Assignment 1

GROUP 4

Hao Truong
-Email : truongh@oregonstate.edu
-ONID : truongh

Woonki Kim
-Email : kimwoon@oregonstate.edu
-ONID : kimwoon

1. **Relational Query Languages and SQL**

(a) Return names of every employee who works in the "Hardware", "Software", and "Research" departments

For Relational Algebra:

$$\pi_{ename}\left(\sigma_{dname="Hardware"}(\text{emp} \bowtie \text{works} \bowtie \text{dept})\right)$$
$$\cap\ \pi_{ename}\left(\sigma_{dname="Software"}(\text{emp} \bowtie \text{works} \bowtie \text{dept})\right)$$
$$\cap\ \pi_{ename}\left(\sigma_{dname="Research"}(\text{emp} \bowtie \text{works} \bowtie \text{dept})\right)$$

For nonrecursive-Datalog:

Q1(y) :- emp(x, y, _, _), works(x, z, _), dept(z, "Hardware", _, _).
Q1(y) :- emp(x, y, _, _), works(x, z, _), dept(z, "Software", _, _).
Q1(y) :- emp(x, y, _, _), works(x, z, _), dept(z, "Research", _, _).

For Relational Calculus:

{e.ename | ∃e ∈ emp, ∃w1 ∈ works, ∃d1 ∈ dept (e.eid = w1.eid ∧ w1.did = d1.did ∧ d1.dname = "Hardware") ∧
∃w2 ∈ works, ∃d2 ∈ dept (e.eid = w2.eid ∧ w2.did = d2.did ∧ d2.dname = "Software") ∧
∃w3 ∈ works, ∃d3 ∈ dept (e.eid = w3.eid ∧ w3.did = d3.did ∧ d3.dname = "Research")}

For SQL:

(**SELECT DISTINCT** e.ename
FROM emp e, works w, dept d
WHERE e.eid = w.eid **AND** w.did = d.did **AND** d.dname = 'Hardware')

INTERSECT
(**SELECT DISTINCT** e2.ename
FROM emp e2, works w2, dept d2
WHERE e2.eid = w2.eid **AND** w2.did = d2.did **AND** d2.dname = 'Software')
INTERSECT
(**SELECT DISTINCT** e3.ename
FROM emp e3, works w3, dept d3
WHERE e3.eid = w3.eid **AND** w3.did = d3.did **AND** d3.dname = 'Research');

(b) Return the names of every department without any employee.

For Relational Algebra:
$\pi_{dname}(dept) - \pi_{dname}(works \bowtie \text{dept})$

For nonrecursive-Datalog:
Q2(y) :- dept(x, y, _, _), ¬(works(_, x, _)).

For Relational Calculus:
{d.dname | d ∈ dept ∧ ¬∃w ∈ works (w.did = d.did)}

For SQL:
```
SELECT d.dname
FROM dept d
WHERE NOT EXISTS (
    SELECT *
    FROM works w
    WHERE w.did = d.did
);
```

(c) Print the managerids of managers who manage *<u>only</u>* departments with budgets greater than $1.5 million. ()

```
SELECT DISTINCT d.managerid
FROM dept d
WHERE NOT EXISTS (
  SELECT *
  FROM dept d2
  WHERE d2.managerid = d.managerid AND d2.budget <= 1500000
);
```

(d) Print the name employees whose salary is less than or equal to the salary of every employee.

```
SELECT e.ename
```

```sql
FROM emp e
WHERE NOT EXISTS (
  SELECT *
  FROM emp e2
  WHERE e2.salary < e.salary
);
```

(e) Print the enames of managers who manage manage departments with the largest budget.

```sql
SELECT e.ename
FROM dept d
JOIN emp e ON d.managerid = e.eid
WHERE NOT EXISTS (
    SELECT *
    FROM dept d2
    WHERE d.budget < d2.budget
);
```

(f) Print the name of every department and the average salary of the employees of that department. The department must have a budget more than or equal to $50.

```sql
SELECT d.dname, AVG(e.salary) as average_employee_salary
FROM dept d
JOIN works w on d.did = w.did
JOIN emp e on w.eid = e.eid
WHERE d.budget >=50
GROUP BY d.dname;
```

(g) Print the managerids of managers who control the largest amount of total budget. As an example, if a manager manages two departments, the amount of total budget for him/her will be the sum of the budgets of the two departments. We want to find managers that have max total budget.

```sql
SELECT d.managerid  from dept d
GROUP BY d.managerid
ORDER BY sum(d.budget) DESC
LIMIT 1;
```

(h) Print the name of every employee who works only in the ”Hardware” department.

```sql
SELECT  e.ename
FROM emp e
JOIN works w on e.eid = w.eid
JOIN dept d on d.did = w.did
WHERE d.dname = 'Hardware'
AND NOT EXISTS (
    SELECT *
    FROM works w2
    JOIN dept d2 on w2.did = d2.did
    WHERE e.eid = w2.eid
    AND d2.dname <> 'Hardware');
```

2: Prove that non-recursive Datalog without negation and relational algebra with selection, projection, and Cartesian product operators express the same set of queries. In this question, we consider only the non-recursive Datalog without negation queries with a single rule. We also consider only the relational algebra queries that produce non-empty answers over at least one database instance. Theorem 4.4.8 in Alice Book provides a summary of this proof. You should complete this summary and submit your proof.

To prove that any query expression in non-recursive Datalog without negation can be expressed in a relational algebra with selection, projection, and Cartesian product:

Suppose we have a Datalog query:

Q(x, z) :- R(x, y), S(y, z) *(1)*

This means that "find all x and z such that there exists a y for which (x, y) is in relation R and (y, z) is in relation S."

Step 1:

On the right hand side, we have a product of 2 relations R(x, y) and S(y, z). So we combine them using Cartesian product.

This is equivalent to "*R x S*" in relational algebra.

Step 2:

We apply selection for join condition to ensure that *y in R* matches *y in S* using selection operation:

$$\sigma_{R.y=S.y}(R \times S) \qquad (2)$$

Step 3:

Apply projection to (1) to get columns x and z:

$$\pi_{(x,z)}(\sigma_{R.y=S.y}(R \times S)) \qquad (3)$$

Now we have proved that non-recursive datalog without negation query (1) is equivalent to relational query (3).

To prove that any query expression in relational algebra can be expressed in a non-recursive Datalog without negation with selection, projection, and Cartesian product:

Suppose we have a relational algebra query:

$\pi_{(x,z)}(\sigma_{y=5}(R \times S))$ (4)

This mean that give x and z from all tuples resulted in the Cartesian product of R and S where y is equal to 5.

Step 1: Cartesian product *R x S*
In Datalog, the Cartesian product of R and S would look like:

Q(…) :- R(…), S(…) (5)

Step 2: Selection

Now we add a condition y = 5 to (4) to get all the rows that satisfy the condition:

$\mathrm{Q}(\dots) :- \mathrm{R}(\dots), \mathrm{S}(\dots), \mathrm{y} = 5$

Step 3: Projection

We use projection to choose the attributes we want (x and z in this case). So the query would look like:

Q(x, z) :- R(x, y), S(y, z), y = 5 (6)

We have proved that relational query (4) is equivalent to datlalog query (6).

Hence, non-recursive Datalog without negation and relational algebra with selection, projection, and Cartesian product operators express the same set of queries and vice versa.