CSE 4/560: Project #2 (due 04/22/19)

Problem 1 (10 pts)

You are given the following relational schema (keys underlined):

`Employee(`<u>`Ename`</u>`,Salary)`
`Project(`<u>`Pname`</u>`,Agency,Budget)`
`Assign(`<u>`Pname,Ename`</u>`).`

Foreign keys:

- `Ename` in `Assign` references `Employee(Ename)`.
- `Pname` in `Assign` references `Project(Pname)`.

Write the following queries in SQL:

- S_1: Find the employees assigned to exactly one project.
- S_2: Find the employees whose salary is greater than Mark's salary (Mark is an employee).
- S_3: For every project, compute the number of projects whose budget is higher.
- S_4: Find the projects with the budget lower than the average project budget at the same agency.

Which of the above queries has an equivalent relational algebra formulation? Explain your answer. For the queries that have an equivalent relational algebra formulation provide this formulation.

Problem 2 (5 pts)

You are supposed to represent directed graphs in SQL. A directed graph consists of nodes and directed edges connecting the nodes. A node y is reachable from another node x if there is a path from x to y; directly reachable if there is a path from x to y consisting of a single edge.

- Define an appropriate SQL schema.
- Write the following queries in SQL2 or SQL3:
  - For every node, compute the number of the nodes directly reachable from the node (SQL2).
  - For every node, compute the number of the nodes reachable, directly or indirectly, from that node (SQL3).
  - Compute the set of all the nodes not directly reachable from the node A (SQL2).
  - Compute the set of all the nodes not reachable, directly or indirectly, from the node A (SQL3).
- Construct a small database instance, test your queries over it and report the results.

Problem 3 (Extra credit: 5 pts)

You are given a relation R with N columns of the same type. Write an SQL query that returns the tuples (perhaps more than one) having the *minimum number of different values*. The solution should have size polynomial in N and use aggregation in an essential way.

Examples:

1. $R_1 = \{t_1 : (a, a, b), (t_2 : (b, a, c)\}$. The result is t_1.
2. $R_2 = \{t_1 : (a, a, b), (t_2 : (b, a, c), t_3 : (b, b, b)\}$. The result is t_3.

A part of the solution is devising a way to uniquely identify tuples without introducing the notion of a tuple identifier.