Emp (eid, Sal)

Q) Retrieve eids who gets minimum salary. (<= every.)

$\pi_{eid}(emp) - \pi_{eid}(Emp \bowtie_{sal > S} \rho_{I,S} Emp)$ U - (> some) = $A = U - \bar{A}$

universal set — All employees − not minimum = minimum employee.

Select Eid

From Emp T_1

where not exists (Select * one row

from emp T_2

where $T_1.Sal > T_2.Sal$)

$T_1.Sal > T_2.Sal$

not minimal.

Emp (eid, sal, dno)

Retrieve eids of employees who gets min salary for each dept

<= every U - (>)

π_{eid} Emp - π_{eid} (Emp $\bowtie_{sal > S}$ $\rho_{(emp)}$ $_{I,S,D}$)

$\wedge$ dno = D

this condition.

for the previous SQL queries just add

we can use grouping and aggregation function.

Select eid
⋮
Group by dno

✗ eid is not present in group by clause.

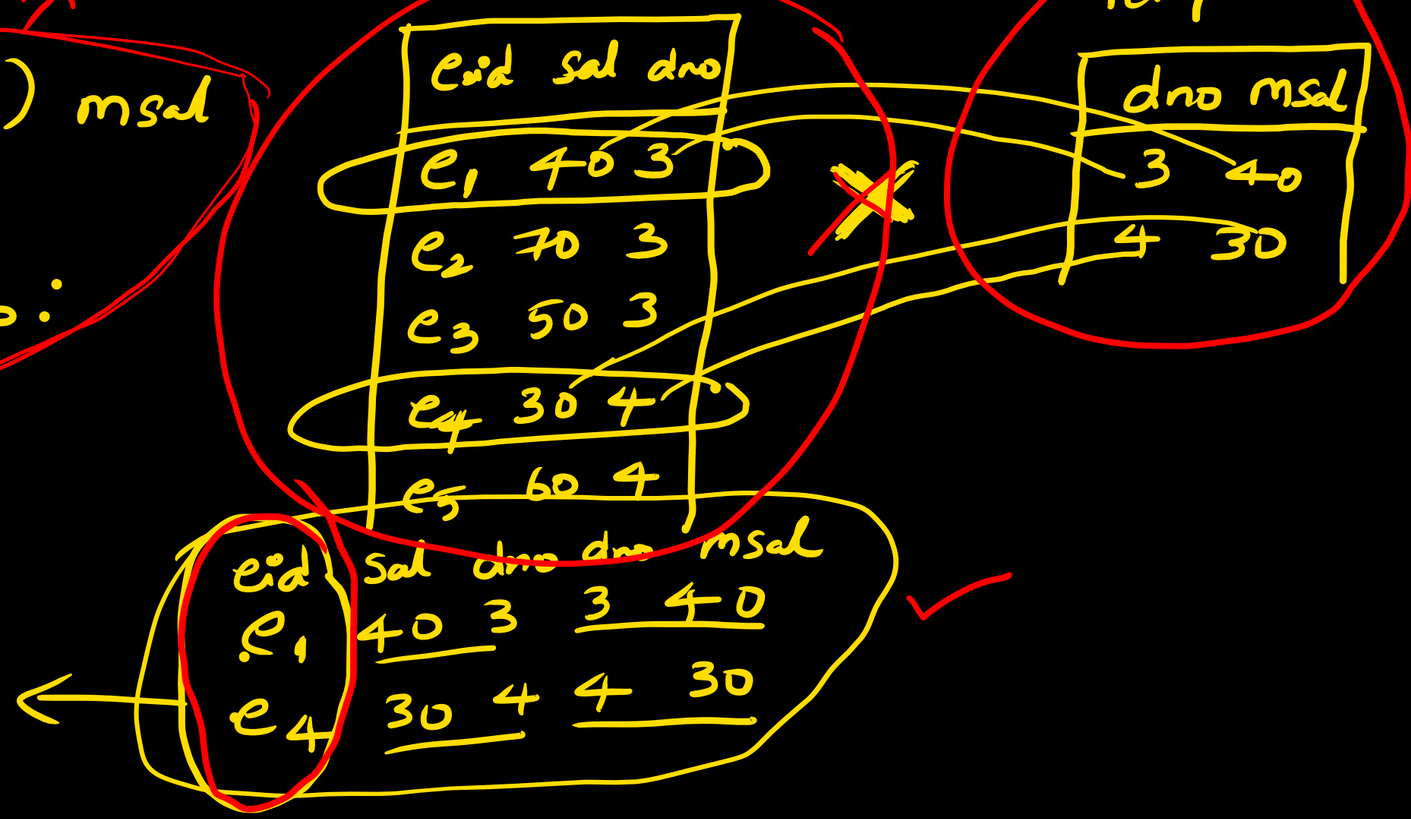
Select dno, min(sal)
⋮
Group by dno;

✗ we want eid not dno

we have to used nested queries.

Select e.id
From Emp, (Select dno, min(Sal) msal
From Emp
Group by dno) temp : table.

where Emp.dno = temp.dno
and Sal = msal



SQL query which is equal to division of RA:

Enroll (Sid, Cid) Course (Cid, Instructor)

Retrieve Sid's who enrolled in every Course taught by Kolth.

RA: $\left(\pi_{Sid,Cid}(Enroll) \div \pi_{Cid}\left(\sigma_{Instr = Kolth}(Course)\right) \right)$ → we saw this.

we have seen.

How to do division in SQL

Enroll (T_1) — Outer

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_4
S_3	C_4

Course

Cid	Ins
C_1	K
C_2	K
C_3	K
C_4	N

Enroll (T_2) — Inner

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_4
S_3	C_4

I have taken two instances of Enroll. For nested query.

S_1 ✓

Retrieve $T_1.Sid$ from Enroll (T_1)

if { all cids of Kolth } − { cids of Courses enrolled by $T_1.Sid$ from Enroll (T_2) } $= \phi$ → not exists.

$(C_1 C_2 C_3) - (C_1 C_2 C_3 C_4) \neq \phi$.

$\forall T_1.Sid \to S_1 \Rightarrow (C_1, C_2, C_3) - (C_1 C_2 C_3) = \phi$. ✓

$T_1.Sid \to S_2 \Rightarrow (C_1 C_2 C_3) - (C_1 C_4) = (C_2 C_3)$

$T_1.Sid \to S_3 \Rightarrow (C_1 C_2 C_3) - (C_4) = (C_1 C_2 C_3)$

Select distinct sid
From Enroll T_1
where not exists (Select cid
From Course
where Ins = Kolth → all Kolth Courses.
Except
Select cid From
Enroll T_2 where
T_2.sid = T_1.sid → Courses taken by one student T_1.sid

= ϕ

)

works for (eid , pid) Project (Pid , name)

Retrieve all eids who works for every project of Pname DB

works for (T_1)

eid	pid
1	1
1	2
1	3
2	1
3	4

Project

Pid	name
1	DB
2	DB
3	DB
4	OS

works for (T_2)

eid	Pid
1	1
1	2
1	3
2	1
3	4

eid = 1 $\{1,2,3\} - \{1,2,3\} = \phi$

eid = 2 $\{1,2,3\} - \{1\}$

$2,3 \neq \phi$

eid 3 $\{1,2,3\} - \{4\}$

$1,2,3 \neq \phi$.

Retrieve T_1.eid from works for (T_1)

If (all pids of pname = DB) $-$ { Pids works by T_1.eid } $= \phi$

```
Select distinct T1.eid
From works AS (T1)
where not exists (
    Select Pid
    from Project
    where Pname = DB
    except
    Select Pid
    From works AS T2
    where T2.eid = T1.eid)
```

$\neq \phi$

$= \phi$ (ϕ)

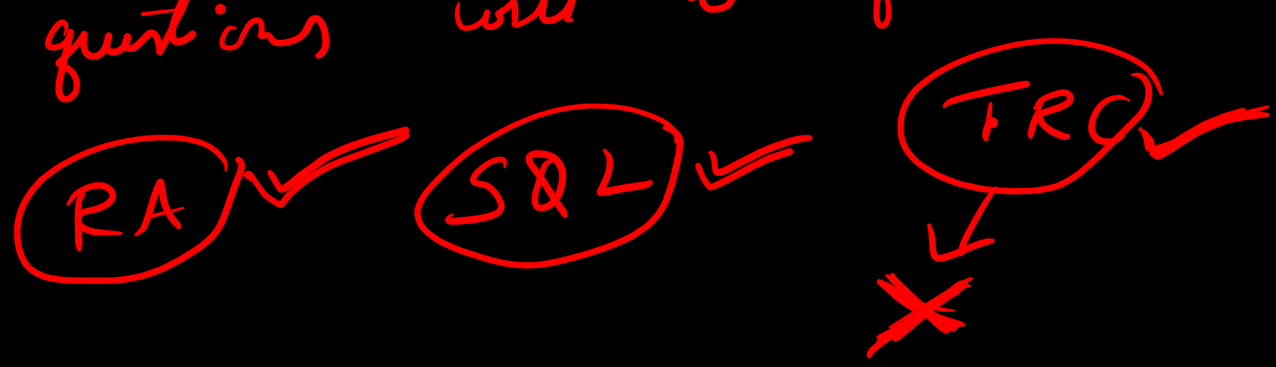
Pids of DB project.
{1, 2, 3}

All the project that some employee is doing
{1, 2, 3}

End of SQL:

We have solved all the models.

Gate questions will be from my notes.

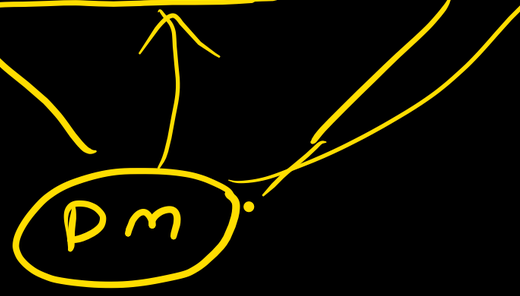


Tuple relational calculus (TRC):

→ Non procedural query language

→ It specifies what we want but not how we want.

→ uses first order logic & predicate calculus:



∴ You will not understand everything.

After DM is over, You revisit this.

Basic formulae used in TRC:

P, Q are predicates
↳ (statement which can be true/false)

$P \vee Q$

$P \wedge Q$

$\neg P$

$P \rightarrow Q$

$x \in Rel$

$\exists x \in Rel\ (P(x))$
For some x which belongs to Rel s.t. P(x) is TRUE

$\forall x \in Rel\ (P(x))$
For every $x \in Rel$ P(x) is always true.

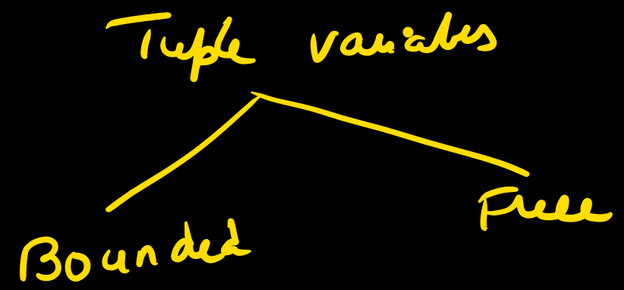
Format of TRC Query:

$\{T / P(T)\}$ T: Tuple variable P(T): formula over tuple variable T.

Returns set of tuples (T) those satisfy P(T)

Ex: $\{ T / T \in Stud \wedge T.age > 20 \}$

Retrieve all students whose age is more than 20.



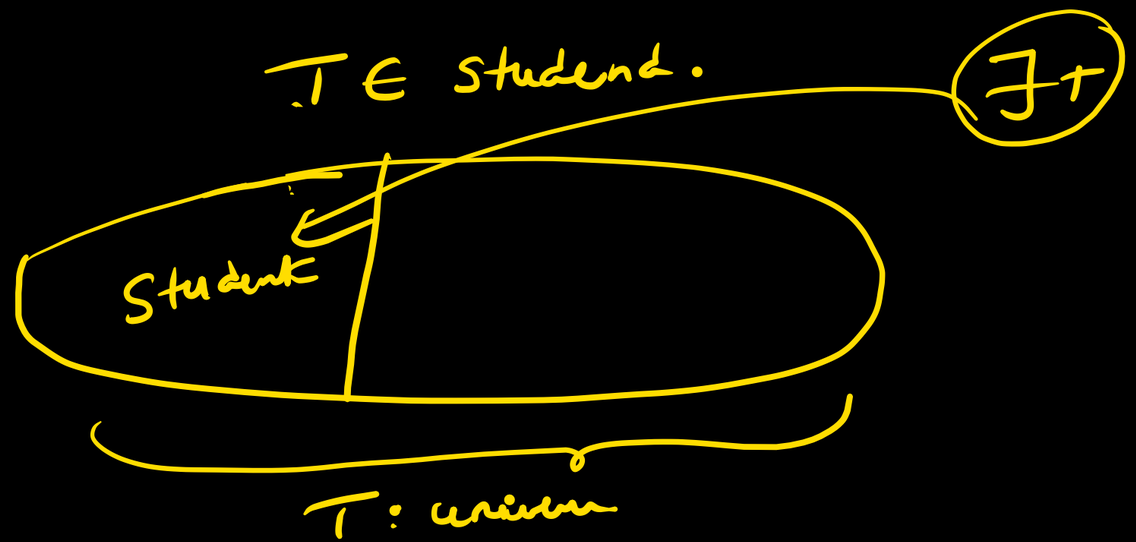
Bounded tuple variables: Variables bounded by quantifier ($\exists, \forall$)

Ex: $\exists T_1 \in$ Student (·····)

$\forall T_2 \in$ Course (·--·--··)

Free variable: not bounded by quantifier

universal variable

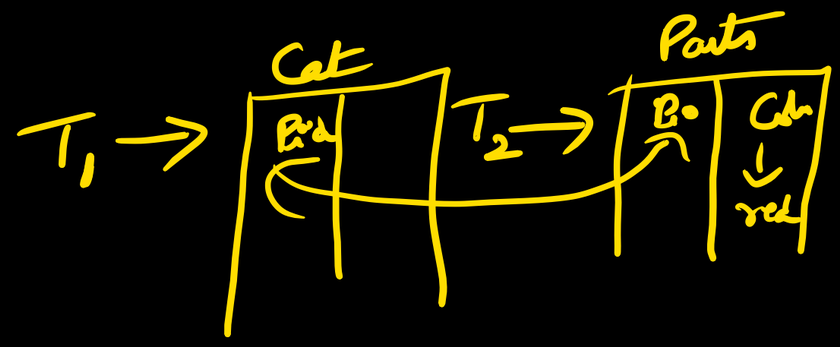


Suppliers (Sid , rating)

Parts (Pid, Pname , color)

Catalog (Sid Pid cost

Retrieve Sids of Suppliers who supplied some red parts.



Catalog (Sid Pid) Parts (Pid , col) red

RA:

$$\pi_{Sid}\Big(\sigma_{Catalog.Pid = Parts.Pid}\big(Catalog \times \sigma_{col = Red}(Parts)\big)\Big)$$

$$\left\{\begin{array}{l} \sigma_b(R \times S) \\ R \bowtie_b S \\ R \bowtie S \end{array}\right\} \begin{array}{l} \text{Some} \\ \text{/any} \\ \text{/at least one} \end{array}$$

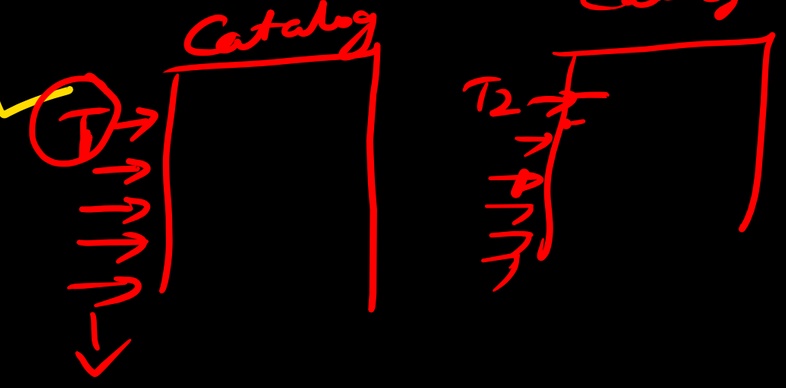
In TRC we will use $\exists$ existential quantifier

Retrieve Sids of Suppliers who supplied atleast two parts.

Catalog(Sid, Pid) = Catalog(Sid, Pid) ≠

RA:

$$\pi_{Sid}\left(Catalog \bowtie_{Sid=S \,\wedge\, Pid \neq p} \rho_{S}(Catalog)\right)$$



TRC:

$$\{T \mid \exists T_1 \in Catalog, \exists T_2 \in Catalog\ (T_1.Sid = T_2.Sid \wedge T_2.Bid \neq T_2.pid \wedge T = T_1.Sid)\}$$

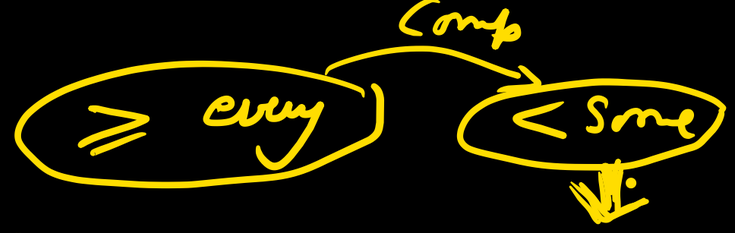
$$\begin{cases} R \cup S = \{T / T \in R \vee T \in S\} \\ R \cap S = \{T / T \in R \wedge T \in S\} \\ R - S = \{T / T \in R \wedge \neg T \in S\} \end{cases}$$

$$R - S = R \cap \overline{S}$$

Supplier (Sid , rating)

Parts (Pno , Pname , Code)

Catalog (Sid, Pid , cost



Retrieve Supplier ids whose rating is maximum.

RA:: $\Pi_{Sid}(Supplier) - \Pi_{Sid}(Supplier \bowtie_{rating < R} \rho_R(Supplier))$

TRC: $TRC = \{ T / \exists T_1 \in Supplier (T = T_1.Sid) \wedge$

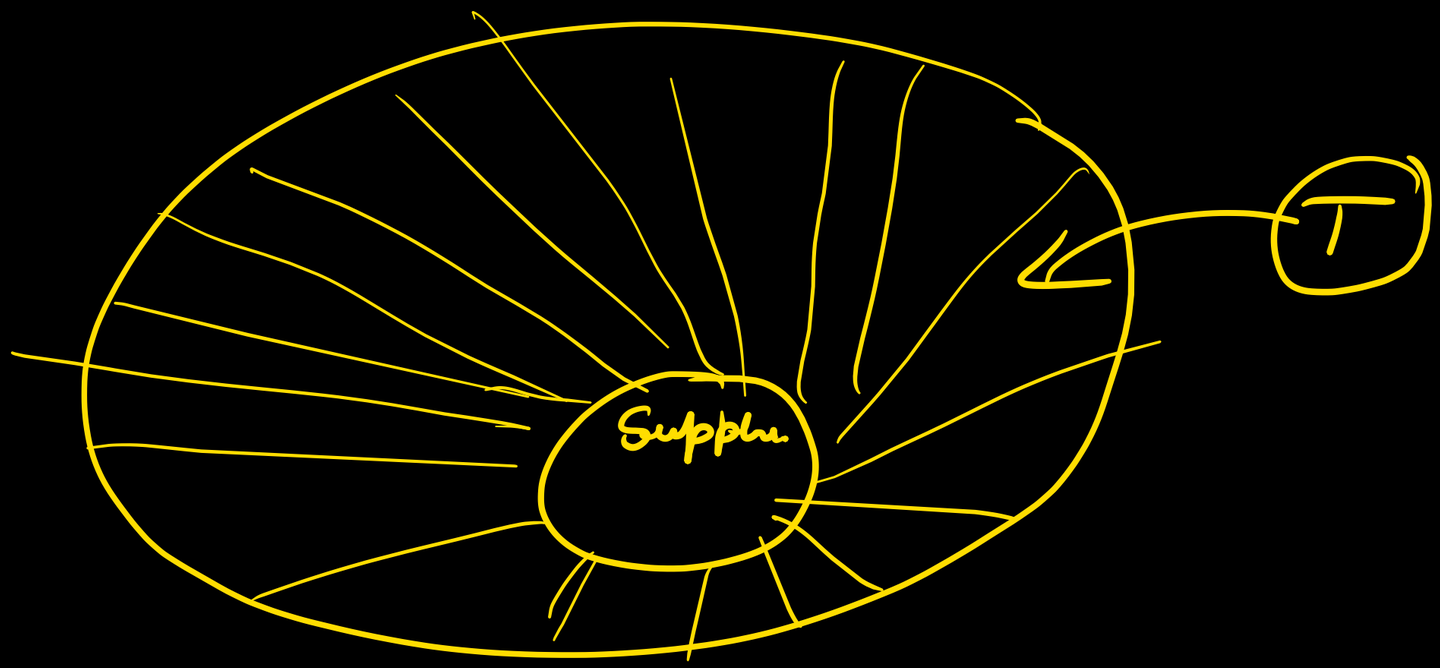
$\neg \exists T_1 \in Supplier \; \exists T_2 \in Supplier$

$(T_1.rating < T_2.rating \wedge T = T.Sid) \}$

Conclusion:

Unsafe: TRC query with infinite tuples in result is unsafe.

$\{T \mid \neg(T \in Supplier)\}$



Expressive Power :-

{ Basic RA queries expressive power } = { Safe TRC queries expressive power }

TRC can do the following operations of RA

{ π, $\times$, $\cap$, ρ, $\cup$, $-$, $\cap$, $\bowtie$, $\bowtie_c$, $\overline{\bowtie}$, $(\bowtie)$, $\overline{\bowtie}$, $/$ $\wedge$ $\%$ }

Safe TRC / RA cannot do the following:

a) Count of records / Count of attribute values

b) Sum of attribute values

c) Avg of attribute values

RA, SQL, TRC.

99%

ER model :