Relational Algebra (Procedural

It is based on set theory, so it removes duplicates.
It uses certain operations to answer a query.

select (σ) :- to select required tuples.

Project (π)
selects the columns, vertical separation

o/p < Unary / Binary.
(in RA) only Unary & Binary.

$\pi_{A_1, A_2, A_3}(R)$ Here relation is the input (single input)
Projection is unary operation

$\underbrace{\pi_{A_1}(\underbrace{R_1 \bowtie R_2}_{\text{this is binary}})}_{\text{Unary}}$

for projection, only single table.
Join is a binary operator.

Issuer (IID, IName, IM, IA)
Book (BID, BN, BA, BC (bole), IC, BP)
IBR (IID, BID).

Issuer issues Book.
many to many.

Issuer

I_1	IAB	91	AP
I_2	IBC	92	MP
I_3	ICD	93	UP
I_4	IDE	94	MH.

Book

B_1	Dbms	Korth.	10	2	100
B_2	OS	Galv.	10	1	200
B_3	DDA	Coreman	10	1	300
B_4	C	Kametkar	10	1	100

IBR

I_1	B_1
I_2	B_1
I_3	B_2
I_3	B_3
I_4	B_4.

π_{BA}(Book) Korth, Galv, Coreman, Kametkar
π_{IName}(Issuer) IAB, IBC, ICD, IDE

Selection:- Horizontal separation.

Filter for rows / records.
Only rows which satisfy our condition.
predicate → It is the condition.

σ_P P: predicate.
The tuples which only satisfy our predicate.

π_{BID} (Book). Projects Book ID from Book

~~$\sigma_{BC>5}(\pi_{BID}(Book))$~~ $\pi_{BID}(\sigma_{BC>5}(Book))$

π_{BID} ~~(Book($\sigma_{BC>5}$))~~ Select only those tuples which satisfy condition & projects BID.

Selection is a unary operator.

Rename:- (ρ) :- also unary operator.

$\rho_{newtablename(NA1, NA2)}(R)$.

R with A1, A2 is renamed with Newtablename(NA1, NA2).

$\rho_{(NA1, NA2)}(R)$ attributes are renamed.

for renaming attributes, write within brackets.

$\rho_{NA1}(R)$

Tablename R is renamed with NA1

All BID where BC > 5.
Rename the new table as Book ID.

$\rho_{BookID}(\pi_{BID}(\sigma_{BC>5}(Book)))$

$\rho_{BA='BookAdd'}(Book)$

it will rename f.

Union:- $R_1 \cup R_2$ arity of R_1 and R_2 must be same.

$R_1 \cup R_2 = \{t \mid t \in R_1 \text{ OR } t \in R_2\}$

(1) no. of attributes should be equal in R_1 & R_2.
(2) Domain of the attributes (corresponding) must be same (Domain compatability)

Suppose Issuer

IID	IName	IM	IA

Book.

BID	BN	BC	BA

Eg: $\pi_{IID}(\text{Issuer}) \cup \pi_{BID}(\text{Book})$

Intersection:- $R_1 \cap R_2 = \{t \mid t \in R_1 \wedge t \in R_2\}$.
Arity of R_1 & R_2 must be same.

Set difference:- $R_1 - R_2 = \{t \mid t \in R_1 \text{ and } t \notin R_2\}$.
Arity must be same

Ex:-

A

Col1	Col2
1	1
1	2

B.

Col1	Col2.
1	1
1	3

A∪B.

Col1.	Col2.
1	1
1	2
1	3

A∩B.

Col1	Col2.
1	1

A−B.

Col1	Col2
1	2.

Cartesian product:- (also called cross Product)

$R_1 \times R_2 = \{\langle r_1, r_2 \rangle \mid r_1 \in R_1 \wedge r_2 \in R_2\}$

cartesian product:-

A×B.

1	1	1	1
1	1	1	3
1	2	1	1
1	2	1	3

Join:- filter acc. to condition (selection criteria/filter).
first finds cartesian product & applies condition.
So, $t < t_1 \times t_2$ if t_1 – no. of tuples in A.
t_2 – " " " B.

selection criteria/filter over cartesian product.

Inner join
- Theta join. (θ)
- equijoin → in this there is a special case of Natural join

R_1 $(C_1\ C_2)$ R_2 $(C_3\ C_4)$

Join represented by ⋈
$\bowtie_\theta$ ⇒ theta join
$\bowtie_=$ ⇒ equijoin.

$R_1 \bowtie_\theta R_2$.

$\bowtie_{R_1.C_2 > R_2.C_3}$. ⇒ theta join (other than equal to).

$\bowtie_{R_1.C_2 = R_2.C_3}$ ⇒ equijoin (relational o/p is equal to)

$\bowtie_{R_1.C_1 = } \bowtie_{R_2.C_1}$ — Suppose $R_1(C_1 C_2)$, $R_2(C_1 C_3 C_4)$

Here common attribute. So it is a natural join.

If there are an equi join & the attributes are common then Natural join.

$\bowtie_{R_1.C_2 = } \bowtie_{R_2.C_3}$ — C_2 SID, C_3 SID

then both equi & Natural join.

if C_2 - SID C_3 - BID

then only equi join.

R_1, R_2 can be instances of one relation or may be diff. totally.

$R_1 \bowtie R_2$

$R_1 \bowtie_{R_1.C_1 > R_2.C_2} R_2 \Rightarrow$ Theta join.

~~Here R~~

~~In inner~~ joi

Outer join:-

left outer Join:- We care about the tuples from left which didn't satisfy join criteria.

Right outer join:- We care about the tuples from right which didn't satisfy join criteria.

Full outer join:-

$A \bowtie_{A.col2 > B.col2} (B)$ (Theta join)

col1	col2
1	2

A×B

1	1	1	1
1	1	1	3
1	2	1	1 (✓)
1	2	1	3

$A \bowtie_{A.col2 = B.col2} (B)$ (Equi join)

col1	col2
1	1

Natural join.

C

Num	Sq.
2	4
3	9

D

Num	Cube
2	8
3	27

C×D

2	4	2	8
2	4	3	27
3	9	2	8
3	9	3	27

C ⋈ D.

Num	sq	Cube
2	4	4
3	9	27

Outer join:-

A

Num	Sq
2	4
3	9
4	16

B

Num	Cube
2	8
3	18
5	75

A×B

2	4	2	8 ✓
2	4	3	18
2	4	5	75
3	9	2	8 ✗
3	9	3	18 ✓
3	9	5	75
4	16	2	8
4	16	3	18
4	16	5	75

$A ⟕ B$

Num	Sq	Cube
2	4	8
3	9	18
4	16	NULL

$A ⟖ B$

Num	Sq	Cube
2	4	8
3	9	18
5	NULL	75

$A ⟗ B$

Num	Sq	Cube
2	4	8
3	9	18
4	16	NULL
5	NULL	75

Relational Calculus

Relational calculus is not procedural.

Relational → DRC (Domain), TRC (Tuple)

Query form $\{T \mid F(T)\}$ → formula.

T: Tuple variable.

(1) variables.

(2) Many T.V are allowed in formulas.

has to be a Free variable → which donot include existence quantifiers.

Projection is not allowed. $(\exists\ \forall)$ if $\exists T$ $\forall T$. then its is not a free variable.

F(T) - a formula can be ~~an~~ atomic formula.

(1) $T \in$ Relation)

$S \in$ sailor.

S.Rating > R.Rating

(2) T.a op ~~R~~ S.b $(<, >, <=, >=, =, <> \neq)$.

(3) T.a op constant

if F_1 & F_2 are formula, then. $\neg F_1, \neg F_2, F_1 \wedge F_2, F_1 \vee F_2, F_1 \Rightarrow F_2$ are also formulas.

if F(T) is a formula then $\forall T \in R(F(T))$

$\exists T \in R(F(T))$

Sailor (Sid, SName, ~~Sa~~ age, rating)

Boat (Bid, BN, Color)

Reserve (Bid, Sid, Day)

Find a sailor with rating > 7

$\Rightarrow \{S \mid S \in Sailor \wedge S.rating > 7\}$

Find a sailor whose age < 30.

$\Rightarrow \{S \mid S \in Sailor \wedge S.age < 30\}$.

Find the name of the sailor with rating > 7.

$\Rightarrow \{S \mid \exists S_1 \in Sailor (S_1.rating > 7 \wedge S_1.Name = S.Name)\}$

Find age and name of the sailors with rating > 7.

$\Rightarrow \{S \mid \exists S_1 \in Sailor (S_1.rating > 7 \wedge S_1.name = S.name \wedge S_1.age = S.age)\}$

$S_1 \Rightarrow$ complete table with $R > 7$.

$S \Rightarrow$ only name & age from S_1

selection $\{S \mid S \in Sailor \wedge S.rating > 7\}$. $\rightarrow$ ~~not~~ a free variable

projection $\{S \mid \exists S_1 \in Sailor (S_1.rating > 7 \wedge S_1.name = S.name)\}$

$\rightarrow$ not a free variables.

Find sailors rating > 7 and who have reserved boat id $= 103$.

$\{S \mid S \in Sailor \wedge S.rating > 7 \wedge (\exists R \in Reserve (R.Sid = S.Sid \wedge R.Bid = 103))\}$

Find sailor name for above.

$\{S \mid \exists S_1 \in Sailor (S_1.rating > 7 \wedge S_1.name = S.name \wedge (\exists R \in Reserve (R.sid = S_1.sid \wedge R.Bid = 103)))\}$

Find sailor name who reserved a red Boat.

$\{S \mid \exists S_1 \in Sailor (S_1.name = S.name \wedge (\exists R \in Reserve (R.Sid = S_1.Sid \wedge (\exists B \in Boat (B.Bid = R.Bid) \wedge B.color = Red)))))\}$

S is free variable

S_1, R, B aren't free variable.

or)

$\{S \mid \exists S_1, \exists R, \exists B (S_1.Sid = R.Sid \wedge R.Bid = B.Bid \wedge B.Color = red \wedge S_1.name = S.name)\}$

Find a sailor who has reserved red or green boat.

~~$\{S \mid S \in Sailor (\exists R \in Reserve ($~~

Same as above but $\rightarrow B.color = red \vee B.color = green$.

Find a sailor who reserved Red & Green color boat

($B.color = red \wedge B.color = green$) so,

which means a sailor has a boat with both green & red colour.

so write separates:

$\{S \mid S \in Sailor\ (\exists R \in Reserve\ (R.Sid = S.Sid) \wedge (\exists B \in Boat\ (R.Bid = B.Bid \wedge B.color = red)))$

$\{S \mid S \in Sailor, \exists R \in Reserve, \exists B \in Boat\ (S.Sid = R.Sid \wedge R.Bid = B.Bid \wedge B.color = reed) \wedge (S.Sid = R.Sid \wedge R.Bid = B.Bid \wedge B.color = green)$

Find a sailors who has reserved two diff. boats

$\cancel{\{S \mid S \in Sailor, \exists R \in Reserve\ (R.Sid = S.Sid} \wedge$

$\{S \mid S \in Sailor, \exists R_1 \in Reserve, \exists R_2 \in Reserve\ (R_1.sid = S.Sid \wedge R_2.Sid = S.Sid \wedge R_1.bid \neq R_2.bid)\}$

Domain Relational Calculus (DRC).

Domain of attributes belonging to a relation.

Query format :- $\{ \langle x_1 x_2 \dots x_n \rangle \mid F(\langle x_1 x_2 \dots x_n \rangle) \}$.

$x_1 x_2 \dots x_n \rightarrow$ Domain variables.

In TRC, we can't mention individual attribute we ask for whole tuple. $\{S.Id$

But in DRC, it is possible.

$\langle x_1 x_2 \dots x_n \rangle \in R$. where R is a relation on n attributes and $x_1, x_2, \dots, x_n$ are domain variables.

x op y x op constant.

Find all sailors with a rating above 7.

Sailor (sid, SN, R, age).

$\{ \langle sid, SN, R, age \rangle \mid \langle sid, SN, R, age \rangle \in Sailor \wedge R > 7 \}$

$\{ T \mid T \in Sailor \wedge$ ~~T.rating~~ $T.R > 7 \}$

only Sid

→ $\{ \langle sid \rangle \mid \exists\, SN, R, age\ (\langle SId, SN, R, age \rangle \in Sailor \wedge R > 7) \}$.

Find the names of sailors who have reserved a boat 103.

TRC ~~$\{T \mid \exists T_1 \in Sailor$~~ $\wedge \exists$

$\{ T \mid \exists T_1 \in Sailor \wedge (\exists R \in Reserve\ (R.Sid = T_1.Sid \wedge R.Bid = 103$
$\wedge T_1.name = T.name)) \}$

~~SN~~   I N T A   Reserve (Sid, Bid, Day)
      Sid SN R Age   Ir   Br   D.

$\{ N \mid \exists\, I T A\ (\langle I N T A \rangle \in Sailor \wedge \exists\, Ir\, Br\, D\ (\langle Ir\, Br\, D \rangle \in Reserve$
$\wedge Ir = I \wedge Br = 103)) \}$.

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

S(I, N, T, A), B(B, BN, C) R(Ir, Br, D).

$\{ N \mid \exists\, I T A.\ (\langle I N T A \rangle \in Sailor \wedge \exists$ ~~BB~~ $Ir\, Br\, D\ (\langle Ir\, Br\, D \rangle \in R \wedge$
$Ir = I \wedge \exists\, B, BN, C\ (\langle B\, BN\, C \rangle \in B \wedge B = Br$
$\wedge C = red))) \}$

Division operator

- Emp who is working on all projects
- Sailor who reserved all boats.
- Student who enrolled himself in all courses

Ans:- E_2.

"all"
Possible with division operator.

~~$R_1 \div R_2$~~ $R_1(X)$ $R_2(Y)$ X, Y are set of attributes

$R(Z) = R_1(X) \div R_2(Y) \Rightarrow$ (1) $Z = X - Y$. (attributes only in R_1 not in R_2).
$(X = Z \cup Y)$

(2) T_R is a Tuple from R_1 iff it is associated with all the tuples of R_2.

$R_1 \div R_2 = \{t \mid t \in \pi_{R_1 - R_2}(R_1) \wedge \forall u \in R_2 (tu \in R_1)\}$
(u)
all tuples in S is related with t coming from R_1

Example:- (R)

A	B
α	1
α	2
α	3
β	1
δ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

(S)

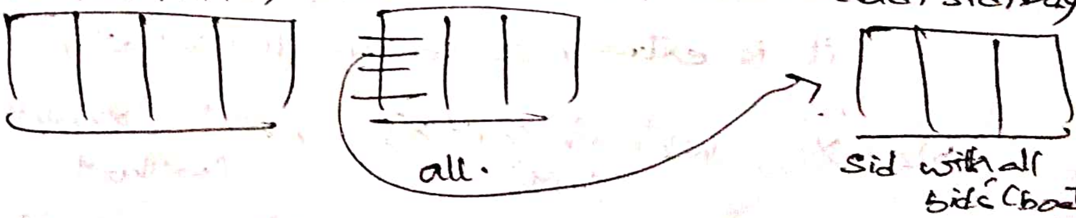
B
1
2

$R \div S$.

A
α
β

Sailor who reserved all boats. (using TRC)

Sailor (Sid, SN, R, A); Boat (Bid, BN, C); Reserve (Bid, Sid, Day)



$\{S \mid S \in Sailor \wedge (\forall B \in Boat (\exists R \in Reserve \wedge R.Bid = B.Bid \wedge S.Sid = R.Sid))\}$

unsafe queries $\{S \mid \neg\{S \in Sailors\}\}$. (or) $\{S \mid S \notin Sailors\}$ infinite no of solutions.

Every query that can be expressed in R.A can be expressed as a safe query in DRC/TRC ; the converse is also true.

R.A → Procedural
R.C → Non-Procedural
→ SQL - Non-Procedural (Implementation)

Canonical cover of FD set / Minimal set of FDs / Irreducible set of FDs.

removing extra or unwanted attributes

F^+. So this is minimal set

we have redundant FD, Extrageneous attributes } removing them.

Suppose $AB \to D$, $B \to D$ here B is sufficient to uniquely identify D. So, A is extra, Total $AB \to D$ is extrageneous

$R(WXYZ)$ $X \to W$, $WZ \to XY$, $Y \to WXZ$.

Step (1) Apply decomposition rule.

$X \to W$, $WZ \to X$, $WZ \to Y$, $Y \to W$, $Y \to X$, $Y \to Z$.

Step (2) Identify redundancy

find X^+ using the FD. $(X \to W)$ $X^+ = \{XW\}$ ✓ → So, this is essential.

X^+ without using the FD. $(X \to W)$ $X^+ = \{X\}$

$WZ \to X$ $WZ^+ = \{X, W, Z, Y\}$ (with) $WZ^+ = \{W, Z, Y, X\}$. (without)

So closure is same

it is extra, so remove it instantly.

$WZ \to Y$ $WZ^+ = \{W, Z, Y, X\}$ (with), $WZ^+ = \{W, Z\}$. (without)

Not same, So $WZ \to Y$ is essential.

$Y \to W$ $Y^+ = \{Y, W, X, Z\}$, (with) $Y^+ = \{Y, X, Z, W\}$. (without)

Same, So remove it.

$Y \to X \Rightarrow Y^+ = \{Y, X, Z, W\}$, $Y^+ = \{Y, Z\}$.

not same, so it is essential.

$Y \to Z \Rightarrow Y^+ = \{Y, Z, X, W\}$, $Y^+ = \{Y, X, W\}$.

not same, so it is essential.

$\underbrace{X \to W, WZ \to Y, Y \to X, Y \to Z}$.

Step(3):- Find redundancy on left hand side.

$\underbrace{X \to W, Y \to X, Y \to Z}_{\text{one attribute}}$ $WZ \to Y$.

$WZ^+ = \{W, Z, Y, X\}$.

$W^+ = \{W\}$
$Z^+ = \{Z\}$ } So both are essential.

So final answer:- $X \to W, Y \to X, Y \to Z$.
(canonical cover)

Eg-2:- $A \to BC, B \to C, A \to B, AB \to C$ $R(ABC)$

(1) Apply decomposition rule.

$A \to B, A \to C, B \to C, A \to B, AB \to C$.

$A \to B, A \to C, B \to C, AB \to C$.

~~$A \to C, A \to C, AB \to C$~~ So $A \to C$ no need.

~~$A \to C$~~

$A \to B$, $B \to C$, $AB \to C$.

B is alone sufficient.

So, $A \to B, B \to C$.
(canonical cover)

Eg-3:- ~~$A \to B$~~, $A \to C, B \to C, A \to B, AB \to C$.

$A \to C$ $A^+ = \{A, C, B\}$, $A^+ = \{A, B, C\}$.
with without

So, remove it.

$B \to C$ $B^+ = \{B, C\}$, $B^+ = \{B\}$.
with without

So, essential.

$A \to B$, $A^+ = \{A, B, C\}$, $A^+ = \{A\}$.

essential.

$AB \to C$, $AB^+ = \{A, B, C\}$, $AB^+ = \{A, B, C\}$.

So remove it.

So, $A \to B, B \to C$

Canonical ~~set~~ cover is not unique.

1, 2, 3, 4, 5
$\longrightarrow$ depends on the order, we go, so not unique.

=

C→B, CB→AC, CAE→FB, D→E, CA→B.

(1) C→B, CB→A, CB→C, CAE→F, CAE→B, D→E, CA→B.

(2) C→B. ; $C^+ = \{C, B, A\}$ $C^+ = \{C\}$.

So, essential.

C→B is sufficient then CA→B is extrageneous, so, remove.
CAE→B " " " So, remove.

(3) CB→A ; $CB^+ = \{C, B, A\}$, $CB^+ = \{C, B, A\}$.

Same.

So, remove it. without $C^+ = \{C, B\}$. $B^+ = \{B\}$
So, B no need.

CB→C ; $CB^+ = \{C, B, A\}$, $CB^+ = \{C, B, A\}$.
Redundant
So, remove it.

CAE→F ; $CAE^+ = \{C, A, E, B, F\}$, $CAE^+ = \{C, A, E, B\}$.
So, essential. , A is no need.

without this $C^+ =$ $A^+ =$ $E^+ =$
then 'F' won't come, So, it is essential.

D→E ; $D^+ = \{D, E\}$, $D^+ = \{D\}$
So, essential.

C→B, C~~A~~E→F, D→E, C→A.

=

F = {A→B, AB→C, ~~D→A~~ D→AC, D→E} G = {A→BC, D→AB}.

F = A→B, AB→C, D→A, D→C, D→E

G = A→B, A→C, D→A, D→B.

$F^+ =$ (Canonical cover of F)

A→B . AB→C

A identifies B. , So A→B, A→C.

AB→C
no need.

A→C. is essential
without this 'C' won't come.

D→A $D^+ = \{A, D, C, E, B\}$, $D^+ = \{D, C, E\}$.
So, essential.

D→C $D^+ = \{A, D, C, E\}$ $D^+ = \{D, E\}$
~~So essential~~

D→C, A→C.
D→A,
So, D→C is no need.

So, $A \to B$, $A \to C$, $D \to A$, $D \to E$.

$F^+ = \{A \to B\cancel{C}, \cancel{A}D \to A, D \to E\cancel{B}, A \to C\}$.

G^+

$\cancel{A \to B}$, $A \to C$, $D \to A$, $D \to B$.

$A \to B$ $\quad A^+ = \{A, B, C\}$, $A^+ = \{A, C\}$.

So, essential.

$A \to C$, $A^+ = \{A, B, C\}$, $A^+ = \{A, B\}$.

So, essential.

$D \to A$ $\quad \to$ essential.

$D \to B$. $\quad D \to A$, $A \to B$ so, $D \to B$ no need.

So, $G^+ = \{A \to B\cancel{C}, A \to C, D \to A\}$

F covers G, but G doesn't cover F.

if suppose G covers H & H covers G.

then H & G are equivalent.

Transaction Management in DBMS.

Transaction :- It is also a sequence of instructions/operations (same like process, job, task, thread..)

only diff. is All the operations of the ~~pro~~ transaction will execute completely at same time. (if some interrupts, then it won't stop there)

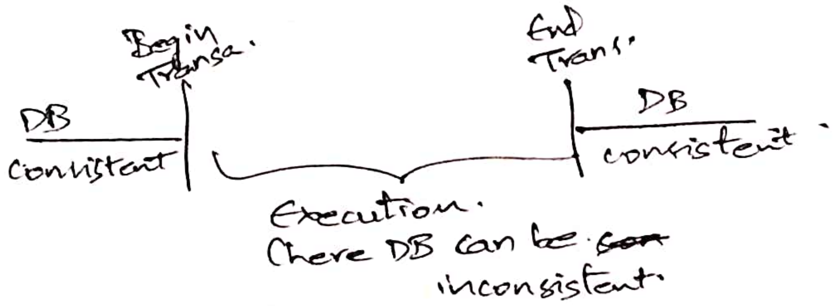
Transactions are atomic in nature (either complete totally or roll back if ~~com~~.

→ A transaction is a single ~~p~~ logical unit of work which access DB and possibly modify the DB.

ACID Properties :- Atomicity, consistency, Isolation, Durability.

~~Assumption~~, initially database is consistent,
only Transaction interacts with DB, so it has to modify the DB such that it transforms one consistent state to another consistent state.

If it transforms to inconsistent state (×)



R(A)
A=A-10
W(A)
R(B)
B+10
W(B)

If an interrupt happens here, then it must roll back.

Atomicity :- ~~Either all the~~ No partial execution of transaction is allowed

Consistency :- A transaction has to lead from one consistent state to another consistent state.

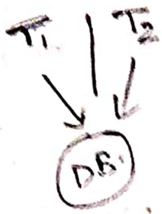
Ensure consistency.

Isolation:- Tatkal tickets, suppose (only the) transactions serially done, it takes lot of time, not possible

So, simultaneously no. of transactions done.

concurrency.

Race condition arises and leading to inconsistency



logical Isolation:- Many ~~atomic~~ transactions simultaneously done.

But, its like only one is executing for them.

but for T_1, it thinks, only it's executing

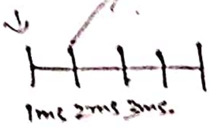
Same for T_2.

Intermediate results of individuals transactions must not be available to each other.

Durability:- ~~tatkal~~

If a transaction commits, whatever changes it does, persist forever.

Even if system fails, then can't roll back,



How to roll back a committed Transaction (not possible).

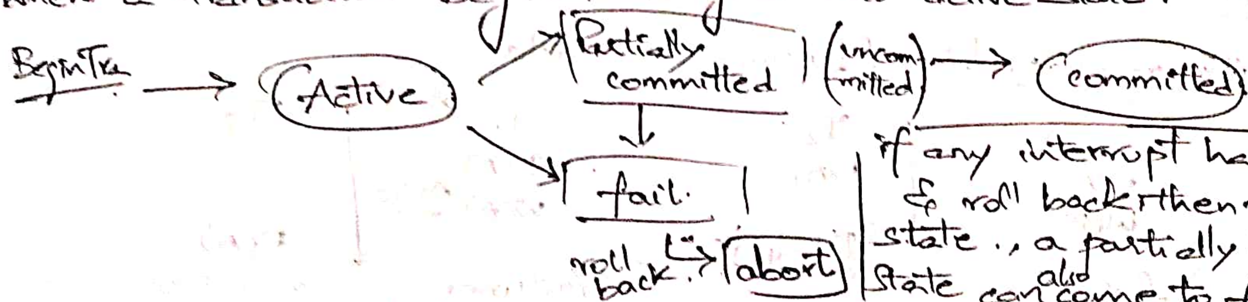
if R(A)
W(A)

Compensating Transaction

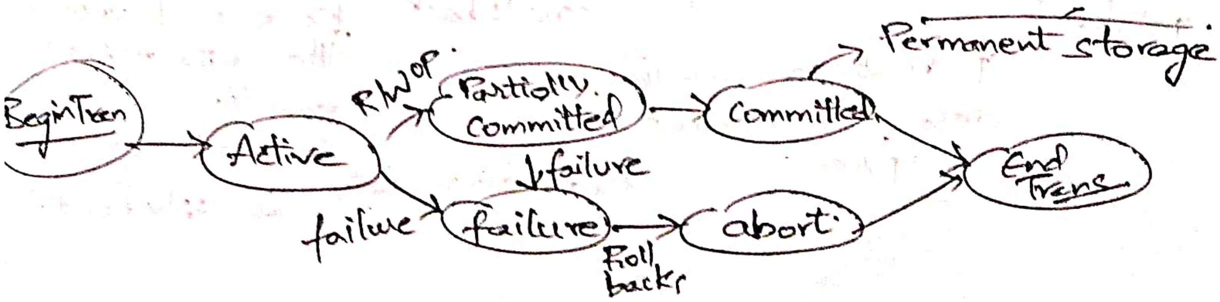
then we need unwrite, write another Transaction.

Transaction states:-

When a transaction begins, it goes into active state.



if any interrupt happens & roll back, then failure state., a partially comm. state can also come to failure.



Concurrency & problems associated with it

If, ATM's transactions are done serially, then no two persons can withdraw money at the same time. (most reliable)

So, concurrency.

Suppose T1 T2 T3 T4

Serially done (order doesn't matter) always leading to a consistent state: T1, T2, T3, T4

Serial execution of Transaction:- The execution is not overlapped.

Non-serial ex. of Trans:- (Concurrent ex. of Trans.)
- always allow overlapped Transaction

T1 T2,
T1 T3,
T2 T4

Advantage of concurrency:-

Drawback of Serial ex.:- no effectiveness, productivity ↓ces.

(1) Response Time ↓ces for T2 (T1 T2).
(2) Avg. waiting Time ↓ces. (T2 must wait for T1 in serial)
(3) Resource Utilization ↑ces
(4) Efficiency ↑es (because of all above, performance ↑es)

But concurrency may lead to inconsistent data.
So, a controlled concurrency is allowed.
(controlled by ACID properties).
A transaction with ACID properties is allowed.

Problems:-

(1) Dirty read problem / uncommitted data

T1	T2
R(A)	R(A)
W(A)	Suppose committed.
failure rolls back.	

If read a committed data & then commit no problem

T1 A=10	T2
R(A)	
W(A) A=20	
	R(A)
	committed.
failure rolls back.	

here read uncommitted data A=20 & commits.
So A=20 → wrong.
~~So reading dirty~~ So, Dirty Read.

Solutions:- (1) Donot read uncommitted data.

(2) If read an uncommitted data, then donot commit, if the previous commits, then commit.

(2). Unrepeatable read problem. (due to isolation problems).

T_1	T_2
R(A)	
	R(A) (A=10)
(A=20) W(A)	
	R(A) (A=20).

for T_2, it hasn't modified the data, but happened because T_1 has, so for $T_2 \to T_1$ is ghost.

So, for $T_2 \to$ it is not isolated

Suspects that some other Trans. is interfering

So, rolls back.

here in T_2, read is repeated, so, problem arised

So, donot repeate the read.

(4) Phantom read problem

T_1	T_2
R(x)	
	R(x)
delete(x)	
	R(x)

here its already deleted, so phantom problem.

write can happen, if can write for the first time. So only phantom read problem.

(5) Lost update problem:-

T_1	T_2
A=10 R(A)	
A=20 W(A)	
	W(A) A=30
	commits.

at last they all share a buffer.

So. previous update. A=20 is lost.