Course: CS540 - DBMS, Winter 2022

Group 15: Assignment – 1

Names: Bharghav Srikhakollu, Akhil Sai Chintala

1. Relational Query Languages and SQL

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

Relational algebra:

$$\Pi\ ename\ \sigma_{eid=emp.eid}\ Emp[emp \bowtie works\ \Pi_{eid,\ did}$$

$$\sigma_{COUNT\ (eid)\ >=\ 3}$$

$$\gamma_{eid,\ COUNT\ (eid)}$$

$$\sigma_{dname\ =\ "Hardware"\ \ OR\ dname\ =\ "Software"\ OR\ dname\ =\ "Research"}\ dept]$$

Non-Recursive Datalog:

```
Q1(y)-Emp(a,b,c,d),Works(a,x,y),Dept(x,"Hardware",w,z)
Q2(y)-Emp(a,b,c,d),Works(a,x,y),Dept(x,"Software",w,z)
Q3(y)-Emp(a,b,c,d),Works(a,x,y),Dept(x,"Research",w,z)
Q(y)-Q1(y),Q2(y),Q3(y)
```

SQL:

```
select ename from emp where eid IN(select eid from works where did IN(select did
from dept where dname IN("Hardware","Software","Research")) group by eid
having count(eid)>=3)
```

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

Relational algebra:

$$\Pi dname(dept) - \Pi dname(dept \bowtie dept.did = works.did\ works)$$

Non-Recursive Datalog:

```
Q1(a,b,c,d)-Dept(a,b,c,d),Works(p,a,q)
Q(y):-Dept(a,b,c,d),notQ1(a,b,c,d)
```

SQL: select d.dname from dept d where d.did not IN(select w.did from works w where d.did = w.did)

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

select distinct d.managerid from dept d where 1500000 < all (select d2.budget from dept d2 where d2.managerid = d.managerid)

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

select ename from emp where salary<= all(select salary from emp)

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

select ename from emp where eid in(select d.managerid from dept d where d.budget = (select max(d2.budget) from dept d2))

(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.

select d.dname,avg(e.salary) from works w join dept d on w.did=d.did join emp e on w.eid=e.eid where d.budget>=50 group by d.dname desc

(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.

create view maxbudget AS

select managerid, sum(budget) as sumb from dept group by managerid;

select managerid from maxbudget where sumb>= all(select max(sumb) from maxbudget);

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

select ename from emp where eid in (select eid from works where eid

not in (select eid from works where did not in (select did from dept where dname = "hardware")))

2. Monotonicity and Satisfiability

Prove that rule based (Datalog) conjunctive queries with union and without negation are monotone and satisfiable.

- ✓ Monotone:

According to Definition 4.2.2 (Alice book), a query 'q' is monotone if

P ⊆ R then q(P) ⊆ q(R) where P and R are two databases, and every conjunctive query (CQ) is monotone.

In given question, for conjunctive queries with union and without negation, consider the union is between two queries which are monotone, then the conjunctive query with union and without negation can also be called as monotone.

For a rule based conjunctive query on the databases P and R, we get datasets and when we do the union, we get a subset of the dataset hence the relation q(P) ⊆ q(R) holds good. Hence, the queries can be referred to as monotone.

- ✓ Satisfiable:

According to Definition 4.2.2 (Alice book), a query 'q' is satisfiable if there is at least one database over which q has non-empty answer and the proposition is every conjunctive query is satisfiable.

For example, for each data rule we can build a dataset and we can do a union of them together. It becomes an instance of the database which has non empty answer.

Constant d is in query q, b ∈ domain. So, there is an instance of P over

relations R such that all tuples (rows) are formed from d U {b}. Hence, we can say that conjunctive queries with union and without negation are satisfiable.

3. Expressivity of Relational Algebra

A) Prove that the difference operator in relational algebra cannot be simulated using selection without negation, projection, join, and union operators.

- ✓ Set difference is the difference operator in relational algebra which gives output of first table tuples which are not present in second table.

Let's assume tables P, Q

P – Q => Gives output of P with tuples which are not in Q.

- ✓ For this difference relation, internally the tables are combined (join/union) and the difference of tuples is done(negation). Hence difference operator in relational algebra cannot be simulated using selection without negation, projection, join and union operators.

B) Prove that the difference operator in relational algebra cannot be simulated using selection with negation, projection, join, and union operators.

- ✓ As stated above, set difference needs the operators. For selection, along with the operators of negation, projection, join and union we need complement and intersection operations to perform the set difference.
- ✓ Hence the difference operator in relational algebra cannot be simulated using selection with negation, projection, join, and union operators.

4. Genericity of Relational Algebra

A total mapping τ from instances over relation R to instances over relation S is generic if for there is a finite C ⊆ dom and each bijection ρ over dom that is the identity on C, τ and ρ commute. That is, τ (ρ(I)) = ρ(τ (I)) for each instance I of R. Prove that each relational algebra query is generic.

- ✓ Data Independence principle: hides the internal view of data and this is the key for the query property of "Genericity" which says that the query depends only on the information of input instance.
- ✓ According to Definition 16.1.3 (Alice book), a query is C-generic if it commutes with permutations.

- ✓ As per the question and the genericity principle, we can say that any renaming of constants does not impact on query as the query focuses on only the relationship between the constants. So, changing of a query and applying it and vice versa is one and only same. Hence, we can say that the relational algebra query is generic.

Commuting diagram:

$$
\begin{array}{ccc}
I & \rightarrow \rho \rightarrow & \rho(I) \\
\downarrow & & \downarrow \\
\tau & & \tau \\
\downarrow & & \downarrow \\
\tau\,(I) & \rightarrow \rho \rightarrow & \tau(\rho(I)) = \rho(\tau(I))
\end{array}
$$