INDEX

Sl. No.	Date	Title	Page No.	Teacher's Sign/Remarks
		ER Diagram.		
		Conversion of E-R Diagram into Relational Model.		
		Basics of Relational model & functional dependency.		
		Idea about Keys / Types.		
		Normalization (1NF - BCNF)		
		Lossless decomposition & Dependency Preserving		
		Indexing and Physical Structure (B & B+ Tree)		
		SQL, RA, RC.		
		Transaction		
		Concurrency Control.		

DEBOJYOTI DAS.
1915001

Orion
PAGE:
DATE: / /

DBMS :-

DATA :-
Raw and isolated facts about an entity (recorded)
e.g text, audio, video, image, map etc.

INFORMATION :
Processed, meaningfull, usable data

DATABASE :
Collection of similar / related data.

DBMS :
S/W used to create, manipulate and delete database.

Disadvantages of file System :-
1) Data redundancy.
2) Data inconsistency.
3) Difficulty in accessing data.
4) Data isolation.
5) Security Problem.
6) Atomicity Problem
7) Concurrent-access anomalies.
8) Integrity Problem

Orion
PAGE:
DATE: / /

DBMS :-

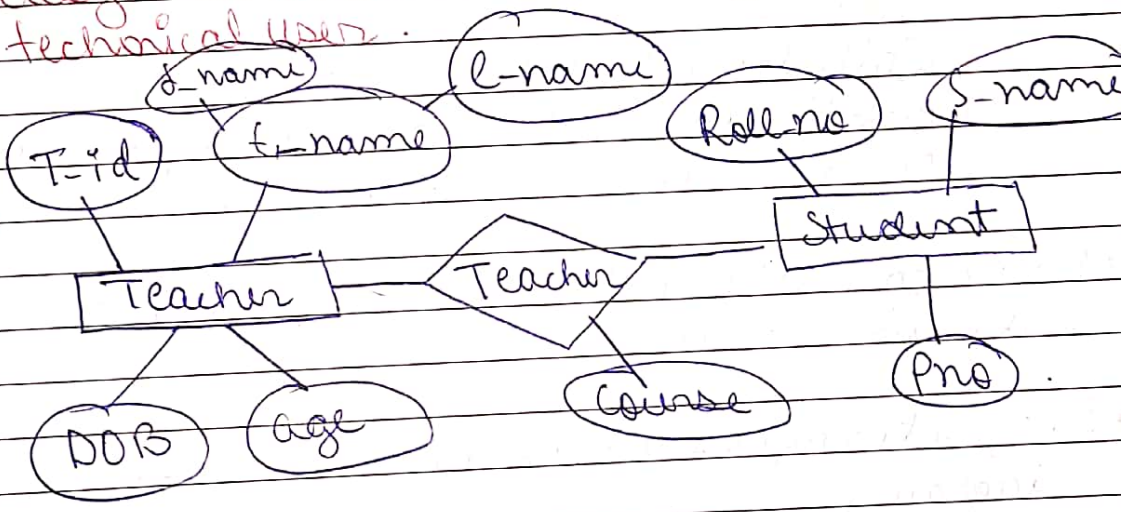
OLAP (online analytical processing)	OLTP (online transaction processing)
• historical data.	• current data
• subject oriented.	• application oriented
• decision making	• day to day operations
• TB, PB.	• MB, GB.
• CEO, MD, GM.	• clerks, managers
• R.	• R/W.

Entity Relationship Diagram

- Introduction and Basics
- Entity and Entity Set.
- Attributes and Types.
- Relationship
- Strong and weak entity set
- Traps.
- Conclusion

Orion
PAGE:
DATE: / /

- Introduced by Dr. Peter chen in 1976.
- A non-technical design method works on conceptual level based on the perception of real world.
- Consists of collection of basic objects called entities and of relationships among these objects and attributes which defines their properties.
- free from ambiguities and provides a standard and logical way of visualizing data.
- basically it is a diagrammatic representation easy to understand even by non-technical users.



GROUP → OBJECT

Orion
PAGE:
DATE: / /

Entity :-

An entity is a thing or an object in the real world that is distinguishable from other objects based on the values of the attributes it possess.

Types of Entities :-

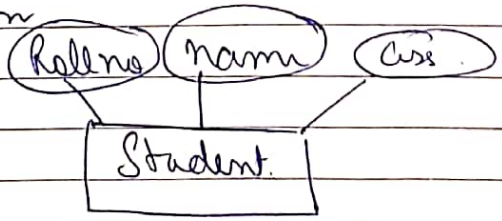
Tangible :
Entities which physically exist in real world.
e.g. car, pen, bank locker.

Intangible :
Entities which exists logically
e.g. Account.

Entity Set :-
Collection / set of same types of entities i.e. that share same properties or attributes called entity set.

→ Entity cannot be represented in an ER diagram as it is instance / data.

→ Entity set is represented by rectangle in ER diagram



→ Entity can be represented in an relational model by row / tuple / record

→ Entity set is represented by table in relational model.

Student

Roll no	name	age

→
→
→

Attributes :-
are the units that describe the characteristics of entities.

→ For each attribute there is a set of permitted values called domains.

→ In ER diagram represented by ellipse or oval which in relational model by a seperate column.

Types

Simple - Composite

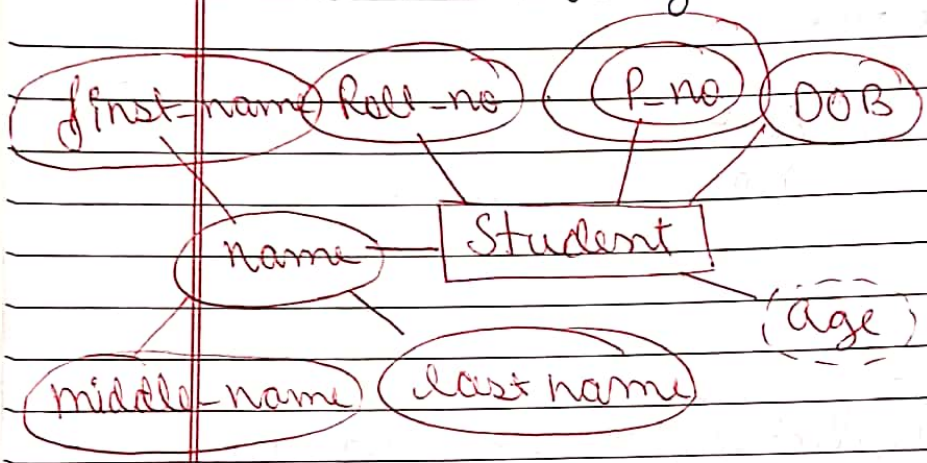
- Simple cannot be divided further represented by simple oval.
- Composite can be divided further in simple attribute, oval connected to a oval.

Simple - Multivalued

- Simple can have only one value at a instance of time
- multivalued → can have more than one value at a instance of time.

Orion
PAGE:
DATE: / /

Stored - derived.
Stored - how value is stored in the data base.
Derived - how value can be completed in run time using stored atributes.



Student

Roll no.	f_name	M_name	l_name	DOB	Age

Roll_no	Pno

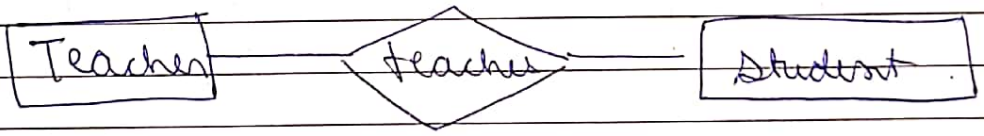
Orion
PAGE:
DATE: / /

Relationship :-

- Is an association between two or more entities of same or different entity set.
- No representation in ER diagram as it is an instance or data.
- In relational model represented either using a row in a table.

Relationship type / Set :-

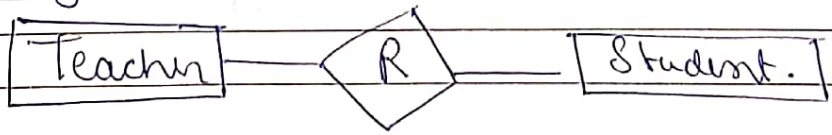
- A set of similar type of relationship
- In ER diagram represented using a diamond.
- In relational model either by a seperate table or by seperate column (foreign key)
- Every relationship type has three components (i) Name (ii) Degree (iii) Cardinality ratio / Participation constraints

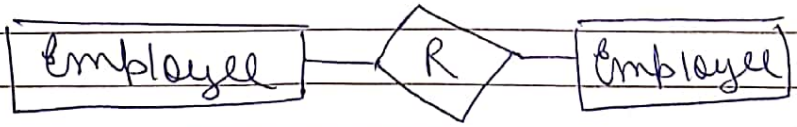


* Degree of a relationship set :-

- Means number of entity set associated (participated) in the relationship set.
- Most of relationship sets in the ER diagram are binary (in industry). Occasionally however relationship sets involve more than two entity sets.

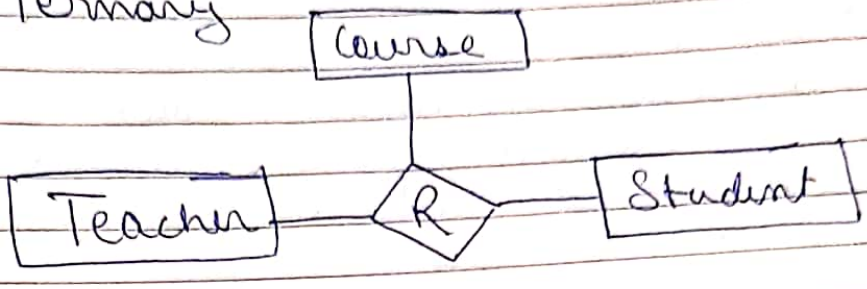
Binary :-

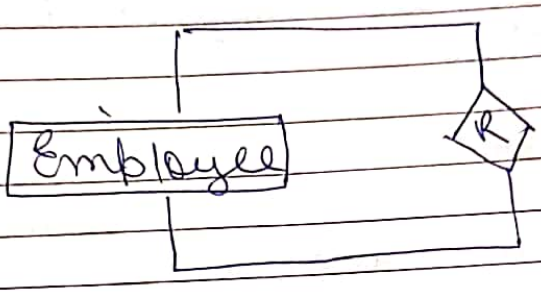




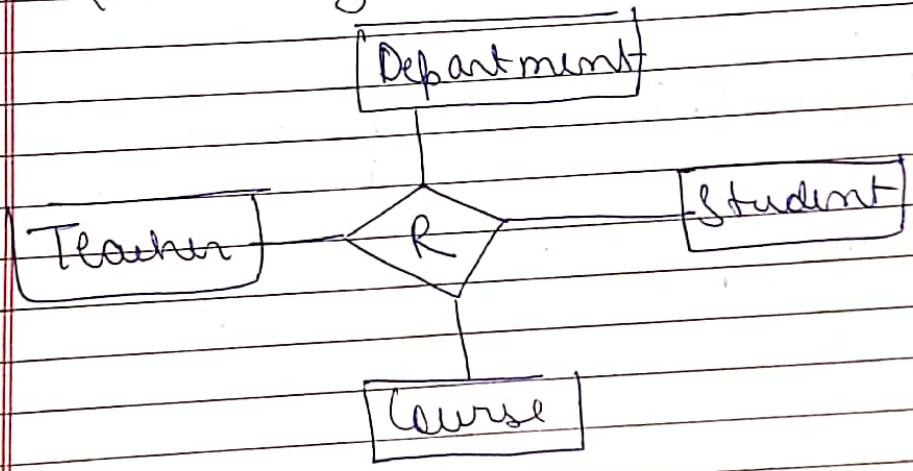
Orion
PAGE:
DATE: / /

Ternary

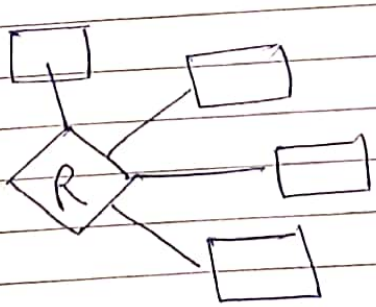




Quaternary



N-ary

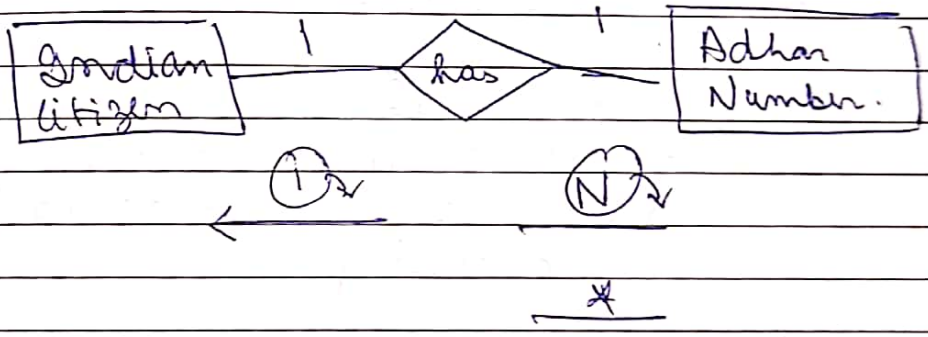


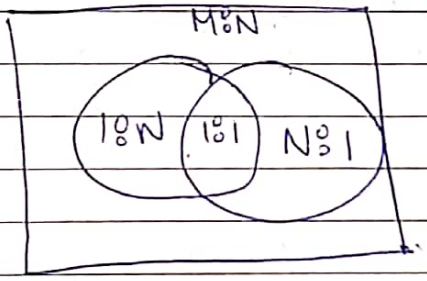
Orion
PAGE:
DATE: / /

Mapping Cardinalities / Cardinality ratio :-
Express the number of entities to which other entity can be related to a relationship.

→ can be used to in describing relationship out of any degree but is most useful in binary relationship.

→ 1:1 (one to one) → Adhar Number.
→ 1:N (one to many) → Mobile Number.
→ N:1 (many to one)
→ M:N (many to many) → Student.





Orion
PAGE:
DATE: / /

Participation constraints:

Specifies whether the existance of an entity depends on its being related to another entity via a relationship type.

There constraints specify minimum and maximum number of relationship instances that each entity can/must participate in.

Max cardinality:

It defines the maximum number of times an entity occurance participating in a relationship.

Min cardinality:

It defines the minimum number of times an entity occurance participating in a relationship.

→ Partial participation.
→ Total participation.

Orion
PAGE:
DATE: / /

DBMS

1. Functional Dependencies, Keys.
2. Normalization.
3. Indexing, Physical structures.
4. Transaction.

$f : \alpha \rightarrow \beta$

α	β
a	1
b	2 1
c	3
d	4

if we know "α" we can search "β"

For the same value of α we can't get two different values of β.

$f : \alpha \rightarrow \beta$.

$f(x) \longrightarrow y$.

1 → a ✓

1 → b ✗

2 → a ✓

$\alpha \subseteq R$, $\beta \subseteq R$

$\alpha \rightarrow \beta$

if $t_1[\alpha] = t_2[\alpha]$

↳ $t_1[\beta] = t_2[\beta]$

determinant — dependent

R

	α	β
t_1 →	a	b
t_2 →	a	b

$\alpha \longrightarrow \beta$

trivial

$AB \rightarrow A$

if $\beta \subseteq \alpha$

non-trivial

$\beta \not\subseteq \alpha$

$AB \rightarrow ABC$

Orion
PAGE:
DATE: / /

Functional Dependencies.

R

A	B	C	D	E
a	2	3	4	5
a	2	3	4	5
a	2	3	6	5
a	2	3	6	6.

a) $A \to BC$ ✓
b) $DE \to C$ ✓
c) $C \to DE$ ✗
d) $BC \to A$. ✓

$\alpha \to \beta$

✗ { a — 1
  { a — 2.

$\alpha \to \beta$
a — 1
b — 1

- check if all "α" are unique · check if all the values of C are same.

Attribute closure / closure on attribute set / closure of attribute set :-

An attribute set 'A' can be defined as a set of attributes which can be functionally determined from it Denoted by F^+

R(ABC)
$A \to B$
$B \to C$.

$A \to BC$

$(A)^+ = A$
AB
$(A)^+ = ABC$

Finitely / Infinitely.

R(ABCDEF)
$A \to B$.
$C \to DE$.
$AC \to F$.
$D \to AF$.
$E \to CF$.

R(ABCDEF)
~~$A \to B$~~
$C \to DE$
$AC \to F$
✓ $D \to AF$
$E \to CF$.

$(D)^+ = D$
$= ADF$
$= \boxed{ABDF}$

$(DE)^+ = ABCDEF$

Orion
PAGE:
DATE: / /

Armstrong's axiom / rule :-

- Axiom is a statement that is taken to be true and serve as a premise or starting point for further arguments.
- Armstrong axioms holds on very relational database, can be used to generate closure set.

~~Primary rules (RAT)~~ ~~Secondary rules~~.

- Reflexivity :
  if $y \subseteq x$
  then $x \to y$
- Union :
  if $x \to y$ && $x \to z$
  $x \to yz$.
- Augmentation :
  if $x \to y$
  then $xz \to yz$.
  (z augmented)
- Decomposition :
  if $x \to yz$
  then $x \to y$ && $x \to z$.
- Transitivity :
  if $x \to y$ && $y \to z$
  then $x \to y$.
- Pseudotransitivity
  if $x \to y$ && $wy \to z$
  then $wx \to z$
- Composition
  if $x \to y$ && $z \to w$
  $xz \to yw$.

$AB \to C$

$A \to C$
$B \to C$.

Orion
PAGE:
DATE: / /

Closure Set of attributes :-

R(ABC)
$A \to B$
$B \to C$
} $A \to BC$.

$F = f_1 + f_2$

$A \to B$ $\quad$ $A \to C$.

$A \to BC$

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

1. R(ABC DEFGH)
$A \to B$.
$BC \to DE$
$AEGH \to GH$.
$(AC)^+ = AC$
ABC
$ABCDE$
$\alpha \to B$.

2. R(ABC DE)
$A \to BC$
$CD \to E$
$B \to D$
$E \to A$
$(B)^+ = B$
BD
2 iterations
Same result

3. R(ABCDEF)
$AB \to C$
$BC \to AD$.
$D \to E$
$CF \to B$.
$(AB)^+ = ABC$
$ABCD$
$ABCDE$.

4. R(ABCD EFGH)
$A \to BC$
$CD \to E$
$E \to C$
$D \to AEH$.
$ABH \to BD$.
$DH \to BC$
$BCD \to H$?

$(BCD)^+ =$
$ABCDEH$

Orion
PAGE:
DATE: / /

Equivalence of Functional Dependency

R(ACDEH)

F: $A \rightarrow C$
$AC \rightarrow D$.
$E \rightarrow AD$.
$E \rightarrow H$.

G: $A \rightarrow CD$.
$E \rightarrow AH$.

a) $F \subseteq G$.
b) $F \supseteq G$.
c) $F = G$.
d) $F \neq G$.

$(A)^+ = ACD$
$(E)^+ = EADHC$

$(A)^+ = ACD$.
$(AC)^+ = ACD$.
$(E)^+ = EAHCD$.

R(PQRS).

X: $P \rightarrow Q$ ✓
$Q \rightarrow R$ ✗
$R \rightarrow S$ ✓

Y: $P \rightarrow QR$ ✓
$R \rightarrow S$. ✓

$(P)^+ = PQRS$.
$(Q)^+ = Q$
$(R)^+ = RS$.

$X \not\subseteq Y$.

$(P)^+ = PQRS$
$(R)^+ = RS$.

$Y \subseteq X$.

a) $X \subseteq Y$
b) $Y \subseteq X$.
c) $X = Y$.
d) $X \neq Y$.

Orion
PAGE:
DATE: / /

Irreducible set of Fd (canonical form)

R(WXYZ)

$X \to W$

$WZ \to XY$.

$Y \to WXZ$.

Redundant: if your presence or absence doesn't affect the system it is called redundant.

R(WXYZ)

$X \to W$

$WZ \to XY$.

$Y \to WXZ$

$X \to W$ ✓ $X^+ = XW$ $X^+ = X$.

$WZ \to X$ ✗

$WZ \to Y$ ✓ $(WZ)^+ = WZXY$?

$Y \to W$ ✗ $(WZ)^+ = WZYX$. ✓

$Y \to X$ ✓

$Y \to Z$ ✓ Since both results are the same so Redundant.

$X \to W$

$WZ \to Y$

$Y \to X$

$Y \to Z$.

$X \to W$

$WZ \to Y$

$Y \to XZ$

$(WZ)^+ = WZYX$.

$W^+ = W$

$Z^+ = Z$

If they are the same it means the presence or absence of Z doesn't create any diff.

Key → chabi
to extract data

R

A	B	C
1	a	p
2	b	q
3	c	q
4	c	n

Orion
PAGE:
DATE: / /

The attributes which can uniquely identify a row in a table is called ..... key.

Key = A, BC

Key = A → BC.

$(A)^+ = R$.

$\boxed{(key)^+ = R}$

1. R(ABCD)

A → BC

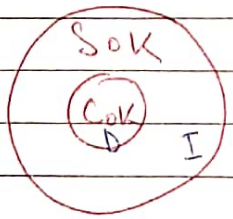
$A^+ = ABC$

$\boxed{(key)^+ = R}$

$(SoK)^+ = R$

↓

→ CoK (efficient)



2. R(ABCD)

ABC → D.

AB → CD.

A → BCD.

$(ABC)^+ = ABCD$.

$(AB)^+ = ABCD$.

$(A)^+ = ABCD$.

subset

3. R(ABCD)

B → ACD.

ACD → B.

$(B)^+ = ACDB$.

$(ACD)^+ = ACDB$

B → } Super key /
ACD → } candidate key

1. R(ABCD)

A → B.

B → C

C → A.

AD, BD, CD

$(AD)^+ = ABCD$.

$(BD)^+ = ABCD$.

$(CD)^+ = ABCD$.

R(ABCD)

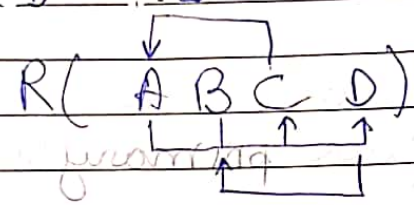
Find the attribute with no incoming edge.

Orion
PAGE:
DATE: / /

Essential.

R(A B C D E F)

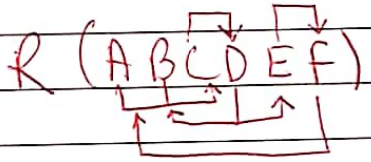
AB → CD
C → D.
D → B



$(A)^+ = AX$ — AB ✓
$(B)^+ = BX$ — ACX
$(C)^+ = CX$ — AD ✓
$D^+ = DB$ X — BC ✓
BD X
CD ✓

ACX
BDX

✗ ACB
✗ A CD
✗ BDA
✗ BDC



AB → C
C → D
D → BE
E → F.
F → A.

AX — AB ✓
BX — AE X
C ✓ — Af X
D ✓ — BE ✓
EX — Bf ✓
FX — Ef X.

Orion

PAGE:
DATE: / /

Insertion, Deletion and Updation Anomalies in DBMS

Student_info.

S_id	name	age	Br_code	Br-name	HOD
1	A	18	101	CS	XYZ
2	B	19	101	CS	XYZ
3.	C	18	101	CS	XYZ
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6	F	19	103	ME	KLM

Idea:-
In the table student_info we have tried to store entire data about student.

Result:-
Entire branch data of a branch must be repeated for every student of the branch.

Redundancy:-
When same data is stored multiple times unnecessarily in a database.

Disadvantages:-
(i) Insertion, deletion and modification anomalies.
(ii) Inconsistency (data)
(iii) Increase in database size & increase in time (space)

Insertion anomalies:-
When certain data (Attribute) can't be inserted into Database, without the presence of other data.

Deletion anomalies:-
If we delete some data (unwanted) cause deletion of some other data (wanted)

Orion
PAGE:
DATE: / /

Updation/Modification anomalies :-

When we want to update a single piece of data but it must be done at all

Student_info

S_id	name	age	Br_code	Br_name	HOD
1	A	18	101	CS	X42
2	B	19	101	CS	X42
3	C	18	101	CS	X42
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6.	F	19.	103.	ME	KLM

Kisi "Table" ko decompose karna

Student_info

S_id	name	age	Br_code
1	A	18	101
2	B	19	101
3	C	18	101
4	D	21	102
5	E	20	102
6	F	19	103.

Branch_info.

Br_code	Br_name	HOD_name.
101	CS	X42
102	EC	PQR
103.	ME	KLM.

Orion
PAGE:
DATE: / /

- As one paragraph contains a single idea similarly one table must contain direct and main data about an Entity.

- Normalization (decomposition of table) of table is done of the basis of functional dependencies.

1NF :- (atomic value).

Rollno	name	course
101	Modi	CN
		OS
102	Sonia	DBMS
		CO

101	Modi	CN
101	Modi	OS
102	Sonia	DBMS
102	sonia	CO

- every cell contains atomic values

2NF :-

$R(A\,B\,C\,D)$ $\quad (AB)^+ = ABCD$.

$AB \leftarrow$ C.K.

A,B ← prime attribute.

C,D ← non prime attribute.

$B \rightarrow C$ (partial dependency) ***

B → non-prime attribute

Only a part of candidate key.

A	B
1	—
—	2
—	—
3	4

ANS :-

$R_1(A\,B\,D)$

$R_2(B\,C)$

Orion
PAGE:
DATE: / /

$R(ABC)$

$B \rightarrow C$

$(AB)^+ = ABC$.

$A, B \leftarrow P$

$C \leftarrow$ N.P.

$P \rightarrow$ N.P (Partial dependency)

$B \rightarrow C$.

R

A	B	C
a	1	x
b	2	y
a	3	z
c	3	z
d	3	z
e	3	z.

R_1

A	B.
a	1
b	2
a	3
c	3
d	3
e	3

R_2

B	C
1	x
2	y
3	z.

1. $R(ABCDE)$.

$AB \rightarrow C$

$D \rightarrow E$.

$R(AB\ CD\ E)$.

$(ABD)^+ = R$.

**(partial dependency).

$R_1(ABC)$

$R_2(DE)$

$R_3(ABD)$

2. $R(A\ B\ C\ D\ E)$

$(AC)^+ = R$

(P.D) $A \rightarrow B$

$B \rightarrow E$

(P.D) $C \rightarrow D$

$R_1(A\ B\ E)$

$R_2(C\ D)$

$R_3(AC)$

Orion
PAGE:
DATE: / /

3. R(ABCDEFGHIJ)

FD $AB \rightarrow C$

FD $AD \rightarrow GH$ $\quad (ABD)^+ = R$.

FD $BD \rightarrow EF$

FD. $A \rightarrow I$

$H \rightarrow J$.

$R_1(ABC)$

$R_2(AI)$

$R_3(ADGHJ)$

$R_4(BDEF)$.

$R_5(ABD)$.

Transitive dependency :-
A FD from $\alpha \rightarrow \beta$ is called transitive if α, β ∉ non-prime.

3NF :- A relation R is in 3NF if
a) it is in 2NF
b) no transitive dependency.

every dependency from $\alpha \rightarrow \beta$.
(i) Either α is superkey.
(ii) or β is a prime attribute.

P.D
$P \rightarrow NP$.
T.D
$NP \rightarrow NP$.

R(A B C)

$A \rightarrow B$

$B \rightarrow C$.

A	B	C
a	1	x
b	1	x
c	1	x
d	2	y
e	2	y
f	3	z
g	3	z

A	B
a	1
b	1
c	1
d	2
e	2
f	3
g	3

B	C
1	x
2	y
3	z

Orion
PAGE:
DATE: / /

3. $R(ABCDE)$ $R(\underbrace{A}\ \underbrace{B}\ \underbrace{C}\ \underbrace{D}\ E)$

$AB \rightarrow C$ × $A \rightarrow B$.
$B \rightarrow D$. × $B \rightarrow E$.
$D \rightarrow E$. ✓ $C \rightarrow D$

$(AC)^+ = ABCDE$.

ABCDE

- $R_1(\underbrace{A}\ \underbrace{B}\ E)$
  - $R_{11}(\underbrace{A}\ B)$
  - $R_{12}(\underbrace{B}\ E)$
- $R_2(\underbrace{C}\ \underbrace{D})$
- $R_3(AC)$

4. $R(ABCDEFGHIJ)$

✓ $AB \rightarrow C$
× $A \rightarrow DE$.
$B \rightarrow F$.
$F \rightarrow GH$.
$D \rightarrow IJ$.

Candidate Key :-
$\therefore (AB)^+ = R$.

- $R_1(\underbrace{A}\ \underbrace{D}\ E\ \underbrace{I}\ J)$
  - $R_{11}(\underbrace{A}\ \underbrace{D}\ E)$
  - $R_{12}(\underbrace{D}\ \underbrace{I}\ J)$
- $R_2(\underbrace{B}\ \underbrace{F}\ \underbrace{G}\ \underbrace{H})$
  - $R_{21}(\underbrace{B}\ \underbrace{F})$
  - $R_{22}(\underbrace{F}\ \underbrace{G}\ \underbrace{H})$
- $R_3(\underbrace{A}\ \underbrace{B}\ \underbrace{C})$

Orion
PAGE:
DATE: / /

BCNF :-
(Boyce Codd Normal Form)

2NF. $\alpha \xrightarrow{\times} B$
prime. no prime

3NF $\alpha \xrightarrow{\times} B$
n.p n.p

$\alpha \longrightarrow \beta$.
P/NP P

$R(A\,B\,C)$

$AB \rightarrow C$.
$C \rightarrow B$.

$(A)^+$
$(AB)^+$ ✓
$(AC)^+$ ✓

$\alpha \longrightarrow \beta$.
P/NP P

$\alpha \longrightarrow B$
S.K.

AB
AC

$R(ABC)$
$AB \rightarrow C$
$C \rightarrow B$.
$\alpha \rightarrow B$.

α must be a superkey

A	C
a	x
b	y
c	z
c	w
d	w
e	w

C	B
x	1
y	2
z	2
w	3

R

A	B	C
a	1	x
b	2	y
c	2	z
c	3	w
d	3	w
e	3	w

- $R_1(CB)$
- $R_2(AC)$
- $R_3(AB)$

Orion
PAGE:
DATE: / /

1. R(ABCDEFGH).

AB → C
A → DE.
B → F
F → GH.
(AB).

BCNF.
~~3NF.~~
~~2NF.~~
(1NF).

2. R(ABCDE).

CE → D.
D → B.
C → A
(CE).

~~BCNF.~~
~~3NF~~
~~2NF.~~
(1NF)

WF

3. R(ABCDEF)

AB → C
DC → AE.
E → F.
(ABD) (BCD).

4. R(ABCDEFGH) #1

AB → C
BD → EF.
AD → GH.
A → I
(ABD).

3NF

1NF.

5. R(ABCDE).

~~AB → CD~~
D → A → prime attribute
BC → DE.
(AB), (BD), (BC)

6. R(ABCDE)

BC → ADE.
(D) → (B).
(BC), (CD).

Candidate key || prime attribute.

Orion

PAGE:
DATE: / /

7. R(VWXYZ)

CK: X → Y V PA

Y → Z

Z → Y

VW → X

VW, XW

INF

8. R(ABCDEF)

ABC → D

ABD → E

CD → F

CDF → B

BF → D

ABC, ACD

INF

9. R(ABC)

A → B

B → C

C → A

A, B, C

BCNF

Orion

PAGE:
DATE: / /

How to find identify normal form in dbms.

1. R(ABCDEF)
   A → BCDEF.
   BC → ADEF
   DEF → ABC.
   A, BC, DEF.
   BCNF.

2. R(ABC)
   AB → C
   C → A.
   AB, BA.
   3NF.

3. R(ABCDE)
   A → B.
   BC → E.
   DE → A
   ACD, BCD, CDE
   3NF.

4. R(ABCDE).
   AB → CD.
   D → A.
   BC → DE.
   AB, BC, BD.
   3NF.

5. R(WXYZ)
   Z → W.
   Y → XZ.
   XW → Y.
   Y, XW, XZ.
   3NF.

6. R(ABCDE)
   A → B n.p.
   B → E.
   C → D.
   AE.
   1NF.

7. R(ABCDEF).
   AB → C
   DC → AE.
   E → F.
   ABD, BCD.
   1NF.

8. R(NWXYZ).
   Z → Y
   Y → Z
   X → YV → n.p.
   VW → X
   VW, XW.
   1NF.

Subset

9. R(ABCDEF)
   ABC → D.
   ABD → E.
   CD → F n.p.
   CDF → B.
   BF → D.
   ABC, ACD.
   1NF.

Orion
PAGE:
DATE: / /

Lossless Join Decomposition / Non-additive

Tuple generation problem doesn't occur after decomposition.

- It is a mandatory property must always hold good.
- If a relation R is decomposed into two relations R_1 & R_2, then it will be lossless iff :-

(i) $attr(R_1) \cup attr(R_2) = attr(R)$.

(ii) $attr(R_1) \cap attr(R_2) \neq \phi$

(iii) $attr(R_1) \cap attr(R_2) \rightarrow attr(R_1)$.

or

$attr(R_1) \cap attr(R_2) \rightarrow attr(R_2)$.

R

A	B	C	D
1	a	b	x
2	b	q	y

R

R_1 R_2

R

R_1

A	B
1	a
2	b

R_2

C	D
b	x
q	y

$R_1 \times R_2$

A	B	C	D
1	a	b	x
1	a	q	y
2	b	b	x
2	b	q	y

Orion
PAGE:
DATE: / /

R

A	B	C
1	a	β
2	b	α
3	a	γ

R_1

A	B
1	a
2	b
3	a

R_2

B	C
a	β
b	α
a	γ

$R_1 \times R_2$

A	B	C
1	a	β
1	a	γ
2	b	α
3	a	β
3	a	γ

Prove it is candidate key in any of the table.

Orion
PAGE:
DATE: / /

If a table R having F.D set F, is decomposed into two tables R_1 and R_2 having F.D set F_1 and F_2

Then $F_1 \subseteq F^+$
$F_2 \subseteq F^+$
$(F_1 \cup F_2)^+ = F^+$

R(ABC)
$A \rightarrow B$.
$B \rightarrow C$
$C \rightarrow D$.
$R_1(AB)$ $R_2(BC)$.

R(ABCD)
$AB \rightarrow CD$.
$D \rightarrow A$
$R_1(AD)$ $R_2(BCD)$.

R(F)
$R_1(F_1)$ $R_2(F_2)$
$R(F_1 \cup F_2) = F$.

R(ABCD)
$F: AB \rightarrow CD$
$D \rightarrow A$.

$R_1(AB)$ $R_2(BC)$
$F_1: A \rightarrow B$ | $F_2: B \rightarrow C$
$B \rightarrow A$ | $C \rightarrow B$

$R_1(AD)$ | $R_2(BCD)$
$F_1:$ | $F_2:$
$D \rightarrow A$ | $BD \rightarrow C$.

$(F_1 \cup F_2)^+ = F^+$
$C^+ = CBA$

$(AB)^+ = AB$.

R(A B C D E F G H I J)

$R_1(A B C)$

$R_2(A D E I J)$

$R_3(B F G H)$

$R_{21}(A D E)$ 3NF.
$R_{22}(D I J)$

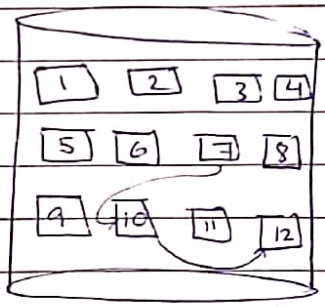
$R_{31}(B F)$ 3NF.
$R_{32}(F G H)$

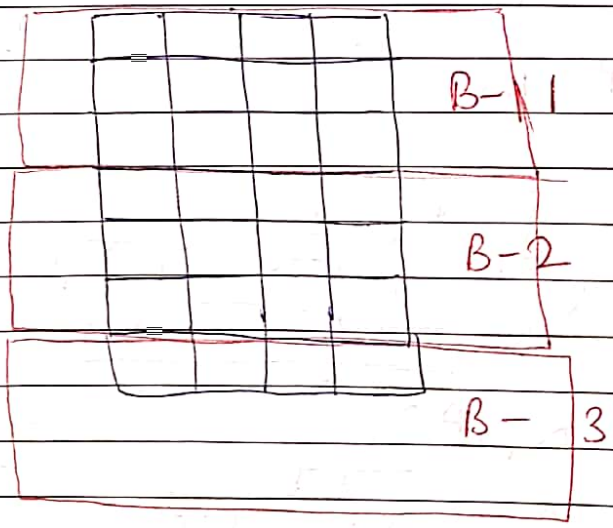
Orion
PAGE:
DATE: / /

File Structures

SRS
↓
ER-diagram.
↓
Relation model
↓
normalization.
↓
file structures
(binding of physical structures)

*** File → data structure



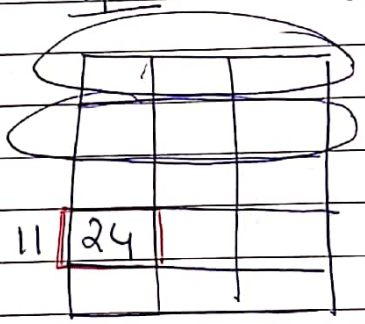


Orion
PAGE:
DATE: / /

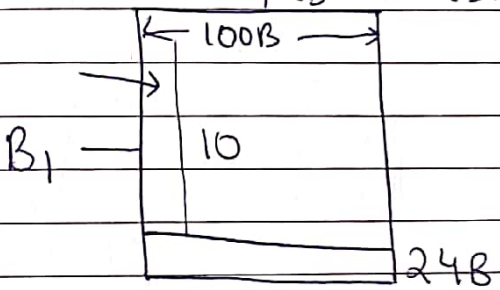
Sorted File	Unsorted File.
◦ Can be sorted acc. to one attribute (search key)	◦ Random order, any record can be placed anywhere
◦ Searching fast	◦ Searching will be slow.
◦ Insertion and deletion can be difficult.	◦ Insertion and deletion will be easy.

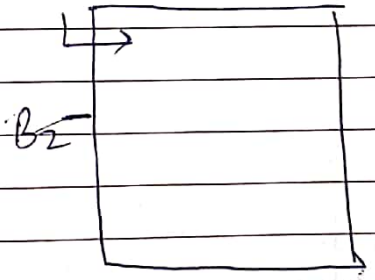
Spanned / Unspanned Mapping

2 part — Completely



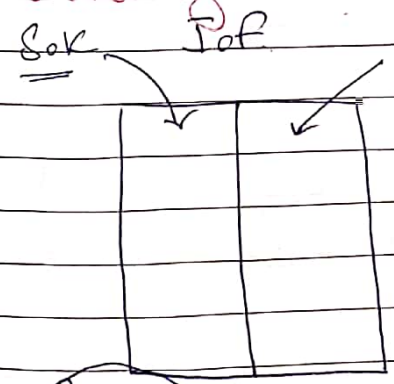
1 KB = 1024 B.





block → record
ignore

Indexing



Book
Index - ch. - pg no

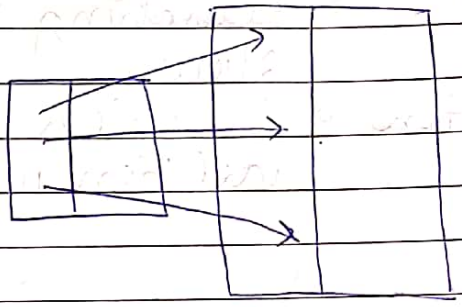
Index → Size small
sorted / unsorted

→ to access the file quickly.

Orion
PAGE:
DATE: / /

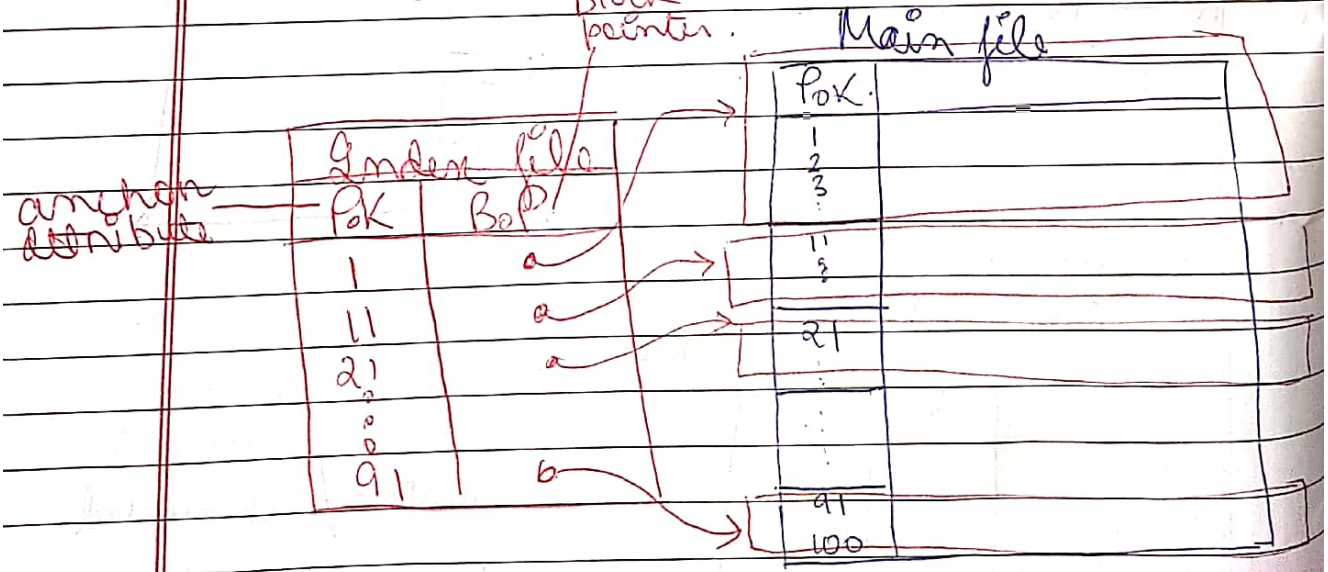
Types of indexing.

1. Primary indexing.
2. Clustered indexing.
3. Secondary indexing.
4. Multilevel indexing.



Primary indexing.

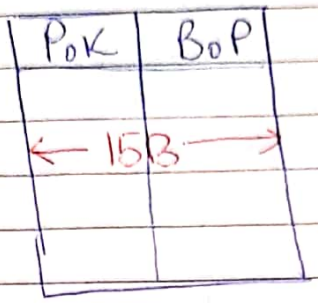
- Main file sorted.
- Primary key is used as anchor attribute (Search key).
- Example of sparse indexing
- no. of entries = no. of blocks acquired in index file (by the main file.
- no. of access required = $\log_2 n + 1$.



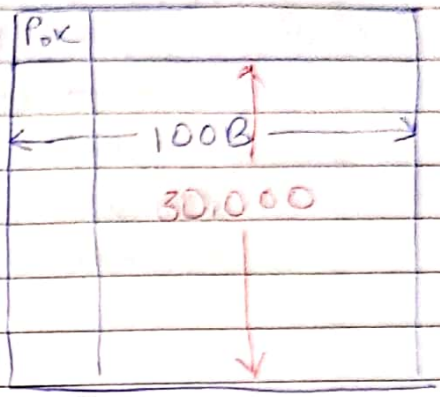
Orion
PAGE:
DATE: / /

Block size = 1024B

INDEX FILE



MAIN FILE:



Blocking factor = $\left[\frac{\text{Block size}}{\text{Record size}}\right]$

no. of Blocks for main file(n) = $\left[\frac{\text{no. of records in main file}}{\text{Blocking factor}}\right]$

no. of block access required = $\log_2 n$

Blocking factor = $\frac{1024B}{100B} = [10.24] = 10$.

no. of blocks for main file (n) = $\frac{30,000}{10} = 3000$.

no. of access read = $\log_2 h$
$= \log_2 3000$.
$= [11...] = 12$

INDEX FILE

Blocking factor = $\left[\frac{1024}{15}\right]$

no. of Blocks for index file = $\left[\frac{30,000}{68}\right]$
$= [44...]$
$= 45$.

no. of block access read = $[\log_2 n] + 1$
$= 6 + 1 = 7$.

Orion
PAGE:
DATE: / /

clustering index.

- Main file is sorted (on some non-key attribute).
- there will be one entry for each unique value of the non-key attribute.
- If number of block acquired by index file is n, then block acces required will be
  $\geq \log_2 n + 1$.

Index file

No.K	B.P.
1	
2	
3	
4	
5	

Main file

No.K	
1	B1
1	
2	
3	
3	B2
3	
4	
4	
5	
.	
100	

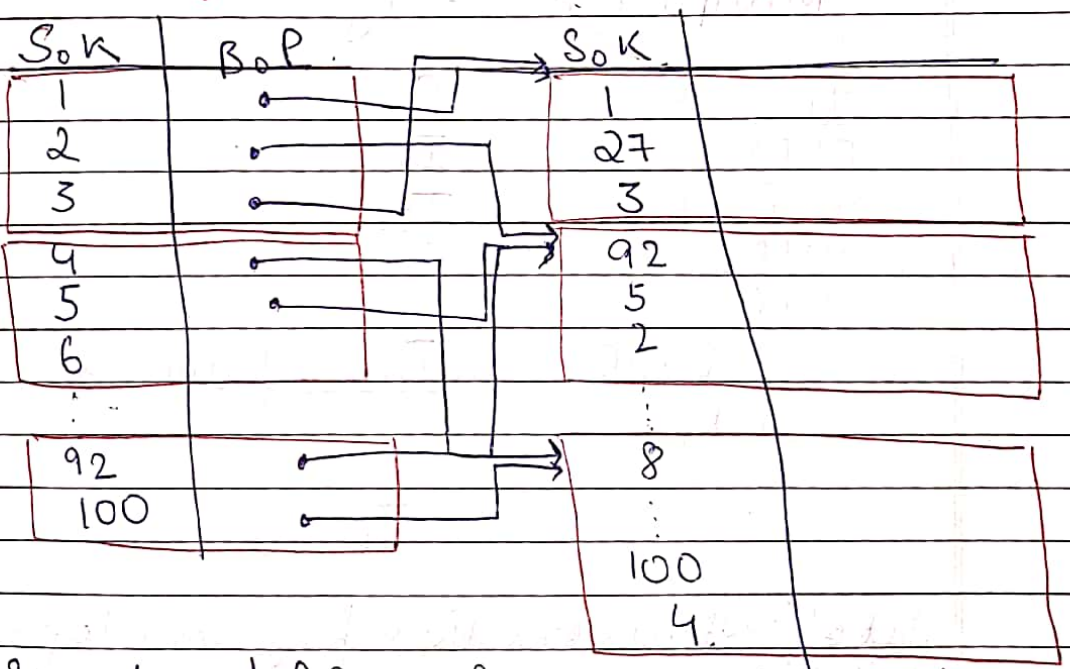
Orion
PAGE:
DATE: / /

Secondary Indexing :-

- Main file Unsorted.
- Can be done on key as well as on non-key attribute.
- Called secondary because normally one indexing is already done

[no of entry in index file = no of entry in main file]

- no of block access = $\lceil \log_2 n \rceil + 1$.



Secondary & Primary index can't be compared since in primary list is sorted.

Transaction.

What is transaction
What are acid properties.
Transaction states.
Problem with concurrency.
Conflict and view serializability.
Recoverability and cascadlessness.
Transaction designing protocols.
- Time stamping.
- Lock based protocols.
2-phase locking.
Graph based protocol.
Multiple granularity.

T_1	P
R(A)	T_1
A = A - 10.	T_2 ←
W(A)	:
R(B)	:
B = B + 10.	T_{60} failure x
W(B).	:
	:
	T_{n+100}.

Set of instructions that perform a logical unit of work - Transaction.

Transaction:-
1. Atomicity
2. Consistency.
3. Isolation.
4. Durability

T ⟶ D.B.

Orion
PAGE:
DATE: / /

- DB can be managed by transaction
- Either all the instructions within a transactions will be executed or neither of them will be executed.

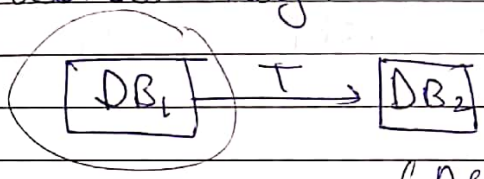
Atomicity → Transaction management component.

Isolation
If working of one transaction doesn't affect other transactions
Concurrency control component.

Durability
If a change has occured, the change will stay.
Recovery management component.

Consistency.

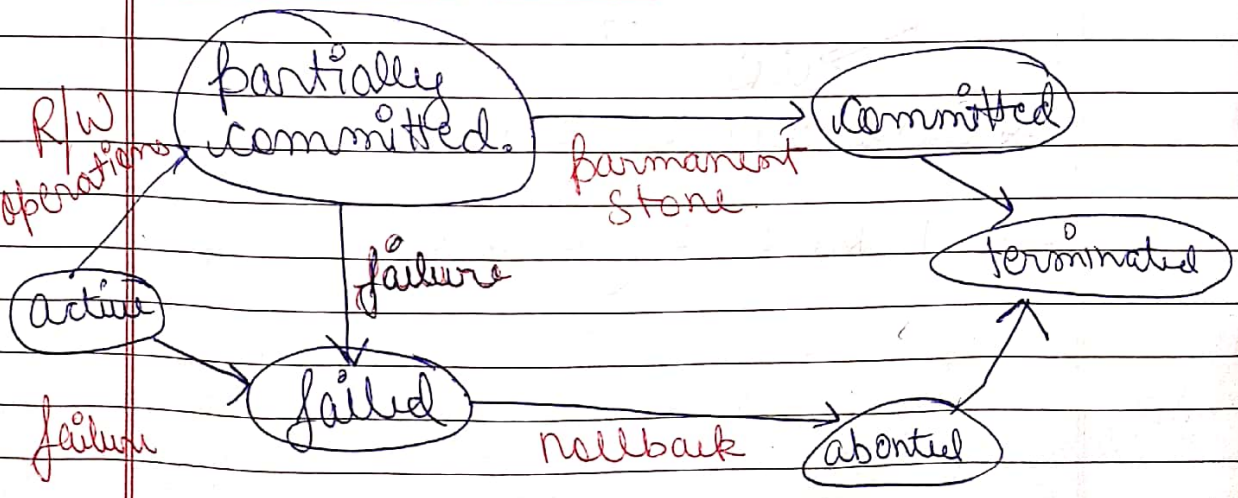
If these prob held good =



Consistent. (After transaction it will be consistent)

A+B = A+B

Transaction States



e.g Xerox photocopy.

Orion
PAGE:
DATE: / /

Initial state → active

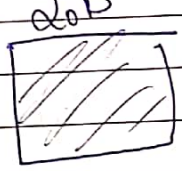
When the transaction is executing in terms of instructions { I_1, I_2, ⋮, I_n

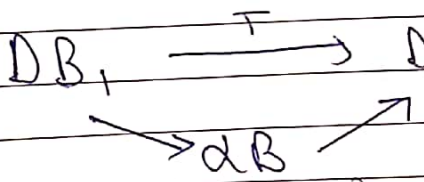
When instruction is running ⇒

There may be a failure (failed state)

On we will mark the partially committed state after performing all the read/write operations.

We do the changes in a copy. (local buffer)
(Because the data is not permanently stored in memory)





* Consistent in both cases.

e.g Ticket booking & cancelling.

Orion
PAGE:
DATE: / /

Advantages of concurrency.
→ If we can run more than one transaction in a system at a time it is called concurrency. interleaved execution of more than one transaction.
1) Waiting time. ↓
2) Response time. ↓
3) Resource utilization. ↑
4) Efficiency. ↑

Dirty Read Problem

T_1	T_2
R(A)	
W(A)	
C	R(A)
	C.

→ two students giving exams & one cheating from the other one. One submitting the paper early. data the T changed ans.

Unrepeatable Read problem:-

$X = 10$

T_1	T_2
R(X)	
	R(X) — 10
X++ W(X)	
	R(X) — 11

(child. bhoot
maggie
→ rajma channal.

Phatom Read Problem.

$X = 10$

T_1	T_2
R(X)	
	R(X)
Delete(X)	
	R(X)

do $X = 10$

this same story
thief

Because a transaction is like independent doesn't depend on other transaction. But here it does.

Orion
PAGE:
DATE: / /

Lost update problem (write write conflict)

	T_1	T_2
DB A = ~~10~~ 50	R(A)	
	W(A)	
		W(A)
		C

→ conflict b/w these two writes

DB A = ~~10~~ ~~11~~ 50. C

* Serial Schedule and non-Serial schedule in DBMS.

Schedule:

S1

T_1	T_2
	R(B)
	W(B)
	R(A)
	W(A)
R(A)	
W(A)	
R(B)	
W(B)	

consistent

Serial schedule

T_1	T_2
R(A)	R(B)
W(A)	W(B)
R(B)	R(A)
W(B)	W(A)
n_1	n_2

$n_1 + n_2$.

S2

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
R(B)	
	R(A)
W(B)	
	W(A)

non-serial schedule.

$T_1, T_2, T_3 \ldots T_n$

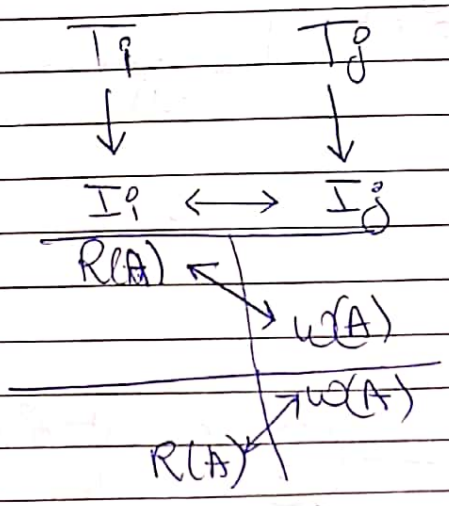
$n \times (n-1) \times (n-2) \ldots \times 1$

Total no. of Non Serial Schedule $= \frac{(n_1 + n_2 + \ldots + n_n)!}{n_1!\, n_2! \ldots n_n!} - n!$

Orion

PAGE:
DATE: / /

- If we can prove non serial ≡ serial then it is consistent.
- When instruction belong to same data item conflicts happen.



no changes in TA1, TA2 & DB after swap.

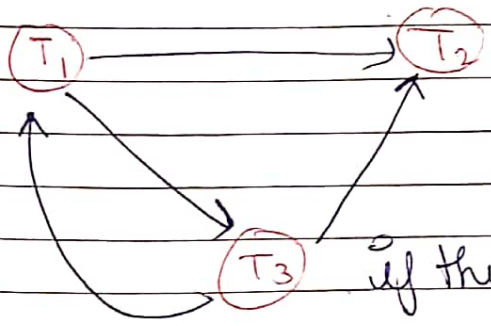
9/2/2022

Conflict Serializable Schedule

S

T_1	T_2	T_3
R(x)		
		R(z)
		W(z)
	R(y)	
R(y)		
	W(y)	
		W(x)
	W(z)	
W(x)		

1st → 2nd



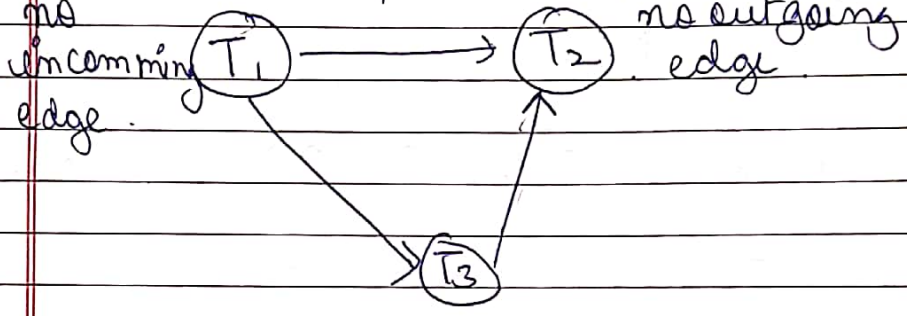
if there is a cycle

"NOT" a conflict serializable schedule

Orion
PAGE:
DATE: / /

S

T_1	T_2	T_3
R(X)		
	R(Y)	
		R(Y)
	W(Y)	
W(X)		
		W(X)
	R(X)	
	W(X)	



Cost of finding = $O(n^2)$

$n \rightarrow$ no of transaction

$T_1 \rightarrow T_3 \rightarrow T_2$

S

T_1	T_2	T_3	T_4
	R(X)		
		W(X)	
		C	
W(X)			
C			
	W(Y)		
	R(Z)		
	C		
			R(X)
			R(Y)
			C

check till it is committed

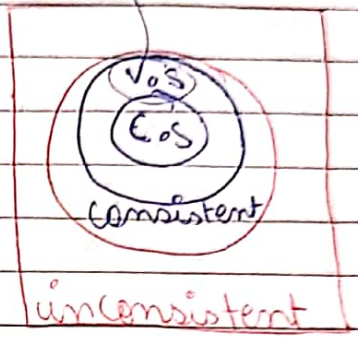
when one of its view serial is view equivalent

Orion

PAGE:
DATE: / /

View Serializability

If it is not C.S
but it is V.S
it will have "Blind write"



S

T_1	T_2
R(a)	
W(a)	
	R(a)
	W(a)
R(b)	
W(b)	
	R(b)
	W(b)

S

T_1	T_2
R(a)	
W(a)	
R(b)	
W(b)	
	R(a)
	W(b)
	R(b)
	W(b)

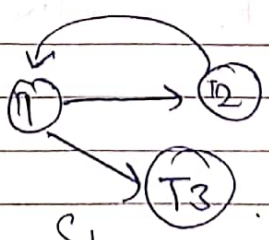
1) T_1 a T_1 } Initial Read same
T_1 b T_1 }

2) Final write same.

3) Intermediate read same.

S

T_1	T_2	T_3
R(a)		
	W(a)	
W(a)		
		W(a)



S_1 1 2 3
S_2 1 3 2
2 1 3
2 3 1
S_6 3 1 2
3 2 1

S_1

T_1	T_2	T_3
Ra		
W(a)		
	W(a)	
		(W a)

Orion
PAGE:
DATE: / /

Recoverable Schedule :-

S

T_1	T_2
10 - R(A)	
A = A+10	
20 - W(A)	
	R(A) — (20)
	A = A - 5 (15)
	W(A)
C	C
C	C → recoverable

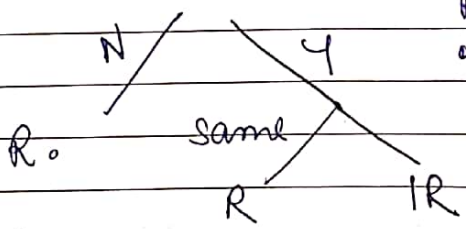
D.R (Dirty Read) when a transactions reads the uncommitted changes of another transaction.

$$A = \cancel{10}\ 15$$

$$\text{If } B \quad A = \cancel{10}\ 10$$
$$\cancel{20}$$
$$15$$

we roll backed T_1 but not T_2.
when T_1 was rolling back we wanted T_2 to roll back as well.

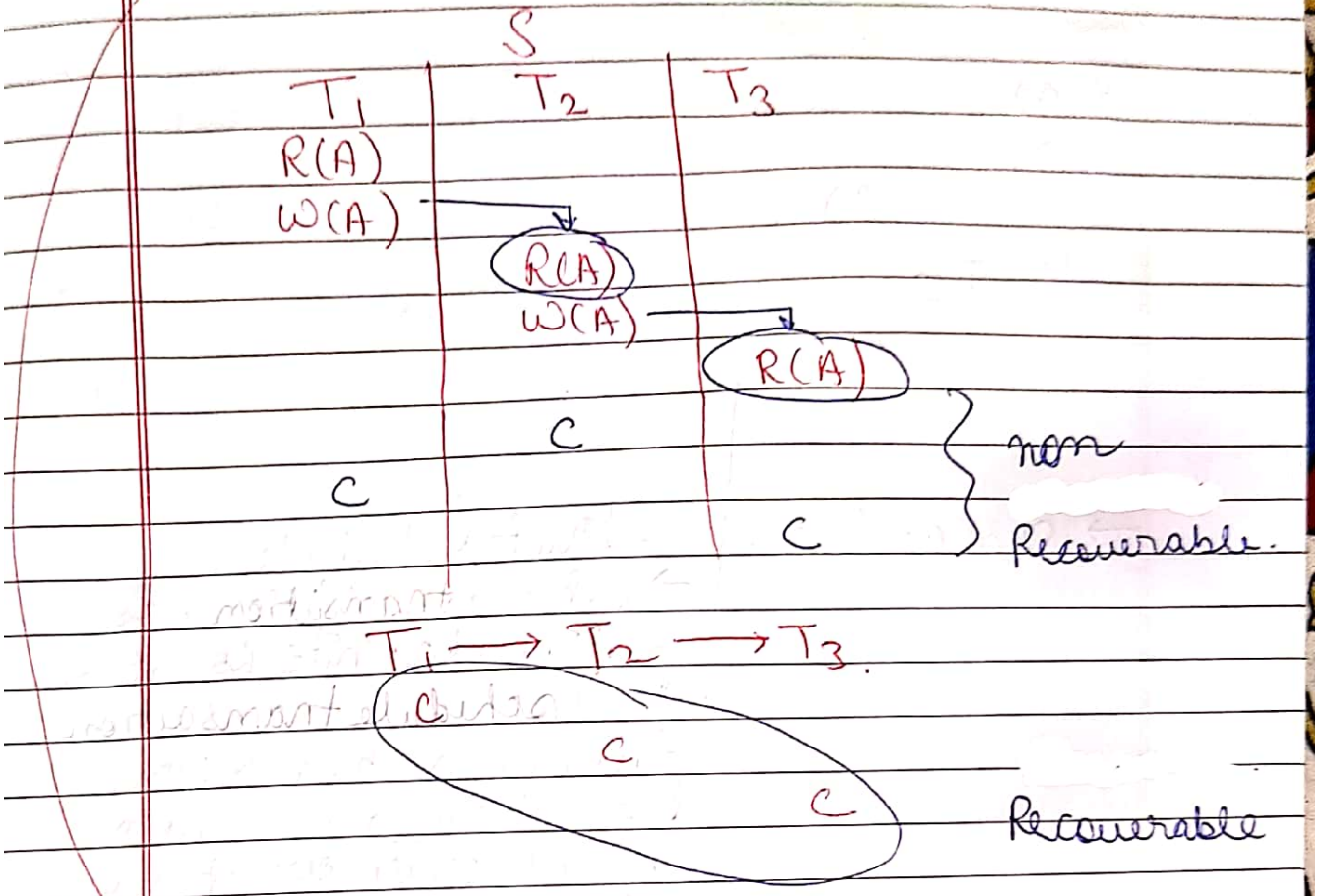
Dirty Read

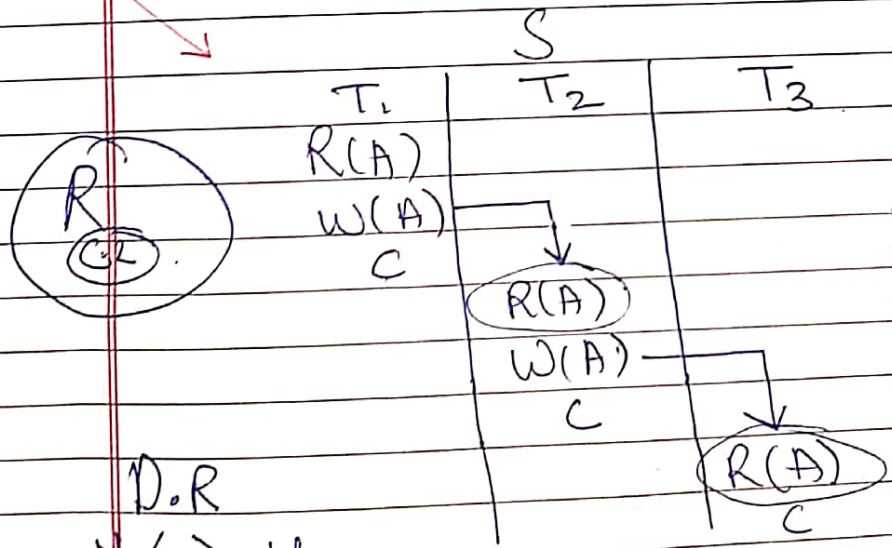


Recoverable →
- the order dirty read = the order of commit.

Orion
PAGE:
DATE: / /

Cascadless schedule :- (no cascading rollbacks)





D.R

N / Y

CL, R CLx

same / same x

R IR

Orion
PAGE:
DATE: / /

Strict Schedule

S1

T_1	T_2
R(a)	
W(a)	
	W(a)
C	
	R(a)
	C

S2

T_1	T_2
R(a)	
W(a)	
C	
	W(a)
	R(a)
	C

S3

T_1	T_2
R(a)	
	R(b)
W(a)	
	W(b)
C	
	R(a)
	C

→ cascadeless.

Strict schedule
→ when a schedule/transaction has performed a write operation on a data item then no other transaction can perform a read/write operation on that data item.

Orion
PAGE:
DATE: / /

Concurrency Control techniques:-

- Till now we already know how to check whether a schedule will maintain the consistency of DB or not (CoS, VoS, Recoverability, Cascadeless, strict etc)
- Now will study protocols that guarantees to generate schedule which satisfy these properties specially (CoS).
- For CoS we must avoid conflicting instructions. We remember three conditions of conflicting instruction.
- Actual problem is different transaction trying to access data at same time.
- How to approach conflict serializability.
  - time stamping protocal (Ordering)
  - lock based Protocal (lock-unlock)
    - ↳ 2PL (Basic, conservative, strict, rigorous).
    - ↳ Graph based.
  - Validation Protocal.
- Goals:-
  Concurrency, properties, time, logic.

Orion
PAGE:
DATE: / /

Time Stamp Ordering Protocal :-

- Basic idea of Time stamping is to decide (the order) between the transactions by one that enters into the system. So that in case of conflict during execution, we can resolve the conflict using Ordering.
- the reason we call time-stamp not stamp because, for stamping are the values of system clock. (as it will always be unique can never repeat itself)

Two ideas of time stamping :-

- Time stamp with transaction :- with each transaction T_i, we associate a time-stamp denoted by $T.S(T_i)$.
- It is the value of the system clock when a transaction enters into the system, so if a new transaction T_j enters after T_i then $T.S(T_i) < T.S(T_j)$ always unique, will remain fixed through the execution
- also determine serializability order if $T.S(T_i) < T.S(T_j)$ then, system ensure that in the resultant C.S.S T_i will execute first before T_j.

Orion
PAGE:
DATE: / /

Time stamp with data item :-
For each data item Q protocal maintains two time-stamps.

W-timestamp(Q) :-
is the largest time-stamp of any transaction that executed write(Q) successfully.

R-time stamp(Q) :-
is the largest timestamp of any transaction that executed read(Q) successfully.

Ti request for Read(Q) :-

if T.S(Ti) < W.T.S(Q),
means Ti needs to read a value of Q that was already overwritten.
Hence request must be rejected & Ti must rollback

if T.S(Ti) ≥ W.T.S(Q)
Operation can be allowed and R.T.S(Q) will be max(RTS(Q), TS(Ti))

Ti request for Write(Q)
if T.S(Ti) < RTS(Q)
means value of Q that Ti is producing was needed previously and the system assumed that the value would never be produced
Hence reject & rollback.
if T.S(Ti) < WTS(Q)
Ti is attempting to write an obsolete value of Q reject and Rollback
Otherwise
WTS(Q) = max(WTS(Q), TS(Ti))

S

T_0 S(TP)=5	T_0S(Tx)=10.
R(Q)	
	W(Q)
R(Q)	

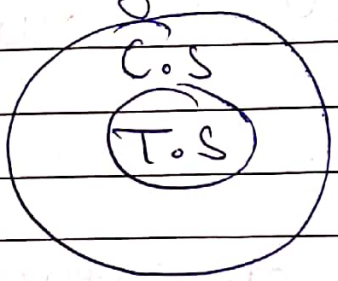
TS(T_P)=10	T_0S (Tx)=5.
	W(Q)
R(Q)	

junior/ professor doing some activity outside college gate.

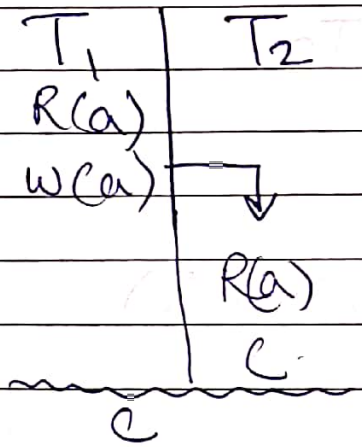
Orion
PAGE:
DATE: / /

Properties of Time Stamping Protocal :-

- It ensure conflict serializability.
- It ensure view serializability.
- possibility of dirty read no restriction on commit, irrecoverable schedule and cascading rollbacks are possible.
- here either we allow or reject so no idea of deadlock.
-



- may suffer from starvation relatively slow.



Orion
PAGE:
DATE: / /

Thomas Write Rule :-

Modify time stamping protocal to make some improvements and may generate those schedules that are V.S but not C.S and provide better concurrency.

- It modify time-stamping protocal in absolute write case when Ti request write(Q) if TS(Ti) < WTS(Q).
- here Ti attemps to write absolute value of Q rather than rolling back Ti, Write operation is ignored.

T_1	T_2	T_3
R(Q)		
	W(Q)	
×W(Q)		
		W(Q)

RTS(Q) = 1.
WTS(Q) = 2.

Orion
PAGE:
DATE: / /

Q = 10 14.
Q = 10 12 14.
T_i = 12.
T_x = 14.

T.S(T_i) = 5	T.S(T_x) = 10
W(Q)	
	W(Q)
~~W(Q)~~	

Lock Based Protocal :-

- To achieve consistency isolation is the most important idea. locking is simplest idea to achieve isolation i.e. first obtain a lock on a data item then perform a desired operation and then unlock it.
- To provide better concurrency along with isolation we use different modes of locks.

Shared Mode :-
Denoted by lock S(Q) transaction can perform read operation, any other transaction can also obtain same lock on same data item at same time (so called Shared)

Exclusive mode :-
Denoted by lock - X(Q) transaction can perform both Read/write operations, any other transaction cannot obtained either Shared/exclusive mode lock.

Orion
PAGE:
DATE: / /

Compatability table.

	Shared	exclusive
Shared	T	F
exclusive	F	F

Properties of Lock Based Approach

S

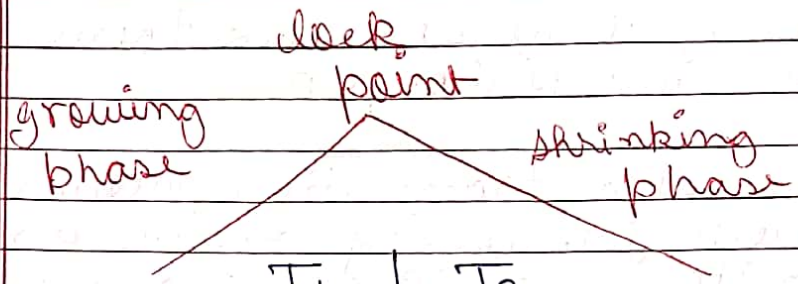
T_1	T_2
lock-X(A)	
R(A)	
W(A)	
Unlock(A)	
	lock S(B)
	R(B)
	Unlock(B).
lock-X(B)	
R(B)	
W(B)	
Unlock(B)	
	lock-S(A)
	R(A)
	Unlock(A)

Orion
PAGE:
DATE: / /

- if we do unlocking inconsistency will arise, if we do not unlock them concurrency will be poor.
- we require that transaction follow some set of rules for locking and unlocking of data item e.g. 2PL or graph based.
- We say a schedule is legal under a protocal if it can be generated using the rules of the protocals.

Two-Phase locking Protocal (2PL) :-
Basic 2PL.

- This protocal requires that each transaction in a schedule will be two phased - Growing phase & Shrinking phase.
- In growing phase - transaction can only obtain locks but can not release any lock.
- In Shrinking phase - transaction can only release locks but can not obtain any lock.
- transaction can perform read/write operation both in growing / shrinking phase.
- Ensure CoS/VoS the Order of serializability is the order in which transaction reaches lock point.
- Do not ensure freedom from deadlock.



T_1	T_2
lock-S(A)	
Read (A)	
	lock-X(B)
	Read (B)
	Write (B).
lock S(B)	
	lock X(A)

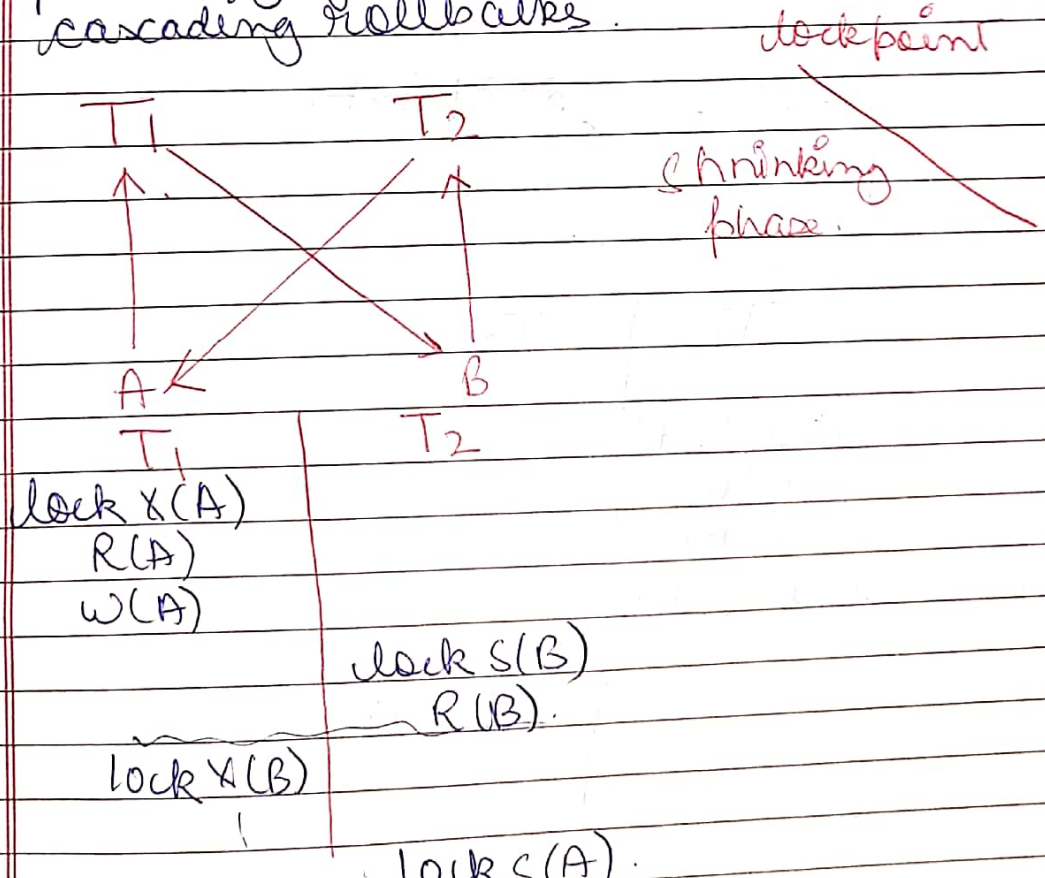
S

T_1	T_2
lock X(A)	
R(A)	
W(A)	
unlock X(A)	
	lock S(A)
	R(A)
	C

Orion
PAGE:
DATE: / /

Conservative / Static 2PL:-

- There is no growing phase transaction first will acquire all the locks need and then directly will start from lock point.
- If all locks are not available then transaction must release the lock acquired so far and wait.
- Shrinking phase works as usual and transaction can unlock any data item.
- Here we must have all the knowledge that what data items will be required during execution.
- Cascading is independent from deadlock.
- possibility of irrecoverable schedules and cascading rollbacks.

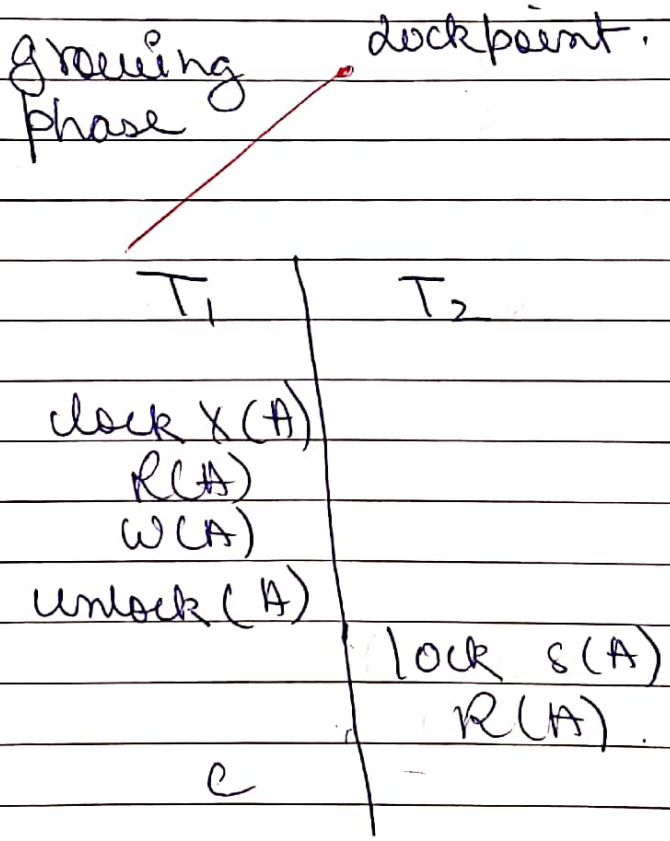


Orion
PAGE:
DATE: / /

Rigorous 2PL:

It is an improvement over 2PL protocol which will try to ensure recoverability and cascadelessness.

- Rigorous 2PL requires that all the locks must be held until transaction commits. i.e. there is no shrinking phase in the life of a transaction.
- will ensure CS, VS, recoverability, cascadelessness
- suffer from deadlock & inefficiency.

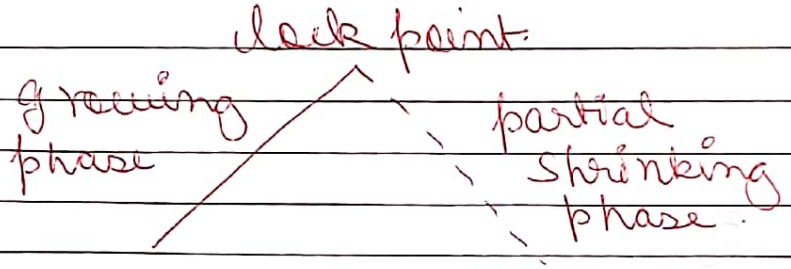


Orion
PAGE:
DATE: / /

Strict 2PL :-

It is an improvement over Rigorous 2PL.

- In the shrinking phase unlocking of exclusive locks are not allowed. But unlocking of shared locks can be done.
- All the properties are same as that of rigorous 2PL, but it provide better concurrency.

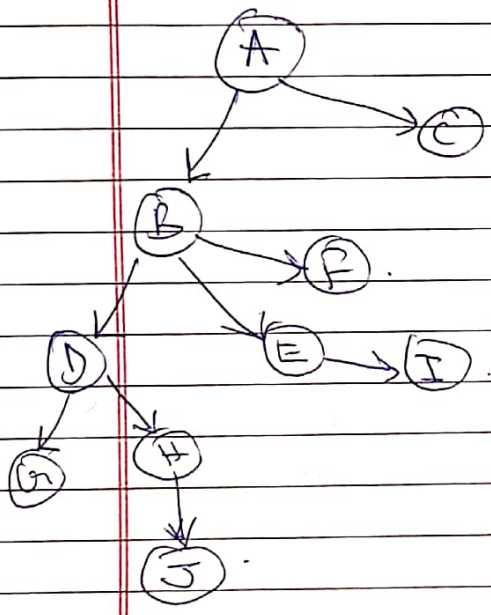


3) Graph based Protocol :-

- If we wish to develop lock based protocol that are not based on 2PL, we need additional info that how each transaction will access the data.
- There are various model that can give additional information each differing in the amount of info provided.
- Our idea is to take prior knowledge about the order in which the database items will be accessed.
- We impose partial ordering - On set of all data items $D = \{d_1, d_2 \ldots d_n\}$ if $d_i \rightarrow d_j$ then any transaction accessing both d_i and d_j must access d_i before d_j.
- P.O may be because logical or physical organisation or only because of concurrency control.

Orion
PAGE:
DATE: / /

- Also P.O set of all data items Duill know be viewed as directed acyclic graph called database graph.
- For sake of simplicity, we follow two restrictions:-
  - ↳ will Study graphs that are rooted trees.
  - ↳ will use only exclusive mode locks.



Orion
PAGE:
DATE: / /

Tree Protocal

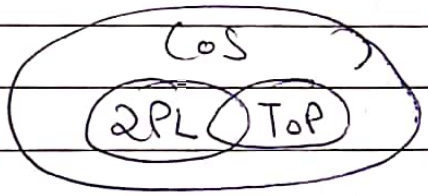
- Each transaction T_i can lock a data item Q will following rules:-
  - ↳ First lock by T_i may be on any data item.
  - ↳ Subsequently a data item Q can be locked by T_i only if the parent of Q is currently locked by T_i.
  - ↳ Data item may be unlocked at any time
  - ↳ Data item Q that has been locked and unlocked by T_i can not subsequently be relocked by T_i.

T_1	T_2
lock X(B)	lock (A)
lock X(D)	lock (B).
lock X(E)	lock (C)
unlock X(B).	lock (B)
lock X(I)	unlock (A).
unlock (D)	lock (D).
unlock (E)	unlock (B).
unlock (I).	lock (G).
	unlock (D).
	unlock (G)
	unlock (C).

Orion
PAGE:
DATE: / /

Properties of Tree Protocal :-

- All schedules that are legal under the tree protocal are C.S & V.S.
- Tree Protocal ensure freedom from deadlock.
- Tree protocal don't ensure recoverability and cascadelessness.
- Early unlocking is possible which leads shortest waiting time on increase concurrency
- A transaction may have to lock data items that it does not access lead to overhead waiting time and decrease in concurrency.
- transaction must know exactly what data items are to be accessed.



- Recoverability & cascadelessness can be provided by not unlocking before commit.

Orion
PAGE:
DATE: / /

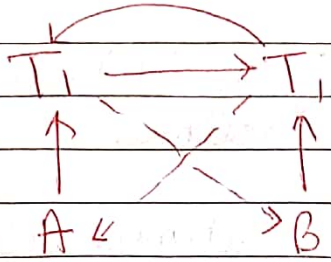
Deadlock handling

- A system is in deadlock state if there exists a set of transaction such that every transaction in the set is waiting for another transaction in the set.

- if there exists a set of waiting transaction $T_0, T_1, \ldots T_{n-1}$ such that
  $T_0 \rightarrow T_1, T_1 \rightarrow T_2, \ldots, T_{n-1} \rightarrow T_0$
  So, none transaction can progress in such situation.

- System must have a proper methods to deal with deadlocks, Otherwise
  - ↳ In real time system it may lead to life and money.
  - ↳ will reduce resource utilization and increase inefficiency.

- There are two principle for dealing with deadlock problem
  - Prevention:- which ensure that system will never enter a deadlock state.
  - Detection and Recovery:- allow system to enter into deadlock, then try to recover.



Orion
PAGE:
DATE: / /

Deadlock Prevention:-

- To ensure no hold & wait, each transaction locks all the data item before it begins execution eg C-2PL.
- To ensure no cyclic wait, impose an ordering of all data item and requires that a transaction lock data item only in the sequence consistent with ordering. e.g tree protocal.

↳ it is difficult to predict what data items are required.
↳ data item utilization will be low.
↳ ordering of data item may be difficult or time taking to follow.

Wait-die: (Non-preemptive)

Ti → Q ← Tj if TS(Ti) < TS(Tj) then Ti must wait (Ti is older).
if TS(Ti) > TS(Tj) then Ti rollback (die).

Wound-wait: (preemptive)

Ti → Q ← Tj if TS(Ti) > TS(Tj) then Ti can wait (Ti is younger).
if TS(Ti) < TS(Tj) then Tj rollback.

Lock-timeouts: a time is allotted for each transaction.