DBMS

Relational Algebra

RELATIONS

Employee (SSN, Name, dno, salary, gender) → department no.
Project (PNO, ~~project~~ name, dno)
Department (dno, dname, address)
Work_on (essn, pno)
↳ employee ID.

(— → primary key ↓ can't be same for 2 things (unique))

pno
essn & pno ~~are~~ is foreign key to ~~employee~~ works_on
pno is foreign key to project
dno is foreign key to employee & project

Types of Queries :-

① Give me tuples that satisfy a condition
eg. give me employee names where salary ≥ 50000.

σ → selection symbol (gives tuple)
π → projection symbol (gives particular columns of tuple selected)

$\pi_{name}(\sigma_{salary \geq 50000}(Employee))$

eg. give me employee names that work in the research department.

For this we need to use 2 relations - employee and department. So we need to join these 2 by using attribute dno.

⋈ → join symbol.

$\bowtie = \sigma(R_1 \times R_2)$

$R_1 : (A, B)$
$R_2 : (A, C)$ (so that the common column is not repeated)

$R_1 \bowtie R_2 = \pi_{A,B,C}(\sigma_{R_1.A = R_2.A}(R_1 \times R_2))$

↓ means when column of R_1 = column of R_2, join

Scanned by CamScanner

$$\pi_{name}\left(\sigma_{employee.dno = department.dno \wedge dname = \text{"Research"}}(\text{Employee} \times \text{Department})\right)$$

(Hard)

eg. Give me employee names that work on a project owned by the Research Department.

$$\pi_{name}\left(\sigma_{ssn = essn \wedge project.pno = workson.pno \wedge project.dno = department.dno \wedge dname = \text{"Research"}}(E \times D \times P \times W)\right)$$

We can optimize the above expression by choosing the research dept. inside

$$\pi_{name}\left(\sigma_{ssn = essn \wedge project.pno = workson.pno \wedge project.dno = department.dno}(E \times \sigma_{dname = \text{"Research"}}(D) \times P \times W)\right)$$

(2) eg. Give me those employee names that have worked on all the projects owned by the Research Department.

This is a DIVISION QUERY.

Explanation

$R_1(A, B)$		$R_2(B)$
a_1	1	1
a_1	2	2
a_2	1	3.
a_2	2	
a_2	3	
a_3	1	
a_2	4	

Scanned by CamScanner

$R_1 \div R_2$ (division query)

We are looking for a_i that have worked on all 1, 2 & 3.

So $R_1 \div R_2$ gives us a_2.

$S = \pi_A(R_1) \times R_2$

$\pi_A(S - R_1)$ = gives us employees that are missing at least one project that we need, i.e., disqualified employees.

③ eg. Give me those ~~guarded~~ employees names that have worked only on projects ~~worked~~ owned by the research department.

↳ This is ONLY QUERY.

(X)

Employees who only work on research dept. projects	Employees who have worked on ~~pro~~ other projects.

Using != , won't give you the answer as we have tuple-wise comparision.

↳ b/w 2 diff. relations.

$S = \pi_{pno}(\sigma_{dname != \text{"Research"}}(P \times D))$ → gives all projects not owned by research dept.

$A = \pi_{essn}(\text{Works_on} \times S)$

$\pi_{essn}(E) - A$ → Answer

Scanned by CamScanner

(4) eg Give me names of employees that have not worked on any projects by SSH department.

✓	X

employees who have not worked on any project by SSH dept.

employees who have worked on some project worked by SSH dept.

$\pi_{pno}\left(\sigma_{department.dno=project.dno \wedge dname="SSH"}(D \times P)\right) = S.$

can be replaced by $(S \bowtie works_on)$

automatically figures out the common attribute

$\pi_{essn}\left(\sigma_{pno=S.pno}(S \times works_on)\right) = A$

Ans:- $\pi_{ssn}(E) - A$.

Employee :-

SSN	Name	Sex	DNO.	Salary.
e1	Vikey	M	3	1000
e2	sonu	M	2	3000
e3	ram	M	3	1500
e4	sham	M	3	5000
e5	geeta	F	2	9000

gdno → grouping acc. to diff department numbers.

Query → give me max. salary per department

↓

dno, max(salary) gdno.

Scanned by CamScanner

→ dno, count(*) g_{dno} → count no of employees in each dept.

Result →

dno	salary
3	5000
2	9000

KK. Output can only have 2 attributes out of which one is the attribute acc. to which grouping is done.

Query → average salary per gender.

sex, avg(salary) g_{sex}.

Query → find the name of employees who have a higher salary than John Smith.

$$\pi_{G_1.name}\left(\sigma_{G_2.name = \text{"John Smith"} \wedge G_1.salary > G_2.salary}\left(\rho_{E=G_1} \times \rho_{E=G_2}\right)\right)$$

OR.

$$\pi_{salary}\left(\sigma_{name = \text{John Smith}}(E)\right) = A.$$

$$\cancel{\pi_{name}\left(\sigma_{salary > A}(E)\right)} \quad \Big| \quad \pi_{name}\left(E \bowtie_{salary > salary} A\right)$$

Query → find fname & lname of all employees working in Research dept. that earn more than ₹50000.

$$\pi_{fname, lname}\left(\sigma_{dno = dno \wedge salary > 50000 \wedge name = \text{"Research"}}\left(E \times D\right)\right)$$

Scanned by CamScanner

Query → find project name of for all projects worked on by John Smith

$\pi_{pname}\left(\sigma_{e.name='John' \wedge pno=pno \wedge ssn=essn}(E \times P \times Works_on)\right)$

Query → find the highest salary of the employees working in dept. no. 5.

~~$\pi_{max(salary)}(\sigma_{dno=5}(Employee))$~~

$max(salary)\left(\pi_{salary}\left(\sigma_{dno=5}(Employee)\right)\right)$

Query → find the employee(s) who earn the highest salary in the company.

$S = \pi_{salary}\left(\sigma_{max(salary)}(Employee)\right)$

↳ gives maximum salary of employees.

$\sigma_{employee.salary=S.salary}(E \times S)$

Query → find fname & lname of all employees who work on more projects than John.

$S = count(*)\left(\sigma_{name='Js' \wedge SSN=essn}(E \times Works_on)\right)$ → projects worked by John Smith

Ans $\left(count(*)\left(\sigma_{SSS=essn}(E \times Works_on)\right)\right)$

↳ should be greater than S.

SYNTAX :- SSN $\mathcal{G}$ count(*) (works_on)

grouping (partitioning works_on based on SSN) & then counting

output → SSN, count.

H (SSN, freq) SSN $\mathcal{G}$ count(*) (works_on)

for renaming

$\mathcal{G}$count(*) (works_on)

returns a single value, i.e., the no. of tuples in works_on

SQL Query

select – attribute
From – relation names
Where – conditional expressions

eg. Select name
From Employee
Where salary ≥ 5000 ; used in some software to indicate that the statement is over.

Conditional Expression :- (<, >, ≥, ≤)

Atomic value relational operator Atomic value

can be a constant

Scanned by CamScanner

operators :- in, any, all

(*) atomic value in (set of values) → tuple selected if this statement is true.

↓

returns true if atomic value lies in the set of values else returns false.

(*) ~~atomic value~~ salary > any → salary should be greater than any for this to be true

salary > any (set of values)

↓

tuple is selected if the above predicate is true

(*) salary > all (set of values)

↓

tuple is selected only if salary is greater than all values.

Q. Give me employee names whose salary is maximum

```
select  name
From    Employees
Where   salary > all [Select Salary
                      From Employee]
```

↓

sub-query / inner query.

(dependent)

(*) Uncorrelated & correlated sub-query.

above query has an uncorrelated sub-query, as its execution is independent of the actual query.

Scanned by CamScanner

Q. Give me employees who work in the Research Department.

```
Select ~~name~~ *  (all attributes)
From Employees
Where dno in ( Select dno
               From Department
               Where dname = 'Research'
             )
```

↓ Uncorrelated Sub-query.

Q. Give me employees name who get maximum salary in three over departments

```
Select name
From Employees
Where salary >= all ( Select salary
                      From Employee as E
                      Where E.dno = dno
                    )
```

↓ correlated sub-query

Set Operations

$\cup$, $\cap$, $-$, $\times$ → cartesian product.

↳ difference

and, or, in, NOT

x in (S_1) and x in (S_2) → Intersection

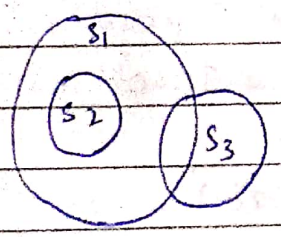
x in (S_1) and x not in (S_2) → Difference

Scanned by CamScanner

a. Give me employees names who have worked on projects offered by Research Department and have dependants.

```
Select name
From Employee, works_on
Where ssn in ( select essn
               From dependants
             )
             and ssn = works_on.ssn
             and works_on.pno in ( select pno
                                   From project, department
                                   Where pno = pnumber and
                                         dname = "Research"
                                 )
```

contains



S_1 contains S_2 (true)

S_1 contains S_3 (false)

not an operator

( ) vs ( )

set of values

Exists, not exists. → key words.

$S_2 - S_1 =$ empty ⇒ S_1 contains S_2

Not exists (true)

Scanned by CamScanner

Select ___
From ___
Where not exists (

)

tuple is selected if the set of the sub-query is empty.

Q. Give me project names which are worked on only by employees of the Research Department.

```
Select pname
From Project
Where not exists ( Select SSN
                   From Employee x.
                   Where ssn in (select essn
                                 From works_on
                                 Where pno = project.pno
                                )
                   and not in(

                   )
```

Scanned by CamScanner

TUTORIAL

Q1 Select T.branchname
From branch T, branch S.
Where T.assets > S.assets and
S.branch-city = 'Brooklyn'

Write efficient relational-algebra expression. Justify your choice.

Q2 Books (Bid, Title, Publisher, Year)
Students (Sid, Sname, Major, Age)
Authors (Aname, Address)
Borrows (Bid, Sid, Date)
Has-written (Bid, Aname)
Describes (Bid, Keyword)

(a) List all books published by McGraw Hill before 1990.
(b) List the names of authors who are living in Davis
(c) List the names of students older than 30 & studying CS.
(d) List the names of students older than 30 & not studying CS.
(e) List all the students with the books they can borrow.
(f) List the names of all students who have borrowed a book and who are CS majors.
(g) Rename Aname in the relation author to name.

Ans 1 Using 'X' is costly, so we should use natural join

(Renaming)

$$\pi_{T.branchname}\left(\pi_{branchname, assets}\left(\rho_T(branch)\right) \underset{T.assets > S.assets}{\bowtie} \left(\pi_{assets}\left(\sigma_{branch\text{-}city = \text{"Brooklyn"}}\left(\rho_S(branch)\right)\right)\right)\right)$$

Scanned by CamScanner

Ans 2.

(a) $\pi_{\text{bnum}}\left(\sigma_{\text{year}<1990 \wedge \text{publisher}=\text{"Hargrove Pub"}}(\text{Books})\right)$

(b) $\pi_{\text{aname}}\left(\sigma_{\text{address}=\text{"boise"}}(\text{Authors})\right)$

(c) $\pi_{\text{sname}}\left(\sigma_{\text{age}>30 \wedge \text{major}=\text{"CS"}}(\text{Students})\right) = A.$

(d) $\pi_{\text{sname}}\left(\sigma_{\text{age}>30 \wedge \text{major}!=\text{"CS"}}(\text{Students})\right)$ or $\pi_{\text{sname}}\left(\sigma_{\text{age}>30}(\text{Students})\right) - A.$

(e) $(\text{Students} \times \text{Books})$

↳ student can borrow any book they want.

(f) $\pi_{\text{sname}}\left(\sigma_{\text{students.sid}=\text{borrows.sid}}\left(\text{Borrows} \times \left(\sigma_{\text{major}=\text{"CS"}}(\text{Students})\right)\right)\right)$

(g) $\rho_{\text{name}(\text{Aname})}(\text{Authors})$

(h) Retrieve the names of students who have borrowed a book written by author 'Silberschatz'.

$\pi_{\text{sname}}\left(\sigma_{\text{students.sid}=\text{borrows.sid}}\left(\text{Students} \times \pi_{\text{borrows.sid}}\left(\sigma_{\text{borrows.sid}=\text{has_written.sid}}\left(\pi_{\text{bid,sid}}(\text{Borrows}) \times \pi_{\text{bid}}\left(\sigma_{\text{aname}=\text{"Silberschatz"}}(\text{has_written})\right)\right)\right)\right)\right)$

Scanned by CamScanner

Q3 Movie (movie id : integer, title : string, year : integer)
Actor (actor id : integer, name : string, gender : string)
Casting (movie id : integer, actor id : integer)
(movie cast to which actor)

(a) Find the names of all actors who have been cast for the movie "Iron Man"
(b) Find the names of all actors who don't work in any movie.
(c) List the titles of movies for which Marilyn Monroe worked.
(d) List the id of the movies done by some female actor.

Q4 Flights (Fno , Src, Dst , Dep , Arr)
Using RA, list all possible connecting flights from one city to another, having one stop and proper flight arrival and departure timing.

Q5 Each customer is given a unique ID number, cid. All current account holders have different (Firstname, lastname) pairs. As per RBI, companies need to collect other card no. also.
can be same individually.
Companies database contains following field :-
(cid, firstname, lastname, age, adhar)
Which of the following do not contain wrong information :-?

	Candidate key	Super keys	Primary keys.
1)	(cid, adhar)	(cid), (adhar, age), (cid, adhar)	(cid), (adhar)
2)	(adhar)	(cid), (firstname, lastname)	(cid)
3)	(cid), (adhar)	(cid, firstname), (adhar, firstname)	(adhar)
4)	(firstname, lastname), (adhar)	(cid, adhar), (adhar, firstname, lastname)	(cid)

Scanned by CamScanner

A Row 1 & 2

B Row 2 & 3

C ✓ Only Row 3

D Only Row 4.

Ans 3.

(a) $\pi_{name}\left(\sigma_{\substack{name = \\ "HanHan"}}\left(\text{movie} \underset{\text{equi join}}{\bowtie} \text{Casting} \bowtie \text{Actor}\right)\right)$

(b) All actors - actors who work on same movie

$\pi_{name}(\text{Actors}) - \pi_{name}(\text{Actors} \times \text{Casting})$

(c) $\pi_{title}\left(\sigma_{\substack{name = \\ "Marilyn \\ Monroe}}\left(\text{Movie} \bowtie \text{Casting} \bowtie \text{Actor}\right)\right)$

(d) $\pi_{\substack{movie \\ id}}\left(\sigma_{\substack{gender = \\ "Female"}}\left(\text{Movie} \bowtie \text{Casting} \bowtie \text{Actor}\right)\right)$

Ans 4.

$\sigma_{\substack{F1.dst = F2.src \\ \wedge\ F1.arrival < F2.dep}}\left(\rho_{F_1}(\text{Flegute}) \times \rho_{F_2}(\text{Flegutes})\right)$

Ans 5 $\boxed{\text{Candidate key}}$

Super keys → all possible keys that can represent a tuple.
↳ includes primary key.

Scanned by CamScanner