DBMS

A · ANI

Date

1

- → E-R model
- → Relational model
- → functional Dependancy (F.D), Normalization
- → SQL
- → Relational Algebra, Relational Calculus
- → Transaction Management, Concurrency Control
- → File organization, Index

:- Abstract in nature (Row fact)

nformation :- Data with added meaning.

Record :- Collection of logically related data

ex:-

< 501 Raj 530 >

Database :- Collection of similar records.

(OR)

Collections of logically related data.

Management :- Through set of programs

DBMS :- Collections of logically related data and set of programs to access those data.

Applications :-

- Banking
- Telecommunications
- Reservation systems
- Sales
- Scientific applications

Goal of DBMS :- Effective storage and retrieval of Data from DBMS.

D·S	Database	DBMS	
Tree	Hierarchial DB	HDBMS	outdated
Graph	Network DB	NDBMS.	outdated
Table	Relational DB	RDBMS ✓	
Objects	Objectoriented DB	OODBMS	
Object/Table	Object relational DB	ORDBMS	

Conceptual Database Design using (Entity-Relationship (ER) Model

Components of E-R Model

1) Entity : An object in the realword.
"Noun"

ex:- Student Book Account

$\underbrace{\text{Student} \quad \text{Book}}_{\text{physical entity}}$ $\underbrace{\text{Account}}_{\text{logical entity}}$

2) Entity set :- Collection of similar entities

$\boxed{\text{Student}}$ ← Rectangle

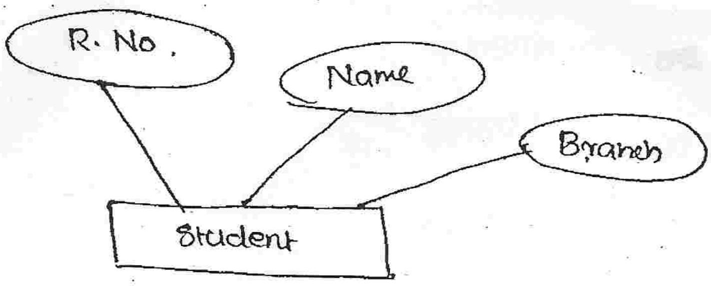
3) Relationship :- Association among the entities
"Verbs"

ex:- Reading
Buying

4) Relationship set :- Collection of similar relationships.

◇ Read ← Diamond.

5) Attributes :- which describes an entity

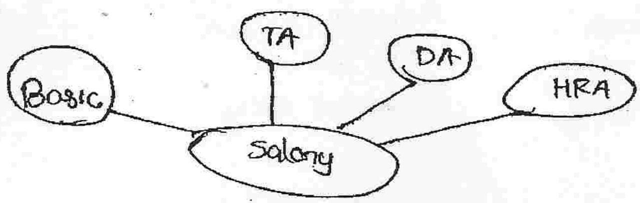


← Ellipse/oval

classification of Attributes

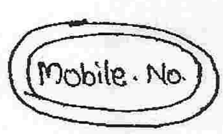
1) Simple attribute :- Which can-not be divided further

2) Composite attribute :- which can be divided further.



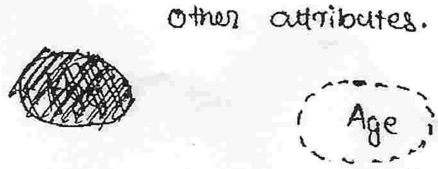
3) Single Valued attribute :- which takes one value per an entity

4) Multivalued attribute :- which takes more than one value per an entity

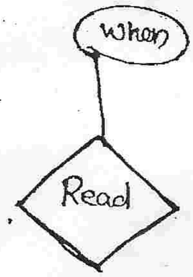


5) Stored attribute :- which does not require any updation

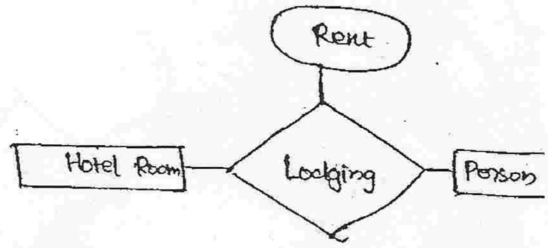
6) Derived attribute :- The Value of an attribute can be derived from other attributes.



7) Descriptive attribute :- which gives information about the relationship set

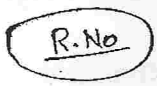


ex:-



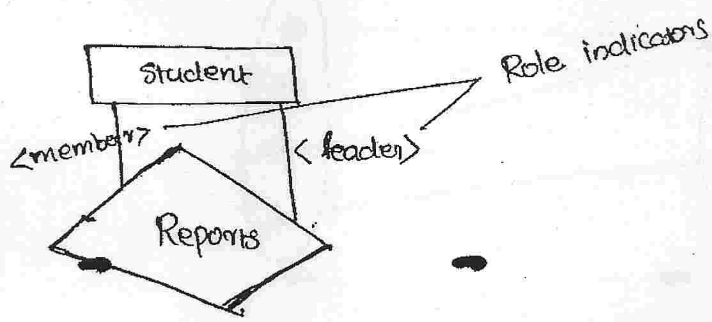
G-2003

8) Key attribute :- which uniquely identifies an entity in the entity set.

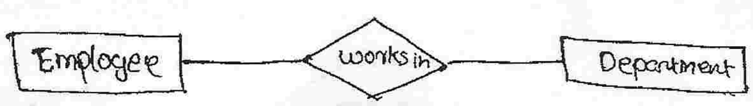


5 Degree of relationship set :- Specifies the no. of entity sets participates in a relationship set.

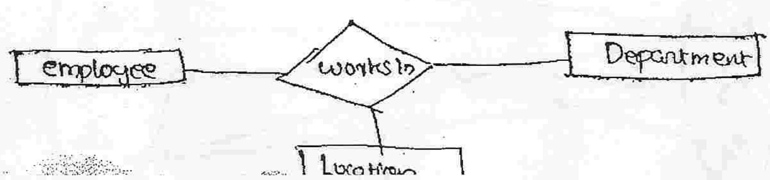
1) Unary :- Relationship among ~~one entity~~ two entities of the same entity set. (Recursive relationship set)



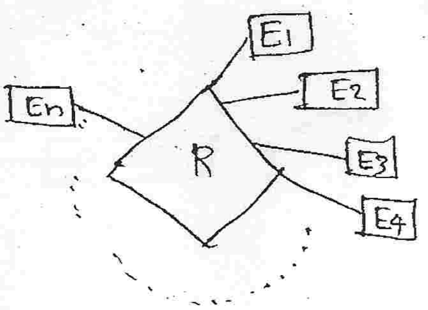
2) Binary Relationship set :- The relationship among two entity sets.



3) Ternary relationship :- Relationship among three entity sets.

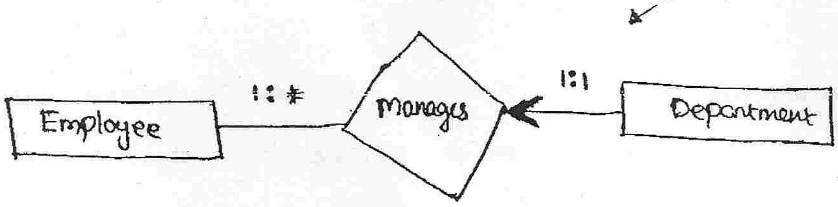


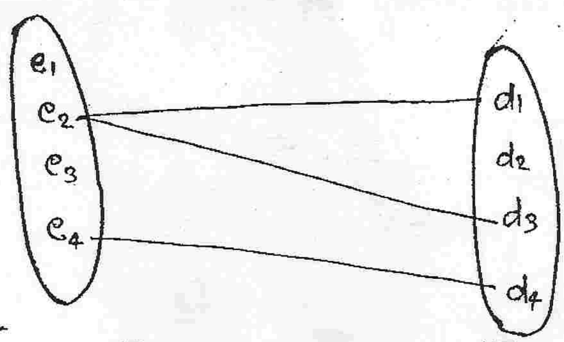
4) n-ary :- A Relationship among n- entity sets



1] Key Constraint :- An entity is acting as a key to another entity through the relationship set. it is denoted in E-R model using as Arrow

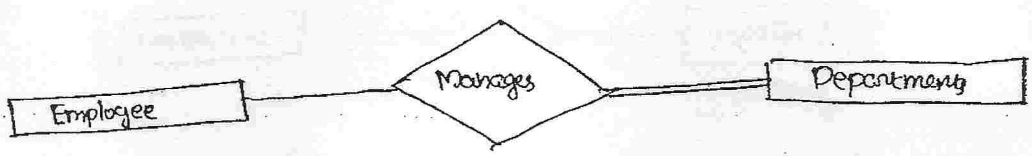
" Each department is managed by atmost one employee ".

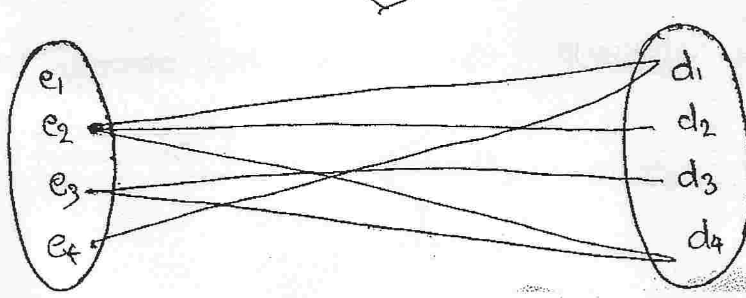




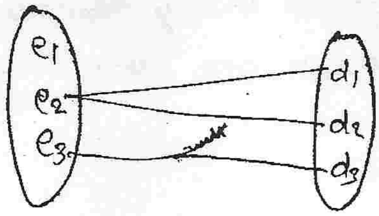
3] Participation Constraint :- If every entity in the entityset participates in a relationship set is called total participation denoted by double line (thick line). otherwise it is called partial participation. (Thin line or single line)

" Eeach department is managed by atleast one employee "





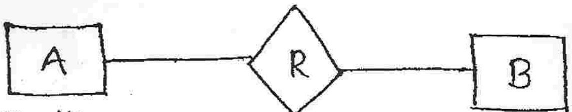
"Each Dept is managed by exactly one employee"



* Mapping Cardinality (Cardinality Ratios)

It Express the no of entities to which another entity can be associated via a relationship set.

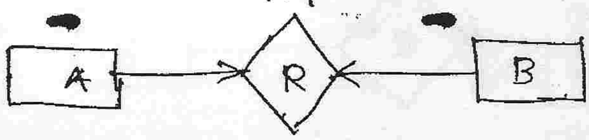
*(only on binary -relationshi



Note:- The cardinality ratios can be expressed on a binary relationship set only

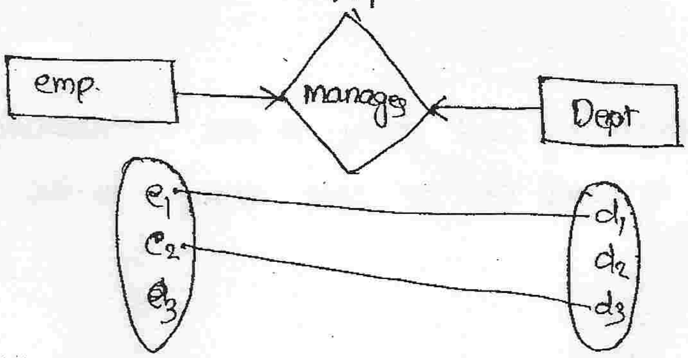
⊛ one to one (1:1)

1:1

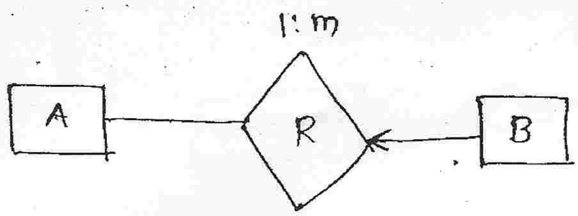


An entity in A is associated with atmost one entity in B and an entity in B is associated with atmost one entity in A.

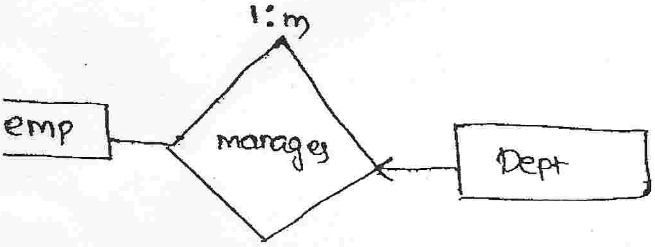
1:1

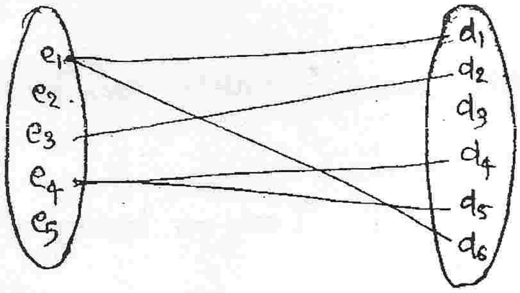


③ one-to-many (1:m)

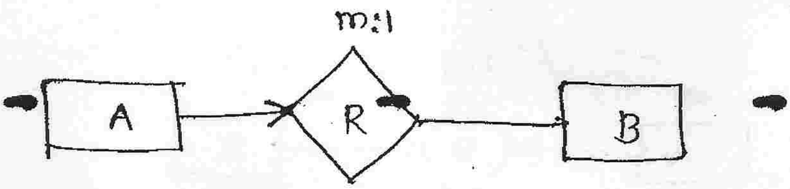


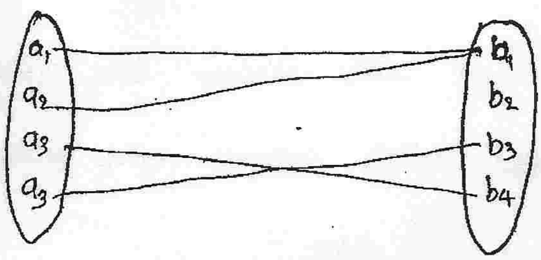
An entity in A is associated with zero or many entities in B and an entity in B is associated with atmost one entity in A.





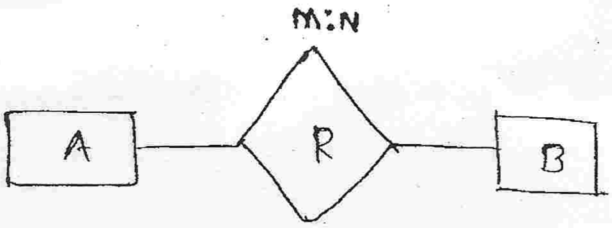
④ many-to-one (M:1)

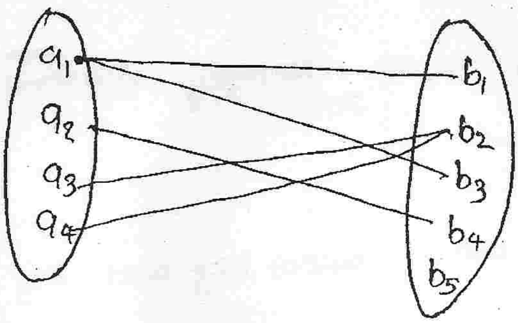




Note: In one-to-many and many-to one relationship set the key constraint is from an 'm' side entity to the relationship set.

(4) Many-to-many (M:N)





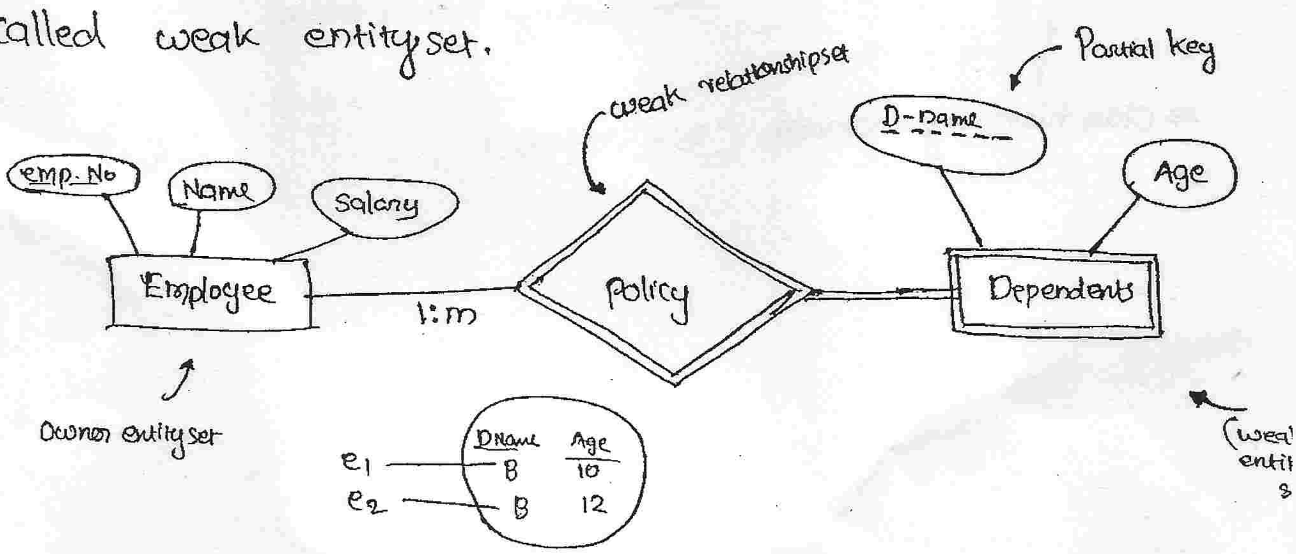
Strong entity set :-

An entity set which has a key is called strong entity set

Ex:-

Weak Entity set :-

- An entity set which does not have a key attribute is called weak entity set.



→ Partial key are descriminating attribute.

→ Weak relationship set (or) Identifying relationship set

→ Owner entity set (or) Identifying owner

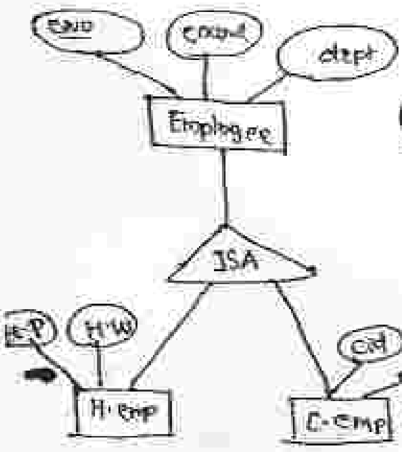
→ The owner entity set to the weak relationship set the cardinality ratio is one to many.

→ ~~The weak entity set to the~~

The participation of weak entity set to the identifying relationship set is always total.

→ The weak entity set is identified using partial key and key of the owner entity set

Class Hierarchy



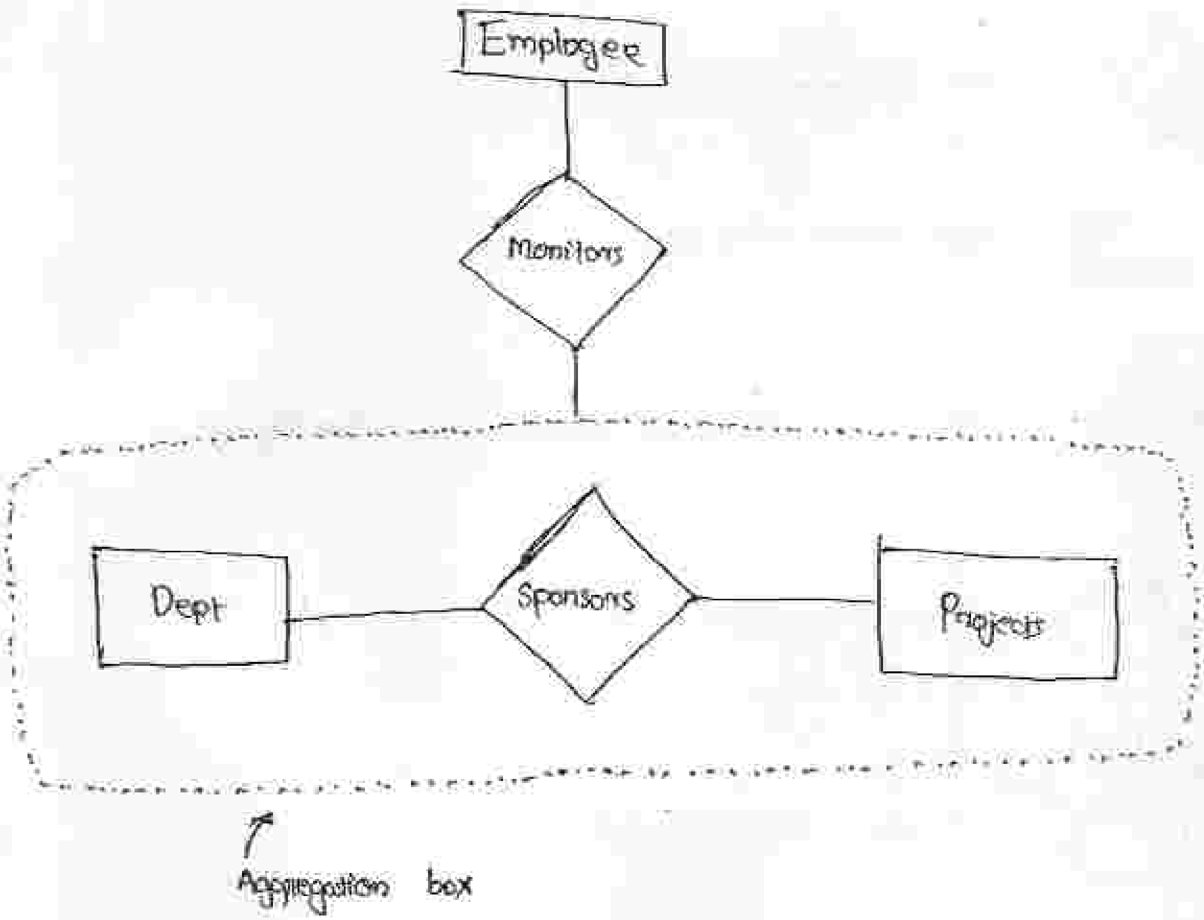
Generalization ← Grouping of entities

Specialization ← Dividing into sub entities

→ Class hierarchy notation (ISA - triangle)

Aggregation

Aggregation allows us to indicate that a relationship sets participates in another relationship set.



Advantages of E-R Model

1) Easy to understand
2) It is an effective communication tool

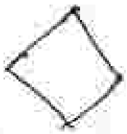
Disadvantages of E-R model

1) Limited constraint capability.
2) Loss of information content

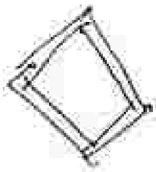
Approach : Top-down approach

E-R- Components

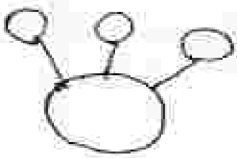

← entity set

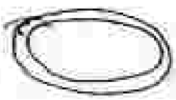

← Relationship set


← weak entity set


← weak ~~entity~~ relationship set (Identifying relationship set)


← attribute

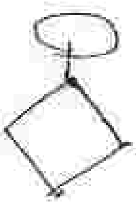

← Composite attribute

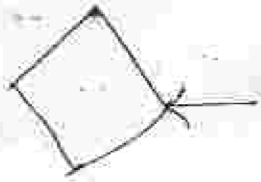

← multivalued attribute


← derived attribute

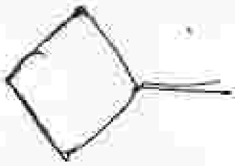

← Key attribute


← Partial key or descriminating attribute

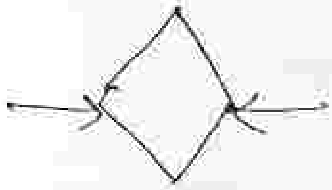

← Descriptive attribute



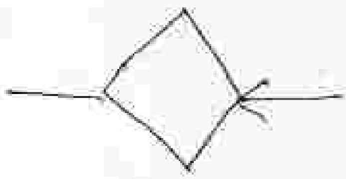
← Key constraint



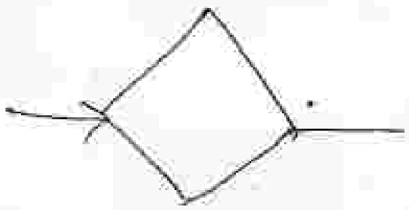
← total participation



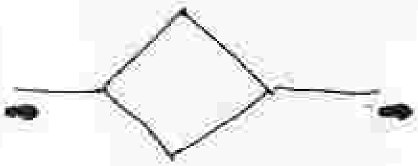
1:1 (one to one)



1:m (one to many)



m:1 (many to one)



many to many (M:N)

← class hierarchy

← aggregation box

Logical Database Design using "relational Model"

Relation :- → Rows & Colums
(Table) · Records · Attributes
Tuples

described using
- Relation Schema :- Structure
- Relation instance :- tuples

Student

Rno Number(2) Primary key	Name char(10) Not null	Branch char(3)
1	A	CSE
2	B	IT

Degree of a relation :-

Degree specifies No. of columns present in a tuples (3)

Cardinality of a relation :-

specifies no. of rows (2)

* **RDBMS :-** Collection of relations

Integrity Constraints :-

Is a condition specified on a database schema and restricts the data that can be stored in an instance of the database.

ex:- PRIMARY KEY, NOT NULL, UNIQUE

Legal Instance :- The instance which satisfies all the Integrity constraints specified on a database schema.

→ Otherwise such an instance is called Illegal Instance

Key Constraints

It is a set of fields of a relation has a unique identifier for a touple. That is each touple in a relation is identified using a set of attributes.

Student (RNo, Name, father, Branch, Passport)

1) RNo ← Key

2) (Name, father) ← key

A	P
A	Q
B	Q

3) Passport ← key

4) (RNo, Name) ← key

5) (RNo, Passport) ← key

I) Candidate key :-

It is a minimal set of attributes which uniquely identifies a touple in a relation

ex:- RNo , (Name, father) , Passport

II) Super key :-

It is a set of attributes which contains a key (candidate key)

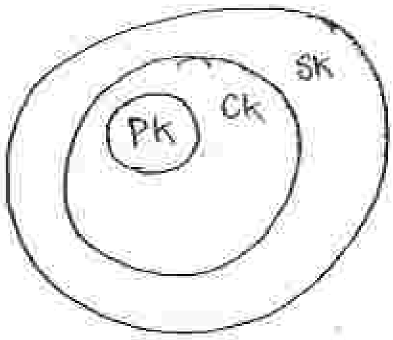
ex:- (RNo, Passport) , (RNo, Name), RNo, Passport , (Name, f

→ Every candidate key is called a super key, but every super key need not be a candidate key

III) Primary key :-

Among all the available candidate key one can be identified as primary key.

ex:- Rno.



IV) Foreign key constraint (Referential Integrity Contraint)

Student

Rno	Name	Branch
1	A	CSE
2	B	IT
3	C	CSE

(Referenced relation) (Parent)

Registration FK

C.No	CName	Rno
101	DBMS	1
102	CD	1
103	TOC	3
104	PL	

(Referencing relation) (Child)

→ The values present in foreign key must be present in primary key of referenced relation. Foreign key may contain duplicates and null values.

* Parent table

✓ Insert <4 D ECE>

X Delete <1 A CSE>

Child table

X Insert <105 GT 5>

✓ Delete <103 TOC 3>

→ Deletion from the referenced relation and insertion into the referencing relation may violate foreign key constraint.

foreign key with ondelete cascade

when data from the parent table is deleted. The related data from the child table also to be deleted cascadelly. (both tables)

→ A Relation can act as both parent and child. ie, a relation may contain a primary key and a foreign key that refers to the same relation.

Qu Consider a relation R(A,B) where A is primary key and B is foreign key referencing the same relation. then which of the following row sequence can be inserted sucessfully 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

Qu Consider a relation R(A,B) where A is a primary key and B is a foreign key referencing A with on-delete cascade consider the following relation instance of R.

R (A B) with odc

A	B
2	4
3	4
4	3
5	2
7	2
9	5
6	4

what are the tuples that must be additionally deleted to preserve the referential integrity constraint when the tuple 2,4 deleted is .

a) (3,4) (4,3)

b) (3,4) (4,3) (6,4)

c) (5,2) (7,2)

d) (5,2) (7,2) (9,5)

First which tuple is deleted ?

a) (5,2)

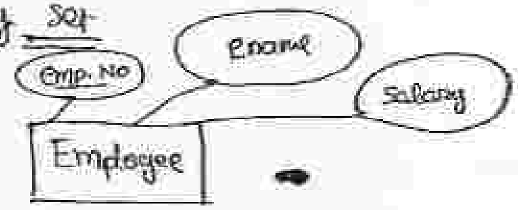
b) (7,2)

c) (9,5)

d) all at once .

E-R to Relational Model

i) Entity set

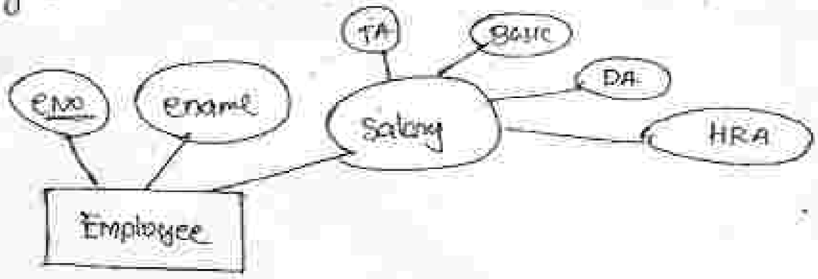


Employee

eno	ename	Salary

- An entity set is mapped with a relation
- The attributes of the relation include the attributes of an entity
- The key attribute of an entity becomes primary key of the relation.

2) Entity set with composite attribute



Employee

eNo	ename	Basic	TA	~~Basic~~ DA	HRA
1	A	5000	3000	5000	1200

The attributes of a relation includes the simple attribute of an entity set

3) Entity set with multivalued attribute

Employee

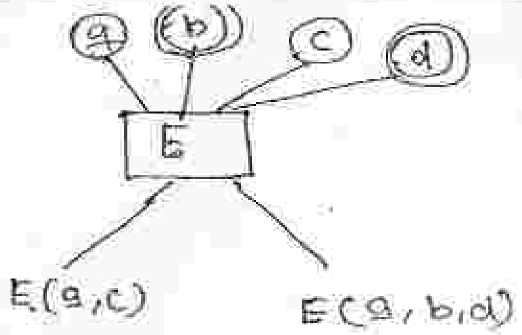
e.no	ename
1	A
2	B

FK

Emp-address

e.No	City
1	Hyd
1	Bang
2	Bang
2	Pune

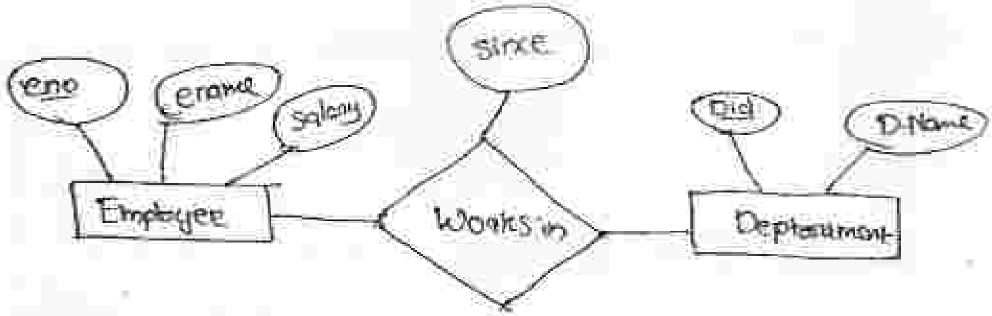
Note :- If an entity contains multivalued attributes it is represented wi' two relations one-with all simple attributes and the other with key and all multivalued attributes.



5 – 8.30 – Digital
9 – 1 – TOC
2 – 5.30 TOC
6 – 8.30 – DBMS
3 hr

04-12-19

Translating relationship set into a table



Employee

e-no	ename	Salary
1	A	5k
2	B	9k

FK

Works in

eno	Did	since
1	101	1-Jan
1	102	2-Feb
2	102	2-Feb

F.K

Department

Did	Dname
101	CSE
102	IT

A Relationship set is mapped into a relation, the attributes of a relation includes

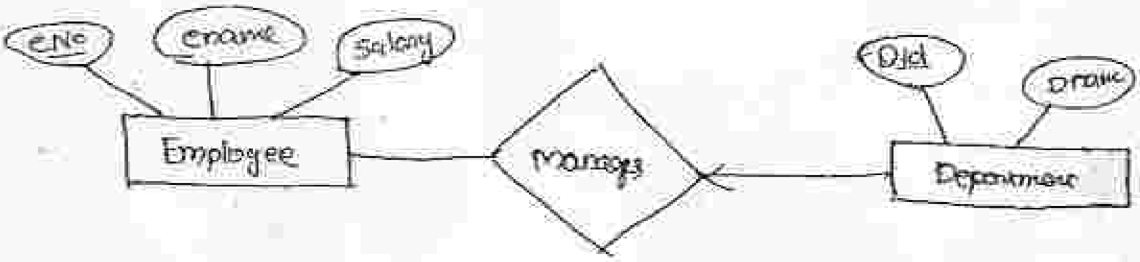
(•) The key attributes of the participating relation and are declared as foreign keys to the respective relation

(•) Descriptive attributes (if any)

(•) Set of Non descriptive attributes is the primary key of the relation.

Relationship set with key constraint

Eeach dept is required to have atmost one employee as a mana



Employee

eno	ename	Salary
1	A	5k
2	B	9k

FK

Manages

eno	Did
1	101
1	102
2	103

FK

Department

Did	Dname
101	CSE
102	IT
103	ECE

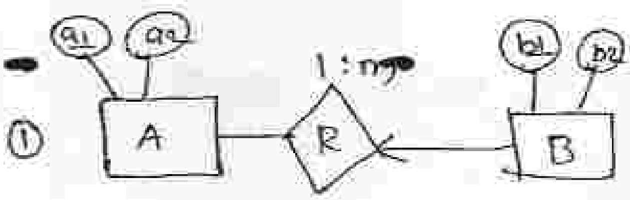
merge these tables

Foreign key

Department

Did	Dname	eno
101	CSE	1
102	IT	1
103	ECE	2

① 1:n



2-table A, BR

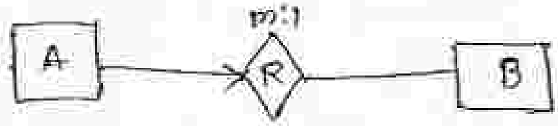
A(a_1, a_2), BR(b_1, b_2, a_1)

Fk

④ 1:1

2-tables AR, B or A, BR

② m:1

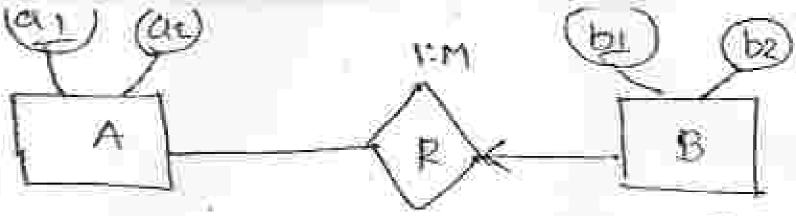


2-table AR, B

③ M:N

3-table A, R, B

①

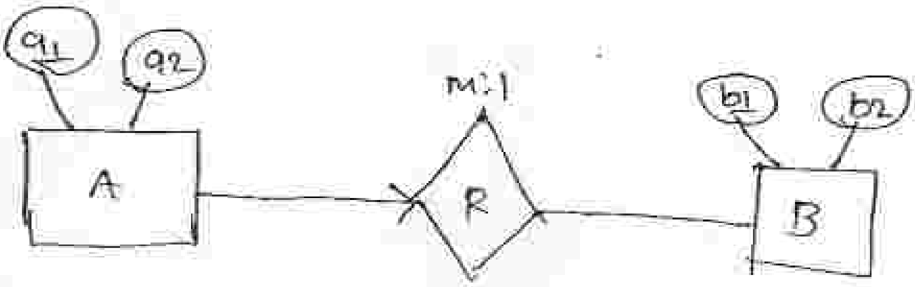


F.K

A(a1, a2) BR(b1, b2, a1)

2- tables

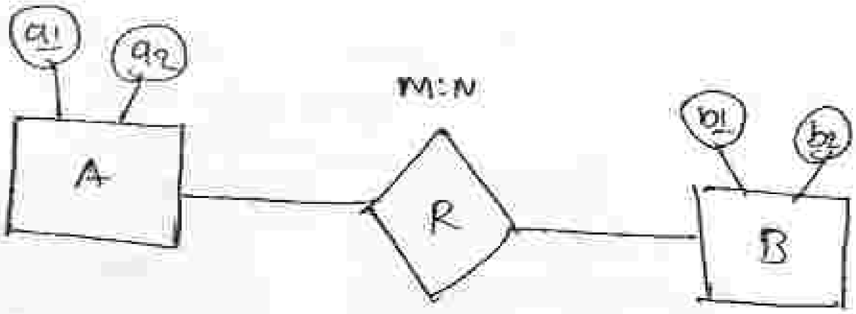
②



F.K

2- table   AR(a1, a2, b1), B(b1, b2)

③



3 - table   A(a1, a2) R(a1, b1) B(b1, b2)

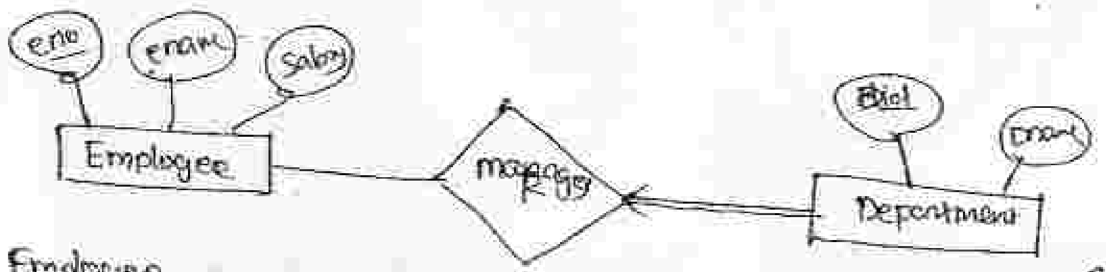
④

2- table.   AR(a1, a2, b1), B(b1, b2) (or)

A(a1, a2), BR(b1, b2, a1).

Relationship set with key constraint & Participation constraint.

" Each Dept is required to have exactly one employee as a manager"



Employee

eno	ename	salary
1	A	5K
2	B	9K

Dept - mngr

F.K, not null

total particip

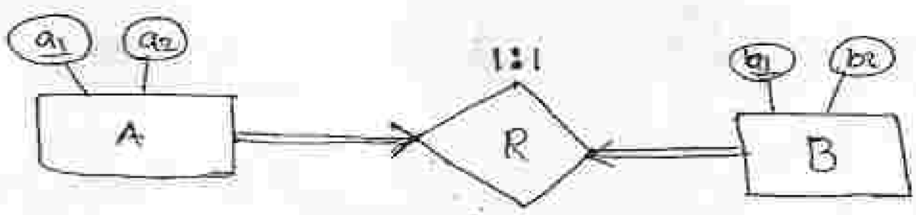
Did	Dname	eno
101	CSE	1
102	IT	2
103	ECE	2

1) If there is a key constraint merge the relationshipset table with an entity set table

2) If the entity set totally participating with relationship set then forienkey with not null constraint

---

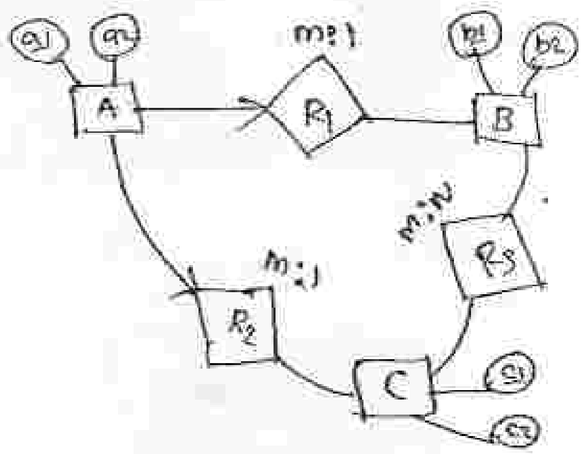
(5)



1 table. ARB ($\underline{a_1}, a_2, b_1, b_2$)

Note: If there is a key constraint from both the sides of an entity set with total participation then we represent that binary relationship using single table.

Qus



find the min. no. of tables that are possible when you translate the ~~below~~ above E-R diagram into relational model.

Ans

4- tables

1) $AR_1R_2(\underline{a_1}, a_2, b_1, c_1)$ — FK: b_1, c_1

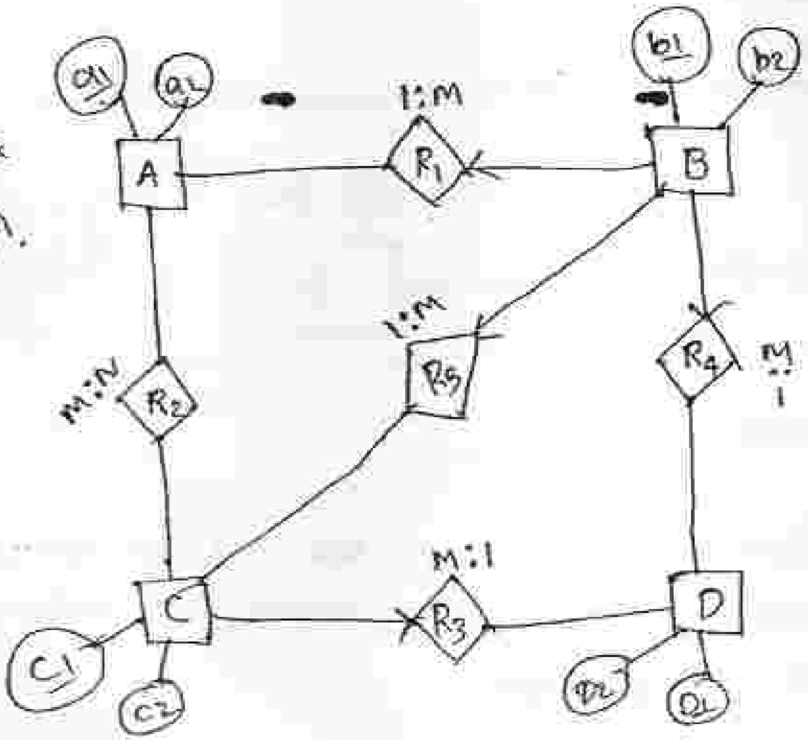
2) $B(\underline{b_1}, \underline{b_2})$

3) $R_3(\underline{b_1, c_1})$ → FK

4) $C(\underline{c_1}, c_2)$

Qus

find min. no of tables = ?



5- tables

1) $BR_1R_4R_5(\underline{b_1}, b_2, a_1, c_1, D_1)$ — FK

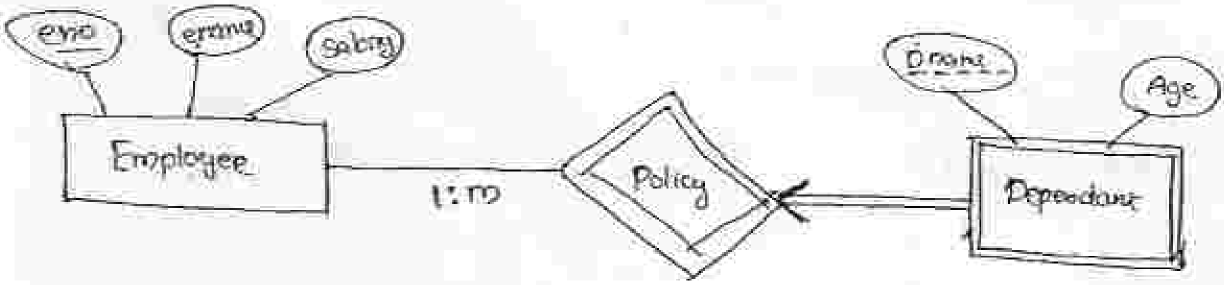
2) $A(\underline{a_1}, a_2)$

3) $R_2(\underline{a_1, c_1})$ — FK

4) $D(\underline{D_1}, D_2)$

5) $CR_3(\underline{c_1}, c_2, D_1)$ — F.K

Weak entity set



key constraint

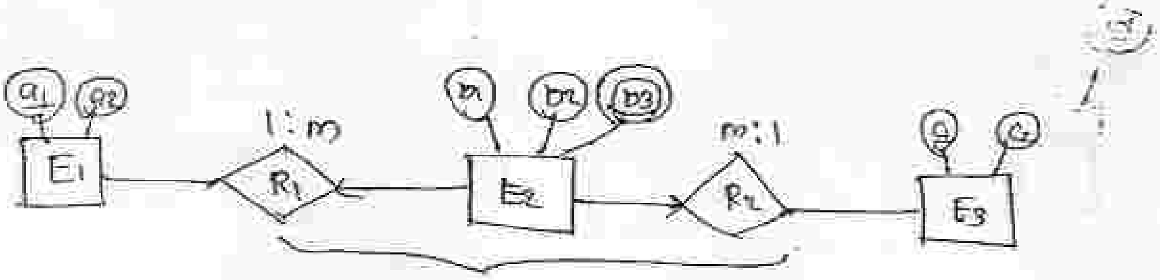
F.K
- on delete cascade

Depd - policy

eno	Dname	age

weak entity set

Que

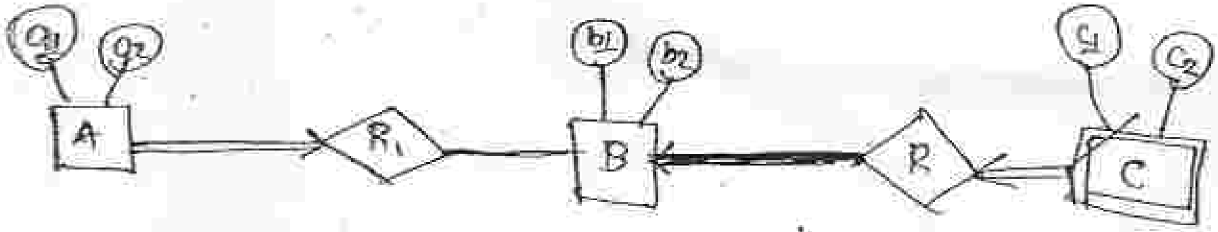


$E_1(a_1, a_2)$ $\quad$ $E_2(b_1, b_2, b_3, a_1, c_1)$ $\quad$ $E_3(c_1, c_2)$

$E_{21}(b_1, b_2, a_1, c_1)$ $\quad$ $E_{22}(b_1, b_3)$

Total no. of tables (min) = 3

Q. min. no. of tables = ?

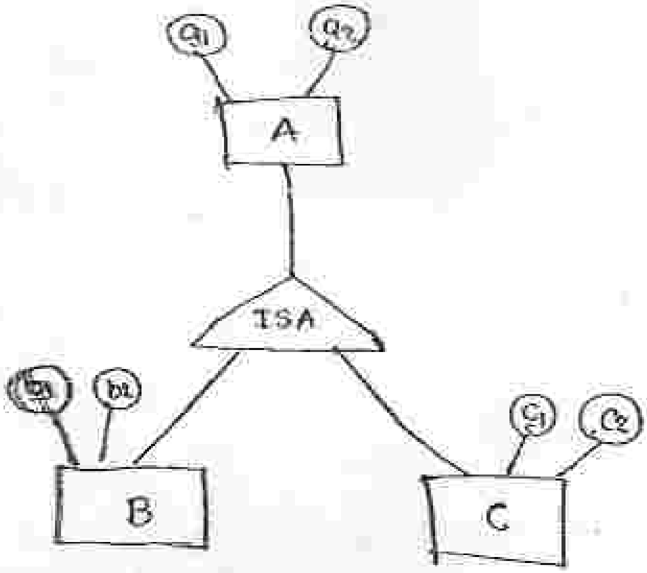


Fk, not null

$AR_1(\underline{a_1}, a_2, b_1)$ $B(\underline{b_1}, b_2)$ $CR_2(\underline{b_1, c_1}, c_2)$

Fk, ODC

Q.

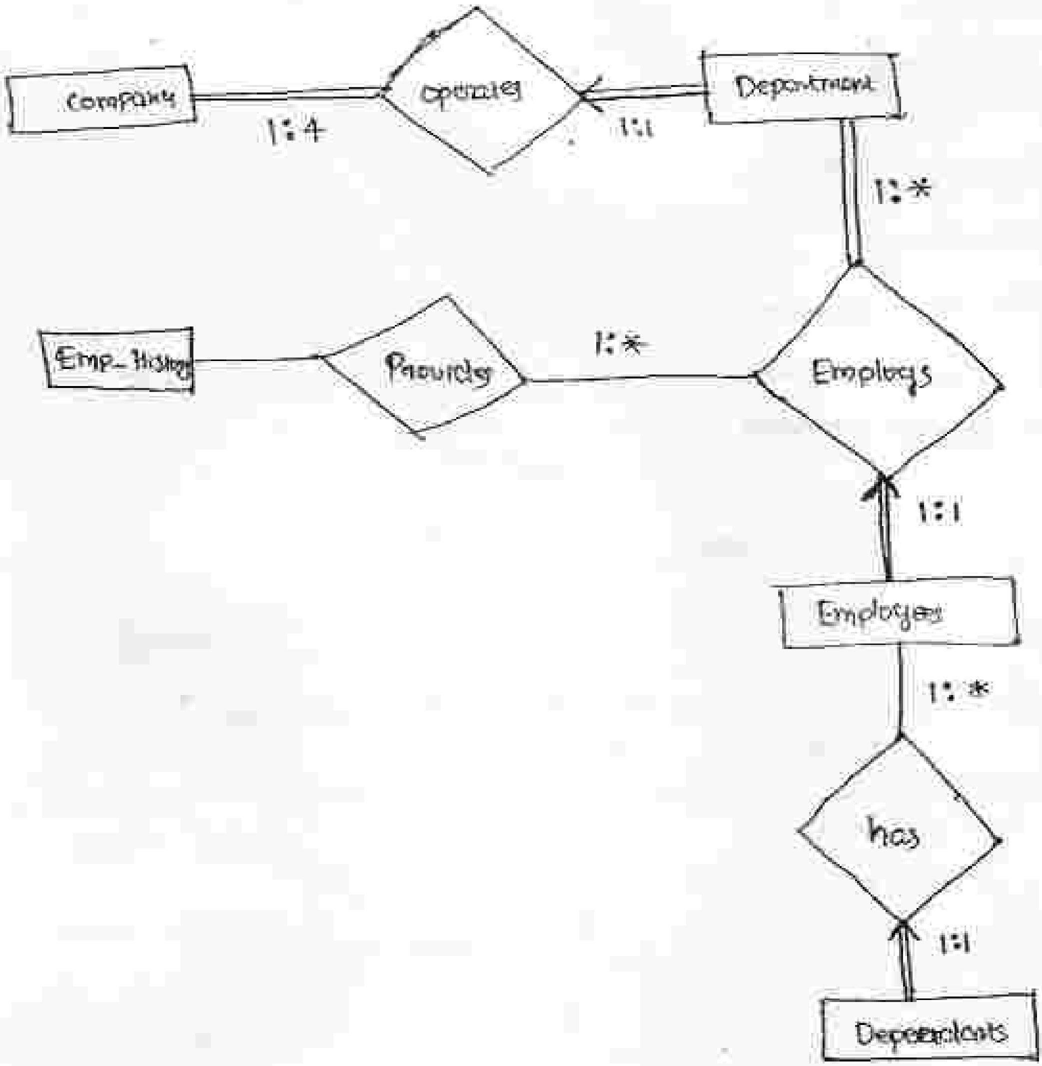


P-131

Level-2

Q (01)

(a)(b)



NORMALIZATION

Schema Refinement

- Redundancy problems
  1) Redundancy
  2) Insertion anomalies
  3) Update anomalies
  4) Deletion anomalies

Employee

functional dependency

eno	ename	grade	salary
1	P	A	~~9000~~ 9500
2	Q	A	~~9000~~ 9500
3	R	B	600
4	S	B	600
5	T	A	~~9000~~ 9500
6	R	A	9800 X

1) wastage of space
2) multiple updation/ insertion needed
3) Deletion causes loss of data.

→ Decomposition is the solution for these problems.

FK

Emp

eno	ename	grade
1	P	A
2	Q	A
3	R	B
4	S	B
5	T	A
6	R	A ✓

Emp_salary

Grade	Salary
A	9000 → 9500
B	600

All the problems arising due to redundancy is resolved using decomposition.

Functional Dependancy (F.D)

$$\underset{\text{determinant}}{X} \longrightarrow \underset{\text{dependant}}{Y}$$

X + Y can be one or many attributes

$X \to Y$

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

X	Y
x_1	y_1
x_2	y_3
x_3	y_1
x_4	y_4

→ The functional dependancy is the generalizziation of the concept of key.

→ $X \to Y$ says that if two tupples agree on the value of attribute X they must agree on the value in attribute Y.

Qn Consider a relation R(ABC) having the tuples

R(ABC)		
1	2	3
4	2	3
5	3	3

Then which of the following dependencies can you infer does not hold over R.

a) $A \to B$

b) $BC \to A$

c) $B \to C$

d) $AC \to B$

Qn Consider the following relation instance

X	Y	Z
1	4	3
1	5	3
4	6	3
8	2	2

which of the following functional dependencies are satisfied by the above instance ?

a) $XY \to Z$, $Z \to Y$

b) $\underbrace{XZ \to X}_{\text{Trivial}}$, $Y \to Z$

c) $XY \to Z$, $Z \to X$

d) $XZ \to Y$, $Y \to Z$

Classification of functional Dependency

1) Trivial functional Dependancies

2) Non- Trivial functional Dependancies.

1) Trivial F.D :-

A functional dependancy $X \rightarrow Y$ is said to be trivial iff $Y \subseteq X$. In other words if R.H.S of some dependancy is the subset of L.H.S of the dependancy then it is called trivial F.D

ex:-

$AB \rightarrow A$

$AB \rightarrow B$

$AB \rightarrow AB$

2) Non- Trivial F.D :-

If there is atleast one attribute in the R.H.S ie, not part of the L.H.S such dependancy is called non- trivial functional dependancy

ex:-

$AB \rightarrow BC$: Non- trivial

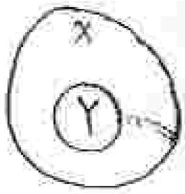
$AB \rightarrow CD$: Completely non- trivial

[6-8.30 - Digital
9-5.30 - DBMS
@ 312]

Closure properties of F.D's (Closure

1) Reflexivity

if $X \supseteq Y$ then $X \rightarrow Y$ is a dependancy



Ex:-
$AB \rightarrow A$

2) Augmentation

if $X \rightarrow Y$ is a dependancy $XZ \rightarrow YZ$ is a dependancy.

3) Transitivity

if $X \rightarrow Y$ is a dependancy and $Y \rightarrow Z$ is a dependancy then $X \rightarrow Z$ is a dependancy

4) Union property

if $X \rightarrow Y$ is a dependancy and $X \rightarrow Z$ is a dependacy. then $X \rightarrow YZ$ is a dependancy

5) Decomposition

if $X \rightarrow YZ$ is a dependancy then $X \rightarrow Y$ and $X \rightarrow Z$ is a dependancy.

Closure set of F.D's (F^+)

It is the set of all F.D'S that can be determined using the given set of functional dependancies 'F.' and is denoted by F^+ (F-closure)

Ex:-

$R(ABC)$

$F: \{A \to B,\ B \to C\}$

$F^+: \{A \to B,\ B \to C,\ A \to C,\ AC \to BC,\ AB \to BC,\ AB \to AC \ \ldots \}$

Que: find the no. of F.D's in F-closure (F^+) for a relation with 2-attributes ?

Ans:

$R(A, B)$

X	$\to$	Y
ϕ		ϕ
A		A
B		B
AB		AB

$4 \times 4 = 16$

$\phi \to \phi$	$A \to \phi$	$B \to \phi$	$AB \to \phi$
$\phi \to A$	$A \to A$	$B \to A$	$AB \to A$
$\phi \to B$	$A \to B$	$B \to B$	$AB \to B$
$\phi \to AB$	$A \to AB$	$B \to AB$	$AB \to AB$

F^+ #dependancies = 16

Qu Consider $R(A,B,C)$

$F^+ = 64$ combinations

$X \rightarrow Y$

$2^3 \times 2^3 = 64$

Qu R (n-attributes)

$F^+ = 2^n \rightarrow 2^n$

$2^n \times 2^n = 2^{2n}$

Closure set of attributes (X^+)

Ex:- $R(A B C)$

$F: \{A \rightarrow B,\ B \rightarrow C\}$ $\quad A^+ = \{A, B, C\}$

The set of all attributes that can be determined using the given set X of attributes is called attribute closure and is denoted by X^+

Qu $R(A, B, C, D, E, F)$

$F: \{AB \rightarrow C,\ BC \rightarrow AD,\ D \rightarrow E,\ CE \rightarrow B\}$

find $(AB)^+ = ?$

Ans $AB^+ = \{A, B, C, D, E\}$ $\quad CE^+ = \{B, A, D, C, E\}$

$BC^+ = \{B, C, A, D, E\}$

$D^+ = \{D, E\}$

Qus Consider a relation with F.D's

$F: \{AB \to CD,\ AF \to D,\ DE \to F,\ C \to G,\ F \to E,\ G \to A\}$

find $CF^+ = \{C, F, G, E, A, D\}$

$BG^+ = \{B, G, A, C, D\}$

$AF^+ = \{A, F, D, E\}$

$AB^+ = \{A, B, C, D, G\}$

Qus Consider a relation $R(A, B)$ with $F: \{A \to B\}$

Find all dependencies in F^+

$F^+ = \{A \to B,\ AB \to B,\ A \to A,\ B \to B,\ A \to AB,\ \phi \to \phi,$
$A \to \phi,\ B \to \phi,\ AB \to \phi,\ AB \to AB,\ AB \to A\}$

$\phi^+ = \phi = 1$

$A^+ = A, B = 4$

$B^+ = B = 2$

$AB^+ = A, B = 4$

$\underline{\quad 11 \quad}$

Qus $R(A, B, C)$

$F: \{A \to B, B \to C\}$ find $F^+ = \_\_\_\_ ?$

$\phi^+ = \phi = 1$

$A^+ = A, B, C = 8$

$B^+ = B, C = 4$

$C^+ = C = 2$

$AB^+ = A, B, C = 8$

$AC^+ = A, B, C = 8$

$BC^+ = B, C = 4$

$ABC^+ = A, B, C = 8$

$\underline{\underline{43}}$

Q: $R(A, B, C)$

$F : \{ A \rightarrow B, B \rightarrow C, C \rightarrow A, B \rightarrow A \}$

$F^+ = \_\_\_ ?$

Ans: $\phi^+ = \phi = 1$

$A^+ = A, B, C = 8$

$B^+ = A, B, C = 8$

$C^+ = A, B, C = 8$

$AB^+ = A, B, C = 8$

$AC^+ = A, B, C = 8$

$BC^+ = A, B, C = 8$

$ABC^+ = A, B, C = 8$

$\underline{\underline{57}}$

Applications of attribute closure

1. To find additional functional dependancies
2. To find key's of a relation
3. To find equivalances of functional dependancies
4. To find minimal set of functional dependancies.

1] To find additional functional dependancies

Ex:- $R(A,B,C,D)$

$$f: \{A \rightarrow BC,\ B \rightarrow CD,\ D \rightarrow AB\}$$

then find $AD \rightarrow C$ is possible ?

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

Qu Consider a Relation $R(ABCDE)$ with FD's

$$F: \{A \rightarrow B,\ A \rightarrow C,\ CD \rightarrow E,\ B \rightarrow D,\ E \rightarrow A\}$$

which of the following F.D's is not implied by the above set. ?

a) $CD \rightarrow AC$ $\qquad CD^+ = \{C, D, E, A, B\}$

~~b)~~ $BD \rightarrow CD$ $\qquad \times\ BD^+ = \{B, D\}$

c) $BC \rightarrow CD$ $\qquad BC^+ = \{B, C, D, E, A\}$

d) $AC \rightarrow BC$ $\qquad AC^+ = \{A, C, B, D, E\}$

② To find keys of a relation

The set of all attributes that can be dermined using the given set of attributes is called attribute closure and is denoted by X^+. If X^+ contains all the attributes of a relation then X is called superkey of R. where R is a relation.

If X is a minimal set then X is called Candidate key of R.

① R(ABC)

F.D: $\{A \to B, B \to C\}$

$A^+ = \{A, B, C\}$ ← key

$B^+ = BC$ ✗

C.K: A

S.K: A — φ, B, C, BC

φ	A
B	AB
C	AC
BC	ABC

4 S.K

③ R(ABCD)

F: $\{AB \to C, B \to D\}$

$AB^+ = \{A, B, C, D\}$, key

$B^+ = \{B, D\}$

$A^+ = \{A\}$

S.K (AB) — φ, C, D, CD

φ	AB
C	ABC
D	ABD
CD	ABCD

4 S.K

③ R(A, B, C, D)

F: { AB → CD, CD → AB }

~~CD → A~~

~~CD → B~~

~~AB → C~~

~~AB → D~~

$AB^+ = \{ABCD\}$: C·K

$CD^+ = \{CDAB\}$: C·K

S·K - AB
- φ - AB
- C - ABC
- D - ABD
- CD - ABCD

4 · S·K

S·K CD
- φ - CD
- A - ACD
- B - BCD
- AB - ABCD

4 · S·K

# of superkeys = 7

④ R(A B C)

F: {A → B, B → C, C → A}

$A^+ = \{A, B, C\}$: C·K

$B^+ = \{B, C, A\}$: C·K

$C^+ = \{C, A, B\}$: C·K

3 · C·K

S·K

A
- φ — A
- B — AB
- C — AC
- BC — ABC

S·K

B
- φ — B
- A — AB
- C — BC
- AC — ABC

S·K

C
- φ — C
- A — AC
- B — BC
- AB — ABC

# of superkeys = 7

5) Consider R(ABCD) with C.K: AB, BC find the no. of superkeys ?

Ans.

1) AB 5) BC
2) ABC 6) BCD
3) ABD
4) ABCD

S-K

AB — ϕ — AB
AB — C — ABC
AB — D — ABD
AB — CD — ABCD

BC — ϕ — BC
BC — A — ABC
BC — D — BCD
BC — AD — ABCD

Total no. of superkeys = 6

6) R(ABCD),

F: { $AB \to CD$, $A \to B$ } find is AB a candidate key or only superkey ?

Ans.

$AB^+ = \{A, B, C, D\}$: key - Superkey

$A^+ = \{A, B, C, D\}$: key - candidate key.

$B^+ = \{B\}$

$\therefore$ AB is super key but not candidate key because A is a candidate key.

⑦ R(ABCD) with F:D (A→B, B→C } find C.K?

Ans

$A^+ = \{A,B,C\}$ ×

$B^+ = \{B,C\}$ ×

$C^+ = \{C\}$ ×

$D^+ = \{D\}$ ×

$AD^+ = \{A,B,C,D\}$: Candidate key

hidden attribute : The attributes that are not part of RHS of dependenci

Hidden candidate key.

⑧ R(ABCDE)

F: { AB→C, C→D } find C.K?

Ans

$AB^+ = \{AB,C\}$

$C^+ = \{C,D\}$

$ABE^+ = \{A,B,C,D,E\}$ ✓

C.K = ABE

⑨ R(A B C) , F: { AB→C , C→A }

~~Ans~~

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

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

CK : AB

CB (C→A)

Note :- The attributes of a key are called key attributes or prime attribute. If prime attribute appeared on the R.H.S of some dependency that can be replaced with its L.H.S to get additional candidate keys.

(16)

R(ABCD) F: { $AB \to C$, $B \to D$, $C \to B$, $D \to B$ }

find all C.K's of R. ?

Ans

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

C.K

AB

AC ∵ ($C \to B$)

AD ∵ ($D \to B$)

Candidate keys

AB

AC

AD

3 candidate keys.

17) R(A,B,C,D,E)

F: { $A \to B$, $BC \to D$, $D \to AE$ } find C.K's of R?

Ans

1) $BC^+ = \{B, C, D, A, E\}$: C.K

2) AC = C.K.

$D \to AE$

$D \to A$, $D \to E$

3) DC : C.K

(12) $R(ABCD)$

$F: \{AB \rightarrow C, C \rightarrow A, B \rightarrow D, D \rightarrow B\}$ $\underline{C.K = ?}$

Ans

1) AB : C.K
2) CB : C.K
3) CD : C.K
4) AD : C.K

1) $AB^+ = \{A, B, C, D\}$
2) CB $C \rightarrow A \Rightarrow$

$D \rightarrow B \Rightarrow$

3) AD
4) CD

(13) $R(ABCDE)$

$F: \{AB \rightarrow C, CD \rightarrow E, DE \rightarrow B\}$

$AB^+ = \{A, B, C\}$ ×

$CD^+ = \{C, D, E, B\}$ ×

$DE^+ = \{D, E, B\}$ ×

$AD^+ = \{A, D\}$ ×

$ABD^+ = \{A, B, C, D, E\}$: C.K

$ACD^+ = \{A, B, C, D, E\}$: C.K

$ADE^+ = \{A, B, C, D, E\}$: C.K

(14) ~~R(~~

(14) R(A B C D E H)

F: { A → B, BC → D, E → C, D → A }   C.K = ?

Ans: ~~ABCEH~~

1) A E H : C.K
2) D E H : C.K
3) B E H : C.K

~~AEH~~

$A \cdot E \cdot H^+ = \{A, E, H, B, C, D\}$

DEH   D → A

BCEH   BC → D

BEH   E → C

3) To find Equivalance of F.D's

To sets of F.D's 'F' and 'G' are said to be equivalent iff $F^+ = G^+$, hence equivalance means that every F.D's in F can be infered using G. and Every F.D in G can be infered using F. ie, F = G iff both

F covers G and G covers F holds.

Ex:- Consider the following two sets of F.D's

F: { A → C, AC → D, E → AD, E → H }

G { A → CD, E → AH }   is they two are equivalent ?

Ans:

~~A → C~~ ~~A → D~~   ~~E → A~~ ~~E → H~~

F covers G

A^+

$A \to CD$ compute A^+ using F

$A^+ = ACD$

$E \to AH$ compute E^+ using F

$E^+ = E, ADHC$

G covers F

$A \to C$ compute A^+ using G

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

$AC \to D$

$AC^+ = \{A, C, D\}$

$E \to AD$
$E \to H$
E^+ using G

$E^+ = \{E, H, A, C, D\}$

F covers G and G covers F $\Rightarrow$ Both F and G are equivalent.

② $F: \{A \to B, B \to C, C \to D\}$

$G: \{A \to BC, C \to D\}$

G covers F ×

$A^+ = \{ABCD\}$

$B^+ = \{B\}$ ×

F covers G ✓

$A^+ = \{ABCD\}$

$C^+ = \{CD\}$

③ $F: \{ A \to B, AB \to C, D \to AC, D \to E \}$

$G: \{ A \to BC, D \to AB \}$

F covers G ✓

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

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

G covers F ✗

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

$D^+ = \{D, A, B, C\}$ ✗ E not covered.

④

$F: \{ A \to B, B \to C, C \to A \}$

$G: \{ A \to BC, B \to A, C \to A \}$

F covers G & G covers F } equal.

④ To find minimal set of functional dependancies

F : given set

↓

G^+ : minimal set

Ex:-

① $F: \{ A \to B, B \to C, \cancel{A \to C} \}$ possible by finding A^+ in G.

$G: \{ A \to B, B \to C \}$ - minimal set

② $F: \{ A\cancel{B} \to C, A \to B \}$ AB → C is possible by finding AB^+ in G.

$G: \{ A \to C, A \to B \}$ - minimal set

A minimal cover for a set of F.D's 'F' is a set of F.D 'G' that satisfies the property that every dependency in F is in G^+. Then G is called minimal set.

Procedure to find minimal set

Step ① Put the F.D's in the standard form.

Ex:-

$A \rightarrow BC \implies A \rightarrow B \;\&\; A \rightarrow C$

Step ② Remove redundant F.D's

Ex:- $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\} \implies \{A \rightarrow B, B \rightarrow C\}$

Step ③ Minimize L.H.S of each F.D

$AB \rightarrow C$

A - can be deleted when B^+ contains A

B - can be deleted when A^+ contains B

If there was any removal of variables in step ③ repeat ② after that ③ — repeat this till there is no-removal. $\implies$ It will be minimal.

Q:- Find the minimal set of F.D for the following

F: { A → C, AC → D, E → AD, E → H }

Soln:-

Step 1:

1) A → C

2) AC → D

3) E → A

4) E → D

5) E → H

Step 2:-

✓ A → C

Compute A^+ using 2, 3, 4, 5

$A^+ = A \Rightarrow$ 1 needed.

✓ AC → D

Compute AC^+ using 1, 3, 4, 4, 5

$AC^+ = AC$

✓ E → A

E^+ (using 1, 2, 4, 5)

= E, D H

E → D

E^+ (using 1, 2, 3, 5)

= E, A, H, C, D

you can get E → D without 4

⟹ delete it.

✓ E → H (using 1, 2, 3, 4)

$E^+ = \{E, A, C, D\}$

Step 3

$AC \to D$

if A deleted: $C^+ = \{C\}$ $\Rightarrow$ A can not be deleted

if C deleted: $A^+ = \{A, C, D\}$ $\Rightarrow$ C can be deleted.

$\Rightarrow$ $A\cancel{C} \to D$ $\Rightarrow$ $A \to D$

Sets after step 3

1) $A \to C$
2) $A \to D$
3) $E \to A$
4) $E \to H$

(repeat step 2) $\Rightarrow$ Nothing will be elimina

minimal set or minimal cover

$= \{A \to C, A \to D, E \to A, E \to H\}$

Canonical cover

$\{A \to CD$
$E \to AH\}$

L.H.S of all the dependencies should be unique.

(2) $f: \{A \to B, C \to B, D \to A, BC, AC \to D\}$

Step 1

1) $A \to B$
2) $C \to B$
3) $D \to A$
4) $D \to B$
5) $D \to C$
6) $AC \to D$

Step 2

①✓
②✓
③✓
④✗

Step 3

$A \to B$
$C \to B$
$D \to A$
$D \to C$
$AC \to D$

(3) $AB \to C$
$C \to B$
$A \to B$

Step 1

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

Step 2

$A\cancel{B} \to C$
$C \to B$
$A \to B$

Step 3

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

(2) $A \to BC$ | $A \to B$
$B \to C$ | $B \to C$
$AB \to D$ | $A \to D$

minimal cover

$\cancel{AB}$ $A \to BD$
$B \to C$

~~minimal~~ canonical cover

P-160
Q-11

R(ABCDEFG)

F: { $A \to BC$, $AB \to DE$, $D \to EF$, $F \to A$ }

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

$B^+ =$

AG^+ : C.K

F.G : C.K

D.G : C.K

$F \to A$

$D \to EF =$
$D \to E$.
$D \to F$

P-163
L-2
Q-1

$A \to C$ ×

$A \to B$ ✓

P-163
Q-05

P-164
Q-21

$ABD \to AC$
$C \to BE$
$AD \to BF$
$B \to E$

$A\cancel{B}D \to A\cancel{X}$
$A\cancel{B}D \to C$ ✓
$C \to B$ ✓
$C \to E$ ✗
$AD \to B$ ✓
$AD \to F$ ✓
$B \to E$ ✓

$AD \to C\cancel{B}F$
$C \to B$
$B \to E$

$AD \to CF$
$C \to B$
$B \to E$
} Canonical cover.

$AD \to C$
$AD \to F$
$C \to B$
$B \to E$
} minimal cov

P-165
Q-22

$A \to BC$
$A\cancel{B}E \to \cancel{C}DGH$
$C \to \cancel{G}D$
$D \to G$
$E \to F$

$A \to C$
$A \to B$
$AE \to H$
$C \to D$
$D \to G$
$E \to F$
} minimal cover

P-165
Q-23

$BC\cancel{D} \to A$
$BC \to E$
$A \to F$
~~[illegible]~~
$F \to G$
$C \to D$
$A \to G$ ✗

$BC \to A$
$BC \to E$
$A \to F$
$F \to G$
$C \to D$
~~$A \to G$~~
} minimal cover.

~~[illegible]~~
~~[illegible]~~

Properties of decomposition

05-12-13

① Loss-less join decomposition

R

$\boxed{R_1 \bowtie R_2 = R}$

Let R be a relationship schema and F be a set of F.D's over R, the decomposition of R into R_1 and R_2 is said to be lossless decomposition w.r.t F iff $R_1 \bowtie R_2 = R$

* $\bowtie$ - natural join

Ex:-

R:

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_1	b_2	c_2

$R_1.A = R_2.A$

R_1

B	A
b_1	a_1
b_1	a_2
b_2	a_1

R_2

A	C
a_1	c_1
a_2	c_1
a_1	c_2

$R_1 \bowtie R_2 = S$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
b_2	b_1	c_1
a_1	b_2	c_1
a_1	b_2	c_2

Spurious tuples

$S \neq R \Rightarrow$ Lossy

R_1

A	B
a_1	b_1
a_2	b_1
a_1	b_2

R_2

B	C
b_1	c_1
b_2	c_2

$R_1 \bowtie R_2 = S$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_1	b_2	c_2

$S = R \Rightarrow$ Lossless

if $\boxed{R_1 \cap R_2 \longrightarrow R_2 - R_1 \quad \text{(OR)} \quad R_1 \cap R_2 \longrightarrow R_1 - R_2}$

Note:

The decomposition of R into R_1 and R_2 is said to be lossless if the attribute that is common in R_1 and R_2 is a key in either of the relations.

Let R be a relation and F be a set of functional dependencies over R. The decomposition of R into R_1 and R_2 is lossless iff F^+ contains either the functional dependency

$R_1 \cap R_2 \longrightarrow R_2 - R_1$

(OR)

$R_1 \cap R_2 \longrightarrow R_1 - R_2$

Qu: consider a Relation R(ABC) with FD $A \rightarrow B$ is decomposed into $R_1(AB)$ & $R_2(BC)$ check whether the decomposition is lossy or lossless.

Ans:

$R_1 \cap R_2 \rightarrow R_1 - R_2 \Rightarrow B \rightarrow A$ ✗

$R_1 \cap R_2 \rightarrow R_2 - R_1 \Rightarrow B \rightarrow C$ ✗

∴ the decomposition is lossy.

$\{AB\} \cap \{BC\} = B$

$R_1 - R_2 = \{AB\} - \{BC\}$
$= A$

$R_2 - R_1 = \{BC\} - \{AB\}$
$= C$

Q: R(ABC)

$f: \{A \rightarrow B\}$ decomposed into $R_1(AB)$, $R_2(AC)$

Sol:-

$R_1 \cap R_2 \longrightarrow R_1 - R_2$

$\{AB\} \cap \{AC\} \rightarrow \{AB\} - \{AC\}$

$A \rightarrow B \implies$ lossless

Qus $R(ABCD)$

$F: \{A \to B, B \to C, C \to D\}$ decomposed into

$R_1(AB)$ $R_2(BC)$ $R_3(CD)$ the decomposition is lossless or lossy ?

Ans

$(R_1 \bowtie R_2) \to B$ key in R_2

$((R_1 \bowtie R_2) \bowtie R_3) \to C$ is key in R_3

} <u>loss less</u>

Ques $R(ABCDE)$

$F: \{A \to B, C \to DE, AC \to F\}$

$R_1(AB)$ $R_2(CDE)$ $R_3(ACF)$ the decomposition is ______

$$[R_1 \bowtie (R_2 \bowtie R_3)] = R$$

loss less

Qus $R(ABCDEFGHIJ)$

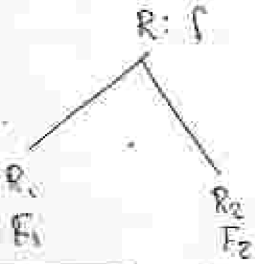
$F: \{AB \to C, A \to DE, B \to F, F \to GH, D \to IJ\}$

$R_1(ABC)$ $R_2(ADE)$ $R_3(BF)$ $R_4(FGH)$ $R_5(DIJ)$

$$[[[(R_1 \bowtie R_2) \bowtie R_3] \bowtie R_4] \bowtie R_5] = R$$

<u>Lossless</u> decomposition

II Dependancy Preserving Decomposition



$\boxed{F_1 \subseteq F^+}$ $\quad$ $\boxed{F_2 \subseteq F^+}$

$$\boxed{F^+ = (F_1 \cup F_2)^+}$$

The decomposition of a relation R with F.D's 'F' into R_1 and R_2 with F.D's F_1 and F_2 respectively is said to be dependancy preserving iff $\boxed{F^+ = (F_1 \cup F_2)^+}$

Ex:- R(ABC), F.D { A → B, B → C, C → A } is decomposed into $R_1(\overset{F_1}{AB})$ & $R_2(\overset{F_2}{BC})$ is the decomposition is dependancy preserving or not ?

Soln:-

$R_1 : F_1$	$R_2 : F_2$
A → B	B → C
B → A	C → B

$(F_1 \cup F_2) = \{ A \to B, B \to C, C \to B, B \to$

$(F_1 \cup F_2)^+ = \{ A \to B, B \to C, C \to A, \ldots$

$\Rightarrow \underline{(F_1 \cup F_2)^+ = F}$

⟹ dependancy preserving decomposition

$F^+ \begin{cases} A^+ = ABC \; \{ A \to A \;\; A \to B \;\; A \to C \} \\ B^+ = BCA \; \{ B \to A, B \to B, B \to C \} \\ C^+ = CBA \; \{ C \to A, C \to B, C \to C \} \end{cases}$

Q. R(ABCD) with F.D $F: \{AB \to C, D \to A\}$

$R_1(AD)$	$R_2(BCD)$
F_1: $D \to A$	F_2: $DB \to C$

is Decomp. is dep. preserving or not

$(F_1 \cup F_2)$: $\{D \to A, DB \to C\}$

Compute $AB^+ = A, B$ from $(F_1 \cup F_2)$

$AB \to C$ is lost

hence not preserving dependency.

$F^+ = \{AB \to C, D \to A\}$

$A^+ = A$

$B^+ = B$

~~ABC~~

$D^+ = D, A$

$DB^+ = D, B, A, C$

$C^+ = C$

$BC^+ = B, C$

$CD^+ = C, D, A$

Qus R(ABCD)

$F: \{A \to B, B \to C, C \to D, D \to A\}$

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

F^+

~~$A^+ = A, B$~~

$$\left.\begin{array}{l} D \to C \\ C \to B \\ B \to A \end{array}\right\} D \to A$$

$\Rightarrow$ Dependency is preserved

Q4 R(ABCDEG)

$F: \{AB \to C, AC \to B, AD \to E, B \to D, BC \to A, E \to G\}$

$R_1(ABC)$	$R_2(ABDE)$	$R_3(EG)$
$AB \to C$ $AC \to B$ $BC \to A$	$AD \to E$ $B \to D$	$E \to G$

$(F_1 \cup F_2 \cup F_3)^+ = F^+$ $\Rightarrow$ Dependancy is preserved

Q5 R(ABCDE)

$f: \{A \to BC, C \to DE, D \to E\}$

$(R_1 \bowtie R_2) = R$

$R_1(ABCD)$	$R_2(DE)$
$A \to BC$ $C \to D$	$D \to E$

I lossy ~~II~~ lossless

~~III~~ D.P IV not D.P

P-166

Q.34

1) C.K = DB D.P

~~...~~ BC & AD is not a decomposition because its lossy

2) C.K : BC, AB

lossless bcs C is key in R_1

It is not preserving dependancy ($AB \to C$ lost).

3)

) $A \to BC$, $C \to AD$

C.K : C, A

D.P ✗ loss less.

ABC	AB
$A \to B$	
$A \to C$	$A \to D$
$AC \to A$	

$AB \to D$

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

C.K : A

AB	ACD
$A \to B$	$A \to C$
	$A \to D$

loss less but Not D.P

$B \to C$ is loss

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

AB, AD, CD

lossy ✗ Not D.P

$B \to C$

Q: R(ABC)

F: { $\underset{FFD}{A \to B}$, $\underset{FFD}{B \to C}$ }

$$\begin{tikzcd} \underset{\text{Key}}{A} \arrow[r, bend left, "FFD"] & B \arrow[r, bend left, "FFD"] & C \end{tikzcd}$$

normal form ~~form~~ : 2NF

R is in 2NF and also in 1NF

Q: R(ABCD) with f: { AB → C, A → D } decompose the above relation into 2NF ?

Ans $R_1(ABC)$ & $R_2(AD)$ –

CK : AB

R is in 1NF but not in 2NF

$R_1(ABC)$ – 2NF

$R_2(AD)$ – 2NF

Q R(ABCDE)

F: { AB → C, A → D, B → E } decompose the above relation into 2NF.

$$\left\{ \begin{array}{l} R_1(ABC) \ – 2NF \\ R_2(AD) \ – 2NF \\ R_3(BE) \ – 2NF \end{array} \right\}$$

C.K : AB

PFD = A → D

B → E

R is in 1NF but not in 2NF

Note:- The decomposition required the closures of the violating dependenc into seperate relations and remaining attributes (if any) and key attributes of decomposed relations forms another relation.

3NF

Un-normalized relation

↓ Remove MVA, composite attribute

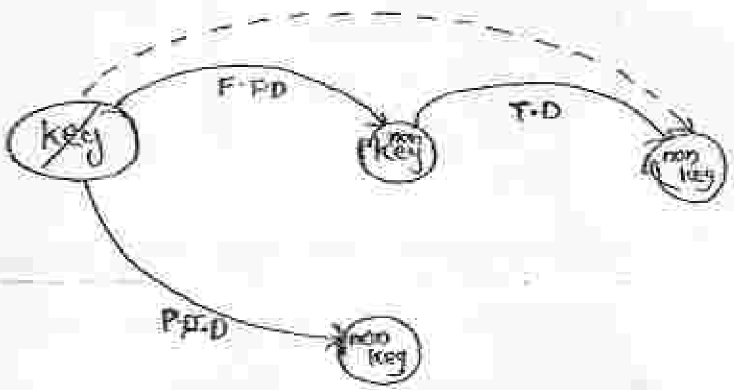
1NF (atomic Values)

↓ Remove P.F.D (partial key → non-key)

2NF (allows only F.F.D)

↓ Remove transitive dependency (non-key to nonkey)

3NF



Normalization

Normalization of data can be looked upon as a process of analyzing the given relation schema based on their F.D's and primary key's to achieve the defined properties of minimizing redundancy and minimizing the insertion deletion and update anomalies. Several normal forms have been proposed. each normal form minimizes the redundancy upto some extent. The higher normal forms are only of theoretical interest but not practically applicable.

Most DB systems uses normalization upto 3NF. (upto BCNF is recomended.)

1NF :-

Un-normalized relation

Employee

eno	ename	Contact
1	A	{98, 99}
2	B	{98, 100}

$\Rightarrow$

1NF

eno	ename	Contact
1	A	98
1	A	99
2	B	98
2	B	100

lossless decomp

Emp_contacts

eno	Contact
1	98
1	99
2	98
2	100

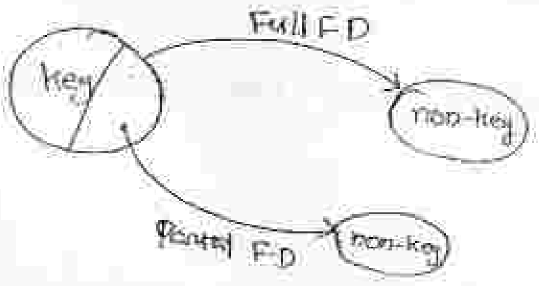
Emp

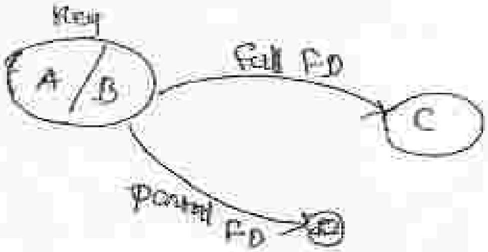
eno	Ename
1	A
2	B

A relation is in 1NF if every field contains only atomic values
ie, The attribute of any tuple must be a single value or null value from its domain.

⊙ Every relation in RDBMS must satisfy 1NF.

2NF :-





Qu: R(ABCD)
F: {AB→C, B→D}

2NF is based on the concept of full functional dependency and it disallows partial functional dependencies.

A Relation schema R is in 2NF if every non-key attribute of R is fully functionally dependent on the key of R

Qu: R(ABCD)

$F: \{\underbrace{AB \to C}_{FFD}, \underbrace{B \to D}_{PFD}\}$ C.K: AB

R is in 1NF but not in 2NF (because PFD: B→D).

Q.27

R(ABCDEFGHIJ)

$AB \rightarrow C$ - PFD

$BD \rightarrow EF$ - PFD

$AD \rightarrow GH$ - PFD

$A \rightarrow I$ - PFD

$H \rightarrow J$ - TD

CK: ABD

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

$R_1 = (ABCI)$ → $R_4(ABC)$, $R_5(AI)$

$BD^+ = \{B, D, E, F\}$

$R_2(BDEF)$

$AD^+ = \{G, H, A, D, I, J\}$

$R_3 = \{A, D, G, H, I, J\}$ TD → $R_6(ADGH)$, $R_7(AI)$ x, $R_8(HJ)$

⇒ 3NF tables

R(ABCDEFGH) → $R_9(ABD$, $R_4(ABC)$, $R_5(AI)$, $R_2(BDEF)$, $R_6(ADGH)$, $R_8(HJ)$

Q. 26

R(ABCDE)

CK: AC

f: {A → B, B → E, C → D}

R in 1NF, not in 2NF

$A^+ = \{AB, E\}$ R_2 $\xrightarrow{3NF}$ B^+ (BE) R_5 ; (AB) R_4

$C^+ =$ (CD) R_1

(AC) R_3

Q-28

(a) R(ABCD) with primary key AB.

f: {AB → C, $\underset{P.FD}{B \to D}$} 1NF not in 2NF

(b) R(ABCD) with key AB and 2NF not in 3NF

f: {AB → C, $\underset{T.D}{C \to D}$}

(c) R(ABCD) with key AB and 3NF

f: {AB → CD}

3NF is designed to disallow transitive dependencies.

(OR)

A Relation schema R is in 3NF if it satisfies 2NF and no-non prim attribute of R is transitively dependent on key ~~attribute~~ of R.

Ex:-
① $R(ABC)$

$$F: \{ \underbrace{A \to B}_{F.F.D},\ \underbrace{B \to C}_{F.F.D} \}$$

T.D

C.K: A

R is in 2NF but not in 3NF because transitive dependency $B \to C$

Decompose the above relation into 3NF

$B^+ = \{B, C\}$ $\quad R_1 \{B, C\}$

$A^+ = \{A, B\}$ $\quad R_2 \{A, B\}$

} 3NF

Ex:
② Decompose $R(ABCDE)$, $f: \{ \underbrace{AB \to C}_{F.F.D}, \underbrace{B \to D}_{F.F.D}, D \to E \}$ into 3NF?

C.K: AB

R is in 1NF.

$B^+ = B, D, E$ $\quad D^+ = \{D, E\}$

$R(ABCDE)$ — $R_1(BDE)$ — $R_3(BD)$, $R_4(DE)$

$R(ABCDE)$ — $R_2(ABC)$ .... $R_2(ABC)$

2NF $\quad\quad$ 3NF

Qu: R(ABC)

F.D: { $\underset{S.F.D}{AB \to C}$, $\underset{P.F.D}{C \to A}$ }

C.K: AB
CB

R is in 3NF and also in 2NF & 1NF

3NF

X → A is a non trivial functional dependency.

S.K (or) Prime attribute

Note:-

A Relation schema R is in 3NF If whenever a non-trivial functional dependency X ⟶ A holds in R then either X is a S.K of R or A is a prime attribute of R.

P-165
Q-25

~~B(B.T, A.N, B.Tg, L.P, A.a,~~

R(A,B,C,D,E,F)

F: { $\underset{P.F.D}{A \to FC}$, C → D, $\underset{P.F.D}{B \to E}$ }

C.K: AB

R is in 1NF

A^+ = { AFCD } R_1 ✓ → C^+ { CD } R_4 , { AFC } R_5 } 3NF

B^+ = { B,E } R_2 ✓

(AB) R_3 ✓ } 3NF

2NF

Boyce Codd Normal form (BCNF)

Ex:-

R (ABC)

f: { AB → C, C → A }

$\overline{A}\underline{BC}$ ← overlapping candidate keys

C.K : AB
CB

3NF but not in BCNF

A Relationship schema R is in BCNF if whenever a non-trivial F.D X ⟶ A holds in R then X is a Superkey of R. ie, the determinants of all functional dependencies must be Superkeys.

Q: R (ABC)

f: { A → B, B → C, C → A }

→ The relation is in BCNF ∵ every attribute is key

C.K : A, B, C

⇓ It is also in 3NF, 2NF, 1NF

Note:- If every determinant is super key of R, the relation is in BCNF

Q: Find normal form of a two attribute relation is ______

R(AB) – BCNF

i) f : {A → B}   C.K : A   – BCNF
ii) F : {B → A}   C.K : B   – BCNF
iii) F : {A → B, B → A}   C.K : A, B   – BCNF
iv) F : { φ }   C.K : AB   – BCNF

Q:- $R(ABC)$: $f:\{\phi\}$

the Nor. for. = BCNF

Note:- A Relation with only trivial dependencies is always in BCNF

Q: $R(ABC)$

$f: \{\underbrace{AB}_{S.K} \rightarrow C,\ C \rightarrow \underbrace{B}_{\text{Prime}}\}$ Decompose the above Relation into BCNF.

C.K : AB, AC R is in 3NF but not in BCNF.

$R \rightarrow C^+ = \{C, B\}\ R_1 \quad f_1: \{C \rightarrow B\}$

$\{A, C\}\ R_2 \quad f_2: \{\phi\}$

$(AB \rightarrow C$ is lost$)$

The BCNF decomposition may not ensures dependancy preserving decomposition.

Q: $R(ABCDEFGHIJ)$

$f: \{AB \rightarrow C,\ \underbrace{A \rightarrow DE}_{P.F.D},\ \underbrace{B \rightarrow F}_{P.F.D},\ F \rightarrow GH,\ D \rightarrow IJ\}$ decompose the above relation into BCNF.

CK AB

2NF

$A^+ = \{A, D, E, I, J\}$ R_1

$B^+ = \{B, F, G, H\}$ R_2

$\{ABC\}$ R_3 ✓ BCNF

$R_2(BFGH)$ — 3NF

- $\{FGH\}$ R_4 ✓ BCNF
- $\{BF\}$ R_5 ✓ BCNF

$R_1(ADEIJ)$ — 3NF

- $\{DIJ\}$ R_6 ✓ BCNF
- $\{A, E, D\}$ R_7 ✓ BCNF

R — BCNF

- $R_3(ABC)$
- $R_7(AED)$
- $R_6(DIJ)$
- $R_4(FGH)$
- $R_5(BF)$

Q: R(ABCD) is a relation. which of the following does not have a lossless join and D.P BCNF decomposition

a) $A \to B$, $B \to CD$

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

✓ c) $AB \to C$, $C \to AD$

d) $A \to BCD$

a) $(AB)R_1$ $(BCD)R_2$ ✓

b) $(AB)R_1$ $(BC)R_2$ $(CD)R_3$ ✓

c) $(CAD)R_1$ $(BC)R_2$ ✗ not D.P → $(CD)R_3$, $(CA)R_4$

d) $(ABCD)R_1$ ✓

Q-29

R(ABCDE)

$F: \{A \to \underset{PA}{B},\ BC \to \underset{PA}{E},\ ED \to \underset{PA}{A}\}$

C.K: ACD

C.K: ECD

C.K: BCD

R is in 3NF

Q-30

R(ABCD)

(i) CK → ?

(ii) N.F → ?

(iii) decompose if not in BCNF

1) $\underset{2NF}{C \to D},\ \underset{2NF}{C \to A},\ \underset{BCNF}{B \to C}$

(i) C.K: B

(ii) 2NF

(iii)

$F: \{B \to C,\ C \to A,\ C \to D\}$

$C^+ = \{C, A, D\}$ R_1 – BCNF

$\{B, C\}$ R_2 – BCNF

2) $F: \{\underset{BCNF}{B \to C},\ D \to A\}$

(i) C.K: BD

(ii) BCNF

3) $F: \{ABC \to D,\ D \to A\}$

(i) C.K: ~~ABC~~ ABC

DBC

(ii) 3NF

(iii)

(*)

BCNF : Superkey → any

3NF : → Prime attribute

2NF : non-key → non-key

1NF : part of key → non-key

Ex:-

R(ABCDE) C.K: AB

$F: \{\underset{BCNF}{AB \to C},\ \underset{1NF}{B \to D},\ \underset{2NF}{D \to E}\}$

R is in 1NF

Ex:- R(ABCDE) C.K: E

$F: \{\underset{2NF}{A \to B},\ \underset{2NF}{BC \to D},\ \underset{BCNF}{E \to AC}\}$

R is in 2NF

2) $\underset{1NF}{\underline{B \rightarrow C}}$, $\underset{1NF}{\underline{D \rightarrow A}}$

(i) C.K = BD

(ii) 1NF

(iii)

$B^+ = \{B, C\}$ R_1 ✓ BCNF

$D^+ = \{D, A\}$ R_2 ✓ BCNF

$\{BD\}$ R_3 ✓ BCNF

3) $\underset{BCNF}{\underline{ABC \rightarrow D}}$, $\underset{3NF}{\underline{D \rightarrow A}}$

(i) ABC : CK

DBC : CK

(ii) 3NF

$D^+ = \{D, A\}$ R_1 ✓ BCNF $\Longrightarrow$ Lossy decomposition ABC → D is lost

$\{DBC\}$ R_2 ✓ BCNF

④ $\underset{BCNF}{\underline{A \rightarrow B}}$, $\underset{2NF}{\underline{BC \rightarrow D}}$, $\underset{BCNF}{\underline{A \rightarrow C}}$

(i) C.K : A

(ii) 2NF

(iii) $BC^+ = \{BCD\}$ R_1 ✓

$\{ABC\}$ R_2 ✓

Q.31* R(ABCDEFGH)

f = { AB → C~~DEG~~H

F → G

FB → H

HBC → ~~ADE~~F~~G~~

FBC → ADE }

⇒

minimal cover

AB → CH

F → G

FB → H

HBC → F

FBC → ADE

AB^+ = ABCHFDE — : key

F^+ = F·G

FB^+ = FB·H·G

HBC^+ = HBCFGADE — : key

FBC^+ = FBC ADEGH — : key.

AB → CH / BCNF , F → G / 1NF , FB → H / 3NF , HBC → F / BCNF , FBC → ADE / BCNF

2NF ⇒ → F^+ = (FG) R_1 ✓

→ ABCDEFH

F_2: {

$AB \rightarrow CH$ BCNF

$FB \rightarrow H$ 3NF

$HBC \rightarrow F$ BCNF

$FBC \rightarrow ADE$ BCNF }

C.K: AB, HBC, FBC

BCNF $\Rightarrow$ R_2

FB^+ (FBH) R_3

(ACDEFB) R_4

c.k: FBC

$HBC \rightarrow F$ lost

$AB \rightarrow H$ lost

F_4: { $FBC \rightarrow ADE$, $AB \rightarrow C$ }

Work Book Problems

P-165

14

R(ABCDEFGHIJ)

$F: \{ \underbrace{AB \rightarrow C}_{\substack{F.F.D \\ BCNF}}, \underbrace{A \rightarrow DE}_{\substack{P.F.D \\ 1NF}}, \underbrace{B \rightarrow F}_{\substack{P.F.D \\ 1NF}}, \underbrace{F \rightarrow GH}_{\substack{T.D \\ 2NF}}, \underbrace{D \rightarrow IJ}_{\substack{T.D \\ 2NF}} \}$

$AB^+ = \{ABCDEFGHIJ\}$: C.K

R is in 1NF ∴ Decompose to 2NF

$A^+ = \{A, D, E, I, J\}$ R_1 : 2NF

$B^+ = \{B, F, G, H\}$ R_2 : 2NF

$\{AB\ C\}$ R_3 BCNF

BCNF

T.D

D^+

$R_1(ADEIJ)$ — $R_4(DIJ)$, $R_5(AED)$

F^+

$R_2(BFGH)$ — $R_6\{FGH\}$, $R_7\{BF\}$

If it is to decompose into

2NF —

$R_1(ADEIJ)$

$R_2(BFGH)$

$R_3(ABC)$

3NF — also in BCNF

$R_3(ABC)$

$R_4(DIJ)$

$R_5(AED)$

$R_6(FGH)$

$R_7(BF)$

P-168

Q25

R(A, B, C, D, E, F)

$f: \{ \underbrace{A \rightarrow FC}_{P.F.D}, \underbrace{C \rightarrow D}_{T.D}, \underbrace{B \rightarrow E}_{P.F.D} \}$

C.K = $AB^+ = \{A, B, C, F, D, E\}$

R is in 2NF

2NF

$A^+ = \{A, F, C, D\}$ R_1 ✓ 2NF $\xrightarrow{3NF}$ $C^+ = \{C, D\}$ R_4

$\{AFC\}$ R_5

$B^+ = \{B E\}$ R_2 ✓ BCNF

$\{AB\}$ R_3 ✓ BCNF

R (2NF)
- $R_1(AFCD)$
- $R_2(BE)$
- $R_3(AB)$

R(3NF)
- $R_2(BE)$
- $R_3(AB)$
- $R_4(CD)$
- $R_5(AFC)$

Q-26 R(ABCDE)

P.K: AC $f: \{\underbrace{B \to E}_{T.D}, \underbrace{C \to D}_{P.F.D}, \underbrace{A \to B}_{P.F.D}\}$ R is in 1NF

2NF

$A^+ = \{A, B, E\}$ R_1 ✓ 2NF

$C^+ = \{CD\}$ R_2 ✓ BCNF

$\{AC\}$ R_3 ✓ BCNF

3NF

$B^+ = \{BE\}$ R_4

$\{AB\}$ R_5

R (2NF)

$R_1(ABE)$ $R_2(CD)$ $R_3(AC)$

R(3NF)

$R_2(CD)$ $R_3(AC)$ $R_4(BE)$ $R_5(AB)$

Q-27 R(ABCDEFGHIJ)

$f: \{\underbrace{AB \to C}_{P.F.D}, \underbrace{BD \to EF}_{P.F.D}, \underbrace{AD \to GH}_{P.F.D}, \underbrace{A \to I}_{P.F.D}, \underbrace{H \to J}_{T.D}\}$

C.K: $ABD^+ = \{A, B, C, E, F, G, H, I, J\}$

$AB^+ = \{A, B, C, I\}$ R_1 ✓ 1NF $R_1(ABCI) \xrightarrow{2NF}$ $A^+ = \{A, I\}$ R_4 ✓ BCNF

$\{ABC\}$ R_5 ✓ BCNF

$BD^+ = \{B, D, F, E\}$ R_2 ✓ BCNF

$AD^+ = \{A, D, G, H, J\}$ R_3 ✓ 2NF

$\{AB\,BD\,AD\} = \{ABD\}$ R_0 ✓ BCNF

$R_3(ADGHJ) \xrightarrow{3NF}$ $H^+ = \{H, J\}$ R_6 ✓ BCNF

$\{ADGH\}$ R_7 ✓ BCNF

BCNF of R

$R_0(ABD)$

$R_7(ADGH)$

$R_6(HJ)$

$R_5(ABC)$

$R_4(AI)$

$R_2(BDFE)$

Q-29

$R(ABCDE)$ $f: \{\underset{FFD}{A \to B}, \underset{FFD}{BC \to E}, \underset{FFD}{ED \to A}\}$

Keys: ACD, ECD, BCD } R is in 3NF

Q-30

(1) $R(ABCD)$

$\underset{\substack{TD \\ 3NF}}{C \to D}, \underset{\substack{TD \\ 3NF}}{C \to A}, \underset{\substack{FFD \\ BCNF}}{B \to C}$ 2NF but not in 3NF

C.K: B

$R(ABCD) \xrightarrow{3NF}$ $C^+ \{C, D, A\}$ R_1, $\{BC\}$ R_2 } BCNF ✓

(2) $\underset{\substack{P.F.D \\ 1NF}}{B \to C}, \underset{\substack{PFD \\ 1NF}}{D \to A}$ R is in 1NF but not in 2NF

C.K: BD

R

$B^+ \{BC\}$ R_1 ✓ BCNF

$\{DA\}$ R_2 ✓ BCNF

$\{BD\}$ R_0 ✓ BCNF

R(ABCDEFGH)   f: { AB → C ~~...~~ H

A → D

F → G

FB → H

HBC → A ~~...~~ F ~~...~~

FBC → ~~...~~ E

}

K. AB, HBC, FBC

f: { AB → CH -- BCNF

A → D -(PFD) 1NF

F → G (PFD) 1NF

FB → H (PFD) 1NF

HBC → AF BCNF

FBC → E BCNF

}

$AB^+ = \{A, B, C, H, D, F, G, E\}$

$HBC^+ = \{H, B, C, F, A, G, D, E\}$

$FBC^+ = \{F, B, C, E, H, A, G, D\}$

} Keys

R is now in 1NF not in 2NF

→ $A^+ = \{A, D\}$ R_1 ✓ BCNF

→ $F^+ = \{F, G\}$ R_2 ✓ BCNF

→ $FB^+ = \{FBH\}$ R_3 ✓ BCNF

→ { A FBC E } R_4 ✓ BCNF

Preserved deps

AB → C

FBC → E

A → D

F → G

FB → H

Not preserved deps

AB → H

HBC → AF

(3) $\underset{\text{P.F.D} \\ \text{3NF}}{\underline{ABC \rightarrow D}}$, $\underset{\text{P.F.D} \\ \text{2NF}}{\underline{D \rightarrow A}}$

C.K : ABC
: DBC

R is in 3NF but not in BCNF

R → $D^+ = \{\underline{D}A\}$ R_1 ✓ BCNF
R → $\{DBC\}$ R_2 ✓ BCNF

Lossy decomposition

<u>Not D.P</u> $ABC \rightarrow D$ lost

(4) $\underset{\text{BCNF}}{\underline{A \rightarrow B}}$, $\underset{\text{T.D (2NF)}}{\underline{BC \rightarrow D}}$, $\underset{\text{BCNF}}{\underline{A \rightarrow C}}$

C.K : A

R is in 2NF but not in BCNF & 3NF

$BC^+ = \{\underline{BC}D\}$ R_1 ✓ BCNF

$\{\underline{A}BC\}$ R_2 ✓ BCNF

(5) $\underset{\text{BCNF}}{\underline{AB \rightarrow C}}$, $\underset{\text{BCNF}}{\underline{AB \rightarrow D}}$; $\underset{\text{P.F.D} \\ \text{3NF}}{\underline{C \rightarrow A}}$, $\underset{\text{P.F.D} \\ \text{3NF}}{\underline{D \rightarrow B}}$

C.K = AB
CB
AD
CD

R is in 3NF but not in BCNF

$C^+ = \{\underline{C}A\}$ R_1 ✓ BCNF
$D^+ = \{\underline{D}B\}$ R_2 ✓ BCNF
$\{\underline{CD}\}$ R_3 ✓ BCNF

<u>Not D.P</u> $AB \rightarrow CD$ lost

Q. 32

R(ABCDEFG)

F: { $BC\cancel{D} \to A$
$BC \to E$
$A \to F$
$F \to G$
$C \to D$
$A \to G$

key: $BC^+ = \{B, C, E, D, A, F, G\}$

F: { $BC \to A$ — BCNF
$BC \to E$ — BCNF
$A \to F$ — T.D (2NF)
$F \to G$ — T.D (2NF)
$C \to D$ — P.F.D (1NF)
$A \to G$ — T.D (2NF)
}

R is in 1NF but not in 2NF

1NF to 2NF

$C^+ = \{C, D\}$ R_1 — BCNF

{CABEFG} R_2 — 2NF $\xRightarrow{3NF}$ $A^+ = \{AFG\}$ R_3 — 2NF

{ACBE} R_4 — BCNF

R_3(AFG) → (FG) R_5 — BCNF
→ (AF) R_6 — BCNF

D.P & lossless

Q. 33

R{ABCDEFGH}

F:{ $A \to BC$
$A\cancel{B}E \to \cancel{F}H$
$C \to \cancel{G}D$
$D \to G$
$E \to F$
}

⇔

F: { $A \to BC$ (1NF)
$AE \to H$ (BCNF)
$C \to D$ (2NF)
$D \to G$ (2NF)
$E \to F$ (1NF)
}

key: $AE^+ = \{A, B, C, D, E, F, G$

1NF $\xRightarrow{(2NF)}$ $A^+ = \{A, B, C, D, G\}$ R_1 — 2NF

~~{AEFGH} R_2~~

$E^+ = \{EF\}$ R_2 ✓ BCNF

$\{AEH\}$ R_3 ✓ BCNF

2NF to 3NF $\Longrightarrow$

$R_1(ABCDG)$ — $C^+ = \{C, D, G\}$ R_4 ✓ 2NF — $D^+ = \{DG\}$ R_6

$\{ABC\}$ R_5 ✓ BCNF — $\{CD\}$ R_7

BCNF ✓

Q35

$R(ABCDE)$ $F: \{\underbrace{AB \rightarrow \cancel{C}D\cancel{E}}_{BCNF}, \underbrace{A \rightarrow C}_{2NF}, \underbrace{D \rightarrow E}_{2NF}\}$

C.K: AB

R is in 1NF but not in 2NF

1NF to 2NF $\Longrightarrow$ $A^+ = \{A, C\}$ R_1 ✓ BCNF

$(ABDE)$ R_2 ✓ 2NF

2NF to 3NF $\Longrightarrow$ $D^+ = \{DE\}$ ✓ BCNF R_3

$\{ABD\}$ R_4 ✓ BCNF

R is decomposed into $R_1(AC)$ $R_3(DE)$ $R_4(ABD)$ (BCNF form)

Q:36 $R(ABCDE)$ $F: \{\underbrace{AB \rightarrow \cancel{CD}E}_{BCNF}, \underbrace{A \rightarrow C}_{1NF}, \underbrace{C \rightarrow D}_{2NF}\}$

C.K = AB

R

$A^+ = \{ACD\}$ R_1 ✓ 2NF — $C^+ = \{CD\}$ R_3 ✓ BCNF

$\{AC\}$ R_4 ✓ BCNF

$\{ABE\}$ R_2 ✓ BCNF

R is decomposed to $R_2(ABE)$, $R_3(CD)$, $R_4(AC)$ and it is D.P.

37] R(ABCDEF) f: { [illegible] (BCNF), [illegible] (2NF), [illegible] (3NF), [illegible] (2NF), [illegible] (2NF) }

CK: A

$E^+ = \{EBFCD\}$ ~~2NF~~ $\xrightarrow{3NF}$ $B^+ = \{BCD\}$ $\xrightarrow{3NF}$ {CD} R
{BC} R
BCNF

{EBF} R_3
BCNF

R — {AE} R_4
BCNF

Q-38 R{ABCDE} f: { $AB \to CDE$ (BCNF), $C \to A$ (3NF), $D \to E$ (2NF) }

C·K = AB
CB

$D^+ = \{DE\}$ R_1 ~~BCNF~~

(ABCD) R_2 ~~3NF~~ $\xrightarrow{BCNF}$ $C^+ = \{CA\}$ R_3 ~~BCNF~~

(AB→CD is lost)

(CBD) R_4 ~~BCNF~~

Q-39 R=(ABCD) f: { $A \to B$ (BCNF), $B \to C$ (2NF), $C \to D$ (2NF) }

CK: A

$B^+ = \{BCD\}$ $\xrightarrow{3NF}$ $C^+ \{CD\}$ R_1 ~~BCNF~~

{BC} R_2 ~~BCNF~~

{AB} R_3 ~~BCNF~~

Q-40 R(ABCD) f: { $AB \to D$, $C \to A$ (3NF), $A \to C$ (3NF) }

C·K : AB
CB

$C^+ = \{CA\}$ R1 ~~BCNF~~ F_1 : { $C \to A$, $A \to C$ }

{ABD} R_2 ~~BCNF~~ F_2 { $AB \to D$ }

$C^+ = (CA)$

(ABD) (or) (CBD)

Structured Query Lang (SQL)

Database language

- Data Definition language (DDL) :- " Structure " :- (create, alter, drop, ---
- Data manipulation language (DML) :- " Data " :- insert, select, update, delete ...

SQ Examples

Basic queries

> Create table Student (Rno number (2), name char(10));

> insert into Student Values (1, 'Ram');

> Update student Set Name = 'Rahul' where Rno = 1;

> Delete student where rno = 1;

Query Evaluation Process

1.) from (cartition product of all tables)

A	
p	q
1	2
3	4

B	
r	s
5	6
7	8

A×B

p	q	r	s
1	2	5	6
1	2	7	8
3	4	5	6
3	4	7	8

4

(A×B)

2) Where (selects the tuples according to the 'where' condition)

3) Group by (Divides the rows into groups)

4) Having (selects the group)

5) Expressions in Select clause are evaluated (if any)

6) Distinct (Duplicates are eliminated)

7) Set operations (union, intersect

7) Set operations (Union, intersect, except)

8) Order by (Sorts the rows of result)

Simple Select

> Select rno from student;

> select rno, name from Student;

> Select * from Student;

> Select rno, marks+5 from Student;

> Select distinct branch from Student;

> Sele

Select where

- and
- or
- not
- in
- not in
- between .. and
- not between .. and
- is null
- is not null
- like %
- like _

- <, >, <=, >=, <>

not eq

Student (rno, name, branch, email , marks, city, passport);

1) Find students of CSE living in Hyderabad (HYD)

Select * from student where branch = 'CSE' and City = 'HYD';

(OR)

> Select *
  from student
  where branch = 'CSE' and City = 'Hyd';

) Find all students of CSE and IT

> Select *
  from student
  where branch = 'CSE' OR branch = 'IT' ;

) find all students ~~of~~ except of CSE.

> Select *
  from student
  ~~[illegible]~~
  where not branch = 'CSE';
  (OR)
  banch != 'CSE';
  (OR)
  branch <> 'CSE';

4.

4) find all students who are living in 'Hyd', 'Bao' & 'Chennai'

> Select *
from student
where city IN ('Hyd', 'Ban', 'Chennai');

(same as) :-
city = 'Hyd' or City = 'Bang' or City = 'Chen
Dis-adv :- * long list

Note:-
The IN operator is used to compare, a value or column with a list of values.

5) find all students who scored the marks from 10 to 20

> Select *
from student
where marks between 10 AND 20

(same as) :-
inclusive   exclusive
marks ≤ 10 and marks < 20;

Note:-
The between. and operator is used to compare a column with range of values.

Find all students who have no passport

> Select *
from student
where passport is null;

NULL is
- not zero
- Value does not exist
- Value is unknown
- Even if known, but Value not specified.

Ex: R - 10 tuples

(i) Select *
from R
where 1=1;

(ii) Select *
from R
where null is null

} 10 rows returned.

(iii) Select *
from R
where 0 is null

(iv) Select *
from R
where 1>2

} no rows returned

> update student
set marks = marks +5;

Select * from student;

(before) marks

1	A	10
2	B	
3	C	9

(after) marks

1	A	15
2	B	
3	C	14

Qu. Find all students whose name starts with 'S'.

```
Select *
from student
where name like 'S-%';
```

→ Starts with S

'%A';

→ ends with A

'%I%'

→ contains I

'___'

→ All 3-length name

'S__'

→ length 3 & starts with S

'S__%'

→ starts with S and minimum length 3.

Note:-

The 'Like' condition is used to specify certain search condition for a pattern in a column.

A percentile (%) sign can be used to define wildcards (missing char.) both before and after the pattern. % sign can be used to replace an arbitrary number of zero and more characters.

Underscore (_) replaces a single character.

Order-by

Order-by clause is used to sort the rows.

* Company (name, invoice_no)

Display all the company in alphabetical order of their names.

```
> Select *
from company
order by name asc;
```

Display all the company in reverse alphabetical order of their names.

> Select *
from Company
Order by name desc;

> Display all the companies in reverse alphabetical order of their names and numerical order of their invoice no.

I/p

name	invoice_no	qty
A	10	100
B	7	101
A	10	102
C	9	103
A	2	104

> Select *
from company
Order by name desc, invoice_no asc;

O/p

name	invoice_no	qty
C	9	103
B	7	101
A	2	104
A	10	100
A	10	102

Note:- when first order by clause fails it sorts the rows using 2nd order by clause when 2nd fails it sorts the row by 3rd order by clause and continues. If all order by clause fails it display

Aggregate functions

Avg

min

max

Sum

Count

> Select sum(marks), avg (marks)
from student;

marks
5
10
15

o/p:

Sum (marks)	avg (marks)
30	10

(OR)

aliasing

> Select sum(marks) as Total , avg (marks) as average
from student;

o/p.

Total	Average
30	10

Q) find a query to find diff between max. and min marks of the student

> select (max(marks) - min(marks)) as difference.
from student

o/p

Difference
10

Eg: find the no. of Students in a class.

> Select Count (*) as Strength
from Student

o/p Strength
3

Eg:-

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

> select max (name) from Student;

o/p : F

Note :- min and max functions can be used over ~~[illegible]~~
numbers, strings, dates

> Select avg (name) from Student; ✗
Error

Note :- Avg and sum functions can only be used with numbers.

> select count(*) from student;

o/p: 6 ← returns no. of rows in table.

> select count(m1) from student;

o/p: 5 ← returns no. of non-null values

> select count(4) from student;

6

→ count(constant) ;- returns no. of rows

> select count('Ramesh') from student;

6

> select count(m1+m2) from student;

3, 6, 10, 12

4

> select count(distinct m1) from Student;

3

Note:- with count function we can use any kind of data.

> Select sum(m1, m2) from student; X

Error

> select sum(m1+m2) from student;

o/p: 31

> Select name
from student
where $m_2 = max(m_2)$;

Error

Note:- Aggregate functions can-not be used in where clause.

> Select name, max(m_2)
from student

Error

name	max(m_2)
{A, B, C, D, E, F}	8

Violates 1NF

> Select name
from student
order by max(m_2);

Error

Note:-
Order by - expects a column name

> Select name
from student
order by m_2
where $m_2 >= 4$;

Order by should be here

Error

Q:- Consider a table T with following tuples

T

Rno	marks
1	10
2	20
3	30
4	null

The following sequence of SQL Statements was successfully executed on table T.

> Update T set marks = marks + 5;

> Select avg (marks) from T;

What is the output of the select T ?

a) 18.75 b) 20 c) 25 d) Error

Group by - clause and Having - clause

Group by clause is used to compute aggregate functions on a group.

```
> Select count (*)
  from Student
  where branch = 'CSE';
               = 'IT';
               = 'ECE';
```

```
> Select count (*), branch
  from Student
  group by branch;
```

Note:- we can reduce the burden on the query execution engine by using groupby instead of 1 by 1

Student

Rno.	name	Branch	Year	Gender
1	A	CSE	I	M
2	B	CSE	II	F
3	C	IT	I	F
4	D	IT	I	F
5	E	CSE	II	M
6	F	ME	I	F

Qu write a query to find number of students in each branch ?

> select Branch, Count(*)
from student
group by Branch

O/p:

Branch	count (*)
CSE	3
IT	2
ME	1

Qu find the no. of Students in each branch and year ?

> Select Branch, year, count (*)
from Student
group by Branch, Year ;

O/p

Branch	year	count (*)
CSE	I	1
CSE	II	2
IT	I	2
ME	I	1

◉ If the "year" is not included in the group by clause

...... group by Branch ;

O/P

Branch	Year	count (*)
CSE	{I, II}	3

violates 1NF , Error.

Note:-

All the columns that appear in the select clause must appear in the group by clause

Q. find no. of female students in each branch ?

```
> select Branch , count (*) ........④   execution order
  From Student ......................①
  where Gender = F ..................②
  Group by Branch ; .................③
```

O/P

Branch	count (*)
CSE	1
IT	2
ME	1

Q. find no. of female students in each branch and display the result if there are more than one student in the branch.

(P.T.O)

> select Branch, count(*) ...... ⑤
from student ...... ①
where gender = 'F' ...... ②
group by Branch ...... ③
having count(*) > 1 ; ...... ④

execution order

O/p

Branch	count (*)
IT	2

Note:-

Having clause is used to write group conditions

Aggregate functions can be used in having class.

Q. Find the no. of students in each branch except of CSE

A. > Select ~~[illegible]~~ Branch, count (*) ④
from student ①
where branch <> 'CSE' ②
group by branch ; ③

O/p:

Branch	count (*)
IT	2
ME	1

Q2 → Select Branch, count(*) ...... ④
from student ...... ①
group by branch ...... ②
having branch <> 'CSE' ; ...... ③

O/p

Branch	count(*)
IT	2
ME	1

Q_1 is more efficient than Q_2

Note:-

The column's that appear in having clause must be an aggregate function or must be part of a group by clause.

ie, The column appearing in having clause must be a single value per the group.

Where Vs Having

⇒ 'where' is used to select the rows whereas 'having' is used to select the group

⇒ Aggregate functions can-not be used in where clause, but can be used in having clause.

Qu which of the following statements are true about an SQL query.

✓P : An SQL query can contain a having clause even if it does not have a group by clause.

Q : An SQL query can contain a having clause only if it has a group by clause.

R : All the attributes used in the group by clause must appear in the select clause.

✓S : Not all attributes used used in the group by clause must appear in the select clause

a) P and R

~~b)~~ P and S

c) Q and R

d) Q and S

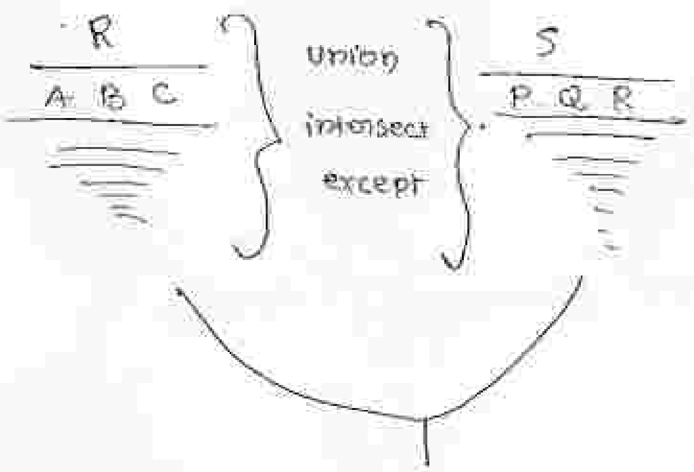
Ans

> Select count(*)
from Student
having count(*) > 2

$\frac{O/P}{6}$

Note: In the absence of group by clause, the whole relation will be considered as one group.

Set manipulation operations



compatible
- → Same no. of columns
- → Corresponding fields must have same dom

Since the answer to a query is multi set of rows it is natural to consider the set operations between two compatible relations ie, both must have same set of columns and corresponding columns taken in order from left to right must have same domain.

SQL offers set manipulation under the names 'Union', 'Intersect' and 'exept'.

Ex:-

Depositor (ac_no , cust_name)
Borrower (loan_no, cust_name)

Q. write a query to find name of the customers who have an account or loan or both at the bank.?

> (select cust_name from Depositor) {a, b, c}
  union = {a, b, c, p, q}
  (select cust_name from Borrower) {a, p, q}

Q. find name of the customer who have both an account and loan ?

> (select cust_name from Depositor) {a,b,c}

intersect

= {a}

(select cust_name from Borrower) {a,p,q}

Q. find the name of the customer who have an account but not loan. ?

> (select cust_name from Depositor) {a,b,c}

except

{b,c}

(select cust_name from Borrower) {ap,q}

JOIN

The join operation is used to combine related tuples from two relations into a single ~~table~~. tuple.

Employee (eno, ename, city, deptno) ;

Dept (dno, dname) ;

Q. find name of the employee's working in research department.

> Select ename

from Employee, Dept

where deptno = dno and dname = 'R';

o/p:

ename
A
C

Employee

Ename	Deptno
A	99
B	100
C	99

Dept

Dno	dname
99	R
100	S

Employee × Dept

A	99	99	R	✓
A	99	100	S	×
B	100	99	R	×
B	100	100	S	✓
C	99	99	R	✓
C	99	100	S	×

join .. on

> Select ename

from employee join Dept on deptno = dno

where dname = 'R';

o/p

ename
A
C

Natural join

When we have a cartesian product with equality condition using columns having the same name in the where condition it is called natural join.

Employee(eno, ename, cty, dno)

Dept (dno, dname);

> select ename

from Employee natural join Dept ⇒ [Employee join dept on Employee.dno = Dept.dno] Impsem

where dname = 'Research';

o/p ⇒

ename
A
C

R(ABC) ⋈ S(ABP)

R natural join S

R join S on R.A = S.A and R.B = S.B

Outer Join

Faculty

Fid	Fname
1	A
2	B
3	C

Courses

cid	cname	fid
99	DBMS	1
100	CD	1
101	TOC	3
104	CDS	

(o/p)

Fname	Cname
A	DBMS
A	CD
B	null
C	TOC

(o/p)

Cname	Fname
DBMS	A
CD	A
TOC	C
CDS	null

Outer join is used when we want to output all rows from one table even if it does not have a corresponding row in another table.

- Left outer join
- Right outer join
- Full outer join

Left outer join :- Returns all rows from the first table even if there are no-matches in the second table.

Q: Find names of all faculty and courses they teaches (if any).

> select fname, cname
from faculty Natural left outer join Courses ;

(o/p)

Fname	Cname
A	DBMS
A	CD
B	null
C	TOC

<u>Right - outer join</u> :- Returns all the rows from the second table even if there are no-matches in the first table.

Q. find all courses and name of the faculty if teaches the courses.

> Select course_name, faculty-name
from faculty Natural right outer join courses;

(o/p)

Cname	Fname
DBMS	A
CD	A
TOC	C
CDS	null

<u>Full outer join</u> :- Returns all the rows from both the tables even if there is no-matching row appearing in another table.

FOJ = { LOJ } ∪ { ROJ }

> Select cname, fname
from faculty Natural full outer join courses;

o/p

cname,	Fname
DBMS	A
CD	A
TOC	C
CDS	null
null	B

Self Join

It is a join in which a table can be joined by itself

Employee

eno	ename	salary	mgreno
1	kiran	9000	—
2	John	6000	1
3	Rajesh	6000	1
4	Mahesh	4000	2

Q. find em_name and his manager. ?

> Select e.ename ename, m.ename manager
from Employee e, Employee m
where e.mgreno = m.eno;

(O/p)

e.name	manager
John	kiran
Rajesh	kiran
Mahesh	John

Qu Write a query to find no. of employees working under kiran. ?

> Select ~~ename~~ count (*)
from Employee e, Employee m
where m.eno = e.mgreno and m.ename = 'kiran';

(O/p)

count (*)
2

Note :-
Table aliases are mandatory in self joins.

Nested Query

A Query inside a query is called nested query

Supplier (sid, Sname, City, turnover)

Supply (sid, partid, qty)

Catalog (Partid, Pname, Color)

Q. Find name of the supplier who is supplying part-id 99.

> Select Sname
from Supplier, Supply
where Supplier.sid = Supply.sid and ~~Partid~~ Supply.partid = 99 ;

Supplier

sid	sname
1	A
2	B

m

Supply

sid	Partid
1	98
1	99
2	98

n

O/p

Sname
A

(m×n) - condition product is generated.

⟹ The memmory is utilized more.

So we have to go for nested queries

```
> Select sname
  from supplier
  where sid in ( Select sid
                 from supply
                 where partid = 99 ) ;
```

o/p

sname
A

here $(m+n)$ touples are considered which is very few considered with $(m \times n)$ of previous join

In nested queries the inner query is evaluated first and the result is supplied to its outer queries where inner query is indipendant and outer query depends on result of inner query.

Q. Write a query to find name of the suppliers who supplies blue color parts.

```
> Select sname
  from supplier
  where sid
    in ( select sid
         from supply
         where partid in (Select partid
                          from catalog
                          where color = 'Blue'))
```

Note:-
Nested queries are evaluated from bottom to top.

Q. find name of the ~~supplies~~ ~~other supplies~~ parts supplied by supplier called A ?

> Select pname
from catalog
where ~~partid~~ in (Select partid
from Supply
where sid in (Select sid
from supplier
where sname = 'A'));

Q. find name of the supplier who has maximum turn over.

> Select sname
from supplier
where Turn over = (Select max(turnover)
from supplier);

Q. consider the following SQL query

```
Select sname
from supplier
where sid not in ( Select sid
                   from supply
                   where partid not in ( Select partid
                                          from catalog
                                          where color <> 'Blue'));
```

Which of the following is the correct interpretations of the above query.

a) find the names of all suppliers who have supplied a non-blue part

b) find the names of all suppliers who have not supplied a non-blue part

c) names of all suppliers who have supplied only blue part

d) find the names of all suppliers who have not supplied only blue part.

Supplier

Sid	sname	TO
1	A	8L
2	B	5L
3	C	2L

Supply

Sid	partid
1	98
1	99
2	98
3	99

Catalog

Partid	color	Part
98	Red	P
99	Blue	Q

a)
A
B

b)
A
C

c)
C

d)
B

{ 2-5.30 - TOC
6-9.00 - DBMS }

MO

Correlated Subquery

(Outer query (Inner query))

In co-related subquery both outer and inner query are evaluated simultaneously, ie, for each row of outer query all the rows of inner query are evaluated, based on the result the tuple of the outer query is selected for the output.

The co-related sub-queries are executed from TOP - BOTTOM - TOP in this order.

Q: find name of the suppliers who supplies part_id = 99

Supplier

Sno	Sname
1	A
2	B

Supply

Sno	partid
1	98
	99
2	98

```
> Select S.Sname
  from Supplier S
  where 99 in ( Select sp. partid
               from Supply sp
               where s.sno = sp.sno ) ;
```

op

s.sname
A

Student S1

Name	marks
A	100
B	500
C	300
D	400
E	200

Student S2

Name	marks
A	100
B	500
C	300
D	400
E	200

S1.marks <= S2.marks

```
> Select S1.Name
  from Student S1
  where 3 = ( select count (distinct marks)
              from Student S2
              where S1.marks <= S2.marks )
```

If there is duplicate marks

O/p S1.Name

S1.Name
C

Q) consider the relation Book (Title, Price) contains the titles and prices of different books assuming that no two books have the same price. what does the following SQL query list?

```
Select title
from Book B
where (select count(*)
       from Book T
       where T.price > B.price) < 5;
```

0
1
2
3
4

a) Titles of four most expensive books
b) Titles of fifth most expensive book
c) Titles of fifth most inexpensive book
d) Titles of five most expensive books

Note :- when ever the inner query uses the reference of outer query then both the queries are said to be co-related.

Book B

Title	Price
A	100
B	200
C	300
D	400
E	500
F	600

Book T

Title	Price
A	100
B	200
C	300
D	400
E	500
F	600

for A - 5
B - 4
C - 3
D - 2
E - 1
F - 0

5 most expensive books displayed.

d: B C D E F

a: C D E F

b: B

c: E

O/p :

Title
B
C
D
E
F

Exists operator

- Exists operator is used for testing whether a given set is empty or not. an Exists opereator on empty set returns false, while on non-empty set returns true.

exists (result) = true
exists (   ) = false

Q. Find name of the supplier who supplies , part_id 99.

(P.T.O)

```
> Select Sname
  from Supplier S
  where exists ( Select *
                 from Supply SP
                 where S.Sno = Sp.Sno and Sp.partid = 99 );
```

O/p:

S.Sname
A

Q)

A

P	Q	R
0	a	65
1	b	66
2	c	67
3	d	69

B

P	S	T
0	A	82
1	A	81
2	S	82
5	A	80
1	S	83
3	A	84

Consider the above table of data and result of the following SQL query

```
> Select P
  from B
  where S='A' and exists ( Select *
                           from A
                           where R>65 and B.P = A.P );
```

B	O/P
0 - x	1
1 - ✓	1
2 - x	3
5 - x	
1 - x	
3 - ✓	

Set comparison Operators

op any ( ) .... ex:- $x <$ any $(5, 10, 15)$

$(x<5)$ or $(x<10)$ or $(x<15)$

op all ( ) .... ex:- $x <$ all $(5, 10, 15)$

$(x<5)$ and $(x<10)$ and $(x<15)$

SQL supports set comparison operators op any & op all where 'op' is any valid arithmatic comparison operators.

op- any

Compares a value with each value in a set and returns true if any value is compared according to given conditions.

op- all

Compares a value with each value in a set and returns true if the given condition satisfied for every value in the set.

Q: find name of the supplier whose turn over is better than the turn over of some ~~suppliers of~~ suppliers of Hyderabad

Supplier

Sno	Sname	City	Turnover
1	A	Hyd	5L
2	B	Bang	4L
3	C	Hyd	5L
4	D	Delhi	6L

> Select sname
  from Supplier
  where City<>'Hyd' and
  turnover> any (Select turn_over
  from Supplier
  where City='Hyd');

O/p:

Sname
B
D

Q. find name of the suppliers whose turn_over is better than the turnover of all suppliers of 'Hyd'.

> Select sname
  from Supplier
  where city <> 'Hyd' and turn_over > all (Select turn_over
  from Supplier
  where City =' Hyd');

O/p:

Sname
D

Q. Consider the following tables

Table1

T1A	T1B
a	aa
b	bb
c	cc

Table2

T2A	T2B	T2C
a	a	a
b	a	null

Q. find the no. of rows returned by the each of the following SQL Query.

(p.t.o)

a) Select *
From Table1
where T1A = all (select T2B
From Table2
where T2B >= 'b');

Ans: 2 - rows returned

= all ( );

b) Select *
From Table1
where T1B in (Select T2A
from Table2
where T2C is null);

Ans: 0 - rows returned

c) Select *
from Table1
where T1A in (Select T2C
from Table2);

Ans: 1 - row returned

d) Select *
from Table1
where exists (Select Count (*)
from table2
where T2B = 'x');

Ans: 3 - rows returned

e) select *
from Table1
where not exists (select *
from Table2
where T2C >= 'a' and T2C <> T1A);

Ans: 1 - rows returned

f) Select *
from Table1
where T1A = all (select T2C
from Table2
where T2C >= 'x');

Ans: 3 - rows returned

Qs
Select S.sname
from Sailors S
where not exists ((select B.bid from Boats B) {100, 200, 300} - {100, ... 300}
except
(select R.bid from Reserves R where R.sid = S.sid)

Sailors (Sid, Sname)

Sid	Sname
1	A
2	B
3	C

Reserves (Sid, bid)

Sid	bid
1	100
1	200
1	300
2	200

Boats (Bid, bname)

Bid	bname
100	BB
200	BBB
300	BBBB

The above query returns ________

a) names of sailors who have reserved any boat.
b) names of sailors who have not reserved any boat
c) names of sailors who have reserved all boats
d) none

consider the following relationschema where the primary keys are shown, underlined.

Student (rollno, name, address) ------- 120

Enroll (rollno, courseno, coursename) ---- 8

The no. of tuples in the Student and enrolled tables are 120 and 8 respectively. what are the maximum and minimum no. of tuples that can be present in (Student natural join enroll)

a) 960, 8

b) 960, 120

c) 120, 8

d) 8, 8

Note:- ~~Normal~~

when two tables are joined (natural join) w.r.t primary key and foreign key the no. of tuples present in the resulting relation is always equals to tuples in foreign key relation.

Qu Consider the following table

A	B	C
P	S	10
Q	R	5
R	S	7

? select count (*)

from (

(select A, B from Table 1) as X

natural join

(select B, C from Table1) as Y)

);

A	B
P	S
Q	R
R	S

B	C
S	10
R	5
S	7

5

The result of the above query is ——

a) 3
~~b)~~ 5
c) 6
d) 9

P-102

Staff (Staff No, name, dept, Skill Code)

Skill (Skillcode, description, chargeOutRate)

Project (ProjectNo, StartDate, end Date, budget, Project manager Staffno)

Booking (StaffNo, projectNo, date workedon, timeworkedon).

1)
```
Select *
from Skills
where chargeoutrate > 60
Order by Description;
```

2)
```
Select *
from Staff
where dept = 'Special projects' and skill code =
        ( Select skillcode
          from Skills
          where description = 'programmer');
```

3)

2pm - 5:30pm TOC
6pm - 9:00 DBMS

(3)

```
> Select S.name, B.projectNo, B.dateWorkedOn, B.timeWorked on
from Staff S, Booking B
where B.project no in (Select projectno
                       from projects
                       where startDate <= 01-July-1995 and
                             endDate >= 31-July-1995)
                       and S.staffno = B.staffno
Order by S.name, B.project No, B.date worked on;
```

4)

```
Select count(*)
from Staff
where Skillcode in (Select Skillcode
                    from skill
                    where Description = "Programmer");
```

35)

```
Select *
from Projects
where projectNo in (Select projectNo
                    from Booking
                    Group by Project No
                    having count(*) ≥ 2);
```

(OR)

```
Select *
from projects P
where ( select count(*)
        from Booking B
        where P.project = B.projectno ) >= 2);
```

(05)
```
Select avg(chargeOut Rate)
from skill;
```

(07)
```
Select * from staff where skill Code
in (Select skill Code
      from skill
      where chargeOutRate > (Select avg(chargeOutRate)
                             from skill));
```

Relational Algebra

Relational algebra is a procedural language. Queries in Relational algebra are composed using a collection of operators and each query describes a step by step procedure for computing the desired answer.

ie, Relational algebra expression represents a query evaluation plan.

Operators in Relational Algebra

1) Selection (σ) - "Rows"

2) Projection (π) - "Columns"

Student (sno, name, age);

① find all students of age above '16' ?

$$\sigma_{age>16} \text{ Student}$$

② Display the names of all Students

$$\pi_{name} \text{ Student}$$

③ Find names of all students of age above '16'.

$$\pi_{name}\left(\sigma_{age>16} \text{ Student}\right)$$

Consider the following ~~statement~~ statement

S1: $\pi_{name} (\sigma_{Age>16} \text{ Student})$

S2: select name
from student
where age > 16;

which of the following is true about the results of S1 & S2 alwa

a) $S_1 = S_2$

b) $S_1 \subseteq S_2$

c) $S_2 \subseteq S_1$

d) $S_1 \neq S_2$

Student

name	age
A	17
B	18
A	20

O/p

S_1	S_2
A	A
B	B
	A

Note: Relational algebra assumes duplicate elimination is implicit

Q: which of the following is true about an SQL Query?

Select l from R where C;

a) $\pi_c (\sigma_l R)$

b) $\sigma_l (\pi_c R)$

c) $\sigma_c (\pi_l R)$

d) $\pi_l (\sigma_c R)$

3) Set - Operations

$$R \left\{ \begin{matrix} \cup \\ \cap \\ - \end{matrix} \right\} S \quad \text{compatible}$$

Relational algebra supports ~~special op~~

(i) Set union (∪)
(ii) Set intersection (∩)
(iii) Set difference (−)

between the results of '2' relational algebra expression if they are compatible

Depositor (cust_name, ac_no)
Borrower (cust_name, loan_no)
Loan (loan_no, branch_name, City, amount);

• Find name of the customer who have an account or loan or both at the bank

$$(\pi_{customer} \text{ Depositor}) \cup (\pi_{cust\_name} \text{ Borrower})$$

∩ ← who have both account and loan

− ← who have an account but no loan

4) Cross Product (x)

R×S - returns a relation instance whose schema contains all the fields of R followed by all the fields of S. The result of R×S contains a tuple ⟨R, S⟩ for each r ∈ R, s ∈ S.

Q) find name of the customer who have a loan in ABIDS branch.

$$\pi_{cust\_name}\left(\sigma_{Borrower.Loan\_no = Loan.Loan\_no \wedge Loan.branch = 'Abids'}(Borrower \times Loan)\right)$$

5) Rename (ρ_{rho})

ex:-

1. $\rho\,[A, (Borrower \times Loan)]$
2. $\rho\,[B, (\sigma_{B.Loan\_no = L.Loan\_no \wedge L.branch\_name = 'ABIDS'}\ A)]$
3. $\pi_{Cust\_name}\ B$

Rename operator is used to represent a relational algebra expression with a shorter name.

JOINS

Join is defined as a cross product followed by selection then projection.

Variants of Joins

1) Conditional join ($\bowtie_c$)

Conditional join is a join in which the two relations are joined based on some condition.

ex:

$$\pi_{Cust\_name}\left[\sigma_{Branch\_name = 'ABIDS'}\left(Borrower \bowtie_{Borrower.Loan\_no = Loan.Loan\_no} Loan\right)\right]$$

2) Equi Join ($\bowtie_=$)

Equi join is a join in which the join condition must be an equality of the form. $R \underset{R.A = S.P}{\bowtie} S$

$R \underset{R.A=S.P}{\bowtie} S$	
A B C	P Q R
1	1
2	2
⋮	⋮
x	x

$\simeq \left(\pi_{A,B,C,Q,R} \left(\sigma_{R.A=S.P} \, R \times S \right) \right)$

Projected out

Note :-

Every equi join is called a conditional join, but reverse is not true always.

3) Natural join ($\bowtie$)

Natural join is a join in which the join condition is implicit a equality on all columns having the same name.

R
A B Q

S
A P Q

$R \bowtie S$
A B Q A P Q

$\Rightarrow \pi_{ABQP} \left(\sigma_{R.A=S.A \,\wedge\, R.Q=S.Q} \left(R \times S \right) \right)$

projected out

ex:- $\pi_{cust\_name} \left[\sigma_{branch\_name = Abids} \left(Borrower \bowtie Loan \right) \right]$

Q. Table 1 T_1

P	Q	R
11	a	b
16	b	9
26	a	7

Table 2 T_2

A	B	C
11	b	7
26	c	4
11	b	6

Consider the tables shown above. What is the no of tuples in the result of the given algebraic expression

i) $T_1 \bowtie_{T_1.P = T_2.A} T_2$ $\Rightarrow$ No. of tuples = 3

ii) $T_1 \bowtie_{T_1.Q = T_2.B} T_2$ $\Rightarrow$ No. of tuples = 2

iii) $T_1 \bowtie T_2$ $(T_1.P = T_2.A$ and $T_1.R = T_2.C)$ $\Rightarrow$ No. of tuples = 1

iv) $T_1 \bowtie T_2$ $\Rightarrow$ No. of tuples = 9

Note :- If there is no column in common, Natural join results in a cross prod

Q. Consider the following table

A

ID	Name	age
12	A	60
15	S	24
99	R	11

B

ID	Name	age
15	S	24
25	H	40
98	T	20
99	R	11

C

id	phone
10	2200
99	2100

$$(A \cup B) \bowtie_{A.Id > 40 \vee C.Id < 15} C$$

Q. find the no. of rows returned by the above relational algebra expression. assume that the schema of $A \cup B$ is same as that of A.

a) 7
b) 4
c) 5
d) 9

Q.

Let $R_1(\underline{A}BC)$ and $R_2(\underline{D}E)$ be two relationship schemas were the primarykeys are shown underlined. and let C - be a foreign key in R_1 refering to R_2. Suppose there is no-violation of referential integrity constraint in the corresponding relation instances r_1 and r_2 which of the following relational algebra expressions would necessarily produce an empty result?

a) $\pi_D(r_2) - \pi_C(r_1)$

✓ b) $\pi_C(r_1) - \pi_D(r_2)$ $\{F.K\} - \{P.K\} = \underline{\underline{\{\}}}$

c) $\pi_D(r_1 \bowtie_{C \neq D} r_2)$

d) $\pi_C(r_1 \bowtie_{C \neq D} r_2)$

2pm - 5:30 - TOC } $\frac{3}{2}$

Division Operation (÷)

Find the name of students who have registered for all courses?

Student

rno	name
1	A
2	B

registration

fee	rno	cno
5k	1	99
2k	2	99
5k	2	100

Courses

cno	cname
99	DB
100	TOC

$$\pi_{name}\left[\text{Student} \bowtie \left[\left(\pi_{rno,cno}\ \text{Registration}\right) \div \left(\pi_{cno}\ \text{Courses}\right)\right]\right]$$

o/p:

name
B

Consider two relation instances A and B in which A has two fields X and Y and B has just one field Y with the same domain as in A. We define the division operator $A \div B$ as the set of all X values such that for every Y value in B there is a tuple X, Y in A.

A (X Y)

X	Y
x_1	Y_1
X_1	Y_2
X_2	Y_2
X_3	Y_1
X_1	Y_3
X_2	Y_1

B_1 (X)

Y_1

B_2 (X)

Y_1
Y_2

B_3 (X)

Y_1
Y_2
Y_3

$A \div B_1$ (X)

X_1
X_2
X_3

$A \div B_2$ (X)

X_1
X_2

$A \div B_3$ (X)

X_1

Qus

find name of the customer who have a loan in all branches of Hyd.

Borrower (cust_name, loan_no)

Loan (loan_no, branch_name, city, amount)

Ans

$$\left[\pi_{\text{cust\_name, branch\_name}} \left(\sigma_{\text{city}=\text{Hyd}} (\text{Borrower} \bowtie \text{Loan}) \right) \right] \div \left[\pi_{\text{branch\_name}} \left(\sigma_{\text{city}=\text{Hyd}} \text{Loan} \right) \right]$$

Qus find name of the publisher who published all category of Books?

Books (ISBN, title, Category, Price, Pub_Id)

Publisher (Pub_Id, pname, email)

Ans

$$\left[\pi_{\text{pname, Category}} (\text{Books} \bowtie \text{Publisher}) \right] \div \left[\pi_{\text{Category}} (\text{Books}) \right]$$

Q:- Consider two tables R_1 and R_2 with N_1 and N_2 rows where $N_2 > N_1$ find min and maximum rows returned for each of the following ~~exp~~ expressi

Ans

	expression	min	max
1)	$\sigma_{\text{age}>20} R_1$	0	N_1
2)	$\pi_{\text{name}} R_2$	1	N_2
3)	$R_1 \cup R_2$	N_2	N_1+N_2
4)	$R_1 \cap R_2$	0	N_1
5)	$R_1 - R_2$	0	N_1
6)	$R_1 \bowtie R_2$	0	$N_1 * N_2$

Qu Consider the following Relations R(ABC) S(BDE) with the following functional dependencies A → C B → A, the relation R contains 200 tuples and S contains 100 tuples what is the maximum no. of tuples possible in $R \bowtie S$?

F.K

Ans

100

Qu Let r be a relation instance of Schema R(ABCD) we define $r_1 = \pi_{ABC}(r)$ and $r_2 = \pi_{AD}(r)$. Let $S = r_1 \bowtie r_2$ assuming that the decomposition of r into r_1 and r_2 is lossy then which of the following is true

a) $S \subset R$ b) $S = r$

c) $r \subset S$ d) $r * S = S$

⑥ The decomposition generates spurious tuples if it is lossy

Relational Calculus → (Based on predicate Calculus)

Relation calculus :- In this a query describes the result set without specifying how the answer is to be computed. This non-procedural style of querying is called relational calculus.

Two - variants of relational calculus

Tuple relational Calculus (TRC)

→ Query describes results in the form of set of tuples

→ Tuple variable :-
It is a variable that takes on tuples of a Relation schema as values.

→ Form of Query

$\{ T / P(T) \}$

Tuple Variable (T)

formula which describes Tuple Variable ($P(T)$)

Domain relational Calculus (DRC)

→ Query describes results in the form of set of columns (Domain)

→ Domain Variable :-
It is a variable that ranges over the domain of some attributes.

→ form of Query

$\{ \langle x_1\, x_2 \ldots x_n \rangle \mid P(x_1, x_2 \ldots x_n) \}$

Domain Variables ($\langle x_1\, x_2 \ldots x_n \rangle$)

formula which describes the domain ($P(x_1, x_2 \ldots x_n)$)

Borrower (cust_name , loan_no)
(Cn, L.b)

Loan (loan_no, branch_name, amount)
(Ln, Bn, Am)

Qn Find all the loans of an amount above 5000

TRC: $\{ T / T \in Loan\ (T.amount > 5000) \}$

DRC: $\{ <loan\_no, branch\_name, amount> / <loan\_no, branch\_name, amount> \in Loan\ (amount > 5000) \}$

Qn Display loan_no and amount of all loans

TRC: $\{ T / \exists L \in Loan\ (T.loan\_no = L.loan\_no \wedge T.amount = L.amount) \}$

DRC: $\{ <loan\_no, branch\_name> / \exists branch\_name\ (<loan\_no, branch\_name, amount> \in Loan) \}$

Qn Find name of the customers who have a loan in Abids branch and display the loan amount also.

TRC: $\{ T / \exists B \in Borrower\ (T.customer\_name = B.cust\_name) \wedge \exists L \in Loan\ (L.branch\_name = \text{'ABIDS'} \wedge B.Loan\_no = L.Loan\_no \wedge T.amount = L.amount)) \}$

(OR)

$$\{ T \mid \exists L \in Loan\ (L.branch\_name = 'ABIDS' \wedge T.amount = L.amount$$
$$\wedge\ \exists B \in Borrower\ (B.loan\_no = L.loan\_no$$
$$\wedge\ T.cust\_name = B.cust\_name)) \}$$

DRC:

$$\{ <C_n, A_m> \mid \exists L_b (<C_n, L_b> \in Borrower \wedge$$
$$\exists L_n, B_n (<L_b, B_n, A_m> \in Loan (B_n = 'ABIDS'$$
$$\wedge\ L_n = L_b))) \}$$

Que what is the result of the following TRC expression?

Q:-

$$\{ T \mid \exists S \in Student\ (T.sname = S.sname \wedge$$
$$\exists C \in Course\ (C.Sno = S.sno \wedge C.Course\_name = 'cs')) \}$$

Ans

It finds the names of all students studying the course 'cs'.

Que Translate the following TRC expressions into relational algebra.

$$\{ T \mid T \in R\ (T.A = 10 \wedge T.B = 20) \}$$

Ans $(\sigma_{A=10} R) \cap (\sigma_{B=20} R)$ (OR) $(\sigma_{A=10 \wedge B=20} R)$

① R(ABCDE) R is in 1NF

$f: \{\underbrace{AB \to C}_{BCNF}, \underbrace{C \to D}_{2NF}, \underbrace{B \to E}_{1NF}\}$ C.K: AB

$B^+ = \{B, E\}$ R_1 ✓ BCNF

(ABCD) R_2 ✓ 2NF $\xrightarrow{3NF}$ $C^+ = \{CD\}$ R_3 ✓ BCNF

{ABC} R_4 ✓ BCNF

R in BCNF

$R_1(BE)$: $f_1: \{B \to E\}$

$R_3(CD)$: $f_2: \{C \to D\}$

$R_4(ABC)$: $f_3: \{AB \to C\}$

Decompossition is lossless & D.P

② R(ABCDEF)

$f: \{\underbrace{A \to BC}_{BCNF}, \underbrace{BC \to AD}_{BCNF}, \underbrace{B \to F}_{1NF}, \underbrace{D \to E}_{2NF}\}$

C.K = A, BC

$B^+ = \{BF\}$ R_1 ✓ BCNF

(ABCDE) R_2 ✓ 2NF $\xrightarrow{3NF}$ $D^+ = \{DE\}$ R_3 ✓ BCNF

{ABCD} R_4 ✓ BCNF

BCNF
+
Lossless - join
+
D.P

③ R(ABCDE)

$f: \{ \underset{BCNF}{A \to BC}, \underset{BCNF}{CD \to E}, \underset{3NF}{B \to D}, \underset{BCNF}{E \to A} \}$

C.K A
E
CD
BC

R is in 3NF

$B^+ = \{BD\} R_1$ BCNF $f_1\{B \to D\}$

$(ABCE) R_2$ BCNF $f_2: \{A \to BCE$
$BC \to AE$
$E \to ABC\}$

BCNF
+
lossless join
+
CD^+ using $f_1 \cup f_2$
$CD^+ = \{CD\}$
$CD \to E$ lost

Not D.P

Because $CD \to E$ is not in R_1 or R_2 than to preserve

Dependency take $R_3(CDE)$ $f_3: \{CD \to E\}$

④ R(ABCD)

$f: \{ \underset{BCNF}{AB \to CD}, \underset{3NF}{D \to A} \}$

C.K AB
DB

$D^+ = \{DA\} R_1$ BCNF $f_1\{D \to A\}$

BCNF
+
Lossless join

$(BCD) R_2$ BCNF $f_2\{BD \to C\}$

take $AB \to C$ in $R_3(ABC)$ BCNF $f_3\{AB \to C\}$

using $f_1 \cup f_2$: $BD^+ = \{BDAC\}$

$AB^+ = \{AB\}$

$AB \to CD$ is lost

take $AB \to D$ in $R_4(ABD)$ $f_4\{AB \to D$
$D \to A\}$

R_4 is 3NF

R_6(BD) BCNF — AB → D is lost

Note:- Dependency preserving decomposition into BCNF always may not be possible ~~always~~.

observation

If ~~the~~ a relation contains overlapping candidate keys. Dependency preserving BCNF decomposition may-not be possible.

Note: ① Relation schema R with no-non-trivial functional dependencies are always in BCNF.

② A Relation is failed in BCNF only if there should be atleast one non-trivial dependency X → Y ~~when~~ with X as not a Superkey.

③ Relation R with only 2-attribute is always in BCNF

④ If Relation R consists only of simple candidate keys then R always in 2NF.

⑤ If Relation R consists only of prime attribute then R - always in 3NF

⑥ If relation consists of only simple candidate keys and R is in 3NF then R must be in BCNF

(S-A) 2-5:30 NA -2
C-2. DBMS -3

Concurrent execution

(Interleaving execution of the operations of a Transaction)

Why?

1) For avoiding longer waiting time
2) If transaction consist of multiple steps. Some involves I/o activities and other involve CPU activities. In a computer system CPU and I/o operations can be done in parallel. therefore I/o activity can be done in parallel with processing at the CPU
3) The processor and Disk utilization increases.

Schedule

It represents the order in which instructions of a transactions are executed.

The problems due to concurrent execution of the transactions

① **Lost Update problem. (W-W conflict)**

$A \xrightarrow{50} B$ T_1	$A \xrightarrow{4\%} A$ T_2
Read (A)	
A = A−50	
	Read (A)
	X = A * 0.04
	A = A+X
Write(A)	
	Write(A)
Read (B)	
B = B+50	
Write (B)	

$\frac{A}{1000}$

$\frac{B}{200}$

Suppose that the operations of T_1 and T_2 are interleaved in such a way that T_2 reads the value of account A before T_1 updates, now when T_2 updates the value of account A in the data base, the value of account A updated by transaction T_1 is over written and hence is lost. This is known as lost update problem

Test:-

If there is any two write operation of different transactions, between that there is no read operation the second write operation overwrites the first write hence it causes loss of first updation

2) Dirty Read problem (W-R conflict)

T_1	T_2
Read (A) A = A − 50 write(A)	
	Read (A) X = A * 0.04 A = A + X write (A)
Read (B)	
B = B + 50 write (B)	

Rollback

fail

A
~~1000~~
~~950~~
~~988~~
1000

B
2000

Reading an un-committed data is called dirty read operations.

③ Unrepatable read problem (or) Non-repeatable read problem (R-W conflict)

~~when 2 transactions~~

T1	T2
Read (A)	
	Read (A)
	A = A - 15,000
	Write (A)
Read (A)	
A = A - 10,000	
Write (A)	

A

20,000

fail

When a transaction tries to read the value of a data item twice and another transaction updates the same data item inbetween the two read operations of the first transaction, as a result the first transaction reads varied values of same data item during its execution this is known as un-repeatable reads.

④ phantom tuple (Phantom phenomenon)

T1	T2
Select * from Emp where Salary > 3000	
	Insert into Emp Values (4, D, 3500);
Select * from Emp where Salary > 3000;	

eno	ename	Salary
1	A	5000
3	C	4000

eno	ename	salary
1	A	5000
3	C	4000
4	D	3500

Employee

eno	ename	Salary
1	A	5000
2	B	2000
3	C	4000
4	D	3500

Transaction T_1 may read set of rows from a Table based on some condition. now suppose that a transaction T_2 inserts a new row that also satisfies the ~~where clause~~ condition of T_1.
If T_1 is repeated then T_1 will see a row that previously did not exists called a phantom tuple.

5) Incorrect Summary Problem

T_1	T_2
	Sum = 0 Read (K) Sum = Sum + K;
Read (X) X = X + 500 write (X);	
	Read (X) Sum = Sum + X; Read (Y) Sum = Sum + Y;
Read (Y) Y = Y + 200 write (Y);	

	Before T_1	After T_1
K =	50	50
X =	100	600
Y =	200	400
	350	1050

correct o/p's

O/p : 850 - incorrect

50 + 600 + 200

~~When a transaction tries to read the value of data~~

If one transaction is calculating an aggregate summary function on a number of records while other transaction updating some of these records the aggregate function may calculate some values before they are updated and others after they are updated results in incorrect summary.

Characterizing schedules

1) Serial schedule :-

$T_1 \longrightarrow T_2$

T_1	T_2
R(A)	
A = A+50	
W(A)	
	R(A)
	A = A+100
	W(A)

$T_2 \longrightarrow T_1$

T_1	T_2
	R(A)
	A = A+100
	W(A)
R(A)	
A = A+50	
W(A	

consistant states of o/p.

T_1	T_2
R(A)	
A = A+50	
	R(A)
	A = A+100
W(A)	
	W(A)

inconsistant state of o/p

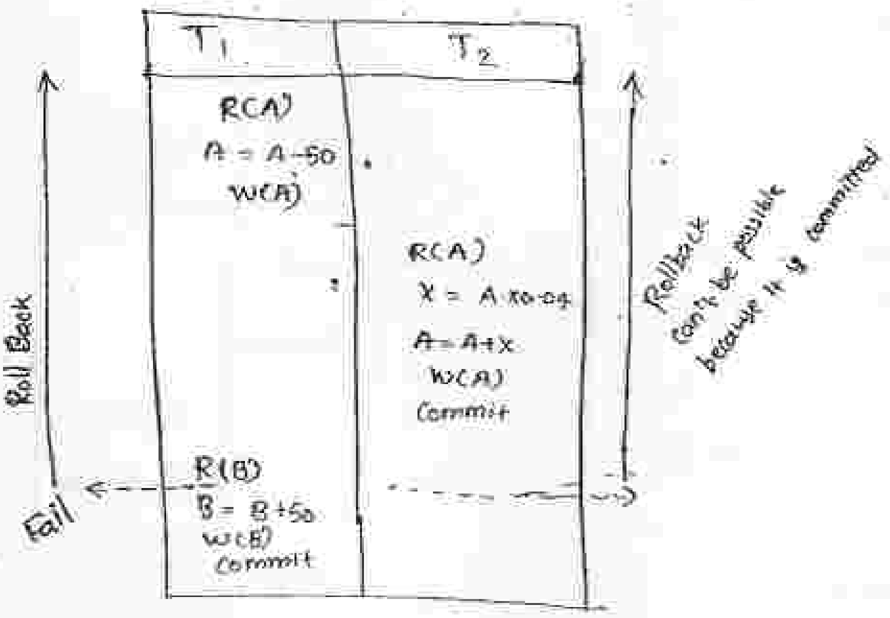
- Serial schedules that does not interleave the actions of any operations of different transaction.
- when transactions are executing serially which always ensures a consistant state.
- If there are n- transactions in a schedule the possible no. of serial Schedules are n! (n! consistant states are possible)

2) Complete schedule :-

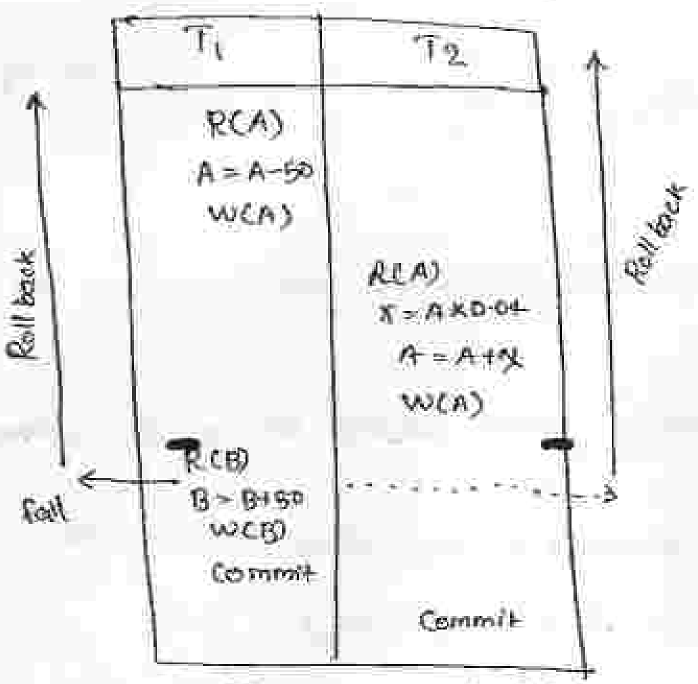
T1	T2
R(A)	
	R(A)
A = A-50	
W(A)	
Commit	
	A = A+50
	W(A)
	abort

A schedule is said to be complete if the last operation of each transaction is ~~complete~~ either abort or commit

3) Recoverable schedule



Non-recoverable schedule.



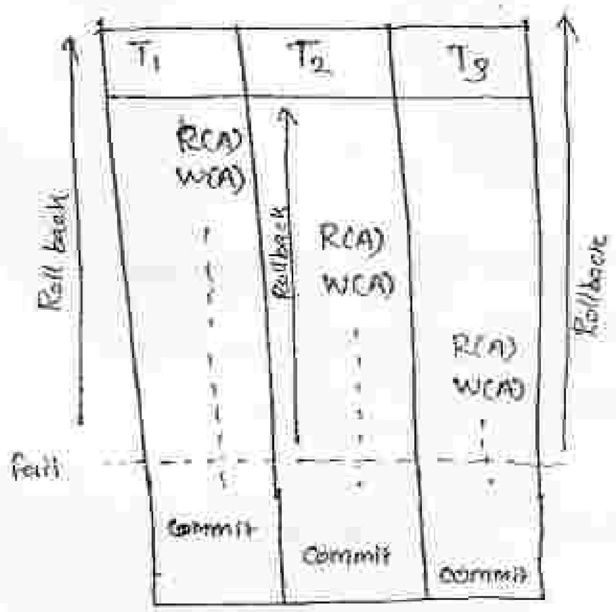
Recoverable schedule

A recoverable schedule is one where for each pair of transactions T_i and T_j such that T_j reads a data item that was previously written by T_i then the commit operation of T_i should appear before commit operation of T_j

4) Cascade less schedule

Cascading aborts:-

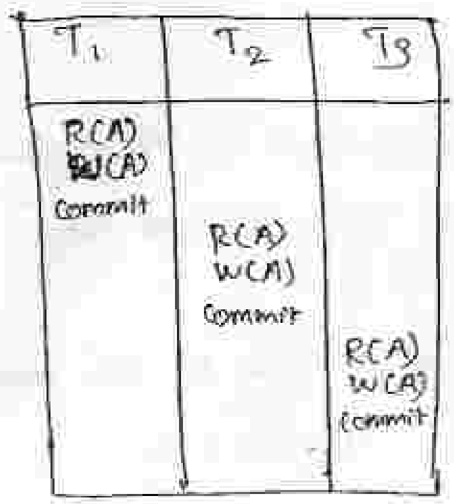
If one transaction failure causes multiple transactions to rollback, it is called cascading rollback or cascading aborts.



Cascading aborts

Cascadeless schedule:-

A cascadeless schedule is one, where for each pair of transactions T_i and T_j such that T_j reads a data item that was written by T_i than the commit operation of T_i should appear before the read operation of T_j.



Cascadeless schedule

ii) Strict schedule

T_1	T_2
R(A)	
W(A)	
Commit	
	W(A)

Schedule that is strict if a value written by a transaction can-not be read or over-written by other transactions until the transaction is either aborts or commits.

Every strict schedule is both recoverable and cascadeless

19-12-13

3) Equivalent schedule

The two schedules S_1 and S_2 are said to be equivalent schedules if they produce the same final database state.

(1) Result equivalent schedules

Two schedules are said to be result equivalent if they produce the same final database state for some initial values of data.

A	S_1	S_2	A
100	R(A)	R(A)	100
110	A = A+10	A = A*1.1	110
110	W(A)	W(A)	110

S_1 and S_2 are result equivalent for A = 100

Qu. Is the following schedules are result equivalent for the initial value of X and Y is (2,5) respectively.

S1

T_1	T_2
R(X)	
X = X+5	
W(X)	
	R(X)
	X = X+3
	W(X)
R(Y)	
Y = Y+5	
W(Y)	

X	Y
~~2~~	~~5~~
~~7~~	10
21	

S2

T_1	T_2
	R(X)
	X = X+3
	W(X)
R(X)	
X = X+5	
W(X)	
R(Y)	
Y = Y+5	
W(Y)	

X	Y
~~2~~	~~5~~
~~5~~	10
11	

{ S_1 and S_2 are not result equivalent }

② **Conflict equivalent**

__Conflict Operations__

T_i	T_j
R(A)	W(A)
W(A)	R(A)
W(A)	W(A)

} Conflict operations

R(A) R(A)
Operating on diff. data } non-conflicting operations.

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

S1

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

R(A) ... W(A)
W(A) ... R(A)

S2

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

⊙ S_1 is conflict equivalent to S_2

$S_1 \; \underline{C} \; S_2$

Qu Test which of the following schedules are conflict equivalent? 148

S1: $R_1(A)$ $R_2(B)$ $W_1(A)$ $W_2(B)$
S2: $R_2(B)$ $R_1(A)$ $W_2(B)$ $W_1(A)$

} no conflict is there in both schedules, $\boxed{S_1 \overset{c}{=} S_2}$

S1: $R_1(A)$ $W_1(A)$ $R_2(B)$ $W_2(B)$ $R_1(B)$

S2 $R_1(A)$ $W_1(A)$ $R_1(B)$ $R_2(B)$ $W_2(B)$

$\boxed{S_1 \not\equiv S_2}$

S_1 not conflict equivalent to S_2

S_1

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

$W_2(B) \rightarrow R_1(B)$

S_2

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

$R_1(B) \rightarrow W_2(B)$

S_1

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

S_2

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

$\Rightarrow$ $\boxed{S_1 \overset{c}{=} S_2}$

Conflict Serializable

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

S_1

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

S_2

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

S_1 is conflict serializable to serial schedule $T_1 \rightarrow T_2$

S_1

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

S_2

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

S_3

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

$R_2(B) \rightarrow W_1(B)$ $W_1(B) \rightarrow R_2(B)$ $S_1 \not\equiv S_3$

$S_1 \not\equiv S_2$

$\Longrightarrow$ S_1 is not conflict serializable.

Note:- A schedule that is conflict serializable must ensure consistancy of the database.

Test for conflict serializability using precedence graph

① construct a directed graph where each vertices corresponds to a transaction and each directed edge represents a conflicting operation.

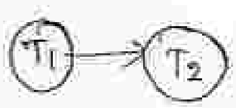
② If the directed graph contains cycles then the concurrent schedule is not conflict serializable.

③ If the graph contains no-cycle then the schedule is called conflict serializable.

The serializability order is determined based on the directed edges.

Ex:-

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



S_1 is C.S

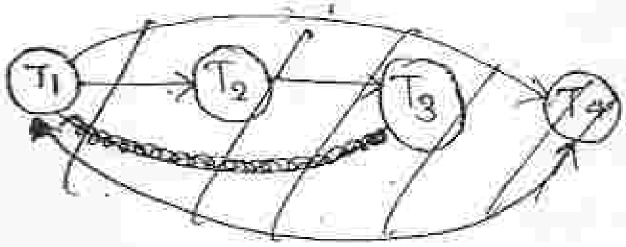
$T_1 \rightarrow T_2$

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

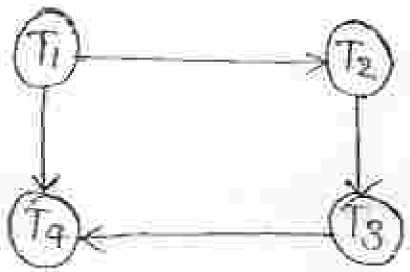
S_1 is not C.S

Ques:-
1) Find the serializability order of the concurrent schedule given below.

$R_1(X)$ $W_3(X)$ $W_2(Y)$ $R_2(Y)$ $W_2(Z)$ $R_4(X)$ $R_4(Y)$

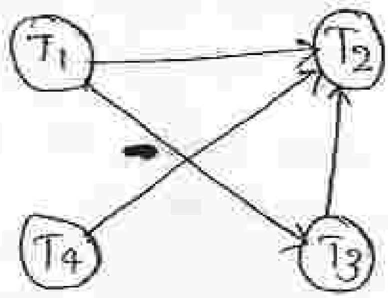


C.S $T_1 - T_2 - T_3 - T_4$



Ques
2) $R_4(A)$ $R_2(A)$ $R_3(A)$ $W_1(B)$ $W_2(A)$ $R_3(B)$ $W_2(B)$

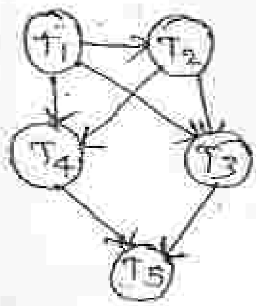
a) not C.S
b) $T_3 T_4 T_1 T_2$
c) $T_1 T_4 T_3 T_2$
d) $T_2 T_3 T_1 T_4$



C.S = T_1 T_4 T_3 T_2
T_4 T_1 T_3 T_2
T_1 T_3 T_4 T_2

Q. Consider the precedance graph given below.

final the no. of possible Serial schedules ?



$$T_1 - T_2 \begin{cases} T_3 - T_4 - T_5 \\ T_4 - T_3 - T_5 \end{cases} \quad C.S = \underline{2}$$

(8) View equivalent schedule

Two schedules S and S' are said to be view equivalent if the following 3 - conditions are met for each data item (say A).

(i) For each data item A if transaction Ti reads the initial Value A in Schedule S then transaction Ti must in schedule S' al read the initial Value of A.

(ii) If transaction Ti executes read - R(A) in schedule S. and that was produced by transaction Tj (if any) then transaction Ti must in schedule S' also read the Value of A that was produced by Tj

(iii) For each data item the transaction (if any) that performs final write of A operation in S must also perform the final write of A operation in S'

Note:- All write - Read Sequence to be maintained in the same order in both S and S'

S_1

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

S_2

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

$S_1 \overset{v}{=} S_2$

$\Rightarrow$ <u>S_1 is view Serializable.</u>

Qu

S_1

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

S_2 $T_1 \rightarrow T_2$

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

$S_1 \overset{v}{\neq} S_2$

S_3 $T_2 \rightarrow T_1$

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

$S_2 \overset{v}{\neq} S_3$

$\Rightarrow$ <u>S_1 is not view Serializable</u>

Que S1:

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)

S2:

T_1	T_2	T_3
R(A)		
W(A)		
	W(A)	
		W(A)

$S_1 \overset{v}{=} S_2$

S1 is View Serializable.

View Serializable

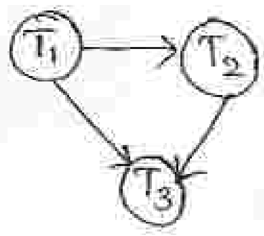
A schedule is view serializable if it is view equivalent to a serial sched

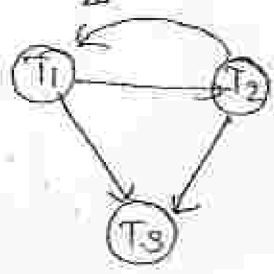
Polygraph

S_1

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)

It is not C.S





V.S: $\underline{T_1 - T_2 - T_3}$

S_1

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

It is ~~not~~ C.S and V.S also

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

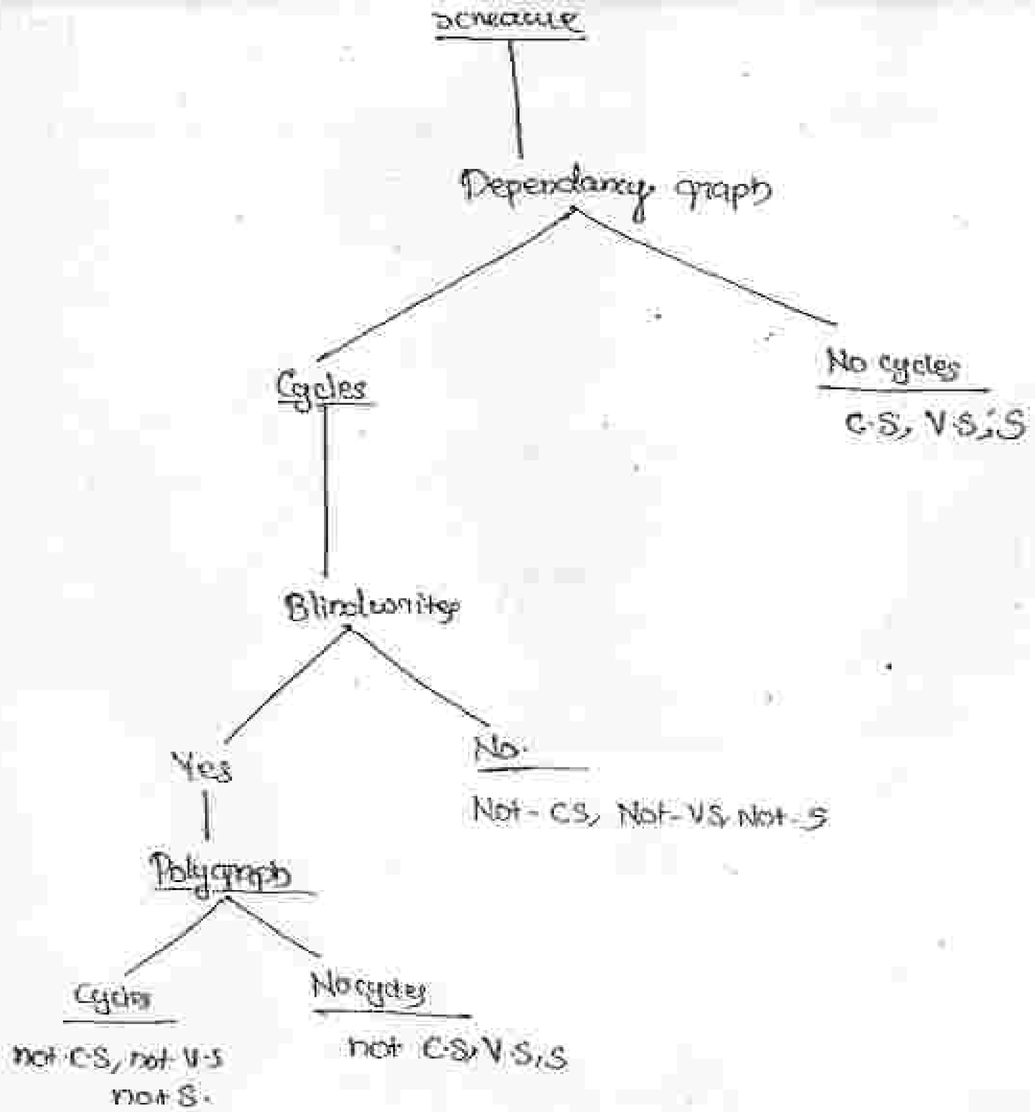
Blind write

It is not C.S and not V.S

Note: A schedule that is conflict serializable is also view serializable. but a view serializable schedule need not be conflict serializable

Blind write:- writes without read is called Blind write.

Note: A schedule is V.S but not C.S. There must be atleast one blind write operation.



Serializable schedules :- A schedule is serializable. If it is either conflict serializable or view serializable or both.

How many concurrent schedules can be formed over two transactions having two-two operations each ?

T_1	T_2
$R_1(A)$	$R_2(A)$
$W_1(A)$	$W_2(A)$

1) $R_1(A)\ W_1(A)\ R_2(A)\ W_2(A)$

2) $R_1(A)\ R_2(A)\ W_1(A)\ W_2(A)$

3) $R_1(A)\ R_2(A)\ W_2(A)\ W_1(A)$

4) $R_2(A)\ W_2(A)\ R_1(A)\ W_1(A)$

5) $R_2(A)\ R_1(A)\ W_1(A)\ W_2(A)$

6) $R_2(A)\ R_1(A)\ W_2(A)\ W_1(A)$ — non serial schedule

Concurrent schedules = 6 ⟶ $\frac{(2+2)!}{2! * 2!} = \frac{4!}{2! * 2!} = \frac{24}{4} = 6$

Non-serial schedules = 6 − 2 = 4

Note:-

The no. of concurrent schedules that can be formed over two transactions having n_1 and n_2 operations respectively are $\frac{(n_1+n_2)!}{n_1! * n_2!}$.

The no. of ~~con~~ non-serial schedules are

$$= \left[\frac{(n_1+n_2)!}{n_1! * n_2!}\right] - 2$$

Note:-

The no. of concurrent schedules that can be formed over m- transactions having $n_1, n_2, \ldots, n_m$ operations respectively are

$$\left[\frac{(n_1+n_2+\cdots+n_m)!}{n_1! * n_2! * n_3! * \cdots * n_m}\right]$$

The no. of non-serial schedules are

$$\left[\frac{(n_1+n_2+n_3+\cdots+n_m)!}{n_1! * n_2! * n_3! \cdots * n_m}\right] - m!$$

Q4 Determine whether the following schedules are conflict serializable or not?

(P.T.O)

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

S_2: $R_1(A)\ R_1(B)\ W_2(A)\ R_3(A)\ W_1(B)\ W_3(A)\ R_2(B)\ W_2(B)$

S_3: $R_1(A)\ R_2(A)\ R_1(B)\ R_2(B)\ R_3(B)\ W_1(A)\ W_2(B)$

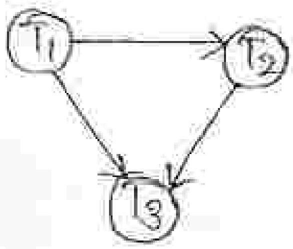
S_4: $W_3(A)\ R_1(A)\ W_1(B)\ R_2(B)\ W_2(C)\ R_3(C)$.

Ans

S_1:

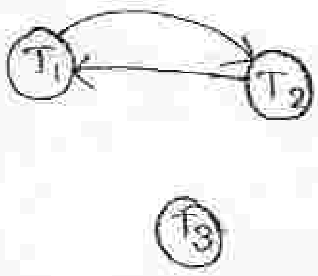
It is C.S $(T_2 - T_1)$

S_2:



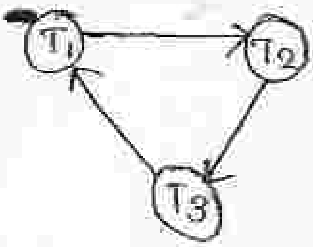
It is C.S $(T_1 - T_2 - T_3)$

S_3:



It is not C.S

S_4:



It is not C.S

Q. Two transactions T_1 and T_2 are given as follows.

T_1: $R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)$

T_2: $R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

Find the total no. of conflict serialisable schedules that can be formed by T_1 and T_2?

(P.T.O)

$T_1 \rightarrow T_2$: $\underline{R_1(A)\, W_1(A)\, R_1(B)\, W_1(B)}$ $\underline{R_2(A)\, W_2(A)\, R_2(B)\, W_2(B)}$ — 6

$\frac{(2+2)!}{2!*2!} = 6$

6-possibilities.

$T_2 \rightarrow T_1$: $\underline{R_2(A)\, W_2(A)\, R_2(B)\, W_2(B)}$ $\underline{R_1(A)\, W_1(A)\, R_1(B)\, W_1(B)}$ — 6

$= \frac{(2+2)!}{2!*2!} = 6$

6-possibilities.

$\underline{12}$

Qn Two transactions T_1 and T_2 are given as follows

T_1: $R_1(A)\, W_1(A)\, R_1(B)\, W_1(B)$

T_2: $R_2(B)\, W_2(B)\, R_2(A)\, W_2(A)$

$T_1 \rightarrow T_2$ $R_1(A)\, W_1(A)\, R_1(B)\, W_1(B)$ | $R_2(B)\, W_2(B)\, R_2(A)\, W_2(A)$

$T_2 \rightarrow T_1$ $R_2(B)\, W_2(B)\, R_2(A)\, W_2(A)$ | $R_1(A)\, W_1(A)\, R_1(B)\, W_1(B)$

Qn Consider the following schedules

S_1: $R_2(X)\, R_2(Y)\, R_1(X)\, R_1(Y)\, W_1(X)\, R_2(X)$

S_2: $R_2(X)\, R_2(Y)\, W_2(X)\, R_1(X)\, R_1(Y)$

S_3: $R_2(X)\, R_2(X)\, R_1(X)\, R_1(Y)\, W_1(X)\, W_2($

a) which of the above ~~started~~ schedules having un-repeatable read problem.

b) which of the above schedules having - lost update problem.

a) S_1: $R_2(X) \ldots\ldots \underline{W_1(X)} \cdot R_2(X)$

b) S_3: $\ldots\ldots \underline{W_1(X)\, W_2(X)}$

c) which of the above ~~un~~ schedules having ~~un-committed~~ dirty read problem?

S2: $\ldots\ldots W_2(X) \cdot R_1(X) \ldots\ldots$

S1: $\ldots\ldots W_1(X)\, R_2(X)$ -

* for the following schedules find whether the schedule is recoverable, cascadeless and strict?

158

1: $R_1(X)\ R_2(X)\ W_1(X)\ W_2(X)$ Commit$_2$ (C_2) C_1(commit 1)

→ Recoverable

→ Cascadeless

→ Not Strict

T_1	T_2
R(X)	
	R(X)
W(X)	
	W(X)
	Commit
Commit	

2: $R_1(X)\ W_2(X)\ W_1(X)\ R_1(X)$ Commit 2 Commit 1

→ Recoverable

→ Cascadeless

→ Not Strict

T_1	T_2
R(X)	
	W(X)
W(X)	
R(X)	
	Commit
Commit	

S3: $R_3(X)\ R_1(X)\ R_2(X)\ W_1(Y)\ R_2(Y)\ W_2(X)\ C_3\ C_1\ C_2$

→ Recoverable

→ cascading

→ Not Strict

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

S4:

T_1	T_2	T_3
$R_1(X)$		
	$R_2(Z)$	
		R(X)
R(Z)		
	R(Y)	
W(X)		
Commit		
	W(Z)	
		W(Y)
	W(X)	
	Commit	
		Commit

→ Recoverable

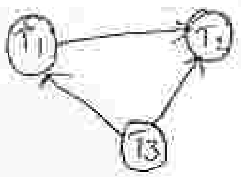
→ cascadeless

→ Strict

P-216

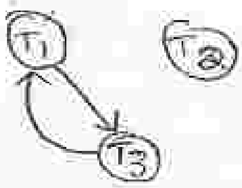
Qu (1)

(1)



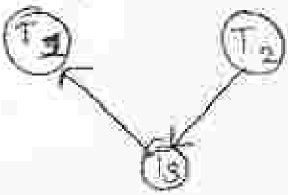
T3 – T1 – T2 (C.S)

(2)



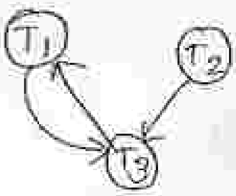
not C.S

(3)



T2 – T3 – T1 (C.S)

(4)



not C.S

Qu (2)

T1
R(X)
R(Y)
W(X)

T2
R(X)
R(Y)
W(X)
W(Y)

a) WR = ?

T1	T2
R(X)	
R(Y)	
W(X)	
	R(X)
	R(Y)
	W(X)
	W(Y)
commit	commit

b) RW = ?

T1	T2
R(X)	
	R(X)
	R(Y)
	W(X)
	W(Y)
R(X)	
R(Y)	
W(X)	

c) WW = ?

T1	T2
R(X)	
R(Y)	
	R(X)
	R(Y)
	W(X)
	W(Y)
W(X)	

3) a)

T_1	T_2
R(X)	
	R(X)
W(X)	
	W(X)

not - serializable

not - conflict serializable

not - view serializable

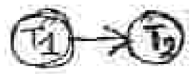
→ Recoverable

→ Cascadeless

→ Not Strict

b)

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



→ C.S

→ V.S

→ S

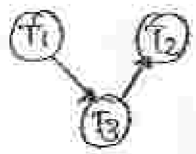
→ ~~Recoverable~~ ~~...~~ recoverable

→ Cascading aborts

→ Not Strict

c)

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



→ C.S

→ V.S

→ S.

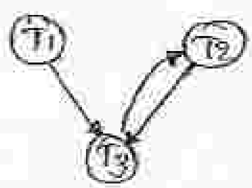
→ Recoverable

→ Cascading aborts

→ Not Strict

d)

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



→ not C.S

→ not V.S

→ not S

→ Recoverable.

→ Cascading aborts.

→ Not Strict

T1 → C.S

e)

T1	T2
R(X)	
	W(X)
W(X)	
	Abort
Commit	

Roll back

- → ~~not~~ C.S
- → ~~not~~ V.S
- → ~~not~~ S
- → Recoverable
- → Cascadeless
- → not Strict

f)

T1	T2
R(X)	
	W(X)
W(X)	
	C
C	

- → not C.S
- → not V.S
- → not S
- → Recoverable
- → Cascadeless
- → Not Strict

g)

T1	T2
W(X)	
	R(X)
W(X)	
	Abort
Commit	

- → C.S [aborted]
- → ~~not~~ V.S
- → S
- → Recoverable
- → Cascading Abort
- → not Strict

h)

T1	T2
W(X)	
	~~W(X)~~ R(X)
W(X)	
	Commit
fail Commit	

- → not C.S
- → not V.S
- → not S
- → not Recoverable
- → Cascadeless
- → not strict

(i)

T1	T2
W(X)	
	R(X)
W(X)	
	Commit
Abort	

- → C.S ✓
- → V.S ✓
- → S ✓
- → Not Recoverable
- → Cascadeless
- → not Strict

(i)

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

- → C.S
- → V.S
- → S
- → Recoverable
- → Cascadeless
- → Strict

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

- → not CS
- → not VS
- → not S
- → Recoverable
- → ~~Cascading aborts~~ Cascadeless
- → Strict

(iii)

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

- → not C.S
- → not V.S
- → not S
- → Recoverable
- → Cascading Aborts
- → not Strict

(2-6-30)

Concurrency Control

1) Lock based protocols

Which requires that all data items must be accessed in a mutually exclusive manner. ie, when one transaction is accessing the data item no - other transaction simultaneously update the data item.

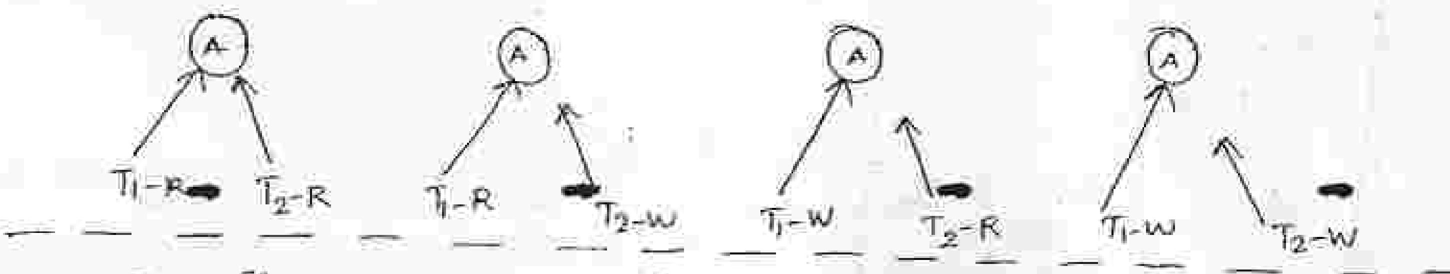
To implement lock based protocols we need two locks.

1) Shared lock :- Only Read (Lock - S)

2) Exclusive lock :- both Read & Write (Lock - X)

	S	X
S	✓	✗
X	✗	✗

} compatibility between lock modes



$T_1 : A \xrightarrow{50} B$	T_2 : Display (B+A)
Lock - X(A)	Lock - S(B)
Read (A)	Read (B)
A = A - 50	Unlock (B)
Write(A)	Lock - S(A)
Unlock (A)	Read (A)
Lock - X (B)	Unlock (A)
Read(B)	Display (B+A)
B = B+50	
Write(B)	
Unlock(B)	

T_1	T_2
Lock_X(A)	
Read (A)	
A = A - 50	
Write(A)	
Unlock (A)	
	Lock_S(B)
	Read (B)
	Unlock (B)
	Lock_S(A)
	Read (A)
	Unlock (A)
	Display(B+A)
Lock_X(B)	
Read(B)	
B = B+50	
Write (B)	
Unlock(B)	

	A	B
before T_1:	1000	2000
After T_1	950	2050

(A+B) = 3000 is the only consistant data item.

Output from S1 : 2950

inconsistant Value may be Because early unlock

modified S1

T_1	T_2
Lock_X(A)	
R(A)	
A = A-50	
W(A)	
	Lock_S(B)
	R(B)
	Lock_S(A)
	R(A)
	Display (B+A)
	Commit
	Unlock (B)
	Unlock (A)
Lock_X(B)	
R(B)	
B = B+50	
W(B)	
Commit	
Unlock(B)	
Unlock (A)	

T_1 waits for T_2 to release lock on B

T_2 waits for T_1 to release lock on A

⇒ Causes Deadlock

Note:-

If we do not unlock a data item before requesting a lock on another data item. deadlocks may occur.

Q4 Consider the following schedule and determine the serializability order

(1)

T_1	T_2
L(A)	
U(A)	
	L(B)
	U(B)
L(B)	
U(B)	

$T_2 \rightarrow T_1$

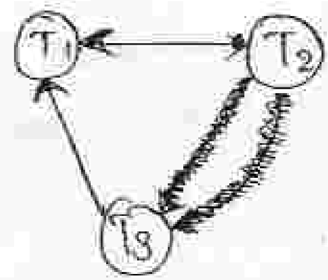
(2)

T_1	T_2
L(A)	
U(A)	
	L(A)
	U(A)
	L(B)
	U(B)
L(B)	
U(B)	

not C.S

(3)

T_1	T_2	T_3
	L-S(A)	
		L-S(A) ---- (Lock point)
(L.P) —	L-X(B)	
	U(A)	
		L-X(C)
	U(B)	
L-S(B)		
		U(A)
		U(C)
(L.P) — L-X(A)		
U(A)		
U(B)		



It is conflict serializable

$$\begin{bmatrix} T_2 - T_3 - T_1 \\ T_3 - T_2 - T_1 \end{bmatrix}$$

2-Phase locking protocol

which requires both locks and un-locks being done in 2-phases.

1) Growing Phase - "Obtain locks"
2) Shrinking phase - "Release locks"

Lock point :-

The point at which the transaction has obtained its final lock is called, a lock point

→ The lock point is used to determine the serializability order of a concurrent schedule

Two phase locking protocol ensures conflict serializability and serializability order is determined based on their lock points.

T_1	T_2	T_3
Lock_X(A) — Lp		
R(A)		
W(A)		
Lock_S(B)		
R(B)		
Unlock(A)		
	Lock_X(A) — Lp	
	R(A)	
	W(A)	
	Unlock(A)	
		Lock_S(A) — Lp
		R(A)

Roll back

Roll back

Roll back

fail

$T_1 - T_2 - T_3$

Cascading rollbacks may occur under 2-phase locking protocols.

- Deadlocks may occur under two phase ~~schedules~~ locking protocols.

Cascading rollbacks can be avoided by a modification of 2-phase locking protocols

(1) <u>Strict 2-phase locking protocol</u> (Strict 2PL) :-

Which requires that in addition to locking being 2-phase all exclusive mode locks taken by a transaction must be held until the transaction commits.

(2) <u>Rigorous 2-phase locking protocol</u> (Rigorous 2PL) :-

Which requires that in addition to locking being 2-phase all locks must be held until the transaction commits.

> Avoids cascading Rollbacks but deadlocks can occur

(3) <u>Conservative 2PL</u> :-

Which requires the transaction to update all the locks before it starts and release all the locks after it commits.

> Avoids cascading Rollbacks and deadlocks

Q: which of the following schedules (Transactions) are in strict 2PL

a)	b)	c)
Lock_S(A)	Lock_S(A)	Lock_S(A)
R(A)	R(A)	R(A)
Lock_X(B)	Lock_X(B)	Lock_X(B)
R(B)	Unlock(A)	R(B)
Unlock(A)	R(B)	W(B)
W(B)	W(B)	Commit
Unlock(B)	Commit	Unlock(A)
	Unlock(B)	Unlock(B)

d) Lock-S(A)
Lock-X(B)
R(B)
W(B)
R(A)
Commit
Unlock(A)
Unlock(B)

e) Lock-S(A)
R(A)
Unlock(A)
Lock-X(B)
R(B)
W(B)
Unlock(B)
Unlock(A)
Commit

→ Simple transaction with

e - not 2PL
a, b, c, d → basic - 2PL
b, c, d → Strict 2PL
c, d → Rigorous 2PL
d → Conservative

T_1	T_2
Lock-S(asset)	
Read(asset)	
Lock-S(di)	
Read(di)	
asset-asset+di	
	Lock-S(asset)
	Read(asset)
Lock-S(wi)	
Read(wi)	
asset=asset-wi	
Upgrade(asset)	
Write(asset)	
Commit	
Unlock(asset)	
⋮	

2- Lock Conversions

Upgrade → Shared to Exclusive

Downgrade → Exclusive to Shared

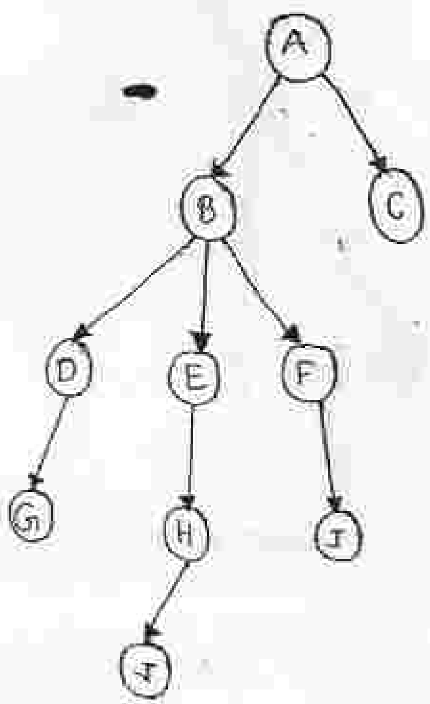
2 Graph based protocol

To implement graph based protocol we need additional information on how each transaction will access the database. The simplest model requires that we have prior knowledge about the order in which the database items will be accessed. To ~~aquire~~ acquire such prior knowledge we impose partial ordering on set of all data items. ie, if $d_i \longrightarrow d_j$ is an ordering any transaction which requires d_j must require to access d_i before d_j.

The partial ordering is represented as a directed acyclic graph called a database graph.

The simplest protocol of graph based protocol is called "Tree protocol" which requires to employ only exclusive mode locks.

Ex:-



In tree protocol the only lock instruction allowed is <u>lock-X</u> and each transaction T_i can lock a data item atmost once and must observe the following rules.

(1) The first lock by T_i may be on any data item.

(2) Subsequently a data item can be locked by T_i only if the parent of the data item is currently locked by T_i

(3) Data items may be unlocked at any time.

(4) A Data item that has been locked and unlocked by T_i cannot subsequently relocked by T_i

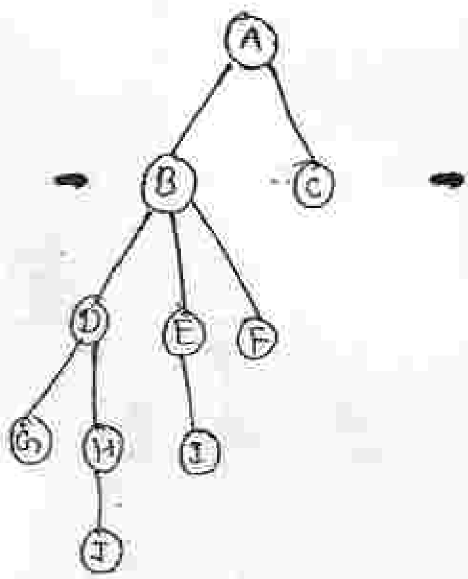
<u>Note :-</u>

All schedules that are legal under tree protocol are conflict serializable.

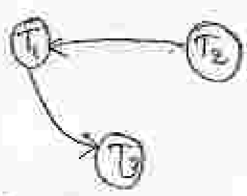
<u>Ex:-</u>

T_1	T_2	T_3
Lock-X(B)		
	Lock-X(D)	
	Lock-X(H)	
	Unlock(D)	
Lock-X(E)		
Lock-X(D)		
Unlock(B)		
Unlock(E)		
		Lock-X(B)
		lock-X(E)
	Unlock(H)	
Lock-X(G)		
Unlock(D)		
		Unlock(E)
		Unlock(B)
Unlock(G)		

<u>Tree of protocol</u>



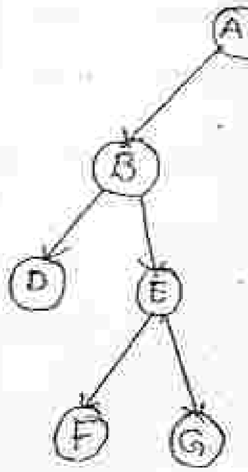
<u>Conflict Serializable</u>

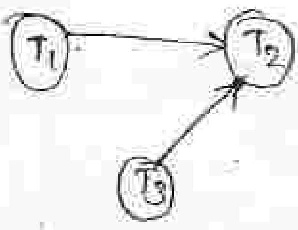


$T_2 - T_1 \; T_3$

Q. Test ~~[illegible]~~ whether the following schedule can occur under tree protocol? If possible ~~[illegible]~~ give the serializability order.

T_1	T_2	T_3
L(A)		
L(B)		
L(D)		
U(B)		
	L(B)	
L(C)		
		L(E)
U(D)		
U(A)		L(G)
		U(G)
		U(E)
	L(E)	
	U(B)	
	U(E)	
U(C)		





$\left\{ \begin{matrix} T_1 & T_3 & T_2 \\ T_3 & T_1 & T_2 \end{matrix} \right\}$ C.S

Advantages:-

- Early unlocks causes increase in concurrency
- Ensures conflict serializability
- It is a deadlock free protocol

Drawbacks:-

- Requires prior knowledge about the order of the data items
- Unnecessary locking overheads ie, in some cases it is required lock the data items that it does not access.

Timestamp based protocol

Which requires that ordering among the transactions is determined in advance based on their timestamps. (ie, the time when ever it enters into the system for execution)

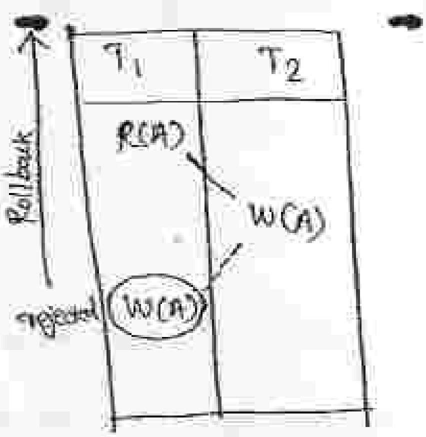
Each transaction is given unique fixed timestamp denoted by $TS(T_i)$.

If any transaction T_j that enters after T_i, the relation between their timestamps be $TS(T_i) < TS(T_j)$ which means that the producing schedule must be equivalent to a serial schedule $T_i \rightarrow T_j$

(1) Timestamp Ordering protocol

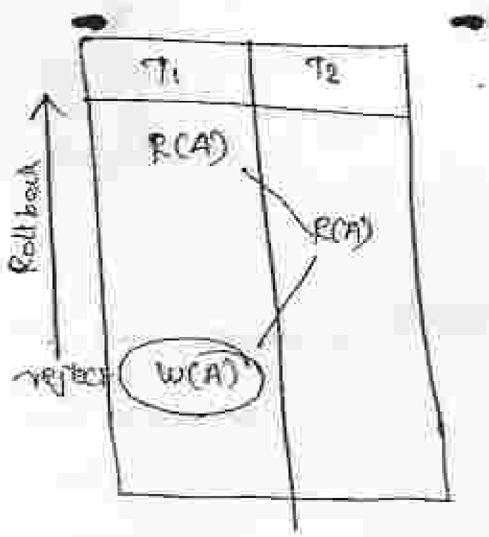
In timestamp ordering protocol ensures that any conflicting read and write operations are executed in timestamp order if not such an operation is rejected and the transaction will be rolled back. The rolled back transaction will be restarted with a new timestamp.

Ex:-



$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$



$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

Q. Check is the following schedule can appear under timestamp ordering protocol:

T_1	T_2
R(B)	
	R(B)
	B = B-50
	W(B)
R(A)	
	R(A)
Display(A+B)	
	A = A+50
	W(A)
	Display (A+B)

It can appear under TOP

$$\frac{TS(T_1) < TS(T_2)}{T_1 \rightarrow T_2}$$

Q.

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

Rollback

reject

Obsolete write

Can't appear under TOP

$$\frac{TS(T_1) < TS(T_2)}{T_1 \rightarrow T_2}$$

3) Thomas write rule

which requires to ignore all obsolete write operations

Ex:-

T_1	T_2
	R(A)
W(A)	
	W(A) — ignore

$TS(T_2) < TS(T_1)$

$T_2 \to T_1$

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

T_1	T_2
	W(A)
W(A)	
	W(A) — reject

~~Timestamp order~~

(Reject and Rollback)

	Not allowed		Allowed	
TOP	$R_1(A)$	$W_2(A)$	$R_1(A)$	$R_2(A)$
	$W_1(A)$	$R_2(A)$		
	$W_1(A)$	$W_2(A)$		
TWR	$R_1(A)$	$W_2(A)$	$R_1(A)$	$R_2(A)$
	$W_1(A)$	$R_2(A)$	$W_1(A)$	$W_2(A)$

- if $T_2 \to T_1$ then $R_2(A)$ must be present
- If $T_1 \to T_2$ if time stamp order and there is $R_1(A)$ is present

Que

8: $W_1(X)$ $W_2(X)$ $W_3(X)$ $R_2(X)$ $W_4(X)$

Consider the schedule and the statement 1 is

Statement 1: The above schedule is possible under timestamp ordering protocol

Statement 2: The above Schedule is possible under thomas right rule.

Which of the above ~~schedules are~~ Statements are true about the Schedule given above?

Sol:

T_1	T_2	T_3	T_4
W(X)			
	W(X)		
		W(X)	
	R(X)		
			W(X)

according to TOP - reject & Roll back

1 - 2 - 3 - 4

The above schedule cannot appear under both timestamp ordering protocol and thomas write rule protocol.

Q:

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

$T_1 \rightarrow T_2$

reject $W_1(A)$ and roll back T_1 according to TOP

ignore $W_1(A)$ and continue with T_1 according to TWR

~~Advantage~~

~~This protocol ensures~~

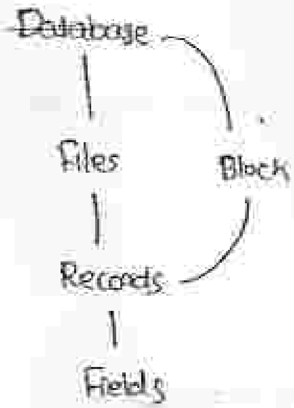
Adv

- This protocol ensures serializability (according to T.S)
- ensures freedom from deadlock

Disadv

- Starvation may occur due to continuously getting aborted and restarting the transaction.

File organization



Data base is a collection of files,
each file is a collection of records,
Each record is a sequence of fields

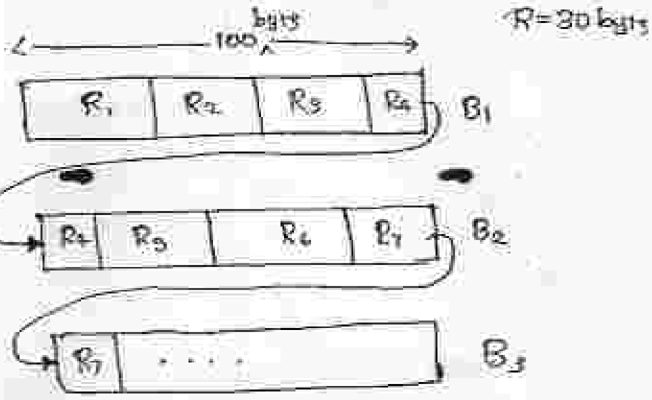
Blocking factor

Blocking factor is the average no. of records per block.

Strategies for storing file of records into block

① Spanned strategy

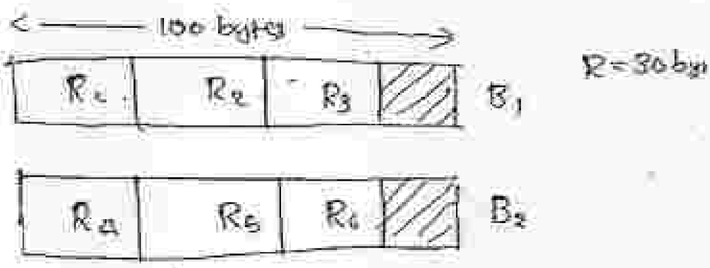
It allows, partial part of record can be stored in a block



- Adv:- No wastage of memory
- Disadv:- Block accesses increases
- Suitable:- It is suitable for variable length record.

② Un-spanned strategy

No record can be stored in more than 1-block.



- Disadvantage:- Wastage of memory
- Advantage:- Block accesses reduced.
- Suitable:- It is suitable for fixed length record.

Organization of records in a file

1) Ordered file organization

All file of records are ordered based on some search key value

Searching :- Binary Search

B - data blocks

to access a record, the average no. of block access

$= \log_2 B$ blocks

Adv :- Searching is efficient

Disadvantage :- Insertion is expensive due to re-organization of the entire file.

2) Un-ordered file organization

All file of records are inserted at where ever the place is available, usually at the end of the file.

Searching :- Linear Search

B - data block

to access a record, the average no. of block access $= \frac{B}{2}$ Blocks

Adv :- Insertion is efficient

Dis adv :- Searching is inefficient compared to ordered file organization.

Index :-

Indexes are used to improve the searching efficiency.

Index is a record consists of two fields.

Key	Block pointer

Block pointer ← pointer to a block where 'key' is available.

- Index is an ordered file
- Searching: Binary Search
- To access a record using index, the avg no. of block accesses

$$= \underbrace{\log_2 B_i}_{\text{index block access}} + \underbrace{1}_{\text{Data block access}}$$

B_i - index block

☐ Index can be created on any field of a relation, (primary key, non-key, candidate

Index

- Single level index
  - Primary index (PK + Ordered)
  - ~~Non-~~ Clustered index (Non-key + Ordered)
  - Secondary index ($^{N \cdot K}/_{CK}$ + Un-ordered)
- multi level index
  - B-tree
  - B^+-tree

Classification of Indexes

1) Dense index

2) Sparse index

Dense index

If an index entry is created for every search key value that index is called dense index.

Sparse index

If an index entry is created only for some search key values it is called sparse index

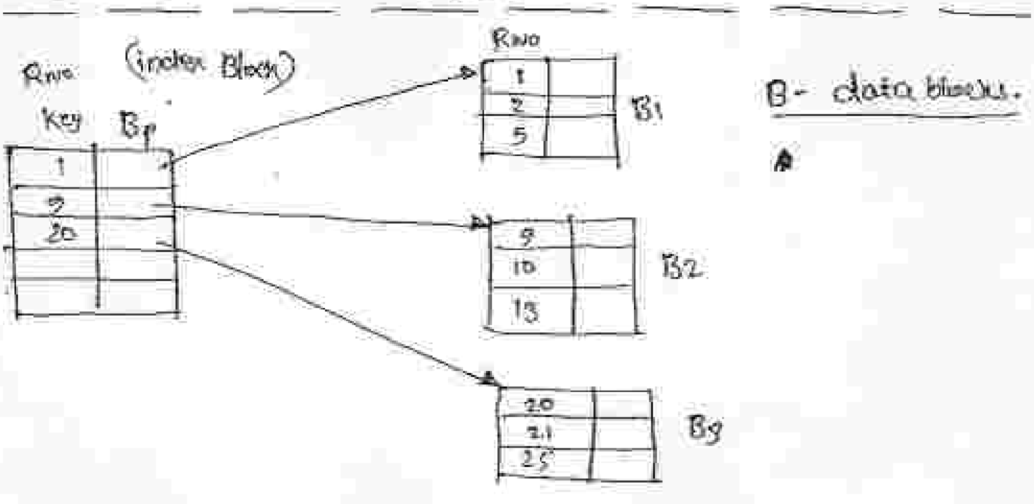
Primary Index (P.K + ordered)

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 and the second field is a pointer to ~~data~~ data block. where the key is available.

- Index entry is created for first record of each block * called "block anchor"
- The no. of index entries is equal's to the no. of data blocks.
- The average no. of block access using index = $\log_2 B_i + 1$ where B_i - ~~data~~ index blocks.
- The type of index is called sparse ~~block~~ index because it is indexing only first record of each block

Searching: Binary Search

B_i - index blocks



Q(2) Suppose that we have an ordered file of 30,000 records stored on a disk with block size 1024 Bytes. File records are of fixed length and are un-spanned of size 100 bytes and suppose that we have created a primary index on the key field of the file of size 9 bytes and a block pointer of size 6-bytes. then find the avg no. of blocks to search for a record using with and without index?

Ans

$r = 30,000$ ordered

$B = 1024$ Bytes

$R = 100$ bytes, fixed length, Un-spanned

Blocking factor $= \left\lfloor \frac{1024}{100} \right\rfloor = 10$ records/block.

no. of data block $= \left\lceil \frac{30000}{10} \right\rceil = 3000$ blocks

Avg. no. of block access $\left\lceil \log_2 3000 \right\rceil = 12$ block access [without Indexing]

↳ size of index blocks $= 9 + 6 = 15$ bytes

↳ no. of index records/Blocks $= \left\lfloor \frac{1024}{15} \right\rfloor = 68$

↳ no of index records = no. of data blocks = 3000

no. of index blocks $= \left\lceil \frac{3000}{68} \right\rceil = 45$

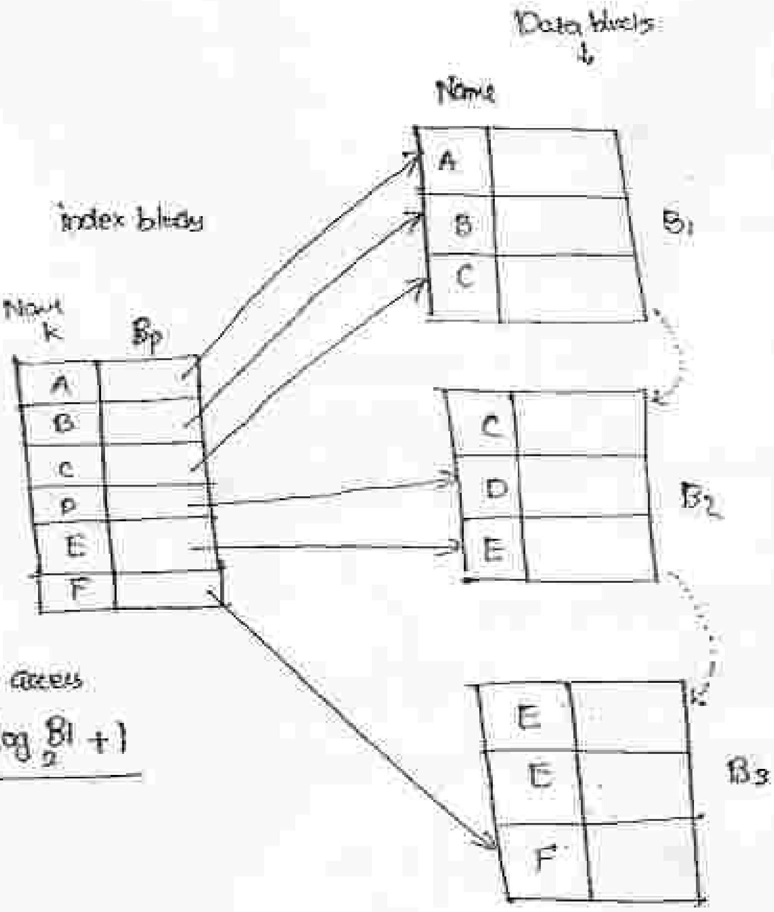
→ Avg. no. of block access $= \left\lceil \log_2 45 \right\rceil + 1 = 6 + 1 = 7$ block access [with Indexi

Clustered Index (NK + Ordered)

clustered index is an ordered file with two fields, the first field is same as the clustering field is called non-key and the second field is a block pointer.

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.

- Index entry is created for ~~distinct~~ each distinct value of a clustering field.
- The block pointer points to first block in which the key is available
- Type of index is Dense (Index entry is created for each distinct key value)/ Sparse (index entry is not created for every record).



B - data blocks

the avg. no. of block access >= $\log_2 B$

Searching: Binary Search

searching: Binary Search

Bi - index blocks

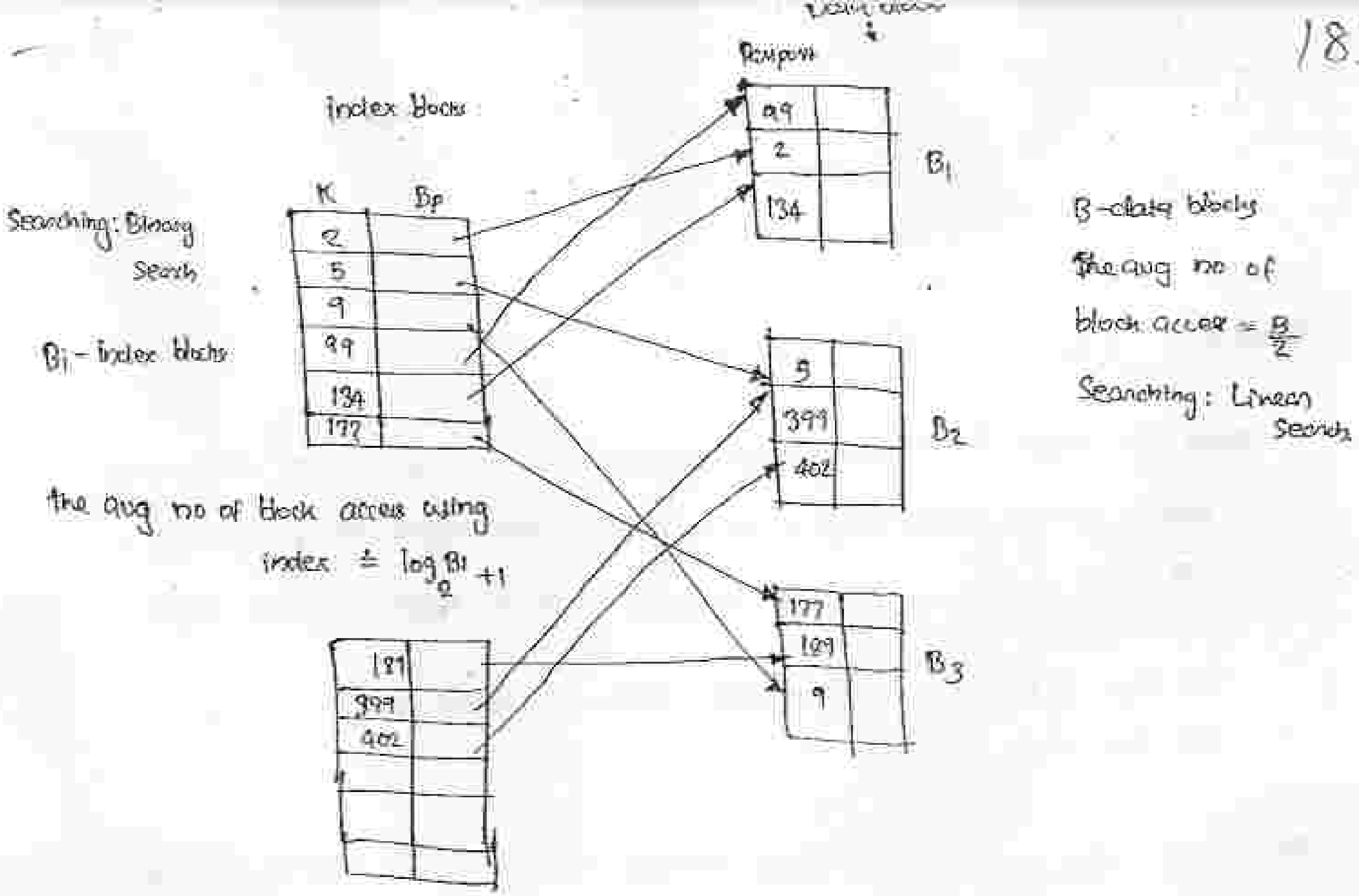
the avg. no. of block access using index >= $\log_2 Bi + 1$

Secondary index (N.K / C.K $\boxed{\checkmark}$ Un-ordered)

Secondary index provides a secondary means of accessing a file for which some primary access already exists.

Secondary index may be on a non-key or a candidate key.

- Index entry is created for each record in a data file.
- No. of index entries = No. of records.
- Type of secondary index is Dense.



Q: consider a secondary index is built on the key field of the file of Question 1. then find avg no of block accesses to access a record using with and without index.

Ans

$r = 30{,}000$, un-ordered

$B = 1024$ Bytes

$R = 100$ bytes, fixed length, Unspanned

$K = 9$ bytes, $B_p = 6$ Bytes

blocking factor $\left\lfloor \frac{1024}{100} \right\rfloor = 10$ records/block

no. of data blocks $= \left\lceil \frac{30000}{10} \right\rceil = 3000$ blocks

the avg. no. of block access $= \frac{3000}{2} = 1500$ block access

→ size of index record = 15 bytes

→ no. of index records/block $= \left\lfloor \frac{1024}{15} \right\rfloor = 68$

→ no. of index records = 30000

→ no. of index blocks $= \left\lceil \frac{30000}{68} \right\rceil = 442$ index blocks.

- Avg no. of Block access $= \log_2 442 + 1$

$= 9 + 1$

$= 10$ block access.

Multilevel index

~~Multi~~ As single level index is an ordered file we can create a primary index to the index itself. In this case the original file is called 1st level index and, the index to index is called 2nd level index

We can repeat the above process until all index entries fit in one disk block.

Qu. Find the avg. no. of block acesses required to search for a record if multi level index is created on the data file of Qu. 2

Ans ~~3000~~ ~~442 index blocks~~

~~9 = records~~

Blocking factor = 10 records / block

no. of data blocks = 3000

1st level

no. of index blocks = 442

2nd level

no. of index records = 442 (no. of 1st level blocks)

$$\text{no. of blocks} = \left\lceil \frac{442}{68} \right\rceil = 7$$

3rd level

no. of index records = 7 (no. of 2nd level blocks)

$$\text{no. of blocks} = \left\lceil \frac{7}{68} \right\rceil = 1$$

Avg. no. of block access. $= 1 + 1 + 1 + 1$

Note:- If there are n- levels in multilevel index the no. of block accesses to search for a record = n+1 (at each level one block and one data block)

Q: Consider a file of 16,384 records each record is of size 32 bytes and key field is of size 6 bytes and the file organization is Un-spanned and records are of fixed length stored on a disk with block size 1024 bytes. and the size of the block pointer is 10 bytes. If the secondary index is built on the key field of the file and multilevel index scheme is used the no. of 1st level and 2nd level blocks in multilevel index respectively are ?

a) 8, 0
b) 128, 6
c) 512, 2
d) 256, 4

Ans

$r = 16384$

$R = 32$ Bytes $= 2^5$

$k = 6$ $B_p = 10$ Bytes

$B = 1024$ bytes $= 2^{10}$

$\Rightarrow 2^5$ records/Block

index record size $= 6+10 = 16 = 2^4$

$\Rightarrow$ no. of index records per block $= 2^6 = \underline{64}$ records/Block

1st level
no. of records = 16384
no. of blocks $= \frac{16384}{64} = 256$

2nd level
no. of records = 256
no. of blocks = 4

2+1+1

Q: If one block access requires 30 ms. 100 blocks index requires ?

$\log_2 100$

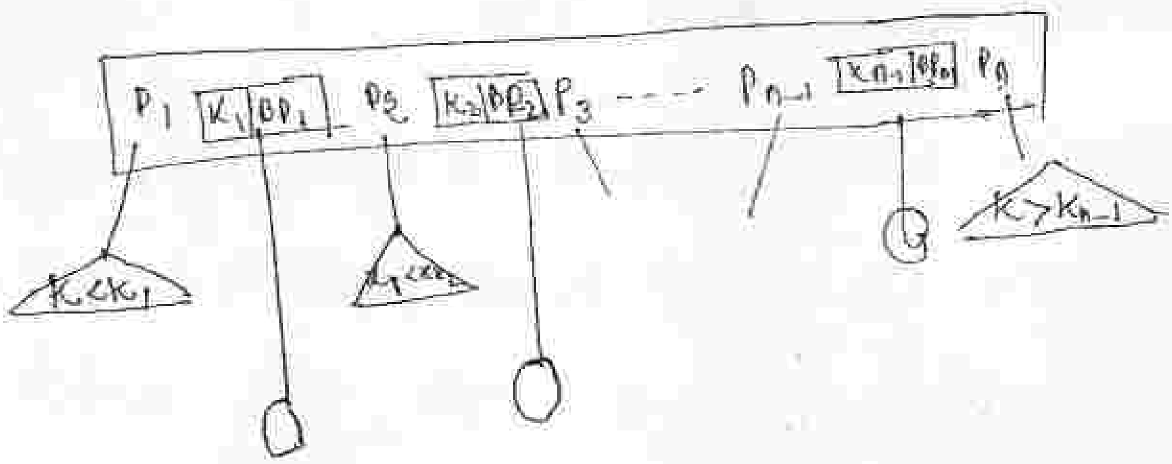
a) 3000
b) 1000
c) 18
d) 210

* B-Tree

→ B-tree is a balanced search tree

→ Node structure of B-tree corresponds to a block

→ Structure of a B-tree node



$K_1 < K_2 < K_3$ ------ $< K_{n-1}$ ← keys

BP_1, BP_2 ------ BP_{n-1} ← Record points / data pointer

P_1, P_2 -------- P_n ← tree pointer / Block pointer

Order of a B-tree : the no of tree pointers
(let n) in node.

n ← tree pointer

(n-1) ← keys & Record pointer

→ $\boxed{n * P}$ ← total size tree points

→ $\boxed{n * P + (n-1)(K + P_r) \leq B}$

P is size of tree pointer

$K + P_r$ is size of index record

B - Block size.

* suppose search field is 9-bytes long; disk block size 512 bytes, & Record pointer is 7-bytes, Block pointer is 6-bytes, then calculate order of b-tree node

sol. $K = 9$, $B = 512$, $P_r = 7$, $P = 6$

$$n * 6 + (n-1)(9+7) \leq 512$$

$$6n + 16n - 16 \leq 512$$

$$22n \leq 528$$

$$\underline{n \leq 24}$$

188

11 suppose that we construct B-tree on the 1, calculate apparent no of n-trees of L3 b-tree. assume that each node 69% full.

order = 24 × 0.69 = 16

Root	1 node	16 points	15 keys
L1	16 node	16×16 points = 256	16×15 = 240
L2	256 node	16×256 = 4096	256×15 = 3840
L3	4096 nodes	4096×16 = 65,536	4096×15 = 61,440

Total no index ents = 15 + 240 + 3840 + 61,440

= 65,535

$\boxed{\text{Total no index ents} = 65,535}$

consider A table 'T', in relation DB, with key field 'K'. A B-tree of order 'p' is used as an access struc, on 'K'. where p denote max no of tree points in B-tree index node. Assume that 'K' is 10 bytes disk block size 512 bytes, each data pointer is 8 bytes & each block pointer is 5 bytes. In order for each B-tree node, to fit in 1 disk block. the max value of 'p' is ______

Sol: $k = 10$; $B = 512$, $P_r = 8$

$n = n*5 + (n-1)(10+8) \le \underline{512}$

$5n + 18n - 18 \le 512$

$23n \le 536$

$n \le 23$

* insertion into a B-tree node:

→ if order is $\underline{n}$

max: n tree pointers — (n-1) keys

min: $\lceil n/2 \rceil$ tree pointer, $\lceil n/2 \rceil - 1$, keys } exception for Root & leaf node

min of 2 children

Eg :

order: 5

max: 5 - tree ptr

4 - keys

Min: 3 - tree ptr

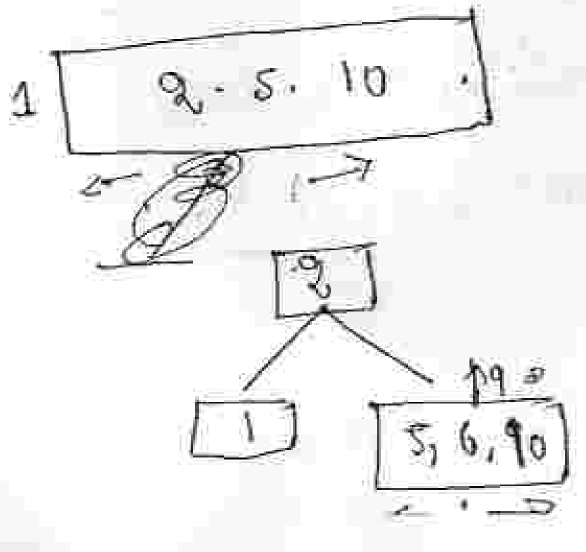
2 - keys

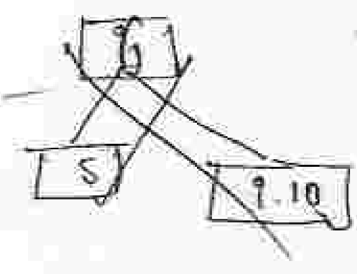
→ new element always inserted at leaf node.
when node is full then split the node like 2-nodes moving the middle element to one higher level. and then insert the element at the proper place

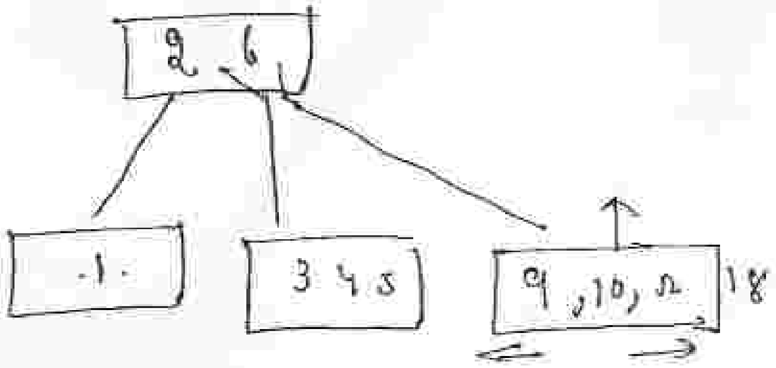
Q// insert the following keys in B-tree of order-4

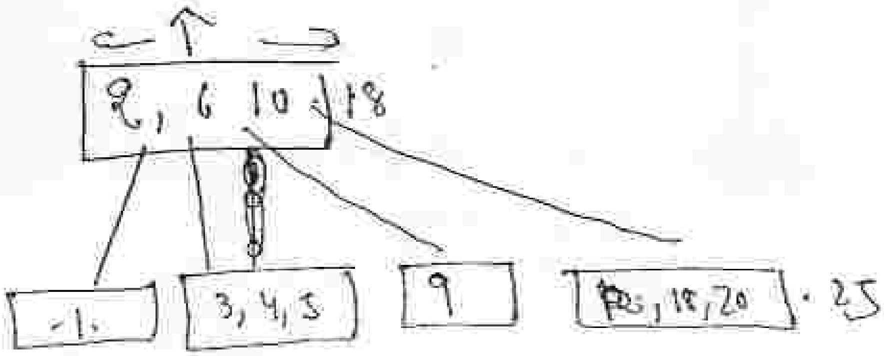
keys: 2, 5, 10, 1, 6, 9, 4, 3, 12, 18, 20, 25

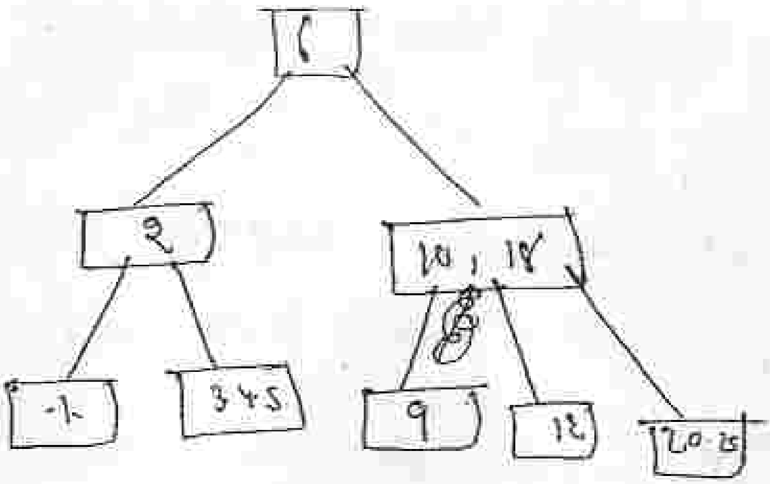
max: 4P, 5K
Min: 2P, 1K









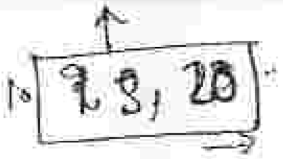


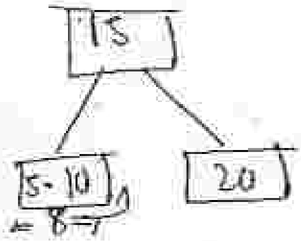
Q. Constrct B-tree of order 3

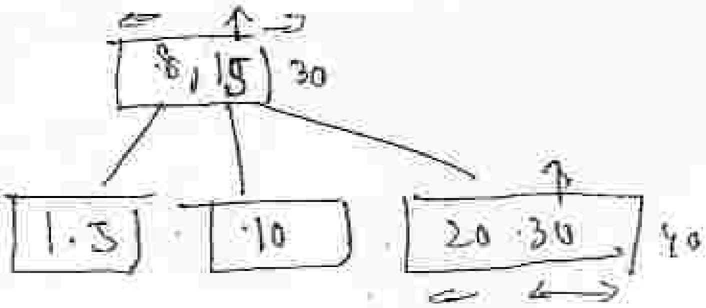
for key 20, 15, 10, 5, 8, 30, 1, 40

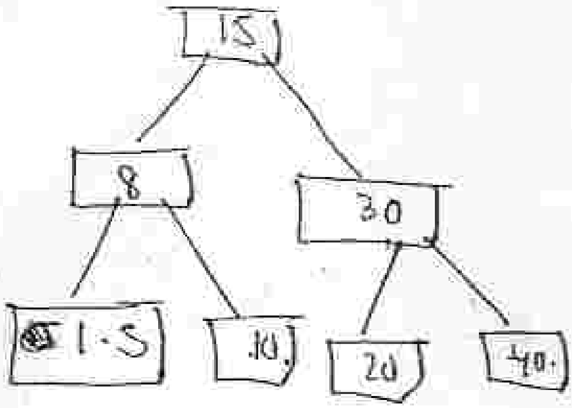
max: 3p, 2 Keys

min: 2p, 1 key

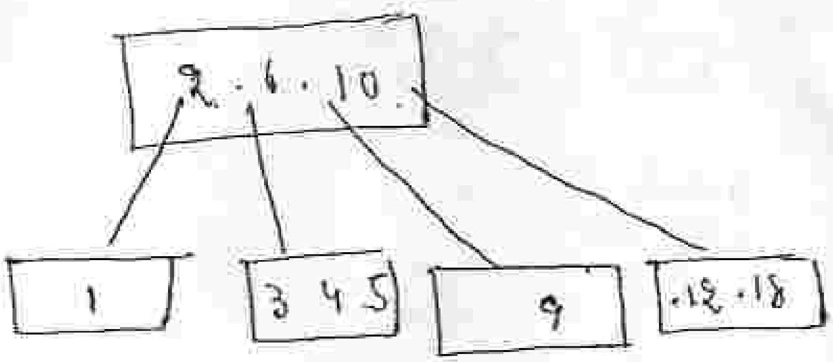








* Deletion

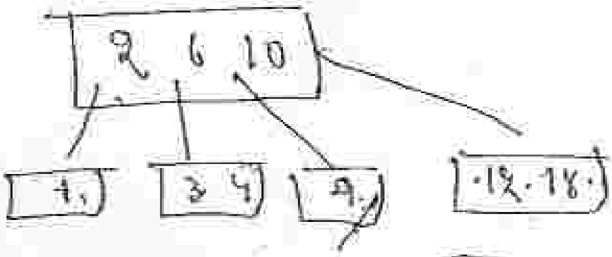


order : 4

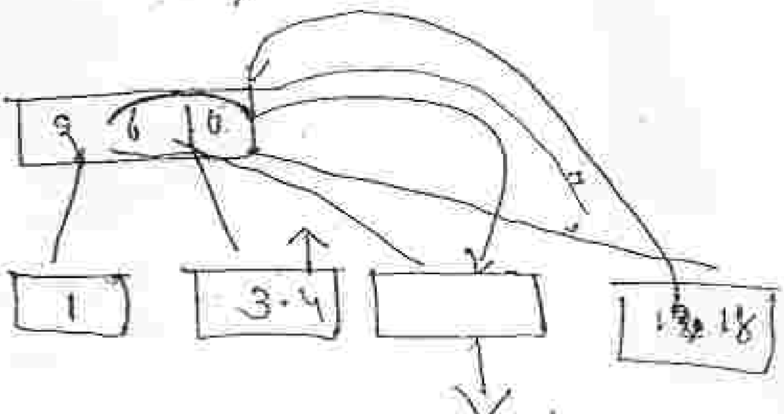
max : 4 pointer 3 key

min : 2 " 1 "

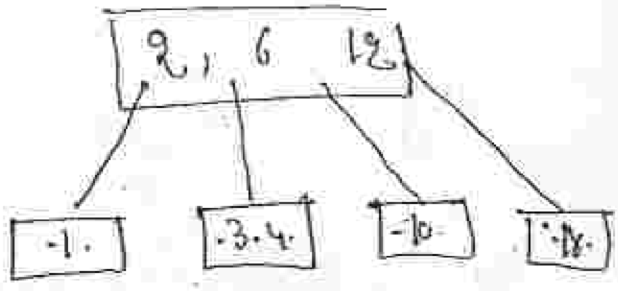
→ delete 5:



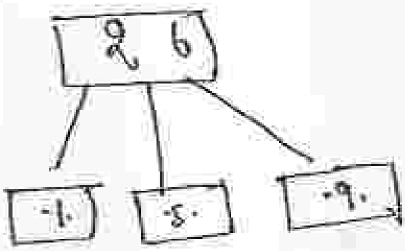
→ delete 9;



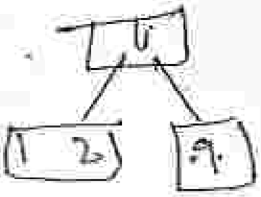
The casing
pruning min!!



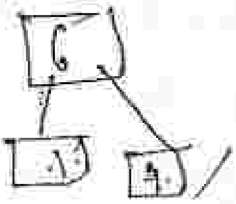
8



Dele 5.

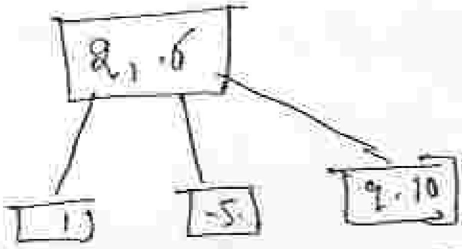


Dele -



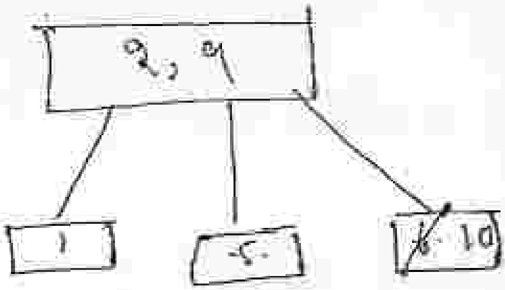
• Delete 9

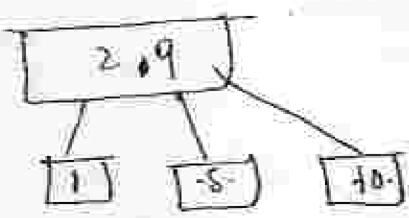
eg :-



D6

D'2





B- tree provide direct access, i.e if key to search is found at some level, it directly access the data block where the key is available, by passing all the lower level of index

* B^+-trees

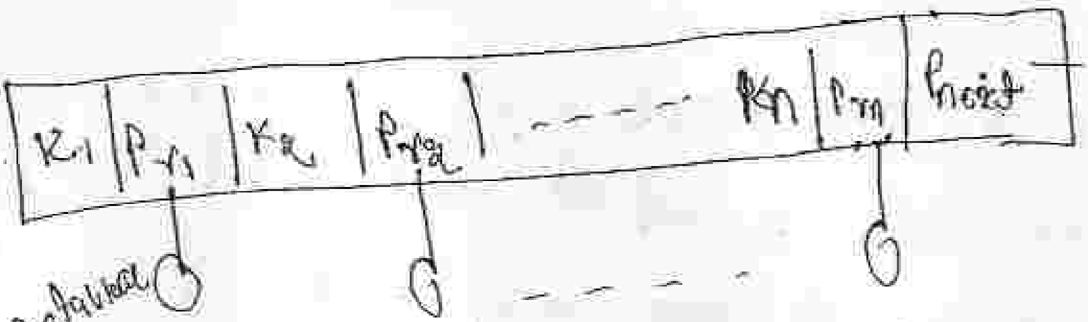
→ from B-tree leaf node, Remove all tree pointers.

→ from B-tree internal node, Remove all Record pointers

→ B+-tree provide an ordered access (i.e. all keys are available at the leaf level).

i.e. searching siblings single level in tree is possible.

* Structure of leaf node



$K_1 < K_2 < \ldots < K_n$ ← keys

$Pr_1, Pr_2, \ldots, Pr_n$ ← Record pointers

P_{next} ⇔ pointer to its sibling / tree pointer order of leaf node:

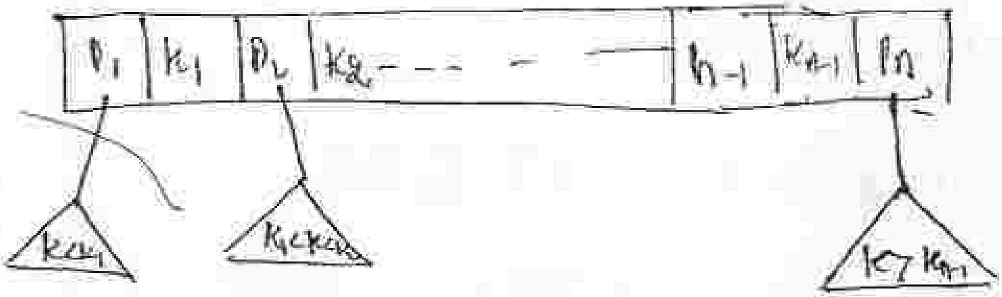
→ no of (key, Record pointer) pairs

$n \leftarrow$ (key, Record pointer)

$1 \leftarrow$ tree pointer

$$\boxed{n(k+P_r) + P_{next} \leq B}$$

* structure of internal node



$K_1 < K_2 < \ldots\ldots < K_{n-1}$ keys

$P_1\ P_2 \ldots\ldots P_n$ tree points

* order of internal node (let n):

no. of tree pointers

$$\boxed{n * P + (n-1)K \leq B}$$

* B⁺-tree allows duplicates, B-tree not contains duplicates

- calulate, order of the B^+ tree node suppose If the search key field is 9 Bytes long, Block size is 512 bytes, record pointer is 7 Bytes, & block pointer = 6 Bytes.

$k = 9$, $B = 512$, $P_r = 7$, $P = 6$

order of internal node

$$n * 6 + (n-1) 9 \leq 512$$

$$6n + 9n - 9 \leq 512$$

$$15n \leq 521$$

$$n \leq 34$$

- order of leaf node

$$n(9+7) + 6 \leq 512$$

$$16n \leq 506$$

$$n \leq \underline{31}$$

Q2. calculate the approximate no. of records in a B^+ tree of level - 3 of que 1 suppose that if each B^+ tree node is 69 % full.

order of internal node = 34 * 0.69 = 23

leaf node = 31 * 0.69 = 21

Root :	1 node	23 pointers
level 1	23 node	23 * 23 = 529
level 2	529 nodes	23 * 529 = 12167 pointers
leaf level	12,167 nodes	21 * 12167 = 2,55,507 keys

Q. A B^+-tree index is to built, on the name attributes of the Relation student assume that all student name are of length 8 Bytes. Disk block are of size 512 Bytes & index pointers are of size 4 Bytes, given this scen what must be the best choice ... the degree ... the ...

$K = 8$, $B = 512$, $P = 4$

$n * 4 + (n-1)8 \leq 512$

$4n + 8n - 8 \leq 512$

$12n \leq 520$

$n \leq \underline{43}$

Q.// The order of a leaf node in a B^+-tree, max no of key, data record pointer pairs it can hold. given that block size 1kb, Data record pointer 7 bytes. the value field is 9 bytes. & block pointer is 6 bytes.

'what is order of leafe node'

$B = 1K$, $Pr = 7$, $K = 9$, $l = 6$

$n(9+7) + 6 \leq 1024$

$16n \leq 1018$

$n \leq \underline{63}$

B+-tree insertion.

Note! we assume that the order of leaf node & internal node is same. But practically it differs

order : n

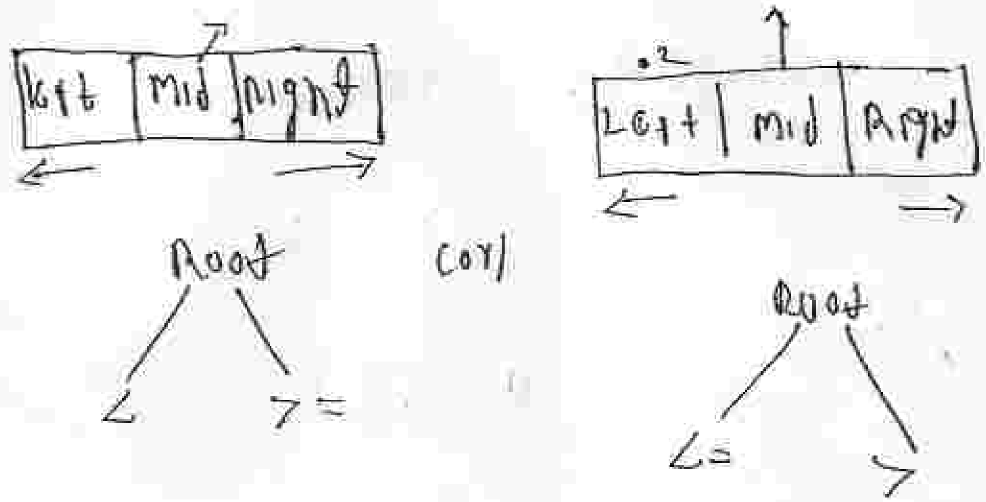
max: n pointers

n-1 keys

min: $\lceil n/2 \rceil$ pointers

$\lceil n/2 \rceil - 1$ keys

Split of leaf node



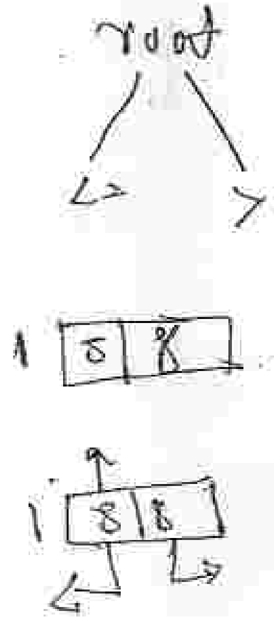
Note: Splitting a leaf node require to maintain duplicate i.e copy of the middle key but internal node split does not require duplicate to maintain

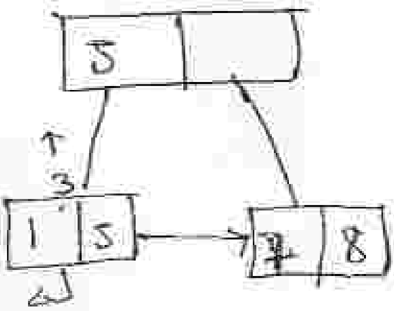
=: Insertion of seq. the keys 8, 5, 1, 7, 3, 12, 9, 6 in a B^+ tree of order 3.

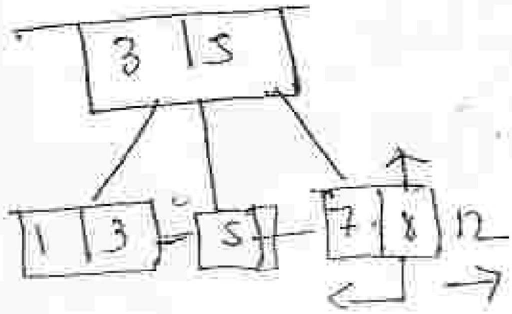
Soln

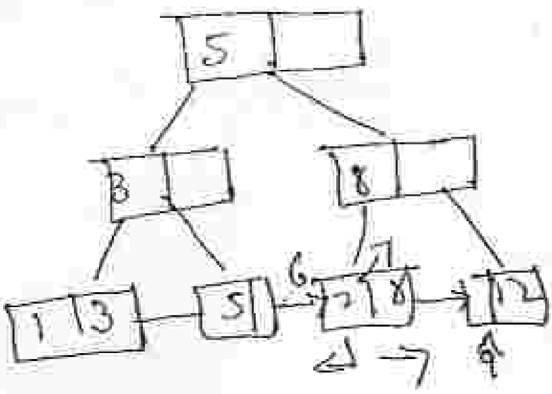
max: 3 p   min: 2 p
2 k   1 k

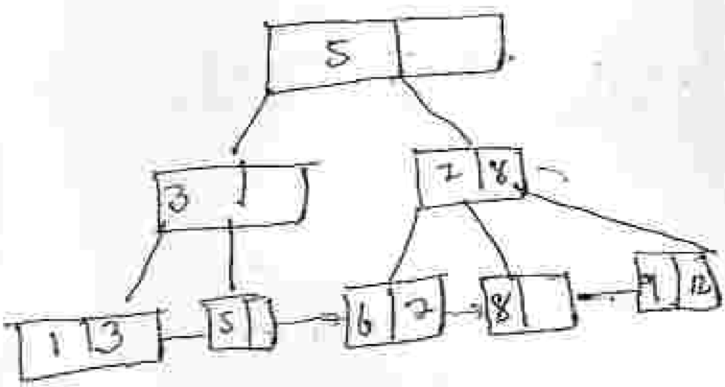
assumption



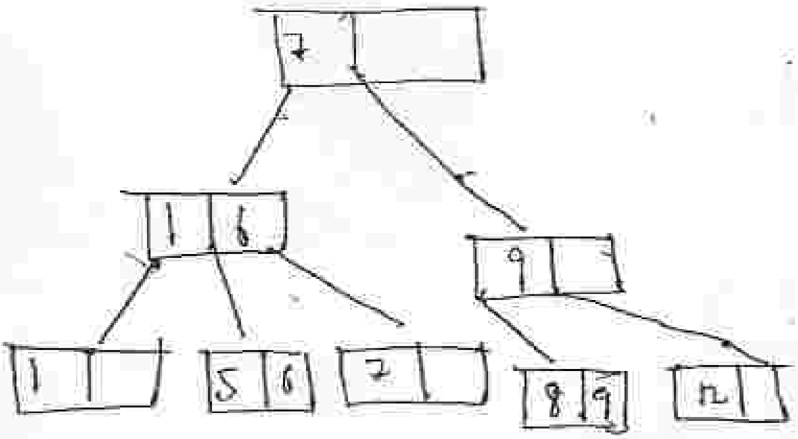

5
3
1 5
7 8


3 5
1 3
5
7 8
12


5
3
8
1 3
5
6
7 8
12
9

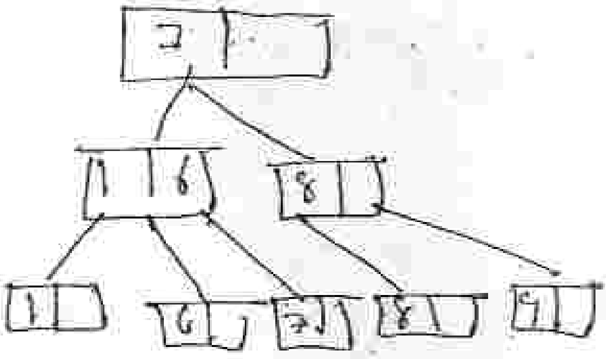

5
3
7 8
1 3
5
6 7
8
9 12

Deletion

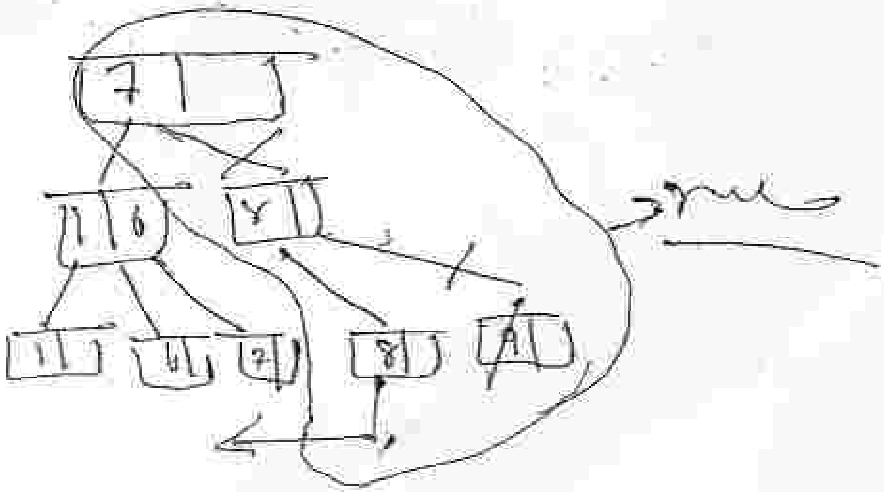


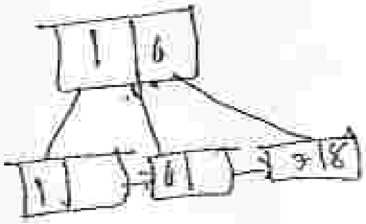
delete 5, 12, 9 in seq

→ D5 ; D12, 1 key mm , 1 ≤ 7 - maxke

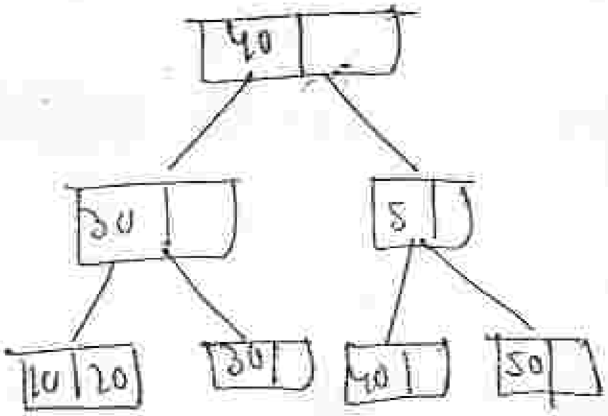


D9 → FMK





Q:



I : insert 15, 25

II : delete 50.

all key 15, 25 dater in that order, who many of the nodes will prest in the tree after two instations.

(a) 1 (b) 2 (c) 3 (d) 4

II delete 50 :

now the key 15 is deleted from the B+-tree consider the tree status —

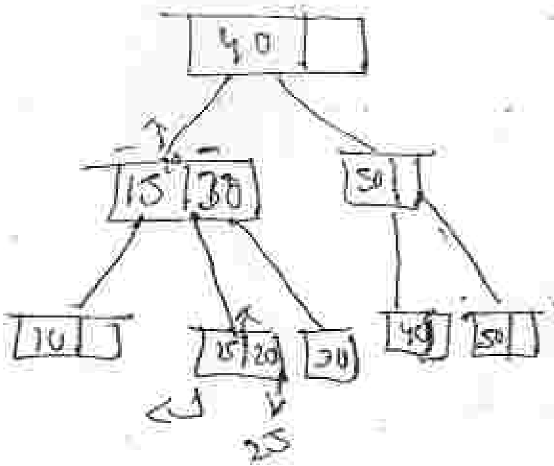
S1: height is same

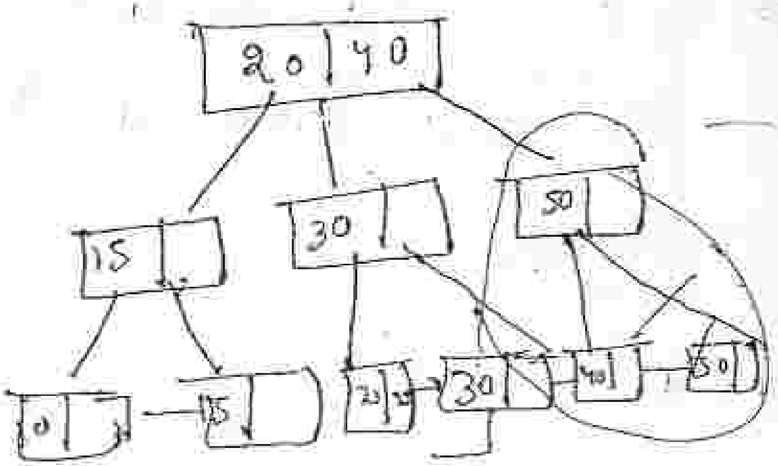
S2: $\boxed{20 \mid \ }$

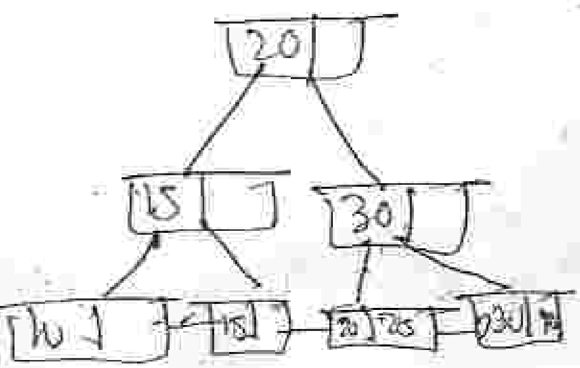
S3: Root remains unchanged

which of the above statements is true

a) S1, S2 b) S2, S3 c) S1, S3 d) all







9