08.01.20 * Unit-2 *

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Fig 5.1. An Instance of S3 of Sailors.

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/08/98
22	104	10/07/98
31	102	11/10/98
31	103	11/06/98
31	104	11/12/98
64	101	09/05/98
64	102	09/08/98
74	103	09/08/98

Fig 5.2. An Instance of R2 of Reserves

bid	bname	colour
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red.

Fig. An Instance B1 of Boats.

Part-1

* Relationa
a procedi
a relati
a relat
Relation
Lang. it m
be Retriev
consists
one or t
a new r

Basic O

Unary
Operator
↳ Selectio
↳ Project
↳ Renam

→ Bankl
1. Br
2. Cus
3. Acc
4. Lo
5. Dep
6. Bo

Part-1

* **Relational Algebra:** Relational Algebra is a procedural Query Language which takes a relation as an input and generates a relation as an Output.

Relational Algebra is procedural Query Lang. it means that it tells what data to be Retrieved & how to be Retrieved. It consists of a set of Operations that takes one or two relations as input and produce a new relation as the output.

Relational Algebra Operators

- Basic Operators
  - Unary Operators
    - Selection (σ)
    - Projection (π)
    - Rename (ρ)
  - Binary Operators
    - Union (∪)
    - Cross product (×)
    - Minus/set Difference (−)
- Extended/Derived Operators
  - Join Operator (⋈)
  - Division (/ or ÷)
  - Intersection (∩)

→ **Bank DataBase:**

1. Branch (bname, bcity, assets)
2. Customer (cname, cstreet, ccity)
3. Account (~~aname, b~~ accno., bname, bal.)
4. Loan (loanno., bname, amount)
5. Depositor (cname, acc.no.)
6. Borrower (cname, loan no.)

1) Selection Operator (σ): The selection operator selects tuples from a relation that satisfy a given predicate.

Notation:

$$\sigma_{\text{selectionpredicate}}(\text{Relation})$$

Ex:

$\sigma_{\text{branch\_name} = \text{"Hyd"}}(\text{Branch})$

$\sigma_{\text{Salary} > 10000}(\text{Employee})$

The selection predicate can use $=, \neq, <, >, \leq, \geq$ operators for comparisions

Two or more predicates can be combined by using the connectives. $\wedge$ (and), $\vee$ (OR), $\neg$ (NOT)

2) Projection Operator (π): The projection produce a relation with the attributes mentioned in the attribute list.

Duplicate rows removed from result since relations are sets.

Notation:

$$\pi_{\text{attribute\_list}}(\text{Relation})$$

Eg:

$\pi_{\text{loan\_no, amount}}(\text{loan})$

$\pi_{\text{Emp no, Salary}}(\text{Employee})$

3) Rename (or) Renaming Operator (ρ):

Change the old name to new name of the relation or fields.

The Rename Operation allows us to name and therefore to refer to the results of relational algebra expressions.

It allows us to refer to a relation by more than one name.

Eg: $\rho_x(E)$

→ Returns the expression 'E' under the name 'x'.

If a Relational algebra expression 'E' has Arity 'n' then,

$$\rho_x(A_1, A_2, A_3 \ldots A_n)(E)$$

→ Returns the result of expression 'E' under the name 'X' & with the Attributes renamed to $A_1, A_2 \ldots A_n$.

* Set Operations (Union, Intersection, Set difference, cross product)

4) Union: The Union Operation Combines tuples of two relations to form a new relation by removing duplicate rows & returns the relation.

Notation: $R \cup S$ → which is defined as

$$R \cup S = \{t / t \in R \text{ or } t \in S\}$$

→ for $R \cup S$ to be valid R & S must be Compatable i.e

i) R, S must have same arity i.e same no. of attributes.

ii) The domain of i^{th} Attribute of r and the i^{th} attribute of s must be same for all i.

Ex: To find all customers with either an account or a loan

$\Pi_{customer\_name}$ (depositor) $\cup$ $\Pi_{customer\_name}$ (borrower)

5) Intersection ($\cap$) Operator:

The Intersection operation ~~refo~~ results the common tuples of two relations to form a new relation.

Notation: $R \cap S$

which is defined as

$R \cap S = \{t \mid t \in r \text{ and } t \in s\}$

for $R \cap S$ to be valid,

i) R and S must be compatable i.e

i) R, S must have the same Arity. that is same no. of Attributes.

ii) The domain of i^{th} Attribute of r and i^{th} attribute of s must be same for all i.

Ex: To find all customer who have a loan and an account.

$\Pi_{customer\_name}$ (depositor) $\cap$ $\Pi_{customer\_name}$ (borrower)

Note: $r \cap s = r - (r - s)$ is also a intersection of two table;

6) Set Differe

The set di

tuples that a

another rela

Notation:

R-S w

R-

for R-S t

compatible.

i) R, S must

ii) The dom

attribute

Eg: To fin

have an

$\Pi_{custome}$

7) Cross pr

The C

is to co

relatio

Notatio

R(A1, A2

Assu

disjoi

If

disjo

6) Set Difference (−):

The set difference operation finds the tuples that are in one relation but not in another relation.

Notation:

R-S which is defined as,

$$R-S = \{t \mid t \in r \text{ or } t \notin S\}$$

for R-S to be valid, R and S must be compatible.

i) R, S must have the same Arity.

ii) The domain of ith Attribute of R and ith attribute of S must be same for all i.

Eg: To find all customers of the bank who have an account but not a loan.

$$\Pi_{\text{customer\_name}}(\text{depositor}) - \Pi_{\text{customer\_name}}(\text{borrower})$$

7) Cross product (Cartesian Product) (X)

The Cartesian Product Operation allows us to combine information from any two relations

Notation: $r \times s = \{tq \mid t \in R \text{ and } q \in S\}$

$$R(A_1, A_2, \ldots A_n) \times S(B_1, B_2, \ldots B_m) = Q(A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_m)$$

Assume that attributes of R and S are disjoint i.e $R \cap S = \emptyset$.

If attributes of R and S are not disjoint then renaming must be used.

8) Join: The join operation is one of the most useful operations in relational algebra if it is used to combine the information of two or more relations which have atleast one field in common.

-> It is defined as a cartesian product followed by selection or projection operator.

NOTE: The cross product result is very large to minimize the result apply selections & projections on the cross product, is nothing but join.

$\Rightarrow R \bowtie_c S = \sigma_c(R \times S)$

i) Condition joins: returns a relation that includes a set of tuples from cross product of two relations A & B such that each tuple satisfies a condition.

Common field has to be present in both tables i.e R and S.

First the cross product of R and S is formed then the condition is checked and minimize the result.

The final result is the join condition.

ii) Equijoin ($\bowtie_\theta$): $R \bowtie_c S$

R join S in equijoin if 'c' contains same values in that attribute. The join operation with this refinement is called equijoin.

In equijoin duplicate columns are dropped.

iii) Natural join: The equijoin expression is actually a natural join denoted as 'R ⋈ S' if no attributes with same name then it will be simply the cross product.

$$R \bowtie S = R \times S.$$

g) Division (/ or ÷): The division operator is used for queries that include the phase "for all".

→ If A & B are two relations & C is equal to A/B then for each tuple in C its product with the tuples of B must be in A.

formal definition: let r(R) & s(S) be two relations and S⊆R i.e every attribute of schema S is also in schema R.

→ The relation r/s is a relation on schema R-S. A tuple t is in r/s if and only if both of two conditions hold.

condition 1:- t is in $\pi_{R-S}(r)$

condition 2: for every tuple t_s in S there is a tuple t_r in R satisfying both of the following.

a) $t_r[S] = t_s[S]$

b) $t_r[R-S] = t$

* Algebra Expression:

A/B as $\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$

Sailors (S1)

sid	sname	rating	age
22	Dustin	7	45
31	Lubber	8	55
58	Rusty	10	35

Sailors (S2)

sid	sname	rating	age
28	Yuppy	9	35
31	Lubber	8	55
44	Rubby	5	35
58	Rusty	10	35

S1 U S2

sid	sname	rating	age
22	Dustin	7	45
31	Luber	8	55
58	Rusty	10	35
28	Yuppy	9	35
44	Rubby	5	35

Reservation (R1)

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

S1 ∩ S2

sid	sname	rating	age
31	Luber	8	55
58	Rusty	5	35

S1 - S2

sid	sname	rating	age
22	Dustin	7	45

sid	sname
22	Dustin
22	Dustin
31	Luber
31	Luber
58	Rusty
58	Rusty

i) Condition

R

e

Output:

(sid)	sn
22	
31	

ii) Equi

Outpu

Si
22
58

$S_1 \times R_1$

sid	sname	rating	age	sid	bid	day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Luber	8	55	22	101	10/10/96
31	Luber	8	55	58	103	11/12/96
58	Rusty	10	35	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96

i) Condition joins:

$R \bowtie_c S = \sigma_c (R \times S)$

eg: $S_1 \bowtie R_1$

~~S1.sid < R1.sid~~

S1.sid < R1.sid

Output:

< > ≤ ≥

(sid)	sname	rating	age	(sid)	bid	day
22	D	7	45	58	103	11/12/96
31	L	8	55	58	103	11/12/96

ii) Equijoin: Syntax: $S \bowtie_c R$

$S_1 \bowtie_{S1.sid = R1.sid} R_1$

Output:

sid	sname	rating	age	bid	day
22	D	7	45	101	10/10/96
58	R	10	55	103	11/12/96

(iii) Natural join:

$R \bowtie S$

Eg: $R1 \bowtie S1$

Output:

sid	sname	rating	age	bid	day
22	D	7	45	101	10/10/96
58	R	10	55	103	11/12/96

* Division:

Ex:

P

A	B
a1	b1
a1	b2
a2	b1
a3	b1
a4	b2
a5	b1
a5	b2

Q

B
b1
b2

R

B
b1

S

B
b1
b2
b3

$P \div Q =$

A
a1
a5

$P/R =$

A
a1
a2
a3
a5

$P/S =$ Null

Ex:

A

Sno	Pno
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

B_1

Pno
P_2

B_2

Pno
P_2
P_4

B_3

Pno
P_1
P_2
P_4

⇒ A/B_1 =

Sno
S_1
S_2
S_3
S_4

⇒ A/B_2 =

Sno
S_1
S_4

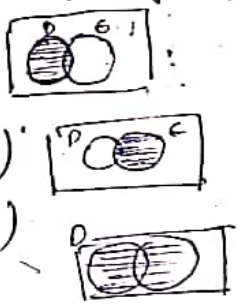
⇒ A/B_3 =

Sno
S_1

Joins

- **Inner Join**
  - Outcome or result contains only the matching tuple
    - Condition join
    - Equi join
    - Natural join
- **Outer Join**
  - The Result will contain all the tuples from one or both of the relation.
    - Left Outer join
    - Right Outer join
    - full Outer join

1) Left Outer join ($\bowtie_\theta$).
2) Right Outer join ($\bowtie_\theta$)
3) full Outer join ($\bowtie_\theta$)



Eg:

Department (D)

Did	Dept
1	CSE
2	ECE
3	IT

Emp (E)

Eid	name	Dept
E_1	X	CSE
E_2	Y	IT
E_3	Z	EEE

$\bowtie_\theta \Rightarrow$

Did	Dept	Eid	name
1	CSE	E_1	X
3	IT	E_2	Y
2	EEE	NULL	NULL

$\bowtie_\theta \Rightarrow$

Did
1
3
Nu

$\bowtie_\theta \Rightarrow$

~~Rename Eg:~~

* Outer joins
Operation th
Outer joins a
which are
does not sat

Ex: Selec
from
where

Output:

Did
1
3

This to
of two ta
that sati
+howeve
rows that

$\bowtie_\theta$ =>

Did	Dept	Eid	name
1	CSE	E1	X
3	IT	E2	Y
NULL	EEE	E3	Z

$\bowtie_\theta$ =>

Did	Dept	Eid	name
1	CSE	E1	X
3	IT	E2	Y
2	ECE	NULL	NULL
NULL	EEE	E3	Z

~~Example~~ Eg: c(

* Outer joins are the spl. variant of join Operation that are dependent on null values. Outer joins allows us to select those rows which are common and even those rows that does not satisfy given condition.

Ex! Select *.D, *.E Eg: (Normal Query)
from Department D, Employee E
where D.Dept = E.Dept ;

Output:

Did	Dept	Eid	name
1	CSE	E1	X
3	IT	E2	Y

This table show simple join operation of two tables only those rows are selected that satisfied the condition.

However, if we want to include those rows that doesn't satisfy the cond., then use outer joins

9. Left Outer Join:

List all those rows which are common to both the tables along with the unmatched rows of 2nd table.

SQL Query:

```
Select *.D, *.E
from Department D LEFT OUTER JOIN Employee E
where D.Dept = E.Dept;
```

Output:

Did	Dept	Eid	name
1	ESE	E1	X
3	IT	E2	Y
2	ECE	NULL	NULL

ii) Right Outer Join:

List all those rows which are common to both the tables along with the unmatched rows of 1st table

SQL Query:

```
select *.D, *.E
from Department D RIGHT OUTER JOIN Employee E
where D.Dept = E.Dept.
```

O/P:

Did	Dept	Eid	name
1	CSE	E1	X
3	IT	E2	Y
NULL	EEE	E3	Z

iii) Full Outer Join:

Unmatched rows of both tables are listed along with the common rows of the tables. i.e Comb. of left Outer Join & right Outer Join.

SQL Query:

```
Select *.D, *.E
from Department D FULL OUTER JOIN Employee E
  where D.Dept = E.Dept;
```

O/P:

Did	Dept	Eid	name
1	CSE	E1	X
3	IT	E2	Y
2	ECE	NULL	NULL
NULL	EEE	E3	Z

* Examples of algebra Queries:

Q1. List all sailors whose rating is greater than 8 ?

$$\sigma_{rating > 8}(sailors)$$

Q2. Select the name and rating of a sailor whose rating is >8 ?

$$\pi_{sname, rating}(\sigma_{rating > 8}(sailors))$$

Q3. Find the name of sailor, who ~~or~~ have reserved boat ~~for~~ 103?

$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$

Q4. Find the names of sailors who have reserved a red boat?

A: $\pi_{sname}(\sigma_{colour='red'}(Sailors \bowtie Reserves \bowtie Boats))$.

(or)

$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{colour='red'}(Boats) \bowtie Reserves) \bowtie Sailors))$

Q5. Find the colours of boats reserved by Lubber?

A: $\pi_{colour}(\sigma_{sname='Lubber'}(Sailors \bowtie Reserves \bowtie Boats))$

Q6. Find the name of the sailors whose rating = 8 and age > 30

A: $\pi_{sname}(\sigma_{rating=8 \wedge age>30}(Sailors))$

Q7: Find the names of sailors who have reserved atleast one?

$\Rightarrow \pi_{sname}(Sailors \bowtie Reserves)$

Q8) Find the reserved

$\pi_{sname}(\sigma_{col}$

Rename:

→ ρ(Tempbo

→ π_{sname}(Tem

Q9. Find th reserved

A: ρ(Tempbo

π_{sname}(T

Using Uni

ρ(Tem

ρ(Tem

π_{snam}

Q8) find the names of sailors who have reserved red (or) green boat.

$\pi_{sname}(\sigma_{colour='red' \vee 'green'}\ Boats) \bowtie Reserves \bowtie Sailors)$

(or)

Rename:

→ $\rho(Tempboats, (\sigma_{colour='red' \vee 'green'}\ Boats))$

→ $\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$

Q9. Find the name of sailors who have reserved red and green boat.

A: $\rho(Tempboats, ((\sigma_{colour='red'}\ Boats) \cap (\sigma_{color='green'}\ Boats)))$

$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailor)$

Using Union @8:

$\rho(Tempred, (\sigma_{colour='red'}\ Boats) \bowtie Reserves \bowtie Sailors))$

$\rho(Tempgreen, (\sigma_{colour='green'}\ Boats) \bowtie Reserves \bowtie Sailors))$

$\pi_{sname}(Tempred \cup Tempgreen)$

* Relational Calculus:

Relational Calculus is non procedural or declarative language where It focus on what to retrieve but not on how to retrieve It express the query using variable and the formulaes.

Same expressive power as relational algebra.

Relational calculus is of two types
i) Tuple Relational Calculus (TRC)
ii) Domain Relational Calculus (DRC)

Note: TRC influenced by SQL; DRC is influenced by QBE (Query by Example).

i) Tuple Relational Calculus (TRC): Variables in TRC take on tuples as the values

Tuple variable (t) ranges from

Ex:

Q1. Find all sailo

→ {S/S∈ sa

Q2. Find the na
a rating ab

{P/∃S∈ sa
S.s

Q3. Find the
reservation

{P/∃S ∈ sa
∧ P.s
∧ P.

Q4. Find th
boat 103'.

{P/∃S ∈

Q5. Find
'red' bo

→ {P/∃S

Ex:

Q1. Find all sailors with the ratings above 7

→ $\{S/S \in \text{sailors} \wedge S.\text{rating} > 7\}$

Q2. Find the names and ages of sailor with a rating above 7.

$\{P/\exists S \in \text{Sailors}(S.\text{rating} > 7 \wedge P.\text{sname} = S.\text{sname} \wedge P.\text{age} = S.\text{age})\}$

Q3. Find the sailor name, boat id and reservation date for each reservation.

$\{P/\exists S \in \text{sailors} \exists R \in \text{Reserves}(S.\text{sid} = R.\text{sid} \wedge P.\text{sname} = S.\text{sname} \wedge P.\text{day} = R.\text{day} \wedge P.\text{bid} = R.\text{bid})\}$

Q4. Find the name of sailors who reserved boat 103.

$\{P/\exists S \in \text{sailors} \exists R \in \text{Reserves}(S.\text{sid} = R.\text{sid} \wedge R.\text{bid} = 103 \wedge P.\text{sname} = S.\text{sname})\}$

Q5. Find the name of sailors who reserved 'red' boat

→ $\{P/\exists S \in \text{sailors} \exists R \in \text{Reserves} \exists B \in \text{Boats}(S.\text{sid} = R.\text{sid} \wedge R.\text{bid} = B.\text{bid} \wedge B.\text{color} = \text{'red'} \wedge P.\text{sname} = S.\text{sname})\}$

Q6. Find the name of sailors who have reserved atleast two boats.

A: $\{P / \exists S \in Sailors \exists R1 \in Reserves \exists R2 \in Reserves$
$(S.sid = R.sid \wedge R1.sid = R2.sid \wedge$
$R1.bid \neq R2.bid \wedge P.sname = S.sname)\}$

Q7. find the name of sailors who have reserved all boats.

A) $\{P / \exists S \in Sailors, \forall B \in Boats (\exists R \in Reserves$
$(S.sid = R.sid \wedge R.bid = B.bid \wedge$
$P.sname = S.sname))\}$

Q8. find the sailors who have reserved all red boats:

A) $\{S / \exists S \in Sailors \wedge \forall B \in Boats (\exists R \in Reserves$
$(S.sid = R.sid \wedge R.bid = B.bid \wedge B.color = 'red'$
$\wedge$ ~~P.sname = S.sname~~))}

(OR)

$\{S / \exists S \in Sailors \wedge \forall B \in Boats (B.colour = 'red' \Rightarrow$
$(\exists R \in Reserves (S.sid = R.sid \wedge R.bid =$
$B.bid \wedge$ ~~P.sname = S.sname~~))}

$P \Rightarrow q \equiv \sim p \vee q$

(Or)

$\{S / \exists S \in Sailors \wedge \forall B \in Boats (B.colour = 'red'$
$\Rightarrow (\exists R \in Reserves (R.sid = S.sid \wedge R.bid = B.bid \wedge$
$P.sname = S.sname))\}$

P	q	P
T	T	
T	F	
F	T	
F	F	

* Domain Re

→ Dom

Ranges over some Attribu

{< x_1, x_2,

↳ domai

syntax of

$P \Rightarrow q = \sim P \cup q$

P	q	$P \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

$\sim P$	q	$\sim P \cup q$
F	T	T
F	F	F
T	T	T
T	F	T

* Domain Relational Calculus: (DRC)

→ Domain Variable is a Variable that Ranges over the values in the domain of some Attributes

$\{< x_1, x_2, \ldots, x_n > / P(< x_1, x_2, \ldots, x_n >)\}$

↳ domain variable ↑ Relation Variable

Syntax of DRC Queries:

Examples:

Q1: Find all sailors with a Rating above 7?

$\{ \langle I, N, T, A \rangle / \langle I, N, T, A \rangle \in \text{Sailors} \wedge T > 7 \}$

Q2: Find the Name of sailor who have reserved boat '103'.

$\{ \langle N \rangle / \exists I, T, A (\langle I, N, T, A \rangle \in \text{sailors} \wedge \exists D (\langle I, 103, D \rangle \in \text{Reserves})) \}$

(or)

$\{ \langle N \rangle / \exists I, T, A (\langle I, N, T, A \rangle \in \text{Sailors} \wedge \exists \langle Ir, Br, D \rangle \in \text{Reserves} (Ir = I \wedge Br = 103)) \}$

Q3: Find the Name & Ages of sailors with a rating above 7?

$\{ \langle N, A \rangle / \exists I, T (\langle I, N, T, A \rangle \in \text{Sailors} \wedge T > 7) \}$

Q4: find the sailor Name, Boat Id and reservation date for each ~~sai~~ Reservation;

$\{ \langle N, Br, D \rangle / \exists I, T, A (\langle I, N, T, A \rangle \in \text{Sailors} \wedge \exists (\langle I, Br, D \rangle \in \text{Reservers})) \}$

Q5: find the Name of sailors who have reserved a 'red' boat.

$\{ \langle N \rangle / \exists I, T, A (\langle I, N, T, A \rangle \in \text{sailors} \wedge \exists (\langle I, Br, D \rangle \in \text{Reserves} \wedge \exists (\langle Br, Bn, \text{'red'} \rangle \in \text{Boats})) \}$

Q6: find Name o
2 boats

Q7: find Nam
all boats

Q8: find the
red boat

Q6: Find Name of sailors who have reserved atleast 2 boats

A: $\{\langle N \rangle / \exists I, T, A\ (\langle I, N, T, A \rangle \in \text{Sailors} \wedge \langle Ir1, Br1, D \rangle \in \text{Reserves} \wedge \langle Ir2, Br2, D \rangle \in \text{Reserves} \wedge I = Ir1 \wedge I = Ir2 \wedge Br1 \neq Br2)\}$

Q7: Find Names of Sailors who have reserved all boats:

A: $\{\langle N \rangle / \exists I, T, A\ \langle I, N, T, A \rangle \in \text{Sailors} \wedge \forall \langle B, Bn, C \rangle \in \text{Boats} \wedge (\exists Ir, Br, D \langle Ir, Br, D \rangle \in \text{Reserves} \wedge I = Ir \wedge Br = B)\}$

Q8: Find the Sailors who have reserved all red boats:

A: $\{\langle I, N, T, A \rangle / \langle I, N, T, A \rangle \in \text{Sailors} \wedge \forall \langle B, Bn, C \rangle \in \text{Boats} \wedge (\exists Ir, Br, D \langle I, Br, D \rangle \in \text{Reserves} \wedge B = Br \wedge C = \text{'red'})\}$

Sailors (Sid, Sname, rating, age)
I N T A

Reserves (Sid, Bid, day)
I Br D

Boats (Bid, Bname, color)
Br Bn C

* Expressive power of Algebra and Calculus:

-> A Query Language is said to be relationally complete if it is atleast as powerful as RA which is logically equivalent to RC.

-> All the practical Query lang's are relationally complete as the logical power of these lang is more than the power of RA.

-> There exist certain Queries that cannot be expressed in RA.

Eg: Consider the following query which is relationally complete as it can be formulated in both SQL as well as RA. find the names, loan no. & and loanamt. for all customer who have a loan from the bank.

SQL: Select c_name, b.l_no, amt
from borrower b, loan l
where b.l_no = l_no;

RA: $\Pi_{c.name, l\_no, amt}$ (borrower ⋈ loan)

-> Unsafe Queries are the Query that are syntactically correct but results in infinite set of tuples. These queries are different from safe Queries where in the resulting set consist of limited tuples. It is often desirable to disallow the execution of such queries bcoz of the following reasons,

↳ More CPU time is consumed in processing the Query.

↳ Searching the desired record from the resultant set is tedious.

Eg: $\{S/J(S \in Sailors)\}$ is an Unsafe Query since this Query will result in the set of

all sailors that doesnot belong to the sailor relation.

-> Both the Lang's RA & RC are logically Equivalent that is all the Queries which can be expressed in RA can also be expressed in RC, But the reverse may not be true.

{S/T [S ∈ sailors)} -> is unsafe Query & it results in an infinite set.

-> A Query is said to be safe if it results in the set with limited tuples.

-> Any Calculus formula is said to safe if it is one of the following,

1) The resulting set R contains the values which are in the domain of Q and S, where Q is a Query & S is set of relational instances.

R ← Domain(Q,S) ⇒ for any S, R should consist of the values present in Domain(Q, S)

2) The Sub formula, ∃x(P(x)) is true for any value 'a', then this value 'a' contains only constants present in the domain.

3) The Sub formula, ∀x(P(x)) is true for a value 'a' then this value 'a' contains Constraints i.e not present in the domains.

But these definition are not Enough to check whether a given Query is safe or not.

28.01.20

* SQL Queries:

- Form of a Basic SQL Query is as follows

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification;
```

* Syntax of a basic SQL Query in Detail:-

i) The from-list in the FROM clause is a list of table names.

ii) The select-list is a list of column names of tables named in the from-list.

iii) The qualification in the where clause is a boolean combination (i.e. an expression using the logical connectives AND, OR & NOT) of conditions of the form expression

expr op expr where OP is one of the comparison operators.

$\{<, \leq, =, >, \geq, \neq\}$

An Expression is a column name a constant or an expression (arthmetic or string)

iv) The DISTINCT keyword is optional, it indicates that the result of Query should not contain duplicates. The default is that duplicates are not eliminated.

RA:

* preceding rules describe the syntax of a basic SQL Query:

i) Compute the cross product of the tables in the from list.

ii) Delete rows in the cross product that fail the qualification conditions.

iii) Delete all columns that do not appear in the select list.

iv) If DISTINCT is specified, eliminate duplicate rows.

Joins: ~~R×S~~ ~~R×S×B~~ S ⋈ R ⋈ B

sid	sname	rating	age	bid	day	bname	colour
22	D	7	45	101	10/10/98	Inter	blue
22	D	7	45	102	10/10/98	Inter	red
22	D	7	45	103	10/8/98	Cli	green
22	D	7	45	104	10.7.98	Marine	red
31	h	8	55.5	102	11.10.98	Inter	blue
31	h	8	55.5	103	11.06.98	Clippa	green
31	h	8	55.5	104	11.12.98	Marine	red
64	H	7	35.0	101	9.5.98	Inter	blue
64	H	7	35.0	102	9.08.98	Inter	red
64	H	9	35.0	103	09.08.98	Clipper	green

SQL Queries:

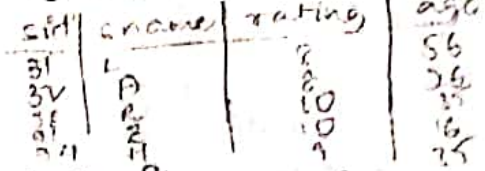
1) Find the names & Ages of all the Sailors?

A) Select sname, age
FROM sailors ;

sname	age
Dustin	45
B	33
L	56
D	36
R	35
H	35
Z	16
H	35
A	26
D	64

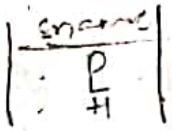
2) Find all the sailors with a rating above 7.

A: select * from sailors where rating > 7;



3) Find the names of sailors who have reserved boat no. 103?

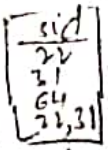
A: Select s.sname from sailors s, Reserves R where s.sid = R.sid and R.bid = 103



4)

4) Find sid of sailor, who have reserved a red boat.

A: Select R.sid from ~~sailors~~, Reserves R, Boat b where ~~s.sid = R.sid and~~ R.bid = b.bid and b.colour = 'red';

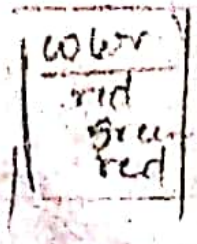


5) Find name of Sailor who have reserved a red boat.

A: select ~~s.sid~~ S.sname from Sailors S, Reserves R, Boat b where S.sid = R.sid and R.bid = b.bid and b.colour = 'red';

6) find the colours of boat reserved by hubber?

A: Selectbcolor from Sailors S, Reserves R, Boat b where S.sid = R.sid and R.bid = b.bid and S.sname = 'hubber';



7) Find the reserved

A: Select

Reserve

~~R.bid~~

8) Comput persons on the s

A: Select from sai S.sid = R R2.bid

9) Find t begins atleas

A: Select like '

10) Find reser

A: Selec Boat and

(Selec Boats B B.col Selec where B.col

7) Find the names of sailors who have reserved atleast one boat;

A: select S.sname from sailor S, Reservers R where S.sid = R.sid;

sname
D (4)
( )
( )

8) Compute Increments for the ratings of persons who sailed two different boats on the same day?

A: Select S.sname, S.rating +1 as rating from sailors S, Reservers R_1, Reservers R_2 where S.sid = R_1.sid and S.sid = R_2.sid and R_1.bid <> R_2.bid and R_1.day = R_2.day;

sname	rating
D	8
D	8

9) find the ages of sailors whose name begins and ends with b and has atleast 3 characters.

A: select age from sailors where sname like 'B_%B';

age
64

10) find the name of sailors who have reserved a green or red Boat.

A) Select S.sname from Sailor S, Reservers R, Boats B. where S.sid = R.sid and R.bid = B.bid and b.color = 'green' or b.color = 'red';

sname
D
E
H

(Select S.sname from sailors S, Reserver R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'red') UNION

(select S.sname from sailor S, Reserver R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'green');

11) Find the name of sailors who have reserved both red & green.

A) Select z.sname from sailors S, Reserves R, Boats B where S.sid = R.bid and R.bid = B.bid and b.colour = red and green b.color = 'green';

B) Select s.sname from sailors S, Reserves R1, Reserves R2, Boats B1, Boats B2 where S.sid = R1.sid and R.bid = B.bid and

A: Select S.sname from sailor S, Reserver R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'red'

INTERSECT

Select S1.sname from sailors S1, Reservers R1, Boats B1 where S1.sid = R1.sid and R1.bid = B1.bid and B1.color = 'green');

Sname
D
J1

12) Name of sailor green boat but not a red boat.

A: Select S.sname from sailors S, Reserve R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'green'

Except

Select S.sname from sailors S, Reserver R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'red';

* Union, Intersect, Except (Set Operation)

Union, Intersect, Except by default, Eliminate duplicate values present in the resultant Output.

for a basic Query distinct is specified to Eliminate duplicate values in the result Output.

TO retain duplicates in set Operation Queries, use union all, Intersect all and Except all.

13) find sid of all sailors who have reserved red boats but not green boats.

A: Select S.sid from sailors S, Reservers R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color = 'red' Except select S.sid from sailors S, Reserver R, Boats B where S.sid = R.sid and R.bid = B.bid and B.color, 'green'; (or) ↓

14) find all sid of sailors who have a rating of '10' or reserved boat '104'.

A) Select S.sid from sailors S, ~~Reservers~~ where S.rating = 10 union ALL (Select R.sid from ~~sailors~~, reserver R where ~~S.sid = R.sid~~ R.bid = '104');

sid
58
31
22
31
64

15) Select R.sid from Reservers R, Boats B
A: where R.bid = B.bid and B.color = 'red' Except Select R.sid from Reservers R, Boats B where R.bid = B.bid and B.color = 'green';

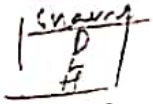
Nested Queries (IN, NOT IN)

One of the most powerful features of SQL is nested Query. A nested Query is a Query that has another Query embedded within it. The embedded is called a Sub Query.

A Sub Query Appears within the where clause of a Query. Sub Queries can also appear in the from clause or the having clause.

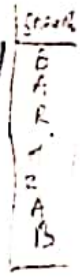
1Q) Find the Names of sailors who have reserved boat 103?

→ Select s.sname from sailors s where s.sid IN (select R.sid from Reserviers R where R.bid=103);



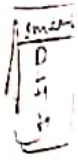
2Q) Find the Name of sailors who have not reserved boat 103?

→ Select S.sname from sailors s where s.sid NOT IN (select R.sid from Reservers R where R.bid=103);



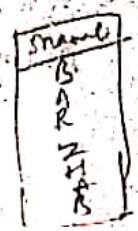
3Q) Find the Name of sailors who have reserved a red boat?

→ Select s.sname from sailors s where s.sid IN (Select R.sid from Reservers R where R.bid IN (Select B.bid from Boats B where B.colour= 'red'));



4Q) Find the name of sailors who have not reserved 'red' boat.

→ Select S.Sname from sailors s where s.sid NOT IN (select R.sid from Reserves R where R.bid IN (Select B.bid from Boats B where B.colour= 'red'));



Note: The 'IN' operator allows us to test whether a value is in a given set of elements. An SQL Query is used to generate the set to be tested.

Note: IN Nested Queries the inner Sub Query has been competely dependend on the Outer Query

Ex) find the Name of Sailors who have reserved both red and Green boat

→ ~~Select S.Sname from sailors S where S.sid IN~~ (Select R.sid ~~from~~ Reserves R where R.bid IN (Select B.bid from Boats B where B.colour = 'red', B.colour = 'green'

→ Select S.Sname from sailors S, reserves R, Boats B where S.sid = R.sid and R.bid = B.bid and B.colour = 'red' and S.sid IN (Select R1.sid from Reserves R1, Boats B1 where R1.bid = B1.bid and B1.colour = 'green');

* Correlated Nested Queries (Exists, NOT Exists, Unique, NOT Unique):

EXIST Operator is another set comparision Operator such as IN, The Sub Query clearly depends on the current row 'S' and must be reevaluated for each row in sailors. The occurence of 'S' in the sub Query is called a correlation and such Queries are called Correlated Nested Queries:

1Q) find the Name of Sailor who have reserved boat no. 103?

A) Select S.sname from sailors S where EXISTS (select * from Reserves R where R.bid=103 and R.sid=S.sid);

2Q) find the Name of Sailor who have not reserved boat no. 103?

A: Select S.sname from sailors S where NOT EXISTS (select * from Reservers R where R.bid=103 and R.sid=S.sid);

3Q) find the Name of sailors who have reserved a red boat?

A) Select S.sname from sailors S where EXISTS(select * from reservers R where EXISTS(select * from Boats B where B.colour='red' and B.bid=R.bid and R.sid=S.sid));

4Q) Not reserved red boat.

A) Select S.sname
from sailors S
where NOT EXISTS
(select *
from reservers R
where EXIST
(select *
from Boats B
where B.colour='red'
and B.bid=R.bid and
R.sid=S.sid));

Scanned by CamScanner

Scanned with CamScanner

Set-Comparision Operators: (ANY, ALL)

SQL also supports OP ANY & OP ALL where OP is one of the arithmetic comparision Operator {<, <=, >=, =, <>, >}

1Q) Find sid of sailors whose rating is better than some sailor called Horatio?

```
select s.sid
from sailors S
where s.rating > ANY (select s1.rating from
                      sailors S1 where
                      S1.sname = 'Horatio');
```

2Q) Find sid of sailor whose rating is better than Every sailor?

```
select s.sid
from sailors S
where s.rating > ALL (select s1.rating from
                      sailors S1 where
                      S1.name = 'Horatio');
```

3Q) Find sid of sailors with the highest rating?

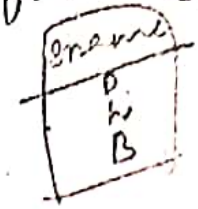
```
select s.sid
from sailors S
where s.rating >= ALL (select s1.rating
                       from sailors S1);
```

sid
58
71

Note: IN and NOTIN are Equivalent to =ANY and <>ALL respectively

4(a) Find the name of sailors who are older than the oldest sailor with the rating of 10?

A: Select S.sname from sailors S where S.age > ALL (Select S1.age from sailors S1 where S1.rating = 10);



* Aggregate Operators:

SQL supports five Aggregate Operations which can be applied on any column of a relation.

1. count ([DISTINCT] colname)
   The no. of (UNIQUE) values in the column.
2. SUM: The sum of all (UNIQUE) values in the column. [: SUM ([DISTINCT
3. AVG ([DISTINCT] colname):
   The average of all (UNIQUE) values in the particular column
4. MAX (colname): The Maximum value in the particular column.
5. MIN (colname): The Minimum value in the particular column

(a) Find the AVG ages of all the sailors:

```
Select AVG(S.age)
from sailors S;
```

AVG(S.age)
37.1000

2Q) Find the avg. age of sailors with a rating of 10?

```
select AVG(s.age)
from sailors S
where rating = 10;
```

3Q) find the youngest age of all sailors?

```
select MIN(S.age)
from sailors S;
```

4Q) find the name and age of the oldest sailor?

```
select s.sname, s.age from sailors s where s.age
=/IN (select max(s.age) from sailors s1);
```

5Q) count the no. of sailors:

```
select count(*)
from sailors S;
```

6Q) Count the no. of different sailor names?

```
select count(DISTINCT(S.sname))
from sailors S;
```

7Q) find the name of sailors who are older than the oldest sailor with the rating of 10?

```
select S.sname
from sailors S
where S.age > (select Max(S1.age)
               from sailors S1
               where S1.rating = 10);
```

* The GROUPBY and Having clauses:

The GROUPBY statements group rows that have the same values into summary rows. The GROUPBY statement is often used with aggregate functions i.e COUNT, SUM, AVG, MAX, MIN.

To group the result set by one or more column.

GROUPBY clause is used with the select statement. In the Query GROUPBY clause is placed after the where clause.

In the Query, GROUPBY clause is placed before ORDER BY clause if used any.

GROUPBY means summarizing data from the database. The Query that contain the GROUPBY clause are called Grouped Queries and only return a single row for every grouped item.

Syntax:

```
1) from-list
2) where
3) selection list
4) GROUPBY
5) Having
6) Order By
7) o/p
```

```
select select_list [column_name(s)]
from table_name(s) [Table_name(s)]
where qualification [cond^n]
GROUP BY grouping_list [column_name]
Having group_qualification [cond^n on group by result]
ORDER BY column_name
```

HOVING clause:

Where clause is used to place conditions on column but it can't be use to place conditions on group

** Having clause is used to place conditions to decide which group will be the part of final result set and we cannot use the aggregate functions like SUM, COUNT, MAX, MIN, AVG with

Note:
where clause, but it can be done in Having clause.

(a) Find the age of the youngest sailors for each rating level?

```
select MIN(s.age), s.rating
from Sailors s
groupby s.rating;
```

(b) Find the age of youngest sailor who is eligible to vote (i.e atleast 18 yrs old) for each rating level with atleast two such sailors.

```
select min(s.age), s.rating
from sailors s
where s.age >= 18
groupBy s.rating
Having count(*) > 1;
```

(c) Find the AVG age of sailors for each rating level that has atleast 2 sailors?

```
Select AVG(s.age), s.rating
from sailors s
group by s.rating
Having count(*) > 1;
```

* Triggers and Active Db:

In Dbms, a trigger is a stored procedure or SQL procedure that initiates an action when an event (Insert, Update or delete) occurs. Triggers are event driven specialised procedure that are stored in and managed by dbms a trigger cannot be called or executed, The dbms automatically

weak entity set always appears in
associate...

fires the trigger as a result of a data modification to the associated table. Each trigger is attached to a single specific table in the database.

Stored procedures are not Event driven and are not attached to a specific table.

Stored procedures are explicitly executed by invoking a call to the procedure while triggers are implicitly executed.

→ In addition triggers can also execute stored procedures.

* Triggers and Active Databases:

→ In a DBMS, a trigger is a SQL procedure initiates an action when an event (INSERT, or UPDATE) occurs.

→ Triggers are event-driven specialited procedure they are stored in and managed by DBMS. A trigger cannot be called or executed; The DBMS automatically fires the trigger as a result of a data modification to the associated table each trigger is attached to a single, specific table in that.

→ Stored procedures are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

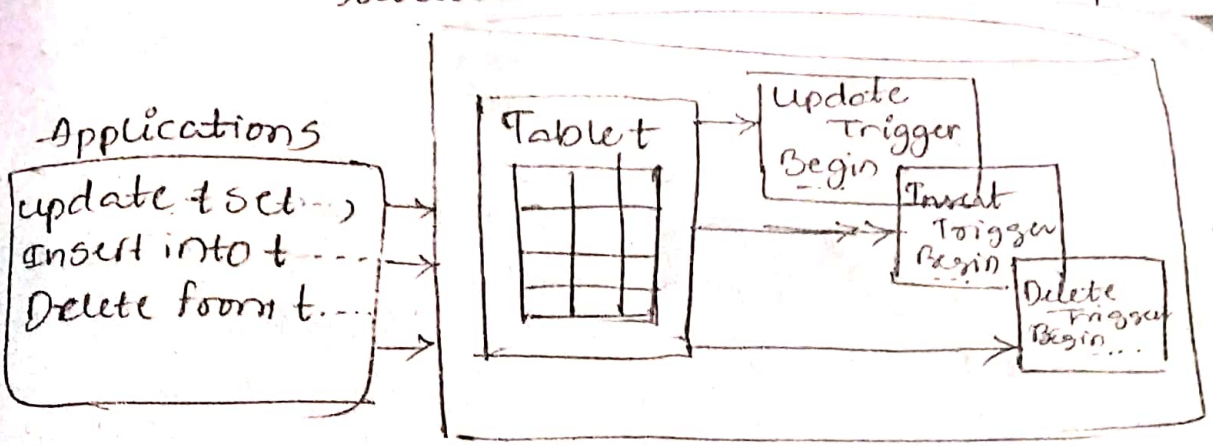


fig. Triggers

Triggers that fire whenever one of the following Operation ~~causes~~ occurs:

1) DML Stmts (INSERT, UPDATE, DELETE) on a particular table or view, issued by any user.
2) DDL stmt. (CREATE (or) ALTER)
3) Database events, such as logon/logoff, errors, or start up/shutdown;

Advantages of Triggers (How Triggers are used)

1) Automatically generate derived Column values
2) Prevent invalid Transactions
3) ~~Explore~~ Enforce Complex security authorizations.
4. Enforce referential Integrity.
5. provide auditing.
6. Provide transparent Event logging.
7. Maintain Synchronous table replicates
8. Modify table date when DML Stmts are issued against views.
9. publish inf. about db events, user events & SQL Stmts to Subscribing app^ns.

* Types of Triggers in Oracle:

Triggers can be classified based on the following parameters.

- Classification based on the timing
  - Before Trigger: It fires before the specified event has occured.
  - AFTER trigger: It fires after the specified event has occured.
  - INSTEAD of Trigger: A special Type. You will learn more about the further topics (only for DML)
- Classification based on the level.

STATEMENT level Trigger: It fires one time How to create Trigger.

Below is the syntax for creating a trigger.

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
[BEFORE | AFTER | INSTEAD OF]
        Table.          >View
[INSERT | UPDATE | DELETE ... ]
ON <name of underlying object>
[FOR EACH ROW]
[WHEN <condition for trigger to get execute>]
DECLARE
<Declaration part>
BEGIN
<execution part>
EXCEPTION
<exception handling part>.
END;
```

* Syntax Explanation:

- The above Syntax shows the different optional Statements that are present in trigger creation.
- BEFORE/AFTER will specify the event timings.
- INSERT/UPDATE/LOGON/CREATE/etc. will specify the event for which the triggers needs to be fired.

• ON clause will specify on which object the above mentioned event is valid. For example, this will be the table name on which the DML event may occur in the case of DML Trigger.

• command "FOR EACH ROW" will specify the ROW level Trigger.

• WHEN ~~clause~~ clause will specify the additional cond. in which trigger needs to fire.

• The declaration part, execution part, exception handling part is same as that of the other PL/SQL blocks. Declaration part & exceptional handling part are optional.

* INSTEAD OF Trigger:

"INSTEAD OF trigger" is the special type of trigger. It is used only in DML triggers. It is used when any DML event is going to occur on the complex view.

Consider an example in which a view is made from 3 base tables. When any DML event is issued over this view, that will become invalid because the data is taken from 3 different tables.

So in this INSTEAD OF trigger is used. The INSTEAD OF trigger is used to modify the base tables directly instead of modifying the view for the given event.