Homework #2
Due: February 20, 2020

1. Practice the 20+ SQL examples discussed in class.

2. Consider the following relational schema:

   Employee(eid: integer, ename: string, age: integer, salary: real)

   Department(did: integer, dname: string, budget: real, mgrid: integer)

   Works(eid: integer, did: integer, pct_time: integer)

An employee can work in more than one department; the `pct_time` field of the Works
relation shows the percentage of time that a given employee works in a given
department.

a) (25 pts.) Write an SQL query to find the IDs of managers who control the largest
amounts. (Hint: Create a table in the FROM clause to compute the total budget for
each manager)

   ```sql
   SELECT DISTINCT tempD.managerid
   FROM (SELECT DISTINCT D.managerid, SUM (D.budget) AS tempBudget
         FROM Dept D
         GROUP BY D.managerid ) AS tempD
   WHERE tempD.tempBudget = (SELECT MAX (tempD.tempBudget)
                                 FROM tempD)
   ```

b) (30 pts.) Write an SQL trigger to express the following constraint: “Whenever an
employee is given a raise, the manager’s salary must be increased to be at least as
much.” (increasing a manager’s salary to be equal to the employee who received
the raise, if the manager’s salary is less than the employee’s new salary)

   ```sql
   CREATE TRIGGER GiveRaise AFTER UPDATE ON Emp
   WHEN old.salary < new.salary
   FOR EACH ROW
   BEGIN
     UPDATE Emp M
     SET M.Salary = new.salary
     WHERE M.salary < new.salary
     AND M.eid IN (SELECT D.managerid
                     FROM Emp E, Works W, Dept D
                     WHERE E.eid = new.eid
                     AND E.eid = W.eid
                     AND W.did = D.did);
   END
   ```
3. (25 pts.) Consider the following relational schema:

- Faculty(fid: integer, fname: string, deptid: integer)
- Student(snum: integer, sname: string, major: string, level: string, age: integer)
- Class(name: string, meets_at: time, room: string, fid: integer)
- Enrolled(snum: integer, cname: string)

Enrolled has one record per student-class pair such that the student is enrolled in the class. Write an SQL assertion for the following integrity constraint: “Every faculty member must teach at least two courses.”

```sql
CREATE ASSERTION TeachTwo
CHECK ( ( SELECT COUNT(*)
                 FROM Faculty F, Class C
                 WHERE F.fid = C.fid
             GROUP BY C.fid
           HAVING COUNT(*) < 2 ) = 0 )
```

4. (20 pts.) We discuss in class a simple way to compute the join operation as illustrated below:

- **Outer relation**
- **Inner relation**

```
foreach tuple r in R do
  foreach tuple s in S where r_i == s_j do
    add <r, s> to result
```

This scheme is very computationally expensive because each tuple of the relation R must be compared with each tuple of the relation S. This strategy incurs substantial disk access. Design a more efficient join algorithm, in which each tuple of relation R needs to be compared with only a small subset of the S tuples.
Answer:

1. Scan table $R$. For each tuple, it is assigned to Data Bucket $R_b$ if dividing the value of the join attribute (e.g., SSN) by $k$ gives $b$ as the remainder. This strategy distributes the tuples of $R$ among $k$ data buckets, $R_0, \ldots, R_{k-1}$.
2. Repeat the above process for table $S$ to create $S_0, \ldots, S_{k-1}$
3. For $i = 0$ to $k-1$, compute $R_i \bowtie S_i$

NOTE: The cost of Step 1 is approximately the cost of reading and writing table $R$. The cost of Step 2 is approximately the cost of reading and writing table $S$. The cost of Step 3 is approximately the cost of reading and writing tables $R$ and $S$. Since $R$ and $S$ have $M$ pages and $N$ pages of tuples, respectively, the cost of the above join algorithm is $2M + 2N + 2(M+N) = 4(M+N)$ disk I/O’s assuming the computation is I/O-bounded. This is a lot more efficient than $M \cdot N$ I/O’s using the basic technique described in the question.