Module 2 :- Relational Approach

Relational algebra and calculus : Relational algebra, Selection and projection, set operations, renaming, joins, division, examples of algebra queries, relational calculus : Tuple relational calculus, Domain relational calculus, expressive power of algebra and calculus.

Basics of Relational Algebra :-

→ Relational Query languages

* Query language :-

A query language is a language in which the user request for the information from the database

. It is a higher level than that of a standard programming language. (front end)

→ categories in query language

1) procedural query language.

2) Non-procedural query language.

① procedural query language :-

→ A sequence of operations on the database.

→ To compute the desired result.

→ What data is required ? in database.

→ How to retrieve those data ?

→ Example :- Relational Algebra

② Non-procedural query language

→ user describe only the desired information

→ what data is required?

→ No need to describe how to retrieve those data?

→ Ex :- Relational calculus
  → Tuple
  → Domain.

Relational Algebra.

Algebra :- It is the branch of mathematics that helps in the representation of problems or situations in the form of mathematical expressions.

1) Abstract :- To study the algebraic structures such as loops, rings, fields, it has wide application area in the field of cryptography.

2) Linear :- It deals with linear equation, linear mapping and it is used in the application area like the weather forcasting.

3) Boolean :- It deals with the variables that are the fourth value that are true or false and it has wider spread application in the cu

of digital secm circuitry design in personal computers pocket calculators, CD players, cell phones etc.

* Relational algebra

- It is a procedual query language.
- It is a set of operations on relation(s)
- It is a set of algebraic operations (addition etc)

Relational algebra is a set of operation which takes one or more relations as input and procedures a relation as output.

- Input : one or more relations
- output : A relation.
- It provides a theoretical foundation for relational database.
- Structured query languages (SQL)
- It allows us to understand db operations in more detail and motivate us to write optimized queries.

* Relational algebra is one of the two formal query languages associated with the re-lational model. Queries in algebra are composed using a

Collection of operations.

- A relational algebra expression is recursively defined to be a relation, a unary algebra operator applied to a single expression, or a binary algebra operator applied to two expressions.

→ Relational Algebra Operations.

1) Fundamental operations:
   - Select (σ)
   - project (Π)
   - Union ($\cup$)
   - Set Difference ($-$)
   - Cartesian product ($\times$)
   - Rename (ρ)

2) Additional operations:-
   - Set intersection ($\cap$)
   - Assignment operation ($=$)
   - Natural join ($\bowtie$)
   - Division operation ($\div$)
   - Outer join
     - left outer join ($\bowtie$)
     - Right outer join ($\bowtie$)
     - Full outer join ($\bowtie$)

1) Selection (σ)

- Select tuples that satisfy a given condition
- It is denoted by lowercase greek letter Sigma (σ)

Syntax :-

$$\sigma_{<\text{selection\_condition}>}(\text{Relation})$$

ex :-

$$\sigma_{D\_ID = 2}(\text{Employee})$$

- Comparison operators : $=, \neq, <, \leq, >$ and $\geq$.
  (less, greater)
- Connectives : AND ($\wedge$) OR ($\vee$) and NOT ($\neg$).
  (Negation)

The relational algebra includes operator to select rows from a relation (σ).

- Select rows that satisfy selection condition.
- No duplicates in result
- Schema of result identical to Schema of input relation
- Result relation can be the input for another relational algebra operation

2) Project (Π)

- It returns its argument relation with certain attributes left out.
- It is a unary operator.
- It is denoted by the uppercase Greek letter Pi (Π)
- Basically a relation is a set.
- In the result, the duplicate rows are eliminated.
- Syntax :- $\Pi_{Attri1, Attri2 \ldots}$ (Relation)
- Selected attributes we use project.

① Ex :- List all instructor ID, name and salary but do not care about the dept_name.

- $\Pi_{ID, Name, Salary}$ (Instructor)

ID	Name	Salary

Instructor

ID	Name	Dept_Name	Sal
10101	John	Bio	65k
12121	Robin	CSE	90K
25252	Alya	ECE	40K
26589	Yusuf	finance	95K
54789	Ravi	Music	60K
78787	Raj	physics	87K
87458	Jayant	His	75K
96985	Pratik	CSE	89K
12547	Neil	finance	80K

Ex: Find the name of all instructors in the CSE

Sol :- $\Pi_{Name}(\sigma_{dept\_name = 'CSE'}(Instructor))$

O/P :-

Name
Robin
pratik

3) Union (U)

- Any relation is a set
- Similar to union operation in set Theory.
- It is a binary operator.
- It is a set of all objects that are a member of A or B or Both.
- Like project, the duplicate rows are eliminated
- It is denoted by U.
- Syntax : $\Pi_{columns}(Relation\_1) \cup \Pi_{column}(Relation\_2)$

R - Alphabets
A
B
C
E
F

S characters
A
B
$
E
F
1
#

R U S
A
B
C
E
F
$
1
#

Ex:- List all customer names associated with the bank either as an account holder or a loan borrower.

Sol:- $\Pi_{cu\text{-}name}(Depositor) \cup \Pi_{cu\text{-}name}(Borrower)$

Depositor (R)

cu-Name	ACC-NO
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

Borrower (S)

cu-Name	loan-no
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

O/P:-

cu-Name
Tom
Amy
Rose
John
Smith
Jack

Two important Conditions For R ∪ S to be Valid.

1. R & S must be of same arity.
2. For all i

   Domain of the ith attribute of R_2

   Domain of ith attribute of S

4) Set Difference (-)

- It is like the same set difference
- It is a binary operation.
- To find tuples that are in one relation but are not in another.
- R - S = Tuples in R but not in S.
- It is denoted by minus (-).

R

Alphabets
A
B
C
D
E
F

S

characters
A
B
$
L
Y
1
#

R-S

C
D
E
F

Ex:- 1 List all customer names those who have a deposit account but not availed loan.

Depositor →

Cu-Name	Ac-No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

BORROWER →

Cu-Name	Loan-No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

R-S →

Cu-Name
Tom
Amy

Sol :- $\Pi_{Cu\text{-}Name}(Depositor) - \Pi_{Cu\text{-}Name}(Borrower)$

5) Cross product (x)

- It is associates every tuple of R_1 with every tuple of R_2.
- It is a binary operation
- It is denoted by cross (x) symbol.
- $R_1 \times R_2$ = All possible pairing
- Same attribute may appear in R_1 & R_2.
- R = Depositor x Borrower.

Ex :-

D →

Cu-N	Acc-No
Tom	A-101
Rose	A-304

B →

Cu-N	Loan-N
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

6) Rename (P)

- Relations in the database have names
- The results of relational-algebra expressions do not have a name.
- It is useful to be able to give them names.
- It is denoted by the lowercase Greek letter rho (P).
- Syntax : $\rho_x(E)$ → Result of operator (x → old ~~table name~~ new)
- $\rho_x(A1, A2, \ldots An)$ (Expression)
- The Rename operation is used to rename the o/p relation,

Ex:-

$$\rho(C(1 \to Sid1,\ 5 \to Sid2),\ S1 \times R1)$$

Combination of two tables of cross product.

- Then the column will be similar, at that time we should change the column name.

- Relation (Table)
- column

} Rename:

ex:- ρ (Student1, Student)

To rename ~~operator~~ student relation to Student1. rho operator is used here.

ρ

Additional Operations :-

1) Set intersection ($\cap$).

- Any relation is a set.
- Similar to intersection operation in Set Theory.
- It is a binary operator.
- It is a set of all objects that are a member of both A & B.
- It is denoted by $\cap$.
- Syntax :- $\Pi_{column}(Relation\_1) \cap \Pi_{column}(Relation$
- $R \cap S = R - (R - S)$

R → Alphabets

A
B
C
D
E
F

S → Characters

A
B
$
E
F
1
#

R ∩ S

A
B
E
F

Ex 1 :- Find the names of all customers who have deposited money & also availed loan.

Sol :- $\Pi_{cu\_Name}(Depositor) \cap \Pi_{cu\_Name}(Borrower)$

R ∩ S →

Cu-Name
Rose
John

$R \cap S = R - (R - S)$

R → Alphabets	S → Characters	R ∩ S	R - S	R - (R - S)
A	A	A	C	A
B	B	B	D	B
C	$	E		E
D	E	F		F
E	F			
F	1			
	#			

Hence $[R \cap S = R - (R - S)]$

2) Assignment operation (=)

- Similar to assignment operator in programming languages.
- Denoted by $\leftarrow$ or $\leftarrow$
- It is a binary operator.
- The db might be modified if assignment to to a permanent relation is made.
- Useful in the situation where it is required to write relational algebra expressions by using temporary relation variable
- Ex :- $P = R \cap S$.

Ex:- $Temp_1 = Expression1$

$Temp_2 = Expression2$

$Result = Temp_1 + Temp_2$

(or)

$Temp_1 \leftarrow R \times S$

$Temp_2 \leftarrow \Pi_{courseid}(\sigma_{semester='spring' \wedge year = 2010(Section)})$

$Result \leftarrow \Pi_{RUS}(Temp_2)$

Division operator (/) or ($\div$)

R_1 / R_2 or $R_1 \div R_2$

if query contains all or every then only we use division operator

$R_2 \subset R_1$ proper subset

R_1 / R_2 which means

the attribute which are present in R_1 but not in R_2

ex:- students

Name	Course
A	DBMS
B	DBMS
A	DS
B	DS
A	COA
B	COA

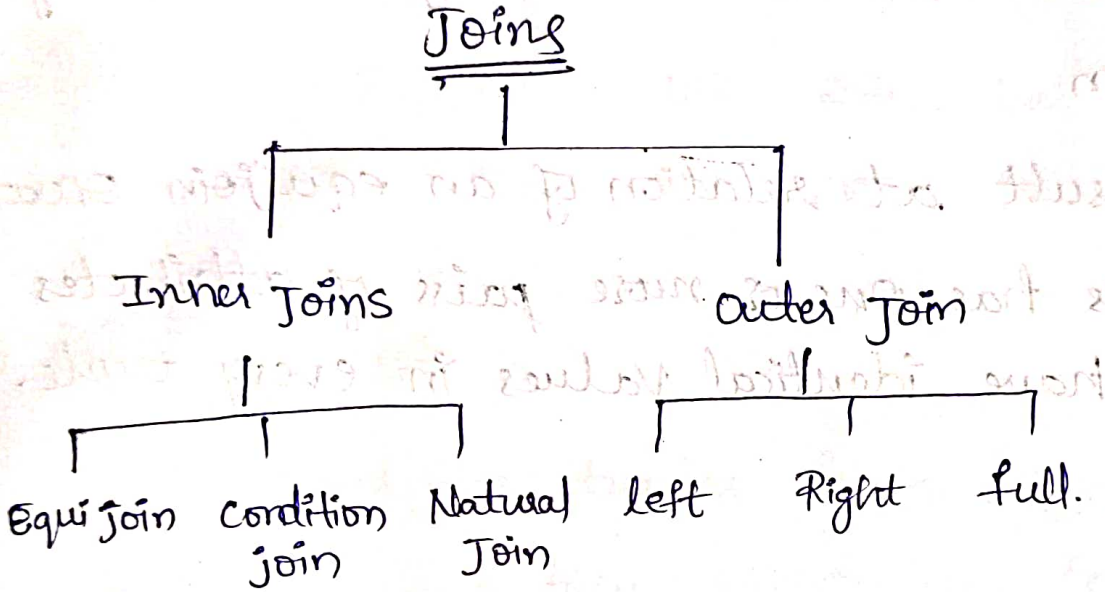
Course
DBMS
DS
COA

R_1 / R_2 → Output

Name
A

Joins

Combining fields from two tables by using values common.



Join :-

The join operation is one of the most useful operations in relational algebra and is the most commonly used way to combine information from two or more relations.

The most general version of the join operation accepts a join condition c and a pair of relation instances as arguments, and returns a relation instances.

Inner join

The inner join command returns rows that have matching values in both tables.

1) Equi join :-

Equi join operation is one in which the join condition consist only of the equality condition.

- The result at relation of an equijoin operation always has one or more pairs of attributes that have identical values in every tuple.

Eg:- Book

ISBN	Title	category	Price	pagecount	pid
1	Xyz	TB	200	50	P_1
2	ABC	NB	300	100	P_1
3	XWT	TB	100	200	P_2
4	TTT	NB	400	300	P_2

publisher

Pid	PName	add
P_1	WT	aaa
P_2	ZT	bbb
P_3	YT	ccc
P_4	XT	ddd

Book ⋈ publishes.

These two relation can be joined for tuples where book.pid = publishes.pid

o/p :-	ISBN	title	category	price	~~amount~~ Pagecount	Book.pid	P.pid	Pname
	1	XYZ	TB	200	50	P_1	P_1	WT
	2	ABC	NB	300	100	P_1	P_1	WT
	3	XWT	TB	100	200	P_2	P_2	ZT
	4	TTT	NB	400	300	P_2	P_2	ZT

2) Condition Join :-

It contains tuples from different relations provided they satisfy the condition

S1 :-

Sid	Name	rating	age
22	dustbin	7	45.6
31	lubber	8	55.0
58	rusty	10	35.0

R1

Sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$\sigma_{S1.Sid < R1.Sid}(S1 \times R1)$$

or

$$S_1 \bowtie_{S_1.Sid < R_1.Sid} R_1$$

Sl.Sid	Sname	rating	age	R_1.Sid	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.0	58	103	11/12/96

3) Natural Join :-

A natural join will remove one of the duplicate attribute from the result of equi join.

- The natural join of two relation R_1 & R_2 is obtained by applying a project operation to the equi join of two relation R_1 & R_2.

Course

Cid	Course	Dept
CS01	Db	Cs
ME01	mech	ME
EE01	Electronic	EE

HOD

Dept	Head
CS	Rohan
ME	Sara
EE	Jiya

Cid	Course	dept	Head
CS01	DB	CS	Rohan
ME01	mech	me	Sara
EE01	Electronic	EE	Jiya

$\Pi_{Cid, Course, dept, head}(\sigma_{dept = Head.dept}(Course \times Head))$

outer join

- The outer join operations is an extension of the join operation that avoids loss of information
- Contains matching tuples that satisfy the matching conditions, along with some or all tuples that do not satisfy the matching condition.
- It is based on both matched or unmatched tuples
- It contains all rows from either one or both relations are present
- It uses null values null significant that the value is unknown or does not exist.

Outer join = Natural join + Extra inform

1. Left outer join :- $[R_1 \bowtie]$

The left outer join, all the tuples from the left relation R_1 are included in the result relation.

- The tuples of R_1 which do not satisfy join condition will have values as Null for attributes

of R_2

Course

Cid	Course
100	DBMS
101	Mech
102	ECE

HOD

Cid	Name
100	Rohan
101	Sara
104	Jiya

Op :- Course ⋈ HOD

Syntax :-

Cid	Course	Name
100	DBMS	Rohan
101	Mech	Sara
102	EEE	NULL

2. Right outer join (⟖)

In right outer join, all the tuples from the right relation R_2 are included in the resulting relation. The tuples of R_2 which do not satisfy join condition will have values as NULL for attributes of R_1

O/P :-

Cid	Course	Name
100	DBMS	Rohan
~~101~~	~~Mech~~	~~Sara~~
102	ECE	NULL
NULL	NULL	Jiya

syntax :- left join

```
Select table1.column1, table2.column2,
    table2.column1 ------
From table1 LEFT JOIN
table2 ON table1.matching_column = table2.
matching_column;
```

Syntax :- Right Join.

```
Select table1.column1, table1.column2, table2.
    column1, ----.
From table1 RIGHT JOIN table2 ON table1.
matching_column = table2.matching_column;
```

3. Full outer join (~~IX~~)

In full outer join, all the tuples from both left relation R_1 and right relation R_2 are included in the resulting relation, their respective unmatched attributes are made NULL

Course(~~I~~ HOD

o/p :-

Cid	course	Name
100	DBMS	Rohan
101	mech	1
102	ECE	
104	NULL	jiya

Syntax :-

```
Select table1.column1, table1.column2,
table2.column1, ----
From table1 FULL JOIN table2 ON
table1.matching_column = table2.matching
- column;
```

Relational calculus :-

Relational calculus is an alternative to relational algebra. In contrast to the algebra, which is procedural, the calculus is non-procedural, or declarative.

Comes in two flavors :

Tuple relational calculus (TRC)

Domain relational calculus (DRC)

1) Tuple Relational calculus :-

A tuple variable is a variable that takes on tuples of a particular relation schema as values. That is, every value assigned to a given tuple variable has the same number & type of fields.

A tuple relational calculus query has the form $\{T \mid P\{T\}\}$

where T is a tuple variable &

P(T) denotes a formula that describes T.

The result of this query is the set of all tuples t for which the formula P(T) evaluates to true

with T = t.

The language for writing formulas.

P(T) is thus at the heart of TRC and is essentially a simple subset of first-order logic.

logical operatoss $\vee$, $\wedge$, $\neg$ also user quantifies.

Eg:- { T.name (result) / Author (T) (tablename) AND

T.article = 'database' }

(condition)

(OR)

{ R / $\exists$ T $\in$ Author (T.article = 'database')

$\wedge$ R.name = T.name) }

eg:-

FN	LN	Age
Pinky	Singh	36
dolly	Singh	31
Tom	King	27
Lucky	Pratap	28

1. Display the last name of those student where age > 30.

Solu :- $\{ t.LN \mid Student(t) \text{ And } t.age > 30 \}$

op :-

LN
Singh

2. Display all the details of student where last name is 'Singh'.

sol :- $\{ t \mid student(t) \text{ and } t.LN = 'Singh' \}$

op :-

FN	LN	age
Pinky	Singh	30
dolly	Singh	31

2) Domain relational calculus (DRC) :-

In DRC the records are filtered based on domain.

- It uses the same Operator as TRC
- Notation : $\{ a_1, a_2 ---- an \mid P(a_1, a_2 --- an) \}$

where a_1, a_2, a_3 are attributes.

P = Condition for fetch the data.

① { <article, page, subject> /
∈ author ∩ subject = 'database' }.

② Find the FN, age of student where age > 27

Sol :- { <FN, age> / ∈ student ∩ age > 27 }

- Correlated subquery

→ It is a subquery that uses values from outer query.

→ Top to down approach

(outer query (inner query)

Eg :- find all emp detail who work in a department.

Sol :- select * from emp where
Exsists (select * from dept where dept.eid
= emp.eid);

emp

emp.id	name	addle
1	A	Delhi
2	B	pune
3	A	hyd
4	B	Delhi
5	C	pune
6	D	mumbai
7	E	hyd.

Dept

Did	DName	Eid
D_1	HR	1
D_2	IT	2
D_3	MECT	3
D_4	Testing	4

...

- Nested query :-

Nested Query is a query ~~where~~ within another query.

- A subquery is a select statement that is nested within another select statements.
- Sql first evaluates the inner query (or) subquery

In nested query, inner query runs first, & only once outer query is executed with result from inner query.

Hence inner query is used in execution of outer query.

Eg :- select empname, highest salary

Sol :- select empname from emp where

Salary = (select max (salary) from emp);

eg2 :- Second highest salary

Select ename, deptno from emp

where sal = (select max (salary) from emp

where sal < (select max (sal) from emp));

Difference b/w Relational algebra & calculus:-

1) language type	- procedural	Non-procedural
2) procedure	- How to obtain the result	what result we have to obtain
3) Order	- The order is specified in which the operation have to perfer	The order is not specified
4) Domain	- Independent of the domain	Dependent because of domain relational calculus.