

INDEX

DBMS :-

DATA :-

Raw and isolated facts about an entity
(recorded)

e.g. text, audio, video, image, mob etc.

INFORMATION :-

Processed, meaningful, usable data

DATABASE :-

Collection of similar / related data

DBMS :-

SW 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

DBMS :-

OLAP

(online analytical processing)

OLTP

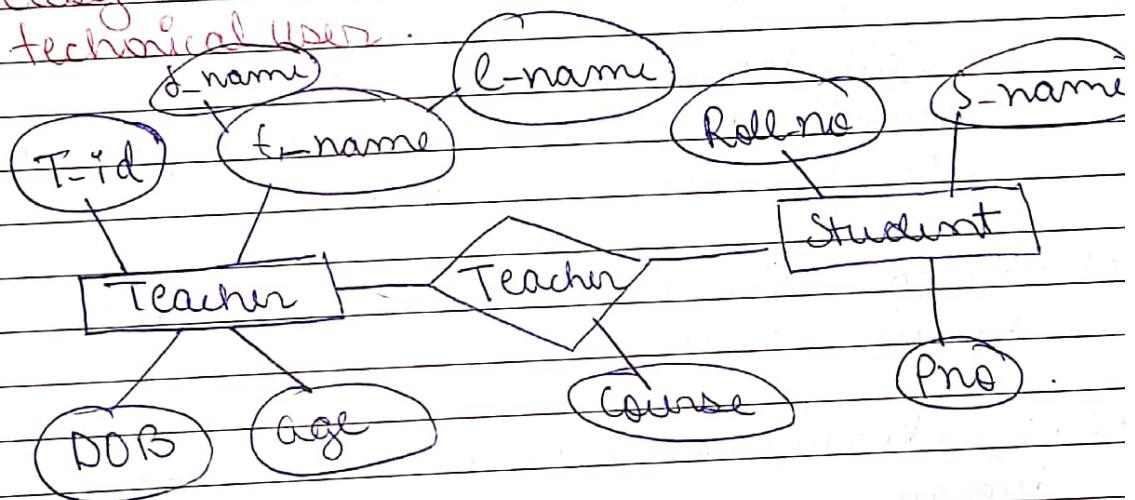
(online transaction processing)

- historical data.
- subject oriented.
- decision making
- TB, PB.
- CEO, MD, GM.
- R.
- current data
- application oriented
- day to day operations
- MB, brB.
- clerks, managers
- R/W

Entity Relationship Diagram

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

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



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

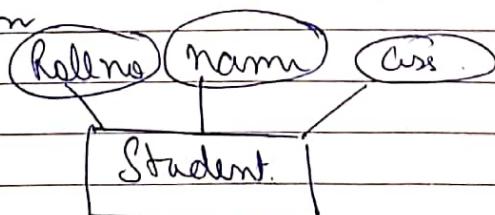
e.g. Account.

Entity Set :-

Collection / set of same types of entities i.e. that share same properties on attributes called entity set.

→ Entity can not 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 separate 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.

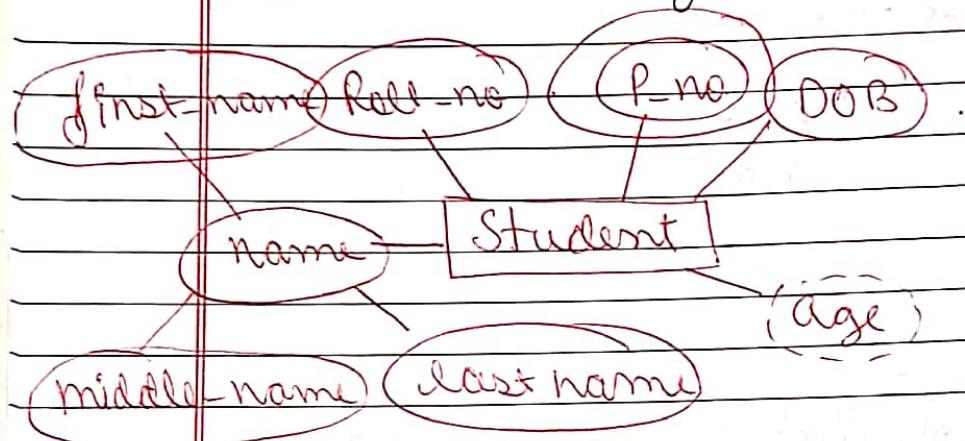
Simple - Multivalued

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

Stonid - derived

Stoned - have value is stored in the data base.

Derived - have value can be completed in run time using stoned attributes.



Student

Roll no.	f-name	M-name	d-name	DOB	Age
----------	--------	--------	--------	-----	-----

Roll-no	Pno				
---------	-----	--	--	--	--

Relationship :-

It is an association between two or more entities of same or different entity set.

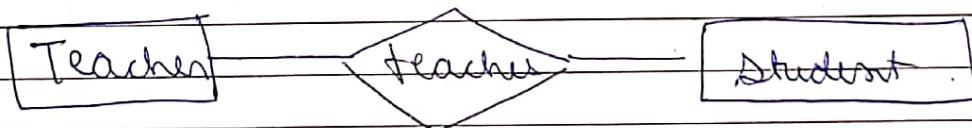
- No representation in ER diagram as it is an instance on data.
- In relational model represented either using a row in a table.

Relationship type / Set :-

A set of similar type of relationship

- In ER diagram represented using a diamond.

- In relational model either by a separate table or by separate column (foreignkey)
- Every relationship type has three components
(i) Name (ii) Degree (iii) Cardinality ratio / Participation constraints

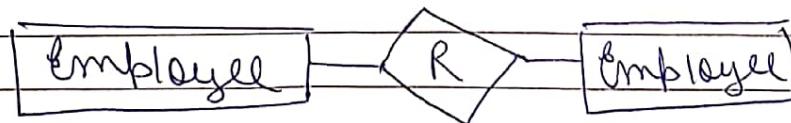
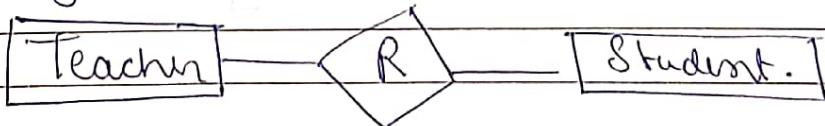


Degree of a relationship set :-

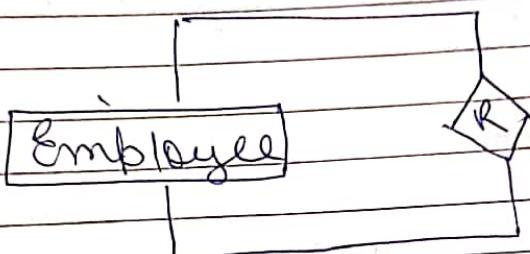
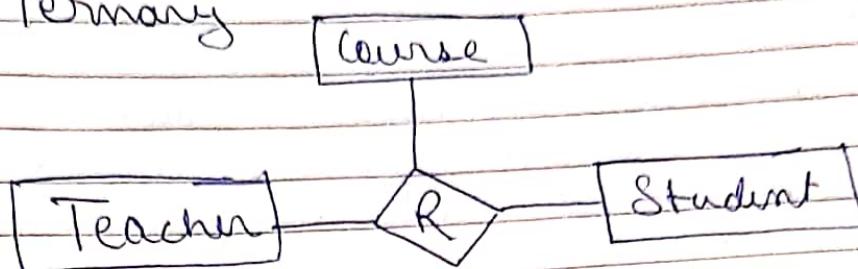
means number of entity set associated (participated) in the relationship set.

- Most of relationship sets in the ER diagram are binary (binarity). • Occasionally however relationship sets involve more than two entity sets.

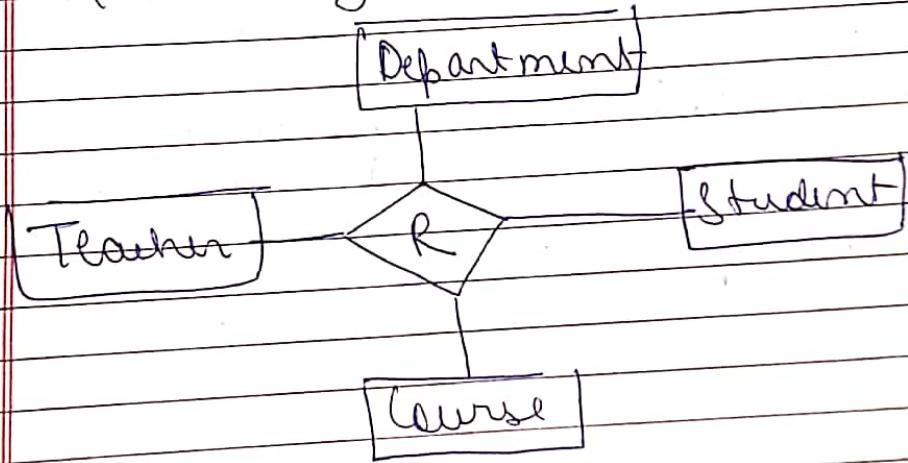
Binary :-



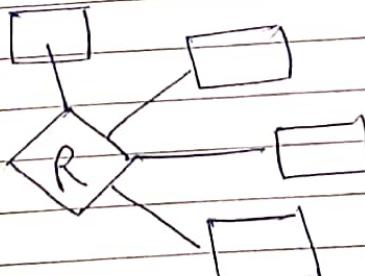
Ternary



Quaternary



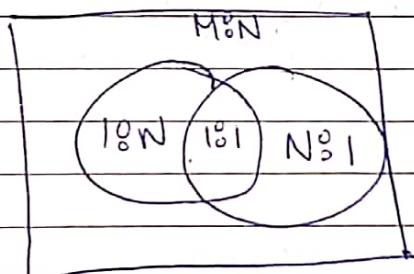
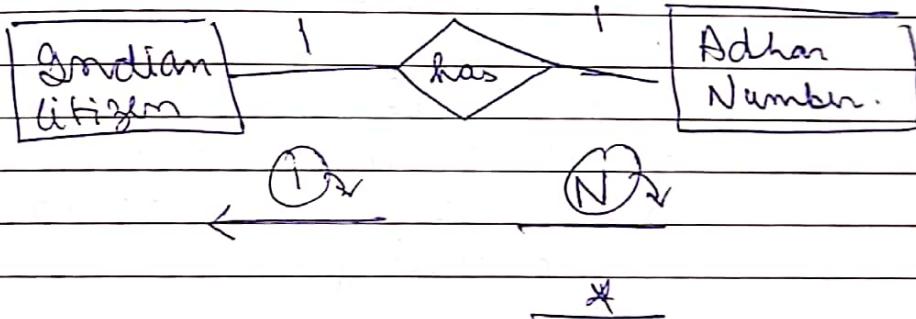
N-ary



Mapping Cardinalities / Cardinality ratio :-
 Expresses 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) → Aadhar Number.
- 1:N (one to many) → Mobile Number.
- N:1 (many to one)
- M:N (many to many) → Student .



Participation constraints:

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

Three 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 occurs/participating in a relationship.

Min cardinality:

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

→ Partial participation.

→ Total participation.

DBMS

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

$$f: \alpha \rightarrow \beta$$

α	β
$d_1 \rightarrow a$	1
$d_2 \rightarrow b$	2
c	3
d	4

if we know "d"
we can search "B"

For the same
value of d we
can't get two
different values of
B.

$$f: \alpha \rightarrow \beta$$

$$f(x) \rightarrow y$$

$$\begin{array}{c} \checkmark 1 \rightarrow a \\ \quad \quad \quad 1 \rightarrow b \end{array}$$

$$\begin{array}{c} \checkmark 2 \rightarrow a \end{array}$$

dependent R

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

$$d \rightarrow \beta$$

$$\text{if } d_1[\alpha] = d_2[\alpha]$$

	α	β
$d_1 \rightarrow$	a	b
$d_2 \rightarrow$	a	b

determinant

$$\hookrightarrow d_1[B] = d_2[B]$$

$$\alpha \rightarrow \beta$$

trivial

$$AB \rightarrow A$$

$$\text{if } B \subseteq \alpha$$

non-trivial

$$B \not\subseteq \alpha$$

$$AB \rightarrow ABC$$

Functional Dependencies

a)	$A \rightarrow BC$	✓	A	B	C	D	E
b)	$DE \rightarrow C$	✓	a	2	3	4	5
c)	$C \rightarrow DE$	✗	2	a	3	4	5
d)	$BC \rightarrow A$	✓	a	2	3	6	5
			a	2	3	6	6.

$$\alpha \rightarrow \beta$$

$\times \left\{ \begin{array}{l} a - 1 \\ a - 2 \end{array} \right.$

$$\alpha \rightarrow \beta$$

$a - 1$

$b - 1$

- check if all "a" are unique • checking 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^+

$R(ABC)$

$A \rightarrow B$

$B \rightarrow C$.

$A \rightarrow BC$

$R(ABCDPEF)$

$A \rightarrow B$.

$C \rightarrow DE$.

$AC \rightarrow F$.

$D \rightarrow AF$.

$E \rightarrow CF$.

$$(A)^+ = A$$

AB

$$(A)^+ = ABC$$

$R(ABCDEF)$

$\checkmark A \rightarrow B$

$C \rightarrow DE$

$AC \rightarrow F$

$\checkmark D \rightarrow AF$

$E \rightarrow CF$.

$$(D)^+ = D$$

= ADF

= ABDF

$$(DE)^+ = ABCDEF$$

Initially /
Indirectly.

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 in very relational database, can be used to generate closure set.

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

• Reflexivity :

if $y \subseteq x$
then $x \rightarrow y$

• Union :

if $x \rightarrow y \text{ & } x \rightarrow z$
 $x \rightarrow yz$

• Augmentation :

(Augmented) if $x \rightarrow y$
then $xz \rightarrow yz$

• Decomposition :

if $x \rightarrow yz$
then $x \rightarrow y \text{ & } x \rightarrow z$

• Transitivity :

if $x \rightarrow y \text{ & } y \rightarrow z$
then $x \rightarrow z$

• Pseudo transitivity

if $x \rightarrow y \text{ & } wy \rightarrow z$
then $wx \rightarrow z$

• Composition

if $x \rightarrow y \text{ & } z \rightarrow w$
 $xz \rightarrow yw$

$$AB \rightarrow C$$

$$\cancel{A} \rightarrow C$$

$$B \rightarrow C$$

Closure Set of attributes :-

$$R(ABC) \quad \left. \begin{array}{l} A \rightarrow B \\ B \rightarrow C \end{array} \right\} A \rightarrow BC.$$

$$F = F_1 + F_2$$

$$\begin{matrix} A \rightarrow B & A \rightarrow C \\ \hline A \rightarrow BC \end{matrix}$$

$$\begin{aligned} A^+ &= \{A, B, C\} \\ B^+ &= \{B, C\} \\ C^+ &= \{C\} \end{aligned}$$

1. $R(ABCDEF)$

$$A \rightarrow B$$

$$BC \rightarrow DE$$

$$AEF \rightarrow C$$

$$(AC)^+ = AC$$

$$ABC$$

$$A \rightarrow B$$

ABCDEF

2. $R(ABCDEF)$

$$A \rightarrow BC$$

$$CD \rightarrow E$$

$$Iterations B \rightarrow D$$

$$E \rightarrow A$$

$$(B)^+ = B$$

BD Same result

3. $R(ABCDEF)$

$$AB \rightarrow C$$

$$BC \rightarrow AD$$

$$D \rightarrow E$$

$$CF \rightarrow B$$

$$(AB)^+ = ABC$$

$$ABCD$$

$$ABCDEF$$

4. $R(ABCD EFGH)$

$$A \rightarrow BC$$

$$CD \rightarrow E$$

$$E \rightarrow C$$

$$D \rightarrow AEH$$

$$ABH \rightarrow BD$$

$$DH \rightarrow BC$$

$$BCD \rightarrow HI$$

?

$$(BCD)^+ =$$

$$ABCDEAH$$

Equivalence of Functional Dependency $R(ACDEH)$ $F: A \rightarrow C$ $AC \rightarrow D$ $E \rightarrow AD$ $E \rightarrow H$ $F \subset G$ $G: A \rightarrow CD$ $E \rightarrow AH$ a) $F \subset G$.b) $F \supset G$.c) $F = G$.d) $F \neq G$. $(A)^+ = ACD$ $(E)^+ = EAHDHC$ $(A)^+ = ACD$ $(AC)^+ = ACD$ $(E)^+ = EAHDHC$ $R(PQRS)$ $X: P \rightarrow Q$ $Q \rightarrow R$ $R \rightarrow S$ $Y: P \rightarrow QR$ $R \rightarrow S$ $(P)^+ = PQRS$ $(Q)^+ = Q$ $(R)^+ = RS$ $(P)^+ = PQRS$ $(R)^+ = RS$ $X \subseteq Y$ $Y \subseteq X$ $X = Y$ $X \neq Y$ $Y \subseteq X$ $d) X \neq Y$

Irreducible set of Fd (canonical form)

$R(wxyz)$

$x \rightarrow w$

$wz \rightarrow xy$

$y \rightarrow wxz$

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

$R(wxyz)$

$x \rightarrow w$

$wz \rightarrow xy$

~~$y \rightarrow wxz$~~

$x \rightarrow w \checkmark$

~~$wz \rightarrow x \times$~~

~~$wz \rightarrow y \checkmark$~~

~~$y \rightarrow w \times$~~

$y \rightarrow x \checkmark$

$y \rightarrow z \times$

$x^+ = xw$

$x^+ = x$

$(wz)^+ = wzxy$

$(wz)^+ = w2yzx$

$y \rightarrow x \checkmark$

$y \rightarrow z \times$

Since both results are the same
So Redundant.

$x \rightarrow w$

$wz \rightarrow y$

$y \rightarrow x$

$y \rightarrow z$

$x \rightarrow w$

$wz \rightarrow y$

$y \rightarrow xz$

$$(wz)^+ = wzxy$$

$$w^+ = w$$

$$z^+ = 2$$

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

Key \rightarrow Chabi

to extract data

R

	A	B	C
1	a	b	f
2	b	c	g
3	c	c	g
4	c	d	n

Set of

The attributes which can uniquely identify answer on a topic called Key.

$$\text{Key} = A, BC$$

$$\text{Key} = A \rightarrow BC$$

$$(A)^+ = R$$

$$(Key)^+ = R$$

1. R(ABCD)

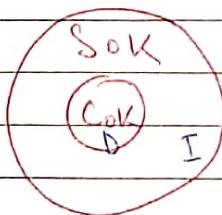
$$A \rightarrow BC$$

$$A^+ = ABC$$

$$(Key)^+ = R$$

$$(S_{OK})^+ = R$$

\hookrightarrow COK (efficient)



2. R(ABCD)

$$ABC \rightarrow D$$

$$AB \rightarrow CD$$

$$A \rightarrow BCD$$

$$(ABC)^+ = ABCD$$

$$(AB)^+ = ABCD$$

$$(A)^+ = ABCD$$

Subst.

3. R(ABCD)

$$B \rightarrow ACD$$

$$ACD \rightarrow B$$

$$(B)^+ = ACDB$$

$$(ACD)^+ = ACDB$$

$B \rightarrow$ Super key

$ACD \rightarrow$ Candidate key

1. R(ABCD)

$$A \rightarrow B$$

$$B \rightarrow C$$

$$C \rightarrow A$$

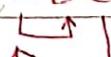
$$AD, BD, CD$$

$$(AD)^+ = ABCD$$

$$(BD)^+ = ABCD$$

$$(CD)^+ = ABCD$$

$R(A \overbrace{B}^C C D)$



Find the attribute with no incoming edge.

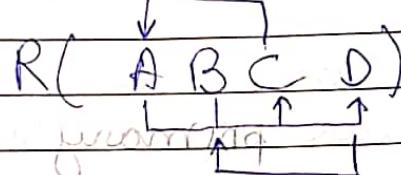
Orion

PAGE:

DATE:

11

Essential.

 $R(A B C D E F)$ $A B \rightarrow C D$ $C \rightarrow D$ $D \rightarrow B$ 

$$(A)^+ = A X \quad AB \checkmark$$

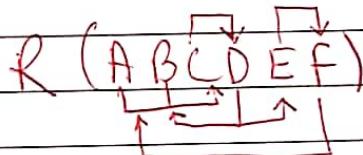
$$(B)^+ = B X \quad ACX$$

$$(C)^+ = C X \quad AD \checkmark$$

$$(D)^+ = D B \times \quad BC \checkmark$$

$$BD \times$$

$$CD \checkmark$$

 $\times ACB$ $\times ACD$ ACX $\times BDA$ $\times BDC$ 

$$AX \quad AB \checkmark$$

$$BX \quad AE \times$$

$$C \checkmark \quad AF \times$$

$$D \checkmark \quad BE \checkmark$$

$$EX \quad BF \checkmark$$

$$FX \quad EF \times$$

 $AB \rightarrow C$ $C \rightarrow D$ $D \rightarrow BE$ $E \rightarrow F$ $F \rightarrow A$

Insertion, Deletion and Update

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

Gender :-

In the table student info we have tried to store entire data about student.

Result :-

Entire branch data of a branch must be reflected 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 (slow)

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)

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

Student - info

	S-id	name	age	Br_code
Kisi	1	A	18	101
"Tabli"	2	B	19	101
Ko	3	C	18	101
decompose	4	D	21	102
Karma	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

- As one paragraph contains a single idea similarly one table must contain direct and main data about an entity.
- Nonnormalization (decomposition of tables) of table is done on the basis of functional dependencies.

INF^o :- (atomic value)

Rollno.	Name	Course
101	Modi	CN
102	Sonia	DBMS
		CO
101	Modi	CN
101	Modi	OS
102	Sonia	DBMS
102	Sonia	CO

- Every cell contains atomic values

$2NF^o$:-

$$R(A B C D) \quad (A B)^* = A B C D.$$

$$A B \leftarrow C \text{ or } K.$$

$A, B \leftarrow$ prime attribute

$C, D \leftarrow$ non-prime attribute

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

non-prime attribute

-ANS o :-

$$R_1(A B D)$$

$$R_2(B C)$$

Only a part of candidate key.

$R(A \sqsubset BC)$

$B \rightarrow C$

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

$A, B \leftarrow P$

$C \leftarrow N_o P.$

$P \rightarrow N_o P$ (Partial dependency)
 $B \rightarrow C$.

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

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

1. $R(ABCD E)$

$AB \rightarrow C$

$D \rightarrow E$.

2. $R(ABCD E)$

$$(AC)^+ = R$$

$R(AB \sqsubset CD E)$.

(P,D) A → B

B → E

(P,D) . C → D

$$(ABD)^+ = R$$

** (Partial dependency).

$R_1(ABC)$

$R_1(AB \sqsubset E)$

$R_2(BE)$

$R_2(CD)$

$R_3(ABD)$

$R_3(AS)$

3. $R(ABCDEFHIJ)$ P.D. $AB \rightarrow C$ P.D. $AD \rightarrow GH$ $(ABD)^+ = R$.P.D. $BD \rightarrow EF$ P.D. $A \rightarrow I$ $I \rightarrow J$. $R_1(A \overline{B} \overline{C})$ $R_2(A \overline{I})$ $R_3(A \overline{D} \overline{G} \overline{H} \overline{J})$ $R_4(B \overline{D} \overline{E} \overline{F})$ $R_5(A \overline{B} \overline{D})$.

Transitive dependency :-

A P.D from $\alpha \rightarrow \beta$ is called transitive if $\alpha, \beta \in$ 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

 $NP \rightarrow NP$.

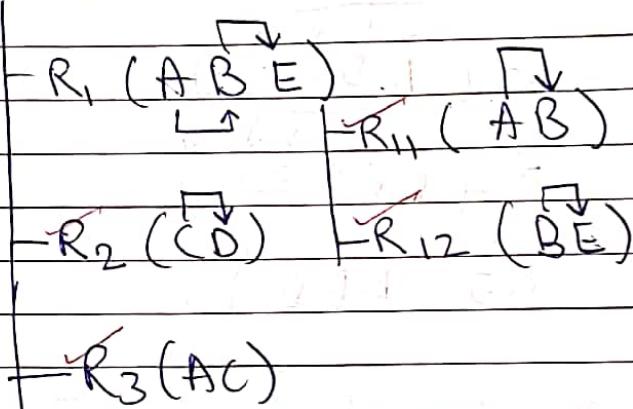
$R(A \overline{B} \overline{C})$	A	B	C	A	B	B	C
	a	1	x	a	1	1	x
$A \rightarrow B$	b	1	x	b	1	2	y
$B \rightarrow C$	c	1	x	c	1	3	z
	d	2	y	d	2		
	e	2	y	e	2		
	f	3	y	f	3		
	g	3	y	g	3		

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

$AB \rightarrow C$ $\times A \rightarrow B$.
 $B \rightarrow D$ $\times B \rightarrow E$.
 $D \rightarrow E$ $\checkmark C \rightarrow D$

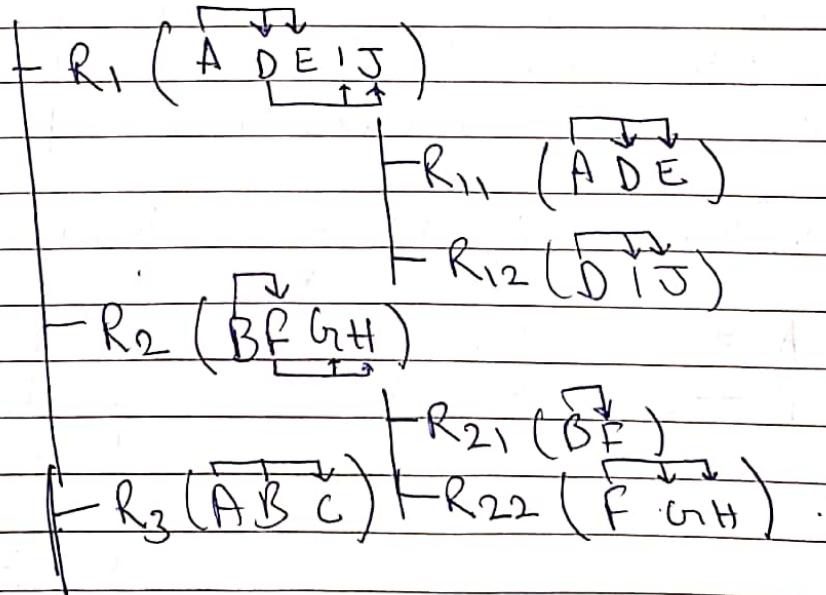
$$(AC)^+ = ABCDE.$$

ABCDE



A. $R(ABCDEFGHIJ)$

$\times AB \rightarrow C$
 $\times A \rightarrow DE$. Candidate Key :-
 $B \rightarrow F$.
 $F \rightarrow GH$. $\therefore (AB)^+ = R$.
 $D \rightarrow IJ$.



BCNF :-

(Boyce Codd Normal Form)

2NF.

$$\alpha \xrightarrow{X} B$$

prime. non prime

3NF

$$\alpha \xrightarrow{X} B$$

non prime non prime

$$\alpha \longrightarrow B$$

P/NP

P

$$R(A \boxed{B} C) \quad (A)^+$$

$$AB \rightarrow C$$

$$(AB)^+ \checkmark$$

$$C \rightarrow B$$

$$(AC)^+ \checkmark$$

$$\alpha \longrightarrow B$$

$$P/NP \quad P$$

$$\alpha \longrightarrow B$$

S.O.K.

R

$$R(ABC)$$

 (AB)
 (AC)

$$AB \rightarrow C$$

$$\begin{array}{c|c|c|c|c} A & C & C & B & \\ \hline a & x & x & 1 & \end{array}$$

$$\begin{array}{c|c|c|c} A & B & C & \\ \hline a & 1 & x & \end{array}$$

$$C \rightarrow B$$

$$\begin{array}{c|c|c|c|c} & y & y & 2 & \\ \hline b & & y & 2 & \end{array}$$

$$\begin{array}{c|c|c|c} & 2 & 2 & y \\ \hline b & 2 & 2 & y \end{array}$$

$$\alpha \longrightarrow B$$

$$\begin{array}{c|c|c|c|c} & z & z & 2 & \\ \hline c & z & z & 2 & \end{array}$$

$$\begin{array}{c|c|c|c} & 2 & 2 & z \\ \hline c & 2 & 2 & z \end{array}$$

 α must be a superkey

$$\begin{array}{c|c|c|c|c} & w & w & 3 & \\ \hline c & w & w & 3 & \end{array}$$

$$\begin{array}{c|c|c|c} & 3 & 3 & w \\ \hline c & 3 & 3 & w \end{array}$$

$$-R_1(C \boxed{B})$$

$$\begin{array}{c|c|c|c|c} & d & w & & \\ \hline e & & w & & \end{array}$$

$$\begin{array}{c|c|c|c} & 3 & 3 & w \\ \hline c & 3 & 3 & w \end{array}$$

$$-R_2(AC)$$

$$-R_3(AB)$$

1. R(AB(CDEF(GH))

$$\begin{array}{ll} AB \rightarrow C & BCNF \\ A \rightarrow DE & 3NF \\ B \rightarrow F & 2NF \\ F \rightarrow GH & INF \\ \textcircled{AB} & \end{array}$$

2. R(AB(CDE))

BCNF.

3NF

2NF.

INF

$CE \rightarrow D$

$D \rightarrow B$

$C \rightarrow A$ WNF

CE

3. R(ABCDEF)

$AB \rightarrow C$

$DC \rightarrow AE$

$E \rightarrow F$

$(ABD) (BCD)$

4. R(AB(CDEF(GH)))

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

ABD

INF.

5. R(ABCDE)

$\{AB \rightarrow CD\}$

3NF

$\{D \rightarrow A\}$ prime attribute
 $BC \rightarrow DE$

$(AB), (BD), (BC)$

6. R(AB(CDE))

$BC \rightarrow ADE$

$D \rightarrow B$

$(B), (C), (CD)$

Candidate key || prime attribute

7. $R(VWXYUZ)$

$$CK \xrightarrow{X} Y \xrightarrow{V} PA$$

$$Y \rightarrow Z$$

$$Z \rightarrow U$$

$$VW \rightarrow X$$

$$VW, XW$$

INF

8. $R(ABCDEF)$

$$ABC \rightarrow D$$

$$ABD \rightarrow E$$

$$CD \rightarrow F$$

$$CDF \rightarrow B$$

$$BF \rightarrow D$$

$$ABC, AED$$

INF

9. $R(ABC)$

$$A \rightarrow B \quad BLNF$$

$$B \rightarrow C$$

$$C \rightarrow A$$

$$A, B, C$$

How to find identify normal form

Orion

PAGE:
DATE:

11

1. $R(ABCDEF)$
 $A \rightarrow BCDEF$
 $BC \rightarrow ADEF$
 $DEF \rightarrow ABC$
 A, BC, DEF
 $BCNF.$

2. $R(ABC)$

$AB \rightarrow C$
 $C \rightarrow A$

AB, BA
 $3NF.$

3. $R(ABCDE)$

$A \rightarrow B$
 $BC \rightarrow E$
 $DE \rightarrow A$

ACD, BCD, CDE
 $3NF.$

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

5. $R(wx42)$

$2 \rightarrow w$
 $4 \rightarrow x_2$

$xw \rightarrow 4$
 $4, xw, x_2$
 $3NF.$

6. $R(ABCDEF)$

$f \rightarrow B$ np.
 $B \rightarrow E$
 $C \rightarrow D$
 $A E.$
 $INF.$

7. $R(ABCDEF)$
 $AB \rightarrow C$
 $DC \rightarrow AE$
 $E \rightarrow F$
 $INF.$
 ABD, BCD

8. $R(NWXY42)$

$2 \rightarrow 4$

$4 \rightarrow 7$

$VW \rightarrow X$

VW, XW

$INF.$

9. $R(ABCDEF)$

$ABC \rightarrow D$

$ABD \rightarrow E$

$CD \rightarrow F$ np.

$CDF \rightarrow G$

$BF \rightarrow D$

ABC, ACD

$INF.$

lossless Join Decomposition

NON-additive

tuple generation problem doesn't occur after decomposition

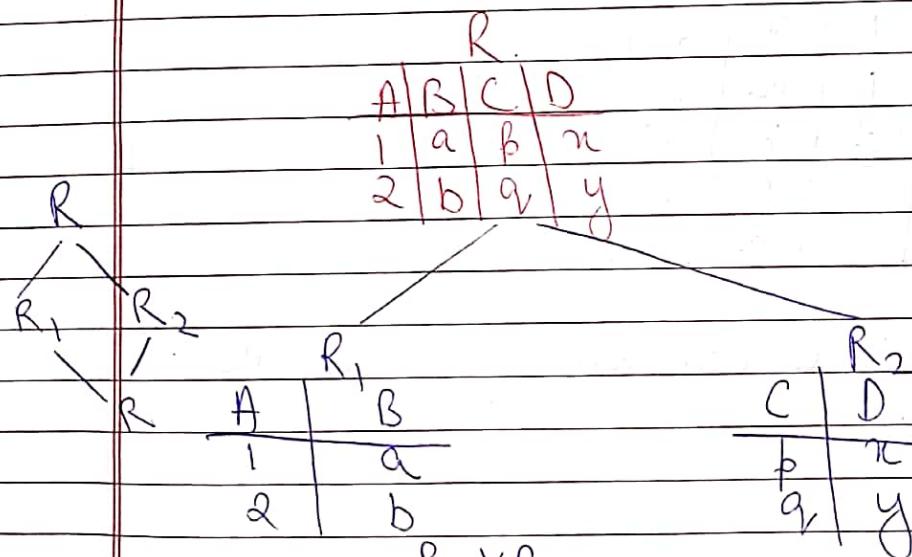
- o It is a mandatory property must always hold good.
- o If a relation R is decomposed into two relations R_1 & R_2 , then it will be lossless iff :

$$(i) \text{ attr}(R_1) \cup \text{attr}(R_2) = \text{attr}(R)$$

$$(ii) \text{ attr}(R_1) \cap \text{attr}(R_2) \neq \emptyset$$

$$(iii) \text{ attr}(R_1) \cap \text{attr}(R_2) \rightarrow \text{attr}(R_1)$$

$$\text{attr}(R_1) \cap \text{attr}(R_2) \rightarrow \text{attr}(R_2)$$



$R_1 \times R_2$			
A	B	C	D
a	p	n	
1	a	q	y
2	b	p	n
2	b	q	y

R

A	B	C
1	a	b
2	b	b
3	a	y

R₁

A	B
1	a
2	b
3	a

R₂

B	C
a	b
b	b
a	y

R₁ × R₂

A	B	C
1	a	b
1	a	y
2	b	b
3	a	b
3	a	y

From it is candidate key in any of the table.

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

$$\text{Then } F_1 \subseteq F^+ \\ F_2 \subseteq F^+ \\ \underline{(F_1 \cup F_2)^+ = F^+}$$

 $R(ABC)$ $A \rightarrow B$. $B \rightarrow C$ $C \rightarrow D$. $R_1(AB) \quad R_2(BC)$ $R(ABCD)$ $AB \rightarrow CD$. $D \rightarrow A$ $R_1(AD) \quad R_2(BCD)$ $R(F)$ $R_1(E_1)$ $R_2(F_2)$ $R(F_1 \cup F_2) = F$. $R(ABCD)$ $F: AB \rightarrow CD$ $D \rightarrow A$ $R_1(AB)$ $R_2(BC)$ $F_1: A \rightarrow B$ $F_2: B \rightarrow C$ $B \rightarrow A$ $C \rightarrow B$ $R_1(AD)$ $R_2(BCD)$ $F_1: D \rightarrow A$ $F_2: BD \rightarrow C$ $(F_1 \cup F_2)^+ = F^+$ $C^+ = CBA$ $(AB)^+ = AB$.
 $R(ABCDEF(GHIJ))$
 $R_1(ABC)$ $R_2(ADEF(IJ))$ $R_3(BF(GHI))$ $R_{21}(ADEF) \quad \} 3NF$ $R_{22}(D(IJ)) \quad \}$ $R_{31}(BF) \quad \} 3NF$ $R_{32}(FGHI) \quad \}$

File Structures

SRS



ER-diagram



Relation model

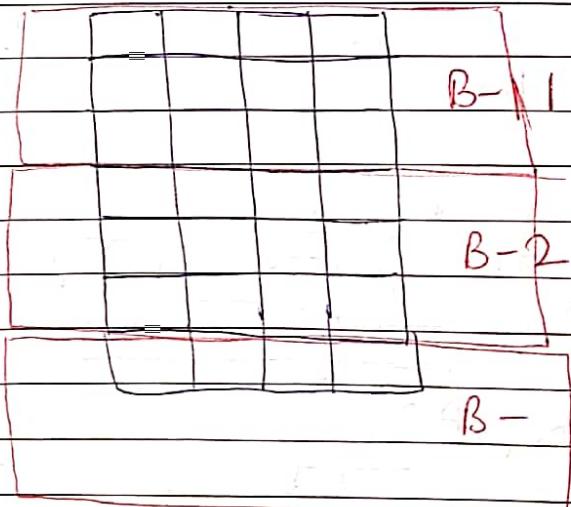
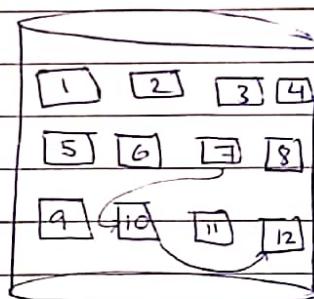


Normalization



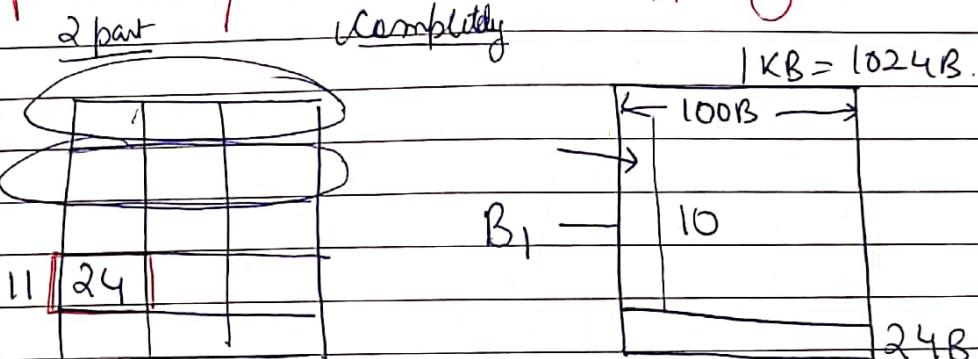
file structures
(underlying physical structures)

*** File → datastructure

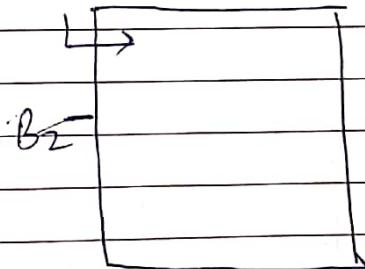


Sorted FileUnsorted File

- 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

block → record
ignore

Indexing

Sok Taf

Index →
size small

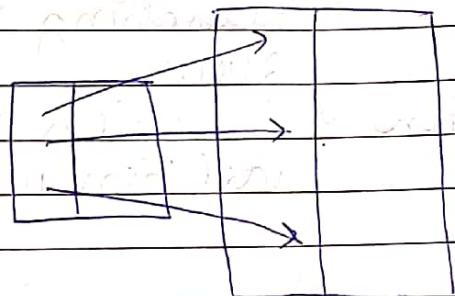
sorted/unsorted

→ to access the file quickly.

Book
Index
ch. - pg no

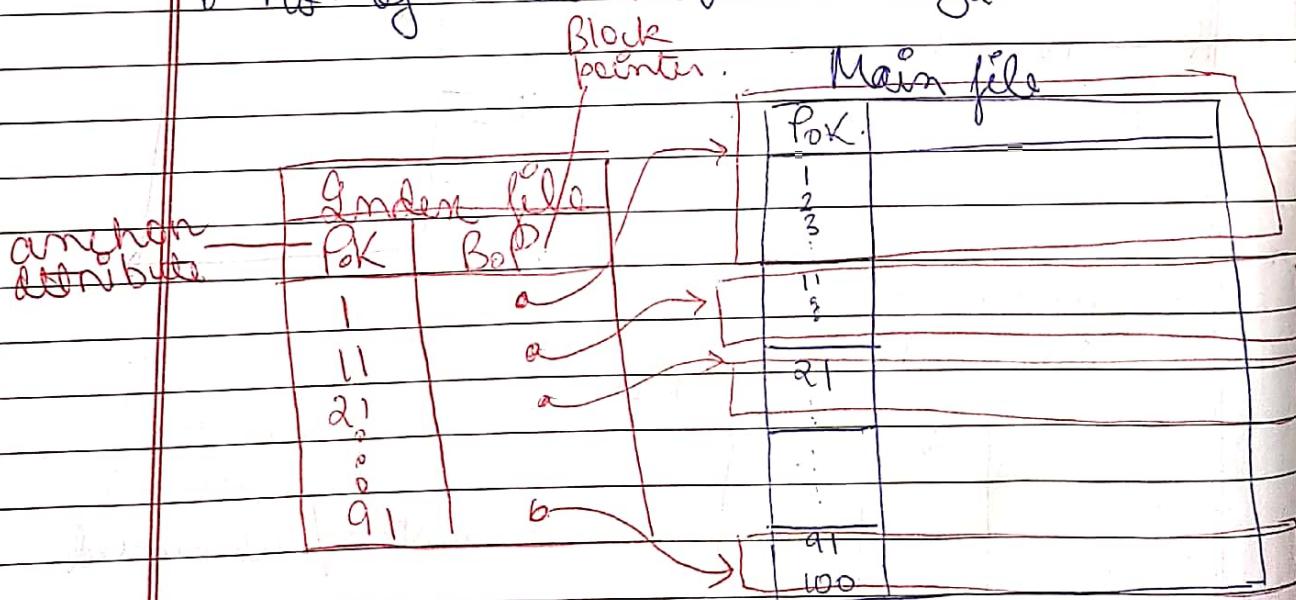
Types of indexing.

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



Primary indexing.

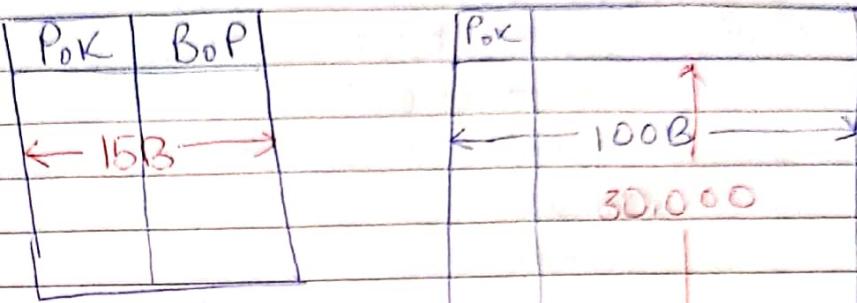
- Main file sorted.
- Primary key is used as anchor attribute (Search key).
- Example of sparse indexing
 - no. of entries = no. of blocks acquired in index file by the main file.
 - no. of access required = $\log_2 n + 1$.



Block size = 1024B.

INDEX FILE

MAIN FILE



Blocking factor =

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

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

$$\text{no. of block access required} = \log_2 n.$$

$$\begin{aligned} \text{Blocking factor} &= \frac{1024B}{100B} = [10.24] = 10. \\ \text{no. of blocks for main file} &= \frac{30,000}{10} = 3000. \end{aligned}$$

$$\begin{aligned} \text{no. of access required} &= \log_2 n \\ &\approx \log_2 3000 \\ &= [11.0 \dots] = 12 \end{aligned}$$

INDEX
FILE

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

$$\begin{aligned} \text{no. of blocks for index file} &= \left[\frac{30,000}{68} \right] \\ &= [44 \dots] \\ &= 45. \end{aligned}$$

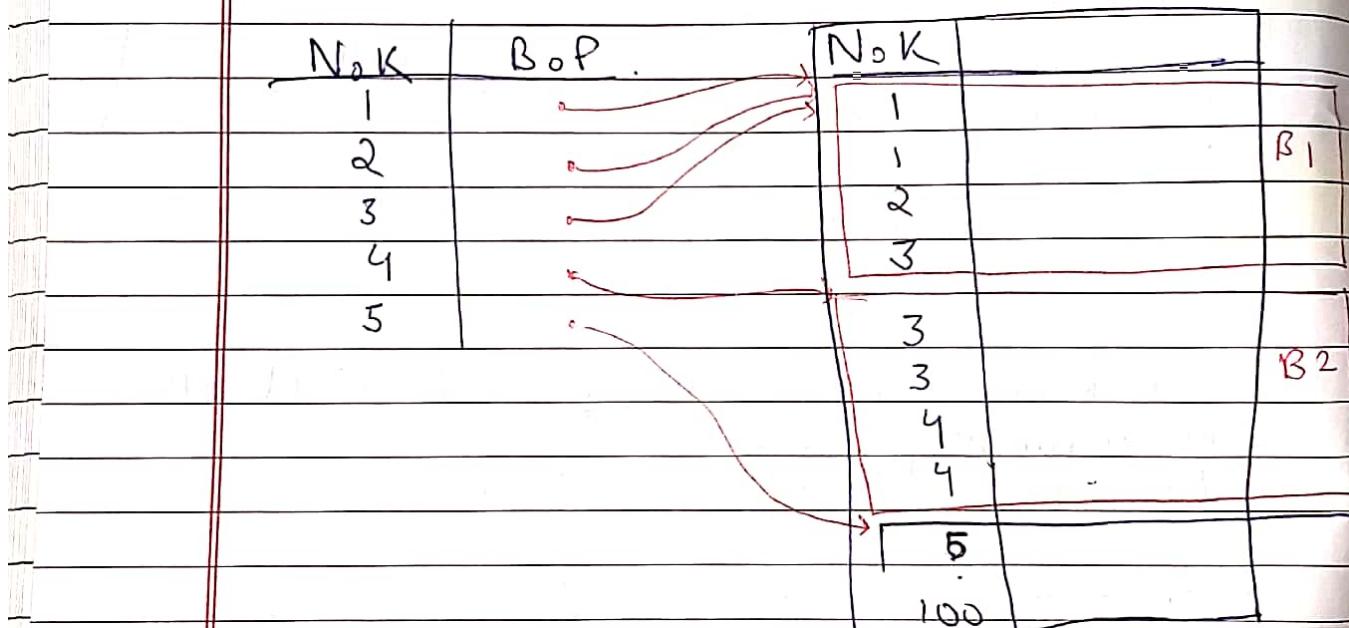
$$\begin{aligned} \text{no. of block access required} &= [\log_2 n] + 1 \\ &= 6 + 1 = 7. \end{aligned}$$

Clustering index

- Main file is sorted (on some non-key attributes).
- 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 access required will be $\geq \log_2 n + 1$.

Index file

Main file



Secondary Indexing :-

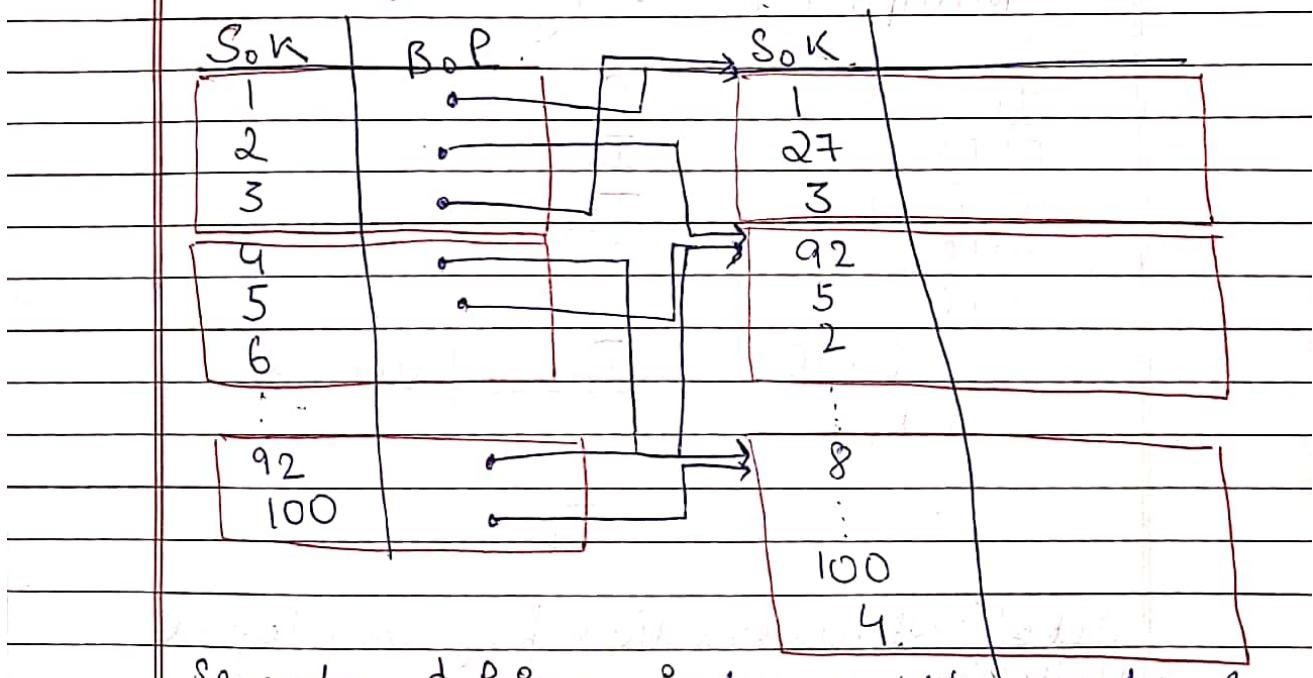
- Main file Unsorted.
- can be done on key as well as on non-key attribute.
- called secondary because normally one indexing is already done

$$\left[\text{no of entry in index file} = \text{no of entry in main file} \right]$$

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

Index file

Main file



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 cascades

Transaction designing protocols

- Time stamping
- lock based protocols

2-phase locking

Graph based protocol

Multiple granularity

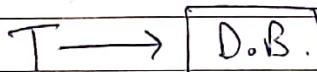
T ₁	P
R(A)	T ₁
A = A - 10	T ₂ ↙
W(A)	
R(B)	
B = B + 10	T ₆₀ failure X
W(B)	

T_n 100

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

Transaction:-

1. Atomicity



2. Consistency

3. Isolation

4. Durability

- 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

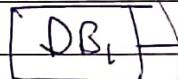
If working of one transaction doesn't affect other transaction
Concurrency control component.

Durability

If a change has occurred, the change will stay.

Recovery management component.

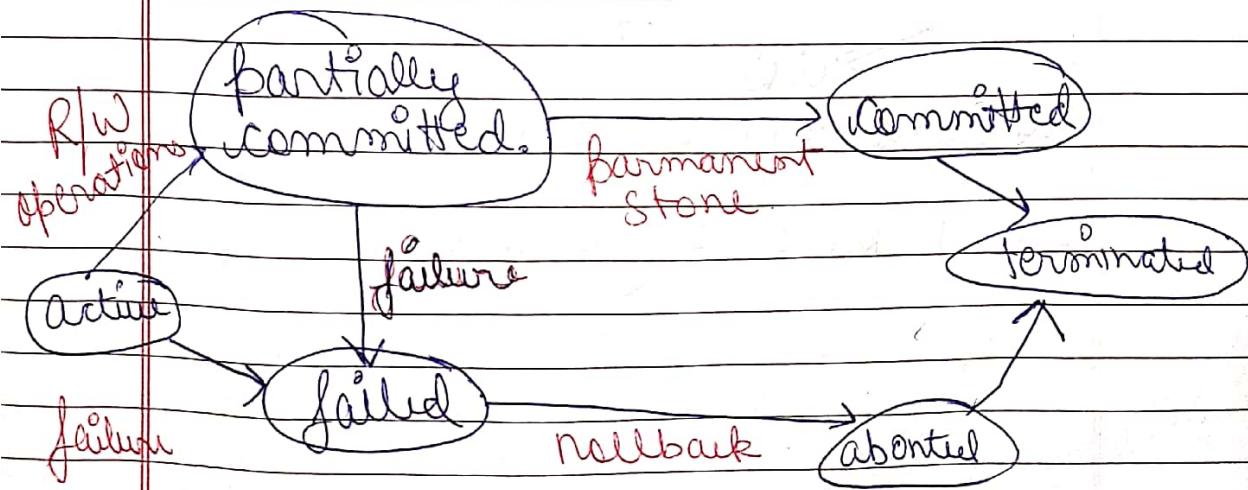
if these holds
prep good =
consistent.



consistent. (After transaction it will be consistent)

$$A+B = A+B$$

Transaction States



Initial state \rightarrow active

When the transaction is executing in terms of instructions

I_1
 I_2
 \vdots
 I_n

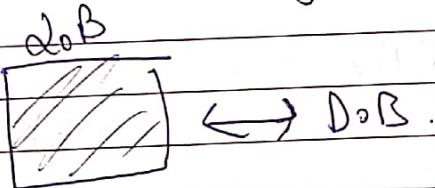
When instruction is running

There may be a failure (faulty state)

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

(local buffer)

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



$DB_1 \xrightarrow{T} DB_2$

DB

\Rightarrow Consistent in both cases.

e.g. Ticket booking & cancelling.

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

→ two students giving

exams & one cheating
from the other one.

One submitting the paper
early. due to the T
changed ans.

Unrepeatable Read Problem:-

<u>X = 10</u>	T ₁	T ₂	
R(X)			child, book
	R(X) - 10		maggie
X++ W(X)		R(X) - 11	→ nagma channel.

Phantom Read Problem

<u>X = 10</u>	T ₁	T ₂	
R(X)			this same story
<u>do B X = 10</u>	D.delete(X)	R(X)	↓ thief

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

lost update problem (write write conflict)

	T ₁	T ₂	
DB			
A = 10	R(A)		→ conflict b/w these
so	W(A)		two writes
		W(A)	
↓		C	
DB = 10		C	
	H		
	50		

* Serial Schedule and non-Serial schedule in DBMS.

Schedule :-

S1	T ₁	T ₂	S2
T ₁	T ₂	T ₁	T ₂
	R(B)	R(B)	
	W(B)	W(A)	R(A)
consistency test	R(A)	R(B)	W(A)
	W(A)	W(B)	R(B)
		n ₁	n ₂
	R(A)	n ₁ + n ₂	R(B)
	W(A)		R(A)
	R(B)		W(B)
	W(R)		W(A)

serial

schedule

non-serial
schedule.

T₁, T₂, T₃, ..., T_n

n × (n - 1) × (n - 2) × ... × 1

$$\text{Total no. of } (n_1 + n_2 + \dots + n_n)! = n!$$

$$\text{Non Serial Schedule} = \frac{n_1! \cdot n_2! \cdot \dots \cdot n_n!}{n!}$$

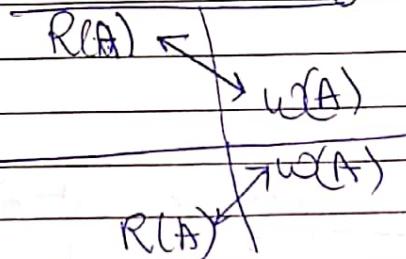
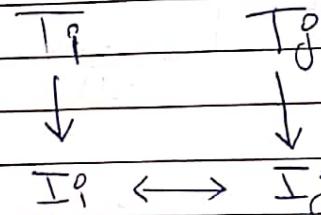
o If we can prove non serial
 ⇔ Serial then it is consistent

Orion

PAGE:

DATE: 11

o When instruction belong to same
 data item conflicts happen.



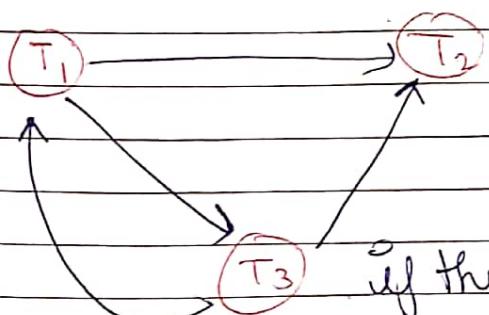
no change in TA1, TA2 & DB after swap.

~~a1 2 2012~~

Conflict Serializable Schedule

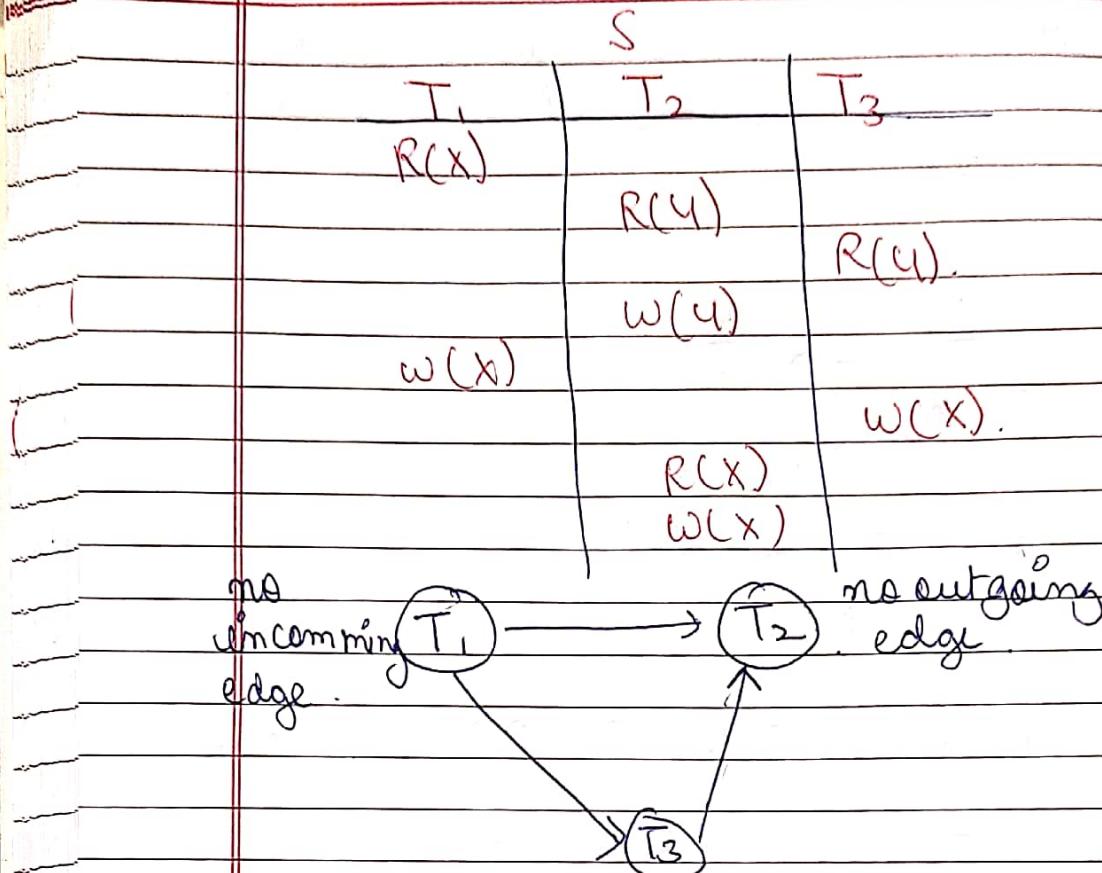
S		
T ₁	T ₂	T ₃
R(x)		
	R(z)	
	w(z)	
	R(u)	
R(y)		
	w(y)	
		w(x)
	w(z)	
w(x)		

1st 2nd

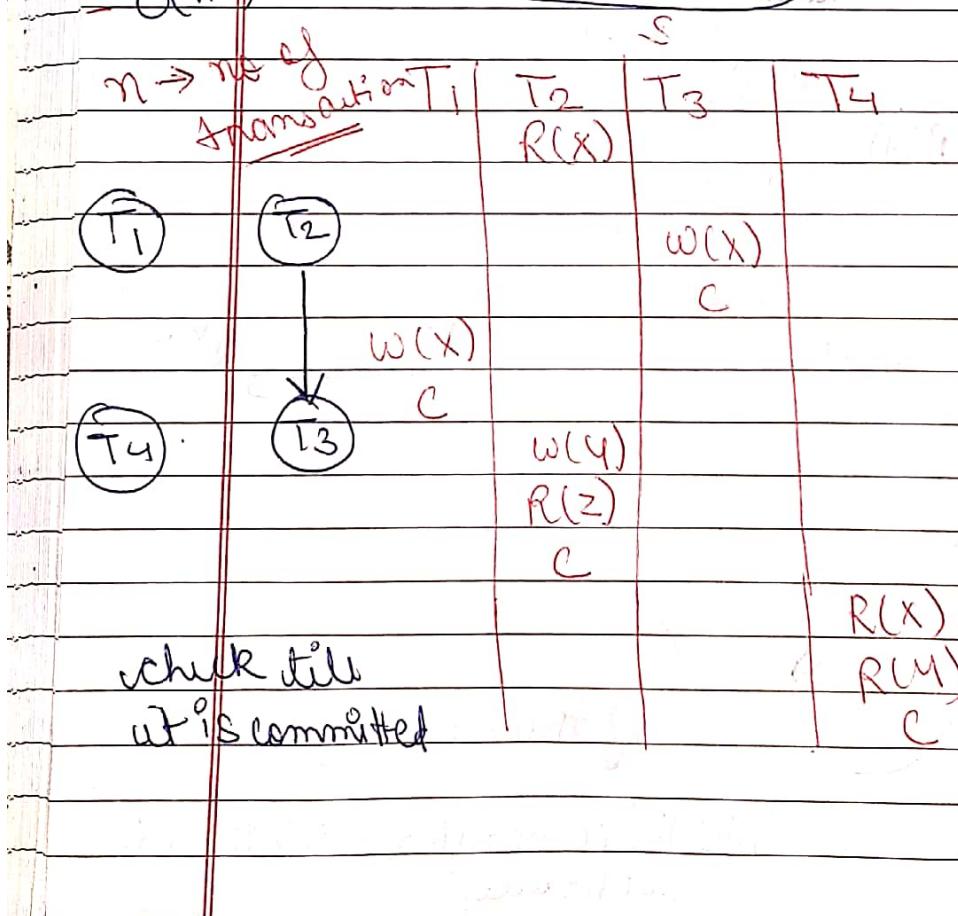


if there is a cycle

"NOT" a conflict serializable schedule



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

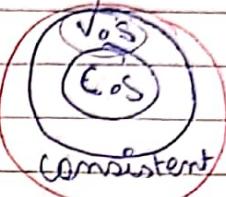


when one of its view serial is view
equivalent

Orion
PAGE: / /
DATE: / /

View Serializability

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



Inconsistent
schedules

S	T ₁	T ₂
	R(a)	
	w(a)	

(R(a))

S	T ₁	T ₂
	R(a)	
	wca)	
	R(b)	
	w(b)	

(R(a))

(R(b))

S	T ₁	T ₂
	R(a)	
	wcb)	
	R(b)	
	w(b)	

(R(a))

(R(b))

1) T₁ a T₁ } initial
T₁ b T₁ } Read.
same

2) final write same

3) Intermediate read same.

S	T ₁	T ₂	T ₃	T ₁	T ₂	T ₃	S ₁	S ₂	S ₃
	R(a)						1 2 3		
	w(a)							1 3 2	
								2 1 3	
								2 3 1	
								3 1 2	
								3 2 1	

(R(a))

(w(a))

(w(a))

(w(a))

Recoverable Schedule :-

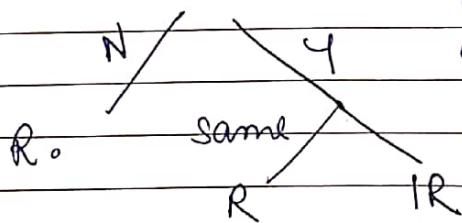
S	
T_1	T_2
$W - R(A)$	
$A = A + 10$	
$20 - W(A)$	
	$R(A) - (20)$ $A = A - 5 \quad (15)$ $W(A)$
<del style="font-size: 2em;">C <del style="font-size: 1.5em;">C <del style="font-size: 1.5em;">C	<del style="font-size: 1.5em;">C <del style="font-size: 1.5em;">C <del style="font-size: 1.5em;">C → <u>recoverable</u>

$$A = 10 \quad 15$$

~~A~~
~~B~~ $A = 10 \quad 10$
~~20~~
~~15~~

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

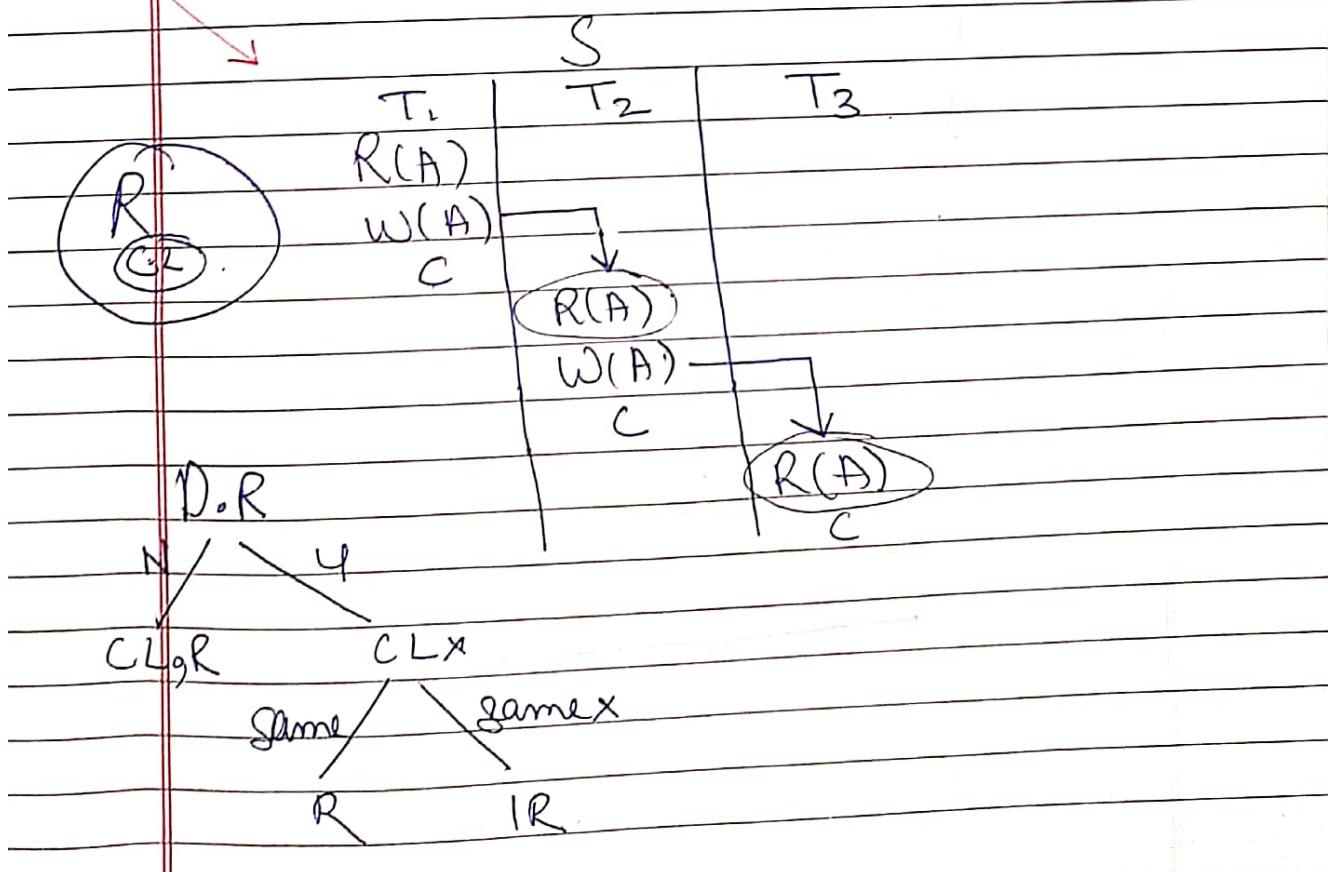
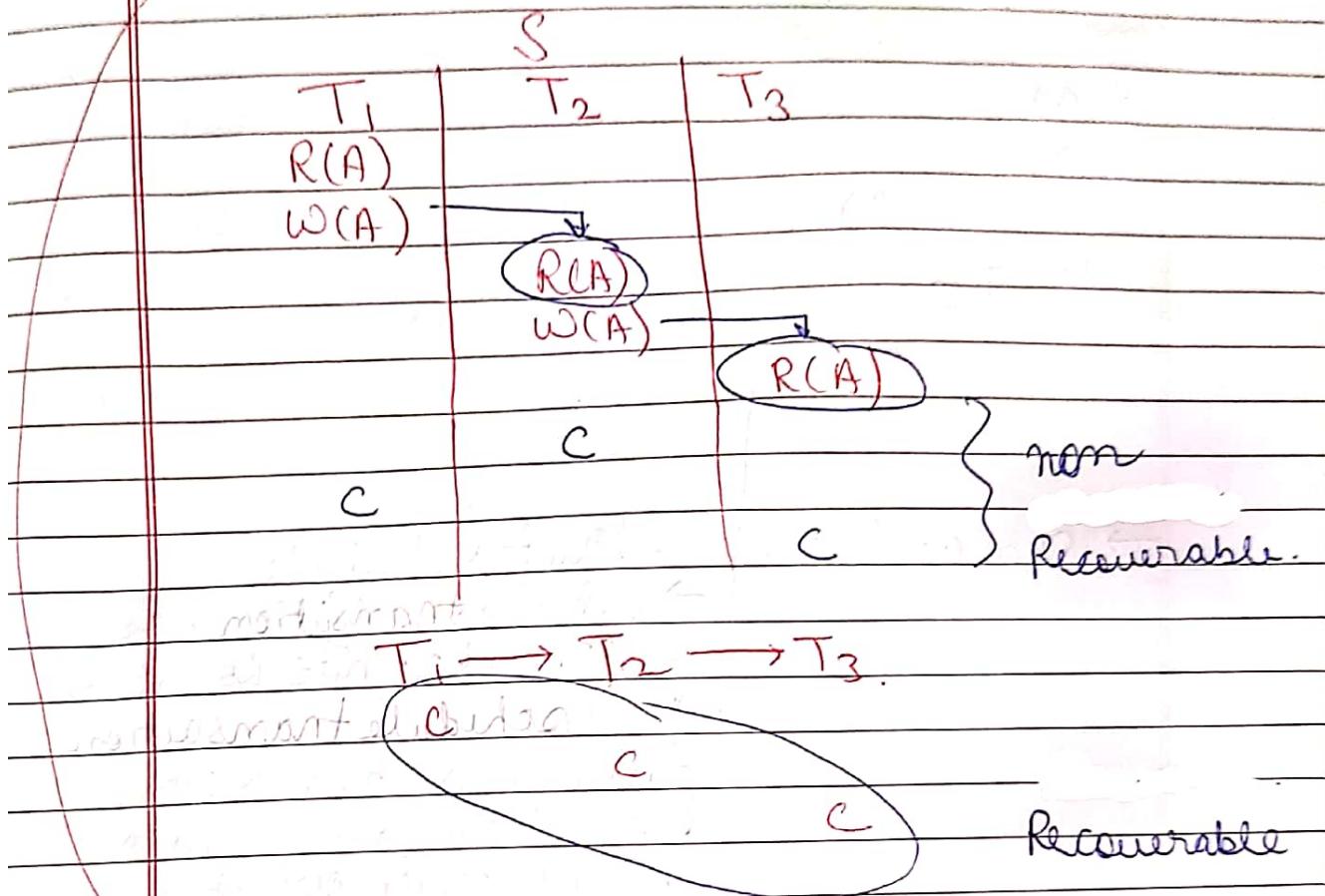
Dirty Read



Recoverable →

- The order dirty read
- = The order of commit.

Cascadless schedule (no cascading rollback)



Strict Schedule

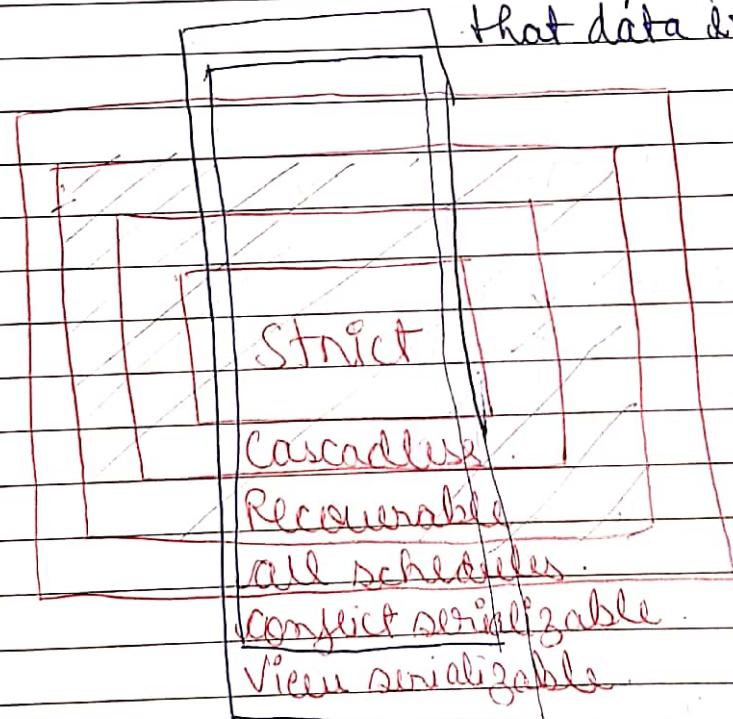
S1	T2	S2	T1	T2	S3	T1	T2
T1 R(a) W(a)		T1 R(a) W(a)	C		T1 R(a)	R(b)	
	W(a)			W(a)	W(a)		W(b)
C				R(a)	C		R(a)
	R(a)		C				C

→ cascadelise.

Strict schedule

→ when a schedule/transaction has performed a write operation and

data item then no other transaction can perform a read/write operation on that data item.



Concurrency Control techniques:-

- o 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).
- o Now will study protocols that guarantees to generate schedule which satisfy these properties specially (CoS).
 - o For CoS we must avoid conflicting instructions. We remember three conditions of conflicting instruction:
 - o Actual problem is different transaction trying to access data at same time.
 - o How to approach conflict serializability.
 - o Time stamping protocol (Ordering)
 - o Lock based protocol (lock-unlock)
 - ↳ 2PL (Basic, conservative, Strict, rigorous).
 - ↳ Graph based.
 - o Validation Protocol.
 - o Goals:-
 - o Concurrency, properties, time, logic.

Time Stamp Ordering Protocol :-

- Basic idea of Time stamping is to decide (the order) between the transactions before 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 $T.S(T_i)$.
- If i 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.o.S.o.S T_i will execute first before T_j .

Time Stamp with data item :-
For each data item Q protocol maintains two time-stamps.

W -time stamp(Q) :-

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

R -time stamp(Q) :-

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

T_i request for Read(Q) :-

if $T_{oS}(T_i) < W_{oT}(Q)$,

means T_i needs to read a value of Q that was already overwritten.

Hence request must be rejected & T_i must rollback.

if $T_{oS}(T_i) > W_{oT}(Q)$

Operation can be allowed and $R.T.S(Q)$ will be $\max(R.T.S(Q), T_{oS}(T_i))$

T_i request for Write(Q)

if $T_{oS}(T_i) < R.T.S(Q)$

means value of Q that T_i is producing was needed previously and the system assumed that the value would never be produced
Hence reject & rollback.

if $T_{oS}(T_i) < W.T.S(Q)$

T_i is attempting to write an absolute value of Q reject and Rollback

Otherwise

$$W.T.S(Q) = \max(W.T.S(Q), T_{oS}(T_i))$$

$$\begin{array}{|c|c|} \hline & S \\ \hline T_{oS}(T_P) = 5 & T_{oS}(T_X) = 10 \\ \hline \end{array}$$

$$\begin{array}{ccc} R(Q) & \xleftarrow{\quad} & W(Q) \\ R(Q) & \xleftarrow{\quad} & \end{array}$$

$$\begin{array}{|c|c|} \hline T_{sC}(T_P) = 10 & T_{oS}(T_X) = 5 \\ \hline \end{array}$$

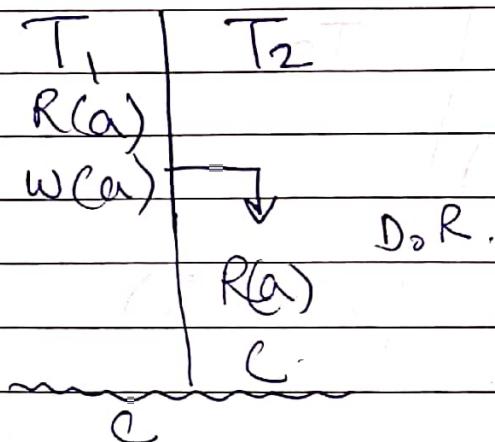
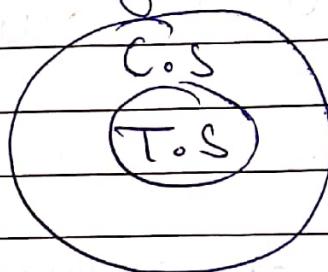
$$R(Q)$$

$$W(Q)$$

junior/ professor doing some activity
outside college gate.

Properties of Time Stamping Protocol :-

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



ThomashWrite Rule:-

Modify time stamping protocol to make some improvements and may generate those schedules that are VoS but not CoS and provide better concurrency.

- It modify time-stamping protocol in absolute write case when T_i nearest $W_i(Q)$
if $T_{oS}(T_i) < WTS(Q)$.
- here T_i attempts to write absolute value of Q rather than rolling back T_i , write operation is ignored.

T_1	T_2	T_3
$R(Q)$		
$\times W(Q)$	$W(Q)$	$(W(Q))$

$$RTS(Q) = 1$$

$$WTS(Q) = 2$$

$$Q = 10 \cdot 14.$$

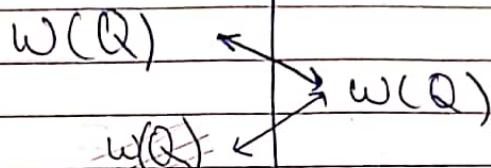
$$Q = 10 + 2 \cdot 14.$$

$$T_i^o = 12,$$

$$T_x = 14.$$

$$T.o.S(T_i^o) = 5$$

$$T.o.S(T_x) = 10$$



lock Based Protocol :-

- 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 can not obtained either Shared /exclusive mode lock.

Compatibility
table:

	Shared	exclusive
Shared	T	F
exclusive	F	F

Properties of lock Based Approach

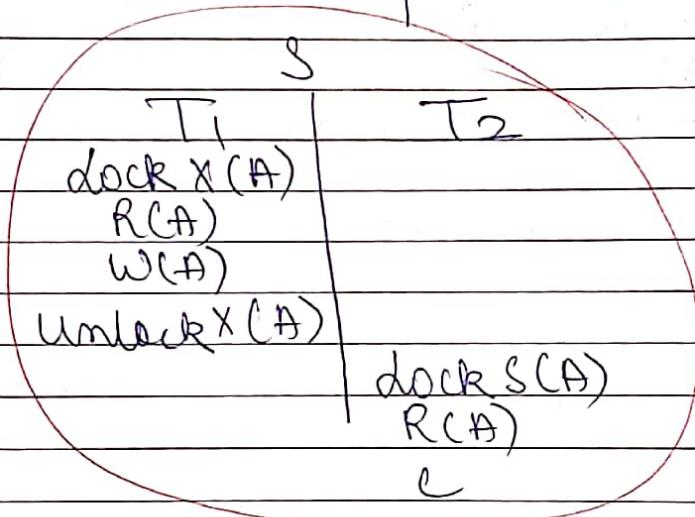
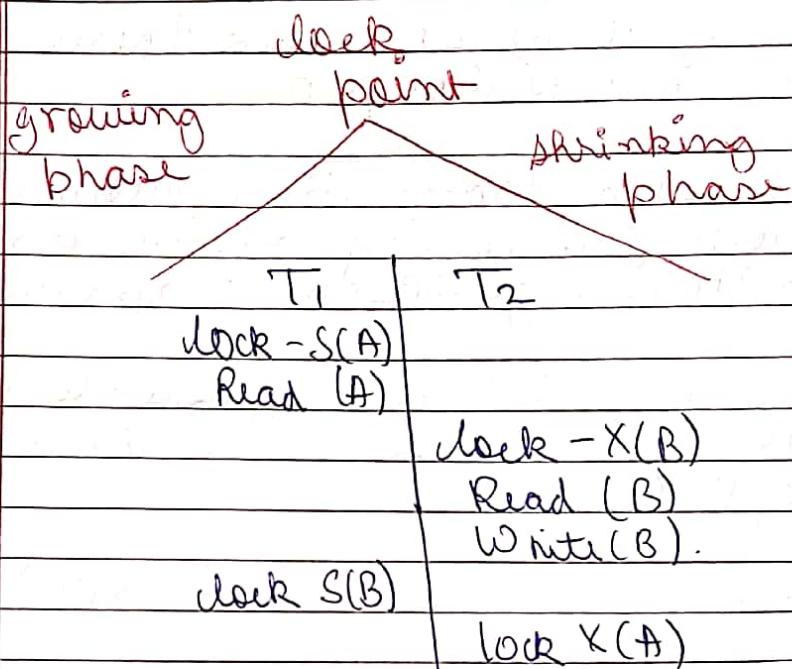
S	T ₁	T ₂
lock - X(A)		
R(A)		
W(A)		
Unlock(A)		lock S(B) R(B) Unlock(B)
lock - X(B)		
R(B)		
W(B)		
Unlock(B)		
		lock - S(A) R(A) Unlock(A)

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

Two-Phase Locking Protocol (2PL) :-

Basic 2PL

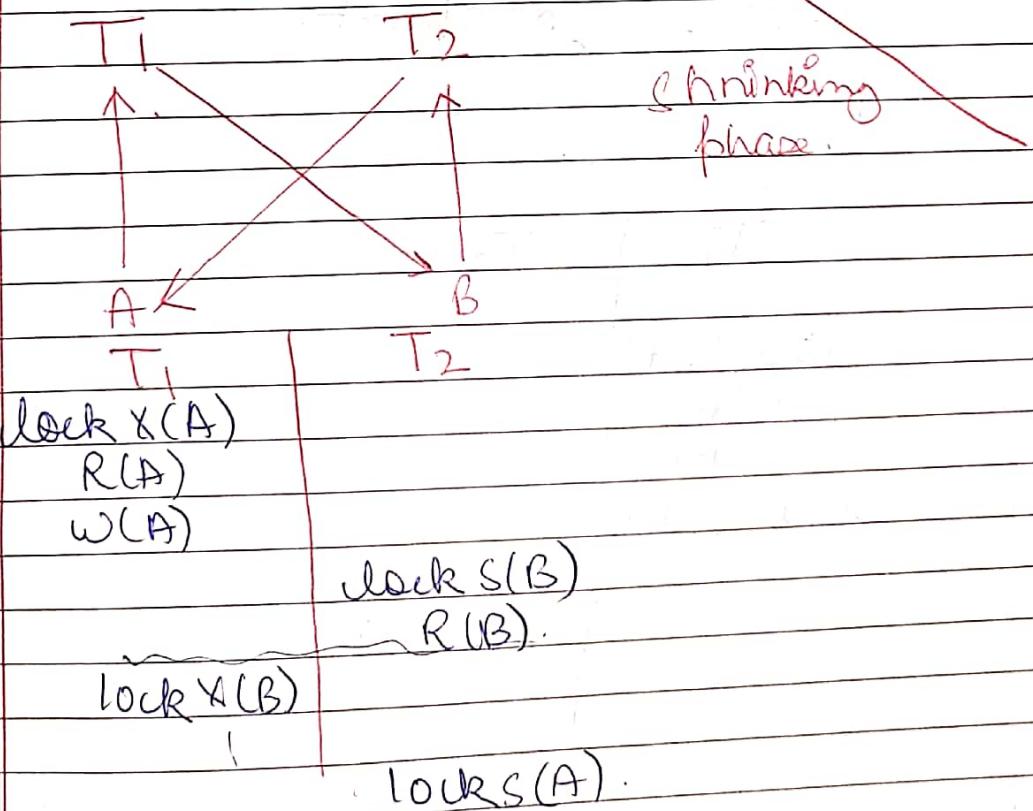
- This protocol requires that each transaction in a schedule will be two phased - growing phase & shrinking phase.
- In growing phase - transaction can only obtain locks but can not release any lock.
- In shrinking phase - transaction can only release locks but can not obtain any lock.
- transaction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



Conservative / Static QPL :-

- There is no queuing 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.
- COS, VOS independent from deadlock.
- Possibility of irreversable schedules and cascading rollbacks.

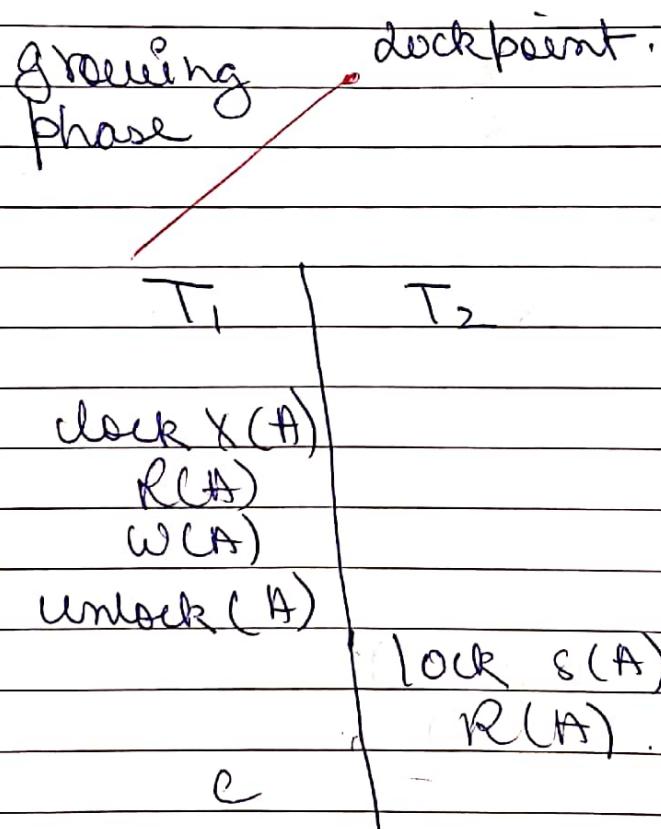
lockpoint



Rigorous QPL

It is an improvement over 2PL protocol while we try to ensure recoverability and cascadeliveness.

- ① 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.
 - ② Will ensure CoS, Vs, recoverability, cascadelssness
 - ③ Suffer from deadlock & inefficiency.

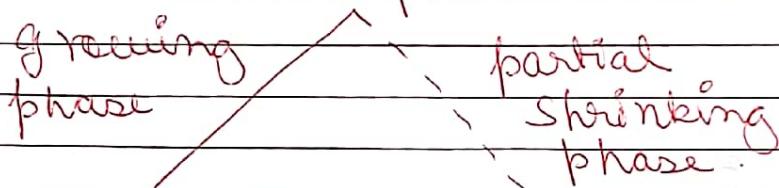


Strict 2PL :-

This is an improvement over Rigorous 2PL.

- In the shrinking phase locking 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.

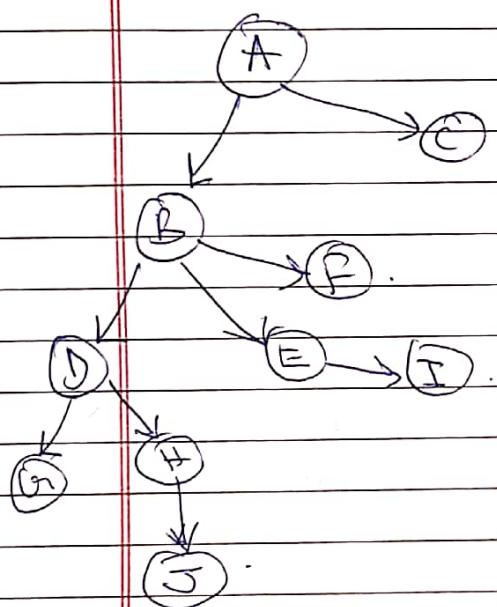
lock point:



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 have prior knowledge about the order in which the database items will be accessed.
- We impose partial ordering - on set of all data items $D = \{d_1, d_2, \dots, d_n\}$
 ~~$d_i \rightarrow d_j$ then any transaction accessing both d_i and d_j must access d_i before d_j .~~
 - This may be because logical or physical organisation or only because of concurrency control.

- o after P.O set of all data items will know be found as directed acyclic graph called database graph.
- o for sake of simplicity, we follow two restrictions -
 - ↳ will study graphs that are rooted tree.
 - ↳ will use only exclusive mode locks.



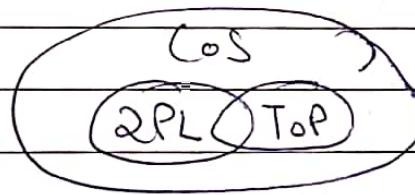
Tree Protocol

- 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(H)
	unlock(C)

Properties of Tree Protocol:-

- All schedules that are legal under the tree protocol are C.o.S & V.o.S.
- Tree Protocol ensure freedom from deadlock.
- Tree protocol don't ensure recoverability and cascadelness.
- 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 overshad waiting time and decrease in concurrency.
- Transaction must know exactly what data items are to be accessed.



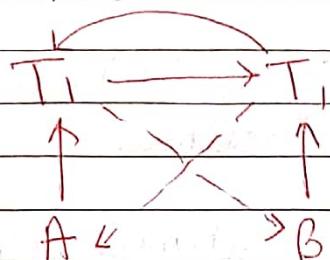
- Recoverability & cascadelness can be provided by not unlocking before commit.

Deadlock handling

- o 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.
- o If there exists a set of waiting transaction T_0, T_1, \dots, T_{n-1} such that

$$T_0 \rightarrow T_1, T_1 \rightarrow T_2, \dots, T_{n-1} \rightarrow T_0$$
 So, none transaction can progress in such situation.
- o System must have a proper methods to deal with deadlocks. Otherwise
 - ↳ In real time system it may lead to loss and memory.
 - ↳ Will reduce resource utilization and increase inefficiency.
- o There are two principle for dealing with deadlock problem

- o Prevention:- which ensure that system will never enter a deadlock state.
- o Detection and Recovery:- Allow system to enter into deadlock, then try to recover.



Deadlock Prevention:-

- To ensure no hold & wait, each transaction locks all the data item before it begins execution eg C-2 PL.
- 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 true protocol
 - ↳ it is difficult to predict what data items are required.
 - ↳ data item utilization will be low.
 - ↳ ordering of data item may be difficult for time taking to follow.

Wait-die: (Non-preemptive)

$T_i \rightarrow Q$ if $TS(T_i) < TS(T_j)$ then T_i must
 wait (T_i is older).
 if $TS(T_i) > TS(T_j)$ then T_i
 rollback (die).

Wound-wait: (preemptive)

$T_i \rightarrow Q$ if $T.S(T_i) > T.s(T_j)$ then T_i can wait
 (T_i is younger).
 if $T.s(T_i) < T.s(T_j)$ then T_j
 rollback.

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

11/02/2022

Query Language in DBMS

Orion

PAGE:
DATE: 11

After designing of database, i.e. ER diagram design them converting it into relational model followed by normalization and indexing. Now task is how to store, retrieve and modify data in database. Though here we will be concentrating more on the query part.

Query languages:-
language in which user request some information from database.

Procedural Query language:-
here user instruct the system to perform a sequence of operations in order to produce desired result. User tells what data to be retrieved and how to be retrieved.

Query language



procedural non-procedural
target + procedure target

Non-procedural query language:-
here user describe the desired information without giving the specific procedure for obtaining the information.

In practice we use RDBMS (practical implementation of relational model).

- SQL is used to write query on it.
- So Relational model is a conceptual / theoretical framework and RDBMS is its implementation.
- Relational Algebra (procedural) and Relational calculus (non-procedural) are mathematical system of query language used on Relational model.

Rational Model | RDBMS

RA, RL SQL

Algo Code

conceptual Reality

Theoretical Practical

Query language

procedural

Relational
Algebra

non-procedural

Relational
Calculus

↓
SQL

Structured Query
language

Relational Algebra

Orion

PAGE:

DATE:

- Relational algebra is one of the two formal query lang. associated with relational model.
- like, any other mathematical system it defines a number of operations and use relations (tables) as operands.
- Every operation in relation algebra take one or two relations as input argument and generate a single relation as a result without a name.
- R.o.A do not consider duplicacy as it is based on set theory.
- In each query we describes a step by step procedure for computing the desired result. So procedural QL.
- no. use of English keywords.

Basic

Fundamental

Select (σ)

Project (π)

Union (\cup)

Set difference ($-$)

Cartesian Product (\times)

Rename (δ)

Derived

Natural join (\bowtie)

Δ , \bowtie , \bowtie^*

Intersection (\cap)

Division (\div)

(can be derived from fundamental)

Select Operation in DBMS.

projection $\Pi \rightarrow$ column
select $\sigma \rightarrow$ row

- Select is a unary operator, so can take only one table at a time.
- It is a fundamental operator.
- Main idea of select is to find those tuples/rows in a relation which satisfies a given condition.
- Syntax σ condition/predicate (table-name)
- It has same function as of where clause in SQL.
- min no of tuples selected is 0.
- max no of tuples selected is all the tuples
[0, all]

branch	Account	Depositor	Customer
branch-name	account-no	curr_name	curr_name
branch-city	branch-name	account-no	curr_stnum
assets	balance		curr-city

loan	borrower
loan-no	curr_name
branch-name	loan_no
Amount	

↓ π

Student

S_id	Name	Branch
1	A	C.S
2	B	M.E
3	C	C.S
4	D	C.S.
5	E	E.E
6	F	E.C
7	G.	E.E.

Branch → C.S (student)

1. Find the details of account having balance $\geq 10,000$?

σ balance $\geq 10,000$ (Account).

2. Find the details of the customer who live in delhi?

σ cust city = 'delhi' (customer).

3. Find the details of those loans having amount ≤ 5000 and from north-delhi branch.

\rightarrow σ branch-name = 'N-D' \wedge amount ≤ 5000 (loan)

σ amount ≤ 5000 σ branch-name = 'N-D' (loan)

or

σ branch-name = 'N-D' (σ amount ≤ 5000 (loan))

4. Find those branch details which are in delhi as having assets more than 10,00,000?

σ branch-city = 'delhi' \wedge \vee assets $> 10,00,000$ (branch)

Project(Π)

- Unary operator, take one table at a time.
- Fundamental operation.
- Main idea of project is select desired columns.
- Syntax Π column-name (table-name)
- It's one of the select clause of SQL.

1. Find all branch-name of the bank?

Π branch-name (branch)

2. Find all account-no along with their balance?

Π account-no, balance (Account)

3. Find the name of all the customers who have loan?

Π cust-name (borrower)

4. Find all details about branch?

(branch)

Q. Find the names of all Students from C.S branch?

X $\Pi_{branch='C.S'}(\text{Tname} \text{ (student)})$

✓ $\Pi_{name}(branch='C.S' \text{ (student)})$.

Q. Find those account no where balance is less than 1000?

- $\Pi_{account_no}(\text{Tbalance} < 1000 \text{ (Account)})$

Q. Find those loan numbers which are from C.P branch with amount > 1000

- $\Pi_{loan_no}(\text{Tbranch_name} = 'C.P' \wedge \text{Amount} > 1000 \text{ (loan)})$

Q. Find branch-name and branch-city with assets more than 100,000?

$\Pi_{branch_name, branch_city}(\text{Tassets} > 1,00,000 \text{ (branch)})$

$$A \cap B = A - (A - B)$$

Orion

PAGE: / /
DATE: / /

Q. Find the name of customers who have a loan or an account or both?

- $T_{\text{customer}}(\text{Depositor})$
U

- $T_{\text{customer}}(\text{borrower})$

Q. Find the name of a branch who have accounts but not loan?

- $T_{\text{branch}}(\text{Account})$

- $T_{\text{branch}}(\text{loan})$

Q. Find the name of a customer who neither have a loan or an account?

- $T_{\text{customer}}(\text{customer})$

- $(T_{\text{customer}}(\text{Depositor}) U T_{\text{customer}}(\text{borrower}))$

Cartesian Product (X)

o Binary Operator takes two tables at a time.

o fundamental operation

o allows us to combine information b/w two tables.

$$A = \{1, 2, 3\}, B = \{a, b, c\}$$

$$A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$$

R_1 R_2 $R_1 \times R_2$

A	B	C	D	F	A	B	C	D	F
1	a	b	101	y	1	a	b	101	y
2	b	q	102	2	1	a	q	102	3
3	c	q	102	2	2	b	b	101	y

2	b	q	102	3
3	c	b	101	y
3	c	q	102	3

Trust_name (Account_no = Deposit_Acc
 & balance < 100.
 (Account X Depositor))

Q. Find those account numbers which are in Delhi?

Account_no

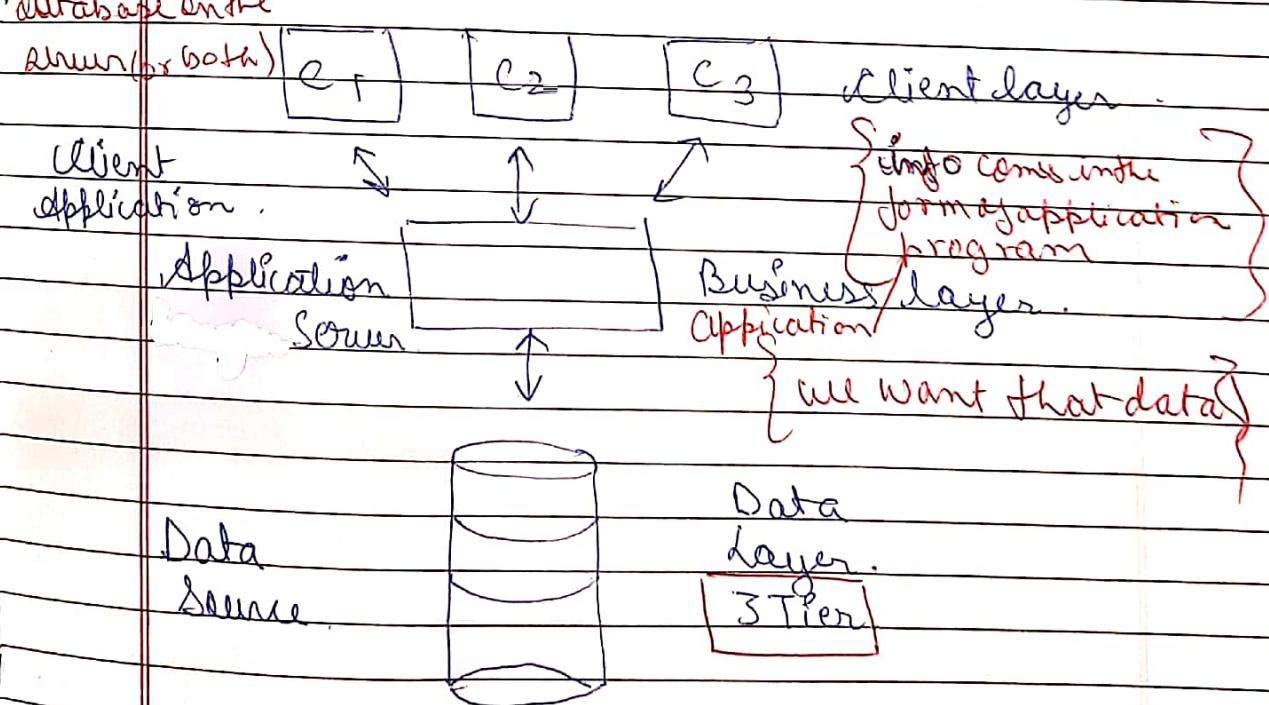
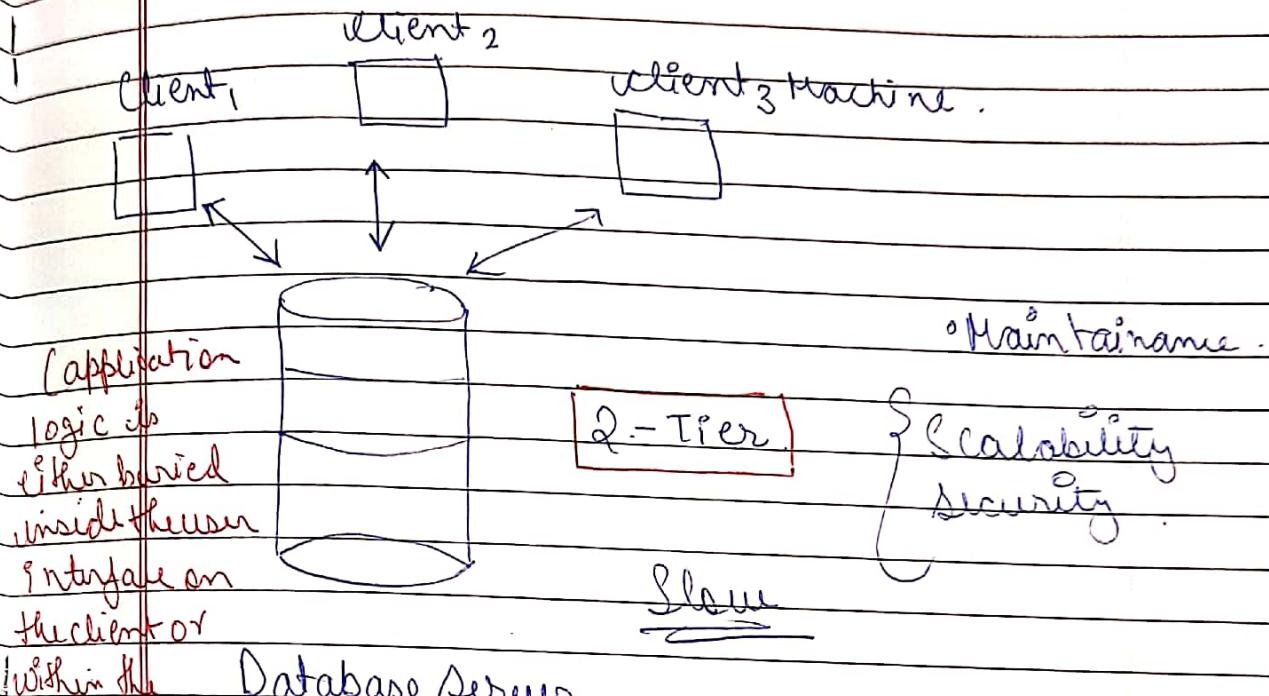
(Branch-city='delhi')
 Branch.branch-name
 = Account.branch-name
 (Branch X Account))

Q. Find those customer names who have a loan from a branch having assets more than 10,00,000.

Trust_name (Assets > 10,00,000 (Branch X Loan X)
 & Branch.branchname borrower
 = loan.branch-name
 & loan.loan-no
 = borrower.loan-no)

13/02/2022

Difference between Two-Tier and Three-Tier database architecture.



Past.

SQL

classmate

Date _____

Page _____

Introduction to SQL

- Number of query languages are in use.
- The most popular and widely accepted query language is SQL (Structured Query language).
- SQL is domain specific not general purpose, used in **design and management of data held in RDBMS**.
- Can do much more than just query a database, can define structure of database, modify data in database, specify security constraints and a number of other tasks.
- Based on relational algebra & tuple relational calculus.
- Name IBM later unchanged.
- IBM developed original version of SQL called SEQUEL, in Project Rum early 1970.
- Name later changed to SQL called because of trademark issues.
- In 1986, American National Standard Institute (ANSI) and ISO published SQL Standard called SQL-86.
- Next version, 1989, 1992, 1999, 2003, 2006, 2008, 2011 & most recently 2016.

Parts of SQL

Data Definition language (DDL)

provide commands for defining **relations**, **schemas**, deleting relations and modifying relational schemas.

CREATE

ALTER Integrity

DROP provide commands for integrity constraints

TRUNCATE removes the data stored in DB

COMMENT e.g. primary key can't be NULL,

RENAME foreign key reference.

View Definition

Commands for defining views e.g. sorting the result according to some attributes.

Authorization

Commands for access rights e.g. read only, R/W grants etc.

Data Manipulation language (DML)

Provides the ability to query information from the DB and insert tuples, delete tuples and modify tuple in the database

e.g. **insert**, **delete** commands etc.

(SQL building set has (134) commands)

→ (132 below)

(E008, PPAI, SPPI, P3A, etc.)

→ (132 mentioned in notes, some 2006)

Transaction control

SQL includes commands for specifying the beginning and ending of transactions e.g. commit, rollback etc.

Embedded SQL and Dynamic SQL

Some SQL statements can be embedded within general purpose programming language such as C, C++, Java etc.

* - Z -

- For data retrieval in SQL i/p & o/p both are relations.
- no of relation i/p to query will be at least one, but o/p will always be a single relation without any name unless specified.
- Basic structure of SQL query consists of three clauses.

select A₁, A₂... A_n (column name)
from R₁, R₂... R_m (relation name)
where P (condition).

→ It should be noted that select and from are mandatory and if not required, then not necessary to write where.

→ SQL in general is not case sensitive.

→ SQL allows duplicate in a i/p relation as well as in the result of SQL expression.

→ SQL query can be used to insert, update and delete data.

(insert, update, delete) → important part
(update, delete) need to know
which column to update or delete

Select clause :-

- used to pick **column required in result of the query out of all columns in i/p relation.**
- Order in which column are represented in the ~~order~~ Select clause will be same in which column will appear in the result.
- If all columns are required then * can be used.
- may also contain arithmetic expression involving operators like +, -, *, / etc. However it doesn't change the database.

Select with where :-

- Where clause in SQL is used to specify conditions / predicates.
- Where clause can have expressions involving comparison operators.
 $<$, \leq , $>$, \geq , $=$, \neq etc.
- SQL allows use of logical connectivities and, or, not, etc.
- Allows between comparison operator to simplify where clause which specify \geq to some value and \leq to some other value.
- Similarly you can 'not between'

Set Operations:-

- SQL provides set operations Union, Intersect, Intersect, except/minus / set difference which corresponds to the mathematical set operations \cup , \cap - resp.
- here union, intersect, minus automatically eliminates duplicate tuples.
- if we want to retain duplicate tuples add **all** after operator.
- set operations are only applicable b/w two relations if they are compatible i.e having same number of columns with correspondingly same domains

e.g.: (select cname
from Depositor)

union all | Intersect |
(select cname
from Borrower).

No of columns \rightarrow same.

All

to retain duplication

if no of columns are not equal then
all the relations need to be converted
into one relation using **cross product**.

Intersection and Union

Queries on multiple relations (Cartesian Product)

- So far wanted simple Query requires Only one relation.
- Queries often need more than one relation for information and to do that our simple idea is cartesian product.
- CP combine two relation into one by concatenation each tuple of first relation with every possible tuple of second relation.
- from clause by itself defines CP by structure relation listed.
- CP is commutative in nature, so order doesn't matter.
- if same attribute name appears in both relation we prefix the name of the relation from which attribute originally comes

Depositor . Account

C-n	A-no	A-no	bal	B-n
C-1	X42	X42	10	N-0
C-2	PQR	PQR	20	S-0

DepositorX Account

C-n	D.Ano	A.Ano	bal	B-n
C-1	X42	X42	10	N-0
X C-1	X42	PQR	20	S-0
X C-2	PQR	X42	10	N-0
✓ C-2	PQR	PQR	20	S-0

redundant tuples

select name from Account , Depositor
 where h.name = 'N-0'
 and Account . A-no = Depositor . A-no

Queries on multiple relations (Natural Join)

- Basically natural join work same as cartesian product but consider only those pair of tuples with the same value on those attribute that appear in the schemas of both relation.
- Notice that common attribute comes only once and prefix is also not required as we allow only same value.
- Commutative in nature.
- may lead to data loss if attributes to be match have different names, or some values occur only in one table.

Depositor

Account

C-n	A-no.
C-1	X42
C-2	PQR

A-no	bal	B-n
X42	10	N-D
PQR	20	S-D

D N-J A

C-n	A-no	bal	B-n
C-1	X42	10	N-D
C-2	PQR	20	S-D

Select C-name

from Account natural join Depositor
where Bname = 'N-D'

In 1st condition (1) R₁ and R₂ are disjoint.

R₁

	A	B
1	a	
2	b	
3	c	

R₂

	B	C
u	p	
d	q	x
e	r	x

R₁ × R₂

	A	R ₁ B	R ₂ B	C
1	a	c	p	
1	a	d	q	
1	a	e	r	
2	b			

R₁ Δ R₂

A	B	C
3	c	p

A	B	C
2	c	p

A ⊂ B ⊂ C

A	B	C
a	b	c
a	b	d
a	b	e

(A ∩ B) ⊂ C

∴ A ⊂ B ⊂ C

Join / inner join

is an operation which works same as Cartesian product if used alone.

- But if Join is used with keyword **'using'** it provides additional functionality.
- Provides ability to explicitly choose **columns** which must be used by join for comparison and removal of redundant tuples.
- If there are more than one tuple common between two tables but we do not want each of them to be considered then we specify which ones to be considered.

SQL Statement:

```
select t1.name
from department
join employee using (l_no)
where amount < 1000.
```

Join with on

- MySQL join with "on" allow us use a general predicate over the relations being joined.
- This predicate predicate is written like a where predicate except for the use of the keyword "on" rather than "where".
- Like Lop here we must specify which attribute must be matched.
- More powerful as predicate of where can also be written with "on".

- 'On' condition can express any SQL predicate.
- improves → readability.
- In case of outer join 'on' condition behave differently from where.

Select c_name
from dealer JOIN Borrower
on d_id = B.d_no.
And B.t < 1000.

Outer Join :-

One Problem with Natural Join is only those value that appear in both relation will manage to reach final table.

but if some data is explicitly in table one or table two then that will be lost.

→ There are three form of outer join.

→ left outer join (left join).

→ right outer join (right join).

→ full outer join (full join / complete join).

(Answers → as)

Joins in SQL

◦ (Inner) Join :- returns records that have matching values in both tables.

◦ Left Outer Join :- returns all records from the left table and matched records from the right table.

◦ Right Outer Join :- right — left

◦ Full Outer Join :- returns all the records when there is a match in either left or right table.

R_1 R_2

A	B	C	D	E
1	b	q	x	
2	q	r	y	
3	r	s	z	

 $R_1 \times R_2$

A	$R_1 \cap B$	$R_2 \cap B$	C
1	b	q	x
2	q	r	y
3	r	s	z
2	q	r	y
3	r	s	z
3	r	s	z

 $R_1 \setminus R_2$ and $R_1 \setminus R_2$

A	B	C	A	B	C
2	q	x	1	b	null
3	r	y	2	q	x
3	r	y	3	r	y

 $R_1 \Delta R_2$ $R_1 \nabla R_2$

A	B	C	A	B	C
2	q	x	1	b	null
3	r	y	2	q	x
null	s	z	3	r	y
null	s	z	3	r	y

String Operations

- SQL specifies strings by enclosing them in single quotes "knowledge gate"
- Equality operation on string may or may not be case sensitive.
e.g. Most of the db system are case sensitive but MySQL and SQL server are not. We can change the behaviour in settings.
- Minimum number of operations available on string such as concatenation, substring, finding length of strings, converting strings to uppercase or lowercase etc.
- Most set of operations may change from system to system.
 - + pattern matching can be performed on a string using "like" operator.
- Two special characters -
 - Percent (%) : matches any substring.
 - Underscore(_) : matches with any character.
- pattern matching is case sensitive.
- if single quote is part of a string, it must be specified by two single quotes.
- for pattern which includes special pattern we can escape character just before the special char.

Ordering the display of Tuples -

SQL gives the user some control over the order in which tuples in a relation are displayed.

- Order by clause causes the tuple in the result of a query to appear in sorted order.
- By default order by clause lists items in ascending order. We may specify asc for ascending order and desc for descending order.
- Ordering can be performed on multiple attributes i.e., if third attribute or the first attribute is sorted according to second attribute.
- ~~Find the total marks obtained by students in each subject.~~

SQL DELETE STATEMENT

The DELETE statement is used to delete existing records in a table.

* DELETE SYNTAX *

DELETE FROM table-name
WHERE
condition;

SQL GROUP BY Statement

The GROUP BY Statement groups rows that have the same values into summary rows like "find the number of customers in each country".

The GROUP BY Statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result set by one or more columns.

SQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword can't be used with aggregate functions.

+
SELECT icolumn_name(s)
FROM table-name.

SQL TOP, LIMIT, FETCH FIRST or ROWNUM Clause.

The SELECT TOP clause is used to specify the number of records to return.

Birthday Para.

SQL TOP clause is used to return a limited number of records from a query. It is often used in conjunction with ORDER BY clause to return the top n records based on some criteria. The syntax for using TOP clause is:

```
SELECT TOP n column_name(s) FROM table-name;
```

For example, if we want to retrieve the top 10 rows from the table "EMPLOYEE" ordered by salary in descending order, we would write:

```
SELECT TOP 10 * FROM EMPLOYEE ORDER BY SALARY DESC;
```

This will return the top 10 highest salary employees from the database.

13/2/2021

classmate

Date _____
Page _____

SQL Joins.

A Join clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL JOINS

- (INNER) JOIN :-

Returns records that have matching values in both tables.

- LEFT(OUTER) JOIN :-

Returns all records from the left table, and the matched records from the right table.

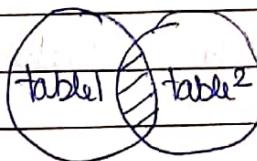
- RIGHT(OUTER) JOIN :-

Returns all records from the right table, and the matched records from the left table.

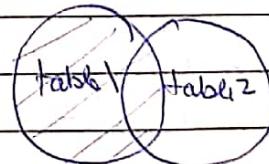
- FULL(OUTER) JOIN :-

Returns all records where there is a match in either left or right table.

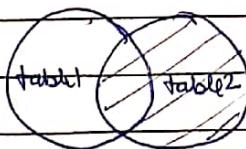
INNER JOIN



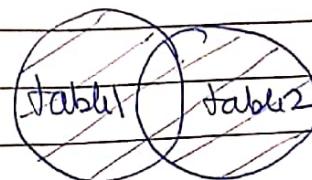
LEFT JOIN.



RIGHT JOIN.



FULL OUTER JOIN.



INNER JOIN EXAMPLE.

→ Select Orders.OrderID, Customers.CustomerName
From Orders
INNER JOIN Customers
ON Orders.CustomerID = Customers.CustomerID;

LEFT JOIN EXAMPLE / RIGHT JOIN

→ Select Customers.CustomerName, Orders.OrderID
From Customers
LEFT JOIN Orders / RIGHT JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

OUTER JOIN EXAMPLE.

→ SELECT Customers.CustomerName,
Orders.OrderID
From Customers
FULL OUTER JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

Self Join

A self join is a regular join, but the table is joined with itself.