DBMS-EXERCISE- SHEET-2

Shubh Pareek(112001039)

February 2023

1 ASSIGNMENT-1

A)

1) $\Pi_{employee\_id,name}(\sigma_{department\_id=101 \ \vee \ department\_id=102}(\text{Departments}))$

2) $\Pi_{employee\_id,name}(\sigma_{department\_id=101}(\text{Departments})) \cup \Pi_{employee\_id,name}(\sigma_{department\_id=102}(\text{Departments}))$

B)

1) $\Pi_{name}(\sigma_{joining\_year<2019}(\text{Employees}))$

2) $\Pi_{name}(\text{Employees}) \setminus \Pi_{name}(\sigma_{joining\_year>2018}(\text{Employees}))$

C)

1) $\Pi_{Employees.name}(\sigma_{(Salaries.salary\_amount>50000 \ \wedge \ Departments.department\_name="Sales")}(\text{ Salaries } \bowtie_{(Salaries.employee\_id=Employees.employee\_id)} \text{ Employees } \bowtie_{(Departments.department\_id=Employees.department\_id)} \text{ Departments})) \setminus \Pi_{Employees.name}(\text{Managers} \bowtie_{(Managers.manager\_id=Employees.employee\_id)} \text{ Employees})$

2) $\Pi_{Employees.name}(\sigma_{(Salaries.salary\_amount>50000 \ \wedge Departments.department\_name="Sales")\wedge(Employees.employee\_id=Salaries.employee\_id)}(\text{Salaries} \times \text{Employees} \times \text{Departments})) \setminus \Pi_{Employees.name}(\text{Managers} \bowtie_{(Managers.manager\_id=Employees.employee\_id)} \text{ Employees})$

2 ASSIGNMENT-2

A)

$\Pi_{M.name,E.name}(\sigma_{M.employee\_id=Managers.manager\_id \wedge E.employee\_id=Managers.employee\_id}(\text{Managers} \times \rho_M(\text{Employees}) \times \rho_E(\text{Employees})))$

B)

• Expression 1 : $\Pi_{name}(\sigma_{department\_id=10}$ (Employees $\bowtie$ Departments))
• Expression 2 : Π_{name}(Employees $\bowtie$ ($\sigma_{department\_id=10}$ Departments))

expression 2 is more optimised since ,before joining the two table we have already constrained the Departments to values with department id = 10 , this will cause lesser tuples to be accessed .whereas in expression 1 both the tables are joined first , so this will even contain dept id's that are not 10 and will be filltered out by σ afterwards.

D)

minimum and maximum both will be 2000 since every salary will have exactly one employee for it ,as employee id is a foreign key.

3 ASSIGNMENT-3

A)
$\Pi_{department\_id,dept_avg}$(department_id$\gamma_{avg(B)}$ as dept_avg((employee_id$\gamma_{avg(salary_amount)}$ as B(Salaries))$\bowtie$ Employees))
B)
$\Pi_{manager\_id,count\_employee}$(manager_id$\gamma_{count(employee_id)}$ as count_employee(Managers))
C)
$\Pi_{manager\_id,A}$(manager_id$\gamma_{count(employee_id)}$ as count(Managers) $\bowtie_{count=A}$ $\Pi_A(gamma_{max(H)}$as A(manager_id$\gamma_{count(employee_id)} asH(Managers)$)))

4 ASSIGNMENT-4

A)

{ E.employee_id | (E $\in$ Employees) $\wedge$ $\forall$ M ((M $\in$ Managers) $\wedge$ $\exists$ m ((M $\in$ Managers) $\wedge$ M.manager_id = m.manager_id $\wedge$ m.employee_id = E.employee_id))}

B)

{T | $\neg$ (T $\in$ Employees) $\vee$ T.name = 'ABC'}

this relational expression means all the tuples such that they are not in employees table or name attribute is equal to 'ABC' , is **unsafe** because it can have infinite values because of first predicate.

{employee_id | $\exists$ manager_id , $\neg$ (manager_id,employee_id) $\in$ Manager $\wedge$ $\exists$ joining_year,employee_id (employee_id,joining_year,department_id) $\in$ Employees}

this relational expression is safe because it will be finite , because at max it can return all the employees.

C)

{E.name | (E $\in$ Employees) $\wedge$ $\exists$ D (D $\in$ Departments $\wedge$ D.department_id = E.department_id $\wedge$ D.department_name = "Sales") $\wedge$ $\exists$ S (S $\in$ Salaries $\wedge$ S.employee_id = E.employee_id $\wedge$ S.salary _ amount > 50,000) $\wedge$ $\neg$ ($\exists$ m (m $\in$ managers $\wedge$ m.manager_id = E.employee_id)) }