Book: Korth, Navathe, Ullman, Rama Krishna

(Vijay agarwal sir)

DBMS [9-11 M]

- Introduction & ER - Model — 1-2 M
- FD & Normalization — 3-4 M
- Transaction & Concurrency control — 3-4 M
- File org & Indexing — 2 M
- query language — 2-3 M

RDBMS (relational database management system)

→ **data**: raw facts

→ **Information**: meaningfull data (preprocessed data)

→ **database**: collection of logically related data

- RDBMS —contains→ Relation → Table
  - Row : Tuple / Record
  - column : field / attribute

  ↓ system in which data is represented in form of relation/table

- eg table : Student

Sid	Sname	marks
S_1	A	9
S_2	B	10
S_3	C	9
S_4	D	10

Arity : # attributes/fields

Cardinality : # tuples/records

Arity = 3 , Cardinality = 4

Scanned by CamScanner

logical structure of table / template of table

Relational Schema : Table abstraction

eg. STUDENT (sid sname marks)

eg. S1 A 9
S2 B 10

Relational Instance : set of records

* **FD (functional dependency)** $X \rightarrow Y$ (Y is functionally dependent on X)

R(XY)

X	Y
1	5
2	6
3	7
4	8
2	6
4	9

(i)

R(XY)

X	Y
1	5
2	6
3	9
4	2

(ii)

R(XY)

X	Y
1	9
2	9
3	9
4	9

(iii)

R(XY)

X	Y
x_1	y_1
x_2	y_2
x_3	y_1
x_4	y_2

(iv)

R(XY)

X	Y
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4
x_2	y_2
x_3	y_3

(v)

R(XY)

X	Y
1	6
2	7
3	8
4	9
5	10
2	7
5	10

(vi)

Q) In which of the above 6 relations $X \rightarrow Y$ is satisfied?

$X \rightarrow Y$: X functionally determine Y or Y is functionally depend on X

Note:

FD definition

Q) Let R be the relational schema & X, Y be the attribute set of R & t_1, t_2 any two tuple such that

If $\rightarrow$

Scanned by CamScanner

$\boxed{t_1.x = t_2.x \text{ then } t_1.y = t_2.y \text{ must be same}}$

Note:

o) In $X \to Y$, whenever X value repeat then corresponding y value must be same.

o) So (i) ✗ (not $X \to Y$) → here 4, 8, 9 → different values
(ii) ✓ ($X \to Y$)
(iii) ✓ (")
(iv) ✓ (")
(v) ✓ (")
(vi) ✓ (")

(means at one x only one y, normal property of function ✓ ✗)

* Types of FD (functional dependencies)

1. Trivial FD → always valid
2. Non-trivial FD
3. Semi non-trivial FD

1 * Trivial FD (always valid)

$X \to Y$ is trivial if $X \supseteq Y$ (Right hand side attribute must be equal to or part of left side attribute)

X is superset equal to Y

Eg) Sid Sname → Sid
Sid Sname → Sname
Sid Sname → Sid Sname
AB → A
AB → B
AB → AB

Scanned by CamScanner

if & only if => iff

② * **Non-trivial FD**

$X \to Y$ is non trivial if and only if $X \cap Y = \phi$ && $X \to Y$ must satisfy FD definition

↳ nothing common b/w X & Y

$A \to C$

$Sid \to Sname$

③ * **Semi-Non trivial FD**

(combination of trivial & non-trivial FD)

$Sid \to Sid\ Sname$

↳ $Sid \to Sid$ (trivial) } semi non trivial FD

decomposition rule $Sid \to Sname$ (non-trivial)

Question (5 instances are there)

R(ABC)

A	B	C
1	1	3
1	2	3
2	1	3
2	2	3
2	1	3

⇒ find valid non-trivial FD

$X \cap Y = \phi$

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

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

$A \to BC$ $B \to AC$ $C \to AB$

$AB \to C$, $BC \to A$, $AC \to B$

↓

now check the FD definition

~~Ans → 12~~

⇒ $A \to B$ ✗ $B \to A$ ✗ $C \to A$ ✗

⇒ $A \to C$ ✓ $B \to C$ ✓ $C \to B$ ✗

⇒ $A \to BC$ ✗ $B \to AC$ ✗ $C \to AB$ ✗

⇒ $BC \to A$ ✗

⇒ $AB \to C$ ✓

⇒ $AC \to B$ ✗

Ans → 3

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

Scanned by CamScanner

o) (Trivial FD's are always valid
need not to check for FD definition)

o) $X \not\to Y$, X does not functionally determine Y
valid

Question

A	B	C
1	1	1
1	2	1
2	1	2
2	1	3
1	3	3

o) find # non-trivial FD

$A \to B$ ✗ $B \to C$ ✗ $C \to AB$ ✗
$A \to C$ ✗ $B \to AC$ ✗ $AB \to C$ ✗
$A \to BC$ ✗ $C \to A$ ✗ $BC \to A$ ✓
$B \to A$ ✗ $C \to B$ ✗ $AC \to B$ ✗

Ans → 1 ($BC \to A$) or $CB \to A$

PSU

P	Q	R
6	6	7
6	7	7
7	3	4
8	3	4

$P \to Q$ ✗ $Q \to PR$ ✗ $QR \to P$ ✗
$P \to R$ ✓ $R \to P$ ✗ $RP \to Q$ ✗
$P \to QR$ ✗ $R \to Q$ ✗
$Q \to P$ ✗ $R \to PQ$ ✗
$Q \to R$ ✓ $PQ \to R$ ✓

Ans → ($P \to R$, $Q \to R$, $PQ \to R$)

Gate given the following relation instance

X	Y	Z
1	4	2
1	5	3
1	6	3
3	2	2

o) which of the following FD's are satisfied by the Instances?

a) $XY \to Z$ & $Z \to Y$
b) $YZ \to X$ & $X \to Z$
c) $XZ \to Y$ & $Y \to X$
d) $YZ \to X$ & $Y \to Z$

Gate R(ABC)

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

a) $A \to B$ & $B \to C$
b) $A \not\to B$ & $B \to C$
c) $A \to B$ & $B \not\to C$
d) $A \not\to B$ & $B \not\to C$

Scanned by CamScanner

* **Attribute closure $[X]^+$** (used to find Candidate key)

→ The set of all possible attribute which is logically determined by X, called Attribute closure of X

R(ABCDE) [A→B, B→C, C→D, D→E] → given FD's

$[A]^+ = [ABCDE]$
$[B]^+ = [BCDE]$
$[C]^+ = [CDE]$
$[D]^+ = [DE]$
$[E]^+ = [E]$

Question R(ABCDE) [AB→C, C→D, B→E] → given FD's

$[A]^+ = [A]$
$[B]^+ = [BE]$
$[C]^+ = [CD]$
$[D]^+ = [D]$
$[E]^+ = [E]$
$[AB]^+ = [ABCDE]$
$[BC]^+ = [BCDE]$

ABCDE
AB→C
B→E
C→D

Gate AB→CD, AF→D, C→G, F→E, G→A
which of the following is / are false

a) $[CF]^+ = [ACDEFG]$ — CFGEAD
b) $[BG]^+ = [ABCDG]$ — BGACDG
c) $[AF]^+ = [ACDEFG]$ — AFED
d) $[AB]^+ = [ABCDFG]$ — ABCDG.

Scanned by CamScanner

o) Key : that determine all the attributes of Relation.

Question R(ABCDEF) $A \to B$, $DE \to F$, $AF \to C$, $B \to D$

$[A]^+ = [ABD]$
$[B]^+ = [BD]$
$[FB]^+ = [FBD]$
$[AB]^+ = [ABD]$

→ primary key, candidate key, alternative key, ...

* **Super Key** (every key is a super key)

→ Let R be the relational schema with FD set F & X be the set of attributes over relation R.

If X^+ determine all the attribute of R, then X is a super key.

Eg) R(ABCDE) $A \to B$, $B \to C$, $C \to D$, $D \to E$

$[A]^+ = [ABCDE]$
$[B]^+ = [BCDE]$
$[C]^+ = [CDE]$
$[D]^+ = [DE]$
$[E]^+ = [E]$

o) from closure of A we can determine all the attributes of relation R
o) So A is a super key here

Note: Any super set of super key is also super key
o) If A is super key, then any superset of A is also super key.

A
AB
AC
⋮

ACD
⋮
ABCDE

⇒ all super key (A should be present)

Scanned by CamScanner

o) All keys are super key except foreign key

* Candidate key

→ minimal of super key

→ If any proper subset of super key is also super key then proper subset is called candidate key.

→ If none of the proper subset of super key is not also a super key then that key is super key & candidate key also.

Q. ① R(ABCDE) $AB \to C$, $C \to D$, $B \to E$

$[AB]^+ = [ABDCE]$

getting all attributes so AB is a super key

AB

proper subset

A B

$[A]^+ = [A]$ $[B]^+ = [BE]$

here AB is candidate key.

o) prime attribute : [A, B]

o) non prime attribute : [C, D, E]

② R(ABCDE) $AB \to C$, $C \to D$, $B \to EA$

$[AB]^+ = [ABCDE]$

AB is super key

prime attr : [B]

non " " [A, C, D, E]

AB

proper subset

A B

$[A]^+ = [A]$ $[B]^+ = [BEACD]$

here B is candidate key

Scanned by CamScanner

o) All keys are super key

o) Key: that define all the attributes of relation.

Note: Every candidate key is a super key, vice versa not true. (every super key is not a candidate key)

Super key
|
| minimal
→ Candidate key

o) minimum to minimum attribute or set of attribute to determine all the attributes of relation R is called as candidate key.

eg) If saying ABC is a candidate key then the minimum set of attribute ~~of~~ to determine the all the attribute is ABC

A, B, C, AB, BC, AC, → Can't determine all attributes of relation R

o) primary key is one of the candidate key.

o)

Super key
|
| minimal
→ Candidate key (let 3 candidate key are there)

(If B is candidate key then we can't say its super set (BA, BD, BCD, ...) is candidate key ∵ candidate key is minimum to minimum)

① one select → primary key

② remaining → alternative / secondary key

eg) In Bank account no is primary key.
(If account no is forgotten by someone, so to access the account the other key like name, adhar no etc are alternative key)

Scanned by CamScanner

* **prime / key attribute** : set of attributes that belongs (present) in any candidate key.

* **non prime / non key attribute**: set of attributes that not belong (present) in any candidate key.

Imp)

* **Finding multiple candidate key** (How to find candidate key)

Procedure: first find any one candidate key then that attribute is prime attribute

o) If. $\boxed{X_{Attribute} \longrightarrow [\text{Prime Attribute}]}$

any X attribute determine that prime attribute then multiple candidate key are there

o) Let E is candidate key
Prime attribute = [E]
$X_{Attribute} \rightarrow [E]$
eg $F \rightarrow E$ here F is candidate key
$FD \rightarrow E$ here we can't say FD is candidate key (candidate key is min to min)
F D
now check for F, D.

$G \rightarrow ED$ G is candidate key
$GA \rightarrow ED$ (can't directly say GA is candidate key, check for G & A individually)
G A

Scanned by CamScanner

Question R(ABCDE) $AB \to C$, $C \to D$, $B \to EA$

find candidate key.

$[AB]^+ = [ABCDE]$

AB is a super key. (jab tak todhenge nahi jab tak bolenge nahi)

A B

$[A]^+ = [A]$ $[B]^+ = [BEA\ CD]$

B → candidate key

prime attribute : [B,

X attribute → B

there is no such relation

So only one candidate key here i.e B

prime attribute [B]

Question R(ABCDE) $AB \to C$, $C \to D$, $D \to E$, $B \to A$, $C \to B$

$[AB]^+ = [ABCDE] = [ABCDE]$

AB → SK

A B

$[A]^+ = [A]$ $[B]^+ = [BACDE]$

B → candidate key.

X attribute → B

C → B

So B & C are C.K

prime attribute [B, C]

now.

AB → C

A B → CK

Ans. Candidate key : B, C P.A [B, C]

Scanned by CamScanner

Question R(ABCDE) $A \to BCDE$, $BC \to ADE$, $D \to E$

$[BC]^+ = [BCADE]$

$BC \to$ S.K

B C

$[B]^+ = [B]$ $[C]^+ = [C]$

so BC is C.K

~~A →~~ B prime attribute [B, C

$A \to BCDE$

A is C.K

$BC \to ADE$

final Candidate key, ~~A, B, C~~ [A, BC]

Question R(ABCD) $A \to B$, $B \to C$, $C \to D$, $D \to A$

$[A]^+ = [ABCD]$

A is C.K

$D \to A$

D is C.K

$C \to D$

C is C.K

$B \to C$

B is C.K

So here A, B, C, D all are C.K

Question R(ABDLPT)

$B \to PT$, $T \to L$, $A \to D$

$[AB]^+ = [ABDPTL]$

Scanned by CamScanner

AB is S.K

A B

$[A]^+ = [AD]$ $[B]^+ = [BPTL]$

AB is C.K prime att = [A, B]

only one C.K i.e AB

Question R(ABCDEF) $AB \to C$, $C \to D$, $D \to E$, $E \to F$, $F \to B$

$[AB]^+ = [ABCDEF]$

AB is S.K

$[A]^+ = [A]$ $[B^+] = [B]$

AB is C.K

prime attribute [A, B, F, E, D, C

$F \to B$

F is C.K

$E \to F$

E is C.K

$D \to E$

D is C.K

$C \to D$

C is C.K

$AB \to C$

Ans C.K : AB, F, E, D, C

$[AF]^+ = [AFBCDE]$

$[F]^+ = [FB]$

So AF is C.K

$E \to F$

$[AE]^+ = [AEFCDB]$

$[E]^+ = [EFB]$

So. AE is C.K

$D \to E$

$[AD]^+ = [ABCDEF]$

$[D]^+ = [DEFB]$

So AD is C.K

$C \to D$

$[AC]^+ = [ABCDEF]$

$[C]^+ = [CDEFB]$

So AC is C.K

Ans: AB, AF, AE, AD, AC

PA [A, B, C, D, E, F]

Scanned by CamScanner

Question R(ABCDEFGHIJ) $AB \to C$, $A \to DE$, $B \to F$, $F \to GH$, $D \to IJ$

$[AB]^+ = [ABCDEFGHIJ]$

AB is S.K

$[A]^+ = [ADE]$

$[B]^+ = [BFGH]$

AB is C.K

prime attribute = [A, B]

Only one C.K i.e AB

Question R(ABCDEF) $AB \to C$, $C \to D$, $D \to E$, $E \to F$, $F \to A$

$[AB]^+ = [ABCDEF]$

$[A]^+ = [A]$

$[B]^+ = [B]$

AB is C.K , prime attri [A, B, F, E, D, C]

$F \to A$

$[FB]^+ = [AFBCDE]$

$[F]^+ = [FA]$

FB is C.K

$E \to F$

$[EB]^+ = [EBFACD]$

$[E]^+ = [EFA]$

EB is C.K

$D \to E$

[DB is C.K]

$C \to D$

$[CB]^+ = [ABCDEF]$

Scanned by CamScanner

$[C]^+ = [CDEFA]$

CB is C.K

Ans.

C.K are AB, FB, EB, DB, CB

Imp////

Question R(ABCDE) $AB \to C$, $BC \to D$

$[AB]^+ = [ABCD]$

E is not a part of FD, whenever any attribute not present in FD, then make a part of C.K

$[ABE]^+ = [ABCDE]$

ABE is S.K

- $[AB]^+ = [ABCD]$
- $[AE]^+ = [AE]$
- $[BE]^+ = [BE]$

So

ABE is C.K, prime att [A, B, E]

only one C.K i.e ABE

Question (2.M) R(ABCDEFGH)

$AB \to C$, $AC \to B$, $AD \to E$, $BC \to A$, $B \to D$, $E \to G$

$[AB]^+ = [ABCDEG]$

F, H not there in FD

$[ABFH]^+ = [ABCDEFGH]$

ABFH is S.K

- ~~[AB]+~~ $[AFH]^+$
- $[BFH]^+$

ABFH is C.K, prime att [A, B, F, H], C

Scanned by CamScanner

$BC \rightarrow A$

$[BCFH]^+ = [ABCDEFGH]$

~~BCFH~~

$[BFH]^+$

$[CFH]^+ = [CFH]$

BCFH is C.K

$AC \rightarrow B$

$[ACFH]^+ = [ABCDEFGH]$

ACFH is C.K

$AB \rightarrow C$

Ans

C.K: ACFH, BCFH, ABFH

Question

(Imp) R(ABCDE) $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$

$[AB]^+ = [ABCDE]$

AB is S.K

$[A]^+ = [ABCDE]$

$[B]^+ = [BD]$

A is C.K

$E \rightarrow A$

$[E]^+ = [EABCD]$

E is C.K

$CD \rightarrow E$

$[CD]^+ = [CDEAB]$

CD is S.K

$[C]^+ = [C]$

$[D]^+ = [D]$

Scanned by CamScanner

CD is C.K prime att (A, E, C, D, B

$A \to BC$ ✓

$B \to D$

~~$[B]^+ = [BD]$~~

$[CB]^+ = [CBDEA]$

CB is S.K

CB is C.K

$A \to BC$ ✓

Ans : A, E, CD, CB four C.K

Question (2 M) R(ABCDEFGH) $AB \to CD$, $B \to EG$, $F \to H$, $C \to EF$, $H \to A$, $G \to B$, $A \to B$

$[AB]^+ = [ABCDEFGH]$

AB is S.K

$[A]^+ = [ABCDEFGH]$

$[B]^+ = [BEG]$

A is C.K

$H \to A$

H is C.K

$F \to H$

F is C.K prime att (A, H, F, C

$C \to EF$

$[C]^+ = [CEFHABDG]$

C is C.K

Scanned by CamScanner

$AB \to CD$

$[AB]^+$ ✓ done

Ans: CK: A, H, F, C

Question R(ABCDE) $AB \to C$, $C \to D$, $D \to E$, $E \to A$, $D \to B$

$[AB]^+ = [ABCDE]$

AB is S.K

$[A]^+ = [A]$

$[B]^+ = [B]$

AB is C.K prime att [A, B, E, D, C

$E \to A$

$[EB]^+ = [EBACD]$

$[E]^+ = [EA]$

EB is C.K

$D \to B$

$[AD]^+ = [ADEBC]$

$[D]^+ = [DEA]$

AD is C.K

~~$C \to D$~~ $D \to E$

$[DB]^+ = [DBEAC]$

DB is C.K

$C \to D$

$[CB]^+ = [CDBAE]$

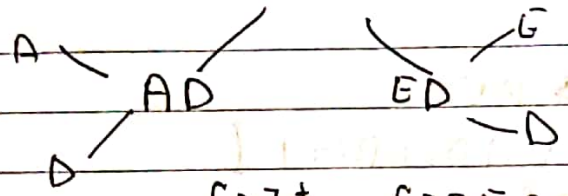
CB is C.K

Scanned by CamScanner

$AB \to C$ ✓

Any, C.K ~~are~~ AB, ~~EB~~, AD, ~~DB~~, ~~CB~~

$D \to B$



$[D]^+ = [DBEAC]$

D is C.K

$C \to D$

$[C]^+ = [ABCDE]$

C is C.K

$AB \to C$ ✓ done

Ans: 4 C.K: AB, EB, C, D

Question

$R(ABCDEFG)$

$AB \to CDEF$

$C \to ADE$

$D \to EBF$

$F \to DA$

$BE \to AF$

$[AB]^+ = [ABCDEF]$

$[ABG]^+ = [ABCDEFG]$

ABG is super key

Scanned by CamScanner

$[AG]^+ = [AG]$

$[BG]^+ = [BG]$

ABG is candidate key ✓

prime attr = [A, B, G, C, F, E, D

$C \rightarrow ADE$

$[CBG]^+ = [ADECBGF]$

$[CG]^+ = [CGADEBF]$

CG is candidate key ✓

$F \rightarrow DA$

$[FBG]^+ = [FBGDACE]$

$[FG]^+ = [DAFGEBC]$

FG is C.K ✓

B, E $\rightarrow$ BE $\rightarrow$ AF

$[BEG]^+ = [BEGAFCD]$

~~$[BE]^+ = [BEAFCDE$~~

$[EG]^+ = [EG]$

BEG is C.K ✓

$D \rightarrow EBF$

$[ADG]^+ = [ADGEBFC]$

$[DG]^+ = [DGEBFAC]$

DG is C.K ✓

$AB \rightarrow CDEF$

$D \rightarrow EBF$

$[DG]^+ = [DGEBFAC]$

DG is C.K ✓

Scanned by CamScanner

$D \rightarrow$ $C \rightarrow ADE$

$[BCG]^+ = [ABCDEFG]$

$[CG]^+ = [CG\ ADE\ BF]$

CG is C.K ✓

$D \rightarrow EBF$

$[BDG]^+$.

CG · DG is C.K ✓

$C \rightarrow ADE$

$[CG]$ ✓

$F \rightarrow DA$

$[FG]^+ = [$ ✓

C.K ABG, CG, FG, BFG, DG,

(S C.K)

Gate 1995

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

determine the minimal keys of relation R

$[AB]^+ = [ABC]$

AB is S.K

$[A]^+ = [A]$

$[B]^+ = [B]$

AB is C.K

$C \rightarrow A$

$[CB]^+ = [CBA]$

$[C]^+ = [CA]$

CB is C.K Ans AB, CB

Scanned by CamScanner

Gate 1999 $R = (A, B, C, D, E, F)$

$C \to F$, $E \to A$, $EC \to D$, $A \to B$

which of the following is a key of R?

a) CD c) AE

~~b)~~ EC d) AC

$[CD]^+ = [CDF$

$[CE]^+ = [CEFABD]$

Gate 2005 F1, F2, F3, F4, F5

$F1 \to F3$, $F2 \to F4$, ~~$F1F2 \to F5$~~ $(F_1, F_2) \to F5$

or

$F1F2 \to F5$

find keys

$[F_1F_2]^+ = [F_1F_2F_3F_4F_5]$

F_1F_2 is S.K

$[F_1]^+ = [F_1F_3]$

$[F_2]^+ = [F_2F_4]$

F_1F_2 is C.K

Gate 2005

$R(A, B, C, D, E, H)$

$A \to B$, $BC \to D$, $E \to C$, $D \to A$

what are the CK of R?

a) AE, BE c) AEH, BEH, BCH

b) AE, BE, DE ~~d)~~ AEH, BEH, DEH

~~$[AB]^+ = [AB$~~

~~$[AB$~~ EH should be present in key

Gate 2006

Consider a relation R with five attributes V, W, X, Y & Z. The following functional dependencies hold: $VY \to W$, $WX \to Z$, $ZY \to V$, which of the following is a c.k for R

a) VXZ c) VWXY

b) VXY d) VWXYZ

X, Y should be present in key

$[VWXY]^+ = [VWXYZ]$

$[VXY]^+ = [VXYWZ]$

& $[VWXYZ]^+ = [VWXYZ]$

c.k means minimal so VXY is Ans

Gate 2013

R(ABCDEFGH)

$CH \to G$, $A \to BC$, $B \to CFH$, $E \to A$, $F \to EG$

How many c.k does relation R have

a) 3 c) 5

b) 4 d) 6

$[ABD]^+ = [ABDCFHEG]$

ABD is c.k

$[AD]^+ = [ABCDFHEG]$

$[BD]^+ = [BDCFHEGA]$

prime attr [A, B, D, E, F AD & BD are c.k

$E \to A$

$[ED]^+ = [EDABCFHEG]$

ED is c.k

$A \to BC$

[AD is c.k]

Scanned by CamScanner

$\{A, B\} \Rightarrow AB$

$F \rightarrow EG$

$[FD]^+ = [FDEG\ A\ BC\ H]$

FD is c.k ✓

$B \rightarrow CFH$

$[BD$

Ans] AD, BD, ED, FD

Gate 2014

$R(E, F, G, H, I, J, K, L, M, N)$

$EF \rightarrow G \quad EH \rightarrow KL \quad L \rightarrow N$

$F \rightarrow IJ \quad K \rightarrow M$

What is key for R

a) $\{E, F\}$ c) $\{E, F, H, K, L\}$

b) $\{E, F, H\}$ d) $\{E\}$

~~$[EF]^+ = [EFG$~~ E, F, H,

$[EFH]^+ = [EFHG\ KLN\ IJM]$

Gate 2014

A prime attribute of a relation scheme R is an attribute that appears

a) in all c.k of R c) in foreign key of R

b) in some " " d) only in the primary key of R

Gate 2016

Which of the following is NOT a superkey in a relational schema with attributes V, W, X, Y, Z are & primary key VY?

Scanned by CamScanner

a) vxyz c) vwxy (∵ vy should be present in super key)
b) vwxz d) vwxyz

Gate 2015

X (P, Q, R, S, T, U)
PR → ST
PSU → QR
which of the following is the trivial FD
a) PR → ST c) PS → S
b) PR → RT d) PSU → Q

---

* Membership set

o) Let F be the FD set and X → Y any FD is member of FD set F if & only if X → Y logically implied in F

o) X → Y is logically implied means from the closure of X ([X]+) determine Y, then we can say that X → Y is logically implied / member / valid FD

eg) F: [X → Y, Y → Z]
Check X → Z is logically implied / member / valid FD or not.
[X]+ = [XYZ]
yes X → Z is valid FD

Questions

① F [A → B, B → C]
Check A → C, C → B

$[A]^+ = [ABC]$

$A \to C$ ✓

$[C]^+ = [C]$

$C \to B$ ✗

② $[A \to B, B \to C]$

check $AB \to C$

$[AB]^+ = [ABC]$

$AB \to C$ ✓

③ $AB \to C, B \to C$

Check $A \to C$

$[A]^+ = [A]$

$A \to C$ ✗

④ $AB \to C, A \to D, CD \to E$

Check $AC \to E$

$[AC]^+ = [ACDE]$

$AC \to E$ ✓

⑤ $A \to B, B \to C,$

Check $A \to C$

$[A]^+ = [ABC]$

$A \to C$ ✓

⑥ $AB \to C, BC \to D$

Check $A \to C$

$[A]^+ = [A]$

$A \to C$ ✗

Scanned by CamScanner

Gate 2005 A, B, C, D

$A \to B$ $\quad$ $B \to D$

$A \to C$ $\quad$ $E \to A$

$CD \to E$

Which of the following fD is not implied by the above set

a) $CD \to AC$ $\quad$ c) $BC \to CD$

b) $BD \to CD$ $\quad$ d) $AC \to BC$

$[CD]^+ = [CDEAB$

$[BC]^+ = [BCD$

$[BD]^+ = [BD]$

$[AC]^+ = [ACB$

---

o) Equality b/w two FD set $\quad$ F, G → 2 FD set

F : [ ] $\quad$ G : [ ]

There is equality b/w two set iff

& F cover G : True
G cover F : True } $F \equiv G$

both should be true for $F \equiv G$

o) meaning of F cover G true : All dependencies (FD's) of G should be logically implied in F

or

F cover all the FD's of G

F cover G :	True	false	true	false
G cover F :	false	true	true	false
	$F \geq G$	$G \geq F$	$F \equiv G$	uncomperable

Scanned by CamScanner

Question

F: [A→B, B→C, BD→E] G: [AB→C, A→B, B→D, D→E, B→C]

F cover G	closure in F
AB→C	$[AB]^+ = [ABC]$ ✓
A→B	$[A]^+ = [B]$ ✓
B→D	$[B]^+ = [BC]$ ✗
D→E	$[D]^+ = [D]$ ✗
B→C	$[B]^+ = [BC]$ ✓

false

G cover F	closure in G
A→B	$[A]^+ = [ABCDE]$ ✓
A→C	$[B]^+ = [BC]$ ✓
BD→E	$[BD]^+ = [BDE]$ ✓

$\boxed{G \geq F}$

Question

F [AB→CD, B→C, C→D]

G [AB→C, AB→D, C→D]

F cover G	G cover F
$[AB]^+ = [ABCD]$ ✓	$[AB]^+ = [ABCD]$ ✓
$[AB]^+ = [ABCD]$ ✓	$[B]^+ = [B]$ ✗
$[C]^+ = [CD]$ ✓	$[C]^+ = [CD]$ ✓

$F \geq G$

Question (PSG)

F [A→C, AC→D, E→AD, E→H]

G [A→CD, E→AH]

Scanned by CamScanner

if $F \geq G$ and $G \geq F$
So $F \equiv G$

F cover G
$[A]^+ = [ACD]$ ✓
$[E]^+ = \cancel{[ADCP]}$ ✗
$[EHAD]$ ✓

G cover F
$[A]^+ = [ACD]$ ✓
$[AC]^+ = [ACD]$ ✓
$[E]^+ = [EAHCD]$ ✓
$[E]^+ = [AEHCD]$ ✓

~~$G \geq F$~~ $F \equiv G$
(2 FD are equal)

---

properties of decomposition
- → Lossless join decomposition
- → dependency preserving decomposition

① **loss-less join decomposition** : Let R be the relational schema with instance r, decomposed into sub relation $R_1, R_2, \ldots R_n$ with instance $r_1, r_2, \ldots r_n$ respectively.

$R \to R_1, R_2$ (decomposed)

$R_1 \bowtie R_2 \equiv R$
↳ natural join

$$\boxed{\text{If } r_1 \bowtie r_2 \bowtie r_3 \bowtie \ldots \bowtie r_n \equiv R \text{ loss less join decomposition}}$$

$$\boxed{\text{If } r_1 \bowtie r_2 \bowtie r_3 \ldots \bowtie r_n \geq R \text{ lossy join decomposition}}$$

$\bowtie$ ↳ natural join

eg. R(ABC)

A	B	C
1	2	1
2	2	2
3	1	2

R(ABC) → $R_1(AB)$ (decomposed)
R(ABC) → $R_2(BC)$

$R_1(AB)$

A	B
1	2
2	2
3	1

$R_2(BC)$

B	C
2	1
2	2
1	2

Scanned by CamScanner

classmate
Date
Page

→ Now whenever we do the join of R_1 & R_2 & we get the exact table then it is loss-less join decomposition.

Natural join ($R \bowtie S$) R & S are two relations

Step 1) Cross product (Cartesian product) of R & S

R	S
n_1 tuple	n_2 tuple
c_1 attributes	c_2 attributed

$$\boxed{\begin{array}{l} R \times S = n_1 \times n_2 \text{ tuple} \\ \qquad\qquad c_1 + c_2 \text{ attribute} \end{array}}$$

Step 2) Select the tuple which satisfy equality condition on all common attributes (from $R \times S$)

Step 3) projection of distinct attribute

(qs1)

$R_1(AB)$

A	B
1	2
2	2
3	1

$R_2(BC)$

B	C
2	1
2	2
1	2

R_1	R_2
3 tuple	3 tuple
2 attri	2 attribute

$R_1 \times R_2$ = 9 tuple, 4 attribute

Scanned by CamScanner

Step 1) $R_1 \times R_2$

$R_1.A$	$R_1.B$	$R_2.B$	$R_2.C$
1	2	2	1
1	2	2	2
1	2	1	2
2	2	2	1
2	2	2	2
2	2	1	2
3	1	2	1
3	1	2	2
3	1	1	2

Step 2) Select the tuple which satisfy equality condition on all common attributes from $R \times S$
(5 tuples fetched)

$R_1.B = R_2.B$

Step 3) projection of distinct attribute

$R_1 \bowtie R_2$

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	1	2

→ lossy join ∵ we are not getting the exact table

(here $R_1 \bowtie R_2 \geq R$ so lossy join)

Case 2)

a) $R_1(AB)$

A	B
1	2
2	2
3	1

$R_2(AC)$

A	C
1	1
2	2
3	2

Scanned by CamScanner

$R_1 \times R_2$

$R_1 \cdot A$	$R_1 \cdot B$	$R_2 \cdot A$	$R_2 \cdot C$
✓ 1	2	1	1
1	2	2	2
1	2	3	2
2	2	1	1
✓ 2	2	2	2
2	2	3	2
3	1	1	1
3	1	2	2
✓ 3	1	3	2

$R_1 \bowtie R_2$

↓

getting same relation R

$R_1 \bowtie R_2 = R$

A	B	C
1	2	1
2	2	2
3	1	2

loss-less join decomposition

<u>Shortcut for lossless join decomposition</u> : Let R be the relational schema with FD set F is decomposed into sub relations R_1 & R_2

$R_1 \bowtie R_2$ is ~~lossless~~ lossless only if

all attributes you are getting

Condition ① $R_1 \cup R_2 = R$ eg. $R_1(AB) \cup R_2(BC) = R(ABC)$

condition ② If common attributes of R_1 & R_2 either a super key of R_1 or super key of R_2

$$[R_1 \cap R_2]^+ \rightarrow R_1$$

or

$$[R_1 \cap R_2]^+ \rightarrow R_2$$

(both condition should satisfy)

Scanned by CamScanner

lossy join decomposition : Let R be the relational schema decomposed into R_1 & R_2. $R_1 \bowtie R_2$ is lossy iff

① If common attribute neither a superkey of R_1 nor super key of R_2

neither

$[R_1 \cap R_2]^+ \not\rightarrow R_1$

nor

$[R_1 \cap R_2]^+ \not\rightarrow R_2$

② R(ABCD)

$R_1(AB)$ $R_2(CD)$] lossy Join (∵ no common attribute)

③ R(ABCDE)

$R_1(ABC)$ $R_2(BCD)$] lossy join bcoz attribute E is missing

Question

① R(ABC) $[A \rightarrow B]$

$R_1(AB)$ $R_2(AC)$

⇒ $R_1 \cup R_2 = R(ABC)$ ✓ → getting all attributes

$R_1(AB) \cap R_2(AC) \Rightarrow [A]^+ = [AB]$

means A is super key of R_1

So loss-less join

② R(ABC) $A \rightarrow B$

$R_1(AB)$, $R_2(BC)$

$R_1 \cup R_2 = R(ABC)$ ✓

$R_1(AB) \cap R_2(BC) \Rightarrow [B]^+ = [B]$ (neither super key of R_1 nor R_2)

lossy join

Scanned by CamScanner

Question $R(ABCD)$ $AB \to CD$, $D \to A$

$R_1(BCD)$ $R_2(AD)$

$R_1 \cup R_2 = R(ABCD)$

$R_1(BCD) \cap R_2(AD) = [D]^+ = [DA]$

↓ Super key of R_2

loss-less join

Question

$R(ABCDEF)$ $AB \to CD$, $D \to E$, $E \to F$

$D: [ABCD, DEF]$

decompose ⇓

$R_1(ABCD)$, $R_2(DEF)$

$R_1 \cup R_2 = (ABCDEF)$

$R_1 \cap R_2 = [D]^+ = [DEF]$

↓ S.K of R_2

loss-less join

Question $R(ABCDEF)$ $AB \to C$, $C \to D$, $D \to EF$

$R_1(ABCE)$, $R_2(DEF)$

$R_1 \cup R_2 = (ABCDEF)$

$R_1 \cap R_2 = [E]^+ = [E]$

lossy join

Question $R(ABCDEG)$ $AB \to C$, $AC \to B$, $AD \to G$, $B \to D$, $BC \to A$, $E \to G$

$R_1(AB)$, $R_2(BC)$, $R_3(ABDE)$, $R_4(EG)$

(a) go one by one

$R_1 \cup R_2 = ABC$

$R_1 \cap R_2 = [B]^+ = [BD]$

can not join

Scanned by CamScanner

$R_1 \cap R_3 = [AB]^+ = [ABC..]$ → getting AB

Super key of R_1

$R_{13}(ABDE) \cap R_4(EG) = [E]^+ = [EG]$

Super key of R_4

$R_{134}(ABDEG) \cap R_2(BC) = [B]^+ = [BD]$

⇓

now finally we can say lossy join

$R_1(AB)$ $R_2(BC)$ $R_3(ABDE)$ $R_4(EG)$

$R_{13}(ABDE)$ $R_2(BC)$ $R_4(EG)$

$R_{134}(ABDEG)$ $R_2(BC)$

lossy join

Question $R(ABCDEG)$ $AB \to C$, $AC \to B$, $AD \to E$, $B \to D$, $BC \to A$
$E \to CG$

$R_1(AB)$, $R_2(BC)$, $R_3(ABDE)$, $R_4(ECG)$

$R_1 \cap R_2 = [B]^+ = [BD]$

Can not join

$R_1 \cap R_3 = [AB]^+ = [ABCDE$

Super key of R_1

(Can join)

$R_{13} \cap R_4 = [E]^+ = [ECG]$

Super key of R_4

Scanned by CamScanner

$R_{134}(ABDEGH) \cap R_2(BC)$
$= [BC]^+ = [BCDAEGH]$

Super key of R_{134}
so loss-less join
$R_{1342} \equiv R$

Question $R(ABCDEGH)$
$AB \to C$, $AC \to B$, $AD \to E$, $B \to D$, $BC \to A$, $E \to G$
$R_1(ABC)$ $R_2(ACDE)$ $R_3(ADG)$
$R_1 \cap R_2 \Rightarrow [AC]^+ = [ACB$ · Super Key of R_1 (join)
$R_{12}(ABCDE)$, $R_3(ADG)$
$R_{12} \cap R_3 \Rightarrow [AD]^+ = [ADEG \ldots]$ join

$R_{123} = R(ABCDEG)$
loss-less join

Grade $R(ABCD)$ $A \to B$, $B \to C$, $C \to D$, $D \to B$
$R_1(AB)$ $R_2(BC)$ $R_3(BD)$

$R_1 \cap R_2 = [AB]^+ = [BCD]$
Super key of R_2
$R_{12} \cap R_3 = [B]^+ = [BCD]$
Super key of R_3
R_{123} ✓ loss less join

Scanned by CamScanner

Gate 2001

Consider a schema R(A, B, C, D) & FD: $A \to B$, $C \to D$

Then decomposition of R in R_1(AB) & R_2(CD) is

a) dependency preserving & lossless join

b) lossless join but not dependency preserving

c) dependency preserving but not loss less join

d) not " " & " " "

AB	CD
R_1	R_2
$A \to B$	$C \to D$

$R_1 \cap R_2 = [\phi]$ no common attribute

~~lossy~~ lossy join

R_2 C → A R_1 no common C not preserved

Gate 2008

Let R(A, B, C, D) be Relational Schema with following FD

$A \to B$, $B \to C$, $C \to D$, $D \to B$

The decomposition of R into (A, B), (B, C), (B, D)

a) gives lossless join, and is dependency preserving

b) " " " , but not " "

c) does not give loss less, but is DP

d) " " " " , & is not DP

R_1(AB), R_2(BC), R_3(BD)

$R_1 \cap R_2$ $[B]^+ = [BC, \ldots]$ join

R_{12} (ABC), R_3(BD)

$R_{12} \cap R_3$ $[B]^+ = [BCD \ldots]$ join

$R_{123} = R(ABCD)$

lossless join

AB	BC	BD
R_1	R_2	R_3
$A \to B$	$B \to C$	$D \to B$

✓ $C \to B$

✓ $B \to D$

$C \to D$ → preserved

R_2 R_3 con. B

$[B]^+ = [CDB$

$[B]^+ = [BCD$

Question

R(ABCDEG) $AB \to C$, $AC \to B$, $AD \to E$, $B \to D$;

$BC \to A$, $E \to G$

{AB, BC, ABDE, EG}.

Scanned by CamScanner

$R_1(AB)$, $R_2(BC)$, $R_3(ABDE)$, $R_4(EG)$

$R_1 \cap R_2$ $[B]^+ = CBD$ ✗

$R_1 \cap R_3$ $[AB]^+ = CAB..J$ ✓

$R_{13}(ABDE)$, $R_2(BC)$, $R_4(EG)$

$R_{13} \cap R_2$ $[B]^+ = CBD$ ✗

$R_{13} \cap R_4$ $[E]^+ = [EG]$ ✓

$R_{134}(ABDEG)$, $R_2(BC)$

$R_{134} \cap R_2$ $[B]^+ = [BD]$ ✗

lossy join

Gate 2015

Consider the relation $X(P, Q, R, S, T, U)$ with following FD

$F = \{$

$\{P, R\} \rightarrow \{S, T\}$

$\{P, S, U\} \rightarrow \{Q, R\}$

$\}$

Which of the following is the derivial FD in F^+ is closure of F?

a) $\{P, R\} \rightarrow \{S, T\}$ c) $\{P, S\} \rightarrow \{S\}$

b) $\{P, R\} \rightarrow \{R, T\}$ d) $\{P, S, U\} \rightarrow \{Q\}$

② Dependency preserving Decomposition

Let R be the relational Schema with FD set F is decomposed into sub Relation $R_1, R_2, R_3 \ldots R_n$ with FD set $F_1, F_2, F_3 \ldots F_n$ respectively.

If $F_1 \cup F_2 \cup F_3 \ldots \cup F_n = F$
then dependency preserving

If $F_1 \cup F_2 \cup F_3 \ldots \cup F_n \subset F$
dependency not preserving

Scanned by CamScanner

Question

R(ABCDEG) AB → C , AC → B , AD → E , B → D , BC → A , E → G

decomposed D: [ABC , ACDE , ADG]

R_1 (ABC)	R_2 (ACDE)	R_3 (ADG)
AB → C	AD → E	
AC → B		
BC → A		

AB → C ✓
AC → B ✓
AD → E ✓
B → D
BC → A ✓
E → G

first directly which is preserved write down

now to check for B → D & E → G (via or indirectly)

① B → D (to check indirectly)

present in R_1 ; present in R_2, R_3

Common attribute in R_1, R_2 are AC

B → D or B → D

if B → A & A → D ; B → C & C → D

[B]+ = [BD] ; [B]+ = [BD]

not getting B → A ; not getting B → C

so B → D not preserved

② for E → G

present in R_2 ; present in R_3

Common attr. R_2, R_3 A, D

E → G or E → G

not getting E → A

E → A & A → G ; E → D & D → G

[E]+ = [EG] ✗ ; [E]+ = [EG] ✗ (not getting E → D)

Scanned by CamScanner

So $E \to G$ not preserved

Ans: dependency not preserved
$B \to D$, $E \to G$

Question

R(ABCDEFGHIJ)

$AB \to C$, $A \to DE$, $B \to F$, $F \to GH$, $D \to IJ$

① D: [ABCDE, BFGH, DIJ]

R_1 (ABCDE)	R_2 (BFGH)	R_3 (DIJ)
$AB \to C$	$B \to F$	$D \to IJ$
$A \to DE$	$F \to GH$	

dependency preserved

② D: [ABC, ADE, BF, FGH, DIJ]

R_1 (ABC)	R_2 (ADE)	R_3 (BF)	R_4 (FGH)	R_5 (DIJ)
$AB \to C$	$A \to DE$	$B \to F$	$F \to GH$	$D \to IJ$

dependency preserved

Scanned by CamScanner

(8)

R_1 (ABCD)	R_2 (DE)	R_3 (BF)	R_4 (FGH)	R_5 (DIJ)
$AB \to C$		$B \to F$	$F \to GH$	$D \to IJ$

To check for $A \to DE$

present in R_1 ; present in R_2

Common attribute of R_1 R_2 : D

$A \to DE$

$A \to D$ $\quad$ $D \to DE$

$[A]^+ = [ADE \ldots$

$A \to D$ ✓

$[D]^+ = [DI]$

$D \to DE$ ✗

$A \to DE$ not preserved

**

Question

$R(ABCD)$ $\{A \to B, B \to C, C \to D, D \to A\}$

D : [AB, BC, CD]

R_1 (AB)	R_2 (BC)	R_3 (CD)
$A \to B$	$B \to C$	$C \to D$

$D \to A$

R_3 $\quad$ R_1

Common : no common attribute

connecting to R_2 (BC)

so go via C

Scanned by CamScanner

$D \to A$

$D \to C$ & $C \to A$

$[D]^+ = [DABC]$

$D \to C$ ✓

$[C]^+ = [CDAB]$

$C \to A$ ✓

So $D \to A$ is indirectly preserved

$D \to C$ & $C \to A$

in R_3 in R_1

---

* Normal Forms

→ **Normalization** : is a process to reduce / eliminate the redundancy

o) Redundancy is the unnecessary repeatition of data.

o) <u>Normal form</u> is set of rules which are used to eliminate the redundancy.

o) There are various normal forms

- → 1NF (first normal form)
- → 2NF (Second normal form)
- → 3NF (Third normal form)
- → BCNF (Boyce-Codd Normal form)

<u>Note</u>: o) Every higher normal form satisfy the lower normal form condition.

o) If relation R is in 2NF i.e R is already in 1NF

o) If relation R is in BCNF that means R is already in

3NF, 2NF & 1NF.

① 1NF [first normal form]

o) A relation R is in 1NF if R does not contain any multivalued attribute

(or)

o) A relation R is in 1NF if all attribute of R are atomic.

eg)

Sid	Sname	Subject
S1	A.	C / Java

→ multivalued attribute (more than one value)

R not in 1NF

↓

Sid	Sname	Subject
S1	A	C
S1	A	Java

now R in 1NF

eg) R(ABC) [A → B]

C is not present in FD. So C is here multivalued attribute.

→ To remove multivalued attribute make a part of candidate key.

$[AC]^+ = [ABC]$

means AC is candidate key.

Note,

o) 1NF ensured by using candidate key

o) Default RDBMS is in 1NF, RDBMS does not accept any multivalued attribute.

Scanned by CamScanner

- BCNF have 0% redundancy

- **Redundancy level**
  1NF > 2NF > 3NF > BCNF

- In 1NF redundancy level is too high.
- We can say that in 1NF we are not eliminating redundancy only converting multivalued attribute to single valued attribute.

→ Possible non trivial FD which create redundancy

Case 1) proper subset of candidate key → non key attribute (partial dependency) — eliminated By 2NF

Case 2) non key attribute → non key attribute (transitive dependency) — eliminated by 3NF

Case 3) proper subset of one candidate key → proper subset of another candidate key — eliminated by BCNF

Note
- In complete database sys there are three types of FD's which create redundancy (listed above)

Note:
- 2NF does not allow case 1, but allow case 2 & case 3
- 3NF does not allow case 1 & case 2, but allow case 3
- BCNF does not allow case 1, case 2 & case 3
  ↳ all three cases are eliminated which create redundancy

Scanned by CamScanner

o) BCNF has 0% redundancy.

Case 1)

proper subset of candidate key → non key attribute

eg. R(ABCDE) {AB → C, C → D, B → E}

[A]+ = [A]

[B]+ = [BE]

[AB]+ = [ABCDE]

AB is candidate key

non key attribute = [C, D, E]

non prime attribute

B → E

proper subset of C.K → non key attribute } not in 2NF

will create redundancy

Case 2)

non key attribute → non key attribute

eg) R(ABC) {A → B, B → C}

Candidate key is [A]

non key attri [B, C]

B → C

non key attribute → non key attribute } not in 3NF

Case 3)

proper subset of one C.K → proper subset of another CK

Scanned by CamScanner

o) Till 3NF { lossless join & dependency preserved both guranteed }

o) But in BCNF lossless join is guranteed but dependency may or may not be preserved

→ that's why we can't say BCNF is best NF, 3NF is adequate NF

(Q) R(ABCD) {AB → CD, D → A}

C.K = [AB, DB] ~~(B should be~~

D → A } Not in BCNF

proper subset of DB CK

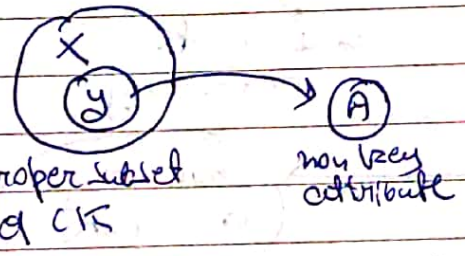
proper subset of AB CK

(2) 2NF [Second Normal Form].

o) Relation R is in 2NF iff

- → R is in 1NF
- → R should not contain any partial dependency

let suppose X is any candidate key



X: any candidate key

y: proper subset of CK (X)

A: non prime/key attribute

y → A is called partial dependency

Question

R(ABCDEF) {AB → C, C → D, B → E, B → F}

Check 2NF

o) first find C.K

AB should be present in CK

[AB]+ = [ABCDEF]

~~As~~ AB is C.K

C.K: [AB]

non prime attribute [C, D, E, F]

Scanned by CamScanner

$B \to E$, $B \to F$ (partial dependency)
not in 2NF

<u>2NF decomposition</u> (converting into 2NF).

$[B]^+$ (attribute which created prob of partial dependency, take the closure)

$[B]^+ = [BEF]$

one table of A, B, C, D

CK : AB

A	B	C	D	R_1

$AB \to C$
$C \to D$

Candidate key to cover attribute remaining make another table (i.e A, C, D) already there in CK

CK : B → one table of B, E, F

B	E	F	R_2

$B \to E$, $B \to F$

a) So R(ABCDEF)
↓ decomposed into

R_1	R_2
(ABCD)	(BEF)
CK : AB	CK : B

now in 2NF + lossless join + dependency preserved

(i) $R_1(ABCD) \cap R_2(BEF) \Rightarrow B$
$[B]^+ = [BEF]$ super key of R_2
So lossless join

(ii)

$R_1(ABCD)$	$R_2(BEF)$
$AB \to C$	$B \to E$
$C \to D$	$B \to F$

dependency preserved

Scanned by CamScanner

Question

R(ABCDEFG)

{ AB → C , C → D, B → E, E → F, A → G }

$[AB]^+ = [ABCDEFG]$

CK : AB

non key attri : C, D, E, F, G

B → E A → G (partial dependency)

not in 2NF

2NF decomp

$[B]^+ = [BEF]$ $[A]^+ = [AG]$

R_1	R_2	R_3
AB C D	B E F	A G
CK : AB	C.K : B	C.K : A
$[AB]^+ = [ABCD]$	$[B]^+ = [BEF]$	$[A]^+ = [AG]$
AB → C	B → E	A → G
C → D	E → F	

(now in 2NF + lossless join + dependency preserved)

R(ABCDEFG)

{AB → C , C → D , B → E , E → F ,
A → G }

(not in 2NF)

→ R_1 (ABCD) AB → C, C → D

→ R_2 (BEF) B → E, E → F

→ R_3 (AG) A → G

(in 2NF)

Scanned by CamScanner

Question

R(ABCD) { AB → C , B → D }

[AB]⁺ = [ABCD]

CK : AB

NKA : C, D

B → D partial dependency

not in 2NF

2NF decomp

[B]⁺ = [BD]

R1	R2
A B C	B D
AB → C	B → D

2NF

+ lossless join

+ dependency preserved

(in 2NF)

Question

Gate 1997

for a db relation R(a, b, c, d) where the domains of a, b, c, d include only atomic values, only the following FD & those that can be inferred from them hold:

a → c

b → d

This relation is

a) in 1NF but not in 2NF

b) in 2NF but not in 3NF

c) in 3NF

d) none.

[ab]⁺ = [abcd]

CK : ab

NPA : c, d

a → c (P.D)

b → d

Not in 2NF

Scanned by CamScanner

Gate 2005

A table has fields F1, F2, F3, F4, F5 with the following FD F1 → F3, F2 → F4, F1F2 → F5

In terms of normalization, this table is in

a) 1NF c) 3NF

b) 2NF d) none

[F1F2]+ = {F1 F2 F3 F4 F5}

CK: F1F2

NPA: F3, F4, F5

F1 → F3
F2 → F4] P.D

not in 2NF

Gate 2008

Let R(ABCDEPG) be relation schema in which the following FD are known to hold:

AB → CD, AE → P, C → E, P → C, B → G

The R is

a) in BCNF

b) in 3NF but not in BCNF

c) in 2NF but not in 3NF

d) not in 2NF

[AB]+ = {ABCDEPG}

CK: AB

NPA: C, D, E, P, G

B → G (P.D)

not in 2NF

Question

→

Scanned by CamScanner

o) R(ABCD) $\{AB \to C, BC \to D\}$

$[AB]^+ = [ABCD]$

CK AB

NPA : C, D no. P.D

already in 2NF

Question

R(ABCDE) $\{A \to B, B \to E, C \to D\}$

CK : [AC]

NPA : B, D, E

$A \to B$, $C \to D$ (P.D)

not in 2NF

2NF decompo

$[A]^+ = [ABE]$ $[C]^+ = [CD]$

AC	R_1: A B E	R_2: C D
not required (∵ all attributes are covered)	$A \to B$, $B \to E$	$C \to D$

in 2NF, but lossy join

R_3	R_2	R_1
A C	A B E	C D

now loss less join

2NF + lossless join + dependency preserving

<u>copy using R_3</u>

o) whenever we try to merge then its useful

to avoid superfluous tuple (extra tuple)(lossy join) we used primary key bcoz if $R_2(ABE) \cap R_1(CD) \Rightarrow$ lossy join

Scanned by CamScanner

Question

R(ABCDEFGHIJ) $AB \to C$, $BD \to EF$, $AD \to GH$, $A \to I$, $H \to J$

Candidate key: [ABD]

NPA: [C, E, F, G, H, I, J]

$AB \to C$, $BD \to EF$, $AD \to GH$, $A \to I$ {P.d}

not in 2NF

2NF decomp

$[AB]^+ = [ABCI]$

$[BD]^+ = [BDEF]$

$[AD]^+ = [ADGHIJ]$

$[A]^+ = [AI]$

CK AB	BD	AD	A
ABCI	BDEF	ADGHIJ	AI
R_1	R_2	R_3	R_4

$R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \Rightarrow$ lossy join

$\Downarrow$ (prescribe original C.K)

R_1	R_2	R_3	R_4	R_5
ABD	ABCI	BDEF	ADGHIJ	AI

2NF + lossless join + DP

Now

ABCI $\Rightarrow$ AI, ABC

$AB \to C$ CK: AB

$A \to I \to$ P.D

$A \to I$ $AB \to C$

ADGHIJ CK AD $\Rightarrow$ ADGHJ, AI

$AD \to GH$

$A \to I \to$ P.D

$H \to J$

Scanned by CamScanner

final answer

R7 [BDEF]

R1	R2	R3	R4	R5	R6
ABD	ABC	AI	ADGHJ	AI	AI

⇓ Convert Similar into only one R

final answer

R1	R2	R3	R4	R5
ABD	ABC	ADGHJ	AI	BDEF
	AB → C	AD → GH H → J	A → I	BD → EF

2NF + lossless join + DP

(3) 3NF (Third Normal Form)

→ Relation R is in 3NF iff
- → R is in 2NF
- → R does not contain any transitive dependency

(or)

Relation R is in 3NF if it does not allow all 2 cases.

case 1) proper subset of C.K → non key attribute — Partial dependency

case 2) non key attribute → non key attribute — transitive dependency

(Bhramastra) (or)

o) Let R be the Relational Schema is in 3NF if every ~~non trivial~~ X → Y non trivial FD must satisfy the following conditions:

Scanned by CamScanner

$X \rightarrow Y$

either X : Super key or Y : prime attribute or key attribute

⇓

voilating above condition so in 3NF

Question

R(ABCDE) { $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$ }

C.K : [AB]

NPA : C, D, E

no partial dependency so in 2NF

$C \rightarrow D$, $D \rightarrow E$ (transitive dependency)

nonkey attribute → non key att

not in 3NF

(or directly check for 3NF)

$AB \rightarrow C$ $C \rightarrow D$ $D \rightarrow E$ $(X \rightarrow Y)$

super key

not satisfying so not in 3NF

either super key or → prime attribute

3NF decomp (make separate table for non key → non key)

R_1	R_2	R_3
A B C	C D	D E
$AB \rightarrow C$	$C \rightarrow D$	$D \rightarrow E$

3NF + lossless join + DP

Scanned by CamScanner

Question

R(ABC) { A → B, B → C }

CK : [A]

NPA : B, C

A → B B → C (transitive dependency)

super key ↳ not in 3NF

3NF decom

R1 (AB): A → B

R2 (BC): B → C

3NF + lossless join + DP

R(ABC), A → B, B → C

A	B	C
1	b	x1
2	b	x1
3	b	x1
4	p	y1
5	p	y1
6	p	y1

here single table

3NF

R1 (AB) A → B

A	B
1	b
2	b
3	b
4	p
5	p
6	p

R2 (BC) B → C

B	C
b	x1
b	x1
b	x1
p	y1
p	y1
p	y1

=>

B	C
b	x1
p	y1

load is increased ∵ now to maintain 2 table (but here 2 table)

(but redundancy is reduced) => we can see unnecessary entries are reduced

Scanned by CamScanner

Question

R(ABCDE) $AB \to C$, $B \to D$, $D \to E$

$[AB]^+ = [ABCDE]$

CK : AB

NPA : C, D, E

$B \to D$ (partial dependency)

not in 2NF

2NF decomp

$[B]^+ = [BDE]$

R_1: $\boxed{AB \quad C}$ R_2: $\boxed{B \quad D \quad E}$

$AB \to C$ $B \to D$, $D \to E$

now in 2NF + lossless join + DP

R_1 (ABC) $\{AB \to C\}$ in 3NF
↳ super key

R_2 (BDE) $\{B \to D, D \to E\}$

C.K : $[B]^+ = [BDE]$

NPA : D, E

$D \to E$ (transitive dep)

not in 3NF

R_2 (BDE) → R_3 (BD) $\boxed{B \quad D}$ $B \to D$

R_2 (BDE) → R_4 (DE) $\boxed{D \quad E}$ $D \to E$

now in 3NF

→

final ans

$R_1(ABC)$ $R_3(BD)$ $R_4(DE)$

AB	C

$AB \to C$

B	D

$B \to D$

D	E

$D \to E$

in 3NF + lossless join + DP

check

$R_1(ABC)$ $R_3(BD)$ $R_4(DE)$

$R_{13}(ABCD)$

$R_{134}(ABCDE) = R$

Question

$R(ABCD)$ $\{AB \to CD, D \to A\}$

$[AB]^+ = [ABCD]$

C.K ; AB , DB

PA [A, B, D]

$D \to A$

$[DB]^+ = [DBAC]$

$AB \to CD$ $D \to A$

super key → prime attribute

already in 3NF

Question

R(ABC) $\{AB \to C,\ C \to A\}$

$[AB]^+ = [ABC]$

C.K : AB, CB

PA : A, B, C

$C \to A$

$[CB]^+ = [CBA]$

$AB \to C$, $C \to A$

Superkey ↳ prime attribute

already in 3NF

(4) BCNF [Boyce-Codd Normal Form]

→ A relational schema R is in BCNF ~~iff~~ if it does not allow all 3 cases.

Case 1) proper subset of CK → non key attribute

Case 2) non key attri → non key attri

case 3) proper subset of one CK → proper subset of another CK

Bhramastra

(or)

•) A relational schema R is in BCNF if every non trivial FD must satisfy the following condition.

$X \to Y$

X: Superkey

Scanned by CamScanner

Question

R(ABCD) $\{AB \to CD, D \to A\}$

1NF ✓,

~~2NF~~ C.K [~~AB~~] AB, DB

~~NPA~~ & PA : A, B, D

$AB \to CD$, $D \to A$

superkey ↳ prime attribute

in 3NF

Check for BCNF:

$D \to A$

proper subset of DB CK ↳ proper subset of AB CK

not in BCNF

(or)

$AB \to CD$, $D \to A$ (not in BCNF)

superkey not a super key

BCNF decomposition

first make separate table for $D \to A$

| B C |

(B, C remaining)

| D A |

Connecting attribute can be either D or A

case 1) let choose A

R_1

| B C A |

R_2

| D A |

$R_1(ABCA) \cap R_2(DA)$

(lossy join) $[A]^+ = [A]$

can't join

Scanned by CamScanner

o) 3NF both lossless join & DP are guaranteed
o) In BCNF lossless join is guaranteed but dependency may or may not be preserved

Case 2)

R1	R2
BCD	DA

$R_1(BCD) \cap R_2(DA)$
$= [D]^+ = [DA]$
Superkey of R_2 (lossless)

Ans:

R1	R2
BCD	DA
C.K = BD	C.K = D

here C.K key

$D \rightarrow A$

** here only one CK is preserved i.e BD & AB CK is lost

BCNF + lossless join, dependency not preserved
$AB \rightarrow CD$ not preserved

$[BC]^+ = [BC]$
$[CD]^+ = [CD]$
$[BD]^+ = [BDAC]$
CK: BD

Note:
o) Some relation may not be possible to decompose into ~~BCN~~ dependency preserving BCNF decomposition.
o) Why So?
↳ one prime attribute removed from one relation & kept in new relation bcz of this movement of prime attribute from original relation to some new relation leads to loss of original Candidate Key & hence due to that, occ of the dependency is not preserved.

Summary

design goal	1 NF	2 NF	3 NF	BCNF
0% redundancy	✗	✗	✗	✓
lossless Join	✓	✓	✓	✓
dependency preserval	✓	✓	✓	may be / may not be

o) 3NF is best normal form.

Question

R(ABCDEFGHIJ) {AB → C, A → DE, B → F, F → GH, D → IJ}

CK $[AB]^+ = [ABCDEFGHIJ]$

CK : AB,

PA : A, B

NPA : C, D, E, F, G, H, I, J

A → DE , B → F, (partial dependency)

Not in 2NF

<u>2NF decomposition</u>

$[A]^+ = [ADEIJ]$

$[B]^+ = [BFGH]$

R_1	R_2	R_3
ABC	ADEIJ	BFGH
AB → C	A → DE	B → F
CK AB	D → IJ	F → GH
	CK A	CK B

In 2NF + lossless + DP.

Scanned by CamScanner

$R_1(ABC)$	$R_2(ADEIJ)$	$R_2(BFGH)$
$AB \to C$	$A \to DE$	$B \to F$
CK: AB	$D \to IJ$	$F \to GH$
	CK: A	CK: B
$AB \to C$	$A \to DE$ $D \to IJ$	$B \to F$ $F \to GH$
superkey in 3NF	superkey (transitive) not in 3NF	SK not in 3NF

$R_2(ADEIJ)$

$A \to DE$, $D \to IJ$

A DE	D IJ
$A \to DE$	$D \to IJ$

$R_3(BFGH)$

B F	F GH
$B \to F$	$F \to GH$

not in 3NF

A B C	A DE	D IJ	B F	F GH
$AB \to C$	$A \to DE$	$D \to IJ$	$B \to F$	$F \to GH$
in BCNF	in BCNF	3NF + lossless + DP in BCNF	in BCNF	in BCNF

Final ans

$R(ABCDEFGHIJ)$ →
- $R_1(ABC)$ $AB \to C$
- $R_2(ADE)$ $A \to DE$
- $R_3(DIJ)$ $D \to IJ$
- $R_4(BF)$ $B \to F$
- $R_5(FGH)$ $F \to GH$

in BCNF + lossless + DP

Scanned by CamScanner

Note:

① 2NF

a) If every candidate key of Relation R is a simple candidate key (no compound candidate key in R) then the relation R is in 2NF but may or may not be in 3NF.

eg.

R(ABCDE) {A→B, B→C, C→D, D→E, C→A}

Candidate key: [A, C, B]

$[A]^+ = [ABCDE]$

all C.K are simple CK so R is in 2NF but not in 3NF

② 3NF

If every attribute of relation R is a prime/key attribute then the relation R is in 3NF but may or may not in BCNF

eg) R(ABCD) {A→B, B→C, C→A}

C.K: AD, CD, BD

Prime attribute: A, B, C, D

R is in 3NF but not in BCNF

③ BCNF

R is in 3NF & all the ~~attribut~~ candidate keys are simple candidate key then R is in BCNF.

eg) R(ABC) {A→B, B→C, C→A}

C.K: A, B, C

in every X → Y

↳ super key

So R is in BCNF

→

Scanned by CamScanner

(4) ∘) Binary Relation (Relation with 2 attribute) always in BCNF.

eg. R(AB) either $A \to B$ or $B \to A$

So in every $X \to Y$
↳ super key

(5) ∘) Relation R with no non trivial FD (trivial FD) always in BCNF

$xyz \to xy$ (trivial FD)

$X \to Y$

$x \supseteq y$
↑
super key

Gate 1995

Consider relation schema R(A, B, C) with the following FD

$AB \to C$
$C \to A$

Show that the schema R is in 3NF but not in BCNF

∘) R(ABC) : $\{AB \to C, C \to A\}$

CK $[AB]^+ = [ABC]$

CK : AB, CB

PA : [A, B, C]

$AB \to C$ — super key

$C \to A$ — ↳ prime attr.

in 3NF

$AB \to C$ — superkey

$C \to A$ — not super key

not in BCNF

Scanned by CamScanner

Gate 1998

Which normal form is considered adequate for normal relational database design?

a) 2NF
b) ✓ 3NF
c) 5NF
d) 4NF

Gate 2002 **

Relation R is decomposed using a set of FD, F, and relation S is decomposed using another set of FD, G. One decomposition is definitely BCNF, the other is definitely 3NF, but it is not known which is which. To make a guaranteed identification, which one of the following tests should be used on decomposition.

a) dependency-preservation
b) lossless-join
c) ✓ BCNF definition
d) 3NF definition

Can't do
bcoz BCNF can be DP

Gate 2004

Performance (name, courseNo, rollNo, grade)

FD:

name, courseNo → grade
RollNo, courseNo → grade
name → rollNO
rollNo → name

The highest normal form is?

a) 2NF
b) ✓ 3NF
c) BCNF
d) 4 NF

Scanned by CamScanner

R(ABCD) $AB \to D$, $CB \to D$, $A \to C$, $C \to A$

C.K : $[AB]^+ = ABCD$

C.K AB, CB

PA : A, B, C

NPA : D in 2NF, in 3NF, not in BCNF

Gate 2005

Which of the following statements about normal forms is false?

a) BCNF is stricter than 3NF

b) loss less, dependency preserving decomposition into 3NF is always possible

c) loss less DP decomp into BCNF is always possible

d) Any relation with 2 attribute in BCNF.

Gate 2012

Which is True?

a) every relation in 3NF also in BCNF

b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R

c) every relation in BCNF is also in 3NF

d) no relation can be in both BCNF & 3NF

Gate 2014

S_1: every table with two single-valued attribute is in 1NF, 2NF, 3NF & BCNF

↓ true

↳ binary relation

Gate 2020

Consider a relational table R that is in 3NF, but not in BCNF, which one of the following statements is True?

a) R has a nontrivial FD $X \to A$ where X is not a superkey & A is a non prime attribute & X is a proper subset of some key.

b) R has a nontrivial FD $X \to A$ where X is not a superkey & A is a non-prime attribute & X is not a proper subset of any key

c) A cell in R holds a set instead of an atomic value

d) R has a nontrivial FD $X \to A$, where X is not a superkey & A is prime attribute.

---

* ~~Mini~~ Minimal Cover : (delete)

→ Objective of minimal cover is to eliminate the redundant (extra) FD.

→ $\boxed{\text{Redundant FD}}$ is a FD if we delete that FD from FD set F then after deletion does not effect the power of FD set

Eg. FD: $(A \to B, B \to C, A \to C)$, $[A]^+ = [ABC]$

If $A \to C$ deleted

Or: $[A \to B, B \to C]$

$[A]^+ = [ABC]$ (same power of FD set)

So:

$A \to C$ is redundant FD

↳ extra FD which can be avoided in FD set

Scanned by CamScanner

Procedure to find minimal cover

i) F = [.....]

→ Let F be the given FD set, let us assume any $X \to Y$ is a redundant FD, then we delete that FD from FD set F & After deletion a reco FD set G is created.

Iff $F \equiv G$ — F cover G : True; G cover F : True

then $X \to Y$ is redundant FD.

Question : F = [A → B, B → C, A → C] given

let A → B is redundant FD	let B → C is R.FD	let A → C is RFD
G : [B → C, A → C]	G : [A → B, A → C]	G : [A → B, B → C]
F cover G : true $[B]^+ = [BC..$ $[A]^+ = [ABC..]$	F cover G : true	F cover G : true
	G cover F : false	G cover F : true
G cover F : false $[A]^+ = [AC]$ A → B ✗ F ≥ G	B → C is not R.FD	F ≡ G So A → C is R.FD
A → B is not redundant FD		

A → C is Redundant FD

Minimal cover = A → B, B → C

Scanned by CamScanner

0) Extraneous Attribute

→ Is a attribute if remove (delete) that attribute, create no problem.

F: [AB → C, A → C]

_ _ → _ (when two things then we can say extra)

AB → C

So either A is extra or B is extra

let A is extra attribute	let B is extra attribute
G: [B → C, A → C]	G: [A → C]
F cover G: false	F cover G: true
G cover F: true	G cover F: true
	F ≡ G

So B is extra attribute

Question

F: [AB → C, A → B]

let A extra	let B extra
G: [B → C, A → B]	G: [A → C, A → B]
F cover G: false	F cover G: true
G cover F: true	G cover F: true
	F ≡ G

B is extra attribute

Scanned by CamScanner

Question

$F = [xyz \to w,\ y \to z,\ x \to y]$

let x extra	let y extra	let z extra
$G: [yz \to w,\ y \to z,\ x \to y]$	$G = [xz \to w,\ y \to z,\ x \to y]$	$G = [xy \to w,\ y \to z,\ x \to y]$
F cover G: false	F cover G: true	F cover G: true
G cover F:	$[xz]^+ = [xzyw]$	G cover F: true
	G cover F: true	$F \equiv G$
	$F \equiv G$	(z is extra)
	(y is extra)	

So y & z are extra attribute

G: minimal cover: $[x \to w,\ y \to z,\ x \to y]$

(or) (Shortcut)

$xyz \to w,\ y \to z,\ x \to y$

$xyz \to w$
$y \to z$
(z is determined by y so z is extra)

$xy \to w,\ y \to z,\ x \to y$
$x \to y$
(y is determined by x so y is extra)

final minimal cover $[x \to w,\ y \to z,\ x \to y]$

→

Scanned by CamScanner

Check Ans:

$F: [xyz \to w, y \to z, x \to y]$

$G: [x \to w, y \to z, x \to y]$

F cover G:

$[x]^+ = [xyzw$ ✓ $x \to w$ ✓

$[y]^+ = [yz]$ $xy \to z$ ✓

$[x]^+ = [xyzw]$ $x \to y$ ✓

true.

G cover F:

$[xyz]^+ = [xyzw$ $xyz \to w$ ✓

$[y]^+ = [yz$ $y \to z$ ✓

$[x]^+ = [xy$ $x \to y$ ✓

true

so $F \equiv G$

0) ~~Procedure to find minimal cover~~

Gate 2014

S1: $AB \to C, D \to E, E \to C$ is a minimal cover for the set of functional depe. $[AB \to C, D \to E, AB \to E, E \to C]$?

$F: [AB \to C, D \to E, AB \to E, E \to C]$

$G: [AB \to C, D \to E, E \to C]$

F cover G: true

$[AB]^+ = [ABC$ ✓

$[D]^+ = [DE$ ✓

$[E]^+ = [EC$ ✓

G cover F: false

$[AB]^+ = [ABC$ ✓

$[D]^+ = [DE$ ✓

$[AB]^+ = [ABC$ ✗

not minimal cover

S1: false

Scanned by CamScanner

o) first eliminate ~~FD~~ R.FD, then delete extra attribute to find minimal cover

* Procedure to find minimal cover

Step 1) Split the FD such that RHS contain single attribute

eg. $A \to BC$

split (decomposition rule)

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

Step 2) find redundant FD & delete from the FD set

eg. $[A \to B, B \to C, A \to C]$

$A \to C$ is extra to delete

$[A \to B, B \to C]$

Step 3) find the redundant attribute on LHS and delete it. (extra)

eg. $AB \to C$

A can be extra if $[B]^+ = [B \cdots A]$ → contain A

B can be extra if $[A]^+ = [A \cdots B]$ → contain B

Question

$V \to W, VW \to X, Y \to VX, Y \to Z$

Step 1) Split the FD such that RHS contain single attribute

$V \to W, VW \to X, Y \to V, Y \to X, Y \to Z$

Step 2) find redundant FD

$V \to W$	$VW \to X$	$Y \to V$	$Y \to X$ (extra)	$Y \to Z$
$[V]^+ = [VW]$	$[VW]^+ = [VW]$	$[Y]^+ = [YXZ]$	$[Y]^+ = [YVWXZ]$ getting X so extra	$[Y]^+ = [YVWX]$
not extra	not extra	not extra		not extra

Scanned by CamScanner

Explain $v \to w$

if $v \to w$ is extra so we should get w from $[v]^+$ with help of other FD

$[v]^+ = [v]$ not getting w so not extra

$v \to w$, $vw \to x$, $y \to v$, $y \to z$

Step 3) find redundant attribute on LHS & delete them

$v \to w$, $vw \to x$, $y \to v$, $y \to z$

v is extra
if $[w]^+$ contain v
$[w]^+ = [w]$
(v not extra)

w is extra
if $[v]^+$ contain w
$[v]^+ = [vw]$
w is extra

Ans) minimal cover : $v \to w$, $v \to x$, $y \to v$, $y \to z$

or

$v \to wx$, $y \to vz$

Question

$[A \to C, AC \to D, E \to AD, E \to H]$

Step 1) $A \to C$, $AC \to D$, $E \to A$, $E \to D$, $E \to H$

Step 2) $A \to C$ $[A]^+ = [A]$

$AC \to D$ $[AC]^+ = [AC]$

$E \to A$ $[E]^+ = [EDH]$

$E \to D$ $[E]^+ = [EAHCD]$ extra

$E \to H$
$[E]^+ = [EACD]$

($E \to D$ is extra)

Scanned by CamScanner

$A \to C$, $AC \to D$, $E \to A$, $E \to H$

Step3) $A \to C$, $AC \to D$, $E \to A$, $E \to H$

o) A extra $[C]^+ = [C]$
not extra

o) C extra $[A]^+ = [AC]$
C is extra

Minimal cover $A \to C$, $A \to D$, $E \to A$, $E \to H$
or
$[A \to CD, E \to AH]$

Question $[A \to B, C \to B, D \to ABC, AC \to D]$

Step1) $A \to B$, $C \to B$, $D \to A$, $D \to B$, $D \to C$, $AC \to D$

Step2)

$A \to B$	$C \to B$	$D \to A$	$D \to B$	$D \to C$
$[A]^+ = [A]$	$[C]^+ = [C]$	$[D]^+ = [DABC]$	$[D]^+ = [DAB$ extra	$[D]^+ = [DAB]$

$AC \to D$
$[AC]^+ = [ACB]$
($D \to B$ extra)

Step3) $A \to B$, $C \to B$, $D \to A$, $D \to C$, $AC \to D$

o) A extra $[C]^+ = [CB]$
no

o) C extra $[A]^+ = [AB]$
no.

Scanned by CamScanner

(Minimal cover or Irreducible set)

Minimal cover : $A \to B$, $C \to B$, $D \to A$, $D \to C$, $AC \to D$

or

$[A \to B, C \to B, D \to AC, AC \to D]$

~~a)~~ B

Gate 2017

The following FD hold true for the relational schema R{v, w, x, y, z}

$v \to w$, $wv \to x$, $y \to vx$, $v \to z$

which of the following is irreducible equivalent for this set of FD?

a) {$v \to w$, $v \to x$, $y \to v$, $y \to z$}

b) {$v \to w$, $w \to x$, $y \to v$, $y \to z$}

c) {$v \to w$, $v \to x$, $y \to v$, $y \to x$, $y \to z$}

d) {$v \to w$, $w \to x$, $y \to v$, $y \to x$, $y \to z$}

Step 1) $v \to w$, $wv \to x$, $y \to v$, $y \to x$, $v \to z$

Step 2) $v \to w$ $[v]^+ = [vz]$

$wv \to x$ $[wv]^+ = [wvz]$

$y \to v$ $[y]^+ = [yx]$

$y \to x$ $[y]^+ = [yvwxz]$ extra

$v \to z$

$[v]^+ = [vwx]$

Step 3) $v \to w$ $wv \to x$ $y \to v$ $v \to z$

w extra

$[v]^+ = [vw \ldots$

w extra

v extra

$[w]^+ = [w]$

v not extra

Ans. : $v \to w$, $v \to x$, $y \to v$, $v \to z$

Scanned by CamScanner

Note

eg) BCNF has 0% redundancy but suffer from multivalued functional dependency [$x \to\to y$]

o) Multivalued Functional dependency [$x \to\to y$]

→ If relation R consist two or more multivalued Attribute then multivalued functional dependency exist.

eg)

Sid	Sname	Book
A	~~C/Java~~ x/y	korth/galvin

2 multivalued attribute

Sid	Sname	Book
A	x	korth
A	x	galvin
A	y	korth
A	y	galvin

eg)

Sid	name	Course
1	A/B	x/y

2 multivalued attribute

	x	y	z
	Sid	name	Course
t1	1	A	x
t2	1	A	y
t3	1	B	x
t4	1	B	y

eg) R(ABCD) {A → B}

$[A]^+ = [AB]$

So C & D are multivalued attribute (∵ C & D not present in FD)

o) to remove multivalued attribute (add) make a part of candidate key

CK: [ACD]

Note

o) Let R be the relational schema, x, y be the attribute set of R & Z = R − [x ∪ y] // all attributes except x & y

Scanned by CamScanner

$x \rightarrow\rightarrow y$ exist in R only if
$t_1\ t_2\ t_3$ & t_4 tuple such that

t	x	y	z
t_1	x_1	y_1	z_1
t_2	x_1	y_1	z_2
t_3	x_1	y_2	z_1
t_4	x_1	y_2	z_2

(i) $t_1.x = t_2.x = t_3.x = t_4.x$
&&
$t_1.y = t_2.y$ and $t_3.y = t_4.y$
&&
$t_1.z = t_3.z$ and $t_2.z = t_4.z$

Gate 2018

Consider the following four relational schema. For each schema, all non-trivial functional dependencies are listed. The underlined attributes are the respective primary key.

Schema I: Registration (rollno, courses)
Field 'courses' is a set-valued attribute containing the set of courses a student has registered for
rollno → courses

Schema II: Registration (rollno, courseid, email)
rollno courseId → email
email → rollno.
C.K [rollno courseid, email courseid]

Schema III: Registration (rollno, courseid, marks, grade)
rollno, courseid → marks, grade
marks → grade
C.K [rollno courseid]

Schema IV: Registration (rollno, courseid, credit)
rollno, courseid → credit
courseid → credit
C.K [rollno courseid]

Scanned by CamScanner

Which one of the relational schemas is in 3NF but not in BCNF?

a) Schema I c) Schema III

b) Schema II d) Schema IV

Schema I BCNF

Schema II in 3NF not in BCNF

" III in 2NF but not in 3NF

" IV not in 2NF

Gate 2008

Consider the following relational schema for library db:

Book (Title, Author, Catalog-no, Publisher, year, price)

Collection (Title, Author, Catalog-no)

Which the following FD

I: Title Author → Catalog-no

II: Catalog-no → Title Author Publisher year

III: Publisher Title Year → Price

Assume {Author, Title} is the key for both schemas.

Which is true?

a) Both Book & Collection in BCNF

b) " " " are in 3NF only

c) Book → 2NF, Collection → 3NF

d) Both Book & Collection in 2NF only

Book (A, B, C, D, E, F) Collection (A, B, C)

AB → C	AB → C	AB → C
C → ABDE	C → ABDE	superkey
DAE → F	DAE → F	(in BCNF)

C.K AB

NPA C, D, E, F

in 2NF, not in 3NF

Scanned by CamScanner

o) Indexing Concept

Gate 1998

There are five records in a db

Name	Age	Occupation	Category
Rama	27	CON	A
Abdul	22	ENG	A
Jenifer	28	DOC	B
Maya	32	SER	D
Dev	24	MUS	C

There is an index file associated with this & it contains the values 1, 3, 2, 5 & 4. Which one of the field is the index built from?

a) Age ✓ c) occupation

b) Name d) Category

o) In category we can't apply since no unique data can't differentiate

o) Occupation alphabet ↑, no. also increases

C 1

E 3 D lies b/w C & E so no. b/w 1 & 3

D 2

S 5

M 4

Scanned by CamScanner

Transaction

1 transaction

```
Read(A)
A = A - 1000
write(A)
Read(B)
B = B + 1000
write(B)
```

Transaction is a (sequence of) set of logically related operation that perform as ~~until~~ unit of work

⇓

transferring 1000 from account A to account B } one ~~transaction~~ transaction

Read(A): Accessing the data item A from db to programmed variable

Write(A): Updating the data Item (A)

Commit: It specify successful completion of transaction

* To maintain Integrity, Transaction must satisfy ACID property

A	C	I	D
Atomicity	Consistency	Isolation	Durability
transaction Manager	Programmer	Concurrency Subsystem	Recovery Management

- Atomicity taken care by transaction manager & recovery management
- Consistency is taken care by programmer
- Isolation " " " " concurrency subsys
- Durability " " " " recovery management

Scanned by CamScanner

① ∘) **Atomicity** : either execute all operation of the transaction successfully or none of them
(full / none)

∘) **Reason for transaction failure**
- power failure
- S/w crash
- h/w crash
- System crash & etc → many more

→ due to any of the reason if atomicity is getting failed then recovery management component are there.

∘) **Recovery management**: If transaction is failed before commit then recovery management component ~~ROLL BACK~~ ROLL BACK [UNDO all modification] the transaction

∘) **Transaction Log's** : log contain all activities (modification) of the transaction.

∘) Log file maintained by the system until the transaction committed or rollback completely.

② **Consistency**: Before & after the transaction database must be consistent.

before:	after:
A : 4000 } '500'	A : 3500
+ B : 2000	+ B : 2500
6000	6000

consistent

∘) Consistency is taken care by programmer

Scanned by CamScanner

③ **Isolation:** If two or more than two transaction executing concurrently then isolation come into the picture.

i) Concurrent execution of two or more transaction should be equal to any serial ~~sch~~ schedule.

④ **Durability** Any change in the database must persist for long period of time.

(or)

database must be able to recover under any case of failure.

Schedule

i) The order sequence of two or more transaction.

Schedule

Serial Schedule

(execution of one transaction complete then another transaction starts its execution)

T_1	T_2
=	
=	
=	
=	
	=
	=
	=
	=

T_1	T_2
	=
	=
	=
	=
=	
=	
=	
=	

$S_1 \langle T_1\ T_2 \rangle$ — T_1 followed by T_2

$S_2 \langle T_2\ T_1 \rangle$ — T_2 followed by T_1

non-serial Schedule

(Interleaved execution of two or more transaction)

T_1	T_2
=	=
=	=
	=

T_1	T_2
=	
=	
	=
	=

T_1	T_2
=	=
=	=
=	=

& many more

Note

o) If there are n transactions then n! serial schedule are there

eg. $n = 2$

T_1, T_2 (2 Serial Schedule) $\langle T_1 T_2 \rangle, \langle T_2 T_1 \rangle$

eg $n = 3$

$3! = 6$ Serial Schedule

T_1, T_2, T_3

$\langle T_1\ T_2\ T_3 \rangle$ $\langle T_2\ T_1\ T_3 \rangle$ $\langle T_3\ T_1\ T_2 \rangle$

$\langle T_1\ T_2\ T_3 \rangle$ $\langle T_2\ T_3\ T_1 \rangle$ $\langle T_3\ T_2\ T_1 \rangle$

Question

o) $T_1 (T_i)$ $A \xrightarrow{+100} B$

Read (A)
A = A-100
Write (A)
Read (B)
B = B+100
Write (B)

$T_2 (T_j)$ $A \xrightarrow{10\%} B$

Read (A)
temp = A * 0.1
A = A - temp
Write (A)
Read (B)
B = B + temp
Write (B)

taking ~~Sch~~ Serial schedall

o) two transactions T_1, T_2 so 2 Serial Schedule

$A = 2000$, $B = 3000$

(I) $\langle T_1\ T_2 \rangle$

T_1	T_2
Read(A)	// A = 2000
A = A -100	// A = 1900
Write(A)	// A = 1900
~~B = B~~ Read(B)	// B = 3000
B = B +100	// B = 3100
Write (B)	// B = 3100
// A = 1900	Read(A)
// temp = 190	temp = A*0.1
// A = 1710	A = A - temp
// A = 1710	write (A)
// B = 3100	Read (B)
// B = 3290	B = B + temp
// B = 3290	write (B)

$A = 2000$, $B = 3000$

(II) $\langle T_2\ T_1 \rangle$

T_1	T_2
// A = 2000	Read(A)
// temp = 200	temp = A*0.1
A = 1800	A = A - temp
A = 1800	write(A)
B = 3000	Read (B)
B = 3200	B = B + temp
B = 3200	write (B)
Read(A)	// A = 1800
A = A -100	// A = 1700
write(A)	A = 1700
Read(B)	B = 3200
B = B +100	B = 3300
write (B)	B = 3300

Scanned by CamScanner

$A = 2000$
$B = 3000$

taking 2 Concurrent Schedule

III

T_1	T_2
Read(A)	A = 2000
A = A - 100	A = 1900
write(A)	A = 1900
A = 1900	Read(A)
temp = 190	temp = A × 0.1
A = 1710	A = A - temp
A = 1710	write(A)
Read(B)	B = 3000
B = B + 100	B = 3100
write(B)	B = 3100
B = 3100	Read(B)
B = 3290	B = B + temp
B = 3290	write(B)

C1

IV

T_1	T_2
Read(A)	A = 2000
A = A - 100	A = 1900
A = 2000	Read(A)
temp = 200	temp = A × 0.1
A = 1800	A = A - temp
A = 1800	write(A)
write(A)	~~A = 1800~~ A = 1900
Read(B)	B = 3000
B = B + 100	B = 3100
write(B)	B = 3100
B = 3100	Read(B)
B = ~~3200~~ 3300	B = B + temp
B = 3300	write(B)

C2

in case - I (Serial schedule) $\langle T_1 T_2 \rangle$

$$\begin{array}{r} A = 1900 \\ + \quad B = 3290 \\ \hline 5190 \end{array} \qquad \begin{array}{r} A = 1710 \\ + \quad B = 3290 \\ \hline 5000 \end{array}$$

original before transaction $\begin{array}{r} A = 2000 \\ B = 3000 \\ \hline 5000 \end{array}$

(Consistency is maintained)

in case II (Serial schedule) $\langle T_2 T_1 \rangle$

$$\begin{array}{r} A = 1700 \\ B = 3300 \\ + \\ \hline 5000 \end{array}$$

Original before transaction $\begin{array}{r} A = 2000 \\ B = 3000 \\ \hline 5000 \end{array}$

(Consistency is maintained)

Scanned by CamScanner

Case (III) (Concurrent Schedule) C_1

A = 1710
B = 3290
5000

before trans. A = 2000
B = 3000
5000

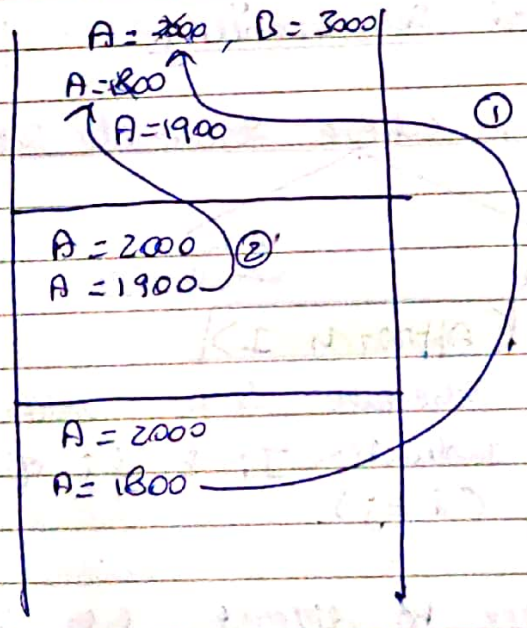
(Consistent)

Case (IV) (Concurrent Schedule) C_2

A = 1800
B = 3300
~~5100~~
9200

before A = 2000
B = 3000
5000

(not Consistent) / (inconsistent)



Note:

1) Serial schedule (all serial schedule (n!)) are always consistent
2) Concurrent schedule may or may not be consistent (But we execute in concurrent manner)

Scanned by CamScanner

o) (In non conflict Instruction, swapping is allowed)

Why Concurrent?

- o) To improve CPU utilization
- o) To Enhance throughput
- o) Faster Response
- o) Effective utilization of Resource
- o) Decrease waiting time & etc.

Objective: we we call execute transaction in concurrent manner but if this concurrent schedule is equal to any of its serial schedule then we can say its a serializable schedule.

Serializable Schedule

- o) If a concurrent schedule has been executed and that could have same effect on database, as a schedule execute without any concurrent (serial) manner called serializable schedule.
- o) Serializable schedule are always consistent
- o) This process is called Serializability.

Ways to achieve Serializable Schedule

- Conflict Serializable
- View Serializable

(1) × Conflict Serializable [Approach I]

→ Let us consider a schedule S in which there are two consecutive Instruction I_i & I_j of transaction T_i & T_j respectively ($i \neq j$).

(1) If I_i & I_j refer to different data Item: Don't conflict (non conflict Instruction) (Swapping possible)

(2) If I_i & I_j refer to same data Item — swapping allowed if

(i) I_i = read(Q) & I_j = read(Q) (non conflict)

(ii) I_i = read(Q) & I_j = write(Q): Conflict instruction

(iii) I_i = write(Q) & I_j = read(Q): conflict instruction

(iv) I_i = write(Q) & I_j = write(Q): conflict instruction

Scanned by CamScanner

o) Conflict Serializable

Concurrent schedule : S

any serial schedule of that concurrent schedule : S'

→ If a schedule S can be transformed into schedule S' by series of swap of non-conflict instructions then we can say schedule S is conflict serializable.

T_1	T_2
Read(A)	
write(A)	
	Read(A)
	write(A)
Read(B)	
write(B)	
	Read(B)
	write(B)

⇒

T_1	T_2
Read(A)	
write(A)	
Read(B)	
write(B)	
	Read(A)
	write(A)
	Read(B)
	write(B)

$\langle T_1\ T_2 \rangle$

S_1

T_1	T_2
	Read(A)
	write(A)
	Read(B)
	write(B)
Read(A)	
write(A)	
Read(B)	
write(B)	

$\langle T_2\ T_1 \rangle$

S_2

⇓

Concurrent Schedule (Cq) Cq

(try to convert it into serial schedule)

↳ this concurrent schedule is equivalent to which serial schedule?

Cq → S_1 or S_2 (??)

o) trying to convert Cq into S_2 <T_2 T_1>

T_1	T_2
Read(A)	
Wo(A)	
	R(A)
	Wo(A)
R(B)	
Wo(B)	
	R(B)
	Wo(B)

Read(A) write(A) Clash
write(A) Read(A) Clash
(unable to convert into S_2)

o) trying to convert Cq into S_1 <T_1 T_2>

T_1	T_2
R(A)	
Wo(A)	
	R(A)
	Wo(A)
R(B)	
Wo(B)	
	R(B)
	Wo(B)

o) no conflict ∵ diff data item

o) So Cq is conflict equivalent to serial schedule S_1 <T_1 T_2>

o) So Cq is Conflict Serializable.

⇓

Consistent

<T_1 T_2>

⟶

classmate
Date
Page

- $C_{10} \rightarrow$

T_1	T_2
Read(A)	
	Read(A)
	Write(A)
Write(A)	
Read(B)	
Write(B)	
	Read(B)
	Write(B)

$\rightarrow S_1$??

$\rightarrow S_2$

- Can't go for S_1 Read (A), write (A) conflict
- Can't go into S_2 also

→ C_{10} not Conflict Serializable ⇓ inconsistent (getting wrong result)

- $\boxed{\text{Approach II}}$ (testing method) (precedence graph method)

→ Testing for Conflict Serializability

- precedence graph method

$G : (V, E)$

V : Set of transactions

E : Edge

Edge $T_i \rightarrow T_j$ occur if any one of the following condition hold.

$T_i \xrightarrow{\text{if any}} T_j$ (CNC : Cycle not conflict)

(i) T_i execute Read(Q) before T_j execute write(Q)

(ii) T_i execute write(Q) before T_j execute Read(Q)

(iii) T_i execute write(Q) before T_j execute write(Q)

→ If precedence graph contain any one cycle then not conflict serializable.

Scanned by CamScanner

Testing for conflict serializability using precedence graph

① Construct a directed graph where each vertices corresponds to a transaction & ~~eat~~ each ~~edge~~ directed edge represents a conflicting operation

② If the directed graph contain cycle then concurrent schedule is not conflict serializable.

③ If graph contain no cycle than the schedule is called conflict serializable.

Note: The serializability order is determined based on the directed edge.

①

T_1	T_2
Read(A)	
write(A)	
	Read(A)
	write(A)
Read(B)	
write(B)	
	Read(B)
	write(B)

after w(A) in T_1 there is R(A) & w(A) in T_2 (if any one than make edge)

$T_1 \rightarrow T_2$

no $T_2 \rightarrow T_1$

$T_1 \rightarrow T_2$

(no cycle) conflict serializable

$\langle T_1\ T_2 \rangle$

(consistent)

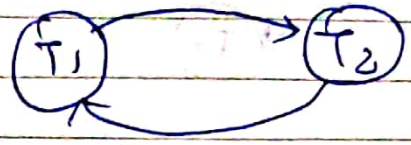
after that

R(A) — w(A)
w(A) — R(A)
w(A) — w(A)

②

→

T_1	T_2
Read(A)	$T_1 \rightarrow T_2$
	Read(A)
	write(A)
write(A)	$T_2 \rightarrow T_1$
Read(B)	
write(B)	
	Read(B)
	write(B)



(cycle formed)

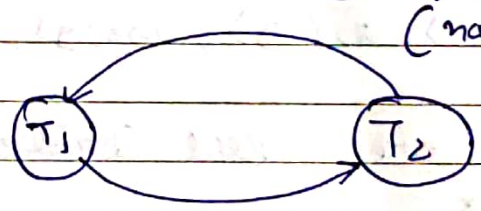
not conflict serializable
(inconsistent)

Question

S: $R_1(A)$ $w_1(A)$ $R_2(B)$ $w_2(B)$ $R_1(B)$ $w_1(B)$ $R_2(A)$ $w_2(A)$

T_1	T_2
Read(A)	
w(A)	
	R(B)
	w(B)
R(B)	
w(B)	
	R(A)
	w(A)

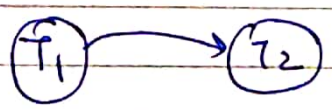
cycle formed
(not conflict serializable)



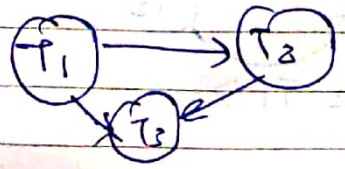
Read & write (A) in T_1
after that anywhere r(A) & w(A) in T_2

o) Serializability order [Topological sorting]

↳ Tells that the concurrent schedule is equivalent to which serial schedule.

① 

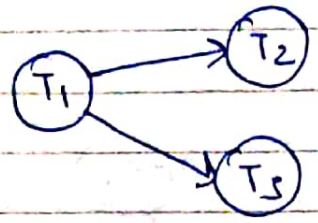
$\langle T_1 T_2 \rangle$ T_1 followed by T_2

② 

$\langle T_1 T_2 T_3 \rangle$

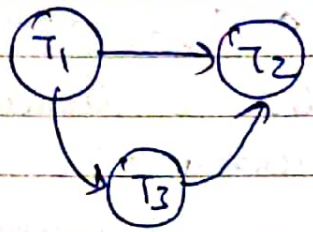
Scanned by CamScanner

③



$\langle T_1\ T_2\ T_3 \rangle$
or
$\langle T_1\ T_3\ T_2 \rangle$

④



$\langle T_1\ T_3\ T_2 \rangle$

Gate 2007

Consider the following schedules involving two transactions. Which one of the following statements is true?

S_1 : $r_1(x), r_1(y), r_2(x), r_2(y), w_2(y), w_1(x)$
S_2 : $r_1(x), r_2(x), r_2(y), w_2(y), r_1(y), w_1(x)$

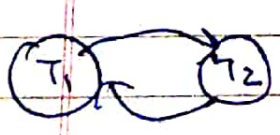
a) Both S_1 & S_2 are conflict serializable
b) S_1 ✓ S_2 ✗
c) S_1 ✗ S_2 ✓
d) S_1 ✗ S_2 ✗

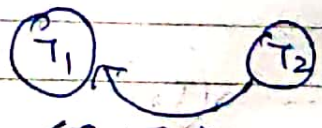
T_1	T_2
$r(x)$	
$r(y)$	
	$r(x)$
	$r(y)$
	$w(y)$
$w(x)$	

S_1

T_1	T_2
$r(x)$	
	$r(x)$
	$r(y)$
	$w(y)$
$r(y)$	
$w(x)$	

S_2





$\langle T_2\ T_1 \rangle$

Scanned by CamScanner

Gate 2009

Consider 4 Schedules S_1, S_2, S_3, S_4

S_1: $r_1(x)$, $r_2(x)$, $r_2(y)$, $w_1(x)$, $w_1(y)$, $w_2(y)$

S_2: $r_1(x)$, $r_2(x)$, $r_2(y)$, $w_1(x)$, $w_2(y)$, $w_1(y)$

S_3: $r_1(x)$ $w_1(x)$ $r_2(x)$ $w_1(y)$ $r_2(y)$ $w_2(y)$

S_4: $r_2(x)$ $r_2(y)$ $r_1(x)$ $w_1(x)$ $w_1(y)$ $w_2(y)$

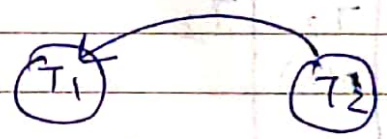
Which of the above schedules are Conflict serializable?

a) S_1 & S_2 c) S_3 only

b) S_2 & S_3 d) S_4 only

S_2:

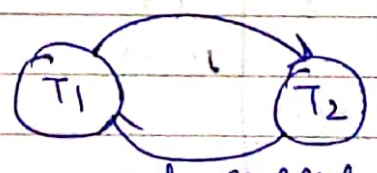
T_1	T_2
$r(x)$	
	$r(x)$
	$r(y)$
$w(x)$	
	$w(y)$
$w(y)$	



$\langle T_2 T_1 \rangle$

S_1:

T_1	T_2
$r(x)$	
	$r(x)$
	$r(y)$
$w(x)$	
$w(y)$	
	$w(y)$



not conflict serializable

Gate 2014

Which one of them is Conflict Serializable?

Scanned by CamScanner

a) $r_1(x)$, $r_2(x)$, $w_1(x)$, $r_3(x)$, $w_2(x)$

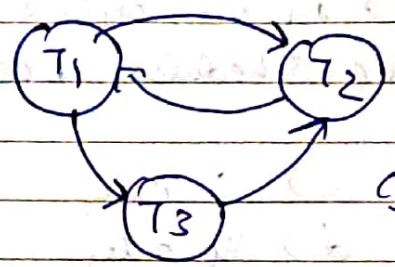
b) $r_2(x)$, $r_1(x)$, $w_2(x)$, $r_3(x)$, $w_1(x)$

c) $r_3(x)$, $r_2(x)$, $r_1(x)$, $w_2(x)$, $w_1(x)$

d) $r_2(x)$, $w_2(x)$, $r_3(x)$, $r_1(x)$, $w_1(x)$

a)

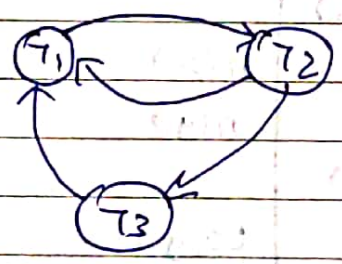
T_1	T_2	T_3
$r(x)$		
	$r(x)$	
$w(x)$		
		$r(x)$
	$w(x)$	



cycle

b)

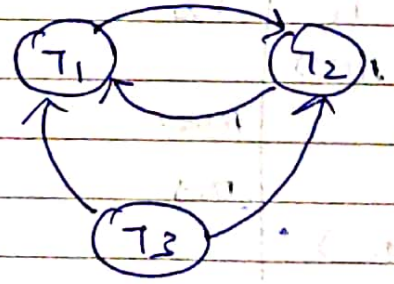
T_1	T_2	T_3
	$r(x)$	
$r(x)$		
	$w(x)$	
		$r(x)$
$w(x)$		



cycle

c)

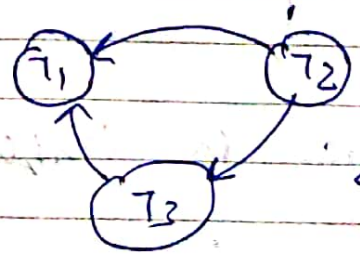
T_1	T_2	T_3
		$r(x)$
	$r(x)$	
$r(x)$		
	$w(x)$	
$w(x)$		



cycle

d)

T_1	T_2	T_3
	$r(x)$	
	$w(x)$	
		$r(x)$
$r(x)$		
$w(x)$		



no cycle

$\langle T_2\ T_3\ T_1 \rangle$

Scanned by CamScanner

Gate 2014

S_1 : $r_1(x)\ r_3(y)\ r_3(x)\ r_2(y)\ r_2(z)\ w_3(y)\ w_2(z)\ r_1(z)\ w_1(x)\ w_1(z)$

S_2 : $r_1(x)\ r_3(y)\ r_2(y)\ r_3(x)\ r_1(z)\ r_2(z)\ w_3(y)\ w_1(x)\ w_2(z)\ w_1(z)$

Which is true?

a) Only S_1 conflict serializable

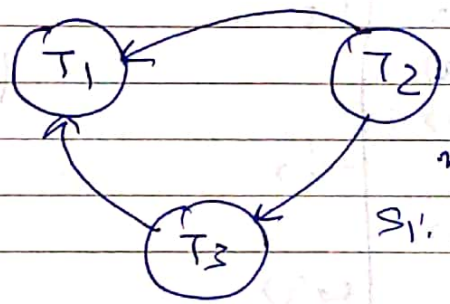
b) Only S_2 " "

c) Both S_1 & S_2 " "

d) Neither S_1 nor S_2 CS

S_1:

T_1	T_2	T_3
$r(x)$		
		$r(y)$
		$r(x)$
	$r(y)$	
	$r(z)$	
		$w(y)$
	$w(z)$	
$r(z)$		
$w(x)$		
$w(z)$		



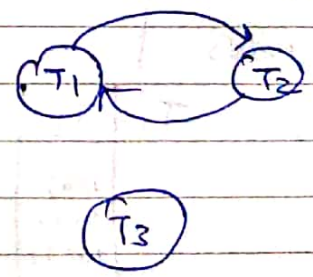
no cycle

S_1: CS

S_2:

T_1	T_2	T_3
$r(x)$		
		$r(y)$
	$r(y)$	
		$r(x)$
$r(z)$		
	$r(z)$	
		$w(y)$
$w(x)$		
	$w(z)$	
$w(z)$		

cycle formed



S_2: NCS

Scanned by CamScanner

Gate 2008

NYO represent the action Y (R for read, W for write) performed by transaction N on object O

S_1: 2RA 2WA 3RC 2WB 3WA 3WC 1RA 1RB 1WA 1WB

S_2: 3RC 2RA 2WA 2WB 3WA 1RA 1RB 1WA 1WB 3WC

S_3: 2RA 3RC 3WA 2WA 2WB 3WC 1RA 1RB 1WA 1WB

Which is true?

a) S_1, S_2, S_3 all ~~ES~~ conflict equivalent to each other.

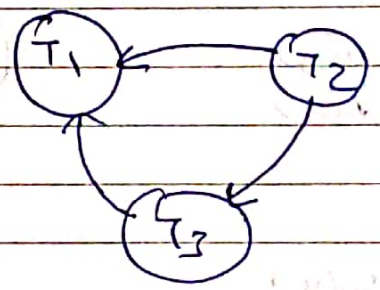
b) No two of S_1, S_2, & S_3 are conflict " " " "

c) S_2 is conflict equivalent to S_3, but not to S_1

d) S_1 " " " " S_2, " " " S_3

S_1:

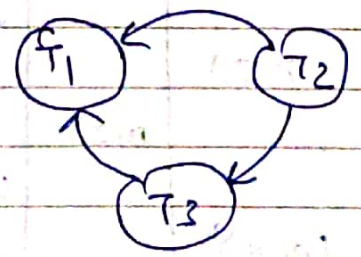
T_1	T_2	T_3
	r(a)	
	w(a)	
		r(c)
	w(b)	
		w(a)
		w(c)
r(a)		
r(b)		
w(a)		
w(b)		



$\langle T_2\ T_3\ T_1 \rangle$

S_2:

T_1	T_2	T_3
		r(c)
	r(a)	
	w(a)	
	w(b)	
		w(a)
r(a)		
r(b)		
w(a)		
w(b)		
		w(c)

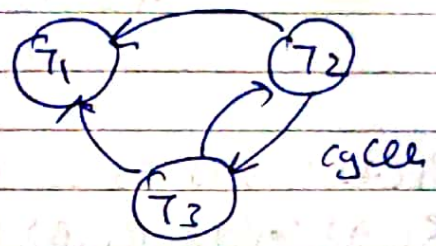


~~$\langle T_3$~~ $\langle T_2\ T_3\ T_1 \rangle$

Scanned by CamScanner

S_1:

T_1	T_2	T_3
	$r(a)$	
		$r(c)$
		$w(a)$
	$w(a)$	
	$w(b)$	
		$w(c)$
$r(a)$		
$r(b)$		
$w(a)$		
$w(b)$		



III rd approach

Conflict equivalent

T_i	T_j	
$R(A)$	$w(A)$	Conflict operation
$w(A)$	$R(A)$	
$w(A)$	$w(A)$	
$R(A)$	$R(A)$	non-conflict operation
operating on diff data Item		

o) Two schedule are said to be conflict equivalent if all the conflicting operations in both the schedule must be executed in same order.

T_1	T_2
$R(a)$	
$w(a)$	
	$R(a)$
	$w(a)$
$R(B)$	
$w(B)$	

S_1

~~R(A) — w(A)~~ conflict operation

$R_1(A) - w_2(A)$: $T_1 \to T_2$

$w_1(A) - R_2(A)$: $T_1 \to T_2$

$w_1(A) - w_2(A)$: $T_1 \to T_2$

T_1	T_2
$R(a)$	
$w(a)$	
	$R(a)$
	$w(a)$
$R(B)$	
$w(B)$	

S_2

$R_1(a) - w_2(a)$

$w_1(a) - r_2(a)$

$w_1(a) - w_2(a)$

or

$T_1 \to T_2$

$T_1 \to T_2$

$T_1 \to T_2$

Scanned by CamScanner

$S_1 \equiv S_2$ (S_1 conflict equivalent to S_2)

Question

S_1: $R_1(A)\ W_1(A)\ R_2(B)\ W_2(B)\ R_1(B)$

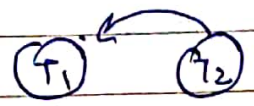
S_2: $R_1(A)\ W_1(A)\ R_1(B)\ R_2(B)\ W_2(B)$

S_1:

T_1	T_2
$r(a)$	
$w(a)$	
	$r(b)$
	$w(b)$
$r(b)$	

Conflict operation

$w_2(b) - r_1(b)$: $T_2 \rightarrow T_1$



$\langle T_2\ T_1 \rangle$

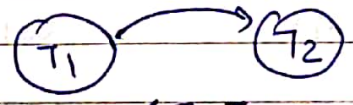
Conflict Serializable

S_2:

T_1	T_2
$r(a)$	
$w(a)$	
$r(b)$	
	$r(b)$
	$w(b)$

Conflict operation

$r_1(b) - w_2(b)$: $T_1 \rightarrow T_2$



$\langle T_1\ T_2 \rangle$

Conflict Serializable

$S_1 \not\equiv S_2$

S_1 not conflict equivalent to S_2

<u>Conflict Serializable:</u>

- A schedule is said to be conflict serializable if it is conflict equivalent to a serial schedule.

* All conflict serializable schedule need not to be conflict equivalent!

Scanned by CamScanner

Gate 2020

T_1	RA			RC		WD		WB	Commit	
T_2		RB	WB		RD		WC			Commit

Which one of the following schedule is conflict equivalent to above schedule?

a) T_1 RA RC WD WB Commit
T_2 RB WB RD WC Commit

b) T_1 RA RC WD WB Commit
T_2 RB WB RD WC Commit

c) T_1 RA RC WD WB Commit
T_2 RB WB RD WC Commit

d) T_1 RA RC WD WB Commit
T_2 RB WB RD WC Commit

$r_2(b) - w_1(b)$
$w_2(b) - w_1(b)$
$r_1(c) - w_2(c)$
$r_2(d) - w_1(d)$

a) $r_2(b) - w_1(b)$
$w_2(b) - w_1(b)$
$r_2(d) - w_1(d)$
$r_1(c) - w_2(c)$

b) $r_2(b) - w_1(b)$
$w_2(b) - w_1(b)$
X $r_2(d) - w_1(d)$
$w_2(c) - r_1(c)$

c) $r_1(c) - w_2(c)$
X

$r_1(c) - w_2(c)$
$T_1 \to T_2$
not $T_2 \to T_1$

d) $r_1(c) - w_2(c)$ ✓
$w_1(d) - r_2(d)$ X
$w_1(b) - r_1(b)$.
$w_1(b) - w_1(d)$

Scanned by CamScanner

0) **Equivalent schedule** : Two schedules S_1 & S_2 are said to be equivalent if they produce the same final database state.

3 types :

1) Result equivalent schedule
2) Conflict equivalent schedule (done)
3) View equivalent schedule.

* **Result equivalent schedule** Two schedule are said to be result equivalent if they produce the same final database state for some initial value of data.

	S_1	S_2
	A=100	A=100
A=100	R(A)	R(A) A=100
	A = A+10	A = A*1.1
	w(A)	w(A)
	$\boxed{A=110}$	$\boxed{A=110}$

S_1 & S_2 are result equivalent ($S_1 \equiv S_2$)

	S_1	S_2
	A=200	A=200
A=200	R(A)	R(A) A=200
	A = A+10	A1 = A*1.1
	w(A)	w(A)
	$\boxed{A=210}$	$\boxed{A=220}$

S_1 not result equivalent to S_2

Question

Is the following schedule are result equivalent for the initial value of x & y is (2, 5) respectively

$x = 2$, $y = 5$

Scanned by CamScanner

S_1		S_2		S_3	
T_1	T_2	T_1	T_2	T_1	T_2
$R(x)$	$x=2$	$x=2$	$R(x)$	$R(x)$	$x=2$
$x=x+5$	$x=7$	$x=6$	$x=x*3$	$x=x+5$	$x=7$
$w(x)$	$x=7$	$x=6$	$w(x)$	$w(x)$	$x=7$
$x=7$	$R(x)$	$R(x)$	$x=6$	$R(y)$	$y=5$
$x=21$	$x=x*3$	$x=x+5$	$x=11$	$y=y+5$	$y=10$
$x=21$	$w(x)$	$w(x)$	$x=11$	$w(y)$	$y=10$
$R(y)$	$y=5$	$R(y)$	$y=5$	$x=7$	$R(x)$
$y=y+5$	$y=10$	$y=y+5$	$y=10$	$x=21$	$x=x*3$
$w(y)$	$y=10$	$w(y)$	$y=10$	$x=21$	$w(x)$
$(x=21, y=10)$		$(x=11, y=10)$		$(x=21, y=10)$	

$\therefore$ (S_1 is result equivalent to S_3)

(2) View serializable

S: any Concurrent schedule

S': any serial schedule of S

o) Two schedule S and S' said to be view equivalent if all 3 condition are satisfied.

initial read
final write
write-Read

(1) for each data item (Q) if transaction T_i perform the initial read on data item (Q) in schedule S then transaction T_i must in schedule S' also perform the initial read on data Item (Q).

(2) for each data Item (Q) if transaction T_i perform the final write (Q) operation in schedule S, then T_i must in schedule S' also perform the final write (Q) operation

Scanned by CamScanner

③ for each data item (Q) if Transaction T_i read the value of data Item (Q) that was produced by T_j in Schedule S then T_i must in Schedule S' also read the value of data Item (Q) that was produced by T_j.

(or)

b) <u>View equivalent</u> S_1 & S_2 are said to be view equivalent

① Initial Read of S_1 & S_2 should be same
↳ on each data item

A=10
B=10

	T_1	T_2	T_3
A=10, B=10	R(A)		
	R(B)		
		w(B)	
			w(B)

S_1

	T_1	T_2	T_3
		w(B)	
A=10, B≠10	R(A)		
	R(B)		
			w(B)

S_2

w(B) → ∵ Some oper performed on B here

R(B) ← will read diff value not 10

Initial Read on A : T_1
" " " B : T_1

So: $S_1 \neq S_2$

Initial read on A : T_1
Initial " " B : not T_1
(∵ w(B) in T_2 is first)

② Final write on each data item of S_1 & S_2 must be same

→

50000 → 1lakh 25lakh → final write

T_1	T_2	T_3
$w(A)$		
$w(B)$		
	$w(A)$	
		$w(A)$
		$w(B)$

S_1

final write on A : T_3

B : T_3

T_1	T_2	T_3
$w(A)$		
		$w(A)$
$w(B)$		
	$w(A)$	
		$w(B)$

S_2

final write on A : T_2 → not same

B : T_3

$S_1 \neq S_2$

③ updated read [write - read] sequence on each data-item of S_1 & S_2 must be same

T_1	T_2	T_3
$w(A)$		
	$w(A)$	
		$R(A)$
		$w(A)$

updated Read : $T_2 \rightarrow T_3$

T_1	T_2	T_3
$w(A)$		
		$R(A)$
	$w(A)$	
		$w(A)$

updated read : $T_1 \rightarrow T_3$

$S_1 \neq S_2$

Question

T_1	T_2
$r(A)$	
$w(A)$	
	$r(A)$
	$w(A)$
$r(B)$	
$w(B)$	
	$r(B)$
	$w(B)$

$S_1 < T_1 T_2 >$

T_1	T_2
$r(A)$	
$w(A)$	
	$r(A)$
	$w(A)$
	$r(B)$
	$w(B)$
$r(B)$	
$w(B)$	

$S_2 < T_2 T_1 >$

initial read ✓

final write ✓

write - read ✓

T_1	T_2
	$r(A)$
	$w(A)$
	$r(B)$
	$w(B)$
$r(A)$	
$w(A)$	
$r(B)$	
$w(B)$	

initial read ✗

final write ✗

write - read ✗

C_{11}

Scanned by CamScanner

C_{11} is view equivalent to $S_1 \langle T_1 T_2 \rangle$

$C_{11} \rightarrow$ Conflict serializable

$\Rightarrow S \langle T_1 T_2 \rangle$

2) C_{22}

T_1	T_2
$r(A)$	
	$r(A)$
	$w(A)$
$w(A)$	
$r(B)$	
$w(B)$	
	$r(B)$
	$w(B)$

$S_1 \langle T_1 T_2 \rangle$

T_1	T_2
$r(A)$	
$w(A)$	
$r(B)$	
$w(B)$	
	$r(A)$
	$w(A)$
	$r(B)$
	$w(B)$

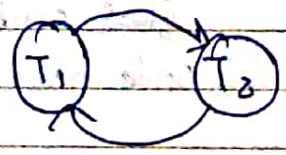
$S_2 \langle T_2 T_1 \rangle$

T_1	T_2
	$r(A)$
	$w(A)$
	$r(B)$
	$w(B)$
$w(A)$	
$r(B)$	
$w(A)$	
$r(A)$	
$w(A)$	
$r(B)$	
$w(B)$	

Initial read ✓
final write ✗
write-read

Initial read ✗
final write
write-read

So. C_{22} is not view serializable
C_{22} is not conflict serializable



C_{22} not serializable
$\Downarrow$
inconsistence

Note:

→ If a schedule is conflict serializable then its already view serializable

→ If a schedule is not conflict serializable then we have to check for view serializable. (may or may not)

Scanned by CamScanner

Note Question

o) If 3 transaction

3! = 6 Serial schedule

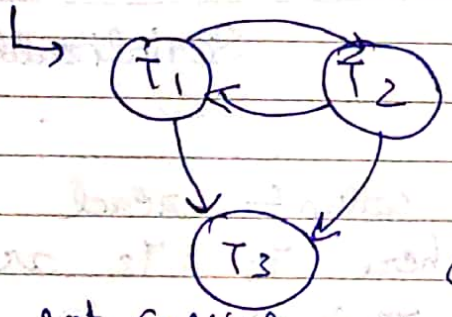
<T1 T2 T3>
<T1 T3 T2>
<T2 T1 T3>
<T2 T3 T1>
<T3 T1 T2>
<T3 T2 T1>

don't check individually for each serial schedule

* Make a dummy & check it

(If saying check for view serializable, then check conflict, if conflict serializable then view serializable hoga hi hoga)

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



not conflict serializable

not conflict serializable (now check for view serializable kyoz it may be or may not be)

not conflict serializable

o) now make dummy to check view serializable

(initial read is performed by T_1)

So <T2 T1 T3>
<T2 T3 T1>
<T3 T1 T2>
<T3 T2 T1>

X (initial read X)

Scanned by CamScanner

o) left with $\langle T_1\ T_2\ T_3 \rangle$ & $\langle T_1\ T_3\ T_2 \rangle$

→ final write performed by T_3 so $\langle T_1\ T_3\ T_2 \rangle$ ✗

→ now left with $\langle T_1\ T_2\ T_3 \rangle$ → check for this
↳ is possible (lets check)

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

$\langle T_1\ T_2\ T_3 \rangle$

initial read ✓
final write ✓
write-read (no write-read sequence)
(view serializable) ✓

o) So.

T_1	T_2	T_2
$R(Q)$		
	$W(Q)$	
$W(Q)$		
		$W(Q)$

→ not conflict serializable but view serializable (Blind write)
Serializable schedule

* **Blind write** (writing without read
↳ like here T_2 & T_3 are blind write (writing without reading)

Note

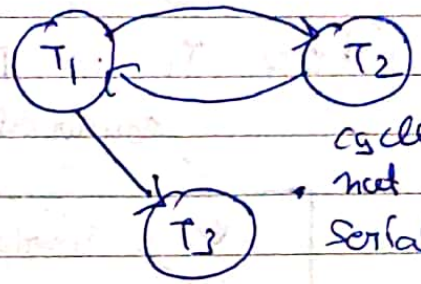
then	Conflict serializable ✓ view serializable ✓	Conflix ✗ view ✓	Conflict ✗ view ✗
	Serializable schedule or Conflict serializable schedule	~~Serial~~ Serializable schedule	not serializable schedule

Scanned by CamScanner

Question

Blind write

T_1	T_2	T_3
	R(A)	
	R(B)	
w(B)		
		R(B)
w(A)		
	w(A)	
		w(A)



cycle
not Conflict Serializable

(now check for view Serializable) write

(∵ initial read T_2) ✗ { $\langle T_1, T_2, T_3 \rangle$, $\langle T_1, T_3, T_2 \rangle$ } ✗

final write A : T_3
" write B : T_1
initial read A : T_2
initial read B : T_2

$\langle T_2, T_1, T_3 \rangle$
$\langle T_2, T_3, T_1 \rangle$

(∵ initial read T_2) ✗ { $\langle T_3, T_1, T_2 \rangle$, $\langle T_3, T_2, T_1 \rangle$ } ✗

Check for $\langle T_2, T_1, T_3 \rangle$

T_2	T_1	T_3
R(A)		
R(B)		
	w	

T_1	T_2	T_3
	R(A)	
	R(B)	
	w(A)	
w(B)		
w(A)		
		R(B)
		w(A)

initial read ✓
final write ✓
write-read ✓

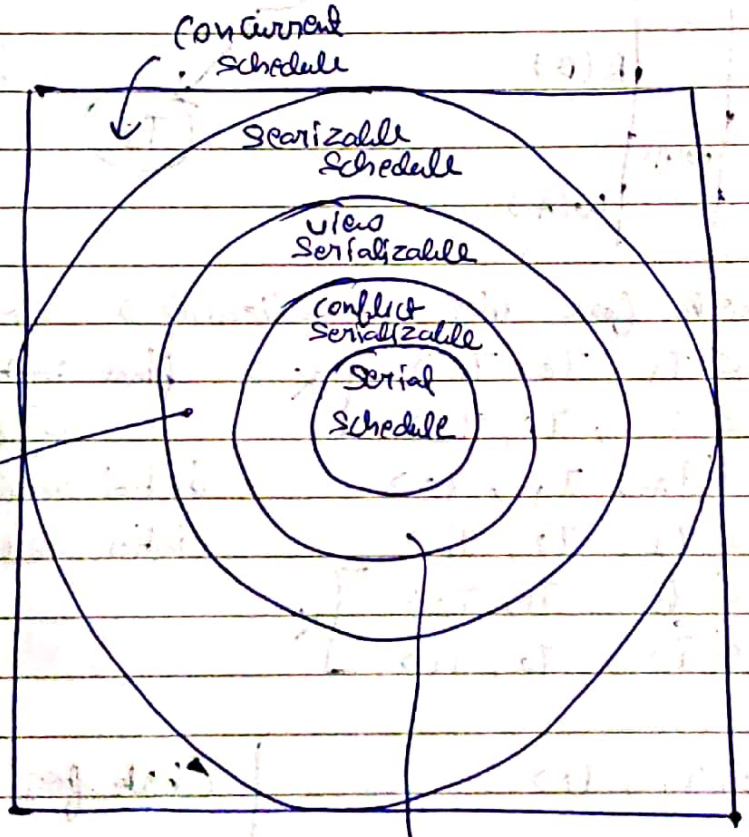
Check for $\langle T_2, T_3, T_1 \rangle$

T_1	T_2	T_3
	R(A)	
	R(B)	
	w(A)	
		R(B)
		w(A)
w(B)		
w(A)		

initial read ✓
final write ✗
write-read ✗

Scanned by CamScanner

not conflict Serializable
view "
Serializable Schedule <T2 T1 T3>
(here T1 is Blind write)



Blind writes
(if view Serializable & not conflict Serializable then case of Blind writes)

If conflict Serializable then view Serializable also

NOTE o) If the schedule is conflict Serializable Schedule we can say its Serializable Schedule.
o) But if Schedule is not conflict Serializable, may or may not be Serializable (depend on view Serializable)

→

Scanned by CamScanner

// only sufficient condition

if (Acyclic precedence graph)
{
Conflict searizable Schedule
(means searizable)
}

else
{
not conflict serializable schedule
↳ may/may not be searizable
}

// Both sufficient & necessary condition

if (S satisfied view searizable schedule condition)
{
schedule is serializable
}

else
{
not serializable schedule
}

Note: A schedule is serializable, if it is either conflict serializable or view serializable or both.

→ If serializable schedule then consistent.

* Problem broz of concurrent execution (conflict instruction)

1. WR (write - read) problem / uncommitted read / dirty read
2. RW (Read - write) problem / non repeatable read / unrepeatable read
3. WW (write - write) problem / lost update problem
4. Phantom tuple problem.

1. Write - read problem

T_1	T_2	
W(A)		
	R(A)	uncommitted read or dirty read

uncommitted read:
Transaction T_2 Read a data item A that is UPDATED (written) by uncommitted transaction T_1

Scanned by CamScanner

$A = 1050$, $B = 2000$

T_1	T_2
$A = 1050$ R(A)	
$A = A - 50$	
$A = 1000$ W(A)	
	R(A) $A = 1000$
	$x = A * 0.1$ $x = 100$
ROLLBACK	$A = A + x$
	W(A) $A = 1100$
$B = 2000$ R(B)	
$B = B + 50$	
W(B)	

$A = \cancel{1050}$ ∴ T_2 works is lost overwrite

$A = 1000$

$A = 1100$

ROLLBACK

$A \neq 1100$ } create prob

$A = 1050$

∴ ~~T_2 is~~ reading ~~wrong~~ value

write

$T_1 \rightarrow$ Read T_2

failure occurs here DO ROLLBACK

T_2 depend on T_1, so if T_1 fails so T_2 should also ROLLBACK)

(2) Read-write problem

T_1	T_2
R(A)	
	W(A)

#copies = 10

$A = 10$

eg) Library db,

```
R(A)
if (A>0)   // if copy available
{
  A = A-1
  write (A)   // issue copy
}
else
{
  no book issue
}
```

	T_1	T_2
$A = 10$	R(A)	
	if (A>0)	
	{ A = A-1	
		R(A) $A = 10$
		if (A>0)
		{
		A = A-1
		W(A) $A = 9$ (update)
		}
	$A = 9$ update W(A)	
	}	

2 book issued but remaining is 9

2
9
11 (inconsistent)

Unrepeatable / non repeatable read

A = 20,000

T_1	T_2	
A=20,000 R(A)		
	R(A)	A = 20,000
	A = A-15000	A = 5000
	W(A)	A = 5000
A = 5000 R(A)		
A = A-10000		
W(A)		

A = 5000 - 10000 (problem)

o) when a transaction tries to read a value of data item twice & another trasaction update the same data item between the two read operation of first transaction as a result first transaction read a varied / different value of same data item during its execution this is known as unrepeatable read.

(3) Write-write problem / lost update problem

A = 1000

T_1	T_2
W(A)	
	W(A)

	T_1	T_2
A = 1000	R(A)	
A = 950	A = A-50	
		R(A) A = 1000
		X = A*0.1 A = 1100
		A = A+10
	update W(A) A = 950	
		W(A) update A = 1100

↳ this is lost

o) Suppose that the operation of T_1 & T_2 are interleaved in such a way that T_2 read the value of account A before T_1 updates, now when T_2 updates then the value of account A in the db.

o) The value of account A updated by T_1 is overwritted by T_2.

lost update prob ← Hence update is lost

o) TEST

→ If there is any two write operation of different transaction & b/co that there is no read operation, the second write operation overwrite the first write hence it cause lost of first updation (Lost update)

(4) Phantom Tuple problem:

Emp

eno	ename	salary
1	A	5000
2	B	3000
3	C	7000

o/p

eno	ename	salary
1	A	5000
3	C	7000

T_1	T_2
Select * from Emp where Salary > 3000	
	Insert into Emp value <4, D, 4000>
Select * from Emp where sal > 3000	

o/p

eno	ename	salary
1	A	5000
3	C	7000
4	D	4000

→ phantom tuple just b/coz of concurrent execution (getting more rows as o/p for same query in same database)

o) Transaction T_1 may read set of tuples from a table based on some condition. Now suppose that transaction T_2 insert a new tuple & that tuple also satisfy the where clause condition of T_1. If T_1 is repeated then T_1 will set a new tuple/row that previously did not exists called a phantom tuple.

Scanned by CamScanner

Conflict
view

Serializability (here deal with Consistency & Isolation property)
↓
Consistent or not ?

Recoverability (now to deal with atomicity & durability property)
↓ if failure
Recover or not ?

o) <u>Dependency</u>

①

T_1	T_2
$w(A)$	
	$R(A)$
commit	

T_2 depend on T_1 / uncommitted read

②

T_1	T_2
$w(A)$ commit	
	$R(A)$

no dependency

③

T_1	T_2
$w(A)$	
rollback	
	$R(A)$

no dependency

④

T_1	T_2
$w(A)$	
	$w(A)$

no dependency

⑤

T_1	T_2
$w(A)$	
	$w(A)$
	$R(A)$

no dependency

reading value performed by same transaction

not uncommitted read

⑥

T_1	T_2
$w(A)$	
	$w(B)$
	$R(A)$
$R(B)$	

T_2 depend on T_1
T_1 depend on T_2

uncommited read : update (write) by one transaction & read by another transaction.

Scanned by CamScanner

0) Recoverability

T_1	T_2
~~W(A)~~	
R(A)	
⋮	
W(A)	
	R(A)
	⋮
	W(A)
	commit

Rollback

* failure

Irrecoverable / non recoverable

0) Recoverable Schedule

T_1	T_2
W(A)	
	R(A)
C/R	
commit / rollback	commit

Recoverable Schedule

0) A recoverable Schedule is one where for each pair of transaction T_i & T_j such that T_j read a data ~~item~~ item that was previously written by T_i, then commit operation of T_i appear before commit of T_j.

(or)

0) If T_2 depend on T_1 (T_2 read data item that is updated by T_1) then commit operation of T_2 should be delayed until ~~com~~ commit / Rollback of T_1.

Note! Recoverable Schedule may / may not free from

1. RW prob
2. WW prob
3. WR prob

Scanned by CamScanner

①

T_1	T_2
R(A)	
w(A)	
	R(A)
w(B)	
	R(B)
Commit	
	Commit

(Recoverable Schedule)

Recoverable schedule but uncommitted read / dirty read ↓ wR problem exist.

② Cascading ~~Roll~~ Rollback are possible

T_1	T_2	T_3	T_4
w(A)			
	R(A)		
		R(A)	
			R(A).

T_2, T_3 & T_4 depends on T_1 if T_1 fails then due to dependency T_2, T_3, T_4 also Rollback (cascading rollback) then T_1 Rollback

* Cascadelen Schedule / cascadelen Rollback recoverable

T_1	T_2
w(A)	
C/R	
	R(A)

Cascadelen Schedule

o) A cascadelen schedule is one where for each pair of transaction T_i, T_j such that T_j read a data item that ~~for~~ was previously written by T_i then commit of T_i appear before read of T_j.

Note

o) No uncommitted Read are allowed.

o) Cascadelen Schedule
- ↳ free from wR prob (no dirty / uncommitted read)
- ↳ no cascading rollback
- ↳ But suffer from RW & WW problem

Scanned by CamScanner

T_1	T_2
R(A)	
	w(A)
R(B)	
	w(B)
	C/R
C	

Cascadless
but RW prob

T_1	T_2
w(A)	
	w(A)
	Commit
Commit	

Cascadless
but WW prob.

* Strict / strict recoverable Schedule :-

T_1	T_2
w(A)	
C/R	
	R(A) / w(A)

Strict recoverable

a) If one transaction T_1 perform write on data item A then other transaction T_2 not allowed to perform read of A or write of A [w(A) / R(A)] until commit or rollback of T_1, called strict recoverable schedule.

Strict recoverable Schedule
- → recoverable schedule
- → Cascadless schedule
- → No WR / uncommitted / dirty read prob
- → No WW / lost update problem
- → no Cascading Rollback.

*) But RW problem there

→

Scanned by CamScanner

Imp

T_1	T_2
$w(A)$	
	$R(A)$
C/R	
	C

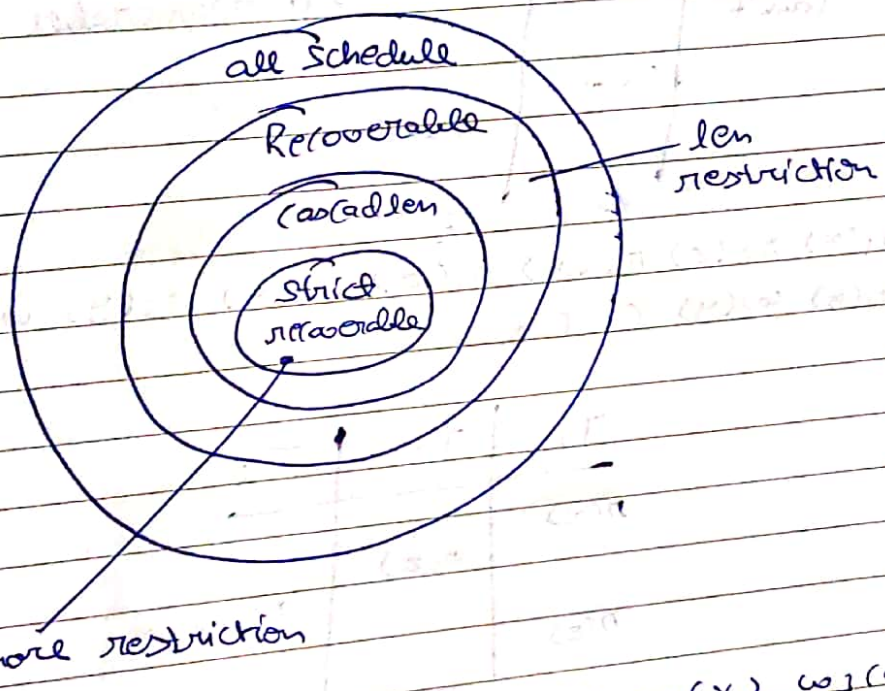
Recoverable

T_1	T_2
$w(A)$	
C/R	
	$R(A)$

Cascadeless

T_1	T_2
$w(A)$	
C/R	
	$w(A)/R(A)$

Strict recoverable



Question

(1) $r_1(x)$ $r_2(z)$ $r_1(z)$ $r_3(x)$ $r_3(y)$ $w_1(x)$ $w_3(y)$
$r_2(y)$ $w_2(z)$ $w_2(y)$ C_1 C_2 C_3

T_1	T_2	T_3
$R(x)$		
	$R(z)$	
$R(z)$		
		$R(x)$
		$R(y)$
$w(x)$		
		$w(y)$
	$r(y)$	
	$w(z)$	
	$w(y)$	

Scanned by CamScanner

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
$r(z)$		
		$r(x)$
		$r(y)$
$w(x)$		
		$w(y)$
	$r(y)$	
	$w(z)$	
	$w(y)$	
Commit		
	Commit	
		Commit

Not recoverable Schedule

(2) $r_1(x)\ r_2(z)\ r_3(x)\ r_1(z)\ r_2(y)\ r_3(y)\ w_1(x)\ C_1\ w_2(z)\ w_3(y)\ w_2(y)\ C_3\ C_2$

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
		$r(x)$
$r(z)$		
	$r(y)$	
		$r(y)$
$w(x)$		
Commit		
	$w(z)$	
		$w(y)$
	$w(y)$	
		Commit
	Commit	

Recoverable ✓
Cascadless ✓

③ $r_3(x)\ r_1(x)\ w_3(x)\ r_2(x)\ w_1(y)\ r_2(y)\ w_2(x)\ C_3\ C_1\ C_2$

T_1	T_2	T_3
		$r(x)$
$r(x)$		
		$w(x)$
	$r(x)$	
$w(y)$		
	$r(y)$	
	$w(x)$	
		Commit
Commit		
	Commit	

recoverable ✓
Cascadeless ✗

④ $r_1(x)\ r_2(x)\ w_1(y)\ w_2(y)\ r_2(y)\ C_1\ C_2$

T_1	T_2
$r(x)$	
	$r(x)$
$w(y)$	
	$w(y)$
	$r(y)$
Commit	
	Commit

recoverable ✓
no dependency
cascadeless ✓
Strict recoverable ✗

∵ after w, R not there
w → R (same transaction)

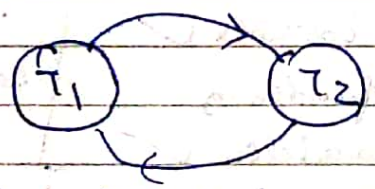
Gate 2004

Consider S of transaction T_1 & T_2

→

T_1	T_2
R(A)	
A = A - 10	
	R(A)
	temp = 0.2 * A
	W(A)
	R(B)
W(A)	
R(B)	
B = B + 10	
W(B)	
	B = B + temp
	W(B)

a) S is Serilizable only as T_1, T_2
b) S is " " " T_2, T_1
c) " " " only as T_1, T_2 & T_2, T_1
d) " " not Serializable either as T_1, or as T_2



not Conflict Serializable

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

$\langle T_1 T_2 \rangle$

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

initial read
final write ✗
write-read ✗

$\langle T_2 T_1 \rangle$

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

initial read
final write ✗

Scanned by CamScanner

Gate 2005

Amongst the ACID properties of transaction the 'Durability' property requires that the changes made to db by a successfull transaction persist.

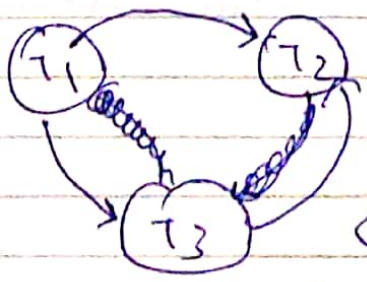
a) Except in case of operating sys crash
b) " " " of disk crash
c) " " " " power failure
d) Always, even if there is failure of any kind

Gate 2010

T_1	T_2	T_3
R(x)		
	R(y)	
		R(y)
	w(y)	
w(x)		w(x)
	r(x)	
	w(x)	

Which one of the schedule below is correct serialization of above?

a) $T_1 \rightarrow T_3 \rightarrow T_2$
b) $T_2 \rightarrow T_1 \rightarrow T_3$
c) $T_2 \rightarrow T_3 \rightarrow T_1$
d) $T_3 \rightarrow T_1 \rightarrow T_2$



no cycle

~~<T1, T3, T3>~~
<T1 T3 T2>

Initial read or : ~~d~~

Gate 2015

Consider the following transaction involving two bank accounts x & y

read(x) ; x := x - 50 ; write(x) ; read(y) ; y := y + 50, write(y)

The constraint that the sum of accounts x & y should remain constant is

a) atomicity
b) consistency
c) isolation
d) durability

Gate 2016

Which one of the following is NOT part of ACID

a) Atomicity
b) Consistency
c) Isolation
d) Deadlock freedom

Gate 2016

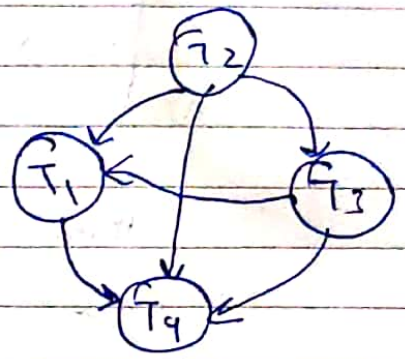
Suppose a db schedule S involves transactions $T_1, \ldots T_n$. Construct the precedence graph of S. If S is serializable which one of following ordering of vertices is guaranteed to yield a serial schedule?

a) Topological order
b) Depth-first "
c) Breadth " "
d) ascending order of transaction indices

Scanned by CamScanner

Gate 2014

T_1	T_2	T_3	T_4
	R(x)		
		w(x)	
		commit	
w(x)			
commit			
	w(y)		
	R(z)		
	commit		
			R(x)
			R(y)
			commit



no cycle

which one is correct?

a) S is conflict seri but not recoverable

b) S is not CS but is recoverable

c) S is CS & R

d) CS X R X

Q) find # concurrent schedule

T_1	T_2
R(A)	R(B)
w(A)	w(B)

all possibility

T_1	T_2
R(A)	
w(A)	
	R(B)
	w(B)

T_1	T_2
	R(B)
	w(B)
R(A)	
w(A)	

T_1	T_2
R(A)	
	R(B)
w(A)	
	w(B)

T_1	T_2
R(A)	
	R(B)
	w(B)
w(A)	

(too serial schedule)

Scanned by CamScanner

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

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

(total 2 serial schedule + 4 ~~concurrent~~ non-serial schedule)

R(A) W(A) R(B) W(B)
<$T_1 T_2$>

R(B) W(B) R(A) W(A)
<$T_2 T_1$>

→ R(A) R(B) W(A) W(B)
→ R(A) R(B) W(B) W(A)
→ R(B) R(A) W(B) W(A)
→ R(B) R(A) W(A) W(B)

$$\boxed{\text{Total \# Concurrent schedule} = \frac{(n_1+n_2)!}{(n_1)!\,(n_2)!}}$$

n_1 & n_2 are # operations in transaction T_1 & T_2 respectively

$$\text{\# Concurrent schedule} = \frac{(2+2)!}{2! \times 2!} = \frac{4!}{4} = 2 \times 3 = 6$$

= 6 Schedule

o) The no of concurrent schedule that can be formed over two transaction T_1, T_2 having n_1 & n_2 operations respectively, is

$$= \frac{(n_1+n_2)!}{(n_1)! \times (n_2)!}$$

⇒ Serial & non-serial both are included

$$\boxed{\text{\# non-Serial} = \text{\# Concurrent Schedule} - \text{\# Serial Schedule}}$$

→ $m!$ m : # transactions

$= 6 - 2 = 4$

Scanned by CamScanner

Note! The no of Concurrent Schedule that can be formed over m transactions having $n_1, n_2, n_3, \ldots n_m$ operations respectively.

$$\Rightarrow \boxed{\frac{(n_1+n_2+\cdots+n_m)!}{(n_1!)(n_2!)\cdots(n_m!)}}$$

Question

find total no of non Serial schedule by three transaction having 2, 3 & 4 operation respectively.

$$= \frac{9!}{2!\times 3!\times 4!} - 3!$$

$$= \frac{5\times 6\times 7\times 8\times 9}{1\times 2\times 1\times 2\times 3} - 6$$

$$= 5\times 7\times 4\times 9 - 6 = 1254$$

---

o) **Implementation of ~~concurren~~ Concurrency control Scheme**
↳ with the help of LOCK BASED Protocol

o) **Lock based protocol**

→ which require that all data item must be accessed in a mutual exclusive manner i.e when one transaction is accessing the data item then no other transaction simultaneously update the data item.

→ Lock is a variable used to identify the status of data item.

→ Transaction should request for a Lock before using the data item.

→ If Lock is free (lock not taken by any other transaction on that data item) then it will be granted otherwise transaction must be wait

Scanned by CamScanner

until lock become available.

To Implement Lock Based Protocol we need two locks
- ① Shared Lock [S-Lock]
- ② Exclusive Lock [X-Lock]

① Shared Lock [S] => only read

Lock-S(A) → if lock free then granted
Read(A) // use data item
Unlock-S(A) → release the lock
(once work is completed then release the lock)

② Exclusive Lock [X] => write

Lock-X(A)
write(A) or write(A) Read(A)
or Read(A) write(A)
unlock-X(A)

* Compatibility b/w locks

T_i \ T_j	S	X
S	Yes	No
X	No	No

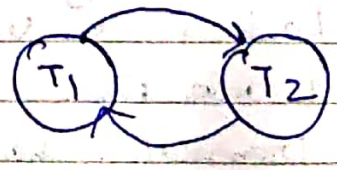
R-R : no conflict
R-W : conflict
W-R : "
W-W : "

o) If Jen I am using shared lock then you can also use shared lock (∵ Read-Read no conflict)

o)

T_1	T_2
Read(A)	
A=A-100	
W(A)	
	R(A)
	R(B)
R(B)	
B=B+100	
W(B)	

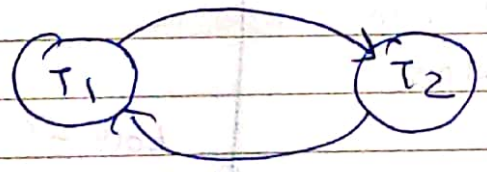
=> want to apply lock in this



cycle
not conflict serializable

Scanned by CamScanner

T_1	T_2	Lock Manager
Lock-X(A)		grant-X(A, T_1)
R(A)		
A = A-100	// Lock-S(A)	// not allowed → bcoz already exclusive lock on A taken
W(A)		
unlock-X(A)		release / revoke-X(A, T_1)
	Lock-S(A)	grant-S(A, T_2)
	read(A)	
	unlock-S(A)	revoke-S(A, T_2)
	Lock-S(B)	grant-S(B, T_2)
	read(B)	
	unlock-S(B)	revoke-S(B, T_2)
Lock-X(B)		grant-X(B, T_1)
Read(B)		
B = B+100		
write(B)		
unlock-X(B)		revoke-X(B, T_1)



Cycle
(Not conflict Serializable)

- We don't use this basic lock concept bcoz after applying lock also we are getting cycle (not conflict Serializable).
- Therefore we go for 2PL.

A 2PL (2 phase locking protocol)

→ In 2PL each transaction issue Lock & unlock

Scanned by CamScanner

request in 2 phase.

- → growing phase
- → shrinking phase

① **Growing Phase** (lock acquire) : In the growing phase transaction may obtain the lock [Acquire the lock] but not release any lock.

② **Shrinking phase** (lock release) : In the shrinking phase transaction may release the lock but must not ask for any new lock.

=> Each transaction finish first its growing phase then shrinking phase.

T_1	T_2
R(A)	
A=A-100	
W(A)	
	R(A)
	R(B)
R(B)	
B=B+100	
W(B)	

2PL =>

T_1	T_2	Lock manager
Lock-X(A)		yes
R(A)		
A=A-100		
W(A)		
	Lock-S(A)	X No [wait]
Lock-X(B)		yes
* unlock-X(A)		
	Lock-S(A)	yes
	Read(A)	
	Lock-S(B)	X No [wait]
Read(B)		
B=B+100		
W(B)		
unlock-X(B)		
	Lock-S(B)	yes
	* unlock-S(A)	
	Read(B)	
	unlock-S(B)	

Since growing phase of T_1 completed

T_1 ⇄ T_2

cycle

Scanned by CamScanner

$T_1 \rightarrow T_2$ $\langle T_1 T_2 \rangle$

Conflict Serializable

o) ~~Let~~ **Lock Point:** Position of Last lock operation or first unlock (Shrinking) operation or from where Shrinking phase Start

(or)

The point at which the transaction has obtained its final lock is called lock point.

→ Lock point are used to determine the serializability order of concurrent schedule.

T_1	T_2	T_3
	*	
*		
		*

* → Lock point (shrinking phase)

Serializability order $\langle T_2 \; T_1 \; T_3 \rangle$

Note:

o) 2PL always ensure conflict serializable schedule
↳ Serializability

o) Serializability order determined by the lock point.

o) 2PL may or may not be free from irrecoverability.

T_1	T_2
R(A)	
W(A)	
	R(A)
	Commit
Commit	

Irrecoverable schedule

T_1	T_2
Lock-X(A)	
R(A)	
W(A)	
Unlock-X(A)	
	Lock-S(A)
	R(A)
	Unlock-S(A)
	Commit
Commit	

→ Irrecoverable schedule but followed by 2PL

Scanned by CamScanner

o) 2PL suffer from Deadlock

① S_1

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

2PL ~~Off~~ ⇒

T_1	T_2	Lock manager
Lock-X(A)		yes
Write(A)		
	Lock-S(B)	yes
	Read(B)	
Lock-X(B)		no
	Lock-S(A)	no

deny by T_2

deny by T_1

Deadlock

② S_2

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

2PL ⇒

T_1	T_2
Lock-S(A)	
Read(A)	
	Lock-X(B)
	Write(B)
Lock-S(B)	
	Lock-X(A)

deny by T_2

deadlock

deny by T_1

o) 2PL suffer from Starvation.

→

S(A) (no problem = Read-Read
S(A) no conflict)

	T_1	T_2	T_3	T_4
		S(A)		
deny by T_2 ←	X(A)			
			S(A) ✓	
deny by T_3 ↳		U(A)		
				S(A) ✓
			U(A)	
→	X(A)			
deny by T_4				U(A)
	X(A)			

granted

	S	X
S	✓	X
X	X	X

* **Strict 2PL:**

→ 2PL + all exclusive Lock (X-Lock) must be hold by the transaction until Commit / Rollback

(or)

→ 2PL + exclusive lock must be release by the transaction after commit / rollback.

T_1	T_2
W(Q)	
C/R	
	R(Q)/W(Q)

Strict recoverable

T_1	T_2
X(A)	
W(A)	
C/R	
U(A)	
	S(A)/X(A)
	R(A)/W(A)

strict recoverable schedule

If T_1 not releasing Lock-X neither T_2 can take Lock-S nor Lock-X

Ⓒ∞

X(A)
S(B)
S(C)
X(D)
⋮
U(B)] Shared
U(C)] lock can be unlock before C/R
C/R
U(A)] exclusive (Should be unlock after
U(D)] lock C/R)

Strict 2PL
- → ensure conflict Serializable Schedule
- → equivalent serial schedule (Serializability order) is determined by lock point
- → ensure strict recoverable schedule
  - → cascadeless
  - → Recoverable Schedule
  - → no cascading Rollback

o) May not be free from
- → deadlock
- → Starvation

* Rigorous 2PL
→ 2PL + all locks [Shared & exclusive [S/X]] must be hold by the transaction until commit or rollback

(or)

→ 2PL + All locks release after Commit / Rollback

Scanned by CamScanner

c) Rigorious 2PL
- ↳ Strict 2PL
- ↳ But suffer from
  - ↳ deadlock
  - ↳ Starvation

S(A)
X (B)
S(C)
X (D)
=
=
=
C/R
U(A)
U(B)
U(C)
U(D)

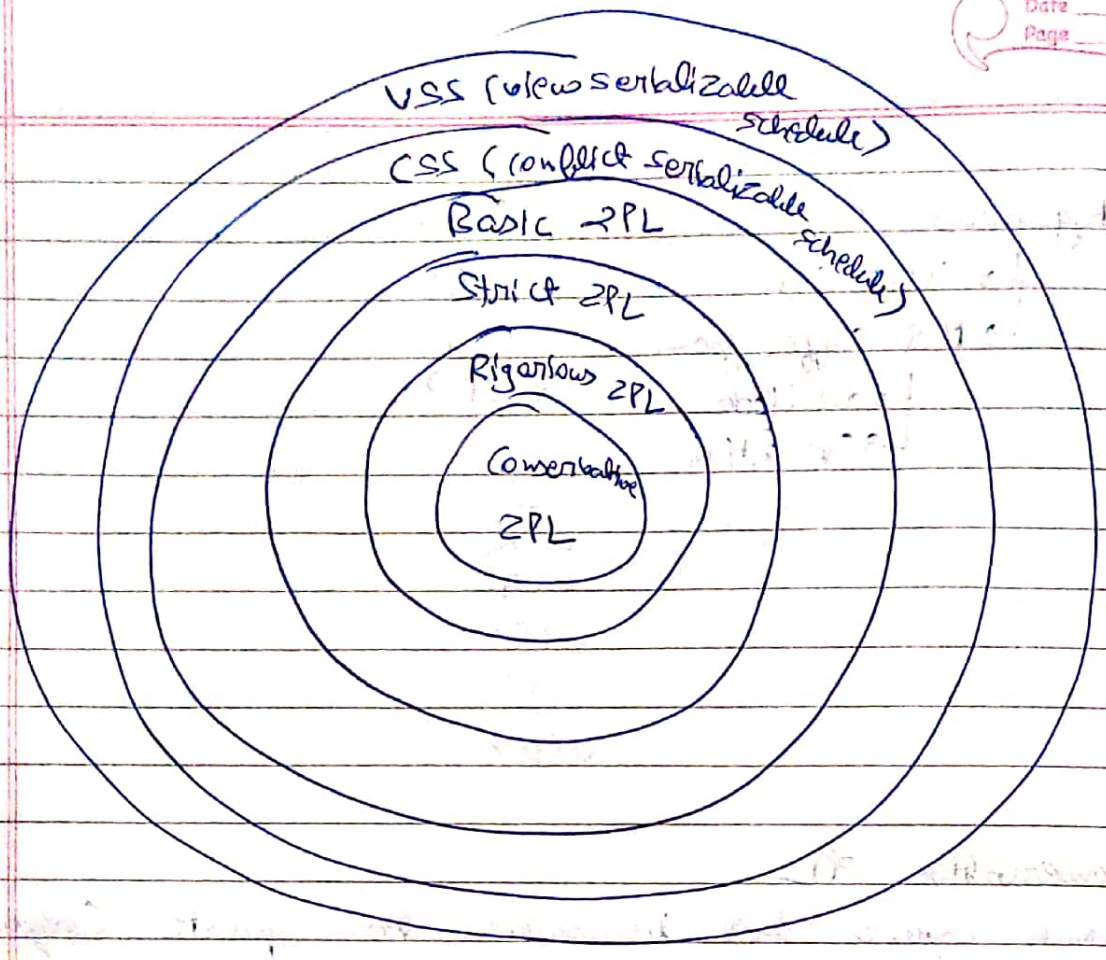
* Conservative 2PL

→ which require that transaction to update (acquire) all the lock before it starts (transaction begin its execution) & release all the locks after it commit.

- ↳ Conservative 2PL
  - ↳ Strict 2PL
  - ↳ Rigorious 2PL
  - ↳ Basic 2PL
  - ↳ no cascading rollback
- ↳ no deadlock
- ↳ But suffer from starvation

→

Scanned by CamScanner



Question

Which of the following schedules are in Strict 2PL?

① Lock-S(A)
R(A)
Lock-X(B)
R(B)
Unlock-S(A)
W(B)
Unlock-X(B)
Commit
~~2PL~~

Basic 2PL ✓

② Lock-S(A)
R(A)
Lock-X(B)
Unlock(A)
R(B)
W(B)
Commit
Unlock-X(B)

Basic 2PL ✓
Strict 2PL ✓

Scanned by CamScanner

③ Lock-S(A)
R(A)
Lock-X(B)
R(A)
Lock-X(B)
R(B)
W(B)
Commit
Unlock-(A)
Unlock (B)

Rigorous ✓
2PL
→ Strict 2PL
→ Basic 2PL

④ Lock-S(A)
Lock-X(B)
R(B)
W(B)
R(A)
Commit
Unlock (A)
Unlock(B)

Conservative ✓
2PL
→ Rigorous 2PL
→ Strict "
→ Basic "

⑤ Lock-S(A)
R(A)
Unlock (A)
Lock-X(B)
R(B)
W(B)
Unlock-X(B)
Commit

~~Basic 2PL ✓~~
Basic ✓
not 2PL

Scanned by CamScanner

o) Time Stamp based protocol

→ Time stamp value: Unique value assigned by the DBMS to every transactions in ascending order.

If time stamp of transaction	T_1	T_2	T_3	T_4	T_5	T_6
	10	30	20	40	60	90

Schedule: $T_1 \rightarrow T_3 \rightarrow T_2 \rightarrow T_4 \rightarrow T_6 \rightarrow T_5$

T_1 → older transaction

T_5 → younger transaction

- In time stamp based protocol ordering among the transactions is determined ~~by~~ in advance based on their timestamps. (i.e the time when they enter into the system for execution)
- Each transaction is given unique fixed time stamp denoted by $TS(T_i)$ {! Timestamp of T_i}
- If any transaction T_j that enters after T_i, the relation b/w their timestamp be $TS(T_i) < TS(T_j)$ which means that the producing schedule must be equivalent to a serial schedule $T_i \rightarrow T_j$ (T_i followed by T_j)

o) Time stamp of data item

→ RTS (read time stamp)

→ WTS (write time stamp)

(i) Read-TS(A) : (RTS(A)): Highest transaction timestamp value that has performed Read(A) operation successfully.

T_1	T_2	T_3
R(A)		
	R(A)	
		R(A)

$30 \rightarrow TS$

~~Initially RTS(A) = 0~~

~~RTS(A) = 10~~

~~RTS(A) = 20 (max~~

Scanned by CamScanner

Initially RTS(A) = 0
RTS(A) = 10
RTS(A) = 20 [max (10, 20)]
RTS(A) = 30 [max (20, 30)]

(ii) <u>write - TS(A)</u> (WTS(A)) : Highest transaction time-stamp value that has performed write (A) operation successfully.

10	20	30	
T_1	T_2	T_3	WTS(A) = 0 (initially)
W(A)			WTS(A) = 10
	W(A)		WTS(A) = 20 max (10, 20)
		W(A)	**WTS(A) = 30** max (20, 30)

o) Time Stamp ~~are~~ ordering protocol
→ Two type of timestamp
↳ Transaction Time Stamp TS(T_i) : fixed
↳ Data Item Time Stamp : RTS(Q), WTS(Q) : variable

→ In time stamp ordering protocol ensure that any conflicting read & write operation (W-R, R-W, W-W) are executed in timestamp order if not such operation is Rejected & transaction will be rolled back. The Rolled back transaction will be restarted with a new timestamp.

⇓ meaning

10	20
T_1	T_2

Schedule
$T_1 \rightarrow T_2$
(then operation ~~must~~ must be T_1 followed by T_2 only)

Schedule
$T_1 \rightarrow T_2$

but operation
T_2
T_1 ↙
roll back

R - W
W - R
W - W (conflict)

(I) T_i : Read(Q) [transaction T_i issue read(Q) operation]

(i) If $TS(T_i) < WTS(Q)$: Read operation reject & T_i rollback

(ii) If $TS(T_i) \geq WTS(Q)$: Read operation is allowed & set read-TS(Q) = max [RTS(Q), TS(T_i)]

*)

10	20	30
T_1	T_2	T_3
	R(Q)	
		W(Q)

let transaction T_2 wants to perform Read on data item (Q)

~~TS(T_i)~~ TS(T_2) = 20
WTS(Q) = 30
TS(T_2) < WTS(Q)
(so reading some old value)
(does not make any sense)

eg. reading old gold syllabus which has to be updated later

X

(Rollback)

*)

10	20	30	40
T_1	T_2	T_3	T_4
		W(Q)	
			R(Q)

T_4 wants to perform read
TS(T_4) = 40
WTS(Q) = 30
TS(T_4) > WTS(Q)
(operation allowed)

(1) If TS(T_i) < WTS(Q): then T_i need to read a value of Q that was already overwritten by later transaction. Hence the read operation is rejected & T_i rollback.

(2) If TS(T_i) ≥ WTS(Q) ; then the read operation is allowed & read time stamp is set to max of RTS(Q) & TS(T_i)

Scanned by CamScanner

$$\boxed{\text{Set } RTS(Q) = \max[RTS(Q), TS(T_i)]}$$

(II) T_i: write(Q) [transaction T_i issue write(Q) operation]

(i) If $TS(T_i) < RTS(Q)$: write operation reject & T_i roll-back

(ii) If $TS(T_i) < WTS(Q)$: write operation reject & "

(iii) Otherwise execute write(Q) operation

$$\boxed{\text{Set } WTS(Q) = TS(T_i)}$$

10	20	30
T_1	T_2	T_3
	w(Q)	
		R(Q)

$TS(T_2) = 20$

$RTS(Q) = 30$

$TS(T_2) < RTS(Q)$

✗

10	20	30
T_1	T_2	T_3
	w(Q)	
		w(Q)

$TS(T_2) < WTS(Q)$

✗

(i) If $TS(T_i) < RTS(Q)$: then the value of data item Q that T_i is producing was needed previously and system assumed that the value would never be produced. Hence write operation is rejected & T_i rollback.

(ii) If $TS(T_i) < WTS(Q)$: then T_i is attempting to write a obsolete value of Q, hence write operation is rejected & T_i rollback & start with new stamp.

Scanned by CamScanner

TSP: time stamp based protocol

(iii) otherwise write oper allowed

$\boxed{\text{Set } WTS(Q) = TS(T_i)}$

Question

①

T_1	T_2
$r(A)$	
$r(B)$	
	$w\,r(C)$
	$r(D)$

TSP ✓

②

10 T_1	20 T_2
	$r(A)$
$r(B)$	
$w(A)$	

$WTS(A) = 10$ $\quad T_1 \rightarrow T_2$

$TS(T_2) = 20$

$TS(T_2) > WTS(A)$

$TS(T_1) = 10$

$RTS(A) = 20$ $\quad TS(T_1) < RTS(A)$

So can't write

not TSP

③

10 T_1	20 T_2
$r(B)$	
$w(D)$	
	$r(A)$
$r(A)$	

$T_1 \rightarrow T_2$

$TS(T_1) = 10$ $\quad$ not TSP

$RTS(A) = 20$

(write not allowed)

$TS(T_2) = 20$

$WTS = 10$

$TS(T_2) > WTS(A)$ ✓

TSP (read allowed).

Scanned by CamScanner

(4)

10	20
T_1	T_2
R(A)	
	w(A)
w(A)	

roll back

reject

$TS(T_2) = 20$, $RTS(A) = 10$

$20 > 10$ so write performed

$TS(T_1) = 10$, $WTS(A) = 20$

$TS(T_1) < WTS(A)$

writing not allowed

not TSP

(or)

10	20
T_1	T_2
R(A)	
	w(A)
w(A)	

NB

reject

$T_2 \rightarrow T_1$ ✗

$T_1 \rightarrow T_2$

So all conflict oper should be from $T_1 \rightarrow T_2$

not TSP

(5)

10	20
T_1	T_2
R(A)	
	R(A)
w(A)	

reject

$T_1 \rightarrow T_2$

not TSP

$TS(T_1) = 10 < RTS(A) = 20$

reject

(6)

10	20
T_1	T_2
R(B)	
	w(B)
R(A)	
	w(A)

$T_1 \rightarrow T_2$

TSP ✓

(7)

10	20
T_1	T_2
R(A)	
	w(A)
w(A)	

$T_1 \rightarrow T_2$

roll-back

$T_2 \rightarrow T_1$ ✗

not in to TSP ✗

Obsolete write

o) Thomas write Rule:

(view Serializability)

- Which require to ignore all the obsolete write operation
- Thomas write rule is used to achieve the view Serializability.

(i) T_i - Read(Q)
Same as TSP

(ii) T_i - write(Q)
(i) $TS(T_i) < RTS(Q)$: Reject & rollback (Same as TSP)

** (ii) $TS(T_i) < WTS(Q)$: write operation is ignored & no rollback

otherwise Same as TSP.

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

$T_1 \rightarrow T_2$

Ignore & no roll back

not allowed under TSP but allowed under thomas write rule

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

allowed under thomas write rule.

Note.

- The stamp ensure Serializable but time stamp ordering protocol does not ensure recoverability & cascadeless schedule.

- But in TSP there is no deadlock

Scanned by CamScanner

* Finding the no. of conflict serializable schedule

Gate 2017

$T_1 : r_1(x)\ w_1(x)\ r_1(y)\ w_1(y)$
$T_2 : r_2(y)\ w_2(y)\ r_2(z)\ w_2(z)$

find total # ~~correct~~ conflict serializable schedule?

first check for ① $T_1 \rightarrow T_2$ then for ② $T_2 \rightarrow T_1$

① $T_1 \rightarrow T_2$

Step 1) first write down all the operations of the following (later) (T_2) (here) transactions.

Step 2) then start from last operation of first transaction (T_1) then check the possibility

$T_1 \rightarrow T_2$ ✓ ✗ $T_2 \rightarrow T_1$ operation (wrong).

last oper of T_1 ($w_1(y)$) $r_1(y)$ $w_1(x)$ $r_1(x)$ $w_1(y)$ ✓ $w(y)$ ✗ $R_2(w)$ ↑ $w_2(y)$ ↑ ✗ $r_2(z)$ ↑ ✗ $w_2(z)$ ↑ ✗

now check where we can insert $w_2(y)$ so that order must be T_1 followed by T_2 (only $T_1 \rightarrow T_2$: 1 way)

---

② $T_2 \rightarrow T_1$

$\boxed{r_2(z)\ w_2(z) \Rightarrow \text{placed anywhere}}$

$\rightarrow T_2$

$\rightarrow T_1$

$r_1(x)$ ↑ $w_1(x)$ ↑ $r_1(y)$ ↑ $w_1(y)$ ↑

$w_2(y)$ ✓ Case 3 | $w_2(y)$ ✓ Case 2 | $w_2(y)$ ✓ Case 1 | $w_2(y)$ ✗ $T_1 \rightarrow T_2$ ✗ | ✗

Scanned by CamScanner

Case 1

$r_1(x)$ $w_1(x)$ $w_2(y)$ $r_1(y)$ $w_1(y)$

fixed

$r_2(y)$ $r_2(y)$ $r_2(y)$ $r_2(y)$ $r_2(y)$ ✓ ✓ ✓ ✗ ✗ ✗

3 way of $R_2(y)$ $\times$ $(3C_1 + 3C_2)$

$= 3 \times (3+3)$

$= 18$ way.

how?

$- | - | -$ (3 place)

$R_2(z)$ $w_2(z)$

either we have to choose one place out of three & place them together $(3C_1)$ (or)

2 place out of three & place them separately $(3C_2)$

Case 2)

$r_2(y)$ $r_1(x)$ $w_2(y)$ $w_1(x)$ $r_1(y)$ $w_1(y)$

fixed

$r_2(y)$ ✓ $r_2(y)$ ✓ ✗ ✗ ✗ ✗

2 way of $r_2(y)$ $\times$ $(4C_1 + 4C_2)$

$= 2 \times (4+6) = 20$ way.

Case 3)

$w_2(y)$ $r_1(x)$ $w_1(x)$ $r_1(y)$ $w_1(y)$

fixed

$r_2(y)$ ✓ $r_2(y)$ ✗

1 way $\times$ $(5C_1 + 5C_2) = 1(5+10)$

$= 15$ ways

Scanned by CamScanner

total ways: $T_1 \to T_2$: 1

$T_2 \to T_1$: $(18 + 20 + 15)$

$= 1 + 53$

$= 54$ ways

Question

T_1: $r_1(a)$ $r_1(b)$ $w_1(b)$

T_2: $r_2(a)$ $r_2(b)$ $w_2(b)$

(1) $T_1 \to T_2$

$w_1(b)$ ✓ $r_2(a)$ (case 2) $w_1(b)$ ✓ $r_2(b)$ (case 1) $w_1(b)$ ✗ $w_2(b)$ ✗

(case 1)

$r_1(b)$ ✓ $r_2(a)$ $r_1(b)$ ✓ $w_1(b)$ ✗ $r_2(b)$ ✗ $w_2(b)$ ✗

$r_1(a)$ $r_1(b)$ or $r_1(a)$ $r_1(b)$

— $r_2(a)$ — $w_1(b)$

$r_1(a)$ or $r_1(b)$

$^2C_1 + {}^2C_2$

$= 2 + 1 = 3$ way

Case 2

$r_1(a)$ $r_1(b)$ ✓ $w_1(b)$ $r_1(b)$ ✗ $r_2(a)$ ✗ $r_2(b)$ ✗ $w_2(b)$ ✗

Only 1 way

total $T_1 \to T_2$: 4 way

(2) $T_2 \rightarrow (T_1)$

$r_1(a)$ $r_1(b)$ $w_1(b)$

$w_2(b)$ ✓ case 2 $\quad$ $w_2(b)$ ✓ case 1 $\quad$ $w_2(b)$ ✗ $\quad$ ✗

case 1)

$r_1(a)$ $\underline{w_2(b)}$ $r_1(b)$ $w_1(b)$

$r_2(b)$ ✓ $\quad$ $r_2(a)$ ✓ $\quad$ fixed ✗ $\quad$ ✗ $\quad$ ✗

$2C_1 + 2C_2 = 3$ way

case 2) $w_2(b)$ $r_1(a)$ $r_1(b)$ $w_1(b)$

$r_2(b)$ ✓ $\quad$ $r_2(b)$ ✗

only 1 way

total: $T_2 \rightarrow (T_1)$: 4 way

Total conflict serializable serial $= 8$ way

nonconflict serializable $= 8$ way $- 2$ way $= 6$

Question

T_1: $r_1(a)$ $r_1(b)$ $w_1(b)$

T_2: $r_2(b)$ $r_2(a)$ $w_2(b)$

(1) $T_1 \rightarrow (T_2)$

$r_1(a)$ $r_1(b)$ $w_1(b)$ ✓ $\quad$ $r_2(b)$ $w_1(b)$ ✗ $\quad$ $r_2(a)$ ✗ $\quad$ $w_2(b)$ ✗

only one way

Scanned by CamScanner

(2) $T_2 \rightarrow (T_1)$

$w_2(b)$ ✓ $r_1(a)$ $w_2(b)$ ✓ $r_1(b)$ ✗ $w_1(b)$ ✗

(case 2) (case 1)

(case 1)

$r_2(a)$ ✓ $r_1(a)$ $w_2(b)$ $r_1(b)$ $w_1(b)$

$r_2(a)$ ✓ ✗ ✗ ✗

$2C_1 + 2C_2 = 3$ way

(case 2)

$r_2(b)$ $r_2(a)$ ✓ $w_2(b)$ $r_1(a)$ $r_1(b)$ $w_1(b)$

only 1 way

total conflict seri $= 1 + 3 + 1 = 5$

total non serial $= 9 - 2 = 3$

conflict serial

Question

T_1 : $r_1(a)$ $w_1(a)$ $r_1(b)$ $w_1(b)$

T_2 : $r_2(b)$ $w_2(b)$ $r_2(a)$ $w_2(a)$

(1) $T_1 \rightarrow (T_2)$

$w_1(b)$ ✓ $r_2(b)$ $w_2(b)$ $r_2(a)$ $w_2(a)$

$w_1(b)$ ✗ ✗ ✗ ✗

Only 1 way

(2) $T_2 \rightarrow (T_1)$

$w_2(a)$ ✓ $r_1(a)$ $w_2(a)$ ✗ $w_1(a)$ $r_1(b)$ $w_1(b)$

Only 1 way.

Scanned by CamScanner

Total Conflict Serial = 2

non Serial " " = 2 - 2 = 0

Question T_1 : $r_1(a)$ $w_1(a)$ $r_1(b)$ $w_1(b)$

T_2 : $r_2(a)$ $w_2(a)$ $r_2(b)$ $w_2(b)$

(1) $T_1 \rightarrow T_2$

$r_2(a)$ $w_2(a)$ $r_2(b)$ $w_2(b)$

$w_1(b)$ ✓ case 3 $w_1(b)$ ✓ case 2 $w_1(b)$ ✓ case 1 $w_1(b)$ ✗ ✗

case 1)

$r_2(a)$ $w_2(a)$ $\underline{w_1(b)}$ $r_2(b)$ $w_2(b)$

$r_1(b)$ ✓ case 3 $r_1(b)$ ✓ case 2 $r_1(b)$ ✓ case 1

~~3 way × [2C1 + 2C2] = 3×3 = 9~~

case 1.1)

$r_2(a)$ $w_2(a)$ $\underline{r_1(b)}$ $\underline{w_1(b)}$ $r_2(b)$ $w_2(b)$

$w_1(a)$ ✓ $w_1(a)$ ✗ ✗ 1 way

case 1.2)

$r_2(a)$ $\underline{r_1(b)}$ $w_2(a)$ $\underline{w_1(b)}$ $r_2(b)$ $w_2(b)$

$w_1(a)$ ✓ $w_1(a)$ ✗ 1 way

= 3 way

case 1.3)

$r_1(b)$ $r_2(a)$ - - -

1 way

(case 2) $r_2(a)$ $\underline{w_1(b)}$ $w_2(a)$ $r_2(b)$ $w_2(b)$

$r_1(b)$ ✓ $r_1(b)$ ✓

1 + 1 = 2 way

Scanned by CamScanner

(case 3) $w_1(b)$ $r_2(a)$ $w_2(a)$ $r_2(b)$ $w_2(b)$

1 way

$T_1 \rightarrow T_2$: 6 way

(2) $T_2 \rightarrow T_1$

$r_1(a)$ $w_1(a)$ $r_1(b)$ $w_1(b)$

$w_2(b)$ ✓ (3) $w_2(b)$ ✓ (2) $w_2(b)$ ✓ (1) ✗ ✗

(case 1) $r_1(a)$ $w_1(a)$ $\underline{w_2(b)}$ $r_1(b)$ $w_1(b)$

$r_2(b)$ ✓ $r_2(b)$ ✓ $r_2(b)$ ✓

3 ways

(case 2) $r_1(a)$ $\underline{w_2(b)}$ $w_1(a)$ $r_1(b)$ $w_1(b)$

$r_2(b)$ ✓ $r_2(b)$ ✓

2 ways

(case 3) $w_2(b)$ $r_1(a)$ $w_1(a)$ $r_1(b)$ $w_1(b)$

1 way

$T_2 \rightarrow T_1$: 6 ways

total conflict serial sch = 12

Scanned by CamScanner

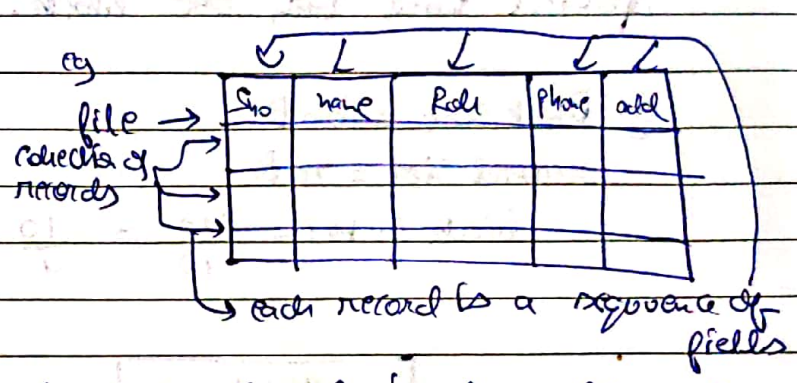
File organization & Indexing

database
db file
records
fields

Blocks

- database is a collection of file
- each file is a collection of records
- each record is a sequence of field

eg

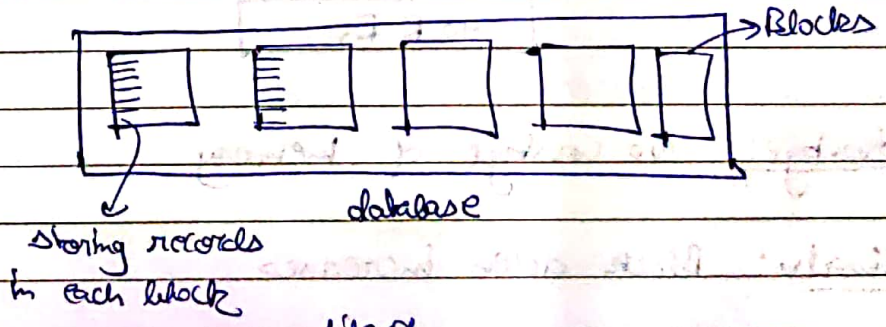


→ Practically records are stored in the form of blocks in database.

→ Db is divided into no. of blocks

→ Each block is divided into records.

→ Records can be stored in a block.



→ Strategies for storing file or records into blocks

- → Spanned org | Strategy
- → unspanned org | strategy

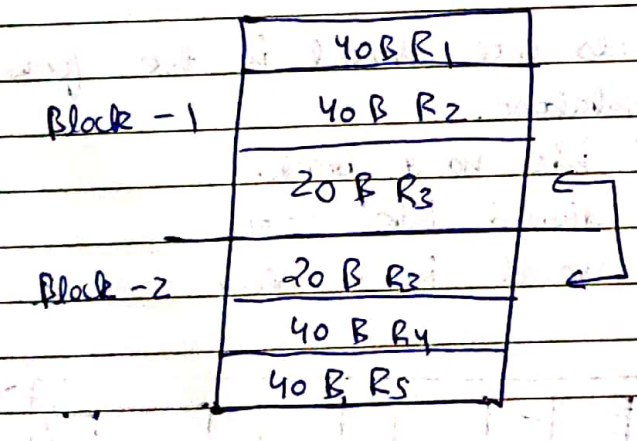
(Block size = 80B, record size = 40B / 2 records per block) → blocking factor

o) Blocking factor : Avg # records per blocks

eg) If blocking factor is 4 i.e 4 records can be stored per block.

① Spanned strategy / org : Partial part of records can be stored in a block. A record can be spanned more than one block.

eg. Block size = 100 B
record size = 40 B
block factor = $\frac{100}{40} = \frac{10}{4} = 2.5$



Advantage: No wastage of memory

disadv: Block access increases

o) Suitable for variable length / size records.

② Unspanned org / Strategy : No record can be stored more than one block.
→ Entire record should be always belongs to a single block.

Scanned by CamScanner

Eg)

Block size = 100 B

record size = 40 B

Blocking factor = $\frac{100B}{40B}$ = $\boxed{\text{Blocking factor} = \frac{\text{Block size}}{\text{Record size}}}$

floor value

block factor = $\left\lfloor \frac{100}{40} \right\rfloor$

= 2

	40 B R_1
B_1	40 B R_2
	/// 20 B Free
	40 B R_3
B_2	40 B R_4
	/// 20 B Free

- disadv: wastage of memory
- adv: block access is reduced
- Suitable for fixed length / size records.

Note! Default org is unspanned org bcoz in unspanned org I/O cost is less.

↳ input - output cost
The no. of block transferred from secondary memory to main memory in order to access same record.

- Organization of records in file
  - ↳ ordered file org
  - ↳ unordered file org.

Search key: attribute used to access the data from ~~the~~ database

→ ordered i.e ascending or descending order
⇓
Binary Search

① **Ordered file org** : All file of records are ordered based on some search key value.

Searching: Binary Search

→ To access a record the avg no of block access $\boxed{= \log_2 B}$ Block
B : data block

Adv: Searching is efficient

disadv: insertion is expensive, due to reorganization of the ~~file~~ entire file.

② **unordered file orga:**
→ All file of records are inserted at wherever the place is available, usually at the end of the file.

Searching: Linear search.

→ To access a record, the avg no. of block access $= \boxed{\frac{B}{2} \text{ block}}$

Adv: Insertion is efficient

disadv: Searching is inefficient compared to ordered file organization.

→ **Index**
→ Indexes are used to improve the searching efficiency.

Scanned by CamScanner

(always consider Book eg to understand)

To

- reduce the I/O cost, indexing is used.
- Index file is also divided into the blocks.
- Index file block size is same as database file block size.

i.e $\boxed{\text{Block size of Index file} = \text{Block size of DB file}}$

- Index is a record consist of two fields

Search Key	Block pointer

Search Key → for data block access

Block pointer → for index block access; → pointer to a block where key is available; → eg. page no

- $\boxed{\text{one Index record size} = \text{size of Search Key} + \text{size of block pointer}}$

- Index is an ordered file

  Searching: Binary Search

- to access a record using Index, avg no. of block access $= \boxed{\log_2 B_i + 1}$

  $\log_2 B_i$ → Index block access

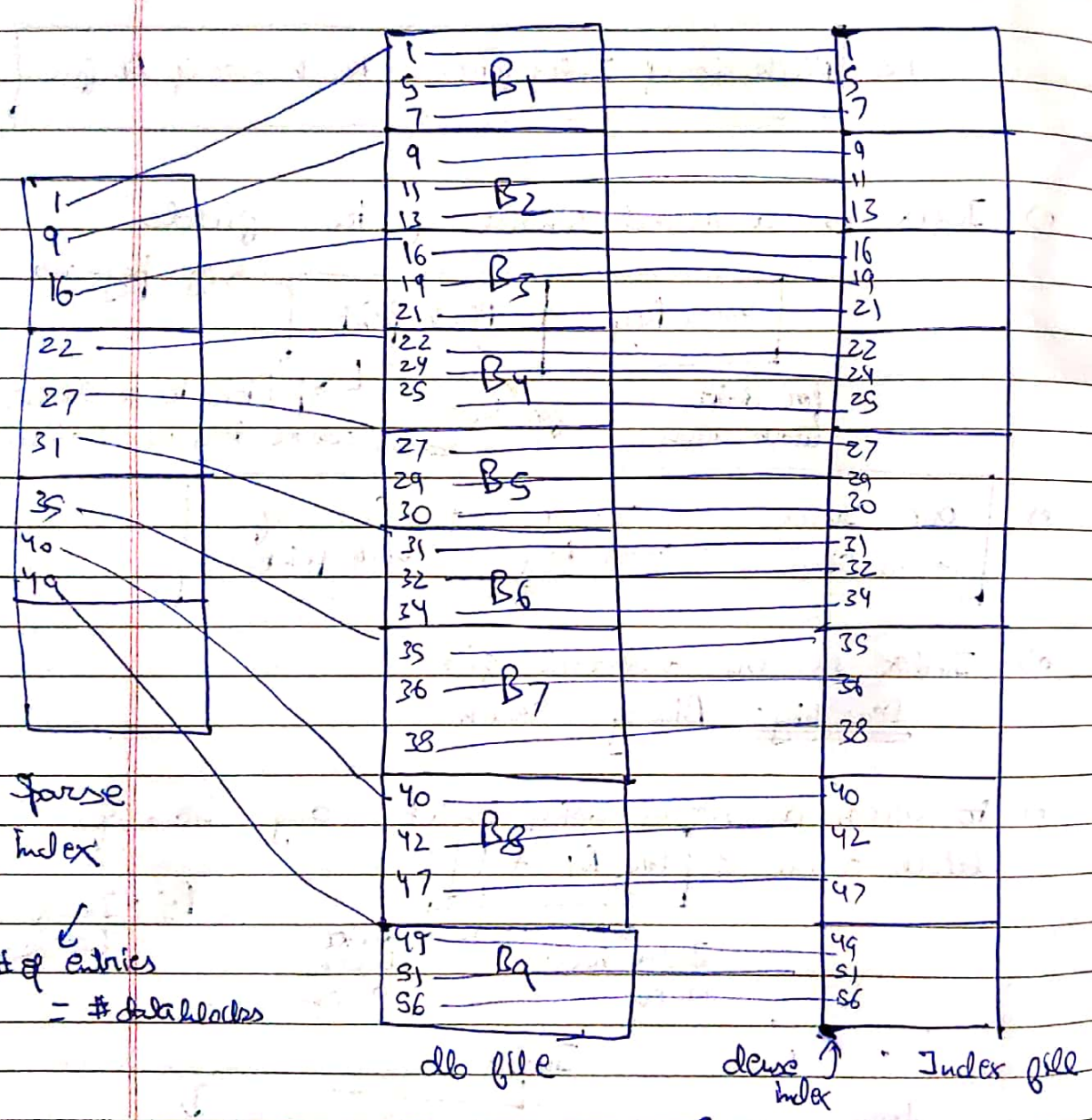
  $+1$ → data block access

  B_i : Index blocks

Index
- Single level Index
  - Primary Index (primary key + ordered)
  - Clustered Index (non key + unordered)
  - Secondary Index (non key + unordered)
    - Candidate key
- Multilevel Index
  - B Tree
  - B+ Tree

Scanned by CamScanner

o) Categories of Index
- dense Index
- Sparse Index



(for every record there exist corresponding entries in index file)

entries in Index file (dense) = # records in file

Scanned by CamScanner

o) **Dense Index:** If an index entry is created for every search key value, that index is called dense index.

o) for every database record there exist corresponding entry in Index file.

no of Index entries = no of records

o) **Sparse Index:** If an index entry is created for some search key value, it is called sparse index.

→ for a set of records there exist one entry in Index file.

no of Index entries = no of blocks

① Primary Index (PK + ordered)

(i) Search key used in the Index file should be used to physically ordered the db records. [i.e ordered file]

(ii) Search key should be candidate key (primary / alternate key)

→ If indexing satisfy both these condition then it is said to be primary index.

→ Primary index can be sparse or dense.

(or)

→ A primary Index is an ordered file whose records are of fixed length with two fields. The first field is same as primary key of data file & second field is a pointer point to a block where the key is available.

Scanned by CamScanner

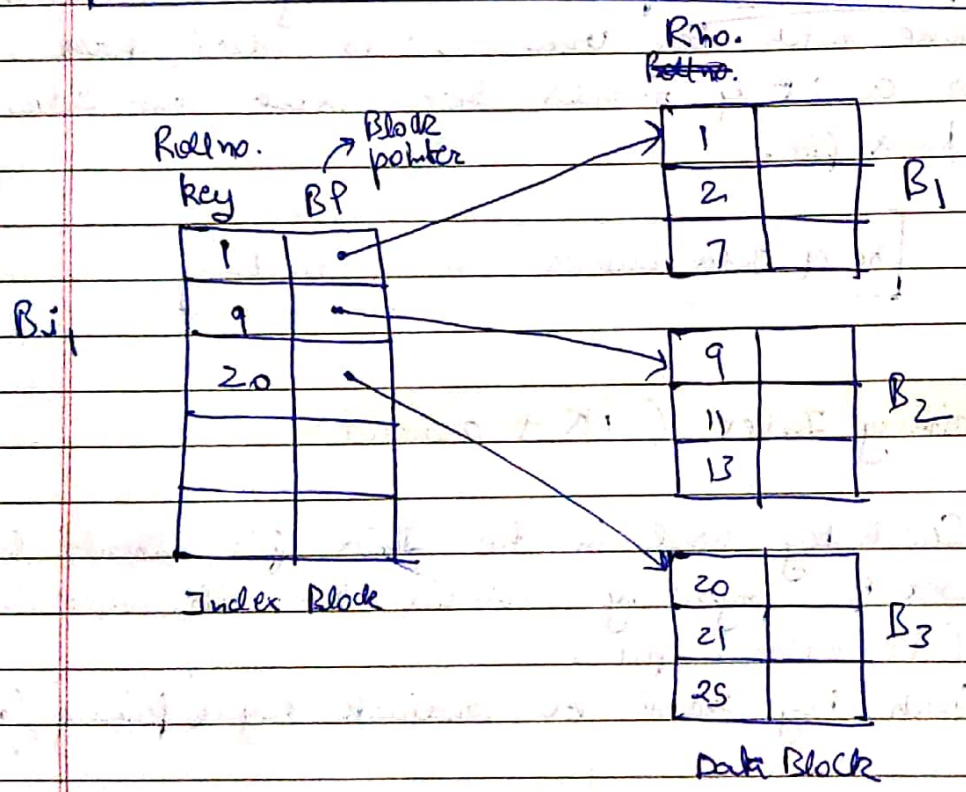
o) Index file is always ordered file

o) Index entry is created for first record of each block called '<u>Anchor Record</u>'

o) This type of Index generally Sparse.

$\boxed{\text{no of Index entries} = \text{no of data blocks}}$

o) $\boxed{\text{To access a record, Avg \# Block Access} = \log_2 B_i + 1}$



1, 9, 20 are anchor record here

Question

Suppose we have a ordered file of 30,000 records stored on a disk with a Block size 1024 B. File record of fixed length and unspanned of size 100 B and Suppose we have have created a primary Index on the key field of the file of size 9 B & block pointer of size 6B. then find the avg no. of Block access to search for records using

(1) without Indexes
(2) with Index.

Given → # records = 30,000
Block size = 1024 B
Record size = 100 B

Blocking factor = $\left\lfloor \frac{1024}{100} \right\rfloor$ = 10 Records per block

without Index

there are total 30,000 records
So. # of Blocks = 3000 data blocks → $\left\lceil \frac{30000}{10} \right\rceil$

To access a record avg no. of Block access (ordered file) = $\log_2 B$
$= \log_2 3000$

= 12 ~~data~~ block access

with Index

~~no. of datablock = 3000~~
~~# index entries = 3000~~
index record size = 9B + 6B = 15 B
~~Index Block size = 3000×15 = 45000 B~~

Block size = 1024 B

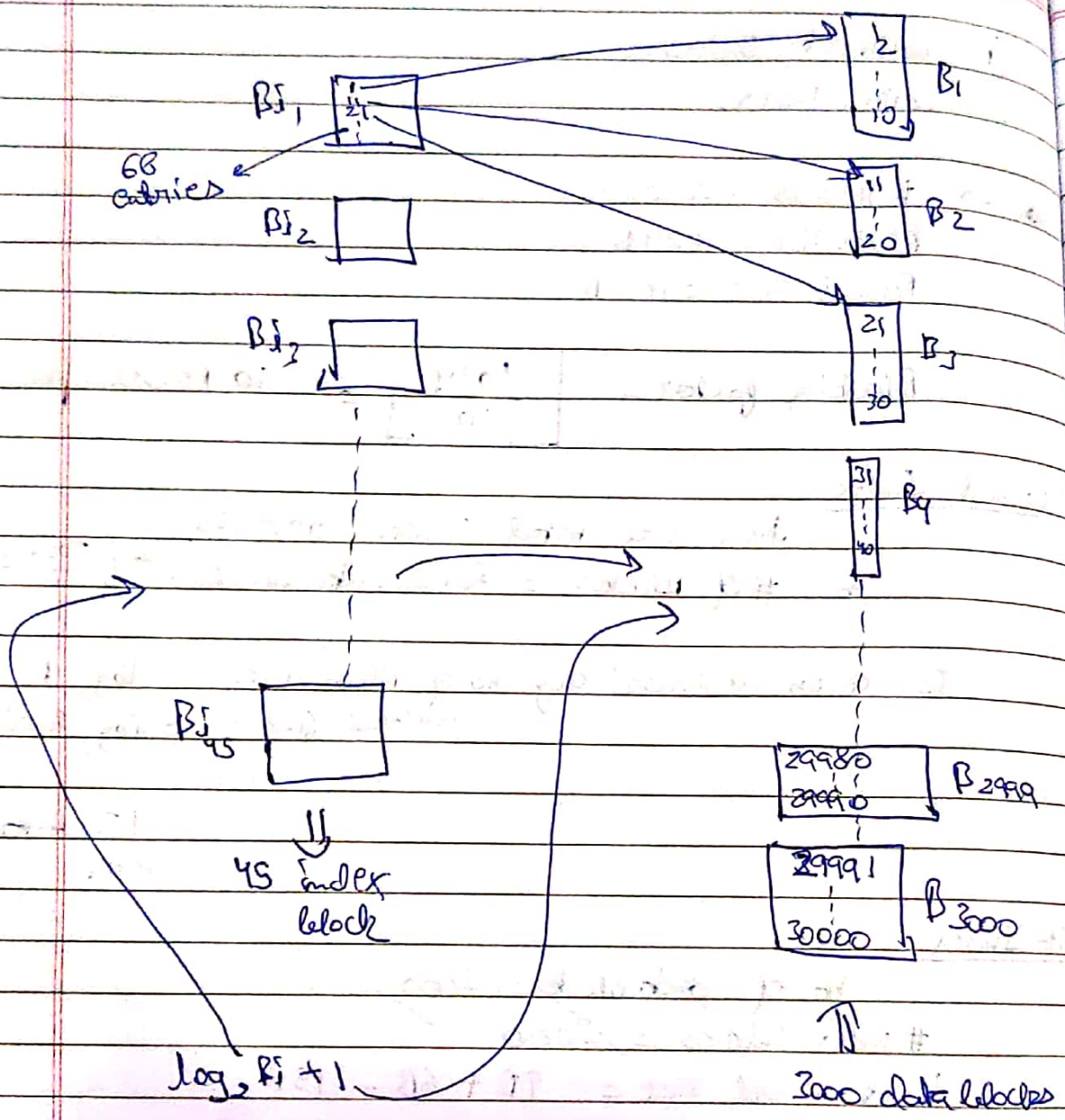
block factor of Index file = $\left\lfloor \frac{1024 B}{15 B} \right\rfloor$ = 68 Index entry per block.

Sparse # entries = no of datablock = 3000

Index block = $\left\lceil \frac{3000}{68} \right\rceil$ = 45 index block

To access a record using Index avg no of block access = $\log_2 45 + 1$
$= 6 + 1 = 7$

Scanned by CamScanner



(2) Clustered Index (non-key + ordered)

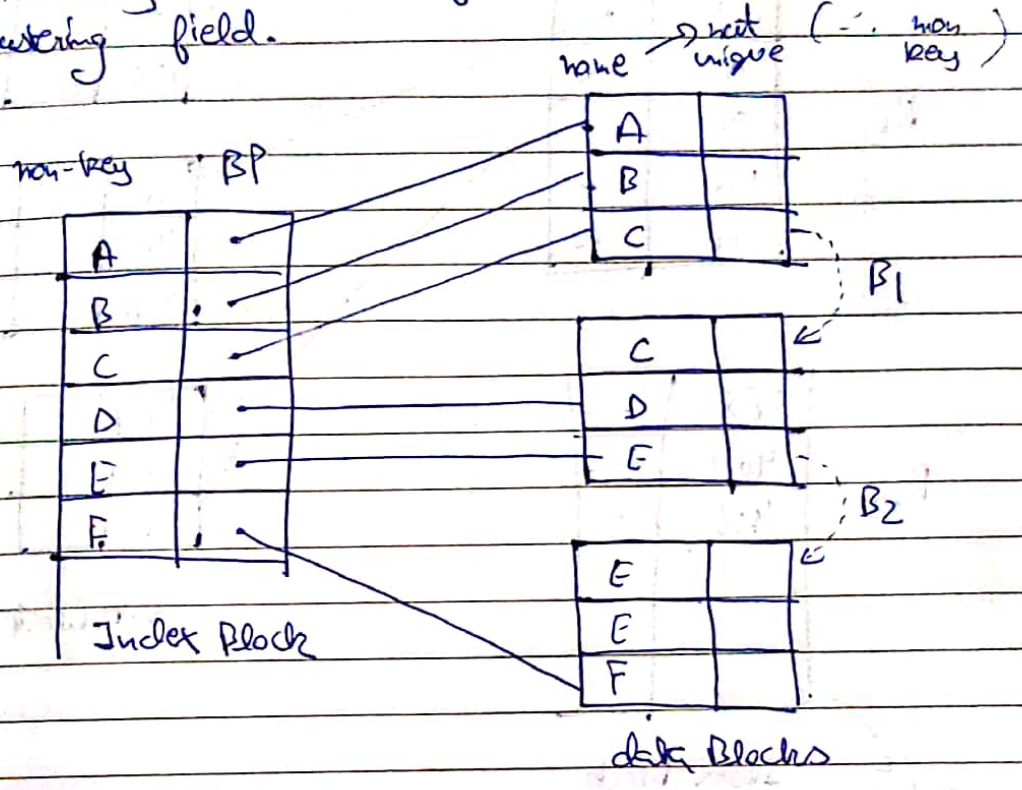
→ Clustered Index is an ordered file with two field, the first field is same as clustering field called as non-key & ~~Secondary~~ Second field is a block pointer.

→ Each cluster has similar group of data.

→ Clustered Index is created on data file whose file records are physically ordered on a non-key field which does not have a distinct value for each record that field is called clustering

field.

v) Index entry is created for each distinct value of clustering field.



③ Secondary Index (non-key / C.K + unordered file)

→ Secondary way to access the data even if either primary or clustering Index already exist

(or)

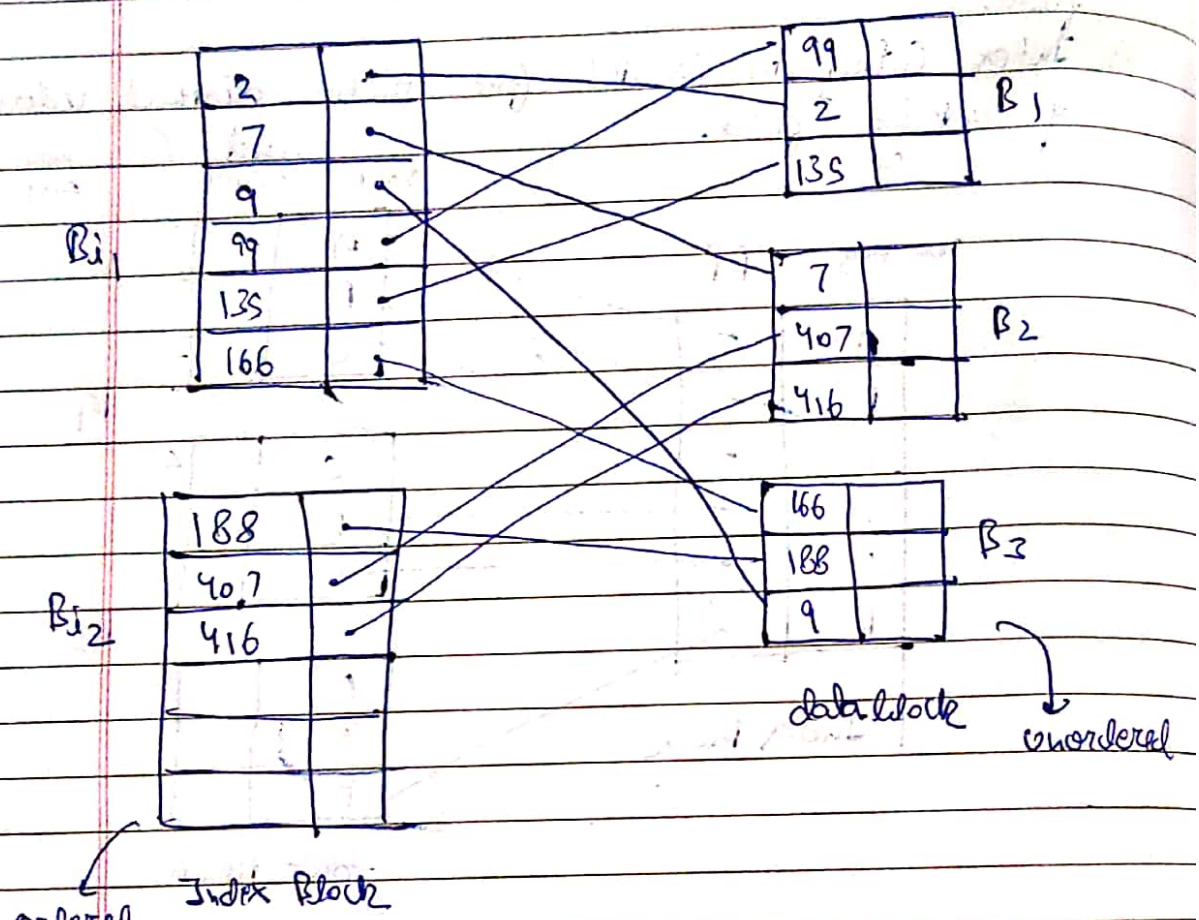
→ Secondary Index provide a secondary way of accessing a file for which some Index already exist.

→ Secondary Index may be non key or candidate key

→ ~~type~~ type of secondary index is dense

$$\boxed{\text{no. of Index entry} = \text{no. of records}}$$

→



(∵ Index is always ordered file)

Question

Consider a Secondary Index is built on the key field of file of question 1 (in primary index). find the avg no. of blocks access a record using (i) Index
(ii) without Index

a) # records = 30,000
Block Size = 1024 B
record Size = 100 B

~~no. of~~ Blocking factor = $\left\lfloor \frac{1024}{100} \right\rfloor$ = 10 records per block

$$\#\text{blocks} = \left\lceil \frac{30{,}000}{10} \right\rceil = 3000$$

avg no. of block to access a record (ordered file) $= \frac{B}{2} = \frac{3000}{2}$

$= 1500$ blocks access

A <u>with Indexing:</u> (always ordered file)

→ Size of record in index file $= 9 + 6 = 15\,B$

records in index file = # entries $= 30{,}000$

block size $= 1024\,B$

Blocking size $= \left\lfloor \frac{1024}{15} \right\rfloor = 68$ Index record per block.

$$\#\text{Index blocks} = \left\lceil \frac{30000}{68} \right\rceil = \lceil 441.17 \rceil = 442 \text{ blocks}$$

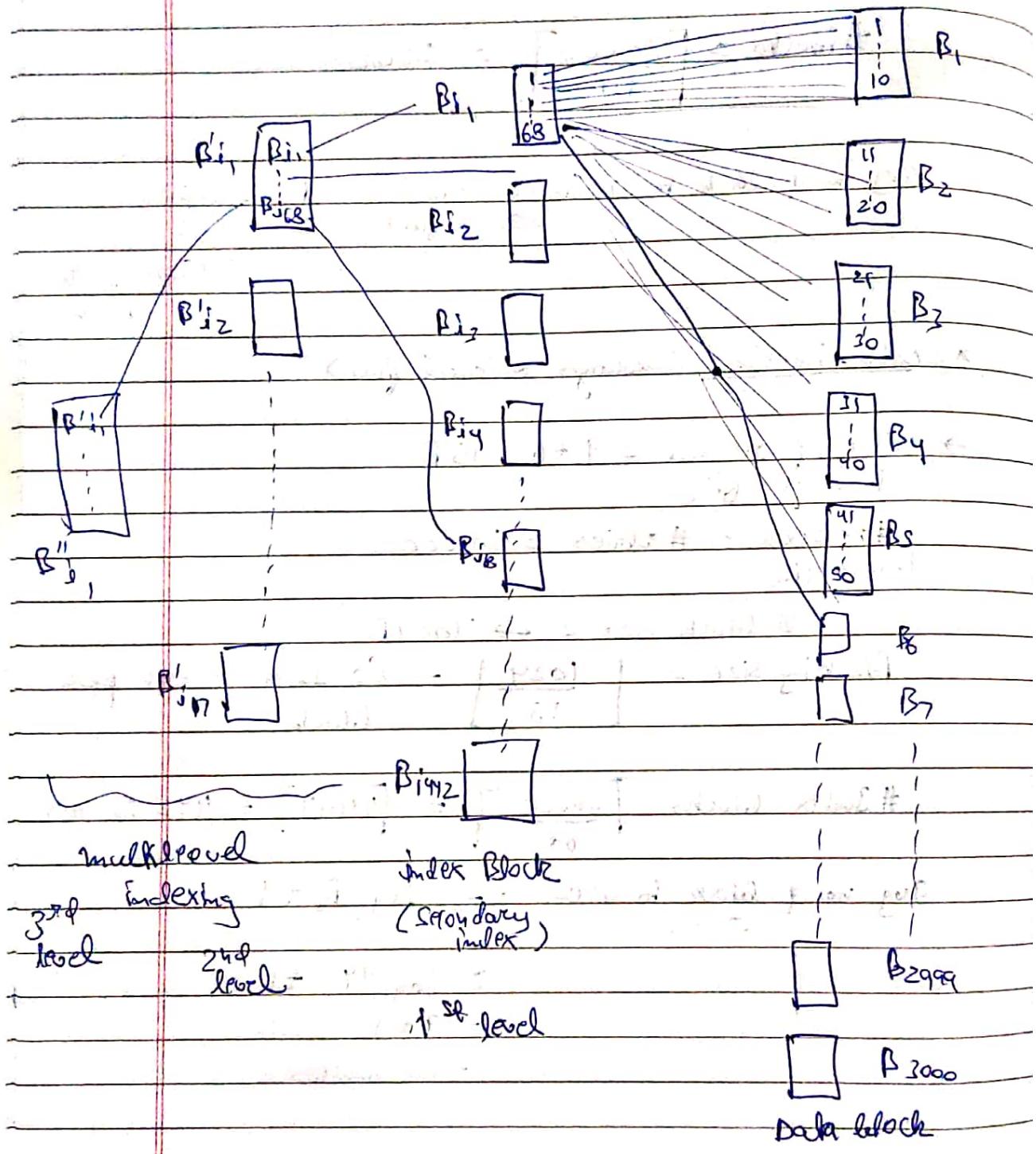
avg no of block to access a record $= \log_2 B_i + 1$

$= \log_2 442 + 1$

$= 9 + 1$

$= 10$ blocks

diag explanation →



- here one index block stores future of only 68 students whereas in case of primary indexing, one block stores future of 68x10 = 680 students
- Till now it was first ~~level~~ level indexing
- Here as we can see there are 142 index blocks which is very large so we apply multilevel indexing on secondary index.

Scanned by CamScanner

classmate
Date
Page

⇒ So applying multilevel indexing ~~State~~ till all index entries fit into one block.

* **multilevel Paging (Static)**

→ As a single level Index is an ordered file we can create a primary index to the Index itself.

→ In this case original file is called 1st level Index & Index to Index is called 2nd level index & so on.

→ We can repeat the above process until all Index entries fit in one disk block.

Question

find the Avg no of block Access required to search for a record if multilevel Index is created on a data file of Question 2 (~~Secon~~ Secondary index)

Block factor of Index file = 68

1st level

Total # Index blocks = 442

IInd level

no. of Index entries (records) = 442 (no of 1st level Block)

Total # Index blocks = $\left\lceil \frac{442}{68} \right\rceil = 7$

IIIrd level

no of Index entries = 7 (no of 1st level block)

Total # Index blocks = $\left\lceil \frac{7}{68} \right\rceil = 1$ (Stop here)

Avg no of Block access = 1 + 1 + 1 + 1 = 4

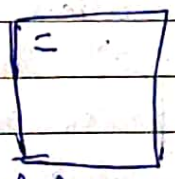
3rd level, 2nd level, 1st level, data block

Scanned by CamScanner

If there are n level in multilevel Index then no of block access to search for a record = n+1

(at each level one & one data block in last Index Block)

~~last~~



If only 2 entries but rest space is wasted (demerit of Static multilevel)

~~last~~ nth level

Gate

Clustering index is defined on the fields which are of type

a) non key & ordering
b) non-key & non ordering
c) Key & ordering
d) key & non-ordering

Gate

Consider a file of 16384 records. Each record is 32 B long & its key field is of size 6B. The file is ordered on a non-key field, & the file organization is unspanned. The file is stored in a file sys with block size 1024 B, & size of block pointer is 10 B. If the secondary index is built on the key field of file, & a multilevel index scheme is used to store secondary index, the no of first level & second-level block in is ?

a) 8 & 0
b) 128 & 6
c) 256 & 4
d) 512 & 5

Scanned by CamScanner

records = 16384

index record size = 6B + 10B
= 16B

Block size = 1024B

Blocking factor of index records = $\left\lfloor \frac{1024}{16} \right\rfloor$ = 64 records per block

index entries = 16384

index blocks = $\left\lceil \frac{16384}{64} \right\rceil$ = 256 blocks

first level → 256 blocks
second level → # entries = 256
blocks = $\left\lceil \frac{256}{64} \right\rceil$ = 4 blocks

0) Dynamic multilevel indexing

B Tree B+ Tree

① B Tree (balanced tree)

→ Special tree in which we have an order

ORDER : P	order : 5
max element can insert : P - 1	max element : 4
min " " " : $\lceil \frac{P}{2} \rceil - 1$	min element : 2

Order : 4
max element = 3
min = 1

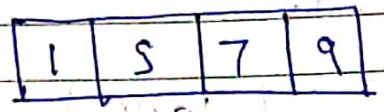
Scanned by CamScanner

c) data ~~1, 5, 7, 9, 11, 13~~ 1, 7, 5, 9, 11, 13

order : 5

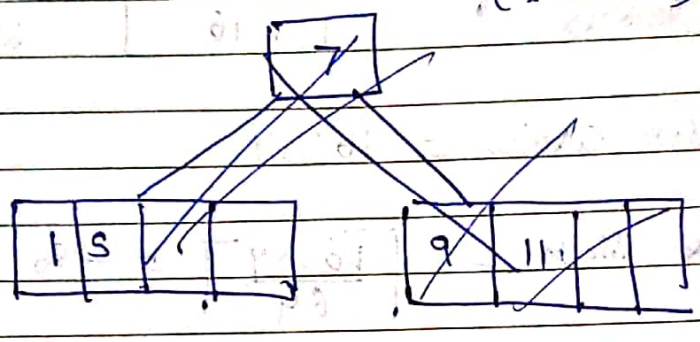
max element : 4 , min ele = 2

(all elements are inserted in ascending order)



← 11

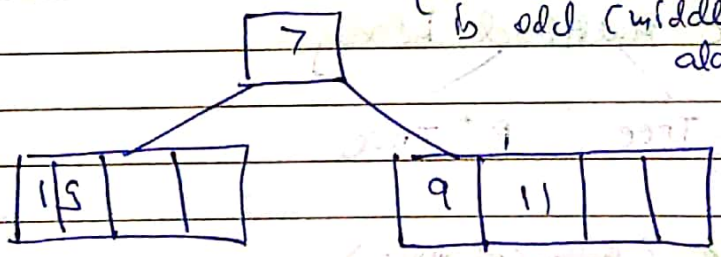
~~(7 becomes root)~~



← but max element inserted is 4 (So lerealeing happens)

7 becomes root

(when order is odd (middle is parent always))



d) when order is even

order : 4

max element : 4 - 1 = 3

min " = $\left\lceil \frac{4}{2} \right\rceil - 1 = 1$

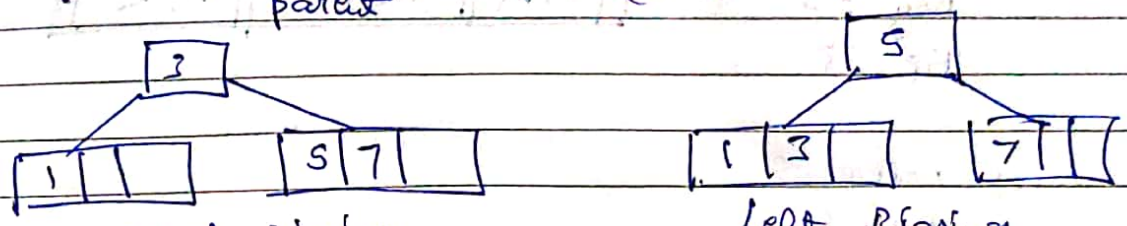
← 7

→

1	3	5	7

(violate condition)

either 3 as a parent (or) 5 as a parent



Right Biasing (right node has more weight)

Left Biasing (left node has more weight)

(this was the B Tree of datastructure)
(now we will make B Tree of DBMS)

Ex

o) B Tree:

in DBMS

ORDER : P

~~max key = P-1~~

Block pointer (BP) = P

Key = P-1

RP = P-1

pointer for reading purpose (if # key = n, then # RP = n to read that key)

in DSA:

ORDER : P

max ele : P-1

min " = $\lceil \frac{P}{2} \rceil - 1$

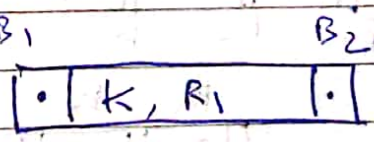
o) ORDER : 2

BP = 2

Key = 1

RP = 1

(~~here key~~

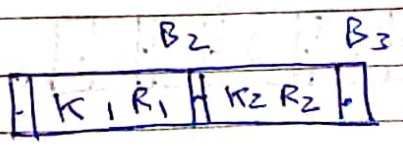


o) ORDER : 3

BP = 3

Key = 2

RP = 2



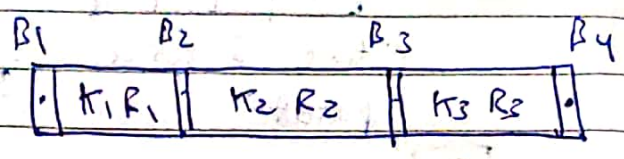
Scanned by CamScanner

o) ORDER : 4

$B_p = 4$

$Key = 3$

$R_p = 3$

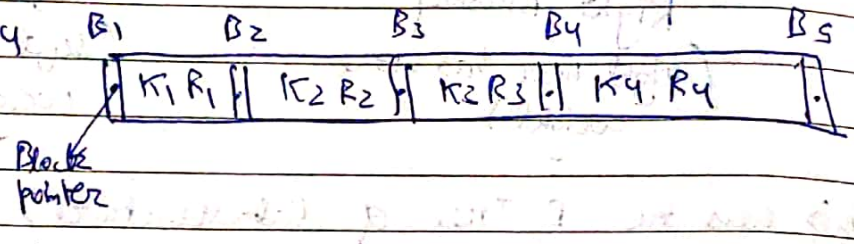


o) order : 5

$B_p = 5$

$Key = 4$

$R_p = 4$

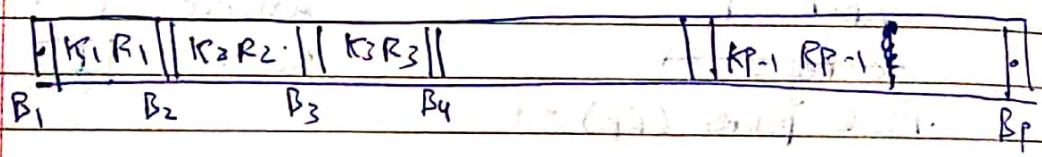


o) order : P

$B_p = P$

$Key = P-1$

$R_p = P-1$



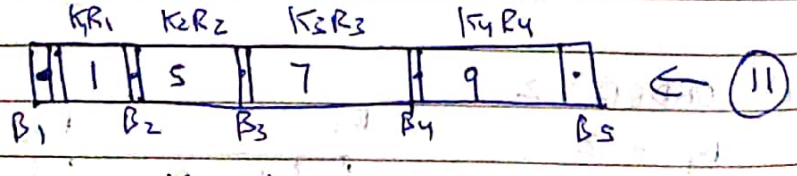
o) 1, 7, 5, 9, 11, 13

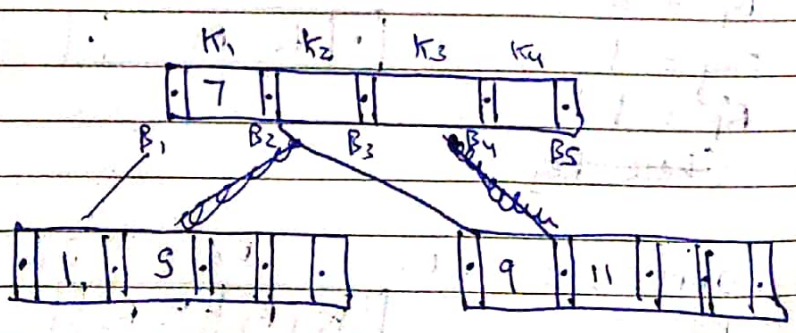
Order : 5

$B_p = 5$

$Key = 4$

$R_p = 4$





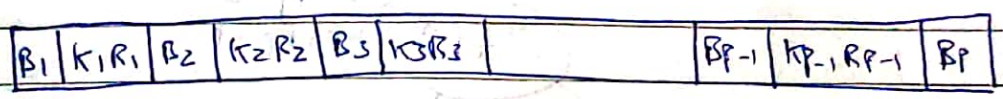
Scanned by CamScanner

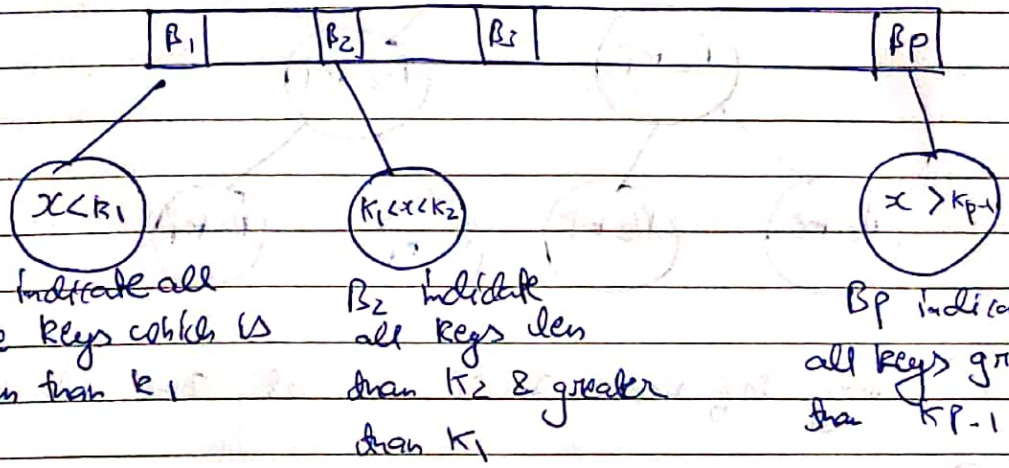
(B_p is only there for indexing)

$$\boxed{B_p = key + 1}$$

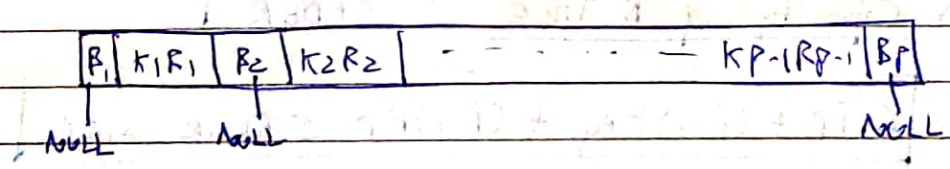
0) **Definition of B Tree** : Order : P

① 0) Structure of internal node





② Structure of leaf node



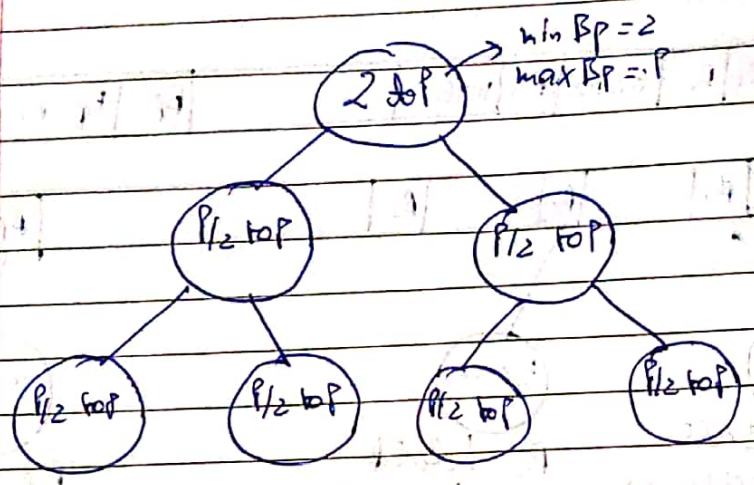
③ Every Internal node except the root node should be at least $\lceil P/2 \rceil$ Block pointer & maximum P Block pointer.

$$\min B_p = key_{min} + 1$$
$$= \lceil P/2 \rceil - 1 + 1$$
$$= \lceil P/2 \rceil$$

$$\max B_p = P$$
$$= key_{max} + 1$$
$$= P - 1 + 1$$
$$= P$$

Scanned by CamScanner

(4) Root can be atleast 2 Block pointer (∵ min key is 1 in root) & at most P Block pointer.
(root node may not be always 50% occupied)



(5) Every leaf node should be at same level

(6) Keys within the node should be in ascending order.

o) Capacity of B Tree : ORDER P

$$\boxed{P * Bp + (P-1)Key + (P-1)Rp \leq \text{Block size}}$$

if order 5
BP = 5
Key = 4
Rp = 4

so all key (4) all Bp (5) & all (Rp (4)) are accomadated in one single block

o)
Bp : block / Tree / Index Pointer
Rp : Read / Data / Record Pointer
key : Search key
P : order

Scanned by CamScanner

Gate

Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes max no of tree pointers in B-tree index node. Assume that K is 10 bytes long, disk block size is 512 B, each data pointer P_D is 8 B long & each block pointer P_B is 5 B long. In order for each B tree node to fit in a single disk block, the max value of P is.

a) 20 c) 23

b) 22 d) 32

$$P \times B_P + (P-1) key + (P-1) R_P \le Block\ Size$$

$$P \times 5 + (P-1) \times 10 + (P-1) \times 8 \le 512$$

$$5P + 10P + 8P - 18 \le 512$$

$$23P \le 530$$

$$P \le 23.043$$

0) Definition of B^+ Tree order : P

① ~~Structure of Internal node~~ Change in B^+ tree

In internal node => no read pointer

all key is copied at leaf node

leaf node only 1 Block pointer

(rest all same as B Tree)

① Structure of Internal node

→

no read pointer

B_1	K_1	B_2	K_2	B_3	K_3	----------	K_{P-1}	B_P

$x \le K_1$ $\quad$ $K_1 < x \le K_2$ $\quad$ $x > K_{P-1}$

(or)

$x < K_1$ $\quad$ $K_1 \le x < K_2$ $\quad$ $x \ge K_{P-1}$

(2) Structure of leaf node:

K_1R_1	K_2R_2	K_3R_3	----	$K_{P-1}R_{P-1}$	

(all Keys are copied at leaf node ∴ it has read pointer)

Block pointer (only one block pointer in leaf node)

(3)
(4)
(5)
(6)

} Same as B Tree

B^+ capacity

In B^+ Tree

Order of Internal node / # child (max)

$P * BP + (P-1) Key \le$ Block size

Leaf node order

$P * Keys + P * RP + 1 BP \le$ Block size

Gate

The order of an internal node in a B^+ tree index is the max no of children it can have. Suppose that a child pointer takes 6 Bytes, the search field value takes 14B, & block size is 512 B. what is

the order of the internal node?

a) 24 ~~c)~~ 26
~~b)~~ 25 d) 27

$$P \times 6 + (P-1) \times 14 \le 512$$
$$\cancel{20P \le 513}$$
$$\cancel{P \le 25.65} \qquad 20P \le 526$$
$$P \le 26.3$$
$$= \lfloor 26.3 \rfloor = 26$$

Crate.

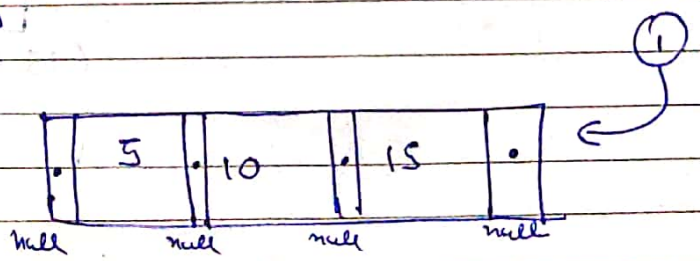
Consider a B+ tree in which the max no of key in a node is 5, what is the min no of keys in any non-root node?

a) 1 c) 3
~~b)~~ 2 d) 4

max key = 5
BP max = 5+1 = 6
P = 6 (order)
min key (non-root) $= \lceil \frac{P}{2} \rceil - 1 = 2$

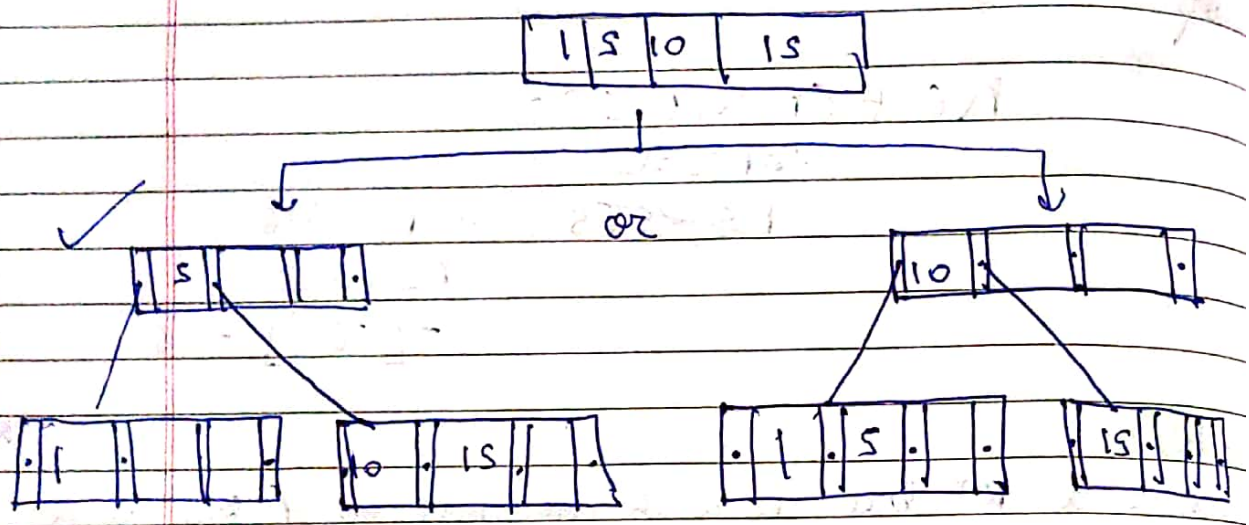
o) Creation of B Tree

Keys 5, 10, 15, 1, 23, ~~54~~ 34, 13, 8
order : 4
max. key = 3
BP = 4



when capacity is full, so to insert 1 we go for node splitting

Node splitting : Write all the keys including current one in ascending order & take middle one as root (parent)



right biasing (right node occupied more)

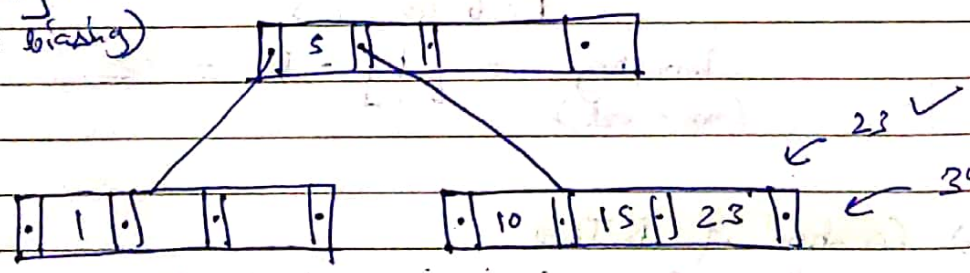
(same root in both)

left biasing (left node occupied more)

eg: let: we take right biasing

23, 34, 13, 8

(now throughout take right biasing)



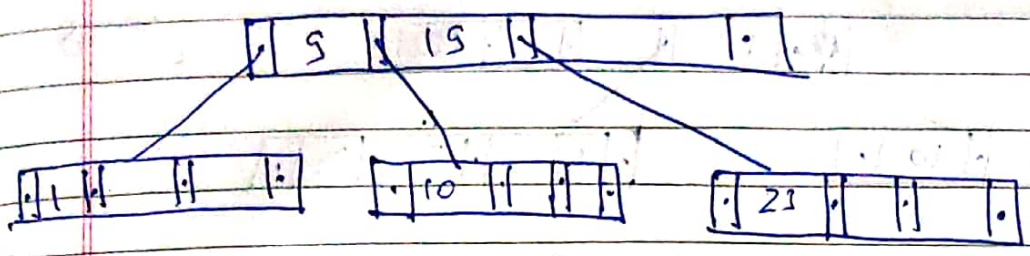
~~10 15 23 34~~

10 15 23 34

(15 become root ∵ right biasing has to be followed)

→

Note: Duplicate keys are not allowed here bcoz it is a Search tree



o) how alternate method (coe try to shift the keys in order to reduce node splitting).

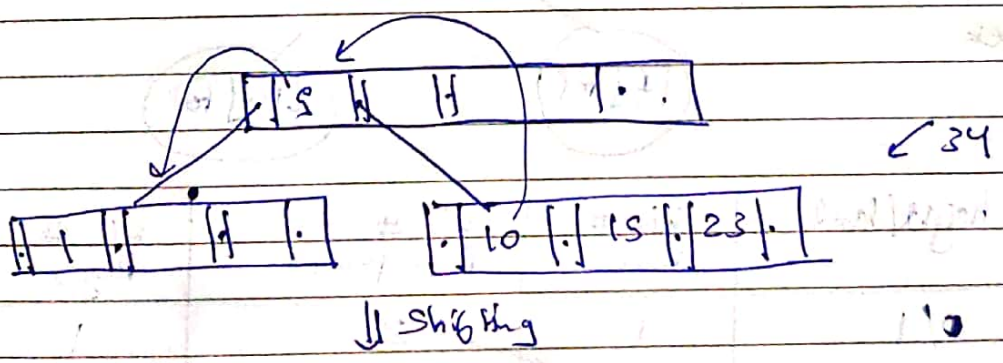
o) Alternative for node splitting

→ Redistribution of keys: if any leaf node contain free space to accomodate a new key then, redistribute the key.

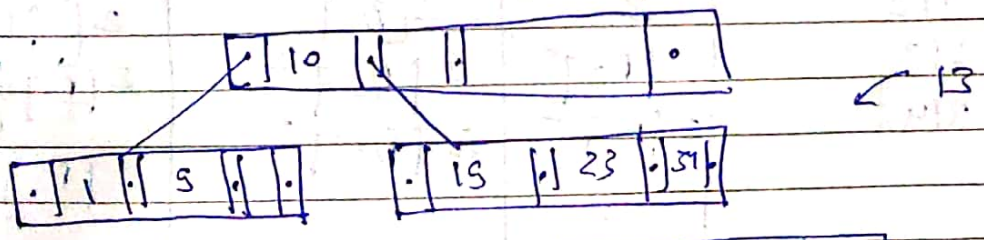
i.e accomadating new key within the existing node without creating a new node / without splitting

Ideal Insertion Scenario

↳ Try for redistribution if not possible then do splitting.

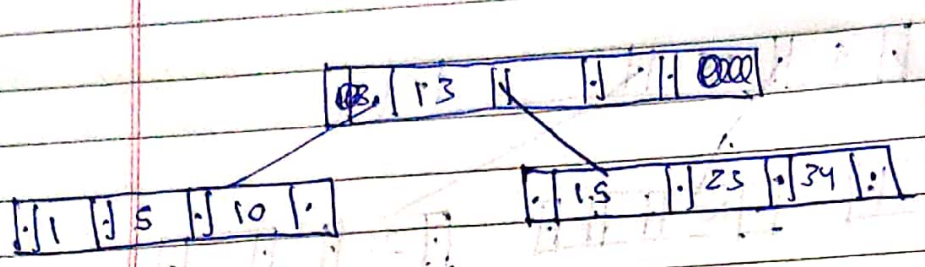


⇓ Shifting



13	15	23	34

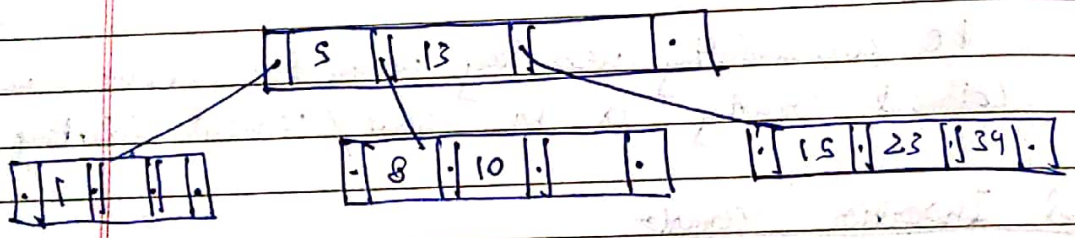
⇓ redistribute again



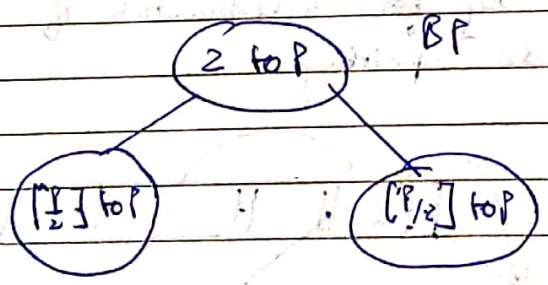
8

(now redistribution not possible ∵ leaf node are full so now node splitting)

(follow right biasing)
(5 will become root)



Order : P



height/level	min # node	min # Block pointer	min # keys
0/1	1	2	1
1/2	2	$2\lceil P/2 \rceil$	$2(\lceil P/2 \rceil - 1)$
2/3	$2\lceil P/2 \rceil$	$2\lceil P/2 \rceil^2$	$2\lceil \frac{P}{2} \rceil(\lceil P/2 \rceil - 1)$
3/4	$2\lceil P/2 \rceil^2$	$2\lceil P/2 \rceil^3$	$2\lceil P/2 \rceil^2(\lceil P/2 \rceil - 1)$
⋮			
n/n+1	$2\lceil P/2 \rceil^{n-1}$	$2\lceil P/2 \rceil^n$	$2\lceil P/2 \rceil^{n-1}(\lceil P/2 \rceil - 1)$

Scanned by CamScanner

Total min # keys in B tree order: p height h

$$\Rightarrow 1 + 2\left[\lceil p/2 \rceil - 1\right] + 2\lceil p/2 \rceil\left[\lceil p/2 \rceil - 1\right] + \cdots + 2\lceil p/2 \rceil^{h-1}\left[\lceil p/2 \rceil - 1\right]$$

$$\boxed{\text{min \# key} = 1 + 2\left[\lceil p/2 \rceil^{h} - 1\right]}$$

h: height
p: order

height	max # node	max # B_p	max # keys
0	1	p	$p-1$
1	p	p^2	$p(p-1)$
2	p^2	p^3	$p^2(p-1)$
⋮			
h	p^h	p^{h+1}	$p^h(p-1)$

max # keys in B Tree order: p, height: h

$$\Rightarrow (p-1) + p(p-1) + p^2(p-1) + \cdots + p^h(p-1)$$

$$\boxed{\text{max \# keys} = p^{h+1} - 1} \quad \text{or} \quad \boxed{p^{l} - 1}$$

l: level
$l = h+1$

Q1) order = 5, level = 5 min # key?

$$= 1 + 2\left[\lceil p/2 \rceil^{5-1} - 1\right]$$

$$= 1 + 2\left[\lceil 5/2 \rceil^{4} - 1\right] = 1 + 2[80] = 161$$

Scanned by CamScanner

Q2) order = 4 , level = 3 max # keys?

$$\text{max \#keys} = p^l - 1$$
$$= 4^3 - 1$$
$$= 63$$

Query Language

(formal QL) procedural query language
- require
  - what to retrieve from db
  - How to retrieve from db

(eg. Relational Algebra)

(informal QL) non procedural query language
- require
  - what to retrieve from db

(user friendly)
Eg:- How to retrieve depends on user

(eg. SQL, TRC)
SQL: Structural QL
TRC: Tuple Relational Calculus

Note! The basic idea of query language is query executed on a db ~~tuple~~ ~~table~~ tuple by tuple, one tuple at a time.

① Relational Algebra

→ Relational algebra by default eliminate the duplicate values but in SQL you have to use keyword "distinct" in order to eliminate duplicate values. (If your name is written 4 times relational algebra will by default display it only one time whereas SQL will display 4

times).

Basic operators:

1. projection (π)
2. Selection (σ)
3. Cross product ($\times$)
4. Union ($\cup$)
5. Minus / Set difference ($-$)
6. rename (ρ)

Derived operators

1. JOIN ($\bowtie$)
2. DIVISION ($\%$)
3. Intersection ($\cap$)

$$\boxed{R \cap S = R - (R - S)}$$

* Selection [σ]

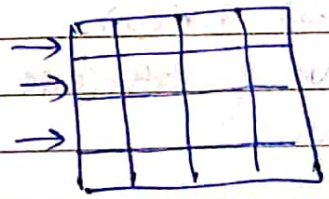
→ It select the tuple based on specified condition

$$\boxed{\text{Syntax: } \sigma_{\text{condition}} (\text{Relation})}$$

eg) $\sigma_{\text{CGPA} > 9}$ (Student) ⇒ Select the tuple where CGPA > 9 from the table Student.

eg) $\sigma_{\text{marks} > 8}$ (Student)

eg) $\sigma_{\text{color} = \text{red}}$ (Parts)

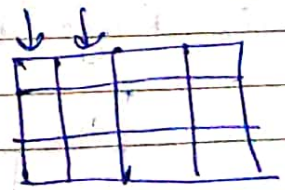


eg) $\sigma_{\text{State} = \text{'Delhi'}}$ (Student)

* Projection (π)

→ It Select attribute or attribute list from relation

$$\boxed{\text{Syntax: } \pi_{\substack{\text{Attribute} \\ \text{or} \\ \text{Attribute list}}} (\text{Relation})}$$



eg.

π_{Sname} (Student)

$\pi_{\text{Sname, contactno}}$ (Student), π_{A} (R), π_{ABCD} (R)

Scanned by CamScanner

Question

Parts

Pid	Pname	Color
P_1	A	Red
P_2	A	Green
P_3	B	Red
P_4	C	Green
P_5	A	Red

→ Retrieve name of read color parts.

$$\pi_{Pname}\left[\sigma_{Color=Red}(Parts)\right]$$

Output

Pname
A
B

will not display duplicate values (bcoz relational algebra :)

* Set operators [$\cup, \cap, -$]

→ Let there are two Relation R & S. R & S is compatable only if (below two condition must satisfy).

① No. of attribute must be same (Arity should be same)

R

S

$\cup$, $\cap$, $-$ } not possible to apply

② Range of attribute should be same

Sid	Sname

Branch	Contact

$\cup$, $\cap$, $-$ } not possible to apply

Scanned by CamScanner

Note: Set operation can be applied only on similar Tables (containing different data)

eg Registeration table of C language, Registration table of Java language

R	Sid	Sname

S	Sid	Sname

now can apply $\cup, \cap, -$

a) Assume R & S consist M & N tuple each

1. Range of tuples in $R \cup S$ = [max(m,n) to m+n]
2. Range of tuples in $R \cap S$ = [ϕ to min(m,n)]
3. Range of tuple in $R - S$ = ϕ to m
4. Range of tuple in $S - R$ = ϕ to n

→ when complete table is cancelled out

let

R	
A	
B	
C	
D	

S	
A	
B	
C	
D	
E	

$R \cup S \Rightarrow$ A, B, C, D, E

range = [max(4,5), 4+5]

= [5, 9]

↳ when all tuples are diff in both table

$R \cap S \Rightarrow$ A, B, C, D

range = [ϕ, min(4,5)] = (ϕ to 4)

↳ when nothing matches

* Cross product (x) / Cartesian Product

R X S : Result all attribute of R & all attribute of S

R

A	B	C
1	2	3
2	1	2

S

B	C	D
2	3	4
2	1	4

R X S

R.A	R.B	R.C	S.B	S.C	S.D
1	2	3	2	3	4
1	2	3	2	1	4
2	1	2	2	3	4
2	1	2	2	1	4

R

n_1 Tuple

c_1 attribute

S

n_2 Tuple

c_2 attribute

R X S

2X2 => 4 tuple

3+3 => 6 attribute

$$\boxed{\begin{array}{l} R \times S = n_1 \times n_2 \text{ Tuple} \\ \qquad\qquad c_1 + c_2 \text{ attribute} \end{array}}$$

Q) Why cross product is required
(If we require something R.B < S.C)

Reason: In query language it is not possible directly, as at a time it only fetch a tuple/row either from relation R or from Relation S, so not possible to compare. That's why we are maintaining a temporary relation (R×S) to convert them into one tuple,

Scanned by CamScanner

hence comparison is possible.

* JOIN (⋈)

1. Natural join
2. Conditional join
3. Left Outer join
4. Right outer join
5. Full outer join.

R

A	B	C
1	2	3
2	1	2

S

B	C	D
2	3	4
2	1	4

1. Natural Join (⋈)

(R ⋈ S)

Step 1) Cross product of R & S

R	S	R×S :
n_1 tuple	n_2 tuple	$n_1 \times n_2$ tuple
c_1 attri	c_2 attri	$c_1 + c_2$ attribute

Step 2) Select the tuple which satisfy equality condition on all common attribute (from R×S)

Step 3) Projection of distinct attribute.

$$R \bowtie S = \underset{\text{distinct attribute}}{\nearrow} \left[\underset{\text{equality condition on all common attribute}}{\sigma} (R \times S) \right]$$

R×S ↳

R.A	R.B	R.C	S.B	S.C	S.D	
1	2	3	2	3	4	✓
1	2	3	2	1	4	×
2	1	2	2	3	4	×
2	1	2	2	1	4	×

Scanned by CamScanner

$$R \bowtie S = \pi_{ABCD} \left[\sigma_{R.B = S.B \,\cap\, R.C = S.C} (R \times S) \right]$$

$=$

A	B	C	D
1	2	3	4

② **Conditional Join ($R \bowtie_c S$)**

$$\boxed{R \bowtie_c S = \pi_{\text{All attribute}} \left[\sigma_{\text{Given condition}} (R \times S) \right]}$$

$$R \bowtie_{R.B < S.C} S = \pi_{ABCBCD} \left[\sigma_{R.B < S.C} (R \times S) \right]$$

output

A	B	C	B	C	D
1	2	3	2	3	4
2	1	2	2	3	4

③ **Left outer join ($R ⟕ S$)**

$R ⟕ S$:- $R \bowtie S$ and (&&) all tuple from left side relation R whose which failed to satisfy join condition.

$R \bowtie S$ =

A	B	C	D
1	2	3	4

$R ⟕ S$ =

A	B	C	D
1	2	3	4
2	1	2	Null

Scanned by CamScanner

(4) Right Outer Join (R ⟖ S)

R ⟖ S = R ⋈ S && tuple from right side relation S whose which failed to satisfy the join condition.

R ⋈ S =

A	B	C	D
1	2	3	4

R ⟖ S =

A	B	C	D
1	2	3	4
Null	2	1	4

(5) Full Outer Join (R ⟗ S)

R ⋈ S ∪ R ⟖ S

⇓

A	B	C	D
1	2	3	4
2	1	2	Null
Null	2	1	4

Question

Consider a database with Relations

Supplier (Sid Sname Rating)

Parts (Pid Pname Color)

Catalog (Sid Pid Cost)

Q) find the sid of supplier whose rating > 7

$$\pi_{sid} \left[\sigma_{rating > 7} \text{ supplier} \right]$$

↳ query for above question

Scanned by CamScanner

○) Division [÷ or /]

Enrolled

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

Course

Cid
C_1
C_2
C_3

Q) Retrieve all students who are enrolled some course or any course or atleast one course

⇓

π_{Sid} (Enrolled)

o/p

Sid
S_1
S_2
S_3

**

Q) Retrieve all students who ~~are~~ have enrolled every course

$\pi_{Sid,Cid}$ (Enrolled) / π_{Cid} (course) → o/p

Sid
S_1

2nd attribute must be same

○) Division is a derived operator, it is calculated as :

π_{Sid} (Enrolled) × π_{Cid} (course)

Sid
S_1
S_2
S_3

×

Cid
C_1
C_2
C_3

⇒

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3

Scanned by CamScanner

find this

$\pi_{sid}(Enrolled) - \pi_{sid}[\pi_{sid}(Enrolled) \times \pi_{cid}(course) - Enrolled]$

o) first finding $\pi_{sid}(Enrolled) \times \pi_{cid}(course)$

sid	cid
s_1	c_1
s_2	c_1
s_3	c_1

Sid	cid
s_1	c_1
s_1	c_2
s_1	c_3
s_2	c_1
s_2	c_2
s_2	c_3
s_2	c_1
s_3	c_2
s_3	c_3

universal result (every student enrolled every course)

o) now $[\pi_{sid}(Enrolled) \times \pi_{cid}(course) - Enrolled]$

$\Downarrow$

π_{sid}

Sid	cid
s_2	c_2
s_3	c_2
s_3	c_3

these students not enrolled these course

$\Downarrow$

Sid
s_1
s_2
s_3

$-$

Sid
s_2
s_3

students id who does not enrolled every course

$\Downarrow$

Sid
s_1

Ans

student who enrolled every course

o) $\pi_{sid, cid}(\text{Enrolled}) / \pi_{cid}(\text{course})$

$= \pi_{sid}(\text{Enrolled}) - \pi_{sid}[\pi_{sid}\text{Enrolled} \times \pi_{cid}\text{course} - \text{Enrolled}]$

o) $\pi_{AB}(R) / \pi_{B}(S) = \pi_{A}R - \pi_{A}[\pi_{A}R \times \pi_{B}S - R]$

2nd attribute must be same

o) $\pi_{ABCD}(R) / \pi_{CD}(S) = \pi_{AB}(R) - \pi_{AB}[\pi_{AB}R \times \pi_{CD}S - R]$

Question

Consider a Database with Relations

Supplier (sid sname Rating)
Parts (Pid Pname color)
Catalog (sid Pid cost)

o) Retrieve sids of supplier whose rating > 8

$\pi_{sid}[\sigma_{rating > 8}\text{Supplier}]$

o) Retrieve Pid where color is Red

$\pi_{Pid}[\sigma_{color = Red}\text{Parts}]$

o) Write a query to retrieve Pid whose cost is > 10,000

$\pi_{Pid}[\sigma_{cost > 10000}\text{Catalog}]$

Scanned by CamScanner

**

Q) Retrive Sid of supplier who supplied some Red color part?

(taking 2 table parts & catalog ∵ they are connecting, can't take parts & supplier ∵ no connection)

Query 1)

$$4 \quad \pi_{sid}\left[\sigma_{\substack{C.Pid = P.Pid \\ \wedge\, P.color = Red}} (catalog \times Parts)\right]$$

500 × 400 → 2 lack entries

Query 2)

$$4 \quad \pi_{sid}\left[\sigma_{P.Pid = C.Pid}\left[\sigma_{\substack{\cancel{Part} \\ P.color = Red}} (Parts) \times catalog\right]\right]$$

11 × 500 = 5500 entries

Query 3)

$$4 \quad \pi_{sid}\left[\sigma_{\substack{color \\ = Red}} (catalog \bowtie Parts)\right]$$

200 tuples

Query 4)

$$4 \quad \pi_{sid}\left[\sigma_{\substack{color = \\ Red}} (Parts) \bowtie catalog\right]$$

11 ⋈

(optimised query)

o) All above 4 queries are correct (now code do analysis)

eg)

Parts — 400 tuples

Catalog — 500 tuples

Red color tuple = 11

200 Tuple (which Pid matches)

4 color = Red & Pid matches

final answer = 4

Scanned by CamScanner

Note: Let An attribut 'A' belongs to R only then

$$\boxed{\sigma_{A='a'}(R \bowtie S) \equiv \sigma_{A='a'}(R) \bowtie S}$$ → more efficient query

Note: Let an attribut A belongs to R only & attribut b belongs to S only.

$$\boxed{\sigma_{A='a' \cap B='b'}(R \bowtie S) \equiv \sigma_{A='a'}(R) \bowtie \sigma_{B='b'}(S)}$$

↓ more efficient query

→ **Rename operator [ρ]**

Catalog (sid pid cost)

form of Rename operator

(i) $\rho(T_1, \text{Catalog})$

T_1

sid	pid	cost

Renaming Table

(ii) $\rho_{S,P,C}(\text{Catalog})$

Catalog

S	P	C

Renaming attribute

(iii) $\rho_{1 \to S,\ 2 \to P}[\text{Catalog}]$

Catalog

S	P	cost

Renaming specific attribute

Scanned by CamScanner

Gate 2007

Consider the following relation schema

b-schema = (b-name, b-city, assets)

a-schema = (a-num, b-name, bal)

d-schema = (c-name, a-number)

Let branch, account, & depositor be respectively instances. Assume that account & depositor relations are much bigger than branch relation

Consider the following query:

$\pi_{c\text{-}name}(\sigma_{b\text{-}city = \text{"agra"} \wedge bal < 0}(branch \bowtie (account \bowtie depositor)))$

Which one of the following is most efficient version?

a) $\pi_{c\text{-}name}(\sigma_{bal<0}(\sigma_{b\text{-}city = \text{"Agra"}}(branch \bowtie account) \bowtie Depositor))$

b) $\pi_{c\text{-}name}(\sigma_{b\text{-}city = \text{"Agra"}}(branch \bowtie (\sigma_{bal<0}(account)) \bowtie Depositor))$

c) $\pi_{c\text{-}name}(\sigma_{b\text{-}city = \text{"Agra"}}(branch \bowtie \sigma_{b\text{-}city = \text{"Agra"} \wedge bal<0}(account \bowtie depo)))$

d) $\pi_{c\text{-}name}(\sigma_{b\text{-}city = \text{"Agra"}}(branch \bowtie (\sigma_{b\text{-}city = \text{"Agra"} \wedge bal<0}(account \bowtie depositor)))$

Gate + PSU

Retrieve sid of Supplier who supplied some Red parts & some green parts.

Query 1 $\pi_{sid}[Catalog \bowtie \sigma_{color = Red \wedge color = green}(Parts)]$ → Tuple

Query 2 $\rho(T_1, \pi_{sid}[Catalog \bowtie \sigma_{color = Red}(Parts)])$

$\rho(T_2, \pi_{sid}[Catalog \bowtie \sigma_{color = green}(Parts)])$

Scanned by CamScanner

$T_1 \cap T_2$

Which of following is correct?

a) Query 1 ✓ Query 2 ✗

b) Query 1 ✗ Query 2 ✓

c) Both correct

d) " Incorrect

$\sigma_{Color = Red \wedge Color = Green}$ (Parts)

→ 0 tuple

∵ One single tuple can't have both the colors.

Question) Retrieve sid of the supplier who supplied at least one part.

π_{sid} (Catalog)

Question) Sids of supplier who supplied at least 2 parts.

$\rho(T_1, Catalog)$ $\rho(T_2, Catalog)$

$$\pi_{T_1.sid}\left[\sigma_{T_1.sid = T_2.sid \wedge T_1.pid \neq T_2.pid}(T_1 \times T_2)\right]$$

Question) Sid of supplier who supplied at least 3 parts

$\rho(T_1, Catalog)$ $\rho(T_2, Catalog)$ $\rho(T_3, Catalog)$

Scanned by CamScanner

$$\pi_{T_1.Sid}\left[\sigma_{\substack{T_1.Sid = T_2.Sid = T_3.Sid \\ \wedge\ T_1.Pid \neq T_2.Pid\ \wedge \\ T_2.Pid \neq T_3.Pid\ \wedge \\ T_3.Pid \neq T_1.Pid}}(T_1 \times T_2 \times T_3)\right]$$

Gate 2008

$R(P, Q, R_1, R_2, R_3)$

$S(P, Q, S_1, S_2)$

where $\{P, Q\}$ is the key for both schemas. Which of the following queries are equivalent?

1) $\pi_P(R \bowtie S)$
2) $\pi_P(R) \bowtie \pi_P(S)$
3) $\pi_P(\pi_{P,Q}(R) \cap \pi_{P,Q}(S))$
4) $\pi_P(\pi_{P,Q}(R) - (\pi_{P,Q}(R) - \pi_{P,Q}(S)))$

$R \cap S = R - (R - S)$

$3 = 4$

a) 1 & 2 c) 1, 2, 3

b) 1 & 3 d) 1, 3, 4

Eg.

R

P	Q	R_1	R_2	R_3
1	2	3	4	5
~~6~~	~~7~~	~~8~~	~~9~~	~~10~~

S

P	Q	S_1	S_2
a	2	c	d
~~e~~	~~f~~	~~g~~	~~h~~
e	f	g	h

$R \bowtie S$

P'	Q'	R_1	R_2	R_3	P''	Q''	S_1	S_2
1	2	3	4	5	a	2	c	d
1	2	3	4	5	e	f	g	h

P	Q
1	e

P	Q
1	e

π_P

P	Q	R_1	R_2	R_3	S_1	S_2
1	2	3	4	5	c	d

P	Q	$\cap$	P	Q	P	Q
1	2		1	2	1	2

Scanned by CamScanner

Gate 2015

$R_1(A, B)$ with tuples $(1, 5), (3, 7)$
$R_2(A, C)$ " " $(1, 7), (4, 9)$

Assume that $R(A, B, C)$ is full natural outer join of R_1 & R_2. Consider the following tuples of form (A, B, C)

$a = (1, 5, \text{null})$ $d = (4, 7, \text{null})$ $g = (4, \text{null}, 9)$
$b = (1, \text{null}, 7)$ $e = (1, 5, 7)$
$c = (3, \text{null}, 9)$ $f = (3, 7, \text{null})$

Which one of the following is correct?

a) R contains a, b, e, f, g but not c, d
b) " " all
c) " " e, f, g but not a, b
d) R contains e but not f, g

R_1

A	B
1	5
3	7

R_2

A	C
1	7
4	9

$R_1 \times R_2$

$R_1.A$	$R_1.B$	$R_2.A$	$R_2.C$
1	5	1	7
1	5	4	9
3	7	1	7
3	7	4	9

$R_1 \bowtie R_2 =$ $R_1 \bowtie R_2$

A	B	C
1	5	7

$R_1 \bowtie R_2 =$

A	B	C	
1	5	7	→ e
3	7	Null	→ f
4	Null	9	→ g

Scanned by CamScanner

Gate 2012

Consider the following relations A, B, C

A

ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

ID	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

Id	Phone	Age
10	2200	02
99	2100	01

How many tuples does the result of the following relational algebra expression contain? Assume that schema of A ∪ B is same as that of A.

$(A \cup B) \bowtie^{C}_{A.Id > 40 \vee C.Id < 15}$ => Conditional join

a) 7 c) 5

b) 4 d) 9

$(A \cup B) \bowtie^{C}_{A.Id > 40 \vee C.Id < 15} = \pi_{\text{all attribute}} \left[\sigma_{A.Id > 40 \vee C.Id < 15} ((A \cup B) \times C) \right]$

A ∪ B =>

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11
25	Hari	40
98	Rohit	20

C

Id	Phone	Age
10	2200	02
99	2100	01

$\sigma_{A.Id > 40, C.Id < 15}$

$(A \cup B) \times C$

A.Id	A.Name	A.Age	C.Id	C.Phone	C.Age
12			10		
15			10		
99			10		
99			99		
25			10		
98			10		
98			99		

7 rows

Scanned by CamScanner

Gate 2011

db table by name Loan_Records is given below:

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderjan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderjan	7000.00

Count is o/p of following SQL query?

Select count(*) from (Select Borrower, Bank_Manager from Loan_Records) as a natural join (Select bank_manager, loan_amount from Loan_records) As T);

a) 3 c) 5
b) 9 d) 6

R

A	B	C
1	x	m
2	y	p
3	x	q

S

B	C
x	m
y	p
x	q

R × S

R1.A	R1.B	S1.B	S1.C
1	x	x	m
1	x	y	p
1	x	x	q
2	y	x	m
2	y	y	p
2	y	x	q
3	x	x	m
3	x	y	p
3	x	x	q

R ⋈ S

R.A	B	C
1	x	m
1	x	q
2	y	p
3	x	m
3	x	q

→ count(*) = 5 tuples

case insensitive

② SQL (Structured query language)

- DDL (data definition language) : modification allowed at schema (definition level) → Create, Alter
- DML (data manipulation language) | modification allowed at data level → Insert, Update, Delete
- DCL (data control language) | control transactional operations → Commit, Rollback / abort
- DQL (data query language) | used to retrive the data from db → Select, From, where

∵ not by default eliminate

SQL
- Select distinct $A_1, A_2, A_3, \dots A_n$ ≡ Project (π)
- from $R_1, R_2, \dots R_m$ ≡ Cross product ($\times$)
- where condition ≡ Selection (σ)

RA

$$\pi_{A_1, A_2 \dots A_n} \left\{ \sigma_{\text{condition}} (R_1 \times R_2 \times R_3 \cdots R_m) \right\}$$

Question

Q) Supplier (Sid Sname Rating)
Catalog (Sid Pid cost)
Parts (Pid Pname color)

Q) Write query [RA & SQL] to find Sname of Supplier who supplied some red color parts.

R.A:

$$\pi_{\text{Sname}} \left[\sigma_{\substack{S.Sid = C.Sid \wedge \\ C.Pid = P.Pid \wedge \\ P.color = Red}} [\text{Supplier} \times \text{Catalog} \times \text{Parts}] \right]$$

Scanned by CamScanner

SQL: Select Sname
from Supplier as s, Part p, Catalog as c
where S.Sid = C.Sid and
C.pid = P.Pid and
P.color = Red

space

Rename operator
AS, ↳ Space

Gate 2003

Select distinct $a_1, a_2, \ldots, a_n$
from $r_1, r_2, \ldots, r_m$
where p
p: arbitrary, predicate

RA =>

a) $\pi_{a_1, a_2, \ldots a_n} \sigma_p (r_1 \times r_2 \times \cdots r_m)$

b) $\pi_{a_1, a_2 \ldots a_n} \sigma_p (r_1 \bowtie r_2 \ldots$

c) $\pi_{a_1, a_2 \ldots} \sigma_p (r_1 \cup r_2 \ldots$

d) $\pi_{a_1, a_2} \sigma_p (r_1 \cap r_2 \ldots$

o) Difference b/w Select [SQL] & π [RA]

Select: Not going to eliminate the duplicate value by default. i.e duplicate retain.
But in Projection (π) it eliminate duplicate values by default

o) But to eliminate the duplicate values in SQL we use <u>distinct</u> keyword.

R

A	B	C
2	5	6
2	5	6
1	2	3

Select A, B from R $\rightarrow$

A	B
2	5
2	5
1	2

$\pi_{A,B}(R) \rightarrow$

A	B
2	5
1	2

Select distinct A, B from R

Q) Retrieve Sid of supplier who supplied some red part or green part.

RA: $\pi_{sid}\left[\sigma_{c.Pid = P.Pid \wedge Color = Red \vee Color = Green}(Catalog \times Parts)\right]$

SQL:

Select sid from Catalog c, Part P where c.Pid = P.Pid and Color = Red or Color = Green;

Q) Retrieve Sid of supplier who supplied some red color part & some green color part.

RA:

$\pi_{sid}\left[\sigma_{Color = Red}(Catalog \bowtie Parts)\right] \cap \pi_{sid}\left[\sigma_{Color = Green}(Catalog \bowtie Parts)\right]$

Scanned by CamScanner

SQL:

Select Sid
from Catalog C_1, Parts P_1
where C_1.Pid = P_1.Pid and Color = Red
Intersect
Select Sid
from Catalog C_2, Parts P_2
where C_2.Pid = P_2.Pid and Color = Green

o) Basic SQL clauses

[ ] ← optional

Select [distinct] ~~from~~ A_1 A_2 A_3 ... A_n
from R_1, R_2 R_3 ... R_m
[where P]
[group by Attribute [Having condition]]
[order by Attribute [DESC]]
↳ default ascending order

o) Execution of Query

(1) from clause: Query execution starts from 'FROM' clause. It is the first executable clause. It is simply a relation or cross product of the two or more relations.

(2) where clause: Second executable clause if used in the query.
o) It select the tuples based on the specified condition.

(3) Group by: third executable clause if used in query. It group the table based on the

Count (name/constant) = # rows

specified attribute.

o) Last executable clause is select clause. (which is used to display the data).

Sid	Branch	Marks
S1	CS	90
S2	IT	70
S3	CS	70
S4	EC	51
S5	CS	Null

group by branch →

Sid	Branch	Marks
S1	CS	90
S3	CS	70
S5	CS	Null
S2	IT	70
S4	EC	51

In what order they are inserted we write in that order only ∵ CS case first then IT then EC.

o) Aggregate operator (always discard null value)

→ There are five aggregate operators supported by SQL but not by RA.

1. count ([distinct] Attribute)
2. Sum ([distinct] Attribute)
3. Avg ([distinct] attribute)
4. min (attribute)
5. max (attribute)

} Can be used over number, strings, dates

Sid	Branch	Marks
S1	CS	90
S3	CS	70
S5	CS	Null
S2	IT	70
S4	EC	51

1. count (marks) = 4
2. count (*) = 5
3. count ([distinct] marks) = 3
4. Sum (marks) = 281
5. Sum ([distinct] marks) = 211
6. avg (marks) = $\frac{\text{Sum(marks)}}{\text{count(marks)}} = \frac{281}{4}$
7. avg ([distinct] marks) = $\frac{211}{3}$

$\frac{\text{Sum([distinct] marks)}}{\text{count([distinct] marks)}}$

Scanned by CamScanner

count(*) → will return no. of tuples in a relation.

Note: ~~Arit~~ Arithematic operation with null will give result null

$\boxed{\text{null} + 10 = \text{null}}$

Question

Q) Student

Rno	name	m_1	m_2
1	A	1	2
2	B	2	4
3	C	4	6
4	D	4	8
5	E	4	null
6	F	null	6

(1) Select max(name) from Student

→ output: F

(2) Select avg(name) from Student

→ o/p: Error

(Avg function can only be used with number)

(3) Select count(*) from Student

→ o/p: 6 #tuples/rows

(4) Select count(m_1) from Student

→ o/p: 5 return # of non-null values

(5) Select count(4) from Student

→ 6 ~~return #~~

return count(constant) = # rows

(6) Select count (suresh) from student
constant
↳ o/p 6 throws

(7) Select count ($m_1 + m_2$) from student

$m_1 + m_2$
3
6
10
12
null
null

⇒ count ($m_1 + m_2$)
↳ o/p : 4

> Select sum (m_1, m_2) from student
error
✗

(8) Select count (Distinct m_1) from student
↳ 3

(9) Select sum ($m_1 + m_2$) from student

$m_1 + m_2$
3
6
10
12
null
null

sum (null discard null values)
sum ($m_1 + m_2$)
↳ $3 + 6 + 10 + 12$
$= 9 + 22 = 31$

o/p : 31

o) Select [Distinct] $A_1 A_2 \ldots A_n$
from $R_1 R_2 R_3 \ldots R_m$ → cross product
[where condition] → selection of tuples
[group by ~~having~~ attribute [having condition]]
↓
grouping of data

Scanned by CamScanner

o) **Having:** fourth executable clause if used in the query.

o) It is used to select the group those which satisfy the condition (condition is applied for each group).

Student

Sid	Branch	Marks
S1	CS	60
S2	IT	70
S3	CS	90
S4	IT	60
S5	EC	51
S6	EC	null

group by branch ⟹

Sid	Branch	Marks
S1	CS	60
S3	CS	90
S2	IT	70
S4	IT	60
S5	EC	51
S6	EC	null

Q) Retrieve student whose branch (group) avg marks are greater than 60

Select * from student group by (branch) having avg (marks) > 60

↳ condition applied for every group in a table.

o/p

Sid	Branch	Marks
S1	CS	60
S3	CS	90
S2	IT	70
S4	IT	60

CS $\frac{60+90}{2} > 60$ ✓

IT $\frac{70+60}{2} > 60$ ✓

EC $\frac{51}{1} > 60$ ✗

o) **where vs having**

→ where is used to select the row whereas having is used to select the group.

→ Aggregate function cannot be used in where clause but can be used in having clause.

Q) Select min (marks) from student

↳ o/p: 51

o) Select min (marks) from student where branch = 'CS';

↳ o/p: 60

Q) Select min (marks) from student group by branch;

↳ o/p: 60
60
51

whenever group by exist then aggregation [g.ki.(marks)] done for every group

Q) Select min (marks), branch from student; ✗ (wrong)

(such syntax not allowed in SQL)

Note:

o) aggregate operator & other attribute used in select clause is allowed only if other attribute must be in group by clause.

o) Select min (marks), branch from student group by (branch)

min (marks)	branch
60	CS
60	IT
51	EC

Scanned by CamScanner

Q) Select min(A), B
from R
group by (C) ✗ → Syntax error
group by (B) ✓

Q) Retrieve Sid & marks of Student who secured highest marks.

Query 1) Select Sid, ~~max~~ max(marks) from Student
✗

Query 2) Select Sid, max(marks) from Student group by (Sid);
✗

Query 3) select Sid, marks from student where marks = (Select max(marks) from student);
✓

Q) SQL Set Operators
1. union / union all
2. Intersect / Intersect all
3. minus / minus all

↑	↑
followed by RA	not followed by RA

Q) R

1
1
1
2
2
4
6
6

S

1
1
2
4
5
5

Scanned by CamScanner

① R union S
↳ result distinct tuples

1
2
4
5
6

② R union all S
↳ result all tuples
(complete R & complete S)

1
1
1
2
2
4
6
6
1
1
2
4
5
5

③ R Intersect S
↳ distinct common tuples from R & S

1
2
4

④ R intersect all S
↳ How many max no of time common in both R & S

1
1
2
4

⑤

R minus S
↳ distinct tuple from R those are not there in S

Scanned by CamScanner

(6) R minus all S

↳ # duplicates in R − # duplicates in S

→

1
2
6
6

o) Other set operators

(1) IN
(2) NOT IN
(3) ANY
(4) ALL

mostly used in nested query.

(5) Exist
(6) not exist

o) Comparison operator <, >, <=, >=, <>, =

<> → not equal to

o) Supplier (Sid Sname Rating)
Parts (Pid Pname color)
Catalog (Sid Pid cost)

Q) Retrieve Sid of Supplier who supplied some red color parts.

Query 1) Select Sid from Catalog C, Parts P where P.Pid = C.Pid and P.Color = red;

Scanned by CamScanner

Query 2) Select Sid
from catalog
where Pid = (select Pid from parts where
color = 'Red');

✗ (one to many relation not allowed)

Pid → | Pid |
| --- |
| P_1 |
| P_2 |

Query 3) Select Sid
from Catalog
where Pid in (Select Pid from Parts where
color = 'Red');

✓

↳ Pid not in (")

↓

Sid whose Pid color not red

o) Any

① $x >$ Any (10, 20, 30)

⇓

$(x > 10)$ or $(x > 20)$ or $(x > 30)$ } o/p 11, 12, 13, ...

② $x <$ Any (10, 20, 30)

⇓

$(x < 10)$ or $(x < 20)$ or $(x < 30)$ } o/p: 29, 28, 27, ...

o) Compare a value with each value in a set & return true if any value is compared according to given condition.

o) All

① x > all (10, 20, 30)
(x>10) and (x>20) and (x>30)
o/p : 31, 32, ...

② x < all (10, 20, 30)
(x<10) and (x<20) and (x<30)
o/p : 9, 8, 7, ...

→ Compare a value with each value in a set & return true if the given condition satisfied for every value in a set.

Q) Find the name of supplier whose turnover is better than the turnover of any (some) supplier of 'Hyderabad'

Supplier

Sno	Sname	city	Turnover
1	A	Hyd	3L
2	B	Delhi	4L
3	C	Hyd	5L
4	D	Noida	6L

> Select Sname from Supplier
where ~~city <> 'Hyd'~~
turnover > any (Select turnover from supplier where city = "Hyd");

any (3L, 5L)

o/p.
Sname
B 4L
C 5L → hyd (this also comes in o/p)
D 6L

Should not be included in o/p

So correct query is

> Select Sname from Supplier
where city <> 'Hyd' and
turnover > any (Select turnover from supplier where city = "Hyd");

Scanned by CamScanner

Q) Retrieve Eid's who get more salary than any employee of dep no = 5.

Emp

Eid	dno	Salary
E1	5	40K
E2	3	15K
E3	5	20K
E4	3	35K
E5	4	55K

> Select Eid from Employee where dno <> 5 and salary > any (Select salary from Employee where ~~dep~~ dno = 5);

> any (40, 20)

o/p.

Eid
E4
E5

Same query.

Q) > select Eid
from Emp
where dno <> 5 and
salary > (select min (Salary) from emp
where dno = '5')

Q) If all then use max (Salary)

* Co-related nested query

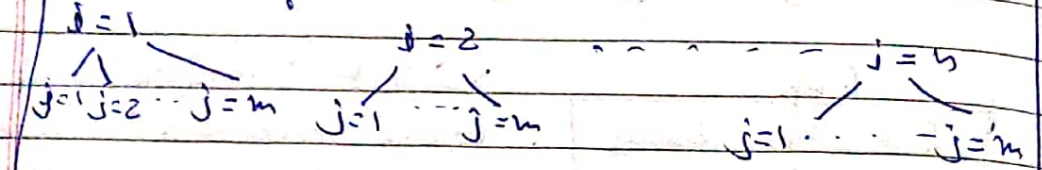
Inner query using attribute defined in outer query

Select C.Sid
from catalog C
where exists (Select *
from Parts P
where P.Pid = C.Pid and
Color = Red)

(it is similar as nested loop)

Scanned by CamScanner

```
for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
```



Nested Query

- ~~tote~~ independent nested query
- Co-related nested query

Independent nested query	Co-related nested query
Execution: Bottom → Top, Inner → Outer	Execution: Top → Bottom → Top; Outer → Inner → Outer

o) **exists:** [check] - return true if Inner query result has empty.

o) **not exists:** return true if Inner query result is empty.

Catalog

Sid	Pid
S_1	P_1
S_1	P_2
S_2	P_3
S_4	P_4

Parts

Pid	Color
P_1	Red
P_2	Green
P_3	Red
P_4	Green

o) > Select C.sid from catalog C
where exists (Select * from parts P where
P.Pid = C.Pid and Color = Red);

Scanned by CamScanner

1st iteration

S_1 P_1 → catalog

compare with Parts Pid & color = Red

compare

S_1 P_1 → P_1 Red ✓ (pid match & color = Red)

→ P_2 Green ✗

→ P_3 Red ✗

→ P_4 Green ✗

1 tuple return

IInd iteration

S_1 P_2

P_1 R ✗

P_2 G ✗ pid match but color not red

P_3 R ✗

P_4 G ✗

0 tuple return

IIIrd iter

S_2 P_3

P_1 R ✗

P_2 G ✗

P_3 R ✓

P_4 G ✗

1 tuple return

IVth

S_4 P_4

P_1 R ✗

P_2 G ✗

P_3 R ✗

P_4 G ✗

0 tuple match

Output:

Sid
S_1
S_2

Scanned by CamScanner

Gate

Given Emp (Eid Ename Sal)
~~Dependent~~ department (Eid dname Code)

Retrieve employee who have no ~~dependent~~ department

Query 1) select E.Eid
from Emp E, dep D
where E.Eid <> D.Eid

Query 2) select Eid
from Emp E
where not exists (select * from dep D
where D.Eid = E.Eid)

Which is true?
a) query 1) correct, query 2) incorrect
b) Q1) ✗ Q2) ✓
c) Q1 ✓ Q2 ✓
d) Q1 ✗ Q2 ✗

Emp

Eid
E_1
E_2
E_3
E_4

dep

Eid	dname
E_1	A
E_1	B
E_2	B

o/p should be

Eid
E_3
E_4

if Query 1)

Eid
E_1
E_2
E_3
E_4

$E_1 <> E_1$

$E_1 <> E_1$ ✗
$E_1 <> E_1$ ✗
$E_1 <> E_2$ ✓
$E_2 <> E_1$ ✓

Query 2)

E1 exist : false

E1 — E1

E1 — E2

E2 — false

E3 — true

E4 — true

o/p

Eid
E3
E4

o) Null

- → non zero
- → unknown / unexisted
- → no two null value are equal

Emp

Eid	Ename	Pno
E1	A	10
E2	B	null
E3	C	9

o) Retrieve Emp id who doesn't have any passport detail?

> Select Eid
from Emp
where Pno = null ; (no two value are equal)

X (no two null value are equal)

Note for comparison with null SQL support is / is not clause.

Select Eid
from Emp
where Pno is null.

eg) > Select Eid
from Emp
where Pno is not null

Emp Id having passport detail

eg)

%	: zero or more character
'_'	: exactly one character

eg) Retrieve student whose name start with A & ends with 'A' & atleast 3 character

> Select * from student
where sname like 'A_ _ _ % A';

like: like condition is used to specify certain search condition for a pattern in a column.

> Select * from student where
name like S% : name starts with S
% A : " end " A
% I % : containing I
_ _ _ : all 3 length name
S _ _ : starts with S & length = 3
S _ % : " " " & at least length = 3

Scanned by CamScanner

Gate 2000

given relations r(w, x) & s(y, z) the result of
Select distinct w, x from r, s is guranteed to
be same as r, provided
a) r has no duplicates & s is non-empty
b) r & s have no duplicates
c) s has no dup & r is non-empty
d) r & s have the same no of tuples.

Gate 2004

Student (roll-no, name, deptid)
Department (dept_id, dept_name)

Insert into department values (1, 'Mathematics');
" " " " (2, 'Physics');
" " student " " (1, 'Navin', 1)
" " " " (2, 'Mokesh', 2)
" " " " (3, 'Gita', 1)

How many rows & columns will be retrieved
by following SQL stat?
Select * from student, department

→ cross product

a) 0 row 4 col
b) 3 r, 4 c
c) 3 r, 5 c
d) 6 r, 5 c

Stud : 3 r, 3 col
dept : 2 row, 2 col
⇓
× (cross product)
6 r, 5 col

Gate 2011

Consider a db table T containing two columns X
& Y each of type Integer. After creation of table
one record (X=1, Y=1) is inserted.

Scanned by CamScanner

Let MX & MY denote respective max values of X & Y among all records in table at any point in time. Using MX & MY new records are inserted in table 1 8 times with X & Y values being MX+1, 2*MY+1 resp.

What is o/p of following SQL query
Select Y from T where X=7;

a) 127
b) 255
c) 129
d) 257

T		Y = 2MY+1
X	Y	X = MX+1
1	1	MX = 1
2	3	MY = 1
3	7	MX = 2
4	15	MY = 3
5	31	MX = 3, MY = 7
6	63	MX = 4, MY = 15
7	127	

Gate 2015

Cinema (theater, address, capacity)
Which of the following options will be needed at end of SQL query
Select P1.address
from cinema P1
such that it always finds the address of theaters with max capacity?

a) where P1.capacity >= all (select P2.capacity from cinema P2)

b) where P1.capacity >= any (Select P2.capacity from cinema P2)

c) " " " > all (Select max(P2.capacity) from cinema P2)

d) " " " > any (Select " " ")

③ a) **Relational Calculus** (non procedural QL)

In relational calculus query describe the result set without specifying how the answer is to be computed. This is non procedural query language.

Two variant of relational calculus

TRC (Tuple relational Calculus)	DRC (Domain relational Calculus) (not in syllabus)
a) query describe the result in the form of set of tuples.	a) query describe the result in form of set of columns (domain)
a) Tuple variable (take tuple as i/p) It is a variable that takes a tuple of a relation schema as values.	a) Domain variable It is a variable that ranges over the domain of some attribute
form of query [T / P(T)] — T: tuple variable; P(T): formula which describe tuple variable	form of query { <x1, x2 ... xn> / P(<x1, x2 ... xn>) } — x1, x2 ... xn: domain variable; P(...): formula which describe domain variable

Scanned by CamScanner

$\in$ → belongs to

T/P(T) : T : Tuple variable
P(T): formula over tuple variable T
return set of tuples such that P(T) is satisfied

T { Select sid
P(T) { from
where ...

Question

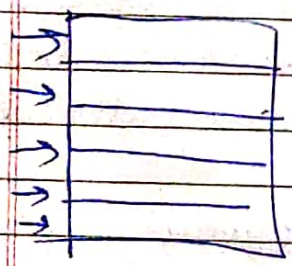
Loan (Loan_no, branch name, amount)

Q) find all the loans of an amount greater than 7000

TRC: { T / T $\in$ Loan (T.amount > 7000)
belongs to

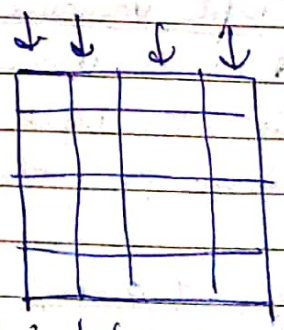
DRC: { <loan_no, branchname, amount> / <loan_no, branchname, amount> $\in$ Loan (amount > 7000) }

TRC: take tuple as i/p & apply condition on it.



takes a tuple & examine it.
In TRC Tuple as a basic unit

DRC



DRC : take a column Range over the column (domain) & examine it.

- TRC is one of the non-procedural query language
- Formula of TRC is expressed using first order logic or Predicate Logic

First order Logic [FOL]

variable are two types

- free variable
- Bounded variable

- ~~Qualifier~~ Quantifier
  - → ∀ for all
  - → ∃ there exist

- If tuple variable is preceded by the quantifier then it is a bounded variable.

  eg.

  ∀ S ∈ Supplier } Bounded
  ∃ S ∈ Supplier } variable

  there exist
  Some S which belongs to supplier

- If tuple variable is not bounded by the quantifier then it is free variable.

  [T / P(T)] T is the Result tuple variable
  & P(T) is the formula
  '∧' used for connectivity

Question

Retrieve sid of supplier who supplied some red color parts.

Scanned by CamScanner

o) RA : $\pi_{sid}\left[\sigma_{C.Pid = P.Pid}\left[\text{Catalog} \times \sigma_{color = Red}(\text{Parts})\right]\right]$

TRC: $\{T \mid \exists C \in \text{Catalog}\ (\exists P \in \text{Parts}\ (P.color = Red \wedge P.Pid = C.Pid \wedge T = C.Pid))\}$

bound variable

free variable

Just to store the result

Gate

The relation Book (Title, Price) contain the title & price of diff books. Assuming that no two books have the same price, what does the following SQL

```
Select Title from Book as B
where (Select Count(*)
       from Book as T
       where T.Price > B.Price ) < 5
```

a) Title of the fourth most expensive book
b) " " " fifthth " " "
c) " " " fifthth " inexpensive "
d) " " " 5 most expensive book

Book

Title	Price
A	100
B	400
C	300
D	500
E	200

1st iteration

	B.Price
A	100

where 4 < 5

2nd iteration

D 500 1 < 5

3rd iteration

	T.Price
A	100
B	400 ✓
C	300 ✓
D	500 ✓
E	200 ✓

Gate 2015

Student

Roll_no	Student_Name
1	Raj
2	Rohit
3	Raj

Performance

Roll_no	Course	Marks
1	Maths	80
1	English	70
2	Maths	75
3	English	80
2	Physics	65
3	Maths	80

Select S.Student_name, Sum (P. Marks)
from Student S, Performance P
where S. Roll No = P. Roll_no
group by S. Student_name

The no. of rows that will be returned by query is
2

S. Roll no	S. Student_name	P. Roll_no	P. Course	P. Marks
1	Raj	1	Maths	80
1	Raj	1	English	70
2	Rohit	2	Maths	75
2	Rohit	2	Physics	65
3	Raj	3	English	80
3	Raj	3	Maths	80

⇓ o/p

S. Student_name	Sum (P. marks)
Raj	150
Rohit	~~150~~ 140
Raj	160

group by S.S._name
Raj will be merged

o/p

Raj	310
Rohit	140

Scanned by CamScanner

Gate 2015

Select operation in SQL is equivalent to

a) the selection oper in RA

b) " " " " except that Select in SQL retains duplicate

c) projection oper

d) projection " " in RA except that Select in SQL retains duplicates

---

* Finding the no of super keys

→ Let R be the relational schema with n attributes

$A_1, A_2, A_3, \ldots, A_n$

→ How many super key are possible?

Q1) with only candidate key A_1?

Q2) with only candidate key A_1, A_2?

Q3) " " " " A_1A_2, A_3A_4?

Q4) " " " " A_1A_2, A_2A_3?

Q5) " " " " A_1, A_2, A_3?

Solution

1) C.K : A_1

→

Super keys with only C.K A_1

A_1
$A_1 \; A_2$
$A_1 \; A_2 \; A_3$
$A_1 \; A_2 \; A_3 \; A_4$
⋮
$A_1 \; \underbrace{A_2 \; A_3 \; A_4 \; A_5 \; \ldots \; A_{n-1} \; A_n}_{(n-1)\text{ attribute}}$

$$\boxed{\#\text{ Super keys} = 2^{n-1}}$$ n: # attributes

Eg) R(ABCD) C.K: A
Super key $= 2^{4-1} = 8$

★ with the help of digital logic concept

fixed	B	C	D	
A	0	0	0	→ A
	0	0	1	→ AD
	0	1	0	→ AC
	0	1	1	→ ACD
	1	0	0	→ AB
	1	0	1	→ ABD
	1	1	0	→ ABC
	1	1	1	→ ABCD

⇒ 8 Super keys.

Solution
2) ∴ C.K: A_1, A_2

→

A_1

$A_1 \; A_2$

$A_1 \; A_2 \; A_3$

A_2

$A_2 \; A_1$

$A_2 \; A_1 \; A_3$

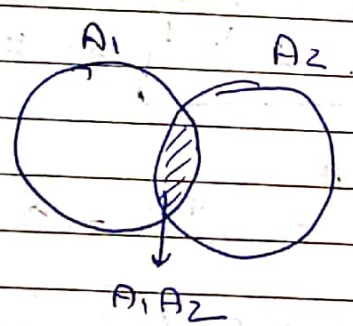
$A_2 \; A_1 \; A_3 \; A_4$

$A_1 \; \underbrace{A_2 \; A_3 \; \cdots \; A_n}_{(n-1)\text{ attri}}$

$A_2 \; \underbrace{A_1 \; A_3 \; \cdots \; A_n}_{(n-1)\text{ attribute}}$

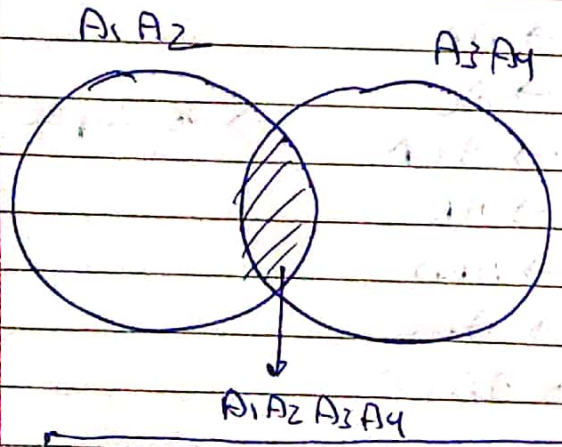
$$\boxed{\#\text{Super key} = 2^{n-1} + 2^{n-1} - 2^{n-2}}$$

broz could be repeated in both



Solution

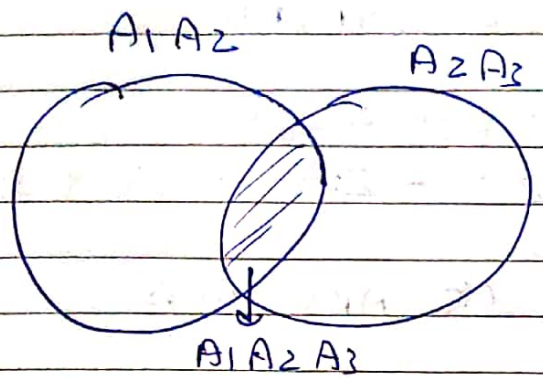
3)



$$\boxed{\#\text{Superkey} = 2^{n-2} + 2^{n-2} - 2^{n-4}}$$

Scanned by CamScanner

Solution 4)



$$\boxed{\#\text{Super key} = 2^{n-2} + 2^{n-2} - 2^{n-3}}$$

Solution 5)

CK : A_1, A_2, A_3

$$\boxed{\begin{aligned}\#\text{Super key} &= 2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} \\ &\quad - 2^{n-2} - 2^{n-2} + 2^{n-3}\end{aligned}}$$

Question

finding no. of Super keys

Q1) R(ABCDE)

CK : A

super key $= 2^{n-1} = 2^4 = 16$

Q2) R(ABCDE)

CK : A, B

$\#SK = 2^{n-1} + 2^{n-1} - 2^{n-2} = 2 \cdot 2^4 - 2^3$

$= 32 - 8 = 24$

Q3) R(ABCDE) CK : AB, CD

$\#SK = 2^{n-2} + 2^{n-2} - 2^{n-4}$

$= 2 \cdot 2^3 - 2 = 14$

Scanned by CamScanner

Q4) R(ABCDE) CK AB, BC

$\#SK = 2^{n-2} + 2^{n-2} - 2^{n-3}$
$= 2 \cdot 2^3 - 2^2 = 16 - 4 = 12$

Q5) R(ABCDE) CK ABC, BCD

ABCD

$\#SK = 2^{n-3} + 2^{n-3} - 2^{n-4}$
$= 2 \cdot 2^2 - 2 = 8 - 2 = 6$

ABCDE

Q6) R(ABCDE) C.K: ABC, CDE

$\#SK = 2^{n-3} + 2^{n-3} - 2^{n-5}$
$= 2 \cdot 2^2 - 2^0 = 8 - 1 = 7$

Gate R(EFGH) C.K : E
#Super key $= 2^{n-1} = 2^3 = 8$

PSU R(ABCDE) C.K : A, BC

#Super key $= 2^{n-1} + 2^{n-2} - 2^{n-3}$
$= 2^4 + 2^3 - 2^2$
$= 16 + 8 - 4$
$= 16 + 4 = 20$

Null value Concept.

→

Scanned by CamScanner

Exception

- not applicable
  - eg) Name
    - first name
    - middle name
    - last name

  If student that have middle name i.e middle name not applicable for particular student

- unknown value
  (we don't have the value)
  eg) Phone no is applicable but student not have the phone no.
  ∴ So Phone no is applicable but value is unknown.

If either certain field is
- → not applicable
- → unknown value

then we use special value in R Model / Relational model called null value.

null value
- → unknown value (value which is unknown)
- → unexisted value
- → Null is non zero
- → Two null are not equal
- → Sequence of Random ASCII character assigned by the dbms.

---

* **Simple key:** Key with single attribute

eg. Account_No,
Roll_No

* **Compound Key:** Key with two or more attribute
  eg. (Name Mobile_No)
  AB

* **Prime / Key Attribute:** Set of attribute that belongs to any candidate key.

* **non-prime / non-key attribute:** set of attribute that not belongs to any candidate key

* **Candidate Key:** minimal of super key.

* **Primary key:** one of the candidate key

* **Alternative / Secondary key:** All candidate keys except primary key.

let 4 C.K (A, BC, D, CE)

→ 1. Primary key let A

→ Alternative key = [BC, D, CE]

* **Primary key**

→ one of the candidate key considered as a primary key. (depending on the requirement any candidate key become primary key).

→ Primary key attribute should be not null

→ Almost one primary key for any Relation. (i.e almost one primary key can be assigned for the table (design constraint))

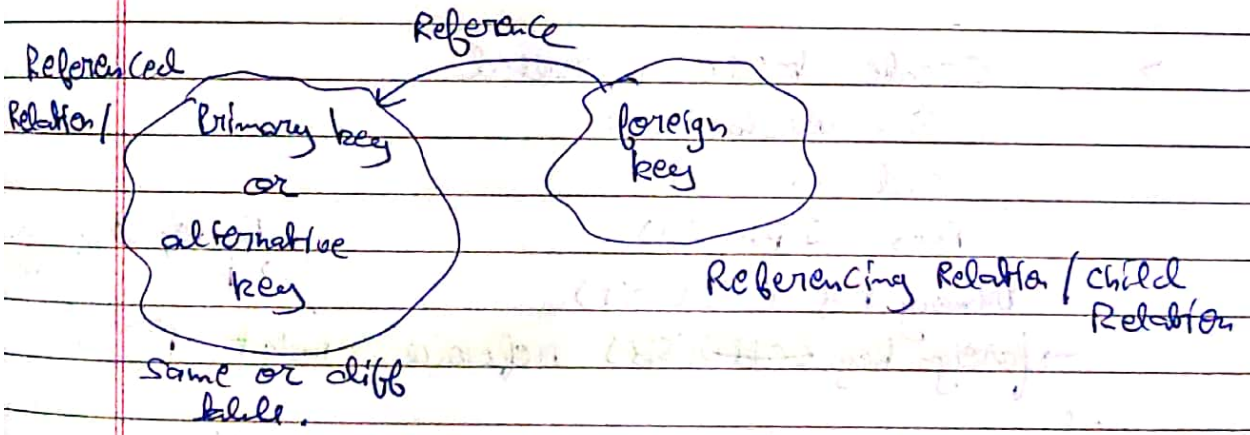
Scanned by CamScanner

→ for any db table ~~or~~ ~~these~~ there should be at least one Candidate key which has no null value.
→ But more than one alternate key is possible.

Rules for choosing Primary key from C.K

→ Choose one Candidate key which is used to Access the data more frequently gets high priority.
→ Integer datatype has more priority compare to character datatype
→ Simple candidate key has more priority compared to compound key.

* **Foreign key:** is a set of attribute reference to the primary key or alternative key of the same table or some other table.



→ It is used to relate one relation (Table) with other or same Relation (table).

o) **Referencing Relation:** Table which contain the foreign key. Also known as Child Relation.

Scanned by CamScanner

o) **Referenced Relation:** Table which is referenced by foreign key.

Student

Sid	Sname	login
S1	A	a
S2	A	a
S3	B	a
S4	C	a

Sid: Primary key
(Referenced relation)
Parent

Enrolled

Sid	Cid	Fees
S1	C1	500
S2	C2	6000
S2	C1	7000
S3	C2	8000

[Sid, cid]: primary key
Sid: foreign key
(Referencing Relation)
Child

o) Sid of Enrolled table is the foreign key referencing to the primary key of student table.

```
> Create table Enrolled
    Sid varchar (10)
    Cid   "     "
    Fees  Integer (11)
    Primary key ( Sid cid)
    foreign key (Sid) reference Student
```

By default foreign key reference to primary key

when Sid is primary key of student so no need to mention attribute name i.e Student(sid)

o) let login is primary key & Sid is alternative key.

then

foreign key (Sid) reference Student(Sid)

when Sid is not primary key

Note:
Sometimes foreign key references to the primary key of ~~same~~ same table.

Eid (p.k)	Ename	Supervisor ID (f.k)
E1		E2
E2		Null
E3		E2
E4		E3

o) **Integrity Constraint** is a condition specified on database schema & restricts the data that can be stored in an instance of the db.

eg. primary key unique not null

o) **Foreign key Constraint (Referential Integrity Constraint)**

Student (referenced relation)

Rollno (p.k)	name	Branch
1	A	CSE
2	B	IT
3	C	CSE

Registration (Referencing relation)

CNo.	Cname	Rollno (f.k)
101	DBMS	1
102	OS	1
103	CD	3
104	ToC	–

o) The value present in foreign key must be present in primary key of referenced relation

o) Foreign key may contain duplicate & null value

→

Note

Parent Table Referenced table	Child table Referencing table.
✓ Insert < 4 D ECE >	✗ Insert < 105 DSA 6 >
✗ delete < 1 A CSE >	✓ delete < 103 CD 3 >

o) Deletion from the referenced relation & Insertion into referencing relation may violate foreign key constraint.

o) A relation can act as parent & child i.e relation may contain a primary key & a foreign key that refer to the same relation.

Question

Consider a relation R(AB) where A is the primary key & B is the foreign key referencing the same relation. Which of the following row sequence can be successfully inserted into R.

a) (1, null) (2, 1) (2, 2) (3, 2)
b) (null, 1) (1, 2) (2, 3) (3, 4)
c) (1, null) (2, 1) (3, 2) (4, 2)
d) all

a)

A	B
1	null
2	1
2	2

1 null
2 1
2 2 ✗

primary key should have unique values

b) null, 1 ✗ p.k value can't be null

Referential Integrity Constraint

(1) Referenced Relation

(i) Insertion : no violation

(ii) deletion : may cause violation if primary key is used by referencing relation.

- If create a problem then we have to take care of 3 actions
  - ↳ on delete no action
  - ↳ on delete cascade
  - ↳ on delete set null (default)

(i) On delete no action : Restrict the deletion if referencittal Integrity violation occur.
(or)

→ deletion is prohibited if it cause referential Integrity violation.

foreign key (sid) reference Student (sid) on delete no action;

imp

(ii) On delete - cascade

→ deletion happen in both referencing & referenced relation if referential Integrity violation occurs.

foreign key (sid) reference Student (sid) on delete cascade

→ when data from the parent table is deleted then the related data from child table also deleted cascadely.

Scanned by CamScanner

c) consider primary key deleted, corresponding tuple in referencing relation also deleted.
i.e if violation occur then delete tuples from both relations

p.k → f.k

Empid	Ename	Salary	Supervisior Id
E1			E2
E2			null
E3			E2
E4			E3
E5			E4

delete (E2, null)

⇒ If we delete (E2, null) then deletion of E2, (E1, E2) (E3, E2)
primary key E1, E3 also deleted

⇒ Due to primary key E3 deleted ⇒ E4 also deleted

⇒ Due to primary key E4 ⇒ E5 also deleted

⇒ In this eg. result in deletion of one tuple (E2, null) completely ~~delete the~~ complete table is deleted.

Note ⇒ So ~~Sred~~ not suggestable when one table contain ~~KF~~ pk & fk both

(iii) on delete set null
If violation occur then foreign key field value tried to set null if ~~Sotem~~ Sotem then deletion allowed in the referenced relation.

o) foreign key can be null but if it is used in primary key of referencing relation then null not allowed.

Note

o) when two table are joined (natural join) with respect to primary key & foreign key then the no of tuples in the resulting relation is always equals to tuples in the foreign key relation.

R ( )
referenced relation
4000 tuple

S ( )
referencing relation
5000 Tuple

$R \bowtie S$ = 5000 tuple

Gate

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

A → primary key
C → foreign key
foreign key referencing A with on delete cascade.

o) The set of all tuples that must be additionally deleted to preserve referential integrity when (2,4) deleted is

a) (3,4) (6,4)
b) (5,2) (7,2)
c) (5,2) (7,2) (9,5)
d)

(2,4) deleted

A	C
3	4
4	3
9	2
7	2
9	5
6	4

this also gets deleted

A	C
3	4
4	3
9	5
6	4

→ also gets deleted due to (5,2)

A	C
3	4
4	3
6	4

Ans (9,2) (7,2) (9,5)

Gate

Let R(a, b, c) & S(d, e, f) be two relation in which d is the foreign key of S that refers to primary key of R. Consider the following four operations on R & S

(i) insert into R (iii) delete from R

(ii) insert " S (iv) " from S

Which of the following is true about the referential integrity constraint above?

a) none of (i), (ii), (iii), (iv) can cause violation

b) all can cause violation

c) (i) & (iv) can "

d) (ii) & (iii) " "

Q

Suppose $R_1(A, B)$ & $R_2(C, D)$ are two relation schemes. Let r_1 & r_2 be the corresponding relations instances. B is a foreign key that refers to C in R_2. If data in r_1 & r_2 satisfy referential integrity constraint, which of the following is ~~always~~ always true.

a) $\pi_B(r_1) - \pi_C(r_2) = E[\phi]$
b) $\pi_C(r_2) - \pi_B(r_1) = E[\phi]$
c) $\pi_B(r_1) = \pi_C(r_2)$
d) $\pi_B(r_1) - \pi_C(r_2) \neq E[\phi]$

---

o) **ER Model** [Conceptual database design]
is entity relationship model.

o) Entity: An object in real world

Student, Book — physical entity
Account — logical entity

o) Entity set: Collection of similar entities

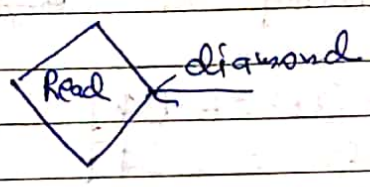
$\boxed{\text{Student}}$ ← entity set is represented with help of rectangle

Entity Set	Entity
o) Schema (Heading)	o) Instance
eg. Person (name, age, gender)	o) (Ajay, 29, male)

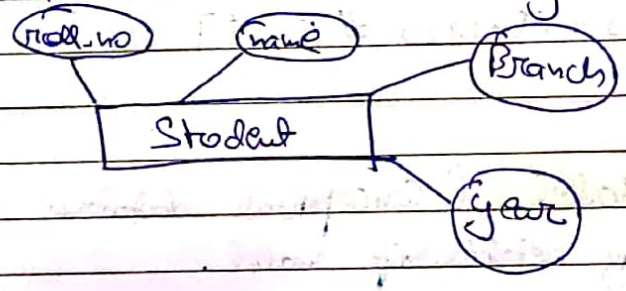
element of set is entity

- **Relationship:** Association among the entities
  eg. Reading
  Buying

- **Relationship set:** Collection of similiar relationship

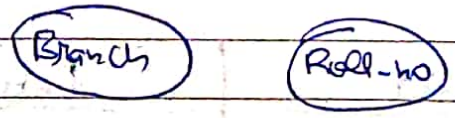


- **Attribute:** which describe an entity.



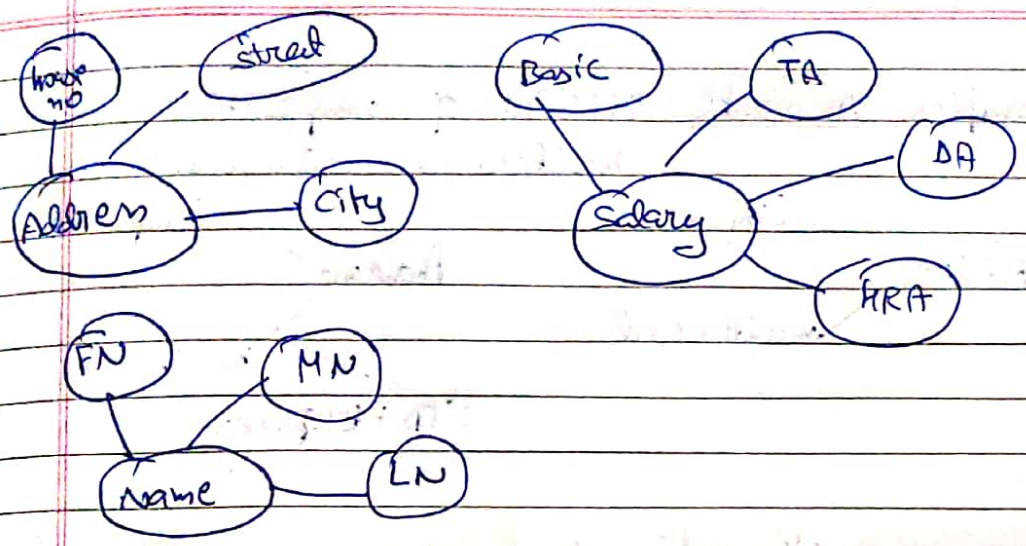
- **Classification of attribute**
  - → Simple vs composite attribute
  - → Single valued - multivalued attribute
  - → Stored & drive attribute
  - → Complex attribute

(1) **Simple attribute** which cannot be divided further

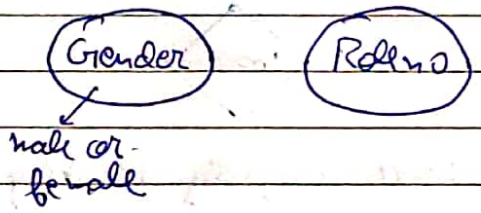


Composite attribute which can be divided further

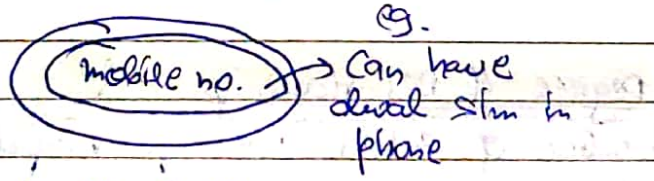
→



② Single valued attribute: which takes one value per an entity.



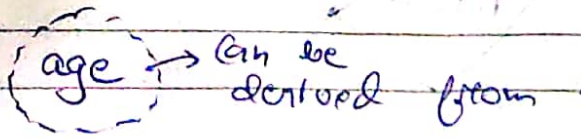
Multivalued attribute which takes more than one value per an entity.



③ Stored & Derived Attribute

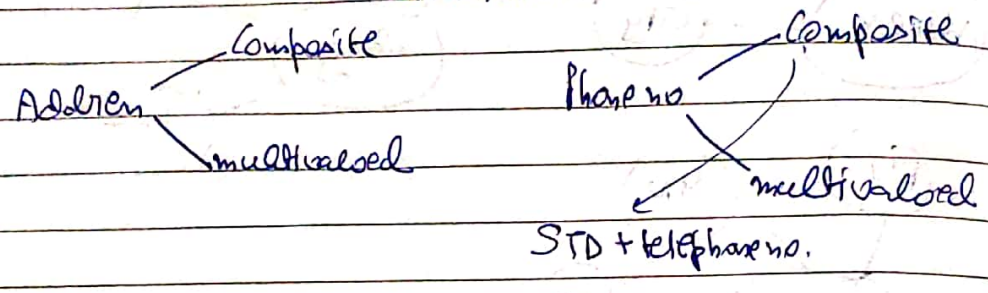
Stored attribute : which does not require any updation.

Derived attribute : The value of an attribute can be derived from other attribute

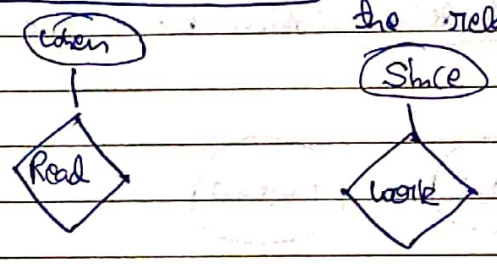


Scanned by CamScanner

(4) Complex Attribute: Collection of composite & multivalued attribute



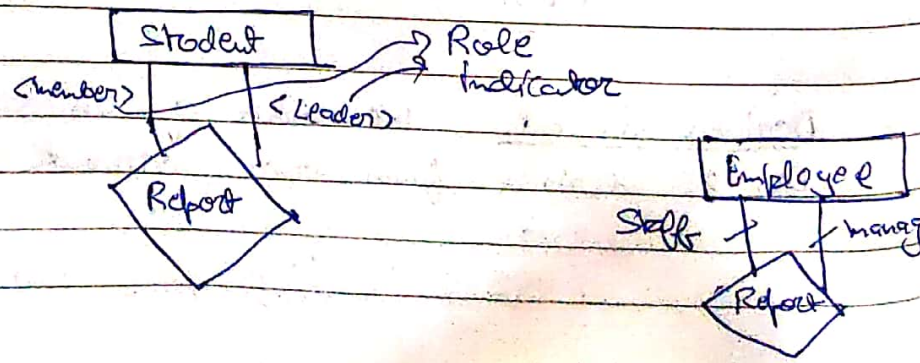
o) Descriptive attribute: which give information about the relationship set.



o) Key attribute: which uniquely identify an entity in the entity set.

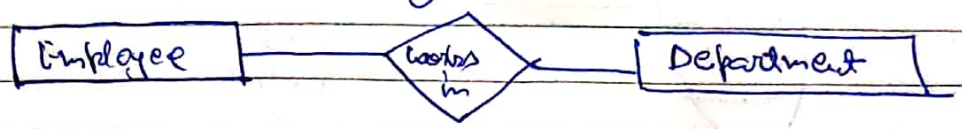
o) Degree of Relationship set: specifies the number of entity set participate in a relationship set.

(1) Unary: Relationship among two entities of the same entity set [eg. Recursive Relationship set]

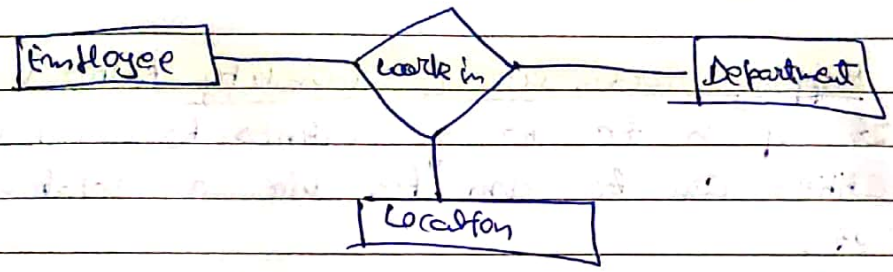


Scanned by CamScanner

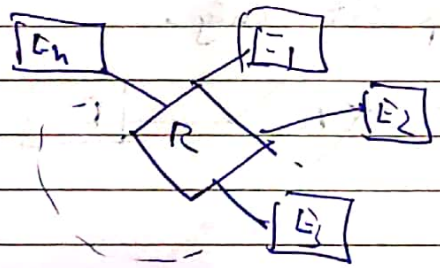
② **Binary Relationship** : The relationship among two entity set.



③ **Ternary Relationship**: Relationship among three entity set.



④ **n-ary**: Relationship among n-entity set



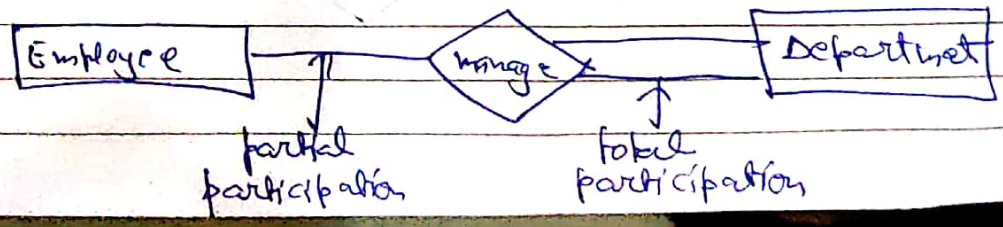
o) **Participation constraint**: If every entity in the entity set participate in a relationship set then it is called total participation.

o) Total participation is ~~do~~ denoded by

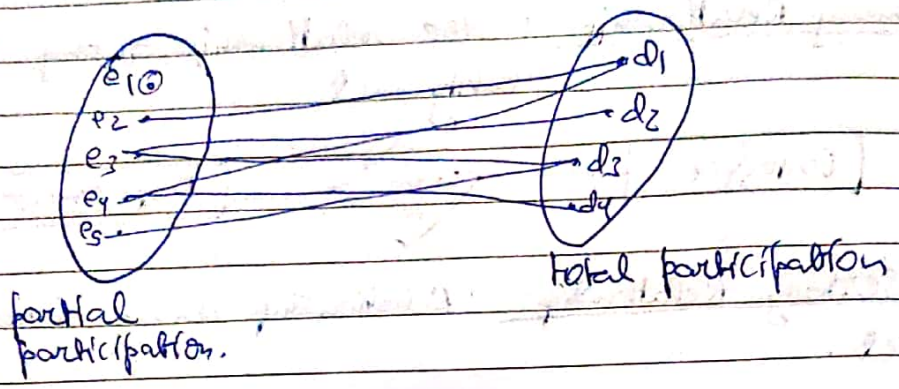
↳ ( )
double line

otherwise it is called partial participation.

eg) Each department is managed by at least one employee.



Scanned by CamScanner

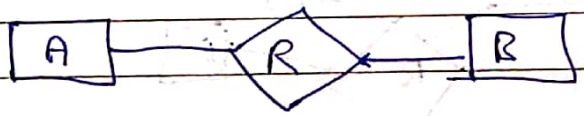


Mapping Cardinality [Cardinality Ratio]

It express the no of entities to which another entities can be associated via a relationship set.

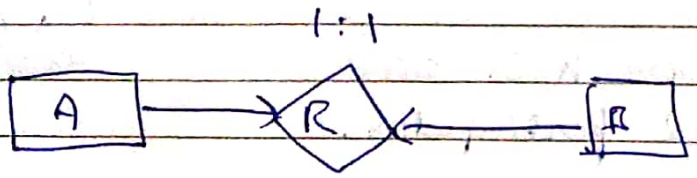
(applied only on binary relationship)

- The cardinality ratio can be expressed on a binary relationship set only.

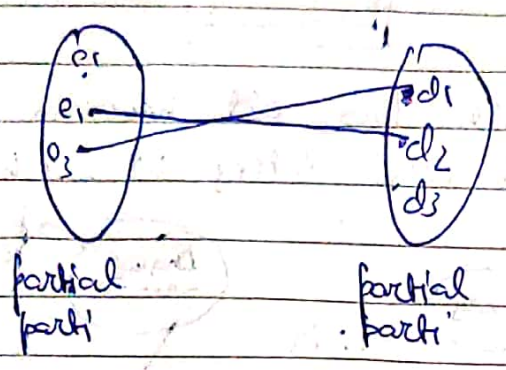


1. one to one (1:1)
2. one to many (1:M)
3. many to one (M:1)
4. many to many (M:N)

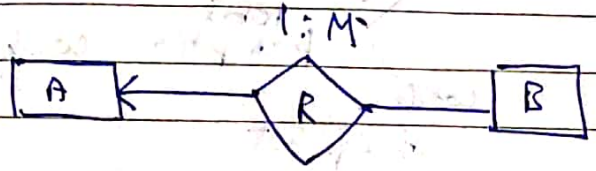
* One to one (1:1)



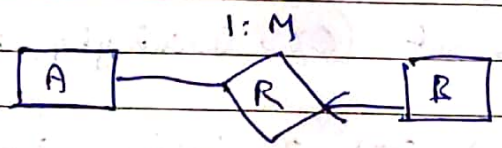
Scanned by CamScanner



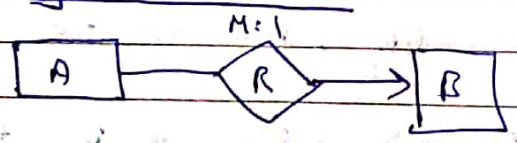
* <u>one to many (1 : M)</u>



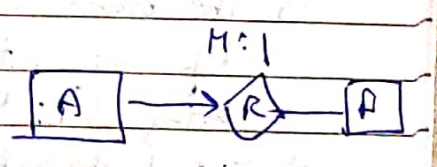
or



* <u>many to one (M:1)</u>

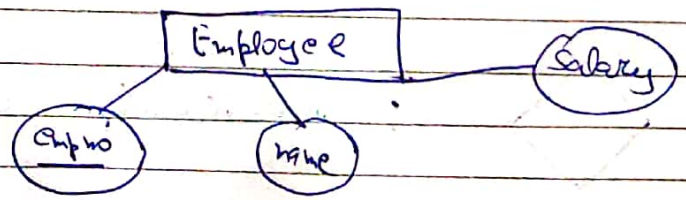


or



* <u>many to many (M : N)</u>

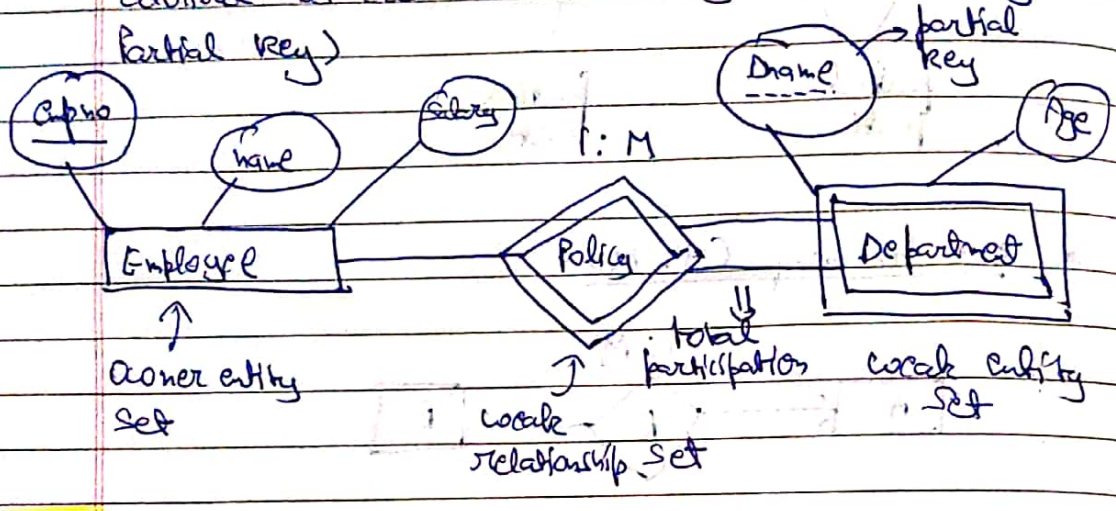
* **Strong entity set:** An entity set which has a (key attribute) key is called strong entity set.



Scanned by CamScanner

* **Weak entity set**

An entity set which does not have a key attribute is called weak entity set. (having partial key)

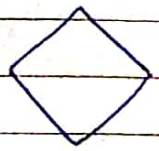


Note:)

- weak relationship set (or) Identifying relationship set
- owner entity set (or) Identifying owner
- The owner entity set to the weak relationship set, the cardinality ratio is one to many (1 : M)
- The participation of weak entity set to the Identifying relationship set is always total participation.
- The weak entity set is identified using partial key & key of owner entity set.

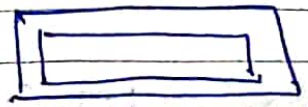
- **ER component (Top-down approach)**

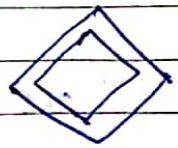
: Entity set



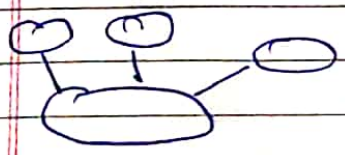
: relationship set

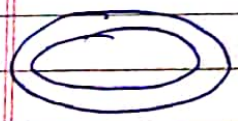
Scanned by CamScanner

 : weak entity set

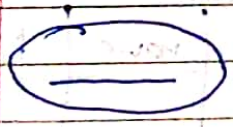
 : weak relationship set / identifying relationship set

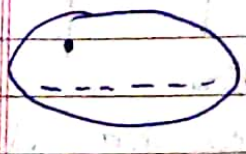
: attribute

 : composite attribute

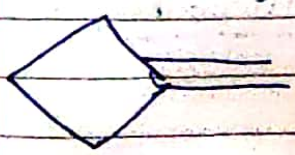
 : multivalued attribute

: derived attribute

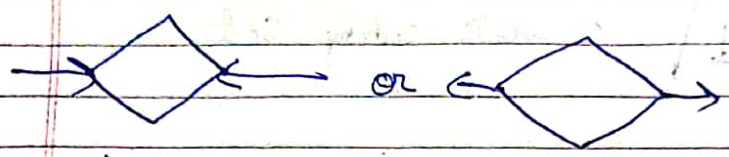
 : ~~p~~ key attribute

 : partial key

: descriptive attribute

 : total participation

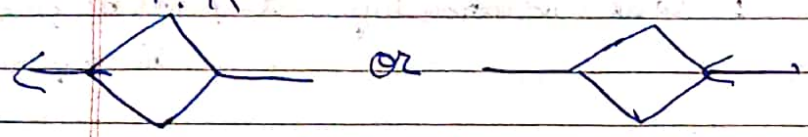
Scanned by CamScanner



or

1 : 1 (one to one)

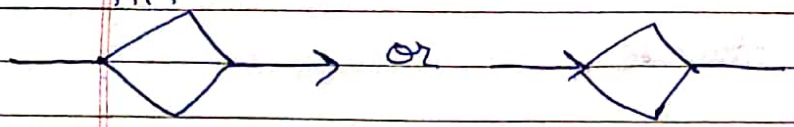
1 : M



or

(one to many)

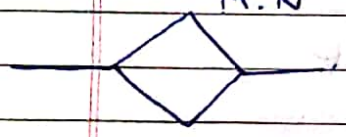
M : 1



or

(many to one)

M : N

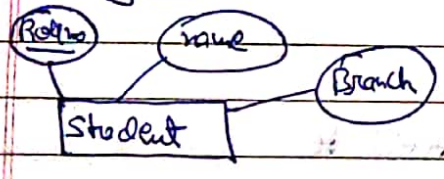


many to many

6) ER to Relational model

(1)

Entity Set



RDBMS

STudent

Rollno	name	Branch

→ Entity set is mapped with a relation

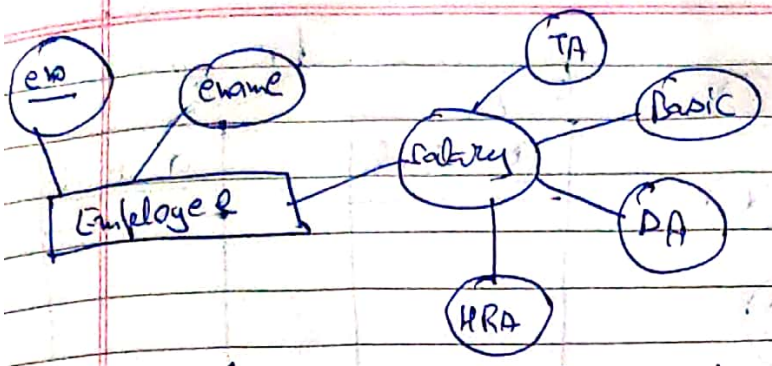
→ The attribute of the relation includes the attribute of an entity

→ Key attribute of an entity become primary key of the relation.

(2) Entity Set with composite attribute

→

Scanned by CamScanner



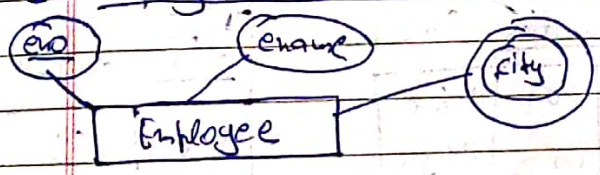
RDBMS

Employee

eno.	ename	TA	Basic	DA	HRA

•) The attribute of relation includes the simple attribute of an entity set.

③ Entity set with multivalued attribute



RDBMS f.k

Employee

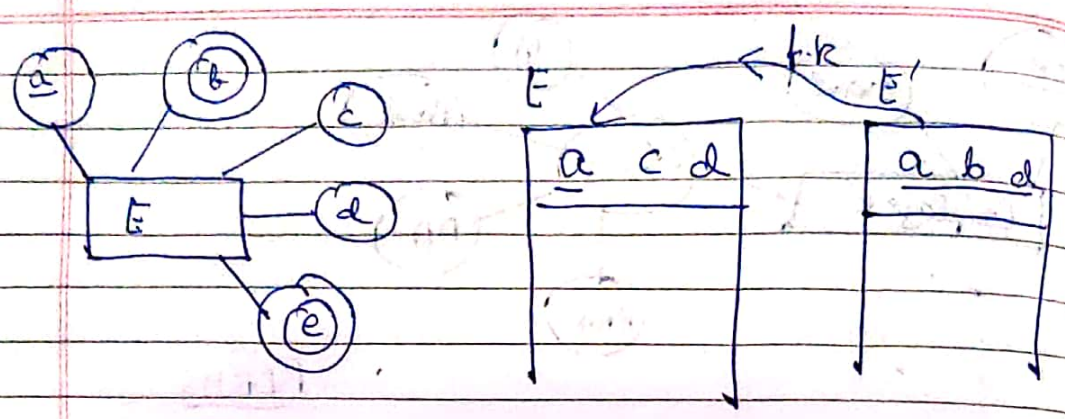
eno	ename
1	A
2	B

Employee'

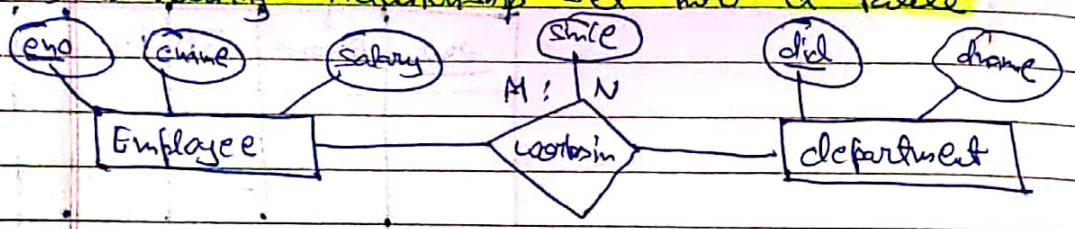
eno	City
1	Bang.
1	Noida
2	Delhi
2	Pune

•) If an entity contains multivalued attribute, then it is represented with two relation one with all simple attribute & other with key & all multivalued attribute.

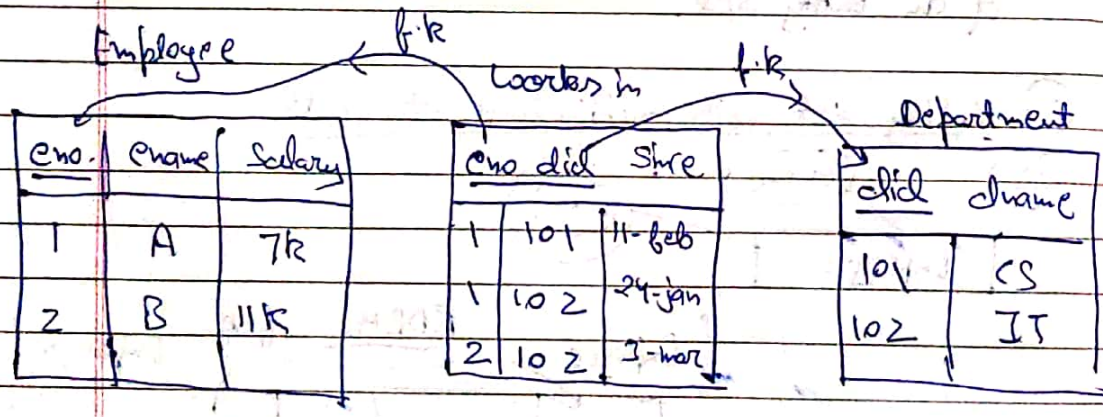
Scanned by CamScanner



o) Translating Relationship set into a table



(many to many)



eno.	ename	Salary
1	A	7k
2	B	11k

eno	did	Since
1	101	11-feb
1	102	24-jan
2	102	3-mar

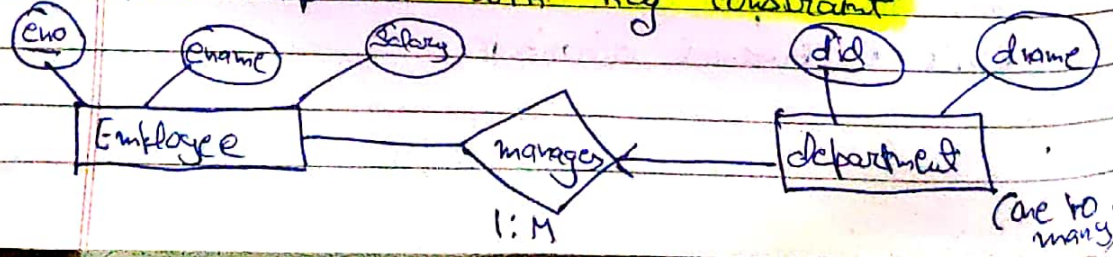
did	dname
101	CS
102	IT

o) A relationship set is mapped into a relation, the attribute of a relation includes:

→ The key attribute of the participating relation & are declared as foreign key to the respective relation.

→ descriptive attribute (if any).

o) Relationship set with key constraint



f.k

f.k

Employee

eno	ename	Salary
1	A	7k
2	B	11k

Manage

eno	did
1	101
1	102
2	103

Department

did	dname
101	CS
102	IT
103	EC

merge these tables

f.k

Employee

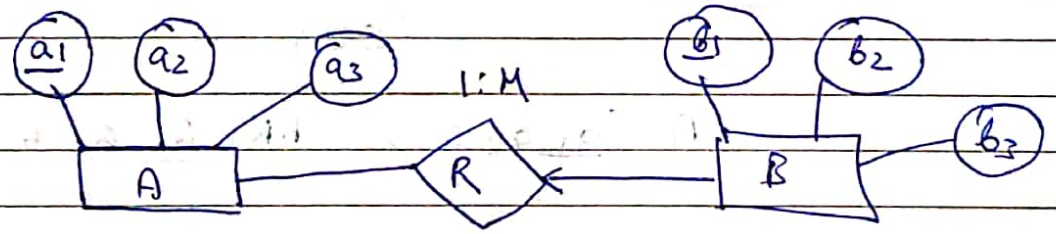
eno	ename	Salary
1	A	7k
2	B	11k

Department

did	dname	eno
101	CS	1
102	IT	1
103	EC	2

Note: merge the table at many side.

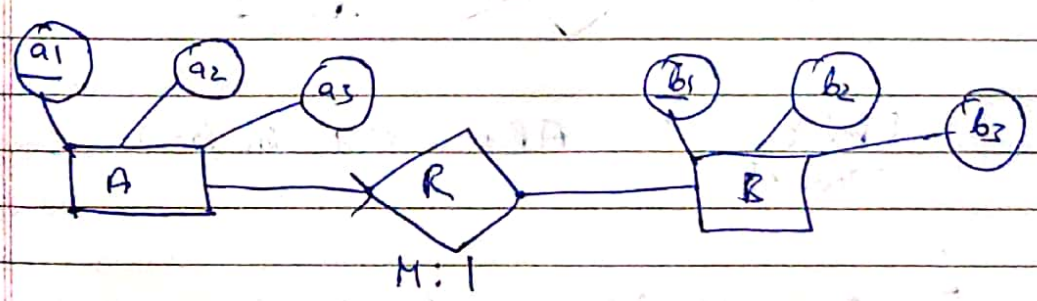
1



2 table required : ① A ($\underline{a_1}$ a_2 a_3)

② BR ($\underline{b_1}$ b_2 b_3 a_1) → foreign key

2



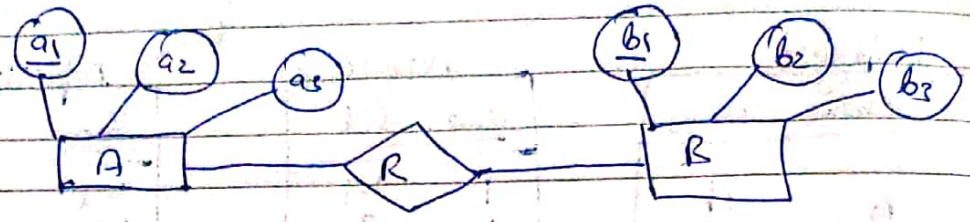
2 table required : ① AB ($\underline{a_1}$ a_2 a_3 b_1)

f.k

② B ($\underline{b_1}$ b_2 b_3)

Scanned by CamScanner

③



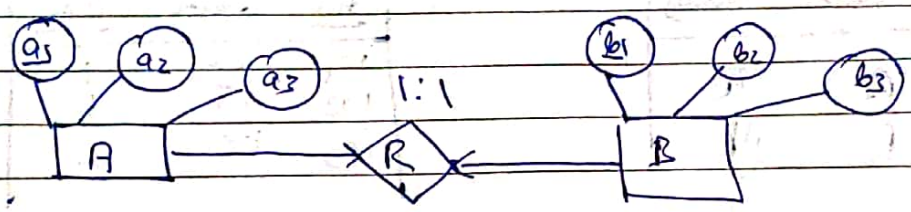
M:N

3 table required : ① A ($\underline{a_1}$ a_2 a_3)

② R (a_1 b_1) — f.k

③ B ($\underline{b_1}$ b_2 b_3)

④

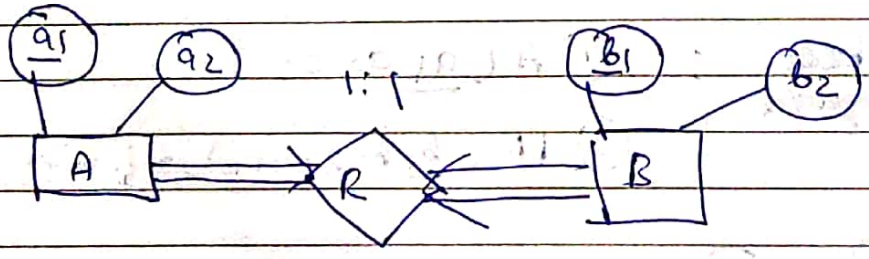


2 table required : AR ($\underline{a_1}$ a_2 a_3 b_1) — f.k B ($\underline{b_1}$ b_2 b_3)

(or)

A ($\underline{a_1}$ a_2 a_3) BR ($\underline{b_1}$ b_2 b_3 a_1) — f.k

⑤

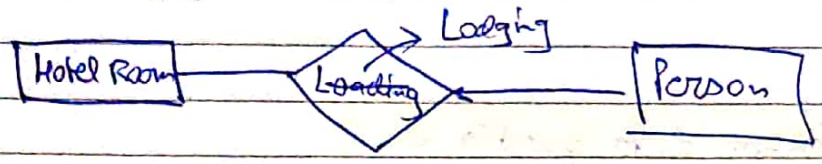


1 table required : ARB (a_1 a_2 b_1 b_2)

Note: If there is key constraint from both the side of entity set with total participation then can represent that binary relationship using single table.

Scanned by CamScanner

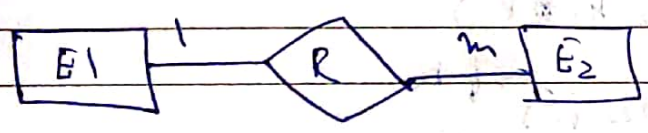
Gate 2005



If we wish to store info about the rent payment to be made by person(s) occupying diff hotel rooms, then this info should appear as attribute of

a) Person
b) Hotel Room
c) Lodging
d) None of these

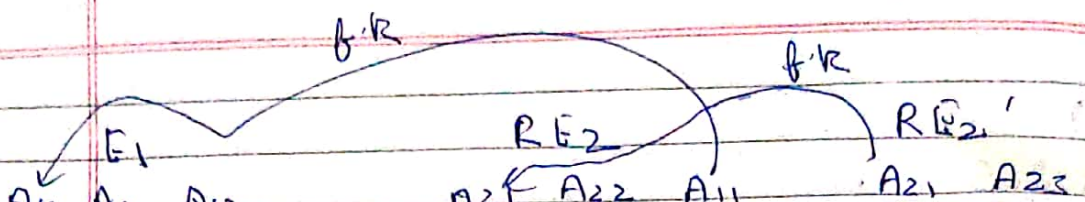
Gate 2004



The attributes of E1 are A11, A12, A13 where A11 is key attr,

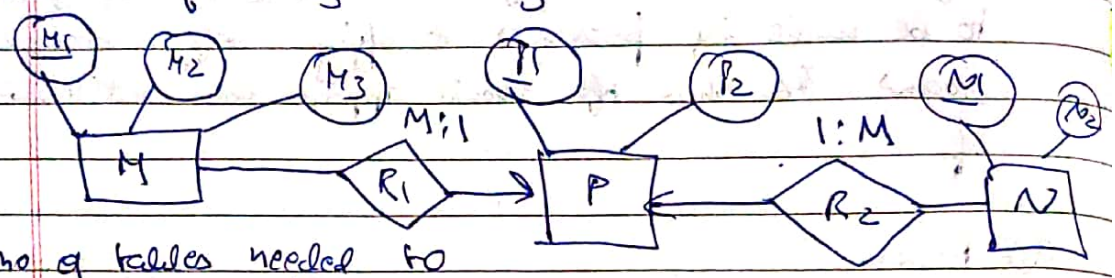
" " " E2 " A21, A22, A23 where A21 " & A23 is multivalued attribute. Relation R does not have any attribute. A relational db containing min no of tables with each table satisfying the 3NF is designed from the above ERD. The no. of tables in db is

a) 2
b) 3
c) 5
d) 4



Gate Common data questio 2008

Consider the following ER diag



Min no of tables needed to represent M, N, P, R1, R2 is

a) 2

~~b)~~ 3

c) 4

d) 5

MR1 (M1 M2 M3 P1) → f.k

R2N (N1 N2 P1) → f.k

P (P1 P2)

Which of the following is a correct attribute set for one of the table for the

~~a)~~ {M1, M2, M3, P1}

b) {M1, P1, N1, N2}

c) {M1, P1, N1}

d) {M1, P1}

Gate 2012

Given the basic ER and relational model, which of the following is incorrect?

a) An attribute of entity can have more than one value.

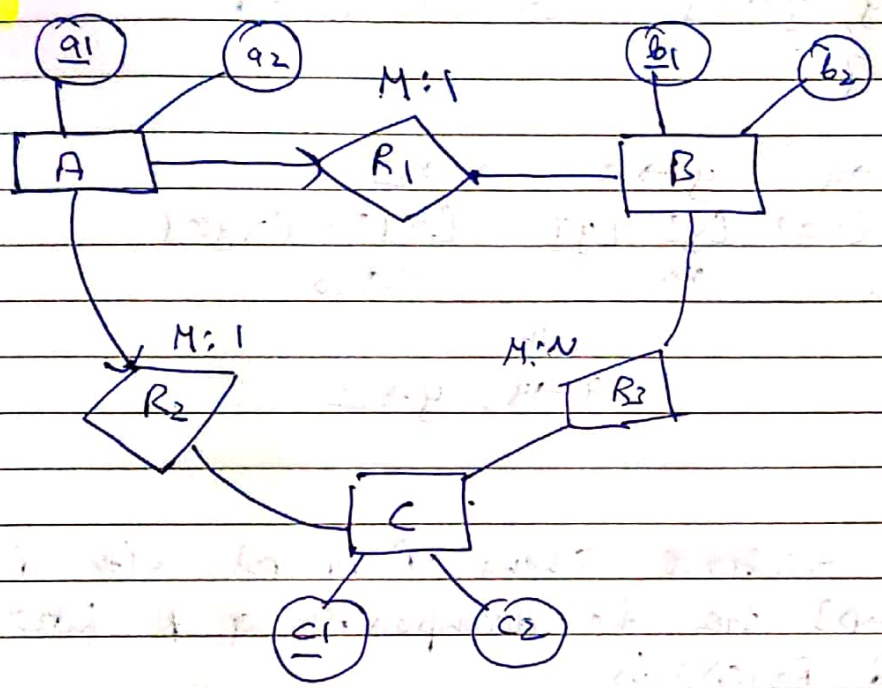
Scanned by CamScanner

b) an attribute of entity can be composite

c) In a row of relational table, an attribute can have more than one value

d) In a row of relation table, " " " have exactly one value or null value

Question



AR_1 ($\underline{a_1}$ a_2 b_1) → f.k

AR_2 ($\underline{a_1}$ a_2 c_1) → f.k

} merge ⇒ ~~ARTA~~

B ($\underline{b_1}$ b_2)

C ($\underline{c_1}$ c_2)

R_3 ($\underline{b_1}$ $\underline{c_1}$)

⇓

AR_1R_2 ($\underline{a_1}$ a_2 b_1 c_1) → f.k → f.k

B ($\underline{b_1}$ b_2)

C ($\underline{c_1}$ c_2)

R_3 ($\underline{b_1}$ $\underline{c_1}$)

f.k → f.k

(4 table required)

Scanned by CamScanner

Question

R(xyz) FD : { x → y, y → z, x → z }
which of the following is minimal cover of relation R?

a) [x → y, x → z]
b) [y → z, x → z]
c) [x → y, y → z] , G (so G = R)
d) none of these

x → y	y → z	x → z
$[x]^+ = (xz)$	$[y]^+ = [y]$	$[x]^+ = (xyz]$
needed	needed	extra

⇓

x → y, y → z

Question

Consider a relational schema R(ABCD) with FD's [A → B, C → D] then the decomposition of R into $R_1(AB)$ & $R_2(CD)$ is

a) lossless join & dependency preserving
b) lossy join & ''
c) lossless join & '' not ''
d) lossy join & '' not ''

$R_1(AB) \bowtie R_2(CD) \Rightarrow$ ~~R(ABCD)~~
no common attribute
lossy join
&
dependency preserved

Question

R(ABCDE) FD's [AB → CD, ABC → E, C → E]
what is normal form of R?

Scanned by CamScanner

a) 1NF
b) 2NF
c) 3NF
d) BCNF

$AB \to CD$, $ABC \to E$, $C \to E$

$[AB]^+ = [ABCDE]$

C.K : AB,

P.A {A, B}

NPA {C, D, E}

no. (proper subset of C.K → non key attr.) in 2NF

$C \to E$

non key attr. → non key attr

not in 3NF

Question

R(ABCD) FD: [$AB \to CD$, $D \to A$]

a) 1NF
b) 2NF
c) 3NF
d) BCNF

C.K $[AB]^+ = [ABCD]$
$[DB]^+ = [DBAC]$

C.K : AB, DB

PAK : A, B, D

NPAK : C

in 2NF
in 3NF
~~in BCNF~~ not in BCNF
$D \to A$

Scanned by CamScanner

Question

R(ABCDE) F: [A→CE, B→D, AE→D]

a) 1NF

b) 2NF

c) 3NF

d) BCNF

$[AB]^+ = [ABCDE]$

C.K : AB

PA : A, B

NPA : C, D, E

A → CE

proper subset of C.K

non key attribute

not in 2NF

Scanned by CamScanner