<> → different (not equal to).

Now, Complete Query :-

```
Select T1.Sid from Study as T1, Study as T2
Where
        T1.Sid = T2.Sid        (Student name same)
            and
        T1.Cid <> T2.Cid.;     (Course diff)
```

So, final output Table :-

S_1	C_1	S_1	C_2
S_1	C_2	S_1	C_1

→ Now, we want S-id from this Table.
But T1 & T2 both have S-id,
So,
we can use any $\Pi_{T_1.Sid}$ or $\Pi_{T_2.Sid}$

Ans - S_1

(40) EQUI - Join : → (=).
(कोई भी attr के बीच (=) लगा सकते हैं)

E_No	E_name	Add.
1	Ram	Delhi
2	Mohan	Chd.
3	Ravi	Chd.
4	Amrit	Delhi

Employee.

Dep_No.	Loco	E_no.
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

Department / Dept.

T ∧ T	= T
T ∧ F	= F
F ∧ T	= F
F ∧ F	= F

T or T	= T
T or F	= T
F or T	= T
F or F	= F

Q:- find the Emp name who worked in a department having loca" same as their address ?

Ans:- By just seeing the 2 Table :→

Ram .

(A) Query :→

```
Select Ename from Emp, Dept where
        Emp.E-no = Dept.E-no    ;; common
        and
        Emp.Add = Dept.Loca" ;
```

Emp.

Delhi	1	Ram
"	1	"
"	1	"
Chd	2	Varun
"	2	"
"	2	"
"	3	Rani
"	3	"
"	3	"
Delhi	4	Amit
"	4	"
"	4	"

Dept.

D_1	Delhi	1
D_2	pune	2
D_3	patna	4
D_1	Delhi	1
D_2	pune	2
D_3	patna	4
D_1	Delhi	1
D_2	Pune	2
D_3	patna	4
D_1	Delhi	1
D_2	pune	2
D_3	patna	4

→ Ram .

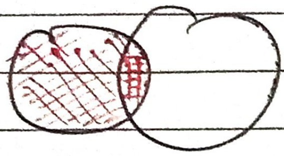
Ans.

Note:- दो Table के common att. के बीच में हमेशा (=) लगाना है & किसी भी 2 Att के बीच हम (=) लगा सकते हैं।

(41) Left Outer Join :→

→ It gives the matching rows & the rows which are in left table but not in the right table.

(~~Natural Join~~ + Something else)



A B.

Emp		
Emp.no	E.name	Dept.no.
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_1
E_4	Nitin	—

Dept.		
Dept.no.	Dname	Loc.
D_1	IT	Delhi
D_2	HR	Hyd.
D_3	Finance	Pune

Query :→

```
Select emp.no, e.name, d.name, loc from
emp left outer Join dept on
(emp.deptno = dept.deptno.)
```

left

→ cond'n of Natural Join.
(Common att. को देखकर करेगा।)

Relational database always shows output in Table form.

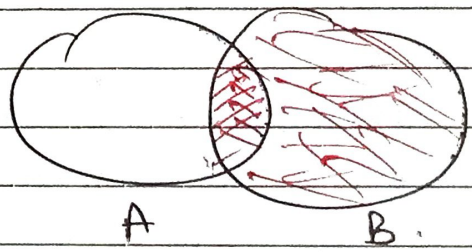
output Table :-

Emp no.	E name	d-name	Loc^n
E_1	Varun	I T	Delhi
E_2	Amrit	H R	Hyd.
E_3	Ravi	I T	Delhi
E_4	Nitin	—	—

→ (Left की तो सभी Row आनी ही है।)

12. Right Outer Join :→

It gives the matching rows & the rows which are in Right Table but not in left Table.



(Emp)

Emp-no	E name	Dept-no
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_3

(Dept)

Dept. No	D-name	Loc^n
D_1	IT	Delhi
D_2	HR	Hyd
D_3	Finance	Pune
D_4	Testing	Noida

Query :-

Select emp-no, e-name, d-name, loc from emp Right Outer join dept on (emp.dept-no = dept.dept-no.)

(natural join.)

⇒ Output Table :→

Emp-no	E-name	D-name	Loc^n
E_1	Varun	IT	Delhi
E_2	Amrit	HR	Hyd.
E_3	Rani	Finance	Pune.
–	–	Testing	Noida.

(Right Table all rows.)

'Full-Outer Join' :- Take the union of left outer join & Right outer join rows.

(Left O.J.) ∪ (Right O.J.)

(93.) Relational Algebra :→ (1970).
(procedural lang. or formal Query lang.) also.

→ have to do these things in Query :→
1.) What to do.
2.) How to do.

⇒ SQL language base is Rela'al Algebra.
(collection of mathematical Expres').

Rela'al algebra → SQL → No SQL
In Today's time.

"operators"

Basic operator	Derived operator
→ projection (π)	→ Join (⋈)
→ Selection (σ)	→ intersect (∩)
→ Cross product (×)	(x ∩ y) = x - (x - y)
→ Union (∪)	
→ Rename (ρ)	→ Division (/, ÷)
→ Set Difference (-)	

⇒ If we understand Rela"al Algebra very clearly, then we don't face any problem in understanding SQL.

---

(44) Projection in Rela"al algebra :→

→ Projection (π) : →

π func. is to Retrieve the data.

Roll no.	Name	Age
1	A	20
2	B	21
3	A	19
⋮	⋮	⋮

← (Student)

→ Query: Retrieve the roll no. from table (Student)

Date:
Page:

2). $\pi_{\text{Roll no.}}(\text{Student})$.

Roll no.
1
2
3

→ Query:- $\pi_{\text{Roll no, name}}(\text{Student})$.

We fetch the 2 col^m (roll no. & name) from the student (Table).

Output →

Roll no	Name
1	A
2	B
3	A

Note: It only gives unique Answers.

Q.2) # Query:- $\pi_{\text{name}}(\text{Student})$.

Name
A
B

→ (only name here).

→ project, we use it in last & before this we use other Operators.

---

45) Selection in Rela^{n}al algebra (R.A.).

σ (Name).

Roll no.	Name	Age
1	A	20
2	B	21
3	A	19.

⇒ It works on Tuples (rows).

→ First, it found rows.

π operator.

* Query: Retrieve the name of student whose Roll no = '2'

⇒ $\pi_{rollno='2'}(Student)$ → [2, B, 21]

↓ final.

$\pi_{name}(\sigma_{rollno='2'}(Student))$ → B

π always last में operate होता है।

* We can also find 2 at a time.

Ex:-

$\pi_{name, Age}(\sigma_{rollno='2'}(Student))$ → B, 21

Note:- Project (π) always works on Columns.
&
Selection (σ) always works on Rows. (Tuples)

---

46. Cross / Cartesian product in Relational algebra

⇒

A	B	C
1	2	3
2	1	4

R1

C	D	E
3	4	5
2	1	2

R2

To join, we must have to cross product.

→ 3 + 3 = 6 (m+n).

2 × 2 = 4 (x×y).

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

→ $(R_1 \times R_2)$.

(R1) (R2)

→ We need a common att. to join 2 tables. (Here, C).

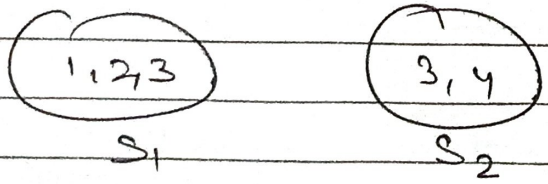
---

(47) Set Difference in R.A :→

(A - B) = A but not B.

= $A \cap B'$

Ex:



$S_1 - S_2 = 1, 2$. $S_2 - S_1 = 4$

→ It is not Commutative :→

i.e.

$A - B \neq B - A$

Condition:-

1.) No. of columns must be same in no.
2.) Domain of every column must be same. (Data of same type). (Ex- Numeric).

Ex:-

Rollno	Name
1	A
2	B
3	C

(Student)

Emp-no	Name
7	E
1	A

(Employee)

⇒ (Student) − (Employee)

Rollno	Name
2	B
3	C

⇒ By default, पहले table के column लेता है।

Q:- Find the name of a person who is a student but not employee.

⇒ (Student − Employee).

→ How to Write?

$(\Pi_{name}(student) - \Pi_{name}(Employee))$.

→ output:-

Name
B
C

$((A(B,C)) - (E,A))$

(48) Union Opera" in R.A. :→

⇒ Same as in Set. (∪).

(1,2,3) S_1 (3,4,5) S_2.

↓

(1,2,3,4,5)

Ex:→

Roll no.	Name
1	A
2	B
3	C

(Student)

Emp. No.	Name
7	E
1	A

(Employee)

* Condⁿ :→

1. No. of col" must be same in R.
2. Domain of every col" must be same.

(student) ∪ (Employee)

Roll no.	Name.
1	A
2	B
3	C
7	E.

→ (Table. left table" ata hai).

Easy Trick! -

2). (π_{name} (Student) ∪ π_{name} (Employee)).

Name.
A
B
C
E .

→ (Student also & who is Employee also or both.).

Note! →

Roll no	name
1	A
2	B
3	C

Numeric

Name	Emp no.
E	2
A	1 .

alphabet

not same domain.
(ie, By Default order pick करता !).

So, here, not Union operaⁿ applies.
(NULL).

49. Division Operaⁿ in R. A. :→

⇒ Divisⁿ operator is actually a Derived operator.
(ie derived them, from normal operators).

/, ÷ (×, −, π, σ).

↳ ie use these

⇒ **Query**: Retrieve Sid of Students who enrolled in every (all) Course.

Date: ....
Page: ....

⇒ every (या) all देखते ही Divisⁿ method.

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_3

'Enrolled'

Cid
C_1
C_2

'Course'

Note of Divisⁿ Method :→

A(X, Y) / B(y) = it results 'x' values

for that there should be tuple <x, y> for every y value of Relⁿ B.

Here,

$$E(Sid, Cid) / C(Cid) = S_1$$

i.e.,

S_1 enrolled in every course (C_1 & C_2).

How it happens ?

We let, all students are enrolled in every course.

(for this, we do the cross product).

⇒ $(\pi_{sid}(Enrolled)) \times (\pi_{cid}(Course))$.

S_1	×	C_1
S_2		C_2
S_3		

S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_2
S_3	C_1
S_3	C_2

−

S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

(Enrolled)

Now, (we subtract actual Scenario i.e, (Actual Table)) from this cross product. → we get S_2, S_3 (who don't enrolled in all courses)

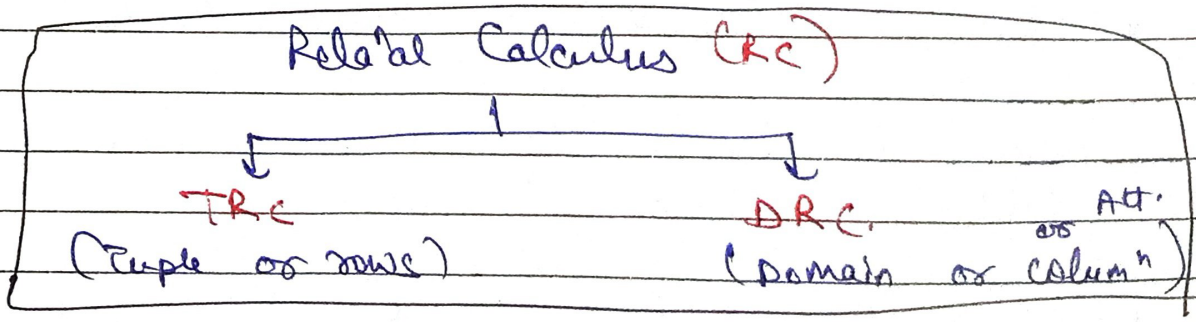
(**) Now, final ! → (Query).

$$\Pi_{sid}(Enrolled) - (\Pi_{sid}(((\Pi_{sid}(Enrolled)) * \Pi_{cid}(course)) - (Enrolled))$$

S_1 — S_2
~~S_2~~ S_3
~~S_3~~

→ S_1 Ans.

So- Tuple Calculus in DBMS : →



→ Tuple Relational Calculus is a Non-procedural Query lang. (only tells what to do & not that how to do) unlike Relational algebra.

Date :
Page :

⇒ In Tuple Calculus, a query is expressed as

$\{t \mid P(t)\}$

where, t = resulting tuples,
P(t) = known as predicate & these are the conditions that are used to fetch t, Thus, it generates set of all tuples t, such that Predicate P(t) is true for t.

* Operations :→

→ P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT (¬).

→ Atomic func's :- (We use one variable here. Ex - (Supply Ids. from a Supply table) → only 1 condⁿ.

→ It also uses quantifiers :

- $\exists t \in r\ (Q(t))$ = "there exists" a tuple in t in relⁿ r such that predicate Q(t) is true.
- $\forall t \in r\ (Q(t))$ = Q(t) is true "for all" tuples in relⁿ r.

* Unsafe Expression :- → (U.E.)

Supplier (Table).

- $\{ S.name \mid \neg Supplier(s) \}$

(not sign)

Here, It means that, all the supplier name who are not in the supplier Table.

So, here, ans is ∞. (So, ∞ loop ~~that~~ will run).

∴ Unsafe Expression.

(These are not in the Relational algebra).

- Both TRC & RA have same expressive powers.

unsafe Exp → $\{ U.E \mid (x) \}$.

(the query which we can write in RA, can also write in TRC & even in DRC.)

But, SQL has more powers than these. SQL has more extra funcs.

* Examples :-

Q-1. Write a query in SQL to display {name of Suppliers}.

Q-2. Write a query in SQL to display {name of Parts whose color is Red}.

('Catalog → सूची')

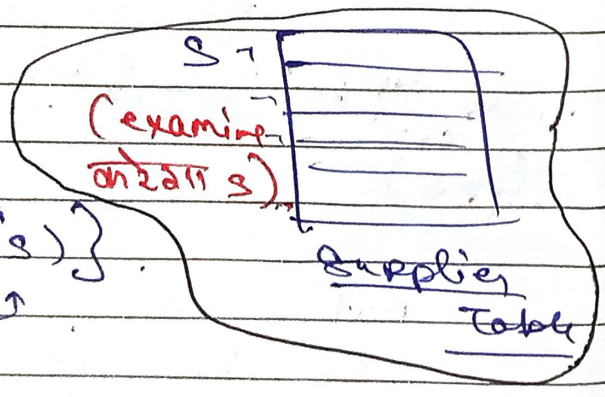
Q-3 Write a query in SQL to display SID of suppliers whose name is 'Varun' & address is 'chandigarh'?

Q-4 Write a query in SQL to display SID of suppliers who supplied some parts?

Q-5 Write a query in SQL to display Sname of suppliers who supplied some parts?

Q-6 Write a query in SQL to display Sname of suppliers who supplied some red color parts?

(1)



{ S.Sname | Supplier (S) }

(2)

{ p.Pname | Parts (p) ∧ p.color = 'Red' }

(3) { S.SID | Supplier (S) ∧ S.name = 'Varun' ∧ S.add = 'chandigarh' }

(4) Here, it extract ans. from the Catalog Table

{ C.SID | Catalog (c) }

Sid की Catalog Table में भी मिल जाती है.
But, Sname only in Supplier table.

(5.) Here, we need 2 Table ← Supplier, Catalog.

&

final ans. from Supplier Table.
(There Exists ∃).

$$\{ S.Sname \mid Supplier(S) \wedge \exists c\,(Catalog(c) \wedge S.Sid = C.Sid) \}$$

Here,

S [illegible] → free variable
c → bounded variable.

(जिसके साथ कोई ∃ या ∀ लगाते है).

(6.) Here, we need 3 Table. ← Supplier, Catalog, parts.

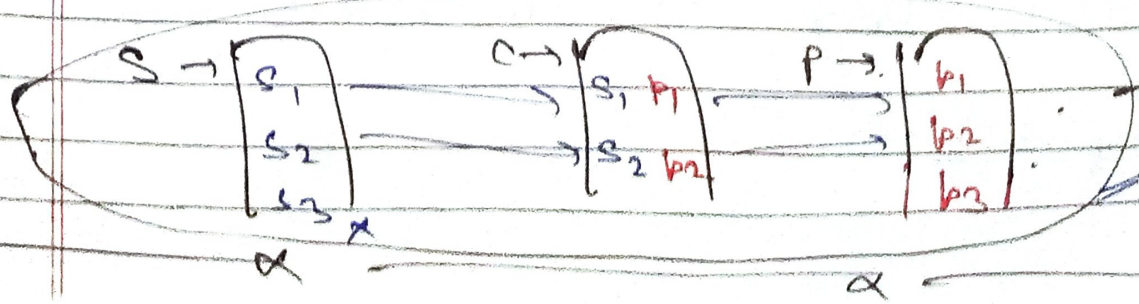
But,

ans from Supplier Table.

$$\{ S.Sname \mid Supplier(S) \wedge \exists c\,(Catalog(c)) \wedge \exists p\,(parts(p)) \wedge S.Sid = C.Sid \wedge C.pid = p.pid \wedge p.color = 'red' \}$$

S variable for Supplier Table.
C —"— Catalog —"—
P —"— parts —"—

S → free variable
P → bounded "

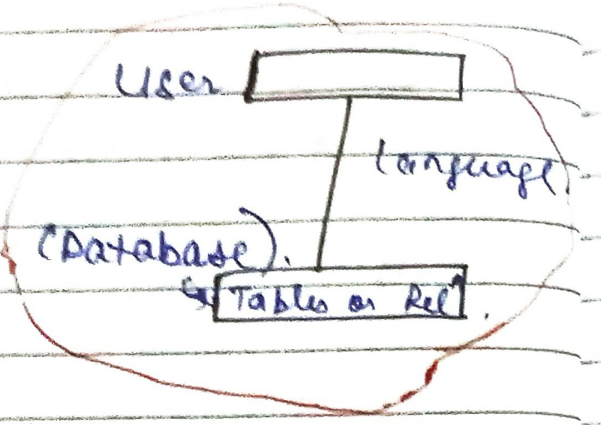


Intro to SQL :- (Structured Query language)

(1970)

→ EF codd give theory of store & fetching of data.

Relational Model:



By the help of Relational algebra & TRC (Tuple Rel. Calculus)

→ IBM converts this concept of database of EF codd into the Structure Query language (SQL).

→ Before,

SEQUEL → Simple English Query language

then,

SQL → Structure Query lang.

→ then, Oracle & Microsoft, etc. comes into this field of databases.

→ Now, No SQL is comes into market.

→ 10% data → Structure
90% data → Unstructure (Spark, Mongodb)

Properties :-

→ General purpose! → Applicability on multiple places. Ex- C/C++, but
Domain Specific! → only use in any particular field. Ex! -
SQL is in only Relational Database. (When we have to store & fetch data from the Relations (Tables), they only we use SQL), and except this, there is no general use of SQL.

1.) SQL is a domain- Specific language.
2.) SQL is a declarative language.

→ declarative lang → what to do.
→ procedural lang → what to do & How to do.
↳ (There is procedure that how to do)

later on,
PL SQL → procedural language.

3.) DDL, DML, DCL, TCL
These are diff. commands in SQL etc. to create, insert, fetch, control data. etc.

4.) keys & constraints. (Ex- P.K, F.K, C.key).
↓
(Primary key (P.K.) constraint,
F.K. "
Not, NULL, default).

aggregate → समान, family, ग्रुप जोड़, संगठन करना।

Exist / Not Exist.

5.) operators (like, between, In, Not in, Conditional).

⇒ We use these operators internally to use Query.

Ex:- IRCTC :- We query on this app, by APIs (Applica' programming interface).

Ex:- Train no', Seat no';
IRCTC based on structured Data.
(Train की Info. Tables के form में ही show होती है।).

6.) clauses (distinct, order by, group by, from having).
Use : majorly use this in query. also

7.) aggregate func's :- (max, average, count, min, sum,)

8.) Joins & Nested Query :-

9.) PL SQL (Triggers, func', Cursor, procedure) same as in C lang.

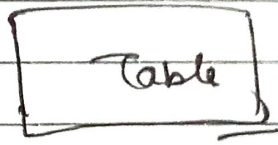
(#) what to do :-
In libraries of oracle or like SQL series, it is already defined that what select do, & what other commands 'do', we don't need to give procedure that how to do.

(27) All Types of SQL Commands :-

* DDL Commands :- (Data Definition language).

→ Deals with Schema (structure | Table).

- → Create
- → Alter (change Table Specifica's).
- → Drop - (Remove Table)
- → Truncate (Remove Data).
- → Rename.



* DML Commands :- (Data manipula" language).

→ If we have to change anything in data, then we use DML. (Manipulation).

- → Select
- → Insert.
- → Update.
- → Delete.

* DCL Commands :- (Data Control language).

→ who has control over the data.
who is authorised on that data.

- → Grant (Give permis")
- → Revoke. (take back that permis").

* TCL commands :- (Transac" control lang.)

→ Mainly used in Transac"s.