# CMPSCI 645: Homework 4 - Part A

Normalization and DB Theory

Due: Monday, April 9 2018, 11:59pm

Name: Mili Shah

| Question | Points | Score |
|---|---|---|
| 1 | 15 | |
| 2 | 20 | |
| 3 | 20 | |
| 4 | 5 | |
| Total: | 60 | |

Please turn in this homework electronically, as a PDF, through Gradescope. You may handwrite your solutions, and then scan the document, or type directly into the PDF form. Make sure the PDF you upload includes all pages (including this front page).

1. **Normalization** [15 points]

   Consider the following relational schema and set of functional dependencies.

$$R(A,B,C,D,E) \qquad \begin{array}{l} CD \rightarrow E \\ B \rightarrow A \end{array}$$

   (a) [5 points] List **all** superkey(s) for this relation. Justify your answer in terms of functional dependencies and closures.

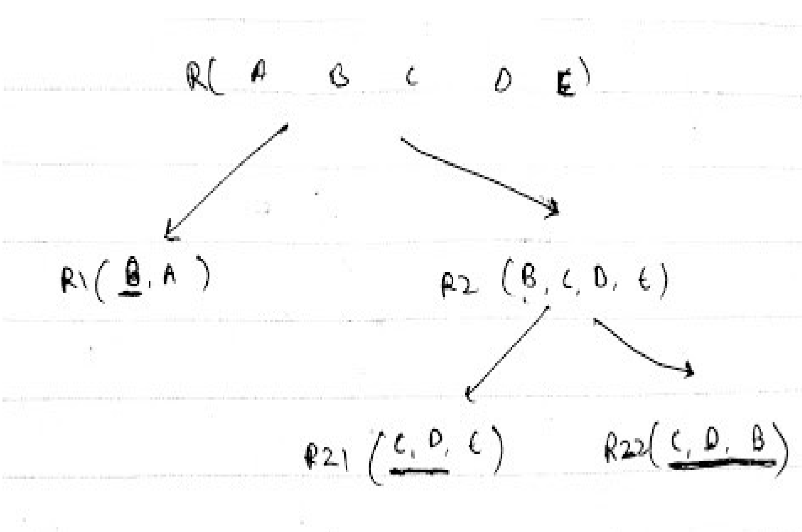
   Using Closure algorithm :
   {C, D}+ = {C, D, E}
   {B}+ = {A, B}

   So, superkeys :
   {B, C, D}+ = {A, B, C, D, E} using Closure algorithm.
   Therefore, {A, B, C, D}+ = {A, B, C, D, E}
   and {B, C, D, E}+ = {A, B, C, D, E}
   and {A, B, C, D, E}+ = {A, B, C, D, E}

   (b) [2 points] Which of these superkeys form a key (i.e., a minimal superkey) for this relation?

   Minimal superkey : {B, C, D}

   (c) [8 points] Decompose R into BCNF. Show your work for partial credit. Your answer should consist of a list of table names and attributes and an indication of the keys in each table (either underline the corresponding attributes, or explicitly state the keys).




   Final relations:
   R1(B, A) : key - B
   R21(C, D, E) : key - C,D
   R22(C, D, B) : key - C,D,B

2. **Datalog** [20 points]

   (a) [10 points] Consider the following two datalog programs computing the transitive closure:

```
P1:
    T(x,y) :- R(x,y)
    T(x,y) :- T(x,z),R(z,y)

P2:
    T(x,y) :- R(x,y)
    T(x,y) :- T(x,z),T(z,y)
```

   Suppose relation $R$ represents a graph that consists of a single path:

$$R(a_1, a_2), R(a_2, a_3), \ldots, R(a_{n-1}, a_n)$$

   Thus, the transitive closure $T$ computed by both programs consists of all $\binom{n}{2}$ ground facts of the form $T(a_i, a_j)$, for $1 \leq i < j \leq n$. Assume that we evaluate both programs using the semi-naive evaluation algorithm.

   i. For a fixed $m = 1, \ldots n-1$, how many times will the fact $T(a_1, a_{m+1})$ be discovered by `P1`?

   i. 1

   Explain your answer:

   The fact T(a(1), a(m+1)) will be discovered by P1 only once by this operation:
   T(a(1), a(m+1)) :- T(a(1), a(m)), R(a(m), a(m+1))

   ii. How many times will the fact $T(a_1, a_{m+1})$ be discovered by `P2`?

   ii. m-1

   Explain your answer:

   The fact T(a(1), a(m+1)) will be discovered by P2 (m-1) times, once by each of (m-1) operations belonging to the following class:
   T(a(1), a(m+1)) :- T(a(1), a(i)), T(a(i), a(m+1)), where i belongs to {2, 3, ..., m}

(b) [10 points] Alice and Bob are playing a game, taking turns to move a pebble on a graph. When it is a player's turn, s/he moves the pebble from its current position at node $x$ to node $y$ only if there exists an edge from $x$ to $y$. A player wins if there are no moves that s/he can make, i.e., the pebble is on a *terminal* node at the player's turn, which is a node that has no outgoing edges.

The graph that Alice and Bob are using for their game has the property that every node $x$ has either zero, or exactly two outgoing edges $(x, y)$ and $(x, z)$. In the former case, $x$ is a *terminal* node, and it is stored in relation $T(x)$; in the latter case, we store the triple $(x, y, z)$, representing the outgoing edges of $x$ to nodes $y$ and $z$ in a relation $G$. (Thus, $x$ is a key in $G(x, y, z)$.)

Write a datalog program that computes the set of starting nodes from which Alice has a winning strategy if she plays first. That is, your program should compute a relation $A(x)$ that returns all nodes $x$ such that, if Alice starts the game on $x$ (and plays smartly!) then she is guaranteed to win the game.

```
A(x) :- T(x)
A(x) :- G(x, y, z), B(y)
A(x) :- G(x, y, z), B(z)
B(x) :- G(x, y, z), A(y), A(z)
```

3. [20 points] **Query containment**

Indicate for each pair of queries $q, q'$ below, whether $q \subseteq q'$. If the answer is **yes**, provide a homomorphism; if the answer is **no**, give a database instance $I$ on which $q(I) \not\subseteq q'(I)$.

(a)

$$q(x) \ :- \ R(x,y), R(y,z), R(z,x)$$
$$q'(x) \ :- \ R(x,y), R(y,z), R(u,v), R(v,z), R(z,u)$$

(a) yes

Explain your answer:

var(q) = {x, y, z}

var(q') = {x, y, z, u, v}

q(x) = R(x, y), R(y, z), R(z, x)

q'(x) = R(x, y), R(y, z), R(u, v), R(v, z), R(z, u)

∴ q is contained in q'

(b)

$$q(x) \ :- \ R(x,y), R(y,z), S(z,x)$$
$$q'(x) \ :- \ R(x,y), R(y,z), R(u,v), R(v,z), S(z,u)$$

(b) yes

Explain your answer:

var(q) = {x, y, z}

var(q') = {x, y, z, u, v}

q(x) = R(x, y), R(y, z), S(z, x)

q'(x) = R(x, y), R(y, z), R(u, v), R(v, z), S(z, u)

∴ q is contained in q'

(c)

$$
\begin{aligned}
q(x) &:- R(x,y), S(y,z), S(z,x) \\
q'(x) &:- R(x,y), R(y,z), R(u,v), S(v,z), S(z,u)
\end{aligned}
$$

(c) no

Explain your answer:

If R only has a tuple ('a', 'b'), and S only has tuples ('b', 'c') and ('c', 'a'), they will be joined in q(x) to give result of one tuple ('a'), but q'(x) will not join these and give no resultant tuple, because it requires the tuple ('b', 'c') to be present in R to join, which is not present in R. So, q(x) is not a subset of q'(x).

(d)

$$
\begin{aligned}
q(x,y) &:- R(x,u,u), R(u,v,w), R(w,w,y) \\
q'(x,y) &:- R(x,u,v), R(v,v,v), R(v,w,y)
\end{aligned}
$$

(d) no

Explain your answer:

If R only has tuples ('a', 'b', 'b'), ('b', 'c', 'd') and ('d', 'd', 'e'), they will be joined in q(x) to give result of one tuple ('a', 'e'), but q'(x) will not join these and give no resultant tuple, because it requires the tuple ('b', 'b', 'b') or ('d', 'd', 'd') or ('e', 'e', 'e') to be present in R to join, which is not present in R. So, q(x) is not a subset of q'(x).

4. [5 points] **Semijoin reduction**

   Find a full semi-join reduction for the query below.

$$q(x) : -R(x,y), S(y,z), T(y,u)$$

$S'(y,z) :- R(x,y), S(y,z)$

$S''(y,z) :- S'(y,z), T(y,u)$

$R'(x,y) :- R(x,y), S''(y,z)$

$T'(y,u) :- S''(y,z), T(y,u)$

$q(x) :- R'(x,y), S''(y,z), T'(y,u)$