CS143: Database Systems
Homework #3

1. Assume the following tables for this problem:

```
ComputerProduct(manufacturer, model, price)
Desktop(model, speed, ram, hdd)
Laptop(model, speed, ram, hdd, weight)
```

   A computer product is either a desktop or a laptop.

   (a) Using a `CHECK` constraint on the `Laptop` table, express the constraint that a laptop cannot have weight larger than 5kg. You do not need to show the entire `CREATE TABLE` statement. Show only the `CHECK` constraint part in the `CREATE TABLE` statement.

   ANSWER:

```
CHECK (weight<=5)
```

   (b) Write a trigger to replace the `CHECK` constraint in (a), so that when trying to add a laptop with weight larger than 5kg, the tuple is still inserted, but the value of the "weight" attribute is set to `NULL`.

   ANSWER:

```
CREATE TRIGGER T
AFTER INSERT ON Laptop
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN nrow.weight>5
BEGIN
UPDATE Laptop SET weight=NULL WHERE model=nrow.model
END
```

2. In this problem you express a referential integrity constraint using a general SQL assertion. Assume that there are two tables `R(a)` and `S(a)`. Using a general SQL assertion, express that `S.a` is a foreign key referencing `R.a`. That is, there should not be any `S.a` value that does not appear in `R.a`. Note that a general SQL assertion is not attached to a particular table. Therefore, whenever a SQL modification statement is executed that may potentially violate the assertion, the DBMS checks the result of the statement and rejects the statement if it causes violation. This behavior is the same as the default semantics of a foreign-key constraint.

   ANSWER:

```
CREATE ASSERTION FKey CHECK (
NOT EXISTS (SELECT * FROM S WHERE a NOT IN (SELECT a FROM R))
```

3. Consider the table `R(A, B)`, which currently has only one tuple (1,0). Assume that the following trigger has already been created for the database.

```
CREATE TRIGGER Times2
AFTER UPDATE ON R
   REFERENCING NEW ROW AS n
   FOR EACH ROW
WHEN (n.B < 5)
BEGIN
   UPDATE R SET B=B*2 WHERE A=n.A;
   INSERT INTO R VALUES(100, 0);
END
```

List all tuples in the table `R` after the following update statement is executed:

```
UPDATE R SET B=2 WHERE A=1
```

ANSWER:
(1,8), (100,0), (100,0)

4. You are the DBA for the VeryFine Toy Company and create a relation called Employees(ename, dept, salary). For authorization reasons, you also define views EmployeeNames(ename) and DeptInfo(dept, avgsalary). The second column lists the average salary for each department.

   (a) Show the view definition statements for EmployeeNames and DeptInfo.

   ANSWER:
   ```
   CREATE VIEW EmployeeNames AS
   SELECT ename FROM Employees

   CREATE VIEW DeptInfo AS
   SELECT dept, AVG(salary) avgsalary
   FROM Employees
   GROUP BY dept
   ```

   (b) You want to authorize your secretary, Mike, to fire people (you will probably tell him whom to fire, but you want to be able to delegate this task), to check on who is an employee, and to check on average department salaries. What is the minimum set of privileges you should grant to Mike?

   ANSWER:
   SELECT, DELETE on EmployeeNames. SELECT on DeptInfo.

   (c) Continuing with the preceding scenario, you do not want your secretary to be able to look at the salaries of individuals. Does your answer to the previous question ensure this? Be specific: Can your secretary possibly find out salaries of some individuals (depending on the actual set of tuples), or can your secretary always find out the salary of any individual he wants to?

   ANSWER:
   Yes, if he really wants to, he can get the salary of any employee. For example, to

get the salary of John, he first delete all tuples in EmployeeNames except John's and look at the DeptInfo table. Since there is only one employee, John, the avgsalary of the single tuple in DeptInfo is John's salary.

(d) Give an example of a view update on the preceding views that cannot be translated into an update to Employees.

ANSWER:
UPDATE DeptInfo SET avgsalary = 10000 WHERE dept='Toy'

(e) You decide to go on an extended vacation, and to make sure that emergencies can be handled, you want to authorize your boss Joe to read and modify the Employees relation and the EmployeeNames relation (and Joe must be able to delegate authority, of course, since he is too far up the management hierarchy to actually do any work). Show the appropriate SQL statements. Can Joe read the DeptInfo view?

ANSWER:
GRANT SELECT, UPDATE ON Employees TO Joe WITH GRANT OPTION
GRANT SELECT, UPDATE ON EmployeeNames TO Joe WITH GRANT OPTION
No, Joe cannot because we did not give him SELECT privilege on DeptInfo. Even through Joe can SELECT on the base table of DeptInfo, it does not give him the SELECT privilege on DeptInfo. This should be given explicitly by the owner of DeptInfo.

(f) After you come back from your vacation, you realize that Joe has been quite busy. He has defined a view called AllNames using the view EmployeeNames, defined another relation called StaffNames that he has access to (but you cannot access), and given his secretary James the right to read from the AllNames view. James has passed this right on to his friend Susan. You decide that, even at the cost of annoying Joe by revoking some of his privileges, you simply have to take away some of Joe's privileges to prevent James and Susan from seeing your data. What REVOKE statement would you execute? What views remain after you execute this statement?

ANSWER:
REVOKE SELECT ON EmployeeNames FROM Joe CASCADE
Since AllNames is dependent on EmployeeNames, and Joe, the owner of AllNames, does not have the right to read from EmployeeNames, AllNames will be automatically dropped after the above REVOKE. If we want to prevent Joe from creating another view on Employees and share it with others, we will have to ''REVOKE SELECT ON Employees FROM Joe CASCADE'' as well.