union of attr R_1, R_2 & R_3 are Equal to R. So, True. (Right decompos'n)

Now

→ Check Dependency preservation :→

- R_1 (AB), So, It has only A, B attr.

So, whatever Dependency we can made from these 2, make them & write in table.

We don't want Trivial F.D. :→

→ Trivial F.D.: in which intersec^n (∩) is not null.

i.e.,

Ex-1 $A \to A$ has $\cap \neq \phi$ (common is A).

Ex-2 $AB \to A$ has $\cap \neq \phi$ (common is A).

We don't want these bcz, trivial are by default, true. तो होती ही हैं like

$\boxed{A \to A}$ is definitely true.

So,

→ Only take non trivial F.D.: →

$(\cap = \phi)$

→ Now:

F.D :- { $A \to B$, $B \to C$, $C \to D$, $D \to B$ }.

$A^+ = ABCD$
$B^+ = BCD$
$C^+ = CDB$
$D^+ = DB$

So, acc. to these. select from the non-trivial F.D. (which we finally select). from table. =

Now
F.D. which we get from Table :-

$A \to B$
$B \to C$
$C \to B$
$B \to D$
$D \to B$.

Now, check whether ~~original~~ F.D. are possible through these or not.

So, F.D : { $A \to B$, $B \to C$, $C \to D$, $D \to B$ }.

Now,
$C \to D$ is not direct.
So, now take 'C' closure.

$C^+ = CBD$ ∴ $C \to D$ also preserved.

All F.D. are preserved. True, Ans.

---

36. Dependency Preserving Decompos^n Example

Q.2: Let R(ABCD) with F.D.
{ $AB \to CD$, $D \to A$ }.

Decompose $R_1(AD)$ & $R_2(BCD)$.

$R_1(AD)$	$R_2(BCD)$
$A \to D$ ✗	$B \to CD$ ✗
$D \to A$ ✓	$C \to BD$ ✗
	$D \to CB$ ✗
	$BC \to D$ ✗
	$BD \to C$ ✓
	$CD \to B$ ✗

attributes of $(R_1 \cup R_2)$ make R ✓

Now,

FD: $\{AB \to CD,\ D \to A\}$.

$A^+ = A$, $C^+ = C$

$B^+ = B$, $D^+ = DA$

$BC^+ = BC$,

$BD^+ = BDAC$

$CD^+ = CDA$

So,

F.D. we get from table:→

$\{D \to A,\ BD \to C\}$

Now, check ∀ original F.D.:→

$\{AB \to CD,\ D \to A\}$.

from table

$AB^+ = AB$

$D^+ = DA$ ✓

We can't get

$AB \to CD$, from our decomposed F.D.

Hence,

$\boxed{\text{Dependency preserved not}}$.

37 Joins & Its Types :→

1. **Join :-** When we have to join 2 or more table, to get result.

E_No	Ename	Add.
1	Ram	Delhi
2	Varun	Chd.
3	Ravi	Chd.
4	Amit	Delhi
5	Nitin	Noida

'Employee'

Dep. No	Name	E.no
D_1	HR	1
D_2	IT	2
D_3	MRKT	4
D_4	Finance	5

'Department'.

Select Address from Employee where Ename = 'Varun'
Output → Chd.

↳ In these basic ques^n, we don't need Join, we easily done those by their separate tables.

Now :-

Q :- find Ename of Emp whose working in HR.
Here, HR is in department Table
& Ename is in Employee Table
So,
Here we need Joins.

Note :- There must be some common att. in Tables to use Join. Here, E_no is common.

→ Join is Cross product
+
Select Statement (condition).

Its Types :→

- Cross Join (or) Cross product.
- Natural Join.
- Conditional Join
  - Equi "
  - Self "
  - Outer "
    - Left Outer.
    - Right "
    - full "

---

(38) "Natural Join" :→

→ Join = Cross product + Condi^n.

(m × n)

m

E.No	E name	Add.
1	Ram	Delhi
2	Varun	chd.
3	Ravi	Chd.
4	Amrit	Delhi

Employee

n

Dep No.	Name	Eno.
D_1	HR	1
D_2	IT	2
D_3	MRKT	4

Department.

→ first find :- Table name

Q :- "find the Emp. Names who is working in a department".

(Emp, Dept Means Cross product)

→ Here, we need both, - Table Employee name
- Department Table.

We do it by "Natural Join".

→ Common Att - 'E_No.'

So,

Whenever we have to equalise (बराबर) the values of the common Att, then, we always use NATURAL JOIN.

→ Now, Write Query :→

Select Ename from Emp, Dept where Emp. Eno = Dept. Eno ;

→ Output :→ Ram, Varun, Amrit

'E_No' - must be same in both the Tables.
i.e. (E_No, Emp_No.)

Emp		Dept	
E_No	E name	Dep No.	E No.
1	Ram	D_1	1
1	"	D_2	2
1	"	D_3	4
2	Varun	D_1	1
2	"	D_2	2
2	"	D_3	4
3	Ram	D_1	1
3	"	D_2	2
3	"	D_3	4
4	Amrit	D_1	1
4	"	D_2	2
4	"	D_3	4

So, finally we get 3 rows.

Eno	Ename	DepNo	Eno
1	Ram	D_1	1
2	Varun	D_2	2
4	Amit	D_3	4

Ram
Varun
Amit → Ans.

Now, how to write Actual Query :→

→ Select E_name from Emp Natural Join Dept;

---

39. Self Join :→

→ in which the Table is joined to itself.

S-id

Referencing Cid Table.

S-id	C-id	Since.
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

student course

Study

Q: find student id who is enrolled in not least 2 co

→ Now, we do self join.

- SQL query always start from - 'from' (sequal)

Query :→

Select ____ (from study as T_1, study as T_2;)

';' → means Cross product.

- So, here we make same Table 2 times & name as T_1 & T_2.

S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

T_2 (m)

S-id	C-id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

(n) T_1

m×n

T_1		T_2	
S_1	C_1	S_1	C_1
S_1	C_1	S_2	C_2
S_1	C_1	S_1	C_2 ✓
S_2	C_2	S_1	C_1
S_2	C_2	S_2	C_2
S_2	C_2	S_1	C_2
S_1	C_2	S_1	C_1 ✓
S_1	C_2	S_2	C_2
S_1	C_2	S_1	C_2

Condi^n :→

1 student (S-id) need 2 course (C-id)

< > → 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 T_1 & T_2 both have S-id,
So,
we can use any $T_1.Sid$ or $T_2.Sid$

Ans - S_1

---

(40) EQUI - Join :- → (=)

(कोई भी attr के बीच (=) लगा सकते हैं)

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

Employee

Dep_No	Loco^n	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

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

Q:- Find the Emp name who worked in a department having locaⁿ 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.Locaⁿ ;
```

Emp.			Dept.		
Delhi	1	Ram	D_1	Delhi	1
"	1	"	D_2	pune	2
"	1	"	D_3	patna	4
Chd	2	Varun	D_1	Delhi	1
"	2	"	D_2	pune	2
"	2	"	D_3	patna	4
"	3	Rani	D_1	Delhi	1
"	3	"	D_2	Pune	2
"	3	"	D_3	patna	4
Delhi	4	Amit	D_1	Delhi	1
"	4	"	D_2	pune	2
"	4	"	D_3	patna	4

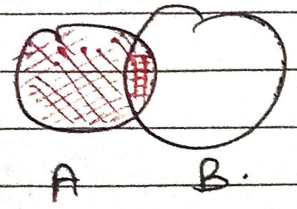
→ Ram

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)

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.dept no = dept.dept no.)
```

left

→ Condi'n of Natural Join. (Common att. को बराबर करना।)

Relational database always shows output in Table form.

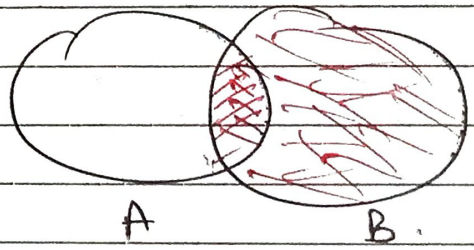
output Table :-

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

→ (Left की तो सभी Row आती ही हैं)

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	Amit	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.)

(43.) Relational Algebra :→ (1970).
(procedural lang. as 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.
(collec^n of mathematical Expres^n).

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

"operators"

Basic operator	Derived operator.
→ projection (π)	→ Join ($\bowtie$)
→ Selection (σ)	→ intersect ($\cap$)
→ Cross product ($\times$)	$(x \cap y) = x - (x - y)$.
→ Union ($\cup$)	
→ Rename (ρ)	→ Division ($/$, $\div$)
→ Set Difference ($-$)	

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

(44.) Projection in Rela^n^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). $\pi_{\text{Roll no.}}$ (Student).

Roll no.
1
2
3

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

We fetch the 2 colm (roll no. & name) from the student (Table).

Output →

Roll no	Name
1	A
2	B
3	A

Note: It only gives unique Answer.

Q.2 # Query :- π_{name} (Student).

Name
A
B

→ (only name here).

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

45. Selection in Relational algebra (R.A.) :→

σ (Sigma).

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'.

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

↓

Final.

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

π always last में operate होगा।

* We can also find 2 at a time.

Ex:- $\pi_{name, Age}(\sigma_{rollno = '2'}(student))$ → B, 21

Note:- Project^n (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.

→. $3 + 3 = 6$ $(m+n)$.

$2 \times 2 = 4$ $(x \times y)$.

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

→ $(R_1 \times R_2)$

R_1 R_2

→ 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: 1, 2, 3 S_2: 3, 4

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

→ It is not Commutative :→

i.e.

$A - B \neq B - A$

Condi^n :-

1.) No. of Columns must be same in no.
2.) Domain of every column must be same. (data of same type). (Ex- Numeric). (1+A) ×.

Ex :-

Rollno	Name
1	A
2	B
3	C

(Student).

Emp-No.	Name
7	E
1	A.

(Employee).

⇒ (Student) - (Employee) [: A दूर हो गया]

⇒

Rollno	Name
2	B
3	C.

⇒ (By default, पहले Table की ही 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 Operⁿ 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 colⁿ must be same in no.
2. Domain of every colⁿ must be same.

→ (Student) ∪ (Employee)

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

→ (Left side Table ⁿ आएगी).

Easy Trick! -

2) $(\pi_{name}(Student) \cup \pi_{name}(Employee))$.

Name.
A
B
C
E.

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

Note: 1

Roll no	name
1	A
2	B
3	C

Numeric

Name	Emp No.
E	2
A	1.

alphabet

not same domain.

(ye, 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.

(ele derived them, from normal operators).

/, ÷ (×, −, π, σ).

→ ele use these

⇒ Query: Retrieve Sid of students who enrolled in every course.
all

Date :
Page:

⇒ Every (या) all पढ़ते ही Division method :

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

'Enrolled'

Cid
C_1
C_2

'Course'

Notation of Division Method :→

$A(x, y) / B(y)$ = it results 'x' values for that there should be tuple $\langle x, y \rangle$ for every y value of Relⁿ B.

Here,

$$\boxed{E(Sid, Cid) / C(Cid) = S_1}$$

ie,
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).

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

$S_1 \times C_1$

$S_2 \quad C_2$.

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).

$$\pi_{sid}(\text{Enrolled}) - \left(\pi_{sid}\left(\left(\pi_{sid}(\text{Enrolled})\right) \times \pi_{cid}(\text{course})\right) - (\text{Enrolled})\right)$$

S_1 − S_2
~~S_2~~ S_3
~~S_3~~

⇒ S_1 Ans.

So- Tuple Calculus in DBMS :→

Rela'al Calculus (RC)
- TRC (Tuple or rows)
- DRC (Domain or Attr. or column)

⇒ Tuple Rela'al Calculus is a non-procedural query lang. (only tells what to do & not that how to do). unlike Rela'al 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^n s :- We use one variable here. Ex - (Supply Ids. from a supply table) → only 1 condi^n.

→ It also uses quantifiers :

- $\exists t \in r (Q(t))$ = "there exists" a tuple in t in rela^n r such that predicate Q(t) is true.

- $\forall t \in r (Q(t))$ = Q(t) is true "for all" tuples in rela^n 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 में फंस जाएगा।).

∴ unsafe Expresⁿ.

(These are not in the Relⁿal algebra).

- Both TRC & RA have same expressive powers.

unsafe Expⁿ ↔ U.E (x).
नहीं होता चाहिए।

(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 xtra funcⁿ.

Examples :→

Q.1. Write a query in SQL to display Sname of Suppliers?

Q.2. Write a query in SQL to display Pname of Parts whose color is Red?

('Catalog → सूची')

Date :
Page :

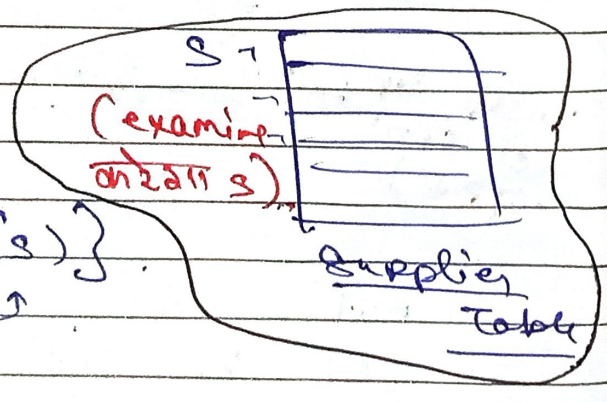
Q-3 Write a query in SQL to display SID of suppliers whose name is 'Varun' & address is 'chandigarh'?

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

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

Q-6 Write a query in SQL to display Sname of suppliers who supplied same 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 तो Catalog Table में भी मिल जाती।
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 → free variable
c → bounded variable

(जिसके साथ हम ∃ या ∀ लगाते।)

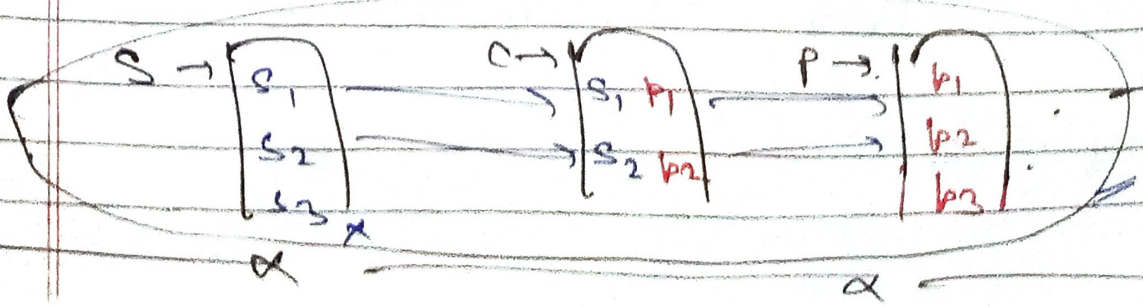
(6.) Here, we need 3 Table — Supplier, Catalog, parts.

But,
ans from Supplier Table.

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

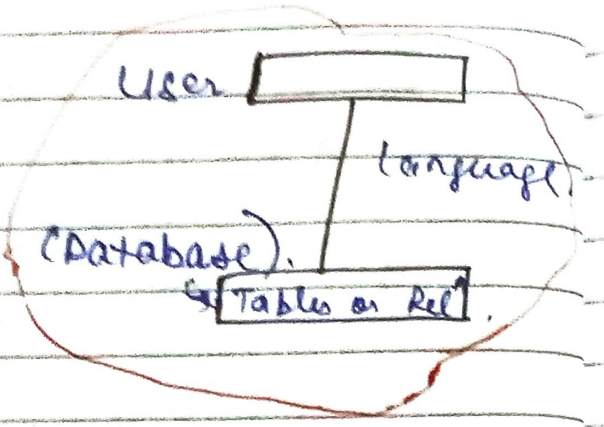
S variable for Supplier Table.
C —— " —— Catalog ——.
P —— " —— parts ——.

S → free variable
C, 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 Relational 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, Mogodb)

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 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, संग्रह 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 (Applicⁿ 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).
We also majorly use this in query.

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.

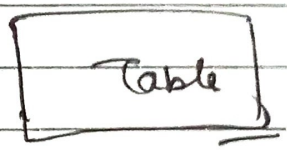
(#) What to Do :-
In libraries of oracle or like SQL series, it is already defined that what select do, & what other commands 'do', we don't need to give procedure that how to do.

52 All Types of SQL Commands :-

* **DDL Commands :-** (Data Definition language).

→ Deals with Schema (structure | Table).

- → Create
- → Alter (change Table specifica^n).
- → Drop (Remove Table)
- → Truncate (Remove Data).
- → Rename.



* ~~Data~~ **DML Commands :-** (Data manipula^n 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^n)
- → Revoke. (take back that permis^n).

* **TCL commands :-** (Transac^n control lang.)

⇒ Mainly used in Transac^ns.

Ex:- Shopping, Bill payment – online

- Commit (If Transac^n is commit (done), then save data)
- Roll back (If Transac^n fails).
- Save point. (Ex: we can save after 20-30% of transac^n).

Ex:- Like in Downloading & Buffering, we have save points. Ex: If downloads fails on 90%, then if it starts from '0' again or from 90%). This is save point.

Note:- To Execute these commands, we can use Oracle, My SQL, Server, etc.

* Constraints :→ We mostly use these Constraints in DDL & DML.

- Primary key (for uniqueness)
- Foreign key (" reference).
- Check
- Unique
- Default
- Not Null (mandatory *).

→ Constraints are rules which we made for store the data.

S3- Create Table in SQL with execution :→

like SQL

→ In DDL Commands, very first command is Create Table.

→ Now, we use Oracle Syntax. (In other like My SQL, SQL server, there is only little diff.).

```
Create table <table-name>
(
 Column1 name datatype ,
 Column 2 name datatype ,
 Column 3 name datatype
);
desc table-name ;
```

// no space
Ex: abc_xyz.

// describe.

Ex:

```
Create table emp
(
id int ,
name varchar2 (20) ,
Salary number (10)
);
desc emp ;
```

→ fixed type of datatype (2 digit fixed)

→ variable — " — (can be increased)

54. Alter Command (DDL) in SQL :→

Use → (बनी हुई टेबल में अगर कुछ change करना है तो।)

Ex:

Student

int ID	name varchar (20)

→ already है

→ Add or Remove colmns/s.
→ Modify datatype.
→ Modify datatype length.
→ Add Constraints
→ Remove Constraints
→ Rename colmn / table.

Id	name	Address

student.

Alter table student Add address varchar(30);

(,);

* multiple colmns.

⇒ Modify datatype:-

Ex:- from int to varchar.

```
* Create table emp
(
id int,
name varchar (10)
);
alter table emp add address varchar (10);
desc emp1;
alter table emp drop colums address;
" " " modify id varchar (30);
" " " rename colums id to seg_no;
—— " —— rename to emp1;
alter table emp1 add primary key (seg_no);
```

X

(55.) DB Alter & Update : →

(*) **Alter** : only change in structure, (DDL) & not in data.

Ex :

Emp

ID	name	Salary	Email
1	A	10k	
2	B	20k	
3	C	30k	

} → can only change in this structure.

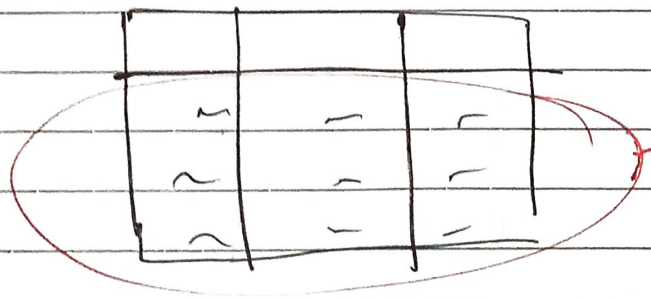
→ Int → varchar

→ name varchar (10) → 20

→ ID → Eid. (colm name change)

→ Emp → Emp_detail (table name ")

(*) **update** : → can only change in data, (DML) & not in structure.



→ can only change in data.

```
update Emp
Set Salary = Salary * 2;
```

→ for all students

```
update Emp
Set Salary = Salary * 2
Where id = 1;
```

→ for only student of id = 1.

Date :
Page:

56 D|B Delete, Drop & Truncate :-

Delete :-
(DML).

Syntax :-
Delete from table name

Ex: Delete from Student, → All rows deleted.

Desc Student;

Student.

ID	name
1	A
2	B
3	C

ID	name

Note :- But here, also flexibility, we can apply condi^n for delete some particular rows.

Ex: Delete from Student
Where id = 1;

⇒ It is a slower process :- bcz, it also creates log.

> log :- ईमारी file Completely delete होने से पहले, Computer एक temp. file बनाता है, to restore data. उसे log कहते हैं।

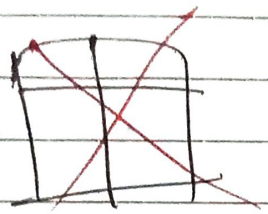
Ex: on delete, our data comes into Recycle Bin. (It is log file).

→ Rollback;

(possible, but it creates log).

(*) Drop:→ Delete the complete table at once.
(DDL)

```
Drop table student;
Desc student;
```



→ no object found.
(all delete).

Rollback; ✗ (not possible).

(*) Truncate:→
(DDL)

```
Truncate student;
Desc student;
```

All rows delete at once.

ID	name

→ faster Process.

→ Rollback; ✗ (but no log).

(S7) Constraints in SQL:→

→ Constraints means (conditions) restric^n.

Ex: In gmail:→ (2 account Ids can't be same).

anurag@gmail.com ✗ (not present).
anurag123@gmail.com ✓ (present). ✓

1. Condi^n → that new mail-id must be unique.

2. Condi^n → Length of password must be 6 digit, Capital letter etc.

[Note:-] We apply conditions on Attributes (column). and the data which fulfill these condi^n, only comes into the column^s.

1.) Unique:- → Ex:- Mobile no. (no duplicate can exist in that m.No. col^n.)

2.) Not Null:- *

(We can't skip that col^n),

i.e. * - Mandatory. (value देनी ही पड़ेगी)

* 3.) Primary key = Unique + Not Null.

Ex:- Reg. no.

4.) Check:- →

age int

Ex: x (-10)

[check (age>18)] → 10 x

→ 20 ✓

i.e. (एक particular Domain में से ही User Id बनेगी।)

[Domain - fixed.]

5.) Foreign key :-

6.) Default :-

[Salary int default 10k].

If we don't fill anything, they default Salary colm take value → 10k

→ We can use multiple constraint in a colm also.

58. SQL Queries & Sub-Queries :-

(from Beginning to end).

Q:-

Emp.

E_id	E_name	Dept	Salary
1	Ram	HR	10k
2	Amrit	MRKT	20k
3	Rani	HR	30k
4	Nitin	MRKT	40k
5	Varun	IT	50k

(i) Write a SQL Query to display maximum Salary from Emp table.

(ii) Write a SQL Query to Display Employee name, who is taking maxm Salary.

Note :- aggregate func :- for max, count, average,

<> → not equal to, = ≠

(i). Select max (salary) from Emp;

Output → 50,000

(ii) ule here, take help of Nested Query or Sub-Query. (Query inside a Query).

→ Select E_name from Emp where Salary = (Select max (Salary) from Emp);

Outer Query. Inner Query

Output → Varun

⇒ Inner Query execute first 1 time

Ex-

Outer		Inner	
10k	=	50k	×
20k	=	50k	×
50k	=	50k	✓

59

(iii) Write a SQL query to display 2nd highest Salary from Emp table?

(iv) → Write a SQL query to display Employee name who is taking 2nd highest Salary?

(iii)

Logic → Highest Salary - remove & Rest Salary - find highest.

(It is 2nd highest in table)

Outer

Max ← 10k - (50k)
20k -
30k -
40k -
50k - ×

↳ Inner

40k Ans

(iii) Query :-

```
Select max (salary) from emp where
salary <> (Select max (salary) from emp);
```

Sol.

ok

(iv). { = , when we have to compare only with 1 value. }
{ in , when we have to compare with multiple values. }

Ex :-

2 = 2 ✓

{ 2 = 1, 3, 2, 4, 5 (Wrong way)
2 in (1, 3, 2, 4, 5) ✓ (True. }

Query :- ✗ only add name in (iii)

```
Select e-name from emp where
salary = ( Select max (salary) from emp where
salary <> (Select max (salary) from emp));
```

Output attition.

60. (v) Write a Query to display all the dept names along with no. of emps. working in that ?

Output :-

HR	2
MRKT.	2
I T.	1

Ex :- In a university, find how many students are there in diff. branches.

AI → 101
CSE → 100