⇒ Relational Model =

Retrieve the data.

Procedural lang. {

Theoritical	Practical
* Relational algebra	* SQL.
* " calculus	

Relational Algebra.

Unary opertion	Set theory option	Bina opertion	Extended (*) derived op.
* selection (σ)	* union (∪)	* Join (⋈)	* outer join
* projection (π)	* Intersection (∩)	- Inner join	- left outer
* Rename (ρ)	* Cross product (×)	- Theta j	- right outer
	* minus / set difference (−)	- Natural j	- full outer
		- division j	* Aggregate ().

selection → row table - ररर values select करता.
↓
horizontal subset. → σ

I Unary operation =

1) Selection =

* Sigma (σ)

* Horizontal subset. [we get rows full]

Selection syntax → syntx → σ (condition) (relation name)

eg →

Student.

id	name	%.
1	AK	90
2	bk	80

σ (% > 80) (Student)

o/T. →

1	AK	90

2) projection (π) =

* pi (π)
* vertical subset.

syntx. → π col₁, col₂ ... (relation name)

eg → π id, name (Studnt)

↳ I get id & name in vertical data

→

1	AK
2	bk.

eg → π id, name σ (% > 80) (Student)
(condition)

→

1	AK

3) Rename (ρ) = change table name.

* RHO (ρ)

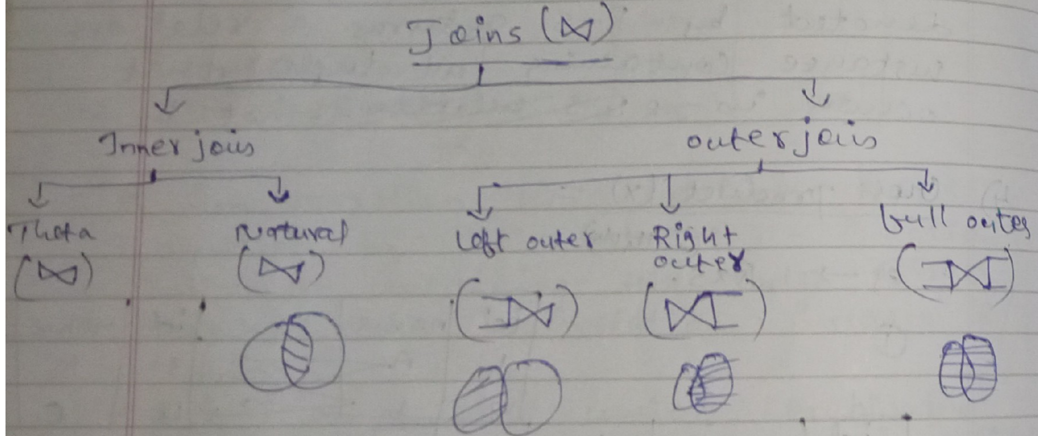
syntx → ρ (newname) (Relation name)

e.g → ρ (final) (Studnt)

[Studnt തന്റെ name ഇനി final ആയിരിക്കും.]

* cross product / cartesian product. operation
b/w 2 relation R & S denoted by
R × S returns a relation instance
whose schema contains all attributes of
the relation instance S followed by all
attributes of the relation instance S.
* Thus output relation R × S as
a schema which is the concatination
of R & S.

=) Joins in DBMS =



1) Theta join =

eg.→

C

car model	price
C1	4000000
C2	5000000
C3	8000000

J

Jeep model	Price
J1	6000000
J2	10000000

J ⋈ J price > c price
cond.

C ⋈ J

Jeep m	price	carmodel	price.
J_1	6000000	C1	4000000
J_1	6000000	C2	5000000
J_2	10000000	C1	4000000
J_2	10000000	C2	5000000
J_2	10000000	C3	8000000

C3 × J1 false. (b c2, J > c)
C3 ✓ bcz ← J_2, J > c True.

2) Natural Join =
relation will get join with common attribute

eg → S

id	name	Dept
101	AK	1
102	bK	2
103	cK	3
104	DK	"

D

Dept	Dname
1	CSE
2	ECE

id	name	Dept	Dname.
101	AK	1	CSE
102	bK	2	ECE.
104	DK	1	CSE

3) Left outer join =

resultant relation will have all tuples of S,
& common tuples of D.

S

id	name	Dept.
101	AK	1
102	bK	2
103	cK	3

D

Dept	Dname.
1	CSE
2	ECE.

id	name.	Dept	D name.
101	AK	1	CSE
102	bK	2	ECE
103	CK.	3	NULL

(3 ঞঞ ßo null)

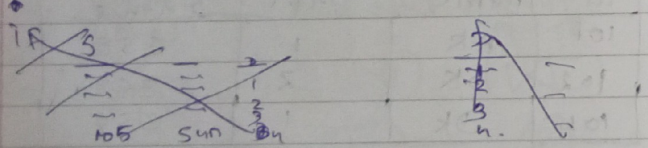
2) Right outer join = S ⟖ D

S D

* Resultant relation will hve all the tuples of D & common tuples of S.

⇒

Dept	D name	id	name.
1	CSE	101	AK
2	ECE	102	bK
3	NULL	103	CK.



5) Full outer join =
S ⟗ D.

Result relation will hve all the tuples of D & tuples of S.

⇒

id	name	dept	Dname.
101	AK	1	CSE
102	bK	2	ECE
103	CK	3	NULL

full outer join both their operations tuples from the left relation that didnot match from right relation, as well as tuples from the right relation that didnot match any from left relation & adding them to the result of the join

(detail) 3) left outer join takes all tuples in the left relation that
(def) didnot match with any tuples in the right relation matche the tuples with null values for all other attributes from the right relation & adds them to the result of natural join.

4) Right outer join is symmetric the left outer join it
(def) pads the tuples with the right relation that did not match any from the right relation with nulls & add them to the rslt of the natural join. Thus all info from the right relation is present in the rslt of the right outer join.

II ⇒ Relational calculus =

* Non-procedural lang.

* 2 types ⟶

1) Tuple relational calculus. (rows)
2) Domain (values) r. c.

1) <u>Tuple r. calculus</u> =

- notation → {T / P(T)} (or)

{T / condition(T)}

where T → resulting tuples.
P(T) → condition used to fetch T.

(only 1 attribute)

Eg → ① Author

name	article
AK	DBMS
bk	python.

T / P(T) ⇒ {T name / Author (T) AND article = "DBMS"}

(DBMS എഴുതിയ ആളുടെ name കിട്ടും)

o/p → AK.

② Student

Fname	Lname	age
AK	BK	20
CK	DK	40

Display last name (Lname) where age is > than 30.

⇒ {T.Lname / Student(T) AND age > 30}

2) <u>Domain r. calculus</u> =

Notation → $\{a_1, a_2, a_3 \ldots a_n / P(a_1, a_2 \ldots a_n)\}$.

where, a_1, a_2 → attributes.
P stands for formula build by the attributes

(more than 1 attribute)

eg →

Author

name	article	date	pg
A	DBMS	1.2.3	20
B	SQL	4.5.6	10
C	Python	7.8.9	30

Display article, pg & date from article = "DBMS."

⇒ {<article, pg, date> / Author AND article = "DBMS"}

① <u>Tuple r. calculus</u> = (def)

Tuple that takes on tuples of a particular relation schema as values (i.e) every value assigned to a given tuple variable as same no. & types of attributes.

(2) <u>Domain r. calculus</u> = (def)

Domain variable is a variable that range over the values in a domain of some attribute.

⇒ ER model = (Entity relationship model)

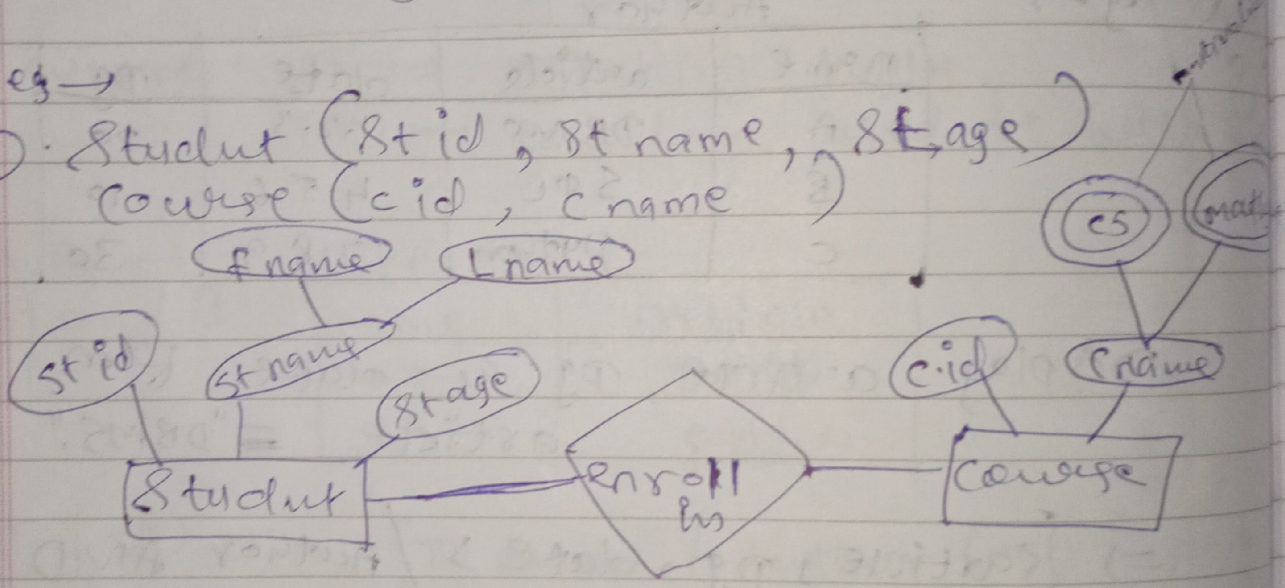
* It describes the str of DB with the help of diagram → ER model / diagram

- Entity
- Attributes
- Relationship

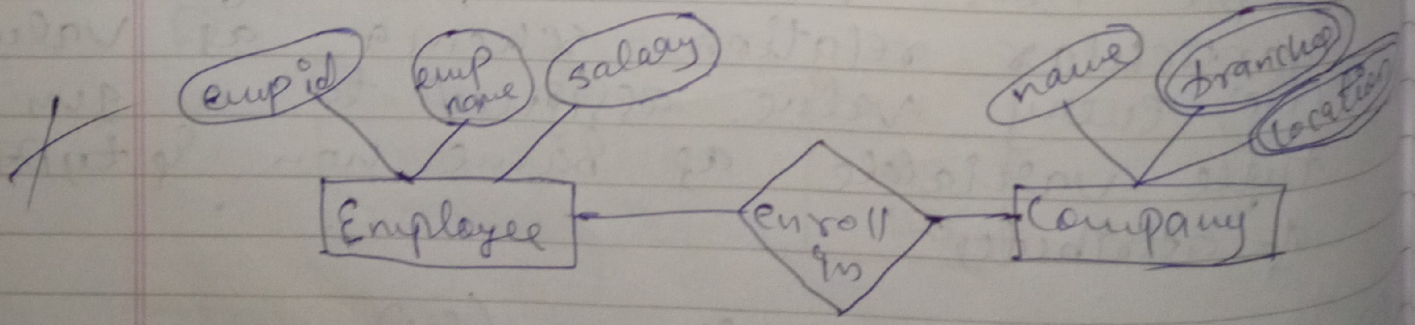
} main comp.

* eg →

① Student (st id, st name, st age)
Course (c id, c name)



② Employee (emp id, emp name, Salary)
Company (name, branches, location)



~~Relational DB~~