for every other customer C'.

Write a single CREATE VIEW SQL statement to find all good customers.
Your answer view **must not** contain any duplicate record.

ANSWER SYNTAX: Your SQL view must conform to the schema shown below, where cid is a customer identifier. Your answer view may use a single CTE statement (defining possibly multiple temporary tables).

CREATE VIEW v17 (cid) AS ;