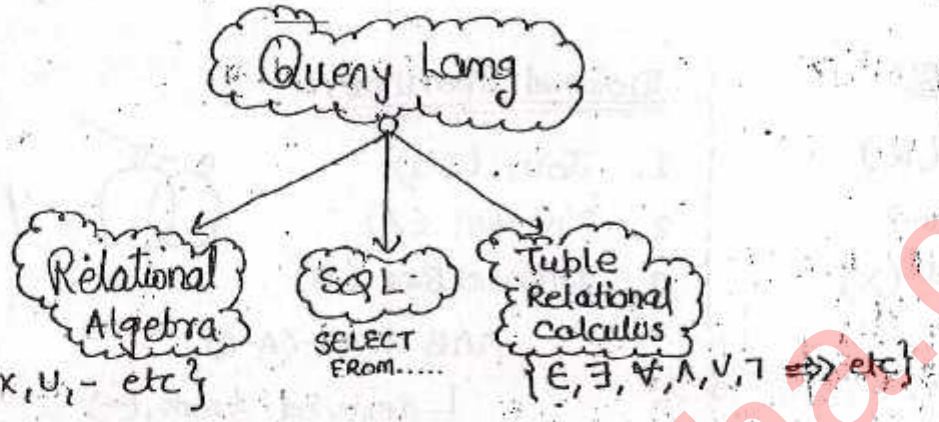
2nd Sep. 2011

Relational Algebra



Formal Query Language :- (Procedural QL)

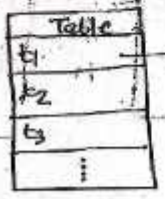
What to retrieve from database and How to retrieve from database } Relational Calculus

Informal Query Language :- (Non-Procedural)

What to retrieve from the db. It is least bothered about How to retrieve it. } Tuple Relational calculus & SQL..

Note

Query is always executed tuple by tuple and one tuple at a time.



t_1 (check condition) ⇒ All
↓
tuple by tuple.

Relational Algebra :-

» RA by default eliminate the duplicate tuples from the result.

Basic operations:

1. Projection (π)
2. selection (σ)
3. Cross Product ($\times$)
4. Union ($\cup$)
5. minus ($-$)
6. Rename (ρ)

Derived operations

1. Join ($\bowtie$)
2. Division ($/$)
3. Intersection ($\cap$)

$A \cap B = A - (A - B)$

└ derived from ($-$)

Projection (π):

⇒ Retrieve name of Red color parts.

Parts

Pid	Pname	Color
P_1	A	R
P_2	A	G
P_3	B	R
P_4	C	G
P_5	A	R

(i) Select

$$\sigma_{Col=Red}(Parts)$$

(ii) Project

$$\pi_{name}\left(\sigma_{Col=Red}(Parts)\right)$$

A → P_1
B → P_3
A → P_5

Result is :

Pname
A
B

→ Relation algebra by default remove duplicates Hence A not duplicated.

Set Operators :- $\{\cup, \cap, -\}$

To apply union, intersection, set difference,

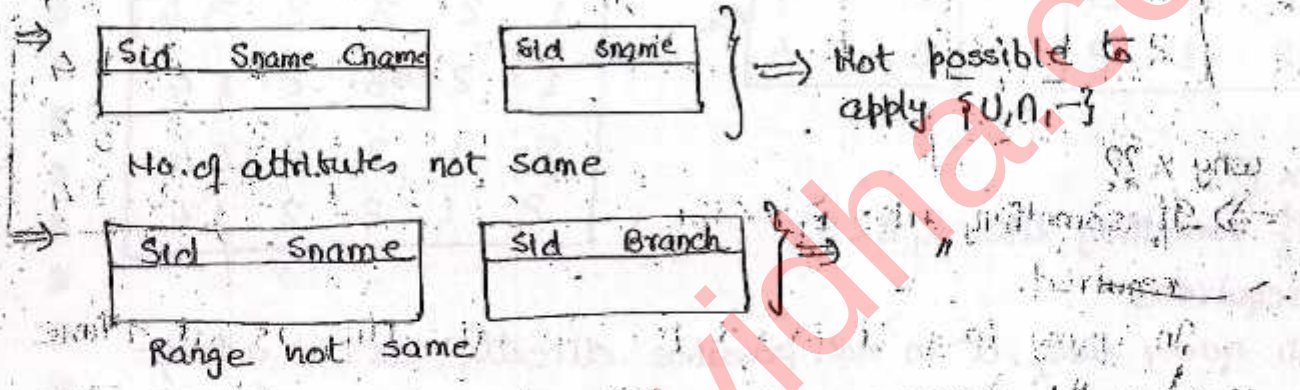
Relation should be Union compatible:-

R and S union compatible only if

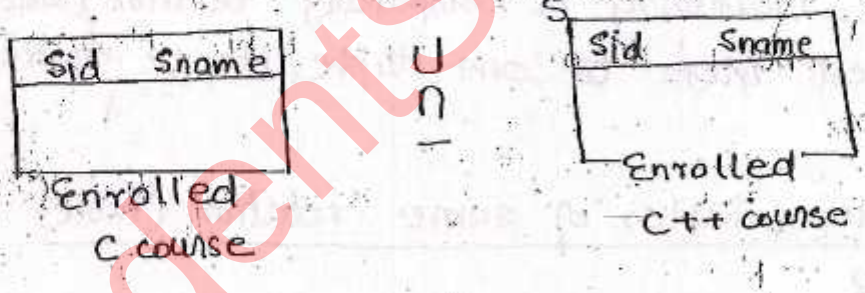
① No. of attributes is same

and

② Range of attributes should be similar



NOTE:- Set operation can be applied only on similar tables (containing diff. data)



Assume R and S consists m and n tuples each

①	Range of tuples in R∪S	{max(m,n) to m+n}
②	" "	R∩S {∅ to min(m,n)}
③	" "	R−S {∅ to m}

CROSS PRODUCT (X) :

RXS = results all attributes of R and all attributes of S.

R

A	B	C
1	2	3
2	1	2

S

B	C	D
2	3	4
2	1	4

RXS

A	B	C	B	C	D
1	2	3	2	3	4
1	2	3	2	1	4
2	1	2	2	3	4
2	1	2	2	1	4

Why x ??

⇒ If something like $R_1.B < S.B$ required,

In query lang, it is not possible directly, as at a time it only fetches a row either from R or from S. Thus not possible to compare.

Thus we are maintaing a temporary relation (table) RXS, to convert them to one tuple. Hence enable comparison.

⇒ To compare two tuples of same relation (Table)

Table = R

We need to perform RXR

⇒ Three tables then RXRXR

CONDITIONAL JOIN :- $\bowtie_c$ → Project column depending on any condition

$$R \bowtie_c S = \prod_{\text{Column}} (\sigma_c (R \times S))$$

Example:- $R \underset{R.B<S.C}{\bowtie} S = \Pi_{ABCBCD}\left(\sigma_{R.B<S.C}(R \times S)\right)$

output:

A	B	C	B	C	D
1	2	3	2	3	4
2	1	2	2	3	4

Natural Join: Common attribute should have equal values.

$$R \bowtie S = \Pi_{ABCD}\left(\sigma_{\substack{R.B=S.B \wedge \\ R.C=S.C}}(R \times S)\right)$$

output

A	B	C	D
1	2	3	4

Left Outer Join: $\bowtie$

$R \bowtie S$ = $R \bowtie S$ and tuples from left side Relation (R) those which failed Join condition.

A	B	C	D	
1	2	3	4	→ Tuple satisfying natural join.
2	1	2	NULL	→ tuple failed $R \bowtie S$

⇒ Not possible to derive from basic operators of relational algebra.

Division — can be derived
Nat. join — "

Right Outer Join: $\bowtie$

$R \bowtie S$ = $R \bowtie S$ and tuples from side side of relation R which failed join condition.

A	B	C	D
1	2	3	4
NULL	2	1	4

Full Outer Join :-

$$R \overline{\bowtie} S = (R \bowtie S) \cup (R \bowtie S)$$

A	B	C	D
1	2	3	4
2	1	2	NULL
NULL	2	1	4

Q: Outer join → can't be derived from basic operators of relational algebra.

mp

Q: R(ABC) S(BDE)

F.D = { B → D, D → E }

and R has 100 tuples

S has 200 tuples

How many maximum number of tuples in R ⋈ S ?

$$R \bowtie S = \Pi_{ABCDE} \left(\sigma_{R.B = S.B} (R \times S) \right)$$

$B^+ = BDE$

{B} : Candidate Key → means all 200 tuples of S are distinct.

max = ?

Let B in R is same

R × S = 100 × 200 tuples

matching condition.

No. of tuples = 100.

∴ max = 100

Minimum = 0 → ∵ values in B of R can be > 200 - hence nothing

DIVISION :- (/ or ÷)

Enrolled

Sid	Cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C3
S3	C1

Course

Course ID
C1
C2
C3

⇒ Retrieve all students who are enrolled some course.
or any course
or atleast one course

Π_{Sid} (Enrolled)

Sid	Result
S1	
S2	
S3	

⇒ Retrieve all students who are enrolled every course.

Result

Sid
S1

$\Pi_{Sid,Cid}$ (Enrolled) / Π_{Cid} (course)

2nd attribute should be same

⇒ It results sid's of Enrolled table for that there should be cid that are specified every cid value in the course table.

⇒ It is a **derived operator**

Π_{Sid} (Enrolled) × Π_{Cid} (course) − Enrolled

Sid
S1
S2
S3

×

Cid
C1
C2
C3

=

Sid	Cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C2
S2	C3
S3	C1
S3	C2
S3	C3

Universal Result ⇒ Every student enrolled every course

−

Sid	Cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C3
S3	C1

=

Sid	Cid
S2	C2
S3	C2
S3	C3

↗ student not enrolling the course.

Sid
S2
S3

Not enrolled for all courses

disqualified.

Student not enrolled all course = { Every student enrolled every course } − { students Enrolled some couse }

its result is compliment of

not enrolled all course ⟷ enrolled some course

Student enrolled all course = { student who are enrolled some course } − { student not enrolled all course }

S1 = S1 S2 S3 − S2 S3

$\Pi_{sid,cid}$ (Enrolled) / Π_{cid} (course) — Division operator is

= (is equivalent to)

Π_{sid} (Enrolled) − Π_{sid} (Π_{sid} (Enrolled) X course − Enrolled)

Q: Retrieve Cid which is enrolled by every student.

→ Student

sid
S1
S2
S3

Enrolled

sid	cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C3
S3	C1

Course

cid
C1
C2
C3

⇒ Course with enrolled by every student.

$$\Pi_{cid,\,sid}(enrolled) \;/\; \Pi_{sid}(student)$$

→ If student not given then Project (Π) sid from Enrolled table.

Consider a database with relations

Suppliers (sid, sname, ~~catalog~~ rating)

S1
S2
S3
S4

Parts (Pid, Pname, Color)

P1
P2
P3

Catalog (sid, Pid, Cost)

sid	Pid
S1	P1
S1	P2
S2	P1

(1) Retrieve the Sid of the suppliers whose rating greater than 10.

$$\Pi_{sid}\left(\sigma_{rating>10}(supplier)\right)$$

— only one table required

(2) Retrieve Sid of suppliers who supplied some parts and whose rating greater than 10.

$$\Pi_{sid}\left(\sigma_{supplier.id = catalog.id \,\wedge\, rating>10}(Catalog \times supplier)\right)$$

Two tables supplier & Catalog required.

also

$$\pi_{sid}\left(\sigma_{rating>10}(Supplier)\right) \cap \pi_{sid}(Catalog) \quad \#$$

also

$$\pi_{Sid}\left(\sigma_{rating>10}(Supplier \bowtie Catalog)\right)$$

Q Retrieve sid's of the supplier who supplied some red color part.

→ Two tables catalog & Parts required.

Catalog

Sid	Pid	Cost
S1	P1	
S2	P2	
S1	P4	

Parts

Pid	Pname	Pcolor
P1		R ✓
P2		G ✗
P3		R ✗
P4		G ✗

100 tuples

C.pid = P.pid ✓ , color ≠ Red] discarded

C.pid ≠ P.pid → discarded

$$\Rightarrow \pi_{sid}\left(\sigma_{C.pid=P.pid \wedge Color=RED}(Catalog \times Parts)\right) \quad \# = (50 \times 100) \text{ tuples after } \times \text{ product}$$

but we need only Red color parts.

Join two tables

$$\Leftrightarrow \pi_{sid}\left(\sigma_{Color=Red}(Catalog \bowtie Parts)\right)$$

more efficient query.

Just 500 tuples required now

500 tuples

$$\pi_{sid}\left(\sigma_{C.pid=P.pid}(Catalog \times \sigma_{Color=Red}(Parts))\right)$$

50 × 10 (let 10 Parts = Red)

$$\equiv \pi_{sid}\left(Catalog \bowtie \sigma_{Color=Red}(Parts)\right)$$

2nd 3rd 2nd

Let an attribute A belongs to (R only), then.

$$\sigma_{A='a'}(R \bowtie S) = \sigma_{A='a'}(R) \bowtie S$$

Let attribute (A) belong to (only R) and (B) belong to (S only).

$$\sigma_{A='a' \wedge B='b'}(R \bowtie S) = \sigma_{A='a'}(R) \bowtie \sigma_{B='b'}(S)$$

(more efficient query)

(3) Retrieve Sid of the suppliers who supplied some RED part or some green part.

$$\Pi_{Sid}\left(Catalog \bowtie \sigma_{Color=Red \vee Color=Green}(Parts)\right)$$

(i) Projected by Sid & Pid

(ii) $\Pi_{Sid,Pid}(Catalog)$

Removing unnecessary columns

or

$$\Pi_{Pid}\left(\sigma_{Color=Red \vee Color=green}(Parts)\right)$$

: Rename & Cartesian product not required.

ρ ⇐ Rename i.e supplier, supplying

red = T_1

green = T_2

$$\rho\left(T_1, \Pi_{Sid}(Catalog \bowtie \sigma_{Color=Red}(Parts))\right)$$

$$\rho\left(T_2, \Pi_{Sid}(Catalog \bowtie \sigma_{Color=Green}(Parts))\right)$$

$$T_1 \cup T_2$$

ρ — Rename operators

Retrieve sid of the supplier who supplied red parts and some green part

(a) $Q_1 = \Pi_{sid}(\text{Catalog} \bowtie \sigma_{(Color=Red \wedge Color=Green)}(\text{Parts}))$ → 0 tuple

always empty ↓ incorrect

$Q_2 = \rho(T_1, \Pi_{sid}(\text{Catalog} \bowtie \sigma_{Color=Red}(\text{Parts})))$

$\rho(T_2, \Pi_{sid}(\text{Catalog} \bowtie \sigma_{Color=Green}(\text{Parts})))$

$T_1 \cap T_2$ → intersection blw sid not color

(a) Q_1 correct Q_2 incorrect

(b) Q_2 " Q_1 "

(c) both correct

(d) Both incorrect

Retrieve sid's of the suppliers who supplied Every part → division

$\Pi(\text{Catalog})_{Sid\ Pid} \;/\; \Pi_{Pid}(\text{Parts})$

m n

Expansion of ÷ operator:-

$$\Pi_{sid}(Catalog) \times \Pi_{sid}(Parts)$$

$$\left[\Pi_{sid}(Catalog) \times \Pi_{sid}(Parts) - \Pi_{sid,pid}(catalog)\right]$$

$$\Pi_{sid}\left(\Pi_{sid}(catalog) \times \Pi_{sid}(Parts) - \Pi_{sid,pid}(catalog)\right) = A(Let)$$

$$\Rightarrow \boxed{\Pi_{sid}(Catalog) - A}$$ → disqualified data

(5) Retrieve Pid's of Parts that is supplied by (Every) supplier.

→ division

$$\boxed{\Pi_{Pid,Sid}(Catalog) \,/\, \Pi_{sid}(Supplier)}$$

Expansion

$$\Rightarrow \boxed{\Pi_{Pid,Sid}(Catalog) \times \Pi_{sid}(Supplier)} \rightarrow A$$

Every part supplied by every supplier.

$$\boxed{A - \Pi_{Pid,Sid}(Catalog)} \rightarrow B$$

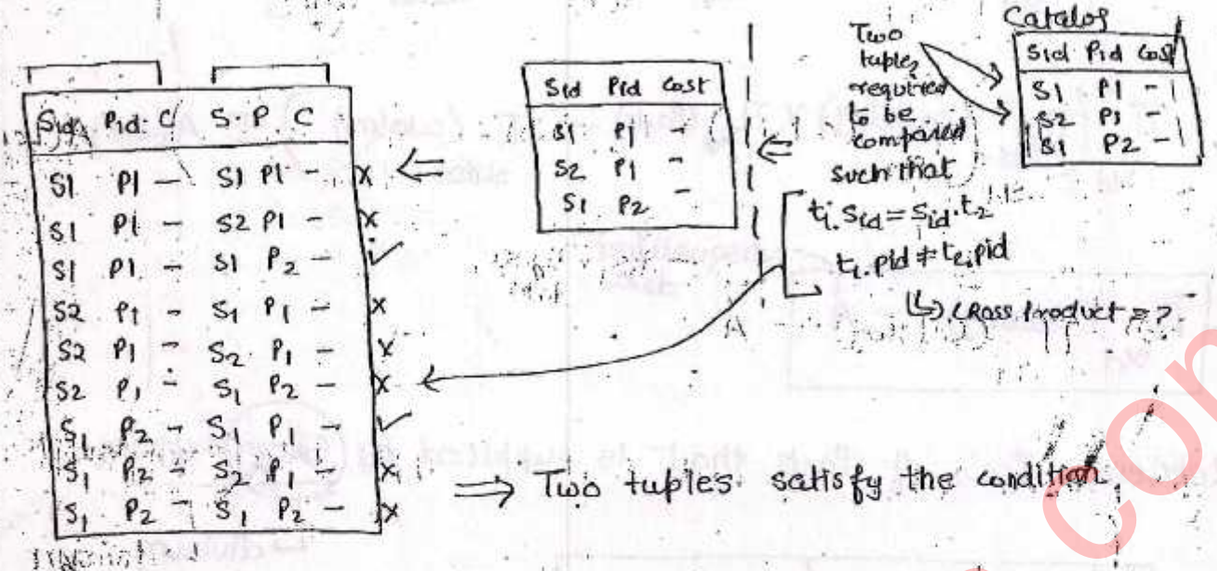
$$\boxed{\Pi_{Pid}(B)} \Rightarrow C.$$

$$\boxed{\Pi_{Pid}(Catalog) - C}$$ Ans

a) Retrieve Sid of the suppliers who supplied atleast one part.

$$\pi_{Sid}(Catalog)$$

b) Retrieve sids of the supplier who supplied atleast two parts.



⇒ Two tuples satisfy the condition.

⇒

Forms of Renaming operator:

$\rho(T_1, Catalog)$ = T_1 (Sid Pid Cost) — Renaming Table

$\rho_{S,P,C}(Catalog)$ = Catalog (S P C) — Renaming Attributes

$\rho_{1\to S,\ 2\to P}(Catalog)$ = Catalog (S P Cost) — Renaming specific attributes

Sol^n $\rho(T_1, \text{catalog})$, $\rho(T_2, \text{catalog})$

Renaming as we want RXR

$R_1.sid = R_1.sid$ create ambiguity hence Renaming used.

$$\pi_{T_1.sid}\left(\sigma_{T_1.sid = T_2.sid \wedge T_1.pid \neq T_2.pid}(T_1 \times T_2)\right)$$

$$\equiv \pi_{T_1.sid}\left(T_1 \bowtie_{T_1.sid = T_2.sid \wedge T_1.pid = T_2.pid} T_2\right)$$

Here $\bowtie$ is $_{A.B=B.A}$ conditional join not the natural join

OR $\pi_{sid}\left(\sigma_{\substack{Sid=S \\ Pid \neq P}}\left(\text{Catalog} \times \rho_{S,P,C}(\text{Catalog})\right)\right)$

$\bowtie_{cond}$ = Conditional Join

$$\equiv \pi_{sid}\left(\text{Catalog} \bowtie_{\substack{sid=S \\ pid \neq p}} \rho_{spc}(\text{Catalog})\right)$$

(8) Retrieve the sid of the supplier who supplied atleast three parts. (RXRXR)

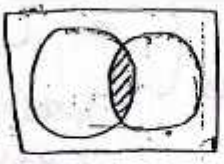
$\rightarrow$ $\rho(T_1, \text{catalog})$ $\rho(T_2, \text{catalog})$ $\rho(T_3, \text{Catalog})$

$$\pi_{T_1.sid}\left(\sigma_{\substack{T_1.Sid = T_2.sid = T_3.sid \wedge \\ T_1.pid \neq T_2.pid \neq T_3.pid}}(T_1 \times T_2 \times T_3)\right)$$

$$\left\{\begin{array}{l} T_1.sid = T_2.sid \wedge \\ T_2.sid = T_3.sid \wedge \\ T_1.pid \neq T_2.pid \wedge \\ T_2.pid \neq T_3.pid \wedge \\ T_1.pid \neq T_3.pid \end{array}\right.$$

(c) Retrieve sid of the supplier who supplied exactly two parts

Sid of the Supplier who supplied at-least two parts — Sid of the supplier who supplied atleast three parts



Eg: $\{S1, S2, S3\} - \{S3\} \Rightarrow \{S1, S2\}$

(d) Retrieve sid of the supplier who supplied atmost two parts.

Total − atleast three

Π_{Sid} (suppliers) − Sid of the supplier who supplied atleast 3 parts

(e) Retrieve pairs of sid's such that S_{id} with the first supplier supplies more cost than S_{id} with the second supplier for some cost.

⇒

Sid	Pid	Cost
S1	P1	10
S2	P1	20
S3	P1	30
S3	P2	30
S1	P2	10

Expected O/P

Sid1	Sid2
S2	S1
S3	S2
S3	S1

$T_1 \times T_2$

T_1			T_2		
Sid	Pid	Cost	Sid	Pid	Cost
S1	P1	10	S2	P1	20
S1	P1	10	S3	P1	30
S2	P1	20	S1	P1	10
S2	P1	20	S2	P1	20
S3	P1	20	S3	P1	30
S3	P1	30	S1	P1	10
S3	P1	30	S2	P1	20
S3	P1	30	S3	P1	30

$T_1.Cost > T_2.cost \wedge$
$T_1.sid \neq T_2.sid \wedge$
$T_1.pid = T_2.pid$