DATABASE MANAGEMENT SYSTEM

Made Easy Class Toppers Latest Notes Databases
Copyright©Theorypoint.com

CONTENT:-

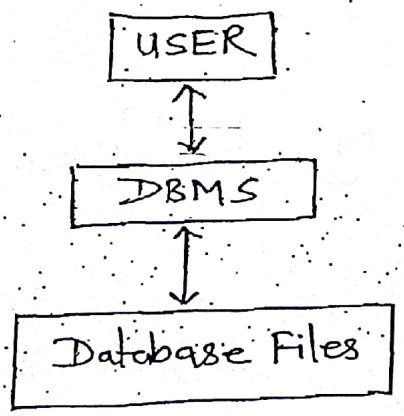
1. INTEGRITY CONSTRAINTS & ER MODEL
2. NORMALISATION
3. QUERIES a) RELATIONAL ALGEBRA
   b) SQL
   c) TUPLE RELATIONAL CALCULUS
4. FILE ORGANISATION & INDEXING
5. TRANSACTIONS & CONCURRENCY CONTROL

copyright©theorypoint.com The Best Gateway Of Success

INTRODUCTION TO DBMS

DATABASE: Database is a collection of related (similar) data.
eg set of Student Info.

Database Management System:- It is a software used to manage and access the DB files in more efficient way



DBMS acts as an interface b/w user & DB files

Flat Files Systems:- DB files managed by users using OS.

→ if DB size is in KB, MB [small DB] then flat file system can be used.

→ if DB size is in GB, TB [Huge DB] then flat file system

failed to manage DB.

Limitations of Flat File System

→ Too complex to access data from set of DB files.

→ More I/O cost (Access cost) to access the required data.

→ Less degree of concurrency.

→ Difficult to implement different levels of access control

Advantages of DBMS over flat file system

→ DBMS uses DATA INDEPENDENCY to access data.

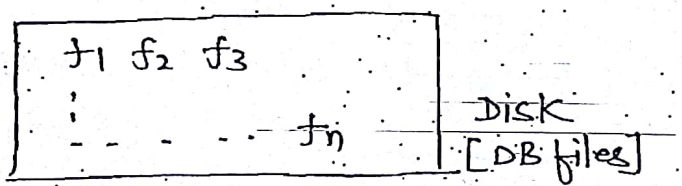
→ DBMS uses INDEXING to access required data.

→ Increase degree of concurrency by its own CONCURRENCY CONTROLLER

→ DBMS uses VIRTUAL TABLES to implement different levels of access control.

LIMITATIONS OF FLAT FILES [or Advantage of DBMS]

1) Difficult to access required data using flat files.



Storage Details

1. Name of file
2. Location of the File
3. Data format
4. File Format etc.

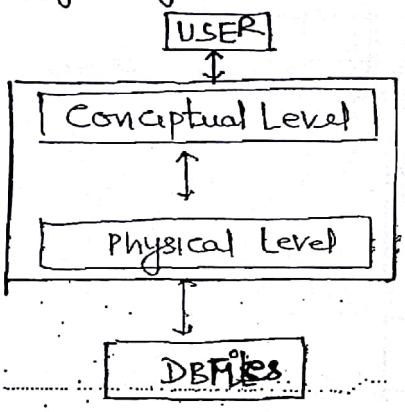
User need to know all these details about the file to access the data.

But if there is large amount of DB files, then it become hard to manage the data & know every details of the file or data stored.

To overcome this difficulty, DBMS implements Data Independency.

DATA DEPENDENCY

Data Dependency means user should be able to access data without knowing the storage details of the data. [Hiding physical details to the external users].



DBMS provides physical layer which has overall control of how data is stored in the Database.

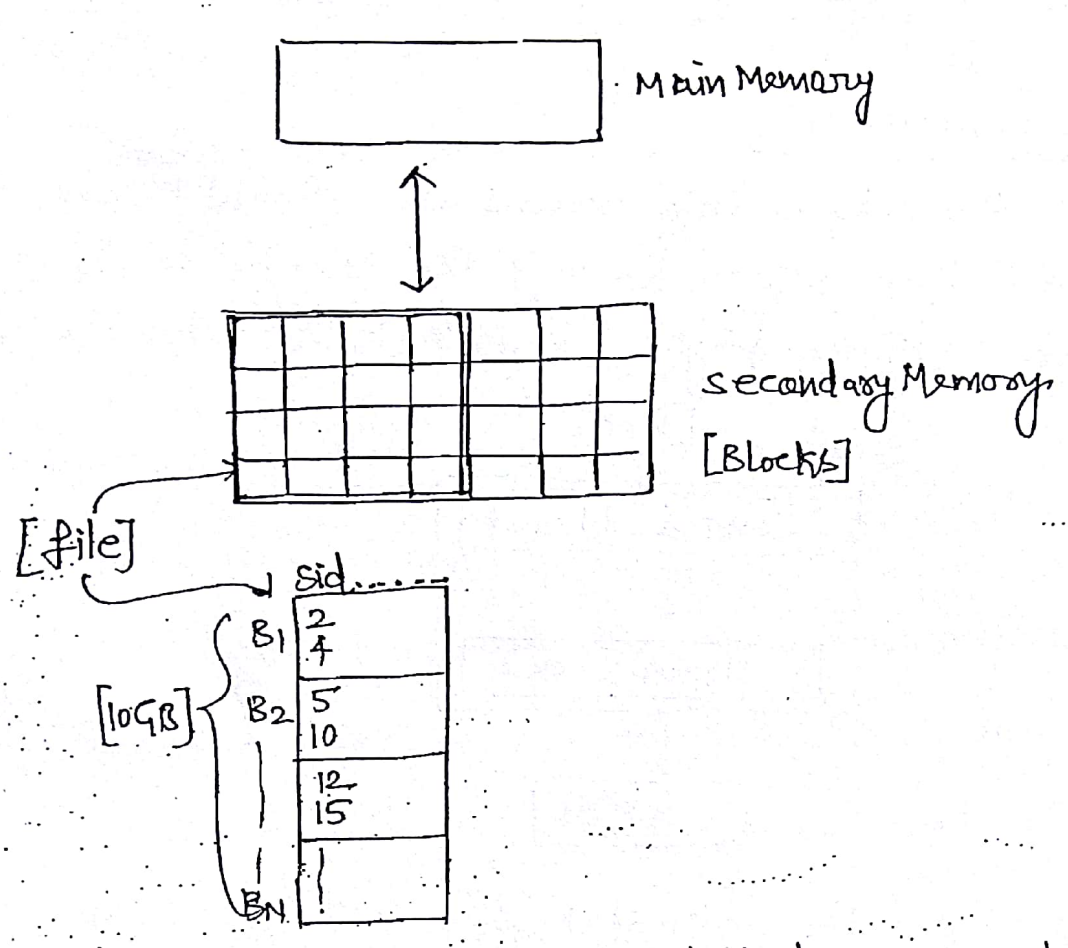
It provides conceptual layer which tells about what data is stored in Database.

ALSO DBMS provides an interface to user to access DB files.

→ The user is not allowed to change the details of the file. He is only allowed to access DB files.

→ The storage details of DB files called metadata is maintained by DBMS.

2) More Access Cost using flat files



In the secondary memory, whole student Management Data is stored. If that file is 10GB. then to access entry of

[student whose sid=x]

we require whole 10GB file to be put in main memory. or whole N blocks will be loaded into Main Memory.

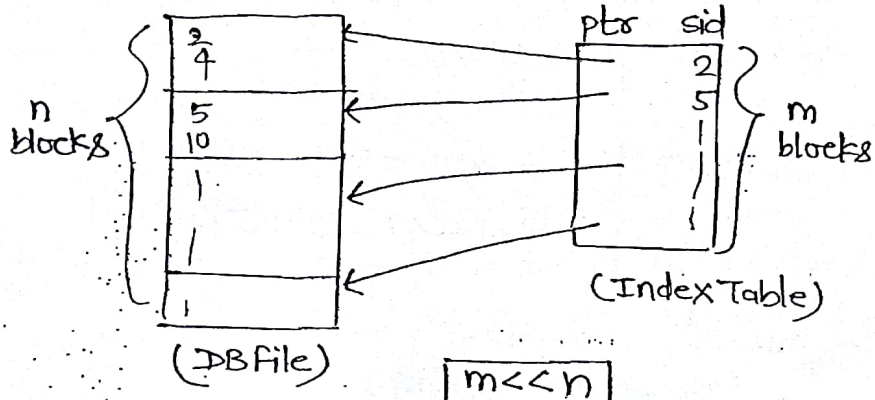
In worst case, all N Blocks will have to be loaded.

Access Cost : to bring N blocks from sec Memory to Main Memory.

To overcome this problem, INDEXING is used.

INDEXING (To reduce I/O cost or Access cost)

For every file stored in secondary memory, an index table is maintained for each file block. It contains number of the block & pointer to that block.



$m << n$

I/O cost without index $\Rightarrow$ n blocks

I/O cost with index $\Rightarrow$ (m+1) blocks

eg: if n = 1000 & m = 30

then

Access cost without indexing = 1000 blocks

Access cost with indexing = 31 blocks

) Less Degree of CONCURRENCY using flat files.

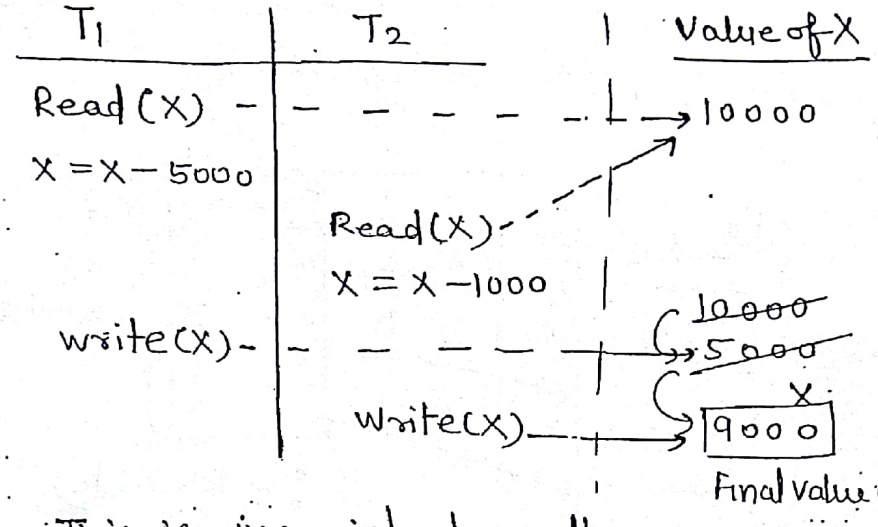
i) Concurrent Execution

→ Simultaneous access of same data by two or more users.

→ Database contain data accessed by multiple users simultaneously which may lead to inconsistency.

Problem of Inconsistency:-

Account X = 10000



This is inconsistent result.

(concurrent execution)

Thus simultaneous access by two persons, lead to improper or unordered execution which resulted in a inconsistency.

To avoid inconsistency in concurrent execution we implement concurrency control.

CONCURRENCY CONTROL

Method used to avoid inconsistency because of concurrent execution.

It allows the users to access DB simultaneously.

DEGREE OF CONCURRENCY

No: of users that can use database simultaneously.

⊛ Degree of concurrency should be high.

OS File Concurrency Control :

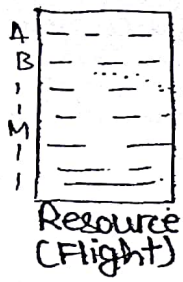
[over File Level]

Here file is a resource.

FILE ⟷ RESOURCE

For eg. complete flight is considered as a single resource. only one user is allowed to reserve a seat or book a ticket at a time.

This will help in case when two user want to reserve the same seat.



Resource (Flight)

But if a user is reserving seat in row A then another user must easily book a seat in row M but he is not allowed to do a simultaneous modification on same resource i.e. flight.

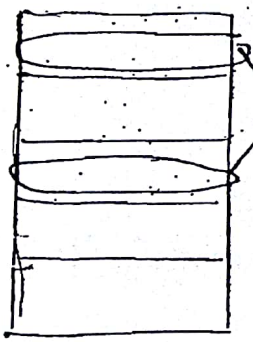
However, data inconsistency is still avoided in this case when they are accessing different seats (records) of same file.

Ⓐ Because of file level concurrency control using flat files, degree of concurrency is too less.

To overcome this problem, DBMS uses-

DBMS File CONCURRENCY CONTROL

Here a record of a file is considered as a resource.



Thus, no. of users who can access/modify the resource (record) i.e. seat in different rows becomes equal to no. of records contained in a file

Because of Record Level Concurrency Control in DBMS filesystem, degree of concurrency is very high.

Q Why indexing is not applied at OS Level?

Because OS is unaware of structure of file. However, DBMS have the power to structure the file into records. Each record having set of entities etc. That's why indexing is applied in DBMS, not in OS.

4) Difficult to implement different levels of Access Control [Security]

3 types of access granted :-

i) Not Allowed Access
ii) Read Only
iii) Read/Write Mode

Student Info

Sid	Sname	marks	Addr
S1	A	80	A1

Here users can be faculty and Admin.

For faculty [Users]

Sid	Sname	Smark
S1	A	80

(File 1)

This part of file is accessed by faculty.

For Admin [Users]

Sid	Sname	Addr
S1	A	A1

(File 2)

This part of file is accessed by admin

Here data is physically seperated.

Or each new table has to be maintained.

→ So, large memory is also required to maintain it

→ Also, this creates redundant data,
Now if a data has to be inserted, then it has to be inserted in all tables
or if deleted/updated from one table, it has to be

Otherwise it leads to data inconsistency.
To overcome this problem; DBMS provides VIEWS

VIEWS [virtual table]
↳ (almost like table but no storage entity)

In these tables, data is not seperated physically
Here all insertion / deletion / updation in virtual table or view is also made in the actual physical table.

Student (Sid	Sname	marks	Addr)	
[Physical Table] S1	A	80	A1	← Insert/ delete/ updation

View1 (Sid Sname marks)
[faculty]

View2 (Sid Sname Addr)
[Admin]

Because of views same data can be viewed differently by different users.
Access Control is possible with views without data redundancy.

→ In views, no seperate table is created for users with different access rights. Same data is represented in different fashion to different users.

Integrity Constraints for RDBMS :-

[RDBMS] Relational DBMS

[designed by Codd]

12 rules to design RDBMS

[RDBMS guidelines]

Codd Rule [RDBMS Guidline]:-

⊛ Data must be stored in tabular format in DB file

Student file

SID	SNAME	AGE
S1	A	20
S2	A	21
S3	B	21

Attribute/Field

Record / tuple

ARITY: # columns of DB table

Cardinality: # records of the DB table

Relation Schema:

DB Table Definition (Abstract details of the table)

eg. Student (sid sname Age)

& key constraints.

Relational Instance: (SNAPSHOT)

It is the Record Set of DB table at particular instance of time.

copyright©theorypoint.com The Best Gateway Of Success

④ No two records of the DB table is same.

Candidate key: Minimal set of Attributes used to differentiate records of the relation uniquely

* Candidate key depends on type of data user stores in DB file.

exp ① student (sid, sname, age)

SID : candidate key

So, here no two records in the DB will have same SID.

SID, SNAME : Candidate key ✗

because it is not minimal set of Attributes to uniquely identify records in DB.

exp ② Enroll (sid, cid, fee)

Assumption: student can enroll in many course;
course can be enroll by many students.

→ In this case, the candidate key depends on the constraints applied on the data in DB.

→ As, here one SID could contain multiple CID's (SID value can repeat)

also one CID can be enrolled by many students. (CID can repeat)

Thus, minimal attribute set to uniquely differentiate records is (SID, CID).

eg ③ what can be the candidate key of

A	B	C
2	4	3
2	4	6
2	5	3
2	5	6
4	4	3

Also, pair of two attributes is also not unique

As none is unique. So, candidate key: (A,B, C)

If for a relation R(X---) , X is a candidate key, then-

i) No two records whose X values are same should exist.

ii) Proper subset of X should not be able to differentiate records uniquely.

eg ④

(A relation can contain more than one candidate key.)

Emp

Eid	Ename	PPno.	Pan ID	Bank	Account
E1	–	P5	X1	SBI	101
E2	–	NULL	X2	SBI	102
E3	–	P2	X3	ICICI	101
E4	–	NULL	X4	ICICI	102

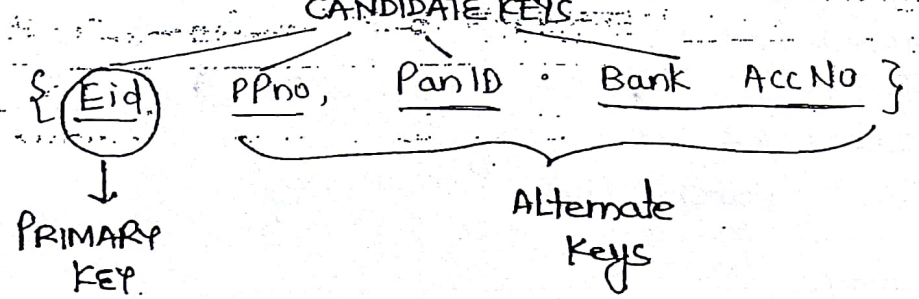
Assume No 2 records will be their with same Bank & Account No , then possible candidate keys for Emp Table –

i) Eid

ii) PPno (Passport No.)

iii) PAN ID

iv) Bank, Account



Any one of the candidate key must be chosen as Primary key.

Primary Key –

Any one candidate key with Field values must be "NOT NULL".

NULL: Unknown ~~keys~~ / UnExisted Values

Primary key Constraints –

1) All Candidate key Constraints are primary key constraints.
2) NULL values are not allowed over primary key fields.
3) Almost only one primary key is allowed for any DB table.

Alternative key –

All candidate keys of the Relation except Primary key.

Alternative Key Constraints –

1) All candidate key constraints are alternative key constraint
2) NULL values are allowed over alternative key fields.
3) More than one alternative key allowed for relation.

TABLE SCHEMA FOR EMP :-

```
Create table Emp
(Eid      varchar(10), Primary key,
 Ename    varchar(30),
 PPno     varchar(15), UNIQUE,
 PanID    varchar(10), UNIQUE NOT NULL,
 Bank     Varchar(10),
 AccNo    Integer(10),
 UNIQUE   (Bank, AccNo)
);
```

(*) All the key constraints are under

INTEGRITY CONSTRAINTS

↙ CORRECTNESS ↘ CONDITIONS

(conditions which are followed to achieve correct data are called integrity constraints.)

(*) RDBMS relation must consist atleast one candidate key and atleast one candidate key with NOT NULL values.

```
Create Table R
(A  int  Unique NotNull,
 B  int  Unique
);
```

R is a relation, although it does not contain Primary key but has atleast one unique key with NOT NULL constraint.

Simple Candidate key: Candidate key with only one attribute.

Compound Candidate key: Candidate key with atleast two attributes.

Prime Attribute: Attribute belongs to some candidate key of Relation. eg, Eid, PPNO, PanID, BankAccNo etc.

Non Prime Attribute: Attribute not belongs to any candidate key of relation. eg. Ename.

eg R(ABCDEF)

Candidate keys = {A, BC, CD}

then

Prime Attribute set of R = {A, B, C, D}

Non Prime Attribute set of R = {E, F}

Super key

set of attributes used to differentiate records uniquely which may or may not be minimal attribute set.

eg, R(sid sname age)

if we say {sid} : Candidate key

then

sid
sid sname
sid age
sid name age

} Super Keys

sid age → means no 2 records have same (sid age)

Ⓐ All candidate keys are super keys but some super keys may not be candidate keys.

Super key set is a superset of candidate key set

Candidate key set

Super key Set

X is superkey of R
iff some subset of X must be a candidate key

(OR)

Minimal Superkey is a candidate key.

Q. R(A,B,C,D)
A: candidate key
How many superkeys in R?

Sol:- A combine with any subset of (B,C,D) will become a superkey.

subsets of (B,C,D) = 8 ($=2^3$)

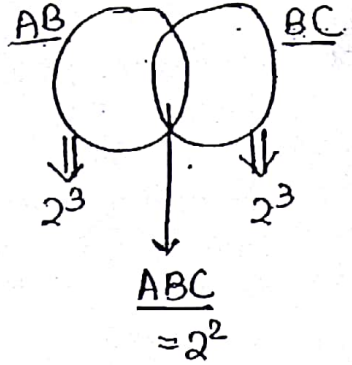
superkeys of R = 8.

{ A; ABC
AB, ABD
AC, ACD
AD, ABCD }

Q. R(A,B,C,D,E)
{AB, BC} : candidate key
How many superkeys are in R?

Using set theory,

superkeys for cand key AB = AB set
Superkeys for cand key BC = BC set
superkey common in AB, BC = having ABC
= # superkey for ABC (i.e. intersection)



ABC [DE]
↓
2^2

superkeys = 8+8−4 = 12 superkey

method2:-

$$\#\text{ of superkeys} = [\#\text{ of superkeys among prime}] * 2^{\text{NO. of None Prime}}$$

↓

{AB, BC, ABC}

# of Non primes = {D, E} $= 3 * 2^2 = 12$ superkeys.

Q R(A1, A2, A3 .... An)

How many superkeys in R if

a) {A1} : cand key

A1 #subset of ($\overbrace{A_2, A_3 \ldots A_n}^{(n-1)}$)

$= 2^{n-1}$

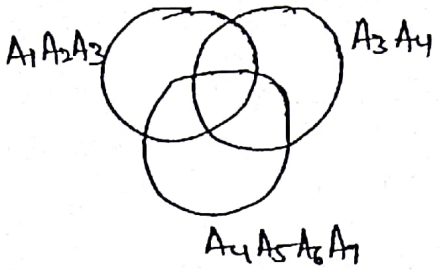
b) {A1, A2A3} : cand key

$$\left[\begin{array}{l} A_1 \\ A_1A_2 \quad A_1A_3 \\ \hline A_1A_2A_3 \\ A_2A_3 \end{array}\right] (5) * 2^{n-3} \Rightarrow 5 * 2^{n-3}$$

c) {A1, A2, A3} : cand key

(7 superkeys among prime) $* 2^{n-3} \Rightarrow 7 * 2^{n-3}$

d) $\{A_1A_2A_3, A_3A_4, A_4A_5A_6A_7\}$ are cand keys.



$$2^{n-3} + 2^{n-2} + 2^{n-4} - 2^{n-4} - 2^{n-5} - 2^{n-7} + 2^{n-7} = \underline{11 * 2^{n-5}}$$

e) every attribute is cand key.

$\{A_1, A_2, A_3 \ldots A_n\}$ cand key

$${}^nC_1 + {}^nC_2 + {}^nC_3 + \ldots + {}^nC_n = 2^n - 1$$

(empty set can't be cand key)

Maximum possible superkeys for a relation come in those case when all attributes are itself cand keys.

f) all attributes form one cand key

R $\{\underline{A_1A_2A_3 \ldots A_n}\}$ → cand key

Only 1 superkey.

Minimum possible superkeys for a relation come in that case when all attributes together form a cand key.

Foreign key

(used to relate database tables)

ex

Student (sid, sname, age)

sid	sname	age
S_1	A	20
S_2	B	21
S_3	B	21
S_4	C	18

All students

Enroll (sid, cid, fee)

sid	cid	fee	
S_1	C_1	–	
S_1	C_2	–	
S_4	C_2	–	
S_{10}	C_1	–	not allowed

students enrolled atleast one course

Foriegn key: Set of attributes used to reference primary key / Alternative key of the same relation / some other relation.

ex

R (A, B, C)

A	B	C
2	–	NULL
4	–	2
5	–	4
6	–	2
7	–	5
8	–	10 ✗

R (A, B) — P.K

A	B
2	–
4	–
5	–
6	–
7	–

S (C, D) — F.K

C	D
NULL	d1
2	d2
2	d3
5	d4
4	d5

10 is not a value in A

- Foriegn key can be NULL
- Foriegn key if is in same/diff. table referring to Prim. key then it should contain only those values which have come in the primary key value of table.

The table which contains foriegn key is known as Referencing table.

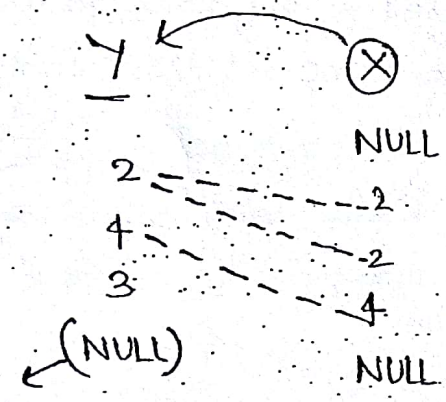
The table which is referred by foriegn key is known as Referred table.

Referential Integrity Constraint [Foriegn key Condition]

Non Null Values of X must belong to values of Y.

'Y': Referrenced Attribute

'X': Referencing Attribute
[Foriegn Key]



incase
Y is alternate key.

Referential Integrity Constraint [Foriegn Key IC]

a) Referenced Relation (Parent Table) → [student]

i) Insertion :- No Violation

Inserting any value to parent table will not violate foriegn key cond^n

ii) Deletion :- May cause Violation

[1] ON DELETE NO ACTION

If data used in Referencing table then Not allowed to delete data from Referenced Relation.

eg: To delete S_1 from student (parent table) –

where sid = S1;

then only we are able to delete data from parent relation

2) Delete From Student
where sid = S1;

} we require 2 queries.

[2] ON DELETE CASCADE :-

Because of deletion of record from Referenced Relation, DBMS is forced to delete related records from Referencing relation also.

Here no requirement of 2 queries.

(ii) Both preserves integrity constraint.

[3] ON DELETE SETNULL :-

If FK field allowed to set NULL

then on deletion of referenced record sets NULL value / is assigned to related foreign key Field

Otherwise deletion from Referenced table is not allowed

(iii) on delete ~~NO ACTION~~ SET NULL is similar to on delete ~~SETNULL~~ NO ACTION in the case when FK is ~~not~~ allowed to set NULL.

iii) Updation :- May Cause Violation

i] ON UPDATE NO ACTION

ii] ON UPDATE CASCADE

iii] ON UPDATE SET NULL

b) Referencing Relation (Child Table)
↳ [Enroll]

i) Insertion : May cause Violation

ii) Deletion : No Violation

iii) Updation : May cause Violation

(Ψ) Because of Insertion & Updation, if violation is caused then insertion & updation are restricted.
No other solution is possible in case of violation.

Create table with Foreign key :-

```
CREATE table Student
  (sid  varchar(10)  Primarykey,
   sname  varchar(30);
   age  integer(3)
  );

CREATE  table Enroll
  ( sid  varchar(10),
    Cid  varchar(10),
    fee  integer,
    primary key (sid,Cid),
    FOREIGN KEY (sid)
      References to student(sid)
      ON DELETE CASCADE → means ON UPDATE NO ACTION
                          for ON UPDATE CASCADE
                          "we have to
  );
```

have to be declared

Referential Integrity Constraint

Emp (EId Ename Sup ID) Superior is also Employee

	E_1	A	NULL	deleted
deleted due to "ON DELETE CASCADE"	E_2	A	E_1
	E_3	B	E_1
	E_4	C	E_2
	E_5	B	NULL
	E_6	C	E_5

If on delete CASCADE is used & E_1 entry is deleted then entry for E_2 & E_3 will be deleted as E_2 is deleted so E_4 will also be deleted.

∴ on delete SetNull is better here.

```
Create table Enroll
( EId  varchar(10)  Primary key,
  Ename  varchar(30),
  Sup ID  varchar(10),
  Foriegn key (Sup ID) references
      to Emp(EId) ON DELETE SETNULL);
```

NORMALIZATION

It is used to eliminate / reduce redundancy in DB relations.

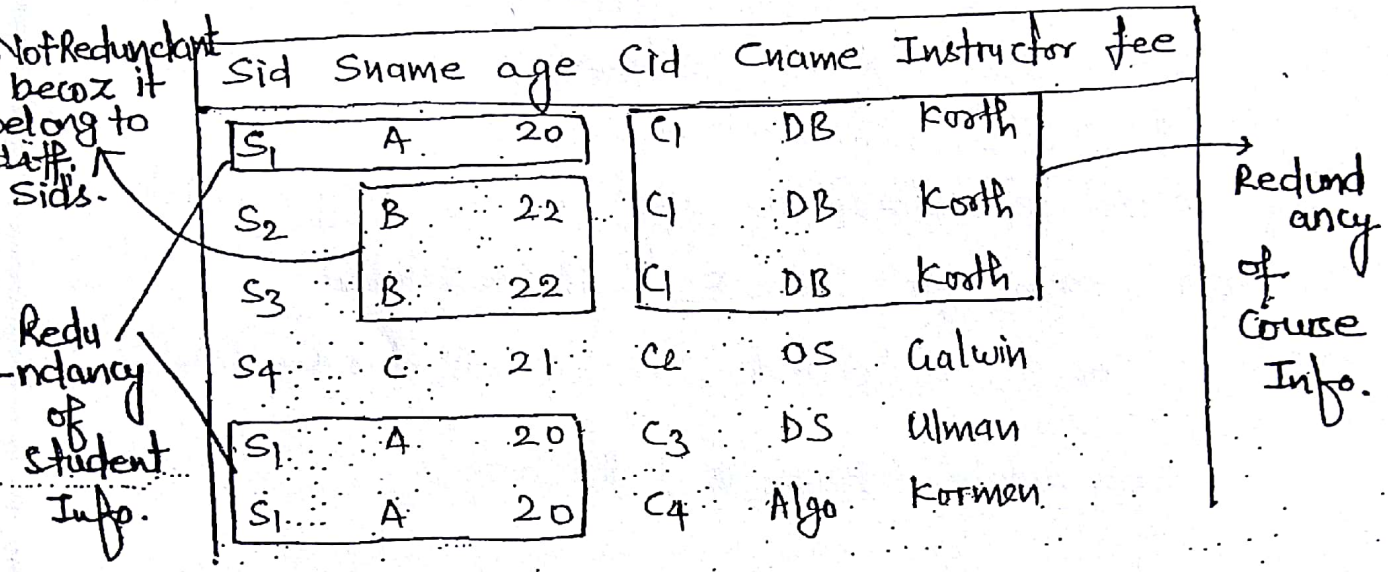
⇒ Redundancy can occur in DB relations if two or more independent relations stored in single Relation

Independent Relations in DB table →

eg. Sid → Sname age

Cid → Cname Instructor

Sid Cid → fee



→ Same course info. or same student info. repeated in DB relation (table), this is called redundancy in DB table

→ when (whole info) means all the attr. of any independent relation repeats, only then we consider it as redundant data, eg. S2 B 22 is not S3 B 22 redundant data. because it belongs to different entities.

→ Here (Sid, Cid) is considered as Cand. Key

PROBLEMS BECOZ of REDUNDANCY (WHY NORMALISATION?)

(*) Becoz of Redundancy, Inconsistency can occur during Insertion / Deletion / Updation of data record.

DB Anomalies

Updation Anomaly:-

ex if someone writes query as

```
Update Student set age = age+1
where Cid = C3
```

Here, updation is done on basis of Cid but not with Sid, there came inconsistency becoz

where Cid = C3 there S_1 age = 21

but S_1 age at other places is 20
for eg at 1st & 6th tuple

So inconsistency arises.

(*) if some redundant copy updated but other copies are not updated, then DB becomes inconsistent.

Deletion Anomaly:-

ex if we have to delete info. of student S_4 then course details of course C_2 as will also get deleted. Loss of course info. will occur.

Becoz of deletion of some data, it is forced to delete other data.

Insertion Anomaly:-

ex if we want to add course info. of some new course then we also have to add some student info. corresponding to it.

Here we call that student info. as dummy data.

This dummy data causes inconsistency.

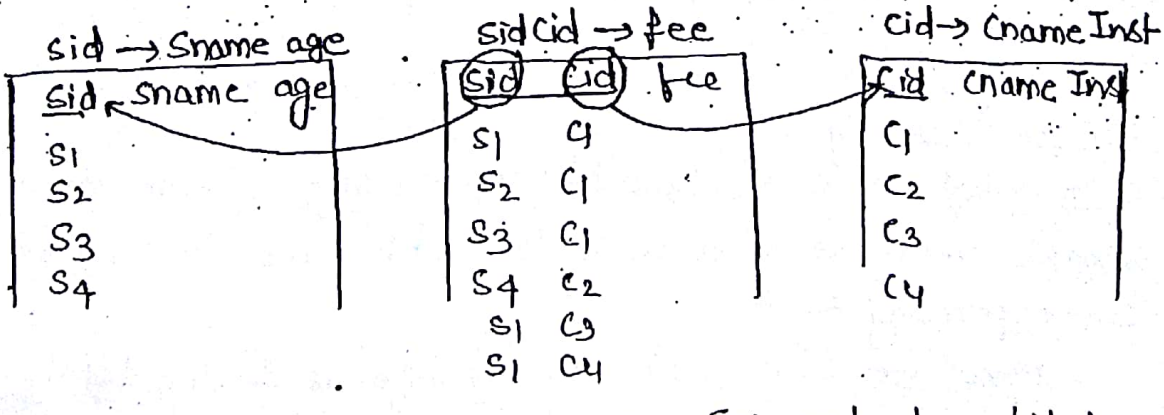
sid	sname	age	Cid	Cname	Inst.	fee
S_1	A	20	C_1	DB	Korth	
S_2	B	22	C_1	DB	Korth	
S_3	B	22	C_1	DB	Korth	
S_4	C	21	C_2	OS	Galvin	
S_1	A	20	C_3	DS	Ullman	
S_1	A	20	C_4	Algo	Hayes	
××	××	××	C_5	CO	Corema	

Here there will be actually 4 students but if someone accesses student table, he will get 5 students which leads to inconsistency

To reduce/ eliminate this kind of problems due to redundancy and inconsistency we decompose a relation to normalise it.

Decomposition:-

Split Relation R into two or more SubRelation to reduce/ eliminate redundancy.



Normalised DB Design [No redundancy/ No Anomalies]

So here, no insertion/deletion/update anomalies will exist.

Goal of Normalization

No Redundancy → Hence No Anomalies.
(To eliminate Redundancy)

FUNCTIONAL DEPENDENCY

So here, no insertion/deletion/update anomalies will exist.

Goal of Normalization

No Redundancy → Hence No Anomalies.
(To eliminate Redundancy)

FUNCTIONAL DEPENDENCY

FUNCTIONAL DEPENDENCY ($X \rightarrow Y$)

R

X	Y	....
x_1	y_1	
x_1	y_1	
x_2	y_2	
x_3	y_1	
x_1	y_1	
x_3	y_2	

$X \rightarrow Y$

For each X value there must be only one Y value.

Two different values of x can have the same values of y but if any value of x repeats, then the value of y must be same corresponding to it.

Formal Definition:

If t_1 and t_2 are two independent tuples, in relation R, then $X \rightarrow Y$ Exists in Relation R only if

$t_1 \neq t_2$

$t_1.X = t_2.X$ then $t_1.Y = t_2.Y$

Normalization takes place only on the functional dependencies.

Q. R(A B C) A: Cand key

$a_1\ b_1\ c_1$

$a_2\ b_1\ c_1$

$a_3\ b_2\ c_1$

$a_4\ b_3\ c_2$

which of the following FDs are correct?

i) $A \longrightarrow B$

always true as A is Cand key (No two values of A in two diff. tuple could have the same values and so every a_i has its own b_i.

ii) $A \rightarrow C$

always true (same reason)

Thus, in a FD if LHS (determinant) is a superkey then it will always be a valid FD.

iii) Possible because by table we can see that for every value b_i in B there is unique value c_i in C.

So, C can be determined by B.

iv) $B \rightarrow A$

Not Possible because same b_i values in B have different a_i in A

v) $C \rightarrow B$

Not Possible

vi) $C \rightarrow A$

Not Possible (same as iv)

Sid	Sname
S_1	A
S_2	B
S_3	C
S_4	C

Sname → Sid

(Not Exists)

→ We can say that a functional dependency does not exist by analysing the instance of database but saying that functional dependency exists by analysing instance of database is not correct.

→ Existence of a functional dependency is decided at the time of database design.

3 types of Functional Dependency :-

1) Trivial FD:-

$X \rightarrow Y$ is trivial FD if $Y \subseteq X$

where X & Y are attribute sets of R.

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_1

Sid → Sid

Sname → Sname

Sid Sname → Sid

Sid Sname → Sname

Sid Sname → Sid Sname

(*) Every possible trivial FD is implied in relation.

(*) Trivial FD are present by default in every DB table, no condition can violate trivial FD.
(It is not possible to violate any trivial FD in any database table)

2) Non Trivial FD :-

$X \rightarrow Y$ is non trivial in R

only if $X \cap Y = \phi$ (i.e. no common attribute in X & Y Attribute sets)

and $X \rightarrow Y$ should satisfy FD definition. (i.e. same value of X should have same corresponding values of Y)

Ex. Sid → Sname
Sid Cid → Sname
} Non Trivial FD set of Relation R

Sname → Cid
Cid → Sid
} Not Non Trivial FD.

3) Semi Non Trivial FDs

Combination of Trivial & Non Trivial FD.

- Sid → Sid Sname (It is not trivial (as $Y \not\subseteq X$) & also not pure Non trivial (as $X \cap Y \neq \phi$))

  we can represent this FD by seperating at RHS,

  Sid → Sid (Trivial)

  Sid → Sname (Non Trivial)

- Sid Cid → Cid Sname

  Sid Cid → Cid (Trivial)

  Sid Cid → Sname (Non Trivial)

- AB → BCD

  AB → B (Trivial)

  AB → CD (Non Trivial)

⇒ Semi Non trivial FD will depend on non-trivial part of FD because

- The trivial FD part will always be true
- but if Non Trivial Part of FD true then only the given FD is true otherwise not.

ex

Sid → Sid (Always true)

Sid → Sname (true for given table)

↳ ∴ Sid → Sid Sname is true

So, semi Non Trivial FD is true only when Non Trivial part of FD is true.

Q.

① R

A	B	C
2	3	4
2	4	5
2	3	6
2	4	4

Find Non trivial FD set of R based on given instance.

Possible Non trivial FDs:

A → B X	B → A ✓	C → A ✓	AB → C X
A → C X	B → C X	C → AB X	BC → A ✓
A → BC X	B → AC X	C → B X	CA → B X

Non trivial FD set for this instance = {B → A, C → A, BC → A}

(2)

R	A	B	C
	4	4	3
	4	4	4
	4	6	3
	4	6	4
	2	4	3

$B \to A$: ✗

$C \to A$ ✗

$BC \to A$ ✗

No Non Trivial FD set.

(4) There can be a possible relation with No NonTrivial FD set.

Armstrong's Rules over FD's :-

1] **Reflexivity :-** $X \to X$ is always true (∵ it is trivial)

2] **Transitivity :-** if $X \to Y$ & $Y \to Z$ exists then $X \to Z$ must also exist

$\{X \to Y, Y \to Z\} \Rightarrow \{X \to Z\}$

we can add another FD

But reverse is not true.

3] **Augmentation :-** if $X \to Y$ exists, Then $XZ \to YZ$ also exists.

If we prove $X \to Y$ is true then we can say that $XZ \to YZ$ is true.

But reverse is not true.

4] **Split & Merge Rule :-** If $X \to YZ$ exists then $X \to Y$, $X \to Z$ also exists

& if $X \to Y$, $X \to Z$ exists then $X \to YZ$ also exists.

If we can prove $\{X \to YZ\}$ then $\{X \to Y, X \to Z\}$ is also true.

Reverse is also true, So, $\Leftrightarrow$ exists.

Note: $A \to BC \Rightarrow$ ($A \to B$, $A \to C$) always true

$AB \to C \Rightarrow$ ($A \to C$, $B \to C$) may / may not be true.

But Left side split is only allowed & is always true.

ex

A	B	C
2	4	3
2	5	4
4	2	4
4	5	3

$AB \rightarrow C$ } True

$A \rightarrow C$ } False
$B \rightarrow C$ }

* We can generate any new FD by given FDs only if it follows these Armstrong Rules.

Attribute Closure (X^+) :-

X^+ = { set of all attributes determined by X }

ex { $AB \rightarrow C$, $C \rightarrow D$, $DE \rightarrow F$, $BC \rightarrow E$, $EG \rightarrow H$ }

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

- A, B : as $AB \rightarrow AB$ (reflexivity)
- C : as $AB \rightarrow C$
- D : as $C \rightarrow D$
- E : as $BC \rightarrow E$
- F : as $DE \rightarrow F$

Now, we can also add "$AB \rightarrow ABCDEF$" in the above FD set.

Because,

- $AB \rightarrow AB$ (Reflexivity)
- $AB \rightarrow C$ (Given)
- $AB \rightarrow D$ (Transitivity, $AB \rightarrow C$, $C \rightarrow D \Rightarrow AB \rightarrow D$)
- $AB \rightarrow E$ ($AB \rightarrow C$, $AB \rightarrow B \Rightarrow AB \rightarrow BC$ and $BC \rightarrow E$, $AB \rightarrow BC \Rightarrow AB \rightarrow E$)
- $AB \rightarrow F$ ($AB \rightarrow C$, $C \rightarrow D \Rightarrow AB \rightarrow D$; $AB \rightarrow D$, $AB \rightarrow E \Rightarrow AB \rightarrow DE$; $AB \rightarrow DE$, $DE \rightarrow F \Rightarrow AB \rightarrow F$)

Implied in above FD set.

Membership test :-

Given FD set f and $X \to Y$ is FD.
This is to be proved.

So, $X \to Y$ is implied (member of / belongs to) in FD set f iff X^+ in FD set f consists Y.

"$X \to Y$ is a member of FD set f" is denoted as

$$F = \{ \ldots\ldots \} \models X \to Y$$

eg
$F = \{ AB \to C, C \to D, BD \to E, E \to F \}$ is the FD set F

Test a) $AB \to F$ implied in FD set (F) or not?

b) $ABGH \to EF$

c) $BD \to C$

Sol:-

a) $(AB)^+ = \{ A, B, C, D, E, F \}$

$\because (AB)^+$ consists of F so, $AB \to F$ is member of FD set f.

b) $(ABGH)^+ = \{ A, B, G, H, C, D, E, F \}$

$\because (ABGH)^+$ consists of EF so, $ABGH \to EF$ is member of FD set f.

we don't have to think here about attr. not present in F; only find the closure.

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

$\because (BD)^+$ doesn't consists C. So, $BD \to C$ is not a member of FD set f.

Superkey

Let R(X.....) be a relation with superkey X then X should not have two records same.

(A) If X is superkey of R(X.....) then X should imply all the attributes of R.

(A) X is superkey of Relation R iff X^+ should determine all attributes of relation R.

$$[X \text{ is superkey of } R] \overset{\text{iff}}{\Longleftrightarrow} [X^+ = \{\text{All Attributes of } R\}]$$

(Necessary & sufficient cond^n)

Q. Find the superkey of relation R(ABCDE) with FD set $F = \{AB \to C, C \to D, B \to E\}$

Sol:- $(AB)^+ = \{A, B, C, D, E\}$

$\therefore$ AB is superkey.

If AB is superkey, then any superset of AB will also be a superkey or any set having AB as a subset will also be a superkey of R(ABCDE).

i.e. $(ABC)^+$, $(ABCDE)^+$ will also be the superkey of R

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

$\therefore$ A is not in BC^+ so, BC can't be superkey.

Candidate key:

It is the minimal superkey of R.

⊛ X is candidate key of R iff a) X must be superkey
or X must be unique
or X^+ = {All attributes of R}.

(and)

b) No Proper subset of X is Superkey

- Prove AB is a cand key.

Sol:- a) AB must be superkey

and b) A is not SK.

B is not SK.

Eg: R(A, B, C, D, E)

{AB → C, C → D, B → E}

AB is the cand key because

AB is a superkey.

(and)

A or B is not SK.

as $A^+ = \{A\}$

$B^+ = \{B, E\}$.

⊛ R(X Y Z W) {X → Y, Y → Z, Z → Y}

The attributes which are guaranteed to be part of candidate key are X and W.

Because, Y and Z come in right side of a FD so, they can be determined by any other attribute of R. This case removes Y & Z to become a part of cand key of R.

(P) $X \rightarrow Y$

If attribute X belongs to determinant of NonTrivial FD in R but not belongs to determiner of any NonTrivial FD of R, then we can say that X will be gurantee a part of every candidate key.

(OR)

If attribute X not belongs to nontrivial FD set of Relation R. Then X must be part of every candidate key of R.

Ex (1) R(A B C D E F)

$\{A \rightarrow C, \; D \rightarrow E, \; F \rightarrow A\}$

So, B, D, F will belong to every candidate key of R.

also, $(BDF)^+ = \{B\ D\ F\ E\ A\ C\}$

So, BDF will be a superkey.

We can't say that any subset of BDF will be a superkey because B, D, F must belong to every cand key of R.

∴ both cond^{n}s are satisfied,

Hence (BDF) will be the candidate key.

Ex (2) R(A B C) No Nontrivial FD in R

then all attributes of such set will be part of cand. key

OR (A B C) will be only cand key.

Continued → R(A B C)

2	3	4
2	3	5
2	4	4
2	4	5
3	3	4

—No Non-trivial FDs are present here.

So, All attributes of relation will be part of cand key

(ABC) : Cand key

and A or B or C or any subset of ABC can't be a superkey.

(*) More than one cand. key can be possible for a relation.

Q. How to decide that more than one cand. key is possible?

Ans. If Non-trivial FD $X \to Y$ with Y as prime Attribute in Relation R Then R should consist atleast two candidate keys.

$$X \longrightarrow \underbrace{Y}_{\text{prime}}$$

(*)

If (WY) is a cand key

i.e. $(WY)^+ = \{$ All attributes of $R\}$

and $X \to Y$ is implied in FD set F of R

Cand Keys →

W(Y) — then (WX) will also be a cand. key.

W(X) — determinant is replaced by determinant to obtain another cand. key.

W Z

Now if $X \to Z$ is implied in F

then

(WZ) is also be a cand key.

Q. Find all candidate keys of R(ABCDEF)

$\{AB \to C,\ C \to DE,\ E \to F,\ F \to B\}$

Sol:- By hit and trial

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

AB is cand. Key

A is guranteed to be a part of cand key but A^+ contain not all elements. So we have to add an attribute i.e. (AB) and AB^+ contain all elements of R

∴ AB is a candidate key.

because B^+ also not contain all attributes of R.

Now,

B comes in the determinant part of a FD

i.e. $F \to B$

So, F can also be a part of other cand. key.

we found that $AF^+ = \{A\ F\ B\ C\ D\ E\}$

So, AF is a second cand. key.

Now,

E comes in determinant part of

i.e. $E \to F$

So, $(AE)^+ = \{A\ E\ F\ B\ C\ D\}$

∴ AE is third Cand Key.

Now,

$C \to D(E) \to$ Prime

So, $(AC)^+ = \{A\ C\ E\ F\ B\ D\}$

∴ AC is fourth cand key.

copyright©theorypoint.com The Best Gateway Of Success

Q $R = (ABCDEF)$

$\{AB \to C,\ C \to D,\ CD \to EA,\ DE \to F,\ EF \to B\}$

① $(AB)^+ = \{A, B, C, D, E, F\}$

AB is cand key. as A^+ or B^+ not contain all elements of R.

Now,

② $EF \to$ (B) prime

So, $AEF^+ = \{A\ EF\ B\ C\ D\}$ AEF is cand key

No subset of AEF is superkey

$CD \to$ (A) prime

$BCD^+ = \{B, C, D, E, A, F\}$ BCD is cand key X

$DE \to$ (F)

$DE^+ = \{D\ E\ F\ B\}$

$C \to$ (D)

$C^+ = \{C\ D\ E\ A\ F\ B\}$ C is cand key

③ $DE \to$ (F) prime

So, $ADE^+ = \{A\ D\ E\ F\ B\ C\}$ ADE is cand key

④ $CD \to$ (EA) prime

$(CD)^+ = \{C\ D\ E\ F\ B\ A\}$

but C is cand key.

So, CD^+ is not a cand key

∴ AB, AEF, C, ADE are cand key

(H) R(A B C D E)

$\{A \to BC, CD \to E, B \to D, D \to A\}$

By Hit and Trial,

$A^+ = \{A\ B\ C\ D\ E\}$ So, A is cand key.

Now

$D \to A$

So, $D^+ = \{D\ A\ B\ C\ E\}$ So D is cand key

Now,

$B \to D$

So, $B^+ = \{B\ D\ A\ C\ E\}$ So B is cand key

{A, D, B} are cand key.

To verify the candidate keys, now check the FDs one by one & check whether they are candidate keys or not and if they are candidate keys, then they are present in the candidate key set or not.

R(ABCD) $\{AB \to CD \quad C \to A \quad D \to B\}$

By Hit & Trial,

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

$C \to A$

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

$D \to B$

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

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

So,

{AB, BC, AD, CD} are cand. keys.

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

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

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

CE

DE

{AE, BE, CE, DE} are cand keys.

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

Here E and F not belongs to R.

So, we will eliminate E & F,

we can see $BC \to F \quad F \to D \Rightarrow BC \to D$

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

Now $AB^+ = \{A\ B\ C\ D\}$ So AB is cand. key.

FD Set Closure :- $[F^+]$

F^+ = { set of all possible FDs which can be derived from given FD set F }

eg $F = \{ A \to B , B \to C \}$

$F^+ = \{$
- $A \to B$
- $AC \to BC$ (Augmentation)
- $AC \to B$ (Split)
- $AC \to C$
- $B \to C$ (transitive) $A \to C$
- $AB \to AC$
- $AB \to A$
- $AB \to C$
- $AB \to BC$
- $AB \to B$
- $AB \to C$ $\}$ (Etc.)

(9) R(ABC)

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

$|X| \longrightarrow |Y|$

no. of attributes in X

		# FDs
|0| where X has 0 attr. i.e. ϕ	$\phi^+ = \phi$	$\phi \to \phi$ (1)
|1| X has 1 attr. i.e.	$A^+ = ABC$	$A \to \phi$, $A \to A$, $A \to B$, $A \to C$, $A \to BC$, $A \to AB$, $A \to AC$, $A \to ABC$
	$B^+ = BC$	$B \to \phi$, $B \to B$, $B \to C$, $B \to BC$
	$C^+ = C$	$C \to \phi$; $C \to C$ (14)
|2| X has 2 attr. i.e.	$AB^+ = ABC$	$AB \to \phi$ - - - - - $AB \to ABC$
	$BC^+ = BC$	$BC \to \phi$ - - - - - - $BC \to ABC$
	$CA^+ = ABC$	$CA \to \phi$ - - - - $AC \to ABC$ (20)
|3| X has 3 attr. i.e.	$ABC^+ = ABC$	$ABC \to \phi$ - - - $ABC \to ABC$ (8)

43 FDs

So there are 43 FDs in this F^+.

↓ this side can become the determinant of FD where determiner will be any subset of its closure.

Equality of FD Sets:-

Let F and G be two FD sets.

then | F & G FD sets are equal iff $F^+ \equiv G^+$ |

But it will become very hard to calculate all members of F^+ and G^+, so our problem is not solved.

Another Method-

F & G FD sets are equal iff

a) F covers G:-

Every FD of G set must be implied in F.

$[F \supseteq G]$

b) G covers F:-

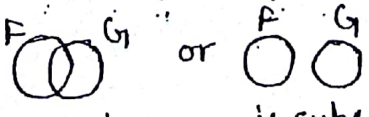
Every FD of F set must be implied in G.

$[F \subseteq G]$

(4)

F covers G	G Covers F	
✓	✓	$F \equiv G$
✓	×	$F \supset G$
×	✓	$F \subset G$
×	×	F & G Not Comparable

F & G Not Comparable means



but no one is subset of the other

Q1. $F = \{A \to B, B \to C, C \to A\}$

$G = \{A \to BC, B \to AC, AB \to C, BC \to A, AC \to B\}$

Which is true a) $F \subset G$ b) $F \supset G$ c) $F = G$ d) NOT

Sol:-

In G,

$A \to BC \Rightarrow A \to B$

$B \to AC \Rightarrow B \to C$

But $C \to A$ can't come in G.

G not covers F

In F,

$A \to B, B \to C \Rightarrow A \to C$

$A \to B, A \to C \Rightarrow \boxed{A \to BC}$

$B \to C, C \to A \Rightarrow B \to A$

$B \to A, B \to C \Rightarrow \boxed{B \to AC}$

$C \to A, A \to B \Rightarrow C \to B$

$A \to B \Rightarrow AB \to B$ (Augmentation)

$AB \to B, B \to C \Rightarrow \boxed{AB \to C}$

$B \to C \Rightarrow BC \to C$

$BC \to C, C \to A \Rightarrow \boxed{BC \to A}$

$C \to A \Rightarrow AC \to A$

$AC \to A, A \to B \Rightarrow \boxed{AC \to B}$

So, F covers G.

or $F \supset G$

Also, for $AB \to C$

$AB^+ = (A, B, C)$

we find AB^+ in F not in G,

$\therefore AB^+$ has C

So, $AB \to C$ can be implied in F.

So, $AB \to C$ is covered by F

Similarly all others are covered by F

Q2. $F = \{A \to BCDEF, BC \to ADEF, B \to F, D \to E\}$

$G = \{A \to BC, BC \to AD, B \to F, D \to E\}$

then which is true a) $F \subset G$ b) $F \supset G$ c) $F \equiv G$ d) None

In G,

$A^+ = \{A\ B\ C\ D\ E\ F\}$ so, $A \to B\ C\ D\ E\ F$ is covered

$BC^+ = \{BC\ A\ D\ E\ F\}$ so $BC \to A\ D\ E\ F$ is covered

$B \to F$ & $D \to E$ is covered.

So, G covers F $F \subset G$

In F, $A \to BC$ is covered

$BC \to AD$ is covered

$B \to F$, $D \to E$ are also covered

So, F covers G.

$F \supset G$

$\because F \subset G$ & $F \supset G$ So, $F \equiv G$.

(c) is correct.

Canonical cover [Minimal cover]

Minimal set of FDs (Fm) which are logically equal to given FD set F. $\boxed{Fm \equiv F}$

eg $F = \{A \to B, B \to C, A \to C, AB \to C, A \to A\}$

Find minimal cover of F.

Sol:

i) $A \to A$ (Trivial FD)

It should not be included in F.

as if we not have $A \to A$ then also it can't influence FD set F.

or $A \to A$ is obviously derivable from F.

So, remove $A \to A$

ii) $AB \to C$

$B \to C \xrightarrow{\text{Augm.}} AB \to AC \xrightarrow{\text{split}} AB \to C$

$\because AB \to C$ can be determined by $B \to C$

so, remove $AB \to C$

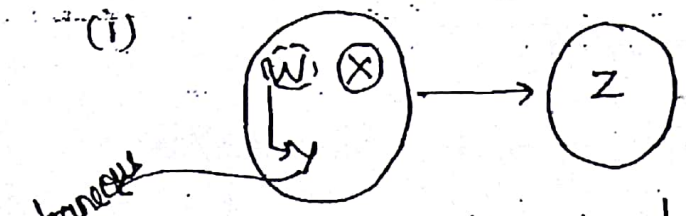
iii) $A \to C$

$A \to B, B \to C \xrightarrow{\text{trans.}} A \to C$

$\therefore Fm = \{A \to B, A \to C\}$

(4) Extraneous

(i)



if here $W \to X$, then X is extraneous.

If $f_1 = \{ WXY \to Z, W \to$

then Y is Extraneous

means

in $W \to Y$

both determinant & dete

comes in determinant of

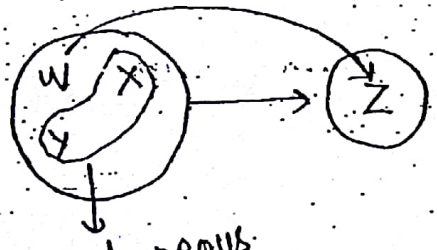
$\underline{\underline{WXY}} \to Z$

then Y becomes extra

so, $f_2 = \{ W.X \to Z, W \to Y$

Then, $\boxed{F_1 \equiv F_2}$

(ii)



If $f_1 = \{ WXY \to Z, W \to Z$

then X and Y are extrane

$f_2 = \{ W \to Z \}$

then $\boxed{F_1 \equiv F_2}$

Q. Find extraneous attribute for

(i) $F = \{ ABCD \to E, \; AC \to F, \; F \to B \}$

$AC \to F, \; F \to B \Rightarrow \underline{AC \to B}$

both are in

$ABCD \to E$

So, B is extraneous.

$\boxed{\{ ACD \to E, \; AC \to F, \; F \to B \}}$

Resultant set after removing extraneous attribute.

(ii) $F = \{ ABCD \to E, \; AC \to F, \; F \to E \}$

$AC \to F, \; F \to E \Rightarrow AC \to E$

$ABCD \to E$

BD are extraneous.

Remember: So, $\{ AC \to E, \; AC \to F, \; F \to E \}$ is resultant set after removing extraneous attribute B

is not minimal

(4) Redundant FD in FDset:-

$X \to Y$ FD of FD set F is redundant iff

$\{F - (X \to Y)\} \models (X \to Y)$

i.e even after removing $X \to Y$ from F, still $X \to Y$ is a member of F.

in the above example,

we got resultant FD $F_1 = \{AC \to E, AC \to F, F \to E\}$

So, $AC \to F, F \to E \Rightarrow AC \to E$

So, $AC \to E$ is redundant.

∴ we remove it and get

$F_2 = \{AC \to F, F \to E\}$

eg.

$F = \{AB \to C, BC \to D, AB \to D, CD \to E, BC \to E\}$

Find redundant FDs.

sol:- When can we say $AB \to C$ is redundant?

(i) when we remove $AB \to C$, still $AB \to C$ should be derived.

For this we test $(AB)^+ = \{A B D\}$.

But C is not the member

So, $AB \to C$ is not redundant.

(ii) $BC \to D$ is also not redundant as

$(BC)^+ = \{B, C, E\}$ and D is not a member.

(iii) $AB \to D$ is redundant as

$(AB)^+ = \{A B C E D\}$

(iv) $CD \to E$ is not redundant as

$(CD)^+ = \{CD\}$ and E is not a member.

(v) $BC \to E$ is redundant as

$(BC)^+ = \{B C E D\}$ and E is member

$\therefore F_1 = \{AB \to C, BC \to D, CD \to E\}$ Resultant set after removing redundant sets.

$F \equiv F_1$

Q $F = \{AB \to CD, A \to C\}$

Is B an extraneous attribute?

Sol:- Here we see that

$AB \to CD$ $\quad A \to C$ (B is extraneous)

But we can't remove B directly from here,

First we have to split $AB \to CD$ as

$AB \to C$

$AB \to D$

Now, B is extraneous only for $AB \to C$ $\quad A \to C$

but not for $AB \to D$ $\quad A \to C$

So, Resultant set after removing extraneous attr,

$F_1 = \{AB \to D, A \to C\}$

$F_1 \equiv F$

PROCEDURE To find Minimal cover of f :-

1) Split FD's of f such that every determiner have single attr
2) Remove trivial FDS from step(1) result.
3) Remove extraneous Attributes from every determinant of step(2) result.
4) Remove redundant FDs from step(3) result.
5) Merge FD's of step(4) if any determinants are equal.

Result of step(5) is minimal cover of FD set F.

Q find minimal cover of RD set.

$F = \{A \to BCDEF,\ BC \to ADEF,\ B \to F,\ D \to E\}$

Sol- Applying Steps :-

(I) $A \to BCDEF$ —apply split→ $A \to B$, $A \to C$, $A \to D$, $A \to E$, $A \to F$

$BC \to ADEF$ —apply split→ $BC \to A$, $BC \to D$, $BC \to E$, $BC \to F$

$B \to F$

$D \to E$

(II) ×

(III) $BC \to F$ $B \to F$

C is extraneous. Now $B \to F$ is left so we can remove $BC \to F$ whole

$F_1 = \{A \to B, A \to C, A \to D, A \to E, A \to F,$
$BC \to A, BC \to D, BC \to E, B \to F, D \to E\}$

(IV) $A \to B$ will be redundant if after removing $A \to B$ A^+ contains B but this does not happen.

So, $A \to B$ is not redundant.

Similarily $A \to C$ is not redundant.

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

So, $A \to D$ is redundant

Now we will not take $A \to D$ in testing process.

Similarily $A \to E$, $A \to F$ are redundant

Now,

$BC \to A$ is not redundant as $BC^+ = \{B\ C\ D\ E\ F\}$ but A is not there.

$BC \to D$ is not redundant.

$BC \to E$ is redundant

$B \to F$, $D \to E$ not redundant as $B^+ = B$, $D^+ = E$

So, Resulting FD set after removing redundant FDs.

$F_2 = \{ A \to B \quad A \to C \quad BC \to A \quad BC \to D \quad B \to F \quad D \to E \}$

(V) Join

$F_3 = \{ A \to BC \quad BC \to AD \quad B \to F \quad D \to E \}$

This is minimal cover of FD set F.

$F_m = \{ A \to BC \quad BC \to AD \quad B \to F \quad D \to E \}$

* we can get more than one minimal cover for FD set F.

foreg ① if we start checking FDs from last then

$F_1 = \{ A \to B, \ BC \to A \quad B \to F \quad D \to E \}$

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

$F_m = \{ A \to BCD, \ BC \to A, \ B \to F, \ D \to E \}$

for ex ② $F = \{ AB \to C, \ A \to B, \ B \to A \}$

if we consider B is extraneous in $AB \to C$ then

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

if A is extraneous then

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

or

NOTE:-

Minimal cover of FD set f may not be unique but all minimal covers are logically equal or equal to FD.

i.e.

$(fm_1 \equiv fm_2 \equiv f)$

or Expressive Power of fm_1 = that of fm_2 = that of f.

Q.

$F = \{ A \to BC,\ B \to AC,\ C \to AB \}$

Find all possible minimal covers of FD set f.

Sol:-

(I) $A \to B$ $\quad B \to A$ $\quad C \to A$

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

(II) ×

(III) There can't be any extraneous attribute as only one attribute is there in determinant of every FDs.

(IV)

$A \to B$ × $B \to A$ × $C \to A$ ✓

$A \to C$ ✓ $B \to C$ ✓ $C \to B$ ✓

$Fm_1 = \{ A \to C \quad B \to C \quad C \to AB \}$

The decomposition of any relation (table) is successful only if the decomposition follows is

① - Lossless Join decomposition

② Dependency Preserving decomposition

① Lossless Join Decomposition

we are using here an query operator : Join (⋈).

Π (Projection)

$\Pi(R)$ ⇒ Eliminates attributes of R those are not in projection list.

↑ Attributes

R

A	B	C
4	2	4
3	2	4
4	2	3

$\Pi_{BC}(R)$ →

B	C
2	4
2	3

Join (⋈)

eg1

R

A	B	C
2	4	2
3	4	2
4	3	8
4	4	6

=

S

C	D	E
2	5	6
8	4	2
7	2	3

R ⋈ S ⇒

A	B	C	D	E
2	4	2	5	6
3	4	2	5	6
4	3	8	4	2

eg2

R

A	B	C
2	4	2
3	4	2
4	4	5

=

S

B	C	D
4	2	6
4	3	5

R ⋈ S ⇒

A	B	C
2	4	2
3	4	2

eg 3

R

A	B
2	4
3	4

S

C	D
4	5
6	5

$R \bowtie S \Rightarrow$

A	B	C	D
2	4	4	5
3	4	4	5
2	4	6	5
3	4	6	5

(i) Here no elimination will be there because no attributes are in common to both relations (tables). [Case: Max. elements after join]

(ii) If one of the relation of join is empty then the resulting join is also empty.

Lossless Join Decomposition

Relation R decomposed into $R_1\ R_2 \dots R_n$

In General, $(R_1 \bowtie R_2 \bowtie R_3 \bowtie \dots R_n) \supseteq R$

If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) = R$
Then Lossless Join Decomposition

If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) \supset R$
Then lossy join decomposition

Ex:- R(Sid Sname Cid)

S_1	A	C1
S_1	A	C2
S_2	B	C2
S_3	B	C3

{Sid → Sname}

Sid Cid: Cand key

Now, let R is decomposed into R_1(Sid Sname), R_2(Sid Cid)

R_1 (Sid Sname)

S_1	A
S_2	B
S_3	B

R_2 (Sid Cid)

S_1	C_1
S_1	C_2
S_2	C_2
S_3	C_3

To test whether the decomposition is lossless or lossy, we will join R_1 and R_2 based on common attribute Sid.

$R_1 \bowtie R_2$ =

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_3	B	C_3

we see that $(R_1 \bowtie R_2) \equiv R$

So, we can say that this decomposition of R into R_1 & R_2 is lossless join decompos

ii) R_1 (Sid Sname) = R_2 (Sname Cid)

S_1	A
S_2	B
S_3	B

A	C_1
A	C_2
B	C_2
B	C_3

$R_1 \bowtie R_2$ =

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_2
S_2	B	C_3
S_3	B	C_2
S_3	B	C_3

SPURIOUS TUPLES

(4 records table are now 6 records table)

causes

inconsistency

↳ becoz result of sql queries will now be wrong.

Here we see that $R_1 \bowtie R_2 \supset R$

So, we say that this decomposition of R into R_1 & R_2 is lossy join decomposition

(*) This decomposition is done during creation of tables and at this time we do not have any record but only FDs.

So, How to decide whether decomposition is lossy or lossless?

Minimum Criteria for Lossless Join Decomposition:

Decomposition of R into R_1 & R_2 is lossless join decomposition

iff a] $R_1 \cup R_2 \equiv R$ (attr of R_1 ∪ attr of R_2 ≡ attr of R)

[no attribute of R should lost becoz of decomposition]

and b] $(R_1 \cap R_2) \longrightarrow R_1$ {$(R_1 \cap R_2)$ is superkey of R_1}

(OR)

$(R_1 \cap R_2) \longrightarrow R_2$ {$(R_1 \cap R_2)$ is superkey of R_2}

Ultimate Goal of Lossless Join Decomposition is to achieve $\boxed{R_1 \bowtie R_2 \equiv R}$

→ if a] is false then no. of attributes of $R_1 \bowtie R_2$ will not be equal to no. of attributes of R.

→ if b] is false then no. of records of R will be the subset of no. of records of $R_1 \bowtie R_2$.

(*) The common attribute of join must be the key (superkey / candidate key) of atleast one relation involved in join.

eg: R(A B C D E)

$F = \{AB \to C,\ C \to D,\ B \to E\}$

1] decomposed into {ABC, CD}

⇒ Lossy Join decomposition as Attribute 'E' is lost as a result of decomposition.

$R_1 \bowtie R_2$			
A	B	C	D

$\neq$

R			
A	B	C	DE

2] decomposed into {ABC, DE}

⇒ Lossy Join decomposition as b] cond^{n} failed.

$R_1 \cap R_2 = \phi$ (gurantees lossy decomposition)

eg:

R

A	B	C	D	E
2	4	4	5	6
3	5	4	5	7

R_1

A	B	C
2	4	4
3	5	4

R_2

D	E
5	6
5	7

Here

$R_1 \bowtie R_2 \supset R$

So lossy decomposition.

3] decomposed into {ABC, CDE},

⇒ a) $R_1 \cup R_2 = R$ (true)

b) $R_1 \cap R_2 \to R_1$ (X) $\quad R_1 \cap R_2 = C$

or

$R_1 \cap R_2 \to R_2$ (X) but $C^+ = CD$

means C determines CD

but C not determines ABC

or E

∴ both are not followed

So, we can say

$(R_1 \cap R_2)$ not superkey for R_1 & R_2

∴ Lossy Join Decomposition.

4]. decomposed into {ABCD, BE}

a) $R_1 \cup R_2 = R$ ✓

b) $R_1 \cap R_2 = \{B\}$

$B^+ = \{BE\}$

So, $B \rightarrow BE$

or $(R_1 \cap R_2) \rightarrow R_2$ ✓ → means B is sk for R_2

∴ Lossless Join Decomposition

5) decomposed into {ABC, ABDE}

a) $R_1 \cup R_2 = R$ ✓

b) $R_1 \cap R_2 = \{AB\}$

$AB^+ = \{A, B, C, D, E\}$ ← AB is superkey of both relations

$AB \rightarrow C$ & $C \rightarrow D$

$AB \rightarrow D$

$R_1 \cap R_2 \rightarrow R_1$ ✓

So, Lossless Join decomposition

6) decomposed into {ABC, CD, DE}

a) $R_1 \cup R_2 \cup R_3 = R$ ✓

b) $R_1 \cap R_2 \cap R_3 = \phi$ (X)

∴ Lossy Join decomposition.

~~7) decomposed~~ Here we test for every 2 relation pair.

[R_i & R_j can join only if $R_i \cap R_j \rightarrow R_i$ or $R_i \cap R_j \rightarrow R_j$]

$R_1(ABC)$ $R_2(CD)$ $R_3(DE)$

for R_1 & R_2 $R_1 \cap R_2 = C$ $C^+ = \{CD\}$

So, $R_1 \bowtie R_2 \rightarrow R_2$ ∴ lossless join

$R_1(ABC)$ $R_2(CD)$ $R_3(DE)$

$R_1 \bowtie R_2$

$R_{12}(ABCD)$

can't join lossless

as $R_{12} \cap R_3 = D$

and $D^+ = D$

so, $R_{12} \cap R_3 \rightarrow R_2$ ✗

$R_{12} \cap R_3 \rightarrow R_3$ ✗

so, lossy decomposition.

7] decomposed into {ABC, CD, BE}

$R_1(ABC)$ $R_2(CD)$ $R_3(BE)$

$R_1 \cap R_2 = C$

$C^+ = \{CD\} = R_2$

so, C is SK for R_2

$R_1 \bowtie R_2$

$R_{12}(ABCD)$

$R_{123}(ABCDE)$

$R_{12} \cap R_3 = B$

$B^+ = \{BE\} = R_3$

so, B is SK for R_3

∴ we can join.

Hence lossless join decomposition.

Q. $R(ABCDEF)$

$\{AB \rightarrow C,\ D \rightarrow E,\ CD \rightarrow F\}$

decomposed into $\{ABC, DE, CDF, ABD\}$

$R_1(ABC)$ $R_2(DE)$ $R_3(CDF)$ $R_4(ABD)$

R_1 can't be joined with R_2 as $R_1 \cap R_2 = \phi$

$R_1 \cap R_3 = C$ but $C^+ = C$ so, join not allowed

$R_1 \cap R_4 = \{AB\}$ $AB^+ = \{ABC\}$

so, $R_1 \cap R_4 \rightarrow R_1$ so join is possible

∴ $R_1 \bowtie R_4 \Rightarrow R_{14}(ABCD)$ with AB as SK for R_1.

Now, $R_{14}(ABCD)$ $R_2(DE)$ $R_3(CDF)$

$R_{14} \cap R_2 = D$ $D^+ = \{DE\} = R_2$

So, $R_{14} \bowtie R_2$ is possible $R_{14} \bowtie R_2 = R_{142}(ABCDE)$

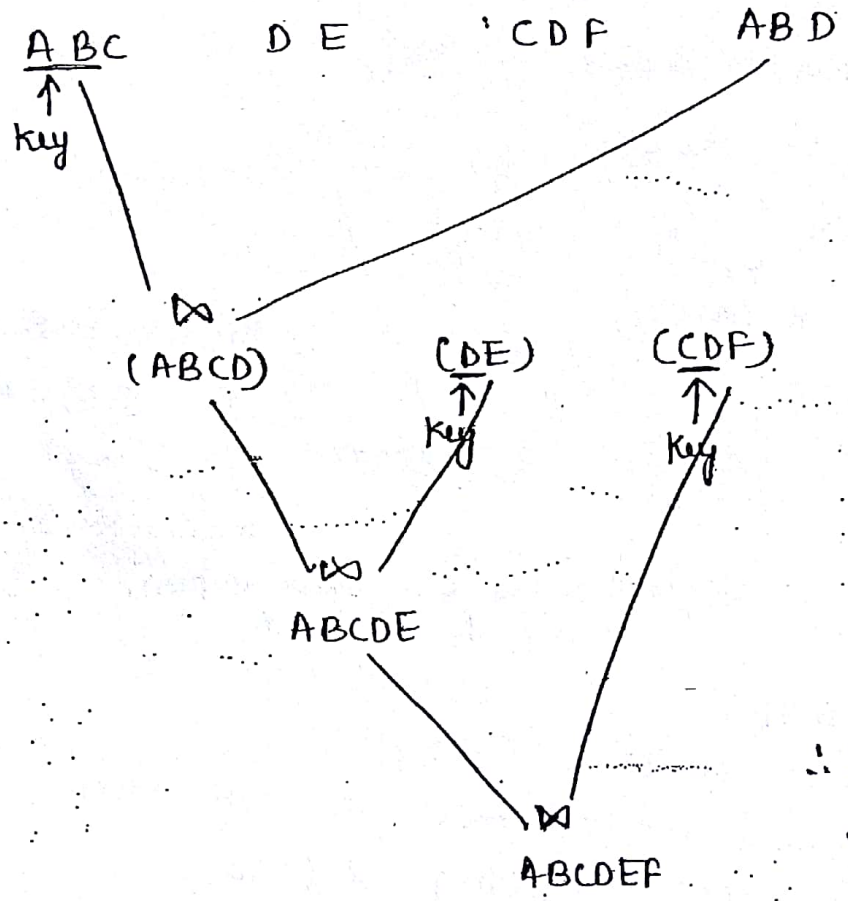
Now,

$R_{1\,4\,2}$ (A B C D E) R_3 (C D F)

$R_{1\,4\,2} \cap R_3 = \{CD\}$ $\quad CD^+ = \{CDF\} = R_3$

$\because R_{1\,4\,2} \cap R_3 \longrightarrow R_3$ so join possible,

$R_{1\,4\,2} \bowtie R_3 = R_{1\,4\,2\,3}$ (ABCDEF)



copyright©theorypoint.com The Best Gateway Of Success

∴ Lossless Join Decomposition.

(2) DEPENDENCY PRESERVING ~~TRANSFORMATION~~ DECOMPOSITION :-

Relational Schema (R) with FDset F decomposed into $R_1, R_2 \ldots R_n$ subRelations with FDsets $F_1, F_2 \ldots F_n$ respectively.

In General

$$[F_1 \cup F_2 \cup \ldots \cup F_n] \subseteq F$$

If $[F_1 \cup F_2 \cup \ldots \cup F_n] \equiv F$
then decomposition is Dependency Preserving Decomposition.

} No FD is lost becoz of Decomposition

If $[F_1 \cup F_2 \cup \ldots \cup F_n] \subset F$
then it is not Dependency Preserving Decomposition.

Basic Meaning of Not Dependency Preserving Decomposition - is atleast one FD in FDset F is lost because of decomposition.

eg R (A B C D E)
$F = \{A \to B, B \to C, C \to D, D \to BE\}$
is decomposed into $\{AB, BC, CD, DE\}$
Test whether dependency preservation is satisfied or not.

Sol:- Find FDs of subrelations ⇒

$R_1(AB)$	$R_2(BC)$	$R_3(CD)$	$R_4(DE)$
(F1)	(F2)	(F3)	(F4)
Find Non Trivial FDs of SubRelation	$B^+ = (BCDE)$	$C^+ = (CDEB)$	$D^+ = (DBEC)$
	$B \to C$	$C \to D$	$D \to E$
$A^+ = (A,B,C,D,E)$	$C^+ = (CDEB)$	$D^+ = (DBEC)$	$E^+ = E$
So, $A \to B$	$C \to B$	$D \to C$	$E \to D$ ✗
but in $B^+ = (B,C,D,E)$ A is not there So $B \to A$ ✗			

Find required, unwanted and Nontrivial FDs.

So,

$F_1 = \{A \to B\}$

$F_2 = \{B \to C \quad C \to B\}$

$F_3 = \{C \to D \quad D \to C\}$

$F_4 = \{D \to E\}$

} These all FDs will surely be present in the FDset of F becoz these all are defined by closure over F only.

But we need to check whether all FD in F, is implied in $[F_1 \cup F_2 \cup F_3 \cup F_4]$.

So, $[F_1 \cup f_2 \cup f_3 \cup f_4] = \{A \to B, B \to C, C \to B, C \to D, D \to C, D \to E\}$

Now $F = A \to B$ ✓

$B \to C$ ✓

$C \to D$ ✓

$D \to BE$ — $D \to B$ ($D \to C, C \to B \Rightarrow D \to B$) ✓

$D \to E$ ✓

So, all FDs of F are covered by $F_1 \cup F_2 \cup F_3 \cup F_4$

$\therefore [F_1 \cup f_2 \cup f_3 \cup f_4] \equiv F$

Hence it is dependency preserving decompⁿ.

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

decomposed into {ABC, BCD, AD}

$R_1(ABC)$

(f1) $\boxed{AB \to C}$

but $AB \to D$ ✗ as D is not in R_1

$R_2(BCD)$

(f2)

- $D^+ = \{D, A\}$ but A is not in R_2
- $BD^+ = \{B D A C\}$

$\boxed{BD \to C}$

$R_3(AD)$

(f3) $\boxed{D \to A}$

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

$[F_1 \cup F_2 \cup F_3] = \{AB \to C,\ BD \to C,\ D \to A\}$

$F = AB \to CD$ ← $AB \to C$ ✓

$AB \to D$ ✗ ($AB^+ = (A\ B\ C)$)

as D is not in AB^+

so, $AB \to D$ is not covered.

$D \to A$ ✓

∵ $AB \to D$ is lost because of decomposition

∴ not preserving dependency decomp^n.

Q. $R(A\ B\ C\ D\ E\ F)$

$F = \{A \to BCDEF,\ BC \to ADEF,\ B \to F,\ D \to E\}$

decomposed into $\{ABC, BCD, BF, DE\}$

Test dependency preservation.

Sol:-

$R_1(ABC)$	$R_2(BCD)$	$R_3(BF)$	$R_4(DE)$
$A \to B$, $A \to C$ ⟹ $A \to BC$	$BC \to D$	$B \to F$	$D \to E$
$AB \to C$			
$BC \to A$			
$AC \to B$			
(F_1)	(F_2)	(F_3)	(F_4)

$[F_1 \cup F_2 \cup F_3 \cup F_4] = \{A \to B,\ A \to C,\ AB \to C,\ BC \to A,\ AC \to B,\ BC \to D,\ B \to F,\ D \to E\}$

$F = A \to BCDEF$ ✓

$\{A \to BC$ ✓
$BC \to D$ $\quad A \to D$
$B \to F$ $\quad A \to F$
$D \to E$ $\quad A \to E\}$ Apply join

All FDs of f set are covered. So, dependency preserving decomp^n

$BC \to ADEF$ ✓

$B \to F$ ✓

$D \to E$ ✓

$BC^+ = \{ADEF\}$

$BC \to D$ $\quad D \to E$

$B \to F$ $\quad BC \to A$

Q. R(ABCDE)

{A→BC, CD→E, B→D, E→A}

decomposed into {ABCE, BD}

Test dependency preservation.

Sol:-

R_1 (ABCE)	R_2 (BD)
$A^+ = (A\ B\ C\ D\ E)$	$B^+ = \{B, D\}$
Since A determines all BCE in R_1	So, $\boxed{B \to D}$ as D is in R_2.
So, A will be cand. key of R_1.	$D^+ = D$ only.
[A may/may not be cand key of R_1]	
Once we know that A is cand. key of R_1 then we need not to compute closure of any subset of (ABCE) which contains A because it will result in redundant dependency.	

eg: $\boxed{A \to BCE}$

Now,

$AB \to CE$

in this FD, B is extraneous,

So, it becomes $A \to CE$

which is already in $A \to BCE$.

Now,

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

but D is not in R_1

so no dependency will result.

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

So, E is cand. key of R_1, $\boxed{E \to ABC}$

Now, $BC^+ = \{B\ C\ D\ E\ A\}$ $\boxed{BC \to AE}$

So, BC is cand key of R_1

Now,

$F_1 = \{ A \to BCE \quad E \to ABC \quad BC \to AE \}$

$F_2 = \{ B \to D \}$

$F_1 \cup F_2 = \{ A \to BCE \quad E \to ABC \quad BC \to AE, \; B \to D \}$

F: $A \to BC$ ✓

$CD \to E$ ✗

$B \to D$ ✓

$E \to A$ ✓

Not dependency preservation decomposition.

Model Question for Properties of Decomposition

R(ABCD)

$\{ A \to B, \; C \to D \}$

decomposed into $\{AB, CD\}$

Then R decomposition is

i) LLJ & DP ii) LLJ & Not DP

iii) Lossy & DP iv) Lossy & Not DP

Sol:-

$R_1(AB)$	$R_2(CD)$
$A^+ = AB$	$C^+ = CD$
A is candidate key	C is candidate key
$A \to B$ (F_1)	$C \to D$ (F_2)

$[F_1 \cup F_2] \equiv F$

so dependency preserving decomp.

$R_1(AB)$ $R_2(CD)$
a) $R_1 \cup R_2 \equiv R$ ✓
b) $R_1 \cap R_2 = \phi$
so, not lossless join decomposition.
Option iii) is correct.

Q. R(ABCD) $F = \{AB \to CD, D \to A\}$

decomposed into {ABC, BCD, AD}

$R_1(ABC)$	$R_2(BCD)$	$R_3(AD)$
$AB^+ = ABCD$		$D^+ = (DA)$
$AB \to C$		$D \to A$
(F_1)	(F_2)	(F_3)

$(F_1 \cup F_2 \cup F_3) \neq F$

So not dependency preserving.

$R_1(ABC)$ $R_2(BCD)$

a) $R_1 \cup R_2 = R$ ✓

b) $R_1 \cap R_2 = BC$

$BC^+ = \{BC\}$

So lossy.

$R_1(ABC)$ $R_3(AD)$

$R_1 \cap R_3 = A$

$A^+ = A$

$R_2(BCD)$ $R_3(AD)$

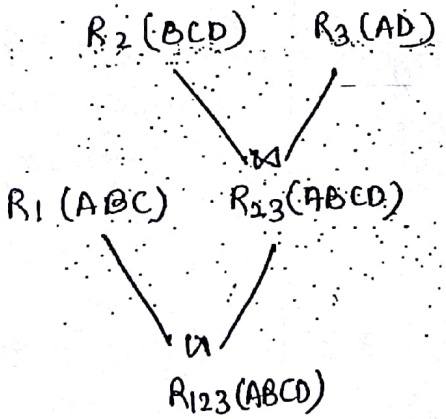
$R_2 \cap R_3 = D$

$D^+ = \{DA\}$

So,

$R_2 \cap R_3 \to R_3$

∴ we can join R_2 & R_3.



So, lossless join decomposition.

$R_1 \cap R_{23} = ABC$

$ABC^+ = (ABCD)$

So,

$R_1 \cap R_{23} \to R_1$

$R_1 \cap R_{23} \to R_{23}$

So we can join

R(sid, sname cid) ... {sid → sname}

Sid Cid : Cand key

d	sname	Cid
	A	C1
	A	C2
	B	C2
	B	C3
	B	C2

INF

⇒ RDBMS table

If we take Sid Cid as Cand. Key then the multivalued attribut becomes Single valued attribu

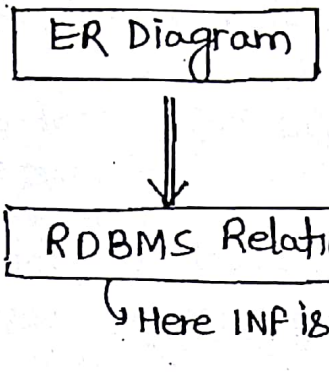
So, when multivalued attribute is taken i Cand key then all attributes become sir valued.

we assume a relation where a student can multiple enrolled courses, multiple Ph.No. multiple

then only cand key is Sid and f = {sid ⇒ sname} will hold only.

sid	sname	Cid	PhNo	Email

Steps to design DB :-



Multivalued attribute are allowed to be represented.

Multivalued attributes are not Allowed.

(*)

2NF 3NF BCNF | 4NF

reduce redundancy over FDs | over MVDs

FD [Single Valued Dependency]

forms Redundancy iff

Non Trivial FD (X → Y) with X not a superkey.

eg R (sid Sname cid)

S1	A	C1
S1	A	C2
S1	A	C3
S2	B	C1

redundancy

{sid → Sname}

sid cid : Cand Key

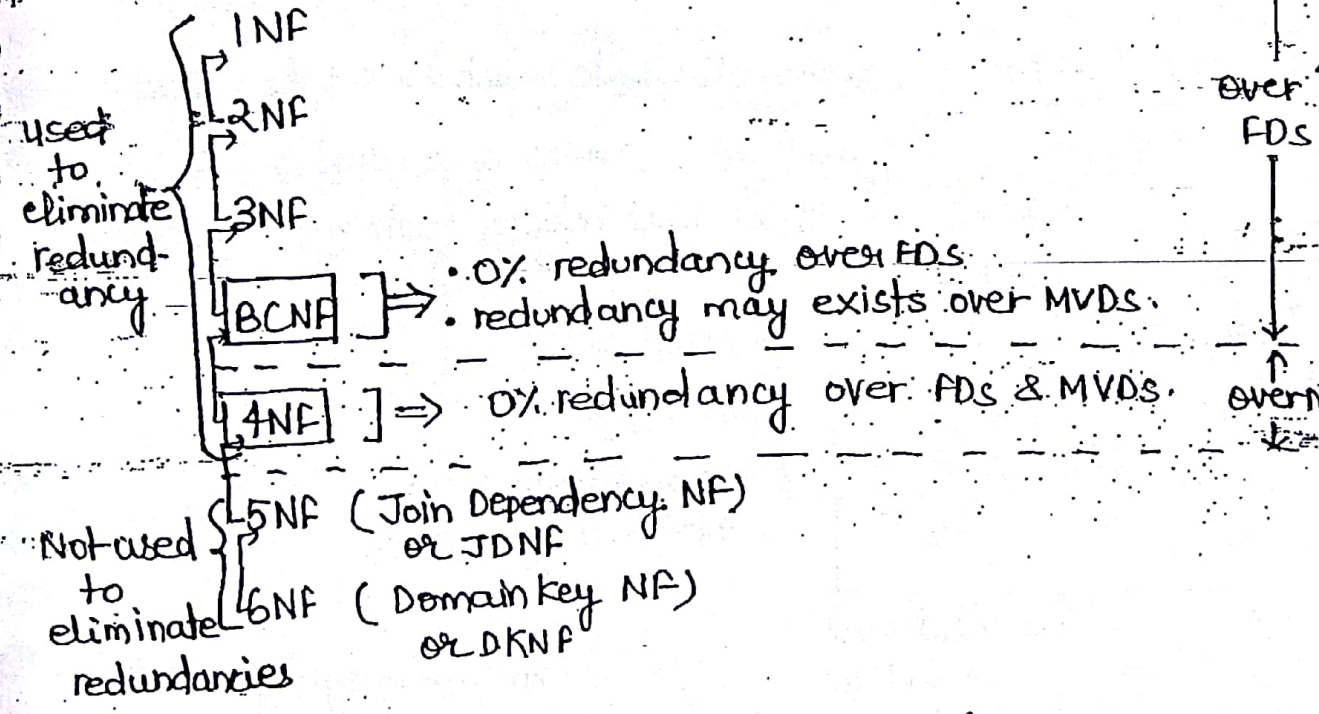
Here sid → Sname is non-trivial & sid is not super key

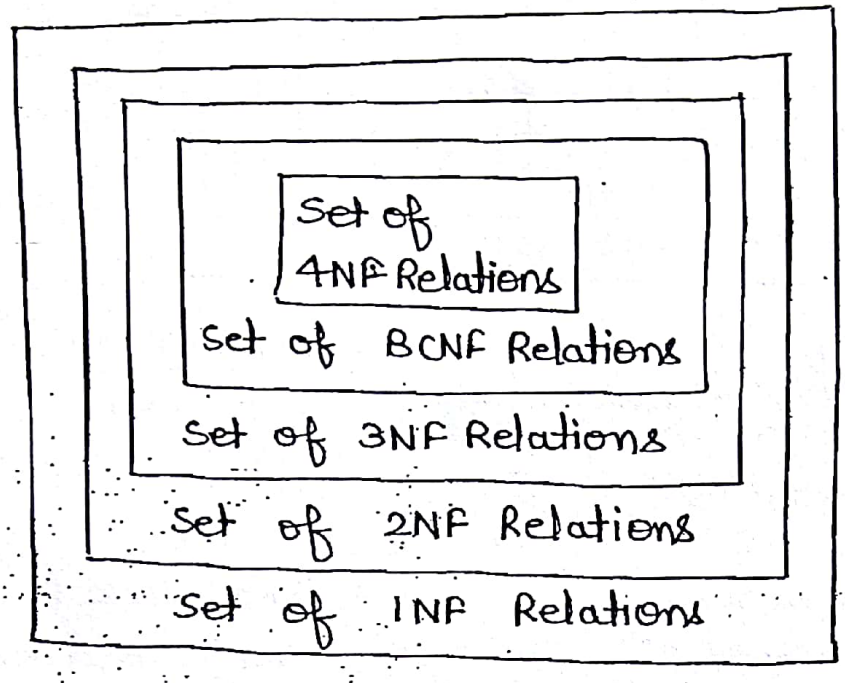
∴ it will form redundancy in the table.

Normal Forms are used to identify degree of redundancy in DB relations.

Redundancy in Relations becoz of

- Non trivial FD (SingleValued Dependency)
  $X \to Y$
  with X not superkey
  ⇓
  To eliminate redundancy over FDs, decompose relation into BCNF.
- Non Trivial MFD (Multiv Deper
  $X \to\to Y$
  with X not superkey
  ⇓
  To eliminate redund over MVDs, decompos relation into 4NF.





FIRST NORMAL FORM (1NF) :-

Relation R is in 1NF iff No Multivalued attributes are in R.

[OR] Every attribute of R must be atomic (single valued).

RDBMS condition :- Every attribute must be single valued.

[1NF is default NF, means if a relation is not in 1NF then that relation will not be in RDBMS]

Cand. key

ex!:

Sid	Sname	Cid
S_1	A	C_1/C_2
S_2	B	C_2/C_3
S_3	B	C_2

Multivalued Attribute

[Set of values over Attribute for some record]

This relation is not in 1NF

↓

Not RDBMS table.

We can only say -

{sid → Sname}

FD $[X \to Y]$ is free from Redundancy.

iff a) $X \supseteq Y$ [Trivial FD]

(or)

b) X : super key

Ex

R (Sid Sname Age)

S_1	A	20
S_2	A	20
S_3	B	21

{(Sid) → Sname age}
Superkey

Not forms Redundancy

But consider the example,

R

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_3	b_2	c_2

{(A) → BC}
Superkey

No redundancy in R

a_4 b_1 c_2 ← can come, so we can't say b_1 c_1 / b_1 c_1 redu because these attributes are independent & th is no relati blw th

R

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_3	b_2	c_2

{(A) → B, (B) → C}
A: SK, B: NotSK

Redundancy

because here when c_2 come in c_1, then always there b_2 will come in B.

X is not superkey of R iff

a) X is proper subset of cand. key

b) X is Non Prime Attribute

c) X is combination of proper subset of cand. key & Non Prime Attribute.

Q.

R(ABCDEF)

{AB → C, B → E, C → A, BE → F, B → D}

Identify FDs which introduce redundancy?

Here Cand. keys : {AB, BC}

↳ found by checking if A or B comes in RHS of any FD. then replacing it by determinant part.

(AB) → C : not redundant data
SK

(B) → E : redundant data as B is proper subset of cand key
not SK

(C) → A : redundant data (same reason)
not SK

BE → F

Here B is extraneous, (E) → F : redundant data as E is not prime
not SK

(B) → D : redundant data B is proper subset of cand key.
not SK

$X \to Y$ nontrivial FD with X not superkey can be:

a] X: Proper Subset of cand key → Y: Non Prime Attr.

b] X: Non Prime Attr. → Y: Non Prime Attr.

c] X: Proper Subset of Cand. key → Y: Proper Subset of other cand Key.

(All forms redundancy).

Ⓐ Non Prime (X) → Proper Subset of CK (Y) } (X) Not Possible

because if determiner (Y) is Proper Subset of CK then determinant (X) will become a part of some other cand key. and thus can't be Non-Prime.

To remove all these redundancy, we use Normal Form.

X → Y, X: not SK	1NF	2NF	3NF	BCNF
a]	Allowed	Not Allowed	Not Allowed	Not Allowed
b]	Allowed	Allowed	Not Allowed	Not Allowed
c]	Allowed	Allowed	Allowed	Not Allowed

- In 1NF all a] b] c] are allowed.
- In 2NF b] & c] are allowed, so redundancy come due to only these types of FDs.
- In 3NF only c] is allowed.
- In BCNF, none of them are allowed. So, there will be no redundancy if the relation will be in BCNF.

Possible Ques:-

i) If R is in 3NF but not in BCNF then which type of Dependency can cause ~~dependency~~ redundancy?

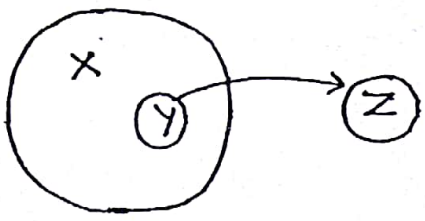
ii) if R is in 2NF then which kind of dependency can cause redundancy?

iii) if R in 3NF (and) NO NonTrivial FD such that Proper Subset of CK → Proper Subset of other CK

Then R also in BCNF.

Relation Schema R is in 2NF iff NO Partial Dependency in R.

PARTIAL DEPENDENCY



X: Any Cand key
Y: Proper subset of Cand key
Z: Non Prime Attr.

$Y \rightarrow Z$ is a partial dependency

THIRD NORMAL FORM (3NF)

Relational Schema R is in 3NF iff Every Nontrivial FD $X \rightarrow Y$ in R with a) X must be super key
b) Y must be prime Attribute

In 3NF,
following dependencies are not allowed:

i) Proper subset of Cand key → Non Prime Attr.

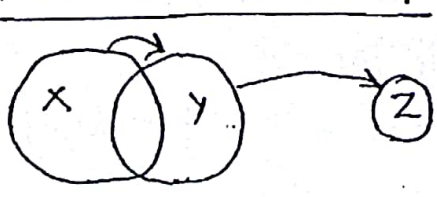
ii) ~~Non~~ Prime Attr. → ~~Non~~ Prime Attr.

but the dependency, Proper subset of Cand key → Proper subset of another cand. key

is allowed which causes redundancy.

3NF def:- Relation Schema R is in 3NF iff No transitive Depen in relation R.

TRANSITIVE DEPENDENCY



X: any cand key
Y: Proper subset of Cand key (Or) Non Prime (or) Proper subset of cand key & Non Prir } Not SK.

$\{X \to Y, (Y \to Z)\}$ — Z: Non Prime Attribute

TD

Non Prime Attribute (Z) is transitively determined by superkey (X) is called transitive dependency

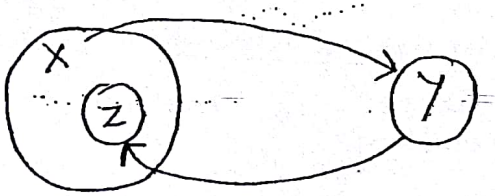
(*) $X \to \underbrace{Y}_{\text{NotSK}}, \underbrace{Y \to Z}$

Z is trasitively determined by Superkey (X) means here Y is not Superkey.

If Y becomes a superkey, then Z is said to be only determined by Superkey (X).

(*) so, transitive dependency is not allowed in 3NF.

But Prime Attribute transitively determined by superkey is allowed in 3NF



$\{(X) \to Y, Y \to (Z)\}$ allowed in 3NF.

SK — prime

(BCNF) Boyce-Codd Normal form

BCNF does not allow any FD which allows redundancy i.e. any FD which is nontrivial,

$$X \longrightarrow Y$$

and determinant X is not superkey will forms redundancy.

So, these kind of FDs are not allowed in BCNF.

⊛ Relation Schema R is in BCNF iff every Non trivial FD $X \rightarrow Y$ in Relation R with X must be superkey.

[Relation R satisfied BCNF] $\Longleftrightarrow$ [No Redundancy in Relation R over FD's]

⊛ Trivial FDs are allowed in any NF, as they don't form redundancy.

∴ No condition for trivial ~~dependen~~ FDs are there.

Model Question for Normal Forms :-

Given: R(.....) ⇓ Set of Attr. F = {.....} ⇓ Set of FDs.

which is the highest NF of R?

a) 1NF b) 2NF c) 3NF d) BCNF

TESTING OF BCNF

Every Non trivial FD [(X) $\longrightarrow$ Y]

(X) SK.

TESTING OF 3NF

a) Find all cand keys of R (because we need to know the prime attribute of R)

b) Every Non Trivial FD [(X) $\longrightarrow$ (Y)]

(X) SK (or) (Y) Prime

TESTING OF 2NF

a) Find all cand. keys of R

b) Every {[Proper Subset of cand key]$^+$ = (Prime Attr)}

{No Partial Dependency}

Q. R(ABCD) $F = \{AB \to C,\ C \to A,\ AC \to D\}$
which is the Highest NF of R?

Sol:-
Candidate keys : {AB, BC}

Testing for BCNF (atleast one FD with determinant not SK)

SK (AB) → C ✓

notSK (C) → A ✗
notSK (AC) → D ✗
} so not BCNF

Testing for 3NF

SK (AB) → (C) Prime ✓

C → (A) Prime ✓

(AC) → (D) NotPrime ✗ } so not 3NF
NotSK

Testing for 2NF

Here we take all proper subset of cand. keys and find their closure (closure) & check whether they contain all prime attributes or not.

Cand key ⇒ AB ⇒ $A^+ = (A)$ Prime ✓
$B^+ = (B)$

BC ⇒ $C^+ = CA(D)$ NotPrime ✗

C → D will be a partial dependency in R.

∴ not in 2NF

Hence highest NF of R: 1NF

(*)

	$F = \{\dots\}$		$F^+ = \{\dots\}$
BCNF	True	⟶	True
3NF	True	⟶	True
2NF	True	⟶	May not be True

So, it means if for F, BCNF is true then for F^+ also BC is t

Same for 3NF.

But for 2NF, the testing is little bit different.

If F is in 2NF then also there can be such partial dependencies in F^+ which lead F^+ to n in 2N

So, we check for F^+, if F^+ is in 2NF then we can easily say that F is also in 2NF.

Q. Test & Identify the Highest NF

R(A B C D E) $F = \{ABD \to C, BC \to D, CD \to E\}$

Sol:- Candidate keys : ABD, ABC

Testing for BCNF

SK: (ABD) → C ✓

Not SK: (BC) → D ✗ So not in BCNF

Not SK: (CD) → E ✗

Testing for 3NF

BC → (D) Prime ✓

CD → (E) Not Prime ✗ So not in 3NF

Testing for 2NF

ABD ⇒ $A^+ = A$ $AB^+ = AB$ ✓

$ABC \Rightarrow \quad BC^{+} = BCD(E)$ Not Prime

$BC \to E$ P.Dependency in R

∴ not in 2NF.

Hence highest NF is 1NF.

Q $R(ABCDEF)$

$\{AB \to C \quad C \to DE \quad E \to F \quad F \to A\}$

Cand keys = { AB, BF, BE, BC }

BCNF:

$AB \to C$ ✓

$C \to DE$ ✗

$E \to F$

$F \to A$

3NF

$AB \to C$ ✓

$C \to (DE)$ not Prime ✗

$E \to (F)$ ✓

$F \to (A)$ ✓ Prime

2NF

$AB \Rightarrow A^{+} = A$

$B^{+} = B$

$BF \Rightarrow B^{+} = B$

$F^{+} = FA$

$BE \Rightarrow E^{+} = EFA$

$BC \Rightarrow BC^{+} = CDE$ Not Prime

$C \to D$ is P-Dependency

So not in 2NF

∴ Highest NF is 1NF

Q. R(ABCDEF)

{AB → C, C → D, CD → AE, DE → F, EF → B}

Cand key : AB, AEF, ADE, ~~CD~~, ~~BED~~, ~~ABD~~, C

{AB, AEF, ADE, C}

BCNF :

AB → C ✓

C → D ✓

CD → AE ✓

DE → F ✗ Not in BCNF

EF → B ✗

Not SK

3NF :

AB → (C) ✓

C → (D) ✓

CD → (AE) ✓

DE → (F) ✓

EF → (B) Prime ✓ So, 3NF

Q. R(ABCD)

{AB → C, BC → D}

Cand keys : {AB}

BCNF :

AB → C ✓

(BC) → D ✗ So not in BCNF

Not SK

3NF :

BC → (D) Not prime so not in 3NF

2NF :

AB → A^+ = (A) ✓ Prime

A^+ = (C) Prime So, 2NF

Q. R(ABCDE)

{A → BC, CD → E, B → D, E → A}

Cand key: {A, E, CD, BC}

BCNF:

A → BC

CD → E

not sk (B) → D so not in BCNF

E → A

3NF:

B → (D) Prime

A → (BC)

CD → (E) Prime So 3NF

E → (A)

Q. (A) R(ABC) No Non trivial FDs:

What is Highest Normal form?

[No Non trivial FD's in R] ⇒ [No Redundancy over FD's] ⇒ [BCNF]

So, when all attributes of a relation combine to form a candidate key, then there will be no non trivial FDs in the relation. Hence it will be in BCNF.

(A) For any two attributes Relation, its highest NF is ~~BCNF~~ 6NF.
For this relation; BCNF ✓ 4NF ✓ 5NF ✓ 6NF ✓

Q. R (AB) Possible set of FDs ⇒

a) {A → B}
b) {B → A}
c) {A → B, B → A}
d) {No non trivial FDs}

} All in BCNF as A & B are SK's.

∴ BCNF ✓ 4NF ✓ 5NF ✓ 6NF ✓

If Relation R consist only simple candidate keys.

Then R is always in

a) 1NF but not in 2NF

b) 2NF but may not 3NF

c) 3NF but may not BCNF

d) BCNF.

Sol:- For such relation having only simple candidate k
2NF failure can't occur.

As for 2NF to be failed,

Proper subset of cand key ⟶ Non Prime

but when cand key is simple, then there can't b
any proper subset of cand key.

④ Hence, 2NF will always hold as Partial Depen
not possible

But 3NF may fail

eg R(ABCD) {A→B, B→AC, C→D}

cand key: {A, B}

2NF ✓

3NF ✗ as C → D not Prime
Not SK

or Non Prime → Non Prime
dependency can cause redundancy

Q. Relation R consists of only Prime Attributes] Then R is always in ___ but may not be in ___.

Sol:-

eg suppose R(ABCDE)

A, B, C, D, E are prime

However, we can have such FD in which determinant is not S.K.

∴ Not in BCNF.

However it will always be in 3NF due to all Prime Attributes. X → Ø always prime.

∴ Always in 3NF but may not be in BCNF.

Q. If Relation R in 3NF && atmost one compound key in Relation R. Then —.

i) R in BCNF also.

ii) R may not be in BCNF

iii) R not BCNF.

iv) None.

Sol:- If R is in 3NF but not in BCNF then the following FD:

Proper Subset of CK. → Proper Subset of other CK.

will exist.

To prove it as BCNF, ~~But to hold this~~ we should have to prove that above FD not exists,

for which we require atleast 2 compound keys

If R is 3NF but not in BCNF then

a) Proper subset of CK → Proper subset of other CK

b) atleast 2 compound cand keys.

c) overlapped candkey

Decomposition of Higher NF

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

Cand key: {AB}

Test for 2NF:

B → E is partial dependency
as B → not key (Proper subset of Candkey)
E → Not Prime

2NF Decomposition

R_1	R_2
A C D	B E

Here B is superkey of this table as
$B^+ = (B\ E)$

Now decomposition must follow:

i) Loss less Join Decomposition
(there must be one common attr which is key of atleast one subrelation)
Here B is cand key of R_2, so make it common in both tables

R_1	R_2
A B C D	B E

ii) Dependency Preservation — $AB \to C$, $C \to D$ | $B \to E$

iii) No Partial Dependency

Q. R(ABCDEF) {AB→C, C→D, B→E, E→F}

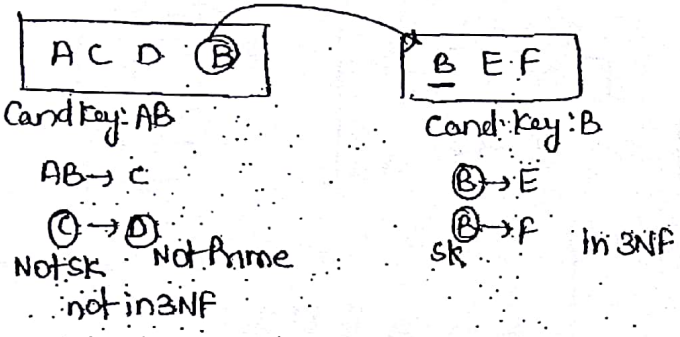
This R is in 1NF but not in 2NF. Cand key: {AB}

as B→E, E→F ⇒ (B→F, B→E) Partial Dependencies.

So, we can't take only BF or B,E in table also.

∴ we take them in one table

2NF Decomposition :-



3NF Decomposition :-

Q. R(ABCDEF) {A→D, B→E, C→F}

How many Relations required for 2NF design?

a) 2 b) 3 c) 4 d) 5

Sol:-

Here cand keys : [ABC]

A→D
B→E
C→F
} all are Partial dependencies. So not in 2NF.

$A^+ = AD$ | $B^+ = BD$ | $C^+ = CD$

A D	B E	C F
A→D	B→E	C→F

to make it LLJ

We have to introduce a common attribute

Method I:-

A D (B)	B E (C)	C F
A→D	B→E	C→F

After introducing common attribute,

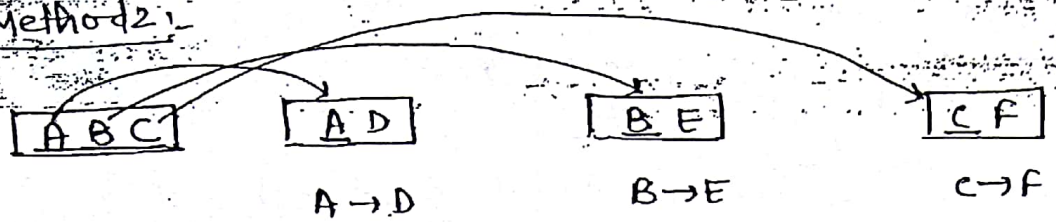
Cand key: (AB)	Cand key (BC)	Cand key: (C)
A→D (becomes P.D.)	B→E (becomes partial dep.)	C→F (No Problem)

Since still partial dependencies are there,

∴ still not in 2NF.

This method fails.

Method 2:-



Now, 2NF ✓
3NF ✓
BCNF ✓

Criteria for BCNF Decomposition :-

1) 4 relations
2) 3 Foreign Keys

Q. R(A B C D E F G)
{AB → C, BC → A, AC → B, B → D, D → E}

Cand key : {ABFG, BCFG, ACFG}

Non Prime Attr : D

& B is proper subset of cand key.

∴ B → D becomes a Partial Dependency

So, not in 2NF.

2NF design

- 2 relations
- 1 FK.

A B C F G	B D E
AB → C	Key B → D
BC → A Prime	Not SK D → E
AC → B	So not in 3NF
So, 3NF	

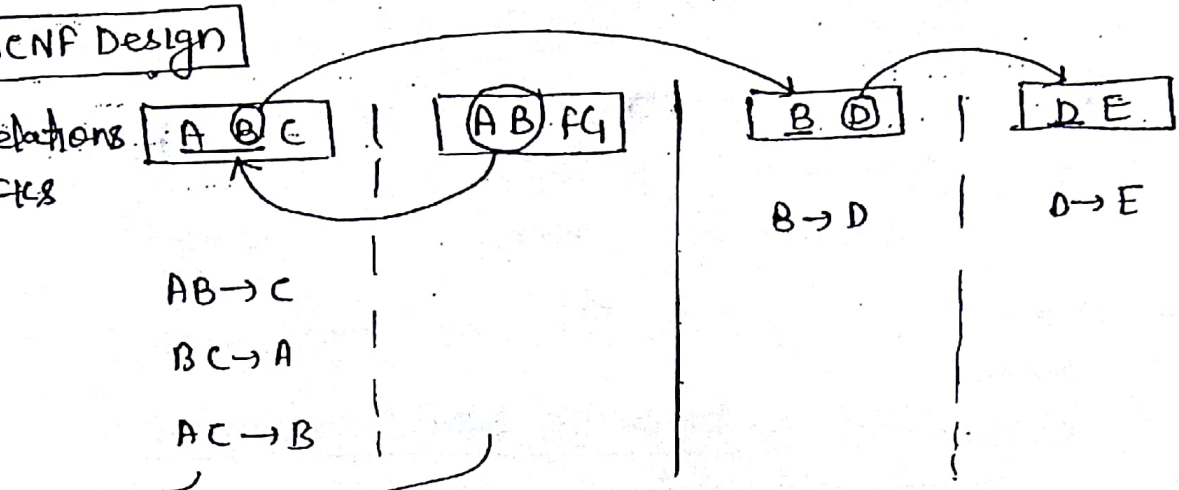
3 NF design

- 3 relations
- 2 FK

A B C F G	B D	D E
AB → C	B → D	D → E
BC → A	SK	SK
AC → B	So, BCNF	So, BCNF
not SK		
so not in BCNF		

BCNF Design

- 4 Relations
- 3 FDs



Here, in place of AB we could take BC or AC also.

→ Here AB, BC, AC are cand. keys.

So, we got BCNF.

Q. R(ABC) { AB → C, C → A }

Sol- Cand. Key ⇒ { AB, BC }

AB → C
C → A
} both satisfy 3NF

but

AB → C ✓

C → A ✗ not BCNF

(C: Not SK)

BCNF Decomposition

(I) BC | AC (C → A)

- LLJ
- BCNF
- Not DP

(II) BC | AB

- Lossy Join
- BCNF
- Not DP

(III) AB | AC (C → A)

- Lossy Join
- BCNF
- Not DP

1NF	2NF	3NF	BCNF	4N
No	No	No	YES → FD's No → MVD's	YES → F YES → M
YES Every relation Possible to decompose into 1NF, 2NF, 3NF, BCNF with Loss Less Join Decomposition.	YES	YES	YES	NO [May
YES Every relation can possible to decompose into 1NF, 2NF, 3NF with Dependency Preserving →	YES	YES	NO [May not] ← Not every relation can decompose into BCNF, 4NF with Dependency Preserva	No [May

So, Although relation may contain some redundancies but the relation should not violate LLJ & DP because design of database itself becomes inconsistent if LLJ & DP are not satisfied.

∴ it is fine to have some redundancies but LLJ and DP have to be satisfied.

MULTIVALUED DEPENDENCY && 4NF

Redundancy in Relations

Non Trivial FD

$X \longrightarrow Y$ (notsk) (For each x value there must be only one Y value)

	Sid	Sname	Cid
Redundancy of FD	S_1	A	C_1
	S_1	A	C_2
	S_1	A	C_3

Sid Cid! Candkey

$Sid \longrightarrow Sname$ (notsk)

These type of redundancies, are removed by decomposition upto BCNF

Non Trivial MVD

$X \twoheadrightarrow Y$ (notsk) (for each X value there can be set of Y values)

$Sid \twoheadrightarrow Cid$ (2 times repeated): S_1 ... C_1/C_2

$Sid \twoheadrightarrow PNo$ (2 times repeated): S_1 P_1/P_2

	Sid	Cid	PNo
Redundancy of MVD	S_1	C_1	P_1
	S_1	C_2	P_1
	S_1	C_1	P_2
	S_1	C_2	P_2

n Simple FD,

$X \rightarrow Y$

each value of X., there must be one value of Y

ut in Multivalued FD,

$X \rightarrow\rightarrow Y$

r each value of X, there can be ~~more than~~ set of values of Y.

eg. sid $\rightarrow\rightarrow$ Cid
S1 C1/C2

sid $\rightarrow\rightarrow$ Ph.No.
S1 P1/P2/P3

Relation R is in BCNF [0% redundancy over FD's]
(and)
Redundancy exists in Relation R
only if atleast two multivalued attributes converted into single valued attributes in R.

, redundancy due to MVD's will be there if

as seen from prev. ex

f R is in BCNF and
atmost one Multivalued Attribute is stored as single valued attribute
hen No Redundancy in R even becoz of MVD's:

R will be in 4NF.

Ex①

Sid	Cid	PNo.
S_1	C_1/C_2	$P_1/P_2/P_3$

Not in 1NF

If Cand key: Sid Cid PNo then no Nontrivial FD or Multivalued FD will be there.

Sid	Cid	PNo
S_1	C_1	P_1
S_1	C_2	P_1
S_1	C_1	P_2
S_1	C_2	P_2
S_1	C_1	P_3
C_1	C_2	P_3

S_1 C_1/C_2 → 3 times repeat

$P_1/P_2/P_3$ → 2 times repeat

Highest NF: EKNF

Sid →→ Cid
Sid →→ PNo

Redundancy Exists

So, we see due to 2 MV attr. it is not in 4NF.

Ex②

Not in 1NF

Sid	Cid	PNo	Email
S_1	C_1/C_2	$P_1/P_2/P_3$	$E_1/E_2/E_3/E_4$
	12 times	8 times	6 times

Repetition

Converting to 1NF

Sid	Cid	PNo	Email
24 tuples			

Highest NF: BCNF

Sid →→ Cid
Sid →→ PNo
Cid →→ Email

Redundancy Exists

If

Sid	Cid	PNo	Email
S_1	C_1/C_2	$P_1/P_2/P_3$	$E_1/E_2/E_3/E_4/E_5$

then total tuples = $2 \times 3 \times 5 = 30$.

Cid repeats ⟶ 15 times

PNo repeats ⟶ 10 times

Email repeats ⟶ 6 times.

Ex③

Sid	Cid
S_1	C_1/C_2

⇒

Sid	Cid
S_1	C_1
S_1	C_2

BCNF ✓

NoRedundancy

∴ 4NF.

Multivalued Dependency :-

X, Y are some set of attributes over R and Z.

R − (XUY)

$X \rightarrow\rightarrow Y$ implied in R only if t_1, t_2, t_3, t_4 tuples over R such that

a) $t_1.X = t_2.X = t_3.X = t_4.X$

(and) b) $t_1.Y = t_2.Y$ and $t_3.Y = t_4.Y$

(and) c) $t_4.Z = t_3.Z$ and $t_2.Z = t_4.Z$

eg Before converting to NF:

X	Y	Z
x_1	y_1/y_2	z_1/z_2

After converting to INF :-

BCNF

(a) cond^n over X (b) cond^n over Y (c) cond^n over Z

R	X	Y	Z
t_1	x_1	y_1	z_1
t_2	x_1	y_1	z_2
t_3	x_1	y_2	z_1
t_4	x_1	y_2	z_2

If cond^n (a) is applied on X and cond^n (b) on Y then -

$$X \rightarrow\rightarrow Y$$

However if we swap t_2 and t_3, then

cond^n (a) is applied on X

cond^n (b) is applied on Z

cond^n (c) is applied on Y

$$X \rightarrow\rightarrow Z$$

(a) cond^n (c) cond^n (b) cond^n

R	X	Y	Z
t_1	x_1	y_1	z_1
t_3	x_1	y_2	z_1
t_2	x_1	y_1	z_2
t_4	x_1	y_2	z_2

If $X \to\to Y$ then $X \to\to R - (XUY)$

eg:

R(ABCD) if $A \to\to B$ [True] $\xrightarrow{\text{Then}}$ $A \to\to CD$ [also true]

Trivial MVD :-

$X \to\to Y$ is trivial MVD only if

$X \supseteq Y$ or $XUY = R$

eg:

R(ABCD)

$AB \to\to CD$, $AB \to\to A$ } Trivial MVD

(A) $X \to\to Y$

x_1	$y_1/y_2/y_3$

MVD free from redundancy if No Z in the relation

R

X	Y
x_1	y_1
x_1	y_2
x_1	y_3

$X \to\to Y$

x_1	y_1/y_2

Forms redundancy because z exists

[Non trivial MVD]

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_1
x_1	y_1	z_2
x_1	y_2	z_2

Here if there are 2 values it will repeat 2 times.

Replication rule:

Every FD can be replaced as MVD if

if $X \to Y$ then $X \to\to Y$.

eg

X	Y
x_1	$[y_1]$ ↑ single Y

$\Longrightarrow$

X	$\to\to$ Y
x_1	$[Y_1]$ ↑ set of Y values

Reverse is not true.

Becoz, Single Y value can belong to set of Y values but set of Y values can't belong to single Y value

(+) $X \to\to Y \not\Longrightarrow X \to Y$

x_1 (y_1/y_2)

(*) MVD not allowed to split/Merge

if $X \to\to YZ$ (true) then $X \to\to Y$, $X \to\to Z$ (may/may not true)

→ No. of times y_1 repeated should be equal to no. of times y_2 repeated in the table.

$X \longrightarrow Y$

(y_1/y_2)

If one record is added, then

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_1
x_1	y_1	z_2
x_1	y_2	z_2
x_1	y_1	z_3

(X) Incorrect entry

then we also have to add 1 record for y_2 also →

(✓) correct entry:

x_1	y_1	z_1
x_1	y_2	z_1
x_1	y_1	z_2
x_1	y_2	z_2
x_1	y_1	z_3
x_1	y_2	z_3

eg

Sid	CNo	RNo
S_1	C_1	R_1
	C_2	R_1
	C_2	R_2

$Sid \rightarrow\rightarrow CNo\ RNo$

Sid	PNo
S_1	P_1
	P_2

$Sid \rightarrow\rightarrow PNo$

6 Records are possible.

Sid	CNo	RNo	PNo
S_1	C_1	R_1	P_1
S_1	C_2	R_1	P_1
S_1	C_2	R_2	P_1
S_1	C_1	R_1	P_2
S_1	C_2	R_1	P_2
S_1	C_2	R_2	P_2

Here,

i) $Sid \rightarrow\rightarrow PNo$ ✓

ii) $Sid \rightarrow\rightarrow CNo\ RNo$ ✓

$Sid \rightarrow\rightarrow CNo$ ✗ $\quad Sid \rightarrow\rightarrow RNo$ ✗

because $Sid \rightarrow\rightarrow CNo$, $Sid \rightarrow\rightarrow RNo$ MVDs to hold, for same student in CNo, the no. of times C_1 repeated must be equal to no. of times C_2 repeated & in RNo, no. of times R_1 repeated must be equal to no. of times R_2 repeated

So, split not possible.

④ But in case,

Sid	CNo	RNo
S_1	C_1	R_1
	C_2	R_2
	C_3	R_3

then split can be possible because in that case no. of times one C_i come other will come only same no. of times for one student.

Q. which is correct?

i) if $A \to BC$ then $A \to\to B$, $A \to C$

ii) if $A \to\to BC$ then $A \to\to B$, $A \to C$

iii) if $A \to BC$ then $A \to\to B$, $A \to\to C$

iv) if $A \to\to BC$ then $A \to\to B$, $A \to\to C$

which are true?

a) i ii b) i, iii c) ii iii

d) i, ii, iii, iv.

copyright©theorypoint.com The Best Gateway Of Success

Sol:- i) True as $A \to BC \Rightarrow A \to B$, $A \to C \Rightarrow A \to\to B$, $A \to\to C$ (can be written as $\to$ can be replaced by $\to\to$)

ii) False $A \to\to BC$ can't split

iii) True same as (i)

iv) False same as (ii)

Fourth Normal Form (4NF)

④ Relational Schema R is in 4NF iff -

a) Every Nontrivial FD $X \to Y$ with X is superkey. [BCNF]

b) Every Nontrivial MVD $X \to\to Y$ with X is superkey.

Contained in

because MVDs can replace FDs.

$X \to Y$ — X superkey

$X \to\to Y$ — X superkey

⑤

Attribute Closure
Candidate key
Super key

should be found using only FD set.

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

Candidate key = ?

Here, $A^+ = \{AB\}$ ignore MVDs $A \to\to C$, $A \to\to D$

∴ candidate key: [ACD]

$[ACD]^+ = \{ABCD\}$

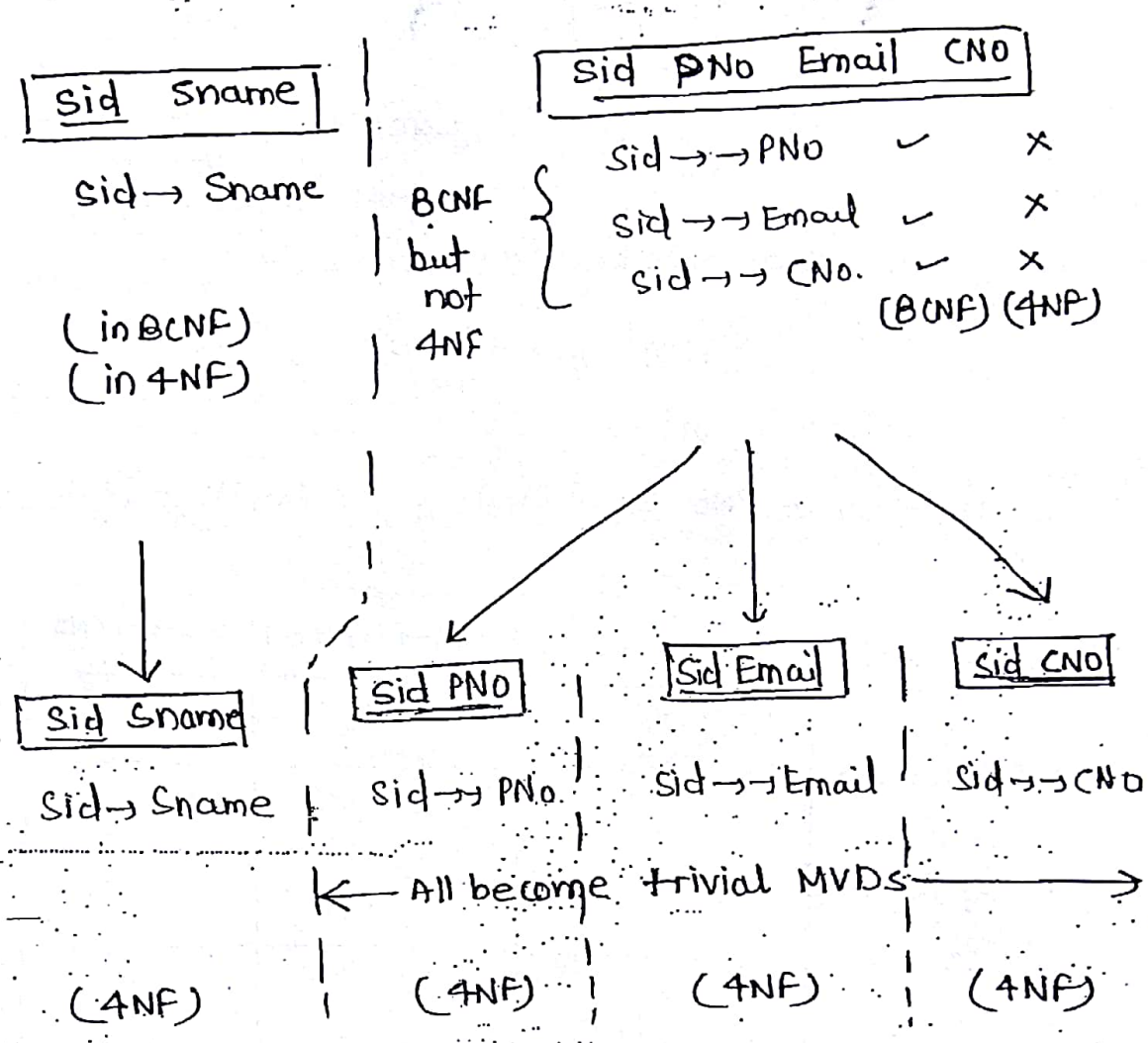
Q.

R(sid Sname PNo Email CNo) Cand key = (sid PNo Email CNo)

{

	sid → Sname	sid →→ PNo	sid →→ Email	Sid →→ CNo
4NF	✗ as sid not SK.	✗ same	✗ same	✗ same
BCNF (allowed all MVDs)	✗ as (sid) → Sname, not SK	✓	✓	✓
3NF (allow all MVDs)	✗ (sid) → (Sname), not SK, not Prime	✓	✓	✓
2NF allow all MVDs	✗ (sid PNo Email CNo) → Sname, Proper Subset, (Partial Dependency)	✓	✓	✓
1NF (allow all MVDs)	✓	✓	✓	✓

Decomposition.



QUERIES

PROCEDURAL QUERIES

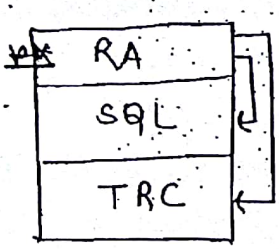
Formulation of

* what data retrieved from DB

(and) * how data retrieved from DB

⇓

RELATIONAL ALGEBRA



NON PROCEDURAL QUERIES

Formulation of

* what data retrieved from DB

x: variable

$\exists_x(P(x))$: stmt

⇓

RELATIONAL CALCULUS (using 1st order Logic)

Tuple Relational Calculus (variable refers to set of varibles)

(T) ∈ student

↳ Tuple Variable

⇓

SQL

Domain Relational Calculus (variable for each column)

<X, Y, Z> ∈ Student

Student (Sid, Sname, Age)

x, y, z

variable for each column

⇓

QBE (Query By Example)

RELATIONAL ALGEBRA

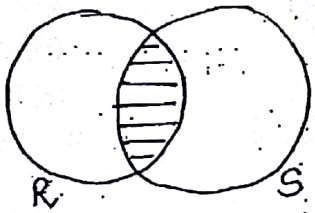
Basic Operators

π	Projection
σ	selection
$\times$	Cross Product
$\cup$	Union
$-$	set Difference
ρ	Rename

Derived Operators

$\bowtie$	Join {using $\times$, σ, π}
$\cap$	Intersection {using $-$}
$/$	Division {using $\times$, $-$, π}

ⓐ Intersection (using two "$-$" operators)



$$R \cap S \equiv R - (R - S)$$

ⓑ Projection (π)

↳ It is used to project required attributes.

$$\pi_{\text{Attributes}}(R)$$

eg R

A	B	C
2	4	6
3	4	2
4	4	6
5	5	3

$\xrightarrow{\pi_{B,C}(R)}$

B	C
4	6
4	2
5	3

} distinct records in Result

↳ It is used to select records based on specified conditions.

R

A	B	C
2	4	6
3	4	2
4	4	6
5	5	3

$\sigma_{A \geq 4}(R)$ →

A	B	C
4	4	6
5	5	3

$\sigma_{Cond^n}(R)$

---

④ Cross Product (X)

(Binary operator) → applied on 2 tables.

R X S ⇒ results all attributes of R followed by all attributes of S

(and)

each record of R pairs with every record of S.

eg:

R

A	B	C
2	4	6
3	4	4
4	2	4

n distinct tuples

← x →

S

C	D
4	3
7	4

m distinct tuples

← y →

RXS

A	B	C	C	D
2	4	6	4	3
2	4	6	7	4
3	4	4	4	3
3	4	4	7	4
4	2	4	4	3
4	2	4	7	4

nxm distinct tuples

← x+y →

⑧ Cross Product is used to compare data of two tables.

⑨ The selection operator ($\sigma_c(R)$) is unary operator & this can be used to select tuples from only one table.
If we want to select tuples from two tables at a time (to compare etc.) we can apply ($\sigma_c(RXS)$).

Q: if $|R| = n$ $|S| = 0$.

then $|R \times S| = ?$

Sol: 0 tuples.

(iv) R×S result empty record set if either R or S or both are empty record set.

eg: $R = \{2, 4, 6\}$ $S = \{\}$

then $R \times S = \{\}$ // Empty

(v) Natural Join ($\bowtie$)

$R \bowtie S$ Equal to

1) $R \times S$

and 2) $\sigma_p(R \times S)$

p: Equality b/w common attributes of R & S

and 3) $\pi_{\text{distinct attr.}}(\sigma_p(R \times S))$

$$R \bowtie S \equiv \pi_{ABCD}(\sigma_{R.C=S.C}(R \times S))$$

ABCD: distinct attr. ; $R.C=S.C$: p

eg

R

A	B	C
2	4	5
3	4	4
4	2	4

S

C	D
4	3
7	4

$R \bowtie S$

A	B	C	D
3	4	4	3
4	2	4	3

Cardinality of $R \bowtie S \equiv \{0 \text{ to } n*m\}$

ural Join = Cross Product (if Join Condn is Empty)

means

(No common attribute)

eg. A(AB) S(CD)

then $R \bowtie S \equiv R \times S$

ardinality of $R \bowtie S = 0$

when there are common attributes

but the equality condition is not satisfied.

eg:

A	B	C
2	4	6
3	4	4
4	2	4

X

C	D
14	3
7	4

when $R \times S = 0$

(or) either R or S is Empty.

one column of common attr. is removed.

R×S

A	B	C	C	D	
2	4	6	4	3	C-C not common
2	4	6	7	4	C-C "
3	4	4	4	3	
3	4	4	7	4	C-C "
4	2	4	4	3	
4	2	4	7	4	C-C "

R⋈S

JOINS

1) Natural Join ($\bowtie$)
2) Conditional Join ($\bowtie_c$)
3) Outer Joins
   a) Left outer Join ($\bowtie$ | $\bowtie_c$)
   b) Right outer Join ($\bowtie$ | $\bowtie_c$)
   c) Full outer Join ($\bowtie$ | $\bowtie_c$)

CONDITIONAL JOIN

$$R \bowtie_c S \equiv \sigma_c (R \times S)$$

↳ instead of equality we can define any condition.

eg

R

A	B	C
2	6	8
3	4	2
7	2	6

S

C	D
9	3
4	5
7	2

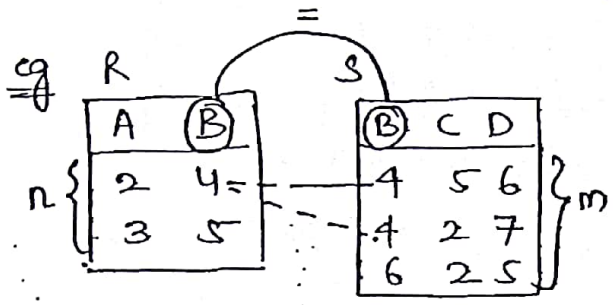
$$R \underset{R.C>S.C}{\bowtie} S \equiv \sigma_{R.C>S.C} (R \times S)$$

A	B	C	C	D
2	6	8	4	5
2	6	8	7	2
7	2	6	4	5

Cardinality : {0 to n*m tuples}

a) Left Outer Join (⟕)

R ⟕ S : results R ⋈ S tuples, (and) tuples of R those have failed join condition. with entries of S as NULL.



R ⟕ S

A	B	C	D
2	4	5	6
2	4	2	7
3	5	NULL	NULL

Cardinality : {n to n*m tuples}

Ⓐ Left Outer Join is a Derived Operator. gives

$R ⟕ S \equiv (R ⋈ S) \cup ((R - \Pi_R (R ⋈ S)) \times (null, Null..))$

Tuple of R which are not in R⋈S

b) Right Outer Join (⟖)

R ⟖ S =

A	B	C	D
2	4	5	6
2	4	2	7
NULL	6	2	5

cardinality : {m to n*m tuples}

c) Full Outer Join (⟗)

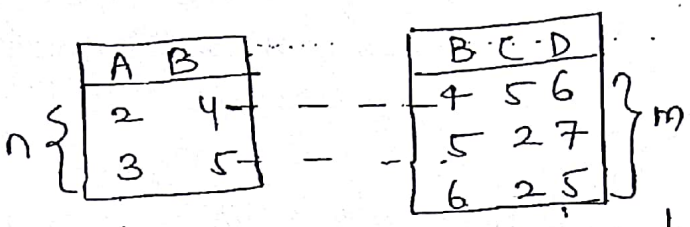
$R ⟗ S \equiv (R ⟕ S) \cup (R ⟖ S)$

R ⟗ S

A	B	C	D
2	4	5	6
2	4	2	7
3	5	NULL	NULL
NULL	6	2	5

Cardinality : { max(m,n) to n*m tuples }

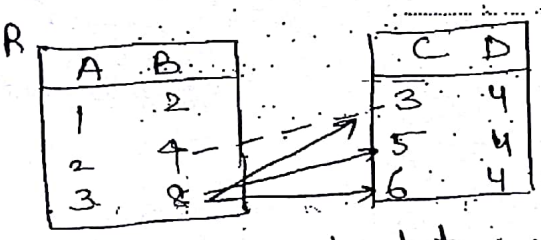
Minimum is max(m,n) as follows -



Here result of outer join contains
max(n,m) = m tuples.

(*) R(AB) S(CD)

Retrieve tuples of R such that 'B' value of R more than some (atleast one) 'C' value of S.
= any or



To retrieve such type of data we use conditional join.

By using cross product ; to retrieve this data,

$$\Pi_{A,B}\left(\sigma_{R.B>S.C}(R \times S)\right)$$

By using Conditional Join

$$\Pi_{A,B}\left(R \bowtie^{S}_{R.B>S.C}\right)$$

Both are same queries. (gives same result)

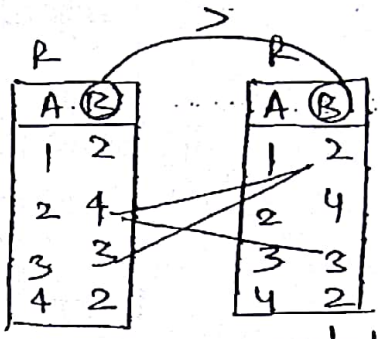
Retrieve tuples of R such that B value of R is more than some B value of R.

eg:

R

A	B	
1	2	X
2	4	✓
3	3	✓
4	2	X

for this we do like R×R →

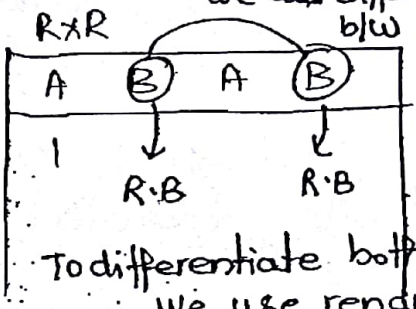


are not able to differentiate b/w both cols

$\rho(T_1, R)$ $\rho(T_2, R)$

R renamed as T_1 R renamed as T_2

$$\pi_{T_1.A, T_1.B}\left(\sigma_{T_1.B > T_2.B}(T_1 \times T_2)\right)$$



To differentiate both cols we use rename (ρ) operator.

(OR)

$$\pi_{A,B}\left(\sigma_{B>BB}\left(R \times \rho_{AA,BB}(R)\right)\right)$$

A	B	AA	BB

(OR)

$$\pi_{A,B}\left(R \bowtie_{B>BB} \rho_{AA,BB}(R)\right)$$

Rename (ρ) :-

student (sid, sname, marks)

① $\rho(T, \text{student}) \Rightarrow$ T (sid sname marks) } Rename Table Name (Table Name changed)

② $\rho_{ID,SN,MK}(\text{student}) \Rightarrow$ Student (ID SN MK) (Table Name same)

③ $\rho_{1 \to ID,\ 3 \to MK}(\text{student}) \Rightarrow$ Student (ID Sname MK) } Rename column Name

(*) Rename operator is used to remove ambiguity between columns with same names.

⑧ Student (sid, gender, Marks)

Retrieve female students who scored marks more than marks of some male student. (atleast)

(I) $\rho_1(T_1, \text{Student})$ $\rho_2(T_2, \text{Student})$

$$\Pi_{T_1.sid}\left(\sigma_{T_1.gender=Male \wedge T_2.gender=female \wedge T_1.marks > T_2.marks}(T_1 \times T_2)\right)$$

(II) $\Pi_{sid}\left(\text{Student} \bowtie_{gender=female \wedge GN=male \wedge marks>MK} \rho_{ID,GN,MK}(\text{Student})\right)$

Now, if there are 5 records in student table, then

in (I)

$T_1 \times T_2$, contain 25 tuples

3 comparisons for each record

$\Rightarrow 25 * 3 = 75$ comparisons

If we apply comparisons (conditions) before computing cross product, then no. of comparisons can reduce.

$$\Pi_{sid}\left[\sigma_{gender=female}(\text{Student}) \overset{2*3}{\bowtie}_{marks>MK} \rho_{ID,GN,MK}\left(\sigma_{gender=male}(\text{Student})\right)\right]$$

5 Comp. 6 Comp 5 Comp.

$5+5+6 = 16$ comparisons

(3) Retrieve tuples of R such that B value of R more than every C values of C

R

A	B
1	2
2	4
3	8

S

C	D
3	4
5	4
6	4

We can't use conditional join operator directly because $\bowtie_c$ is used in case of "some" values to be retrieved

We use here complementary method.

$$\neg(>\text{Every}) \equiv \leq \text{some}$$

[B value more than every C val] ≡ [All B Values] − [B values such that B value of R ≤ some values of C]

means atleast one/a

$$\pi_{A,B} - \pi_{A,B}\left(R \bowtie_{B \leq BB} \rho_{AA,BB}(S)\right)$$

(Now we can use conditional join

(4) Retrieve female students who scored marks more than marks of every male student.

$$\pi_{sid} - \left[\pi_{sid}\left[\sigma_{gender=female}(Student) \bowtie_{Marks \leq MK} \rho_{ID,GN,MK}\left(\sigma_{gender=male}(Stu\right.\right.\right.$$

[Females scored marks more than every male] ≡ [All Female students] − [Female scored Marks <= some male]

$$\pi_{sid}\left(\sigma_{gender=female}(Student)\right) - \pi_{sid}\left(Student \bowtie \rho_{ID,GN,MK}(Student\right.$$

Gender=female ∧ GN=male ∧ marks ≤ mk

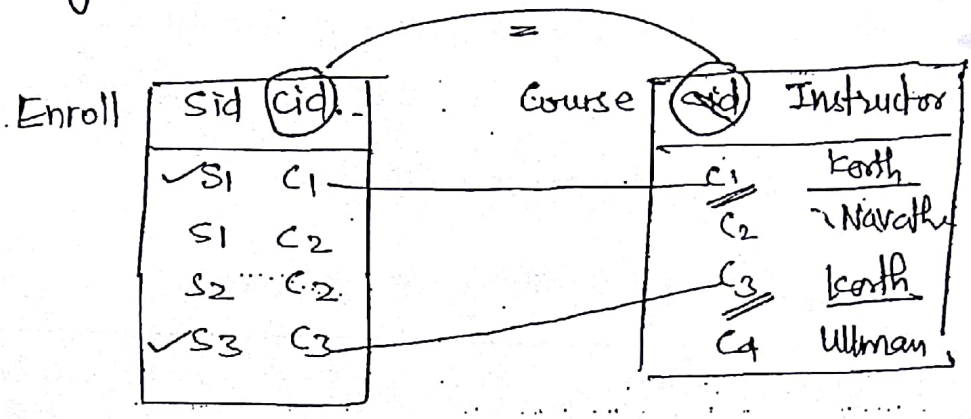
Q

Student (Sid , Sname , Sage) — All Students

Course (Cid , Cname , Instructor) — All courses

Enroll (Sid Cid grade) — Relationship

(*) Retrieve Sid's who enrolled some (atleast) course taught by Korth.



(1) $\Pi_{sid}\left(\sigma_{Enroll.Cid=Course.Cid \wedge Inst.=Korth}\left(Enroll^{4} \times Course^{4}\right)\right)$ — Cross Prod. $(16*2=32)$ comp — 2 Cond^{n} each

(2) More Efficient

$\Pi_{sid}\left(\sigma_{Enroll.Cid=Course.Cid}\left(Enroll \times \sigma_{Inst=Korth}(Course)\right)\right)$ — 8 comp. — 4 comp

$(4+8) = 12$ comp

(3) $\Pi_{sid}\left(\sigma_{inst=korth}\left(Enroll \bowtie Course\right)\right)$

(4) More Efficient

$\Pi_{sid}\left(Enroll \bowtie \sigma_{inst=Korth}(Course)\right)$ — 12 comp — (same as ②)

(5)

$\Pi_{Sid}\left(Enroll \bowtie_{Enroll.Cid=Course.Cid} Course\right)$ (same as ①)

Q. R(A B C)

Write equivalent expressions for

a) $\sigma_{R.C=S.C \wedge B>10 \wedge D<10}(R \times S) \equiv \sigma_{R.C=S.C}\left(\sigma_{B>10}(R) \times \sigma_{D<10}(S)\right)$

b) $R \bowtie_{R.C=S.C \wedge B>10 \wedge D<10} S \equiv \sigma_{B>10}(R) \bowtie_{R.C=S.C} \sigma_{D<10}(S)$

c) $\sigma_{B>10 \wedge D<10}(R \bowtie S) \equiv \sigma_{B>10}(R) \bowtie \sigma_{D<10}(S)$

SET OPERATORS

$\cup$: union

$-$: Minus

$\cap$: Intersection

To Apply $\cup, \cap, -$ b/w Relations R & S, Relations must be union compatible.

R & S are union compatible iff

1) no. of attributes of R & S must be same (and)

2) domain of ith column of R must be same as domain of jth column of S

domain (A) : possible values accepted by Attribute A.

Consider the cases

① $\pi_{sid, sname}(\ldots)$ $\cup / \cap / -$ $\pi_{sid, marks}(\ldots)$

Can't be applied

domains of both is different.

② $\pi_{sid, sname}(\ldots)$ $\cup / \cap / -$ $\pi_{StudID, StudName}(\ldots)$

Domains same

So, $\cup, \cap, -$ allowed.

eg

R

A	B
2	4
3	6
3	7

} n distinct tuples

S

C	D
4	2
3	6
3	7

} m distinct tuples

UNION:-

$$R \cup S \equiv \{ x \mid x \in R \vee x \in S \}$$

(OR)

$R \cup S$ (Resulted schema is same as R) $\equiv$ R

A	B
2	4
3	6
3	7
4	2

} distinct tuples

means table name & col names are same as that of R.

R(A,B) S(A,B) cols are same	then $R \cup S = R \bowtie S$

Cardinality { ... }

INTERSECTION

$$R \cap S \equiv R - (R - S)$$

$$R \cap S \equiv \{X \mid X \in R \wedge X \in S\}$$

(AND)

$R \cap S \equiv R$

A	B
3	6
3	7

} distinct tuples

R (A, B) S (A, B)	Then $R \cap S \equiv R \bowtie S$

same cof. &
same Attr. name

Cardinality of $R \cap S = \{0$ to $\min(n, m)\}$ tuples

MINUS

$$R - S \equiv \{X \mid X \in R \wedge X \notin S\}$$

$R - S \equiv R$

A	B
2	4

Cardinality of $R - S = \{0$ to n tuples$\}$

Course (Cid, Cname, Instructor)

Enroll (Sid Cid grade)

Retreive sid's enrolled some course taught by Korth (or) some course taught by Navathe.

Sol:

$$\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{Inst=Korth \vee \\ Inst=Navathe}}(\text{Course})\right)$$

(or)

$$\pi_{sid}\left(\text{Enroll} \bowtie \sigma_{Inst=korth}(\text{Course})\right) \cup$$

Can use ∩ also, as attr. are same for both projections. (Sid)

$$\pi_{sid}\left(\text{Enroll} \bowtie \sigma_{Inst=Navathe}(\text{Course})\right)$$

(4) Retrieve Sid's enrolled some course taught by Korth (and) some course taught by Navathe. Test the following queries -

Query 1] $\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{Inst=Korth \wedge \\ Inst=Navathe}}(\text{Course})\right)$

Query 2] $\pi_{Sid}\left(\text{Enroll} \overset{(I)}{\bowtie} \sigma_{Inst=Korth}^{(II)}(\text{Course})\right) \overset{(V)}{\cap}$

$$\pi_{sid}\left(\text{Enroll} \overset{(III)}{\bowtie} \sigma_{\underset{IV}{Inst=Navathe}}(\text{Course})\right)$$

Query 3] $\rho(T_1, \text{Enroll})$ $\rho(T_3, \text{Enroll})$

$\rho(T_2, \text{course})$ $\rho(T_4, \text{course})$

~~$\pi_{T_1 \cdot Sid}(\sigma_{T_1 \cdot Cid = T_2 \cdot Cid \wedge T_2 \cdot Inst = Korth \wedge T_3 \cdot Sid = T_1 \cdot Sid \wedge T_3 \cdot Cid = T_4 \cdot Cid \wedge T_4 \cdot Inst = Navathe}(T_1 \times T_2 \times T_3 \times T_4))$~~

$$\Pi_{T_1.sid}\left(\sigma_{T_1.cid = T_2.cid \,(I)\ \wedge\ T_2.Inst. = Korth \,(II)\ \wedge\ T_3.cid = T_4.cid \,(III)\ \wedge\ T_4.Inst. = Navathe \,(IV)\ \wedge\ T_1.sid = T_3.sid \,(V)}(T_1 \times T_2 \times T_3 \times T_4)\right)$$

Query 4]

$$\Pi_{sid}\left(\sigma_{T_1.cid = T_2.cid\ \wedge\ T_2.Inst = Korth\ \wedge\ T_1.cid = T_4.cid\ \wedge\ T_4.Inst. = Navate}(T_1 \times T_2 \times T_4)\right)$$

Sol:-

Query 1] Wrong

Result: Empty Record set

as no record in table have both Inst. as Korth or Navathe

Query 2] Correct

Query 3] Correct

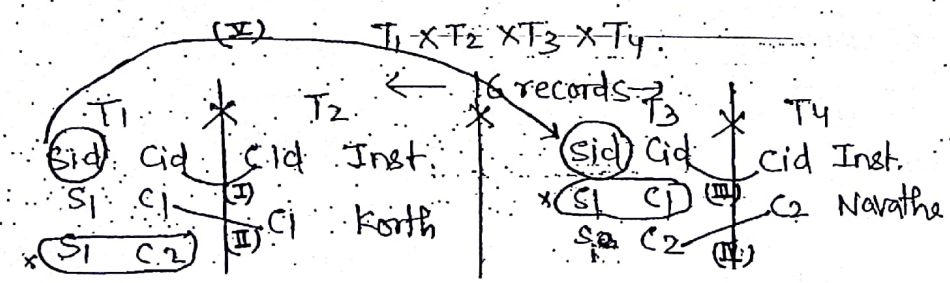
eg

Enroll

Sid	Cid
S_1	C_1
S_1	C_2

Course

Cid	Inst.
C_1	KORTH
C_2	NAVATHE



Result = S_1

Note: Query 2 and Query 3 are same.

Query 4 – Wrong

$T_1 \times T_2 \times T_4$

Sid	Cid
S_1	C_1
S_1	C_2

Cid	Inst
C_1	Korth

Cid	Inst.
C_2	Navathe

① $T_2.Inst = Korth$

$T_1.cid = T_2.cid$ ....

So $T_1.cid = C_1$

② $T_4.Inst = Navathe$

$T_1.cid = T_4.cid$ ... $T_4.cid = C_2$

③ $T_1.cid = T_4.cid$ becomes false (✗)

So, empty relation will be output.

DIVISION (÷)

Enroll (Sid Cid ...)

S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_2
S_3	C_1

⇑ Students enrolled some course

Course (Cid ---)

C_1
C_2
C_3

⇑ All Courses

⊛ Retrieve Sid's of those students which are enrolled in ~~a~~ every course. ~~for every course~~

$$\pi_{Sid\,Cid}(Enroll) \,/\, \pi_{Cid}(course) \Rightarrow$$

Sid
S_1

eg

S_1	C_1
S_1	C_2
S_1	C_3

C_1, C_2, C_3

↳ if S_1 pairs with every course C_1, C_2, C_3 in denominator then S_1 will come in the result.

"A" value must be paired with every "B" in denominator.

$\pi_{AB}(R) \,/\, \pi_{B}(S) \equiv$ Retrieves A values which are pairs with every B value of S

Sid's that have enrolled every course ⇒

a) (Retrieve disqualified data)
Retrieve Sid's not enrolled every course.

⊛ $\pi_{Sid}(\pi_{Sid}(Enroll) \times \pi_{Cid}(Course) - \pi_{SidCid}(Enroll))$

(disqualified tuples for Division)

$\pi_{sid}(Enroll) \times \pi_{cid}(Course) - \pi_{sid\,cid}(Enroll)$

All students: s_1, s_2, s_3 × All courses: c_1, c_2, c_3

sid	cid
~~s_1~~	~~c_1~~
~~s_1~~	~~c_2~~
~~s_1~~	~~c_3~~
~~s_2~~	~~c_1~~
~~s_2~~	~~c_2~~
s_2	c_3
~~s_3~~	~~c_1~~
s_3	c_2
s_3	c_3

Every student that are enrolled in every course

[All possible pairs b/w Sids & Cid's]

−

sid	cid
s_1	c_1
s_1	c_2
s_1	c_3
s_2	c_1
s_2	c_2
s_3	c_1

[Actual pairs b/w sid's & Cid's]

=

sid	cid
s_2	c_3
s_3	c_2
s_3	c_3

[Non pairing data]

π_{sid}

sid
s_2
s_3

[Sid's not enrolled in every Course]

or

Sid's Enrolled in Proper Subset of Courses.

$$\left(\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \supset \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_1c_2 \\ c_1c_3 \\ c_2c_3 \end{bmatrix} \right)$$

b) [Sid's Enrolled in Every Course] = [Sids Enrolled in some Course] − [Sid's not Enrolled in every Course]

$$\pi_{sid\,cid}(Enroll) \,/\, \pi_{cid}(Course) = \pi_{sid}(Enroll) - \pi_{sid}\left(\pi_{sid}(Enroll) \times \pi_{cid}(Course) - \pi_{sid\,cid}(Enroll)\right)$$

→ Result = All tuples − disqualified tuples

General Expression for ...

$$\pi_{AB}(R) / \pi_{B}(S) \equiv \pi_{A}(R) - \pi_{A}\left(\pi_{A}(R) \times \pi_{B}(S) - \pi_{AB}(R)\right)$$

(*) π, $\times$, $-$ are used to derive division ($/$) operator.

$$\pi_{ABCD}(R) / \pi_{CD}(S) \equiv \pi_{AB}(R) - \pi_{AB}\left(\pi_{AB}(R) \times \pi_{CD}(S) - \pi_{ABCD}(R)\right)$$

(*)

$$R/S \equiv \pi_{R-S}(R) - \pi_{R-S}\left(\pi_{R-S}(R) \times S - R\right)$$

$\pi_{R-S}(R)$: Project attr. of R but not S

$\pi_{R-S}(R) \times S$: Here we not applied projection under assumption that S has only C D. attr.

$- R$: Here we not applied projection under assumption that S has only ABC

Q ***

R with attribute set X and n distinct tuples

S with attribute set Y and m distinct tuples

($m > 0$ or S is non empty)

i) To perform R/S required condition: $X > Y$

ii) Resulted attribute set of R/S is $X - Y$

iii) Cardinality [# of tuples in result of R/S] is

{0 tuples to $\lfloor n/m \rfloor$ tuples}

0 tuples

A

Empty record

$\equiv \pi_{AB}(R) / \pi_{B}(S)$

a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_3

$\lfloor n/m \rfloor$ tuples

A
a_1
a_2

$\equiv \pi_{AB}(R) / \pi_{B}(S)$

a_1	b_1
a_1	b_2
a_1	b_3
a_2	b_1
a_2	b_2
a_2	b_3

n tuples, m tuples

Q. Write the query for

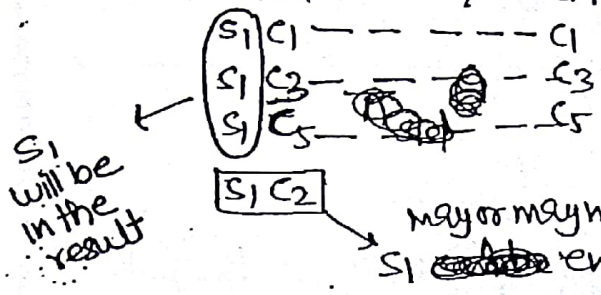
Student (sid name age)

Course (cid Cname Inst)

Enroll (Sid Cid grade)

(1) Retrieve Sid's enrolled for every course taught by Korth.

$$\pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Korth}(Course))$$



$$= \pi_{sid}(Enroll) - \pi_{sid}\left(\left(\pi_{sid}(Enroll) \times \pi_{cid}(\sigma_{Inst=Korth}(Course))\right) - \pi_{sid,cid}(Enroll)\right)$$

sid: enrolled any course

Not enrolled every Korth course

(2) Retrieve Cid enrolled by every student whose age is more than 20.

$$\pi_{cid,sid}(Enroll) / \pi_{sid}(\sigma_{age>20}(Student))$$

Q. Which of the Query correctly represents sid's enrolled Every course taught by Korth & Every course taught by Navathe.

Q1] $\pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Korth \vee Inst=Navathe}(Course))$

Q2] $\left[\pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Korth}(Course))\right] \cap \left[\pi_{sid,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Navathe}(Course))\right]$

i) Q_1 correct but not Q_2

ii) Q_2 correct but not Q_1

iii) $Q_1 Q_2$ both correct

iv) $Q_1 Q_2$ incorrect.

Q.

Enroll (sid cid)

s1	c1
s1	c2
s2	c3
s3	c2

Course (cid Inst)

c1	Navathe
c2	Ullman
c3	Navathe

1) # of tuples resulted by RA query?

$$\pi_{sid,cid}(\text{Enroll}) / \pi_{cid}\left(\sigma_{Inst=Korth}(\text{Course})\right)$$

a) 4 ~~b) 3~~ c) 2 d) 1 e) 0

2) # of tuples resulted by RA query?

$$\pi_{sid}\left(\text{Enroll} \bowtie \sigma_{Inst=Korth}(\text{Course})\right)$$

a) 4 b) 3 c) 2 d) 1 ~~e) 0~~

Soln

1) Expanding the given query,

$$\pi_{sid}(\text{Enroll}) - \pi_{sid}\left(\pi_{sid}(E) \times \pi_{cid}\left(\sigma_{Inst=Korth}(C)\right) - \pi_{sid,cid}(E)\right)$$

π_{sid}(Enroll): s1, s2, s3

$\pi_{sid}(E) \times \pi_{cid}(\sigma_{Inst=Korth}(C))$: $\phi \times \{s1, s2, s3\} \times \phi = \phi$

$\pi_{sid,cid}(E)$: ϕ

3 records s1, s2, s3

2) $\sigma_{Inst=Korth}(C)$ is empty set

Join output will be empty

So # tuples = 0

(*) Note that:

query 1] $\pi_{sid\,cid}(Enroll) / \pi_{cid}(\sigma_{Inst=Korth}(Course))$: Sid's enrolled every Korth Cour

∀

sid
s1
s2
s3

Query 2] $\pi_{sid}(Enroll \bowtie \sigma_{Inst=Korth}(Course))$: sid's enrolled some Korth Course

∃

sid

Korth Course = { } // Empty

(*) Student (sid Sname age)

course (cid cname Inst)

Enroll (sid Cid Grade)

Q) Retrieve sid's enrolled in atleast two courses.

eg ... t_1, t_2 are tuples that belong to ~~Course~~ Enroll

such that $t_1.sid = t_2.sid \wedge$

$t_1.cid \neq t_2.cid$ then t_1 sid Enrolled in atleast 2 cour

=

≠

Enroll

sid	cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s3	c1

×

Enroll

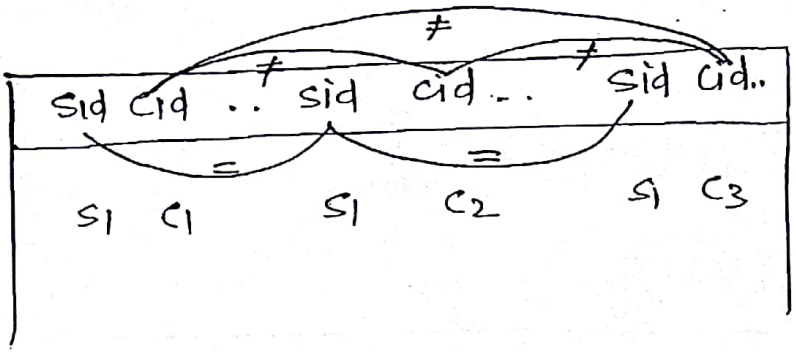
sid	cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s3	c1

$\pi_{sid}(Enroll \bowtie_{sid=S \wedge cid \neq C} \rho_{S,C,G}(Enroll))$

Q) Retrieve Sid Enrolled at least three courses.

$\rho(T_1, Enroll)$ $\rho(T_2, Enroll)$

$\rho(T_3, Enroll)$



$$\Pi_{T_1.Sid}\left(\sigma_{\ldots}(T_1 \times T_2 \times T_3)\right)$$

$T_1.sid = T_2.sid \wedge T_2.sid = T_3.sid$

$\wedge T_1.Cid \neq T_2.Cid \wedge T_2.cid \neq T_3.cid$

$\wedge T_1.cid \neq T_3.Cid$

Q) Retrieve Sid Enrolled for only two courses.

A) Using $\bowtie_c$ or [X and σ] we can retreive atleast two/thrice.

But we can't get "only two" directly from [X and σ] or $\bowtie_c$

$$\begin{bmatrix}\text{Sid's Enrolled}\\ \text{only two}\\ \text{Courses}\end{bmatrix} = \begin{bmatrix}\text{Enrolled}\\ \text{atleast}\\ \text{2 courses}\end{bmatrix} - \begin{bmatrix}\text{Enrolled}\\ \text{atleast}\\ \text{3 Courses}\end{bmatrix}$$

Q) Retrieve Sid Enrolled for only three courses.

$$= \begin{bmatrix}\text{Enrolled}\\ \text{atleast}\\ \text{3 courses}\end{bmatrix} - \begin{bmatrix}\text{Enrolled atleast}\\ \text{4 courses}\end{bmatrix}$$

e sids whose age is highest?

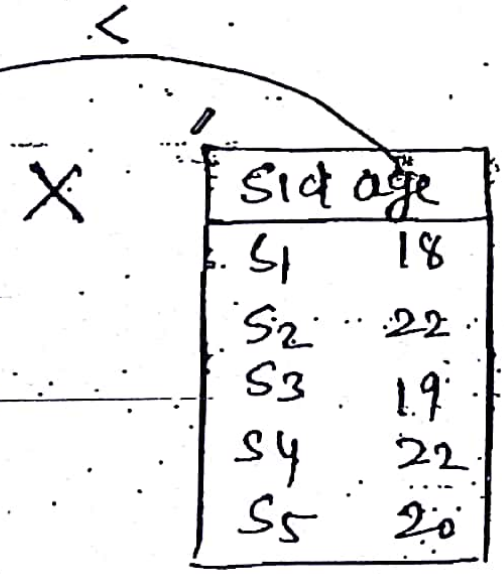
we can't find it directly using RA operators.

id	age
1	18
2	22
3	19
4	22
5	20

For finding highest age:

i) sid's whose age not highest means

(sid's whose age is less than age of some other student atleast one



ge is highest] = [All students] − [sid's whose age is not highest / sid's whose age is less than some other student]

$$\pi_{sid}(Student) - \pi_{sid}\left(Student \bowtie_{age<A4} \rho_{ID,SN,A4}(Student)\right)$$

S1	A	20
S2	A	21
S3	A	22
S4	B	
S5	1	

Q. Retrieve Sid's whose age is highest for each name.

Sid	sname	Age
S1	A	20
S2	A	21
S3	A	22
S4	B	21
S5	B	19

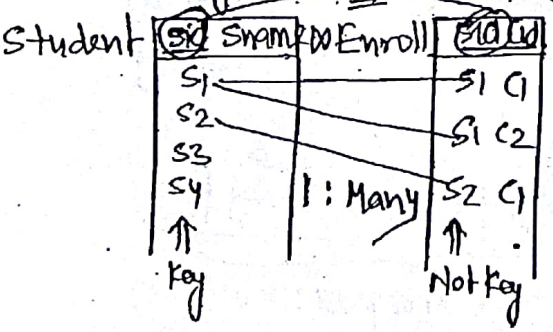
S3
S4

Summary:-

$R \bowtie_c S$	Some / Atleast one record that satisfy condition c.
$R - (\pi_R(R \bowtie_c S))$	every / All tuples that satisfy complement of condition c.
R/S	Records of R that pairs with every record of S.
$R - S$	R but not S
$R \cup S$	R or S
$R \cap S$	R and S
$R \bowtie_c R$	Atleast two records that satisfy condition c.
$R - R \bowtie_{<,>} R$	Records of R with max/min. based on condition c.

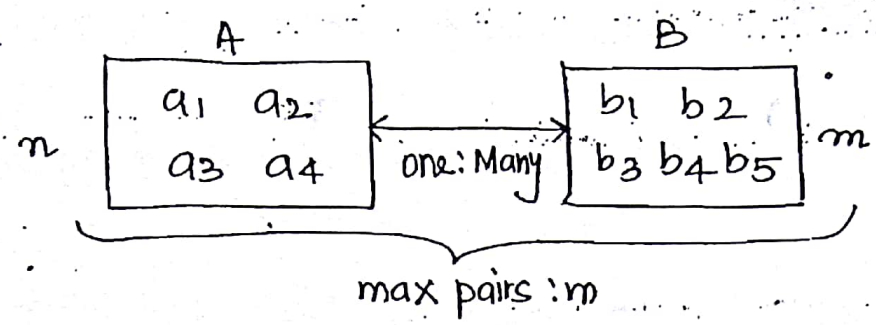
Q) Student (Sid Sname) with 200 tuples
Enroll (Sid Cid) with 100 tuples
How many max. tuples in result of student ⋈ Enroll ?
a) 100 b) 200 c) 2000 d) 20000

Here key constraint plays an imp. role.



When we apply 1 to Many Mapping, then max. tuples result will be equal to # of tuples on "Many" side.
i.e. 100 tuples.

(A)



(*)

Stud ⋈ Enroll

Then Max tuples = 100

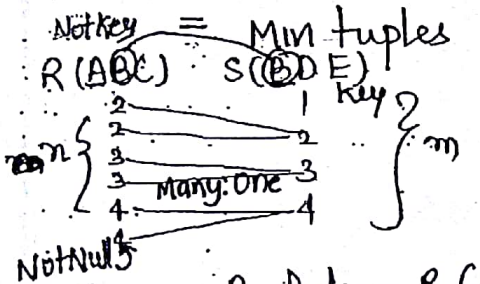
Min tuples = { 100 , (with FK)
0 , (NO FK)

Q Relation R (A B C) A : Primary key && n tuples

Relation S (B D E) B : Primary key && m tuples

Max tuples in R ⋈ S : n

Min tuples in R ⋈ S : { n [with FK]
0 [NO FK]

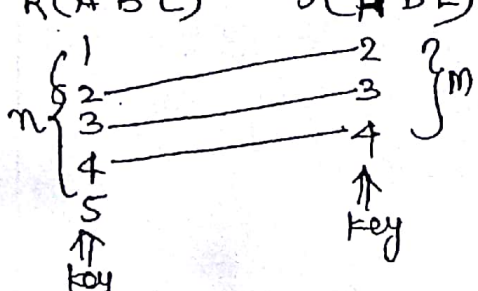


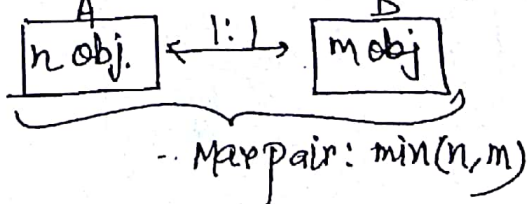
Q

	PK	# of tuples
Relation R (A B C)	A	n
Relation S (A D E)	A	m

Max tuples in R ⋈ S : min (n, m)

Min tuples in R ⋈ S : min (n, m) [with FK]
0 [No FK]

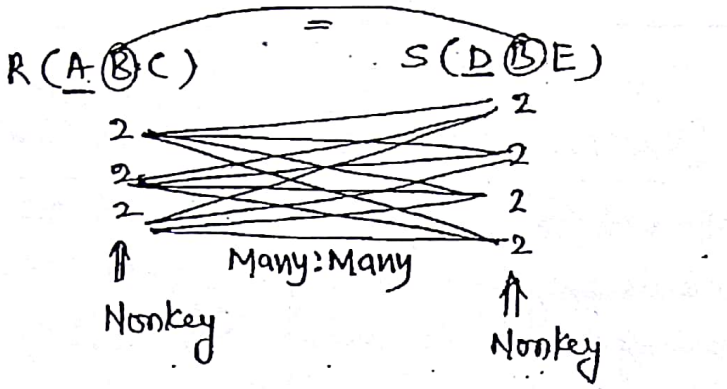


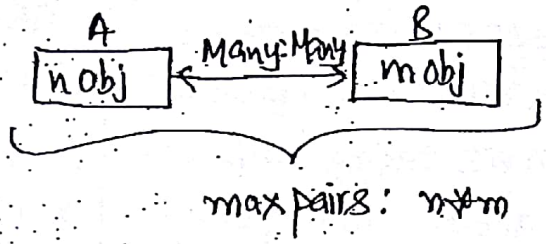


S(D B E) D: PK with m tuples

Max tuples in R⋈S : n*m

Min tuples in R⋈S : 0





SQL (STRUCTURED QUERY LANGUAGE)

SubLang of SQL

DDL [Data Definition lang.]

→ Allowed to modify structure of DBTable and Integrity Constraints

CREATE TABLE <tname>
DROP TABLE ---
ALTER TABLE ---
[Add/remove attr.]
[Modify PK, AK, FK constraints]
CREATE VIEW <Viewname>
CREATE INDEX <Attr.>

DML [Data Manipulation Lang.]

→ Allowed to modify data records only but can't modify structure of table.

INSERT INTO tname...
DELETE FROM tname...
UPDATE tname SET...
→ Access data from set of table

SELECT *
FROM R
WHERE P1

DCL [Data Control Lang.]

→ Allowed assign Access Rights to user (Security)

GRANT ...
REVOKE ...

→ Transaction Mgmt operations

COMMIT
ROLLBACK
CONSISTENCY CHECKPOINT

Relation b/w SQL and Relational Algebra

SQL Query:

Select Distinct $A_1, A_2, \ldots A_n$
FROM $R_1, R_2, \ldots R_m$
WHERE P

SELECT ⇒ Projection (π) operator
FROM ⇒ Cross Product ($\times$)
WHERE ⇒ selection (σ_p) operator

Equivalent RA Query:

$$\pi_{A_1, A_2 \ldots A_n} \left(\sigma_p (R_1 \times R_2 \ldots \times R_m) \right)$$

Q. Sid's enrolled some course taught by Korth.

RA Query:

$$\pi_{sid} \left(\sigma_{\substack{(Enroll \times Course) \\ Enroll.Cid = Course.Cid \wedge \\ Inst = Korth}} \right)$$

SQL Query : SELECT DISTINCT sid

FROM Enroll T_1, Course T_2

WHERE $T_1.cid = T_2.cid$ AND Inst = 'KORTH';

* SQL query may not result distinct sid's.

eg.

T_1

sid	cid
S_1	C_1
S_1	C_2
S_2	C_1

T_2

cid	Inst
C_1	KORTH
C_2	KORTH

RA

sid
S_1
S_2

SQL (select distinct sid)

sid
S_1
S_2

SQL (select sid)

sid
S_1
S_1
S_2

BASIC SQL CAUSES

SELECT [DISTINCT] $A_1, A_2, \ldots A_n$

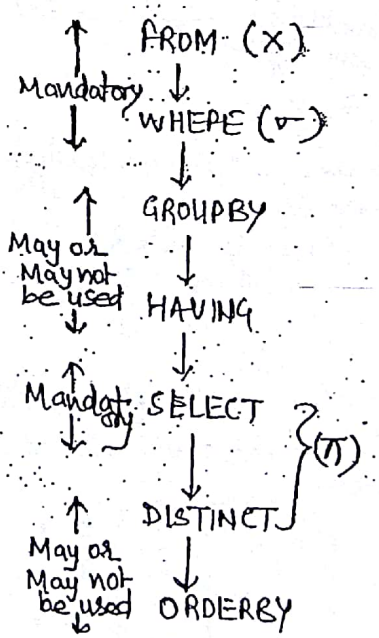
FROM $R_1, R_2, \ldots R_m$

[WHERE condition]

[GROUP BY (Attributes)

[HAVING condition]]

[ORDER BY (Attributes) [DESC]]



* The execution of query or commencement of passing starts by FROM clause always.

Aggregate Functions

count()
sum()
Avg()
Min()
Max()

[Null values not considered for any aggregation]

[Aggregate functions computes aggregation of Non Null values Only]

* Arithmetic operation with NULL results NULL,

NULL + 5 = NULL

(A)

R

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

count(*) will count it

why?

Because even if we created table as

```
CREATE table R
(A int,
 B int
);
```

But DB Manager Software will include/add one more col as RowID.

RowID	A	B
1	2	2
2	4	3
3	NULL	5
4	4	7
5	6	3
6	NULL	NULL

So, count(*) also counts

count :- # of tuples or Non Null Values

count(*) : no. of records (* = all records)

count(A) : # of Non Null Values over A.

count(DISTINCT A) : # of distinct Non Null Values of A.

eg:

```
SELECT count(*) A1
       count(A) A2
       count(DISTINCT A) A3
       count(A,B) A4
FROM   R
```

O/P:-

A1	A2	A3	A4
6	4	3	5

6 → Include NULL also.

4 → only Not Null values

Sum :-

Sum(A) : Sum of Non Null's of A

SUM(DISTINCT A) : Sum of Distinct Non Null Values of A

Avg :-

$$Avg(A) = \frac{Sum(A)}{Count(A)}$$

$$Avg(DISTINCT\ A) = \frac{sum(Distinct\ A)}{Count(Distinct\ A)}$$

Attribute A must be Int/Float

Min (A) :- Min. A value among Non Nulls

Max (A) :- Max A value among Non Nulls

A : char/Number

(*) Min(A) is same as Min(Distinct A)
Max(A) is same as Max(Distinct A)

(*) Select sum(A) A1, sum(Distinct A) A2,
Avg(A) A3, Avg(Distinct A) A4,
min(A) A5, max(A) A6,

FROM R

O/P:-

A_1	A_2	A_3	A_4	A_5	A_6
16	12	4.0 $(\frac{16}{4})$	4.0 $(\frac{12}{3})$	2	6

(*)

A	B
7	2
9	3
NULL	5

Q1: Update R set A = A+5;

Q2: select Avg(A)
from R;

If Q1 executed before Q2, what Avg value of Q2 result?

Result = $\frac{7+9}{2} = \frac{16}{2} = 8.0$

GROUP BY CLAUSE

It is used to group the records based on values of specified attribute.

GROUPBY(A): records groups based on A values
All records with same A value is one group.

GROUP BY (A, B): All records with same (A,B) values is one group.

OLAP ~~overlapped~~ or Data mining Operations

R

A	B	C
a_1	b_1	4
NULL	NULL	5
a_3	b_4	6
NULL	NULL	4
a_1	b_1	5
a_2	NULL	6
a_1	b_2	NULL

⇒

A	B	C	
a_1	b_1	4	4.5
a_1	b_1	5	
a_1	b_2	NULL	
a_3	b_4	6	6
NULL	NULL	5	4.5
NULL	NULL	4	
a_2	NULL	6	6

⇒ Select A, Avg(C)
FROM R
GROUP BY A

A	Avg(C)
a_1	4.5
a_3	6
NULL	4.5
a_2	6

4 Groups

logically data is treated as 4 groups (Null values of A forms a diff group)

⊕ Result of GroupBy clause is distinct even if DISTINCT keyword is not used along with SELECT clause.

# of records in result of SQL Query if Group By Clause Used = # of Groups

⇒ select A
FROM R
GROUP BY A

A
a_1
a_3
NULL
a_2

(4 Groups)

⊛ If Groupby Clause is applied over X set of Attribu
then in select clause, it must select all attributes of Groupby(X) and we can also use aggregate function.

1] select Avg(C)
From R
Group by A;

Error: Incorrect Usage of GroupBy Clause

"INCORRECT SYNTAX"
Here Result if come will be meaningless as we can't correlate avg values with groups of A.

Avg(C)
4.5
6
4.5
6

So, SQL returns (ERROR) for such queries.

2] select A, B
From R
Group by A;

Error: Incorrect Usage of GroupBy Clause.

"INCORRECT SYNTAX"
Because to be correct # of B values must be equal to # of groups of A. otherwise 2 values come for 1 group.

a_1	b_1/b_2
a_3	b_3
NULL	---
a_2	---

3] Select A, Count(C)
FROM R
Group by (A,B)

Error: Incorrect Usage of GroupBy Clause

"INCORRECT SYNTAX"
as No. of attr selected must be equal to No. of attr. in Group By bcoz otherwise there will be ambiguity for B value as only A record is printed.

a_1 b_1
a_1 b_1
a_1 b_2

⇒

a_1
a_1

4] Select Avg(C), A
From R

Error: Incorrect Usage of Aggr. Func.

"INCORRECT SYNTAX"
Table can't be formed as Avg(C) is only one value and A have more values

R

A	C
a_1	2
a_2	4
a_3	6

Avg(C
4

5] Select B
From R
Group by A;

Error: Incorrect Usage of Group By Clause

"Incorrect SYNTAX"
The attr in Groupby must be same as that of in select stmt.
eg. No. of Groups for A can't be same as No. of values of B

HAVING CLAUSE

→ used to select Groups based on specified condition.

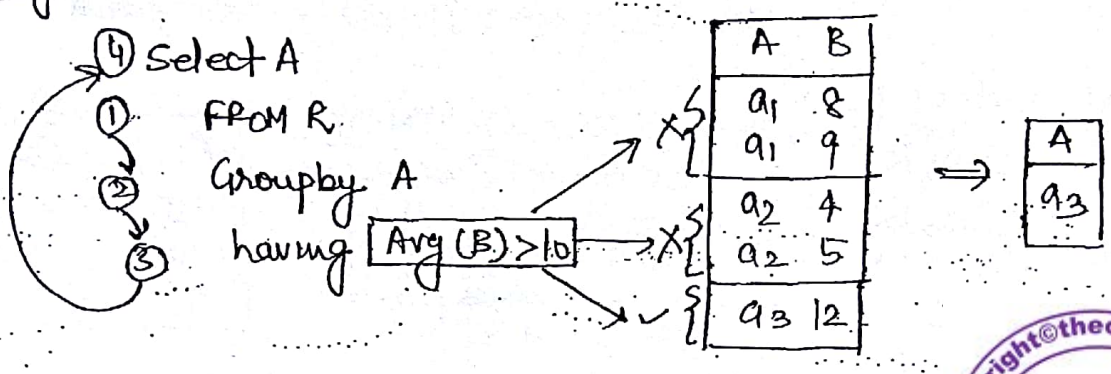
→ optional for Groupby Clause

may / may not be used but always with Groupby Clause

→ if Having used in Query,

# of records in result of Query = # of Groups which are satisfied "having" Clause condition.

eg



copyright©theorypoint.com The Best Gateway Of Success

(Ap) Having Clause Condition must be over aggregate function

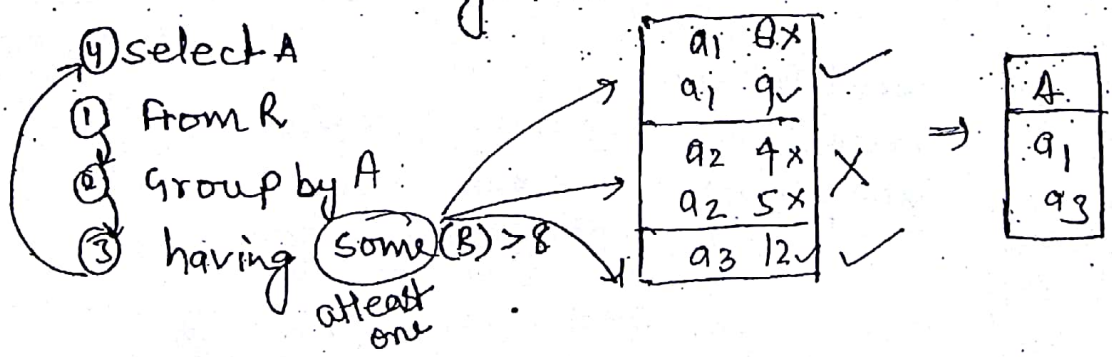
(or)

over special defined function

some()

every()

but should not used direct attribute in "having clause" condition.



Incorrect Usage of Having Clause

Having clause is meant for "Group Selection" but if implement "Record Selection" then it will not be able to retrieve correct results for groups.

✗ Error

```
Select A
FROM R
Group by A
having (B) > 8;
```

B → Direct Attribute

Record selection but Not Group Selection

SQL Standard	Oracle / MySQL
Having clause allowed to use only if Groupby Clause exist.	Having Clause allowed without Groupby also. select A from R [Groupby] Having Avg(B) > 8 If Groupby clause is not used, by default all attributes are considered for grouping. eg here A and B both are considered for grouping In that case group of each attribute will contain one record. OR Avg(B) > 8 will be applied to each record means Group by (*) is applied. So, query will become same as select A from R where B > 8
(H) Having clause is allowed to use without GroupBy Clause : T/F False	

ORDER BY CLAUSE

Used to order resulted records in Ascending / Descending order

↳ by using optional keyword like [DESC] or [ASC]

eg

```
select A, B
from R
order by A [DESC]
```

(DESC: optional)

By Default: ASC

Q: student (Sid, gender, marks, branch)

Retrieve Sids who scored highest marks?

Sol:

Using RA Logic:

```
select sid          : All Students
from student
EXCEPT
Select T1.Sid       : Students not scored HighestMarks
From Student T1, Student T2
Where T1.marks < T2.marks
```

$$\Pi_{Sid}(Student) - \Pi_{T_1.Sid}(T_1 \bowtie_{T_1.marks < T_2.marks} T_2)$$

Using Max func (SQL):

```
select sid
from student
where marks = (select max(marks)
               from student)
```

(Subquery in where clause)

Subquery → gives: marks = 90

Ⓐ Not use Aggr. func. in where cla

✗ Select sid
from Student
where marks = max(marks)
applied to each tuple

s_1	70	✓
s_2	90	✓
s_3	60	✓
s_4	90	✓

70 = max(70) ✓
90 = max(90) ✓

In same way all tuples will satisfy the condn

Ⓐ Select sid
From student T_1, (Subquery in from clause)
Subquery (select max(marks) MMarks
From Student) T_2

Performs $(T_1 \times T_2)$

where marks = mmarks

Subquery gives the table as output to calculate cross product.

T_1

sid	marks
✗ s_1	70
✓ s_2	90
✗ s_3	60
✓ s_4	90

$\times$ T_2

m marks
90

Ⓑ Select sid
✗ From Student
where (max(marks))
ERROR

because where involves any condition to be followed here we are saying where "90". which can't evaluate a condn.

For any condn. we must have
marks = max(marks)
or. marks = 90 or >90 etc.

Q. Retrieve sid's scored 2nd Highest marks.

$$\rho\left(T_1, \Pi_{Sid,\ marks}\left(Stud \bowtie_{marks<mk} \rho_{ID,GN,MK,B8}(Stud)\right)\right)$$

(All students except who scored max)

$$\rho\left(T_2, \text{"} \quad \text{"} \quad \text{"}\right)$$

$$\text{2nd Highest sid} = \Pi_{Sid}(T_1) - \Pi_{T_1.Sid}\left(\sigma_{T_1.marks<T_2.marks}(T_1 \times T_2)\right)$$

$\Pi_{Sid}(T_1)$: Sid except 1st Highest

$\Pi_{T_1.Sid}$: Sid's except 1st & 2nd Highest

But Problem here is that,

T_1 is used indirectly 3 times and in SQL, we have to write the query 3 times for T_1.

To avoid this, sql provides "WITH KeyWORD".

```
WITH T1 (sid, marks) as
    (select A.sid, A.marks
     from stud A, stud B
     where A.marks < B.marks)
select sid
FROM T1
EXCEPT
select T2.sid
FROM T1 as T2, T1 as T3
where T2.marks < T3.marks.
```

Copy of T1 is used at 3 places

lab: RENAME OPERATION in SQL

Using Aggregate Functions

```
(Select Sid
 From Student
 where marks = ((select max(marks)
                 from student
                 where marks < (select max(marks)
                                from student))
```

By this we get the 2nd Highest Marks

For Kth Highest Marks

Nested Query (K+1) Levels or (K+1) Instances of Student → It becomes very long query

Correlated Query 2 Levels, 2 Instances of Students.

(*) Q. Student (Sid, gender, marks, branch)

Retrieve Sid's scored highest marks for each branch.

Sid	marks	branch
✓ S1	90	CS
✗ S2	85	CS
✓ S3	70	CS
✓ S4	40	IT
✗ S5	70	IT

=

Sid	Marks	branch
S1	90	CS
S2	85	CS
S3	70	CS
S4	40	IT
S5	70	IT

RA:-

$$\pi_{sid}(Stud) - \pi_{sid}\left(Student \bowtie_{marks < MK \wedge branch < BR} \rho_{(ID, MK, BR)}(Student)\right)$$

SQL

```
Select Sid
from student
EXCEPT
  select T1.sid
  From  Student T1, Student T2
  where  T1.marks < T2.marks
        and T1.branch = T2.branch
```

Using Group by clause

Student

sid	marks	branch
S1	80	CS
S2	85	CS
S3	70	CS
S4	40	IT
S5	70	IT

X

Temp

branch	max(marks)
CS	85
IT	70

[Group by/Max]

Here we can't handle group by clause with only one query because we have to retrieve sid but we need to group the branch.

Therefore we use "Subqueries"

Here we make a temporary table & store branch and max marks & then match the records in both tables having marks = max marks & branch=branch.

```
Select Sid
FROM student, (select
        branch, max(marks)
        as MMarks
    from student,
    Group by Branch) temp
where student.branch =
        temp.branch
and marks = MMarks
```

Retrieve branches for which atleast three student in branch

least 2/3 : (X, σ)

ly 2/3 : "—" atleast 2/3

most 2: "—" atleast 3

```
select distinct T1.sid branch
FROM STUD T1, STUD T2, STUD T3
WHERE T1.branch = T2.branch and
      T2.branch = T3.branch and
      T1.sid <> T2.sid and
      T2.sid <> T3.sid and
      T1.sid <> T3.sid;
```

ROUP BY && HAVING

sid	branch
S1	CS
S2	CS
S3	CS
S4	IT
S5	IT

```
Select branch
from Student
Group by branch
    having count(*) >=3
```

or sid

will count the no. of students in each group.

If student table has 10 records then T1 X T2 X T3 will have 1000 records.

5 comparisons for each record

∴ total 5000 comparison

Q. Retrieve branches for which Avg marks of the branch is more than avg. marks of all female students.

Gender	Branch	Marks	All fem avg. $(\frac{160}{3})$
F	CS	60	
M	CS	80	
F	CS	40	
M	IT	80	
F	IT	60	
M	IT	90	

```
Select branch
From student
Group By Branch
having Avg(marks) > (select avg(marks)
                     From student
                     where gender=female)
```

to retrieve avg marks of all females.

Q. Retrieve branches for ~~each~~ which Avg marks of the male students of branch more than Avg marks of all females.

M	CS	80	All female Avg marks
M	IT	80	
M	IT	90	

```
select branch
from Student
where gender=male
Group By Branch
having Avg(marks)>(select
                   avg(marks)
                   From Student
                   where gender=female)
```

Q Retrieve branches where Avg marks of the male students of the branch is more than Avg marks of the female students of the same branch.

temp1

branch	avgM
CS	80
IT	85

X temp2

branch	avgF
CS	75
IT	90

```
select temp1.branch
FROM (select branch, avg(marks)
      from students
      where gender=male
      Group by branch) as temp1,
     (select branch, avg(marks)
      from students
      where gender=female
      Group by branch)
      as temp2

where  temp1.branch = temp2.branch
  and  temp1.marks > temp2.marks
```

temp1 x temp2

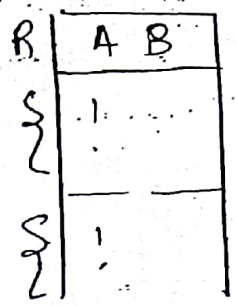
NESTED QUERIES

Query Inside Query

```
SELECT   Attributes X
FROM     tables      <-- Sub query
WHERE    condition   <--
GROUP BY (Attributes) X
HAVING   condition   <--
Order by (Attributes) X
```

can be used only in these 3 clauses.

subquery is not allowed with select, group by, and order by clause because these three clauses works on attributes.

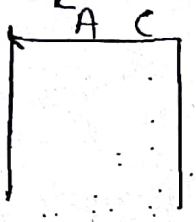


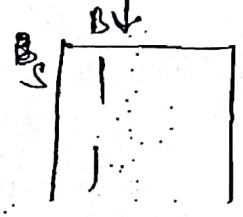
Select A
From R
Group by A

Select A
From R
Group By (Select B from S)

Group by depends on data (table) mentioned in From Clause. i.e. on R but not on S and thus grouping is being done on the basis of attributes of S which are not present in R.

(*) We also can't write group by (Select B from R) because now we have 2 instances of R table & grouping is done on the first instance.





Types of Nested Query

1) Nested Query (No Corelation)

Inner Query independent of outer Query

(Or) Outer Query uses results of Inner Query

2) Corelated Nested Query

eg

```
Select Sid
from Student
where Marks = (Select max(Marks)
               From Student);
```

(2) TOP

(1) BOTTOM

Computes once

(*) Execution flow : BOTTOM → TOP

(*) Inner Query computes only once.

Here if inner query results more than one value, then use of '=' operator will return an error because comparing one record marks with

Nested Query with Co-relation :-

Inner Query uses attributes (data) defined in outer Query

(and) Outer Query uses result of Inner Query

eg:

```
Select R.A
(1) from R
(3) where ( select count(*)
            from S
        (2) where R.A<S.B ) < 2
```

Meaning of correlation ⇒ Outer Query where clause depends on result of Inner Query & Inner Query result depend on data (attr) of outer query i.e. R.A (R's attribute)

eg

R
A
10
15
20

R.A = 10

S
B
12
14
18

R.A=10 → S

```
Select count(*)
FROM S
WHERE 10<S.B  < 2
```

(false)

for R.A = 10 this cond^n fails.

then for R.A = 15 the cond^n satisfied.

R.A = 15 → S

```
Select count(*)
from S
where 15<S.B  <2 (true)
```

R.A = 20 → S

```
Select count(*)
from S
where 20<S.B  <2 (true)
```

Differences from N-Query without correlation

(1) Inner Query depends on Outer Query

(2) Inner Query executes more than once, # of executions depends on # of ~~attributes~~ records of R.

(3) Inner Query Execution Sequence is different. "top → bottom → top"

Special functions used in Nested Queries

1. IN / NOT IN
2. ANY
3. ALL

} Best for Nested Queries with No Correlation

4. EXISTS / NOT EXISTS
5. UNIQUE / NOT UNIQUE

} Best for Nested Queries with Correlation.

(Used to compare record value of outer Query with set of values resulted by Inner Query.)

⊛ IN function (Membership test)

```
where A IN ( Select B
             from S
             where Condition)
```

True, if A value is member of Inner Query Result

eg:

```
SELECT Sid
From Enroll
Where Cid IN ( Select Cid
               from Course
               where Inst = 'KORTH' );
```

Results Cid's of courses taught by Korth

This is representing Join b/w Enroll & Course:

$\Pi_{Sid}(Enroll \bowtie \sigma_{Inst='Korth'}(Course))$

Cid IN Cid
C1
C3
C5

Results Sid of any student enrolled in some atleast one Korth Course

⊛ ANY FUNCTION

X op ANY {set of values (S)}

↳ <, >, ≤, ≥, =, <>

It returns true if X satisfied some value ~~vale~~ of set S

⊛ ALL FUNCTION

X op ALL {set of values (S)}

It returns true if X satisfied ~~all~~ ~~vale~~ given comparison (op) with every value of set S.

⊛ EXISTS FUNCTION

To test result of inner Query whether Empty/N E

EXISTS (Query)

↳ Non Empty → EXISTS returns TRUE

↳ Empty → EXISTS returns FALSE

Q. R(A...) S(B...)

Retrieve values of A from R those are bigger than atleast one B value of S.

1) $\pi_A (R \bowtie_{A>B} S)$

2) Select Distinct A
from R,S
where R.A > S.B

} JOIN Query

3) Select A
from R
where A > ANY (Select B
from S)

} Nested Query

3] Select A
1. FROM R
3. WHERE EXISTS (2. Select *
FROM S
WHERE R.A > S.B)

} Correlated Query

Here comparison is done in inner query

eg.

R

A
5
10
25

B
10
15
30

Execution :-

R.A = 5 EXISTS (Select * From S where 5 > S.B) : FALSE (5 not in result)
(empty)

R.A = 10 EXISTS (Select * From S where 10 > S.B) : FALSE (10 not in result)
(empty)

R.A = 25 EXISTS (Select * From S where 25 > S.B) : TRUE (25 in result)
(non empty)

So, Exists acts as "Atleast One".

↓

Here Inner query result is not empty set
∴ Exists returns true & 25 will be selected.

R(A...) S(B...)

Retrieve A values from R those are more than every B va
of S.

1] Join Query

$\pi_A(R) - \pi_A(R \underset{A \leq B}{\bowtie} S)$

```
select A
from R
WHERE
EXCEPT
select R.A
FROM R,S
WHERE R.A ≤ S.B
```

2] Nested Query

```
select A
from R
WHERE A > ALL ( select B
                from S)
```

3] correlated Query

```
select R.A
from R
WHERE NOT EXISTS ( select *
                   from S
                   where R.A ≤ S.B)
```

NOT (A ≤ someB)

≡ A > EveryB

Q Emp (Eid, dno, salary)

Retrieve Eid whose salary is more than any employee of dept 5.

1. Π_{Eid} (Emp $\bowtie_{Salary>SAL \wedge DN=5}$ ρ (Emp))
ID, DN, SAL

Select DISTINCT T1.Eid
From Emp T1, Emp T2
WHERE T1.salary > T2.salary
and T2.dno=5

2. Select Eid
From Emp
where Salary > ANY (Select salary
From Emp
where dno=5)

min(salary) → both are correct

3. Select T1.Eid
from Emp T1
WHERE EXISTS (Select Eid / * → Here we can use one col. or all col. as this data is not being compared.
from Emp T2
Where T1.salary > T2.salary
and T2.dno = 5)

Q Retrieve Eid's whose salary more than every employee of dept 5.

1. Select Eid
From Emp
EXCEPT
select T1.Eid
From Emp T1, Emp T2
where T1.salary ≤ T2.salary and T2.dno=5

2. select Eid
from Emp ⟹ max(salary)
where salary [>= ALL] (select salary — Both are correct
from Emp
where dno = 5)

3. select T1.Eid
from EmpT1
where (NOT EXISTS (select *
From EmpT2
where T1.salary ≤ T2.salary
and T2.dno = 5)

>Every ≤Some

Workbook
Pg 50
Q26

$$\pi_{Eid}(Emp) - \pi_{Eid}\left(E \bowtie_{E.Eid=C.SRepID \wedge C.Rating <> Good} C\right)$$

↓ All — ↓ (Some customers of Emp not good Rating)

(d) = Every Customer of Emp with good Rating

Q13

$$\pi_{Empid}(Emp) - \pi_{EmpID}\left(Emp \bowtie_{EmpID=EID \,\&\, empage \leq deptage} dept\right)$$

↑

( ) — age ≤ age of some of his/her dept

(d) greater than every age of his/her dept.

Q Book (title, price)

Assume No two books same price.

```
select T1.title
FROM BookT1
WHERE ( select Count(*)
        FROM BookT2
        WHERE T1.price < T2.price) < 3
```

Put > to get least expensive books.

↳ Put = 2 to get 3rd most expensive books.

The query results -

i) Titles of 3 most expensive books

ii) Titles of 2 most " books

iii) Titles of 3 least " books

iv) Titles of 2 least " books

---

i) if inner query results count = 0

then T1 price is most expensive

ii) count = 1

then T1 price is 2nd most expensive

iii) count = 2

then T1 price is 3rd most expensive

So outer query results titles of 3 most expensive books.

iv) for count ≥ 3

query becomes false

	T1 Title	Price		T2 Title	Price
✗	B_1	20	count=4	B_1	20
✗	B_2	30	count=3	B_2	30
✓	B_3	40	count=2	B_3	40
✓	B_4	50	1	B_4	50
✓	B_5	60	0	B_5	60

result

Sid	marks
S_1	90
S_2	90
S_3	80
S_4	80
S_5	70
S_6	60

Highest marks.

```
select T1.sid
From Student T1
where ( select count (distinct T2.marks)
        from Student T2
        where T1.marks < T2.marks
) = 2
```

Student

Sid	marks
S_1	90
S_2	90
S_3	80
S_4	80
S_5	70
S_6	60

Student

Sid	marks
S_1	90
S_2	90
S_3	80
S_4	80
S_5	70
S_6	60

Count (distinct marks)

Here 70 becomes ~~5th~~ highest.
So we have to count distinct marks.

⊛ SQL Query equal to "/" operator of RA

To retrieve sid's enrolled every course.

(I.) RA: $\Pi_{sid,cid}(Enroll) / \Pi_{cid}(Course)$

SQL:

T1 Enroll

sid	cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

Course

Tid...
C_1
C_2
C_3

T2.Enroll

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

Student S_1 in the result if he enrolled in all courses C_1, C_2, C_3

Retrieve T1.sid such that

$$[\text{All courses}] - [\text{Cid's enrolled by } T_1.sid] \equiv \phi \text{ [empty]}$$

$T_1.sid = s_1 \Rightarrow \{c_1 c_2 c_3\} - \{c_1 c_2 c_3\} = \phi$ ✓ s_1 in the result.

$T_1.sid = s_2 \Rightarrow \{c_1 c_2 c_3\} - \{c_1\} = \{c_2 c_3\}$ ✗ s_2 not in the result.

Query →

```
Select Distinct T1.sid
From Enroll T1
Where NotExists ( Select Cid                    } All courses
                  From Course
                  EXCEPT
                  select T2.Cid                 } Cid's enrolled by T1.sid
                  from Enroll T2
                  where T2.sid = T1.sid )
```

(II) $\Pi_{sid,cid}(Enroll) / \Pi_{cid}(Course)$

$= \Pi_{sid}(Enroll) - \Pi_{sid}(\Pi_{sid}(Enroll) \times \Pi_{cid}(Course) - \Pi_{sid,cid}(Enroll))$

```
select sid      — (Sid's enrolled some course)
from Enroll
EXCEPT          -
Select Sid      — (Sid's not enrolled every course)
FROM ( select E.sid, C.Cid
       FROM Enroll E, Course C
       EXCEPT
       select Sid, Cid
       from Enroll ) R
```

Sid's enrolled in all courses.

Retrieve Korth course Id enrolled by every student whose age is more than 20.

RA: (Korth Course) && course Enrolled by Every Student whose age > 20)

(Sid Cid) Sid's enrolled some Korth / Every student age > 20

$$\Rightarrow \pi_{sid\,cid}(Enroll \bowtie \underset{Inst=Korth}{\sigma}(Course)) / \pi_{sid}(\underset{age>20}{\sigma}(Student))$$

$$\rho(T_1, \pi_{Cid\,Sid}(Enroll)) / \pi_{Sid}(\underset{age>20}{\sigma}(Stud))$$

$$\pi_{Cid}(T_1 \bowtie \underset{Inst=Korth}{\sigma}(Course))$$

Sol: Retrieve T_1.cid such that

(T_1.inst = KORTH) and

$$\{[\text{All students}_{age>20}] - [\text{Sid enrolled by } T_1.\text{Cid}] = \phi\}$$

Query ⟶

Sids enrolled every course taught by Korth.

```
Select T1.cid
from course T1
where T1.Inst = KOPTH
and NOT EXISTS ( Select Sid
                 From Stu|ent
                 where age>20
                 ~~where~~ EXCEPT
                  Select sid
                  from Enroll T2
                   where T1.Cid = T2.Cid)
```

COMPARISON WITH NULL

Null is used to represent unknown / unExisted value.

CoddRule:- [RDBMS design Guideline]

Null must be Non Zero and Not empty String

Null is set of ASCII values assigned by DBMS such that No two NULLs is same in DB table.

eg R(A B C)

4 NULL 1
5 NULL 2
5 6 3

```
Select *
From R
WHERE B>=3
```

Comparison with Null is unknown (Neither true nor false)

A	B	C	
4	NULL	1	Unknown
5	NULL	2	Unknown
5	2	3	False
5	6	4	True

SQL fetch records for which condition results true in "WHERE CLAUSE".

```
select *
from R
where B>=3
```

O/P:-

A	B	C
5	6	4

False/Unknown: Record Discarded by WHERE CLAUSE

(4) Emp

Eid	Ename	PPno
E_1	A	P3
E_2	B	NULL
E_3	C	P1
E_4	D	NULL

Retrieve Eid's who don't have passport.

```
select Eid
from Emp
where (PPNo = NULL)
```

O/P:-

Eid
} Empty

Due to comparison with NULL, for all records, result will be unknown.

To avoid this problem, SQL introduced 2 clauses-

IS / IS NOT : clauses used access records based on field value NULL/NOT NULL

eg:

```
select Eid
from Emp
where PPNO IS NULL
```

O/P:-

Eid
E_2
E_4

Consider a WHERE CLAUSE

where A > 10 && B < 10

A > 10 → Unknown (if value of A is unknown)

B < 10 → True/False/Unknown (can be the possible truth values)

Now How to decide b/w Unknown & True/False/Unknown

For this, we define truth tables,

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

False → 0
true → 1
unknown → 1/2

X AND Y = min(X,Y)

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

X OR Y = max(X,Y)

eg

R

A	B	C
8	4	NULL
6	NULL	8
4	6	NULL
8	6	5

Select *
From R
where (A>5) and
(B>C OR C≥5)

How many records in result of SQL Query

a) 4 b) 2 c) 1 d) 0 e) 3

Q ⇒ Relation can contain NULL values.
Assume comparison with NULL is false.
Which of the pair of expressions not equal

a) $X = 5$ — not (not (X=5))

b) $X = 5$ — $X > 4$ and $X \leq 6$, X: integer

c) $X \neq 5$ — not (X=5)

d) None

a) $\underbrace{\overbrace{X = 5}^{\text{NULL}}}_{\text{false}}$ — not (not $\underbrace{\underbrace{(X=5)}_{\text{NULL}}}_{F}$) → $\underbrace{\quad\quad}_{T}$ → False

b) $\underbrace{\overbrace{X = 5}^{\text{NULL}}}_{F}$ — $\underbrace{\underbrace{X>4}_{\text{NULL}}}_{F}$ and $\underbrace{\underbrace{X<6}_{\text{NULL}}}_{F}$ → F

c) $\underbrace{\overbrace{X \neq 5}^{\text{NULL}}}_{F}$ — not $\underbrace{\underbrace{(X=5)}_{\text{NULL}}}_{F}$ → T

So, (C) is ans.

	Comparison with NULL = false	(SQL standard) = unknown
Q R: X: 1, NULL, 5	Q1 ✓ ✗ ✗; Q2 ✓ ✓ ✗	
Q1] Select * From R where X≠5	X: 4	X: 4
Q2] Select * From R where not(X=5)	X: 4, NULL — result is wrong as not 5 ≠ NULL	X: 4

Also both query don't give same result. To avoid this we assume comparison with NULL = unknown

Q.

R

A	B
1	4
2	2
3	5

S

C	D
5	2
3	4
8	6

for all these queries, find # of records resulted :-

i)
```
select *
from R
where A > ANY (select C
               from S
               where D>10)
```
{} // Empty

A > ANY {} → false

~~a)~~ 3
b) 2
c) 1
~~d)~~ 0

ii)
```
select *
from R
where A > ALL (select C
               from S
               where D>10)
```
{}

A > ALL {} → True

~~a)~~ 3
b) 2
c) 1
d) 0

iii)
```
select *
from R
where EXISTS (select *
              from S
              where R.A > S.C
                and S.D>10)
```
{}

EXISTS → false

a) 3
b) 2
c) 1
~~d)~~ 0

iv)
```
select *
from R
where EXISTS (select count(*)
              from S
              where R.A > S.C
                and S.D>10)
```
count
0

Not empty

EXISTS → true

~~a)~~ 3
b) 2
c) 1
d) 0

ANY

$\exists x P(x)$: True ⇒ for some X P(X) is true

False ⇒ for every X P(X) is false

ALL

$\forall x P(x)$: True ⇒ for every X. P(X) is true

False ⇒ for some X P(X) is false.

A > ANY { }
(∃)
Not true ⇒ false

A > ALL { }
(∀)
Not false ⇒ true

(A) X op ANY { } false
(∵ no one satisfies predicate)

X > ANY { } – false
X ≤ ANY { } – false

(A) X op ALL { } true
(∵ everyone satisfies pre

X ≤ ALL { } – tru
X > ALL { } – tru

(A) X IN { } – false

X NOT IN { } – true

(A) X (IN) {2, 4, 6}
= ANY
means X must be equal to 2 or 4 or 6
(or) atleast one value in
∴ we can
AN

[IN] equal to [=ANY]

④ x (NOT (IN) {2,4,6}

↳ <>ALL ↳ x is not equal to 2 & 4 & 6

(or) x is not equal to every value in set

↳ ∴ we can use ALL

[NOT IN] equal to [<>ALL]

TUPLE RELATIONAL CALCULUS

⑧ Non Procedural Query formula

⑨ uses first order logic and predicate calculus.

FIRST ORDER LOGIC

stmt either true/false but defined over set of objects using quantifiers (∀, ∃)

[Every student in class studied maths.] ... T/F

PREDICATE STATEMENT

simple stmt either True/False over single object.

[Khan studied maths.] ... T/F

⑩ P, Q Predicates

[P → Q] if P is true then q must be true

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

[P → Q] is true if

a) P is false (~P)

(OR)

b) if P is true then Q must be true

(*) $\forall x(P(x))$ is true if for every X, P(X) is true
false if for some X, P(X) is false

$\exists x(\neg P(x))$ is true if for some X, P(X) is false
false if for every X, P(X) is true

① $\forall x(P(x)) \equiv \neg \exists x(\neg P(x))$

② or $\exists x(\neg P(x)) \equiv \neg \forall x(P(x))$

for some x P(x) is false — Not every X P(X) true

③ $\exists x(P(x)) \equiv \neg \forall x(\neg P(x))$

for some x P(x) is true — Not every x P(x) is false

④ $\forall x(\neg P(x)) \equiv \neg \exists x(P(x))$

for every x P(x) is false — Not some x P(x) is true

(*) Every student in class studied maths

C(x) : x student in class

S(x) : x studied maths

First order Logic stmt: $\forall x(C(x) \rightarrow S(x))$

$\forall x(\neg C(x) \vee S(x))$

$\neg \exists x(C(x) \wedge \neg S(x))$

(*) Some student in class studied maths.

$\exists x(C(x) \wedge S(x))$ ✓

$\exists x(C(x) \rightarrow S(x))$ ✗

for this it becomes false

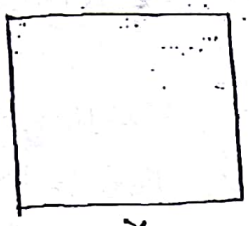
for this it becomes true — Not Studied S1, Not Studied S2 } student in class

(Not in class) Studied maths S3 } Not in class

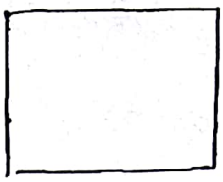
S6.

∴ false

(iv) Atleast two students in class studied Maths



X



Y

atleast one X studied Maths $\exists x (C(x) \wedge S(x))$ $\exists y (C(y) \wedge S(y))$ atleast one Y studied Maths

$x \neq y$

$$\Downarrow$$

$$\exists x \exists y \left(C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge (x \neq y) \right)$$

(or)

$$\exists (x) \left(C(x) \wedge S(x) \wedge \exists y \left(C(y) \wedge S(y) \wedge x \neq y \right) \right)$$

(v) for atleast three, we will use three variables.

(vi) only one student in class ~~student~~ studied maths.

a) (Atleast one student in class studied maths)

$\wedge$

$\neg$ (Atleast two student in class studied maths)

$(R - S \equiv R \wedge \neg S)$

$$\exists x \left(C(x) \wedge S(x) \right) \wedge$$

$$\neg \exists x \exists y \left(C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge (x \neq y) \right)$$

b) (some X student in class studied maths) $\wedge$

$$\left\{ \left(\begin{array}{c} \text{every } Y \text{ who is} \\ \text{student in class} \\ \text{studied maths} \end{array} \right) \rightarrow (x = y) \right\}$$

$$\exists x \left[(C(x) \wedge S(x)) \wedge \forall y \left(\left((C(y) \wedge S(y) \right) \rightarrow (x = y) \right) \right]$$

(All students) ∧ ¬ (Atleast one student studied maths

$\forall x\big(C(x)\big) \wedge \neg\big(\exists x \exists y\,(C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge$

$x \in$ student in class $\wedge$ Not (atleast two students studied maths

TRC Query Format

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

T: Tuple Variable

P(T): formula over tuple variable

results set of tuples (T) those are satisfied formula P(t).

eg: $\{\underbrace{S.Sid}_{T} / \underbrace{S \in Student \wedge S.age > 20}_{P(T)}\}$

To retrieve Sid's of the students whose age is more than 20.

Two types of Tuple Variables

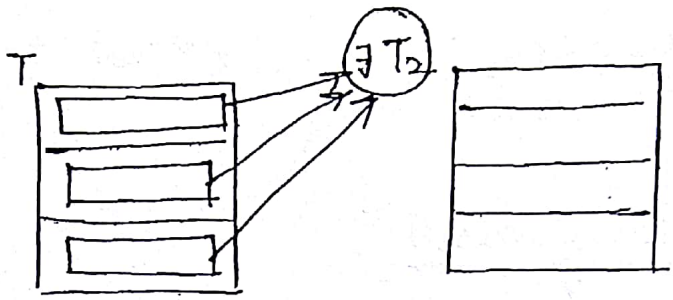
- Free Variables [Variable not defined over quantifiers]
  - eg: $T_1 \in$ Student, $T_2 \in$ Enroll — Free
- Bounded Variables [Variable defined over quantifier]
  - eg: $\exists T_3 \in$ Course, $\forall T_4 \in$ student — Bounded

Variable used for result of TRC query must be free variable.

$\{T / P(T)\}$ → free variable

$\{T \mid T \in \text{Relation} \wedge \exists T_2 \in \text{Relation}(P(T, T_2))\}$

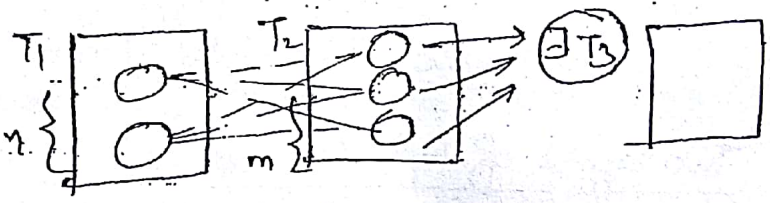
- $T \in$ Relation: Record in the relation
- $\exists T_2 \in$ Relation $(P(T, T_2))$: Atleast one record in T_2 satisfy second predicate.
- T: set of records of T that satisfies Relation with T_2



result [Records of T such that some record in T_2 satisfied $P(T_1, T_2)$]

(iii) $\exists T_2 \in$ Relation $(P(T, T_2))$ is executed (the no. of times there are records in T.) or for each record in T.

(iv) $\{(T_1, T_2) / T_1 \in Rel \wedge T_2 \in Rel \wedge \exists T_3 \in Rel (P(T_1, T_2 T_3))\}$



# of times $\exists T_3$ is tested $= n * m$.

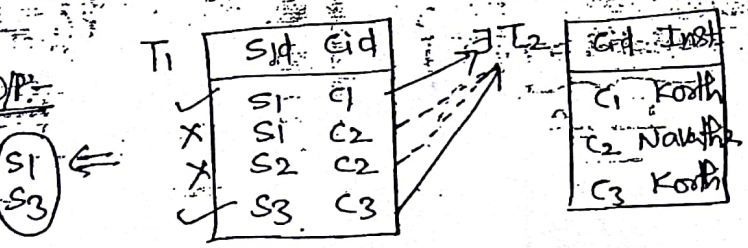
Results record pairs of T_1 and T_2 such that some record in T_3 satisfied $P(T_1 T_2 T_3)$.

$$R \times S \equiv \{(T_1, T_2) / T_1 \in R \wedge T_2 \in S\}$$

No Bounded Variables are used for Cross Product Query in TRC.

Eg: Retrive Sid's Enrolled Some (Atleast one) Course Taught by Korth.

$\{T_1.sid / T_1 \in Enroll \wedge \exists T_2 \in Course (T_2.cid = T_1.cid \wedge T_2.inst = KORTH)\}$



Using 2 Bounded Variables :-

$$\{ T / \exists T_1 \in \text{Enroll} \quad \exists T_2 \in \text{Course} \quad (T_1.\text{cid} = T_2.\text{cid} \wedge T_2.\text{Inst} = \text{KORTH} \wedge T = T_1.\text{sid}) \}$$

we used T_1 bcoz we required some free variable to get result, that's why we assigned $T_1.sid$ to T, to get result from T.

eg Retrive Sid's enrolled only courses taught by Korth.

$$\equiv \begin{bmatrix} \text{Sid Enrolled} \\ \text{some Course} \end{bmatrix} - \begin{bmatrix} \text{Sid Enrolled} \\ \text{Non Korth Courses} \end{bmatrix}$$

$\equiv$ S1, S2, S3 — S2, S1

$$\Pi_{sid}(\text{Enroll}) - \Pi_{sid}\left(\text{Enroll} \bowtie_{\text{Inst} \neq \text{Korth}} (\text{Course})\right)$$

$$\{ T_1.sid / T \in \text{Enroll} \wedge \ldots \neg \exists T_1 \in \text{Enroll} \ \exists T_2 \in \text{Course} \ (T_1.\text{cid} = T_2.\text{cid} \wedge \text{Inst} \neq \text{Korth} \wedge T.sid = T_1.sid) \}$$

$T_1.sid$ enrolled in some non Korth course

(A) Retrieve Sids enrolled only two courses.

[Sid's enrolled atleast 2 courses] − [Sid's enrolled atleast 3 courses]

$$\{T \mid \exists T_1 \in Enroll \ \exists T_2 \in Enroll (T_1.Sid = T_2.Sid \wedge T_1.Cid \neq T_2.Cid \wedge T = T_1.sid) \wedge \neg \exists T_1 \in Enroll \ \exists T_2 \in Enroll \ \exists T_3 \in Enroll ((T_1.sid = T_2.sid = T_3.sid) \wedge (T_1.cid \neq T_2.cid \neq T_3.cid) \wedge T = T_1.sid)\}$$

(A) sid's enrolled some Korth Course and some Navathe Course.

a) $\{T.sid \mid T \in Enroll \wedge \exists T_1 \in Course (T_1.cid = T.cid \wedge T_1.Inst = Korth \wedge T_2.Inst = Navathe)\}$

Result is empty

No single sid will be enrolled for same course with 2 diff. Inst.

b) [Sid Enrolled some Korth Course] ∩ [Sid Enrolled some Navathe Course]

$$\{T.sid \mid \exists T_1 \in Enroll \ \exists T_2 \in Course (T_1.cid = T_2.cid \wedge T_2.Inst = Korth \wedge T = T_1.sid)$$

$$\vee \neg \ (\vee) \ (\wedge) \ \exists T_1 \in Enroll \ \exists T_2 \in Course$$

then only Korth Course but not

then either Korth or Navathe Course

$$(T_1.cid = T_2.cid \wedge T_2.Inst = Navathe \wedge T = T_1.sid)\}$$

(A) Sids enrolled every course

RA: $\pi_{Sid,Cid}(Enroll) / \pi_{Cid}(Course)$

TRC: T_1 Student (or) Enroll

Sid
S_1
S_2

$\forall$ Course

Cid
C_1
C_2
C_3

$\exists T_2$ Enroll

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

so C_1 in result

Retrieve $T_1.sid$ such that for all courses there must be same enrollment for $T_1.sid$.

$$\{ T_1.sid \mid T_1 \in Enroll \wedge \forall C \in Course\ \exists T_2 \in Enroll\ (T_2.Cid = C.Cid \wedge T_2.sid = T_1.sid) \}$$

(A) sid's Enrolled Every course taught by Korth

$$[\{ T_1.sid \mid T_1 \in Enroll \wedge \forall C \in course\ \exists T_2 \in Enroll\ (T_2.Cid = C.Cid \wedge T_2.sid = T_1.sid \wedge C.inst = Korth)\}] \times$$

becoz it says every course is taught by Korth

but our query says other courses also exist which are not taught by Korth.

eg

sid
S_1
S_2

Cid	
C_1	Korth
C_2	Korth
C_3	Korth
C_4	Navathe

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1

Acc to above query, the result will not be S_1 because S_1 doesn't enroll in all courses. But result satisfies.

Therefore in place of $\wedge$ operator we have to use $\rightarrow$

Query becomes,

⇒ Retrieve $T_1.sid$ such that

$$\forall_{courses} (Inst = Korth \rightarrow \text{course enrolled by } T_1.sid)$$

Now if $Inst = Korth$ is false then also query returns true.

and if $Inst = Korth$ then "course enrolled by $T_1.sid$" must be true for query to produce result.

⇒

$$\{T_1.sid \mid T_1 \in Enroll \wedge \forall C \in Course \exists T_2 \in Enroll (C.Inst = Korth \rightarrow (T_2.Cid = C.Cid \wedge T_2.sid = T_1.sid))\}$$

(*) UNSAFE / SAFE TRC QUERY

$\{T \mid T \notin student\}$

$\{T \mid \neg T \in student\}$

⇒ Set of tuples T those not belong to student relation

⇒ result is Infinite set of records

Such queries are called Unsafe TRC Query ↓ which produces infinite set of records.

(*) [Safe TRC Queries Expressive Power] = [Basic RA Queries Expressive Power] ⊂ [SQL Expressive Power]

Basic RA Queries

Queries which can formulate using

$[\pi, \sigma, \times, \cup, -, \rho, \bowtie, \bowtie_c, \cap, /]$ operato

Queries Failed to express using Basic RA

- count, sum(Attr.), Avg(Attr.) only
- grouping of data records and Selection of Group (having) based on Conditions. (group by)
- ordering of resulted records of Query (order by)
- Insert / Delete / Updation of data records.

[ERMODEL]

ENTITY RELATIONSHIP DIAGRAMS

- Diagramatic Representation of DB Design
- High level DB design
- DB Design Steps :-

High Level DB Design

1. Requirements Gathering
2. Design [ER Model]
   - Tables
   - Relationship b/w Tables

Low Level DB Design

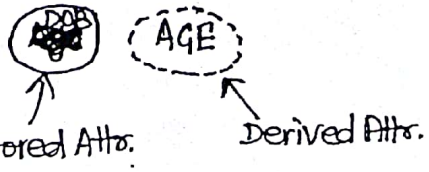
3. Convert ER Model to RDBMS Relation
   Apply Normalization
4. Implement RDBMS Relations [Create table] and Design Indexes. [Create Index]
5. Design Security [Create Views]
   [Access Rights based on end user need]

Main Components in ERD

1. Attributes : denoted by Ellipse (oval) ⬭
2. Entity Set : denoted by Rectangle ▭
3. Relationship Set : denoted by Diamond ◇

Attribute

- Key Attribute
- Multivalued Attribute
- Derived Attribute
  Value derived from other stored attribute
  (only DOB stored in table, AGE is computed whenever required)



[Compound / Complex Attr.]

Attribute can be represented as a set of two or more attr.

2 types of Design : Based on Requirement

a) Student (Sid, Sname, address) ← using only one col. name

b) Student (Sid, Sname, Hno, Street, City, S

Advantage of (b) over (a):

For the query: Retrieve Sid of students those are from city "delhi".

If user wants this specific access, then we should use (b) design to retrieve data based on street city St can be done easily.

Disadvantage of (b): More columns are required to stor the Information.

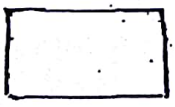
Disadvantage of (a): We are not able to retrieve inf on data based on specific access like city, sta et

Advantage of (a): Less columns are required.

Entity Set

Set of similar entities (objects)

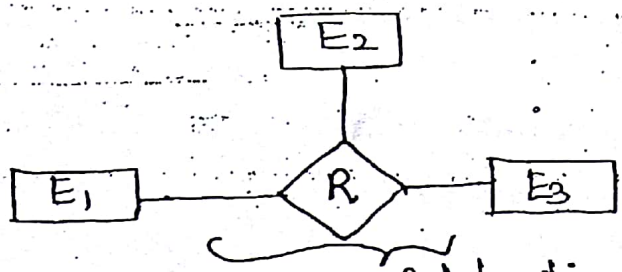
eg: student entity set consists set of students



Relationship Set

Related to 2 or more entity set.

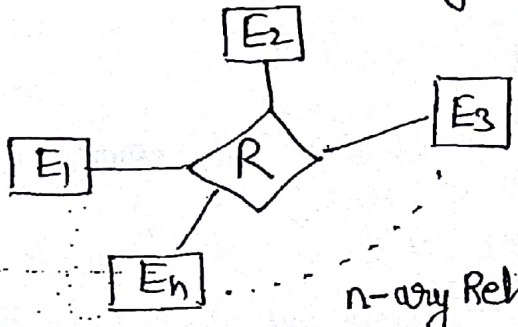
Binary Relationship



Teanary Relationship

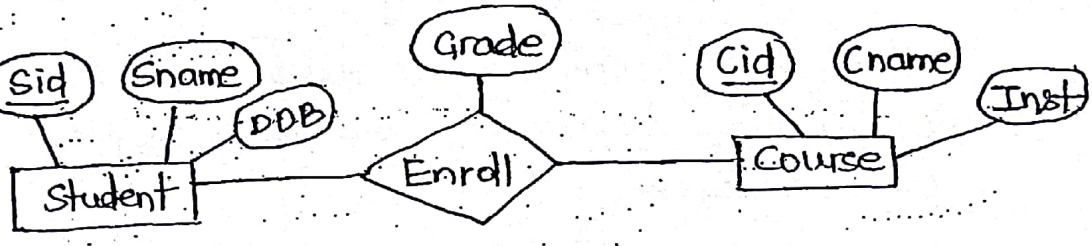
copyright©theorypoint.com The Best Gateway Of Success

R can be related to n Entity sets



n-ary Relationship

eg



Student
(Sid Sname DOB)

Enroll
(Sid Cid Grade)

Course
(Cid Cname Inst)

[Entity set]

Primary key

[Relationship set]

Foreign keys must required for Relationship.

PK = Combination of keys of Entity set it is relating to.

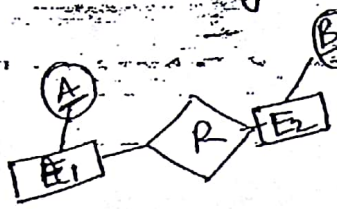
[Entity set]

Primary key

"Grade" Attribute is not only related to student only or course only but to both.

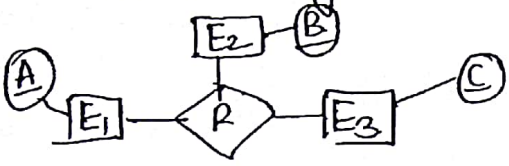
Relationship can also have attributes of its own (Grade) and should have keys of the entity set it is relating to.

For Binary Relationship,



- 2 foreign keys
- PK of R = (AB)

For ternary Relationship,



- 3 foreign keys → refer to E_1, refer to E_2, refer to E_3
- PK of R : (ABC)

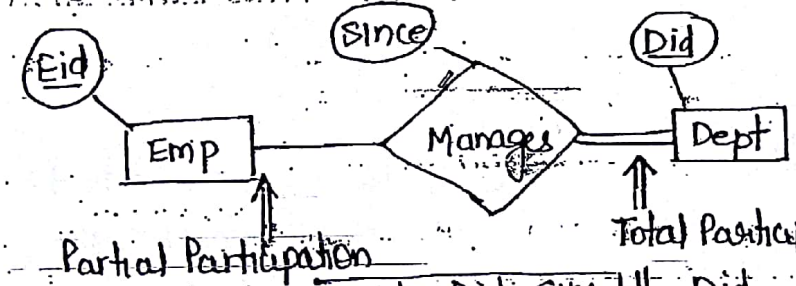
CONSTRAINTS OF Relationship Set

1) PARTICIPATION
2) MAPPING [CARDINALITY]

PARTICIPATION

If every entity of entity set must be related to relationship then called Total Participation

otherwise Partial Participation



Eid	Did	Since
E_1	d_1	2008
E_1	d_2	2010
E_2	d_3	2014
E_2	d_4	2010

E_3, E_4 ·

Some Employees may / may not belong to every dept.

Every Department must have a manager

Did: d_1, d_2, d_3, d_4

Design Constraint

- Every dept must have a manager
- There is no possibility of dept. in Dept table which is not related by manager table.

Mapping Constraint

One : entity of E_1 can relate atmost one entity of E_2 (⟶)

Many : entity of E_1 can relate 0 or more entities of E_2 (—)

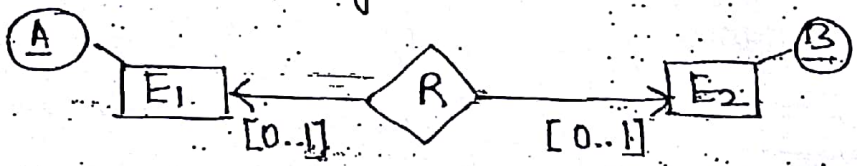
Binary "Relationship" Mapping

1) One to One (1:1)
2) One to Many (1:M)
3) Many to One (M:1)
4) Many to Many (M:N)

*** Candidate keys of Relationship set is based on Mapping Constraints between Entity sets.

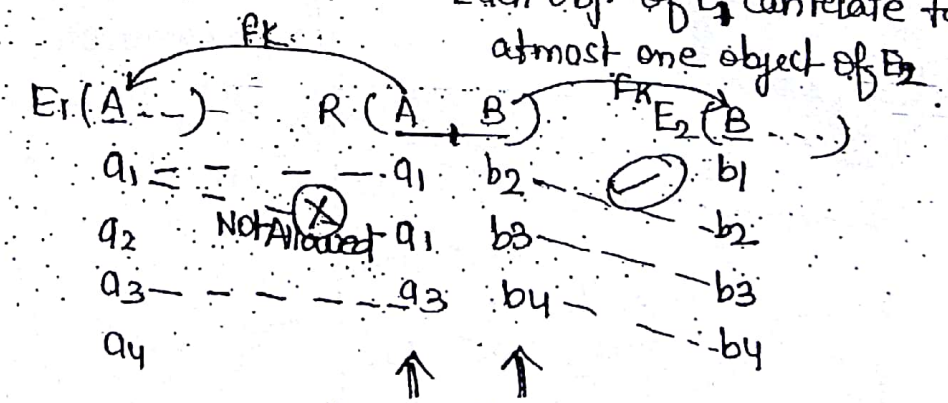
R to L (←) L to R Mapping (→)

(One) to (One) Mapping



Each obj. of E_1 can relate to atmost one object of E_2

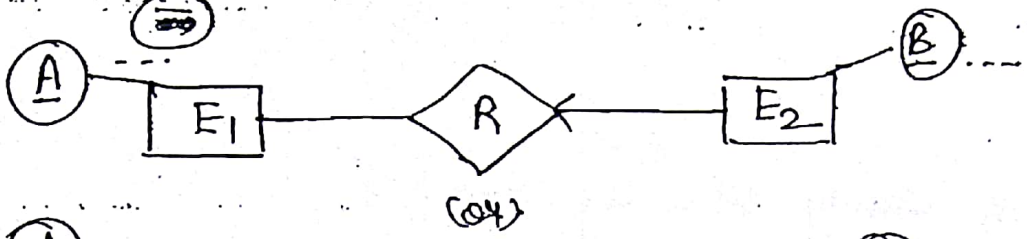
FK is same for every Mapping but Cand key could change.

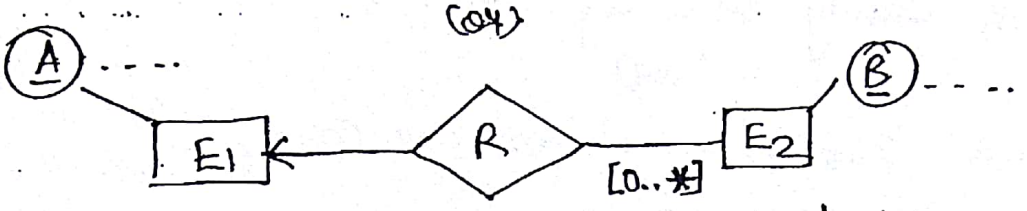


Cand. key of R(A,B) is {A, B} to represent 1:1 relationship b/w E1 & E2

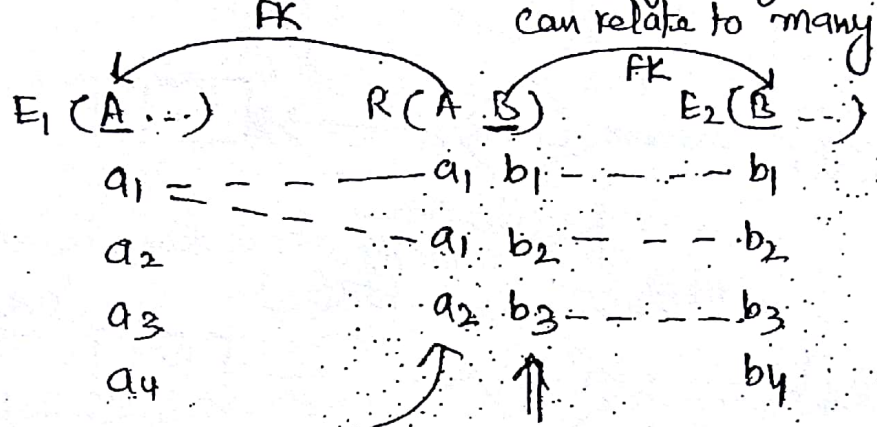
One to Many Mapping

R→L (←) L→R (→)





Each object of E1 can relate to many objects of E2



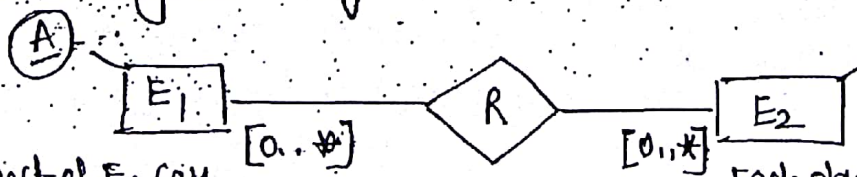
for two records 'a' can be same or one value of a can relate to many values of b

Unique (for no two records b is same)

Cand key of R(A,B) is {B} to represent 1:M relationship b/w E1 & E2

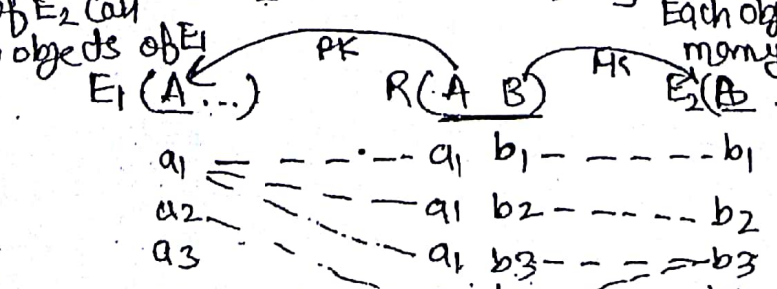
Many to One Mapping (is just same as One to Many but reverse)

Many to Many Mapping



Each object of E2 can relate many objects of E1

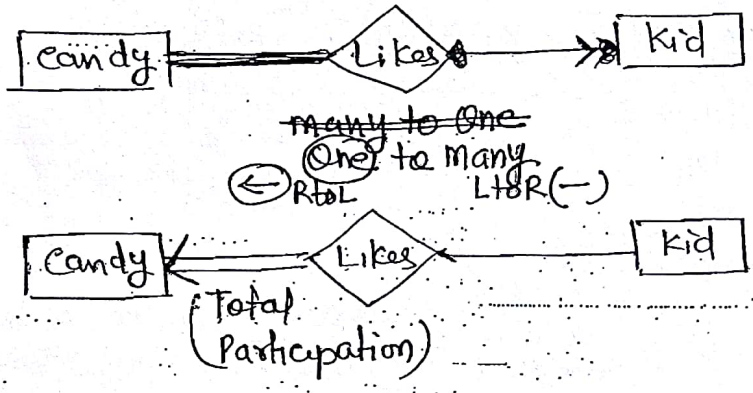
Each object of E1 can relate many objects of E2



Cand key of R(A,B) is {AB}
to represent M:N relationship
b/w E1 & E2.

Q.

Each candy likes by some kid
(and)
Each kid likes at most one candy
Represent in ERD?



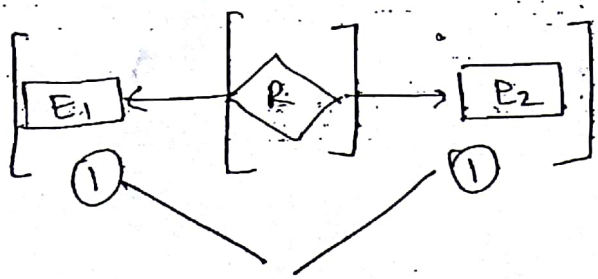
1:M Mapping &
participation towards Candy (Total)
" " Kid (Partial)

Note:

(P) only in Many to Many Mapping it is required to maintain seperate RDBMS table for relationship set.

(M) One to Many
Many to One
One to One
} Relationship sets can combine into Entity Sets

One to One

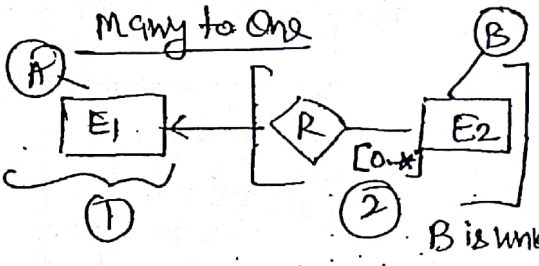


both are possible

either merge E_1 & R to store in DB & E_2 is maintained seperately

Or E_2 & R merge with E_1 seperately

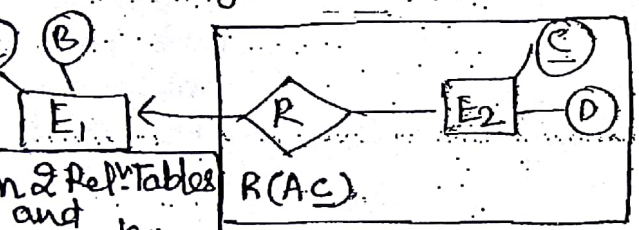
Many to One



B is unique so, B merge with R

Minimum Relational Tables for Given ERD

1) 1:M Relationship b/w E_1 & E_2 with single valued attributes only



2 Rel. Tables and Foreign key

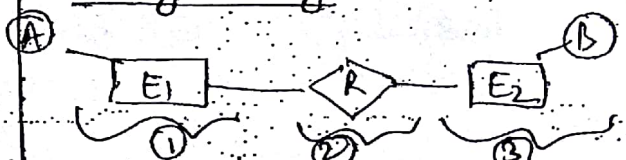
$E_1(\underline{A}\ B)$ $E_2(\underline{C}\ D\ A)$

Here 1 FK is maintained as $E_1(\underline{A}\ B)$ $R(A\ C)$

But the second FK merges (C of R and C of E_2 merges to one col.)

∴ only 1 FK remains.

Many to Many



A is not unique, B is not unique

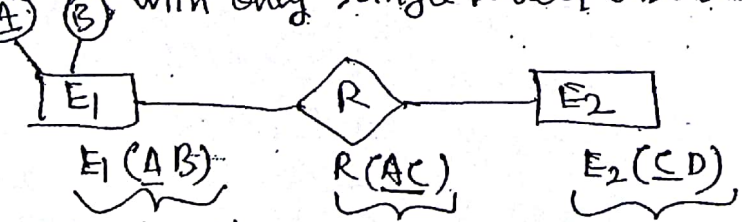
So, Seperate table for E_1

" " for E_2

" " for R

total tables = 3

b) M:N Relationship b/w E_1 & E_2 with only single valued attributes

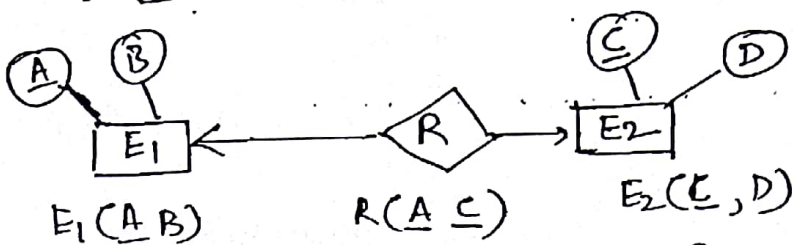


$E_1(\underline{A}\ B)$ $R(\underline{A}\ \underline{C})$ $E_2(\underline{C}\ D)$

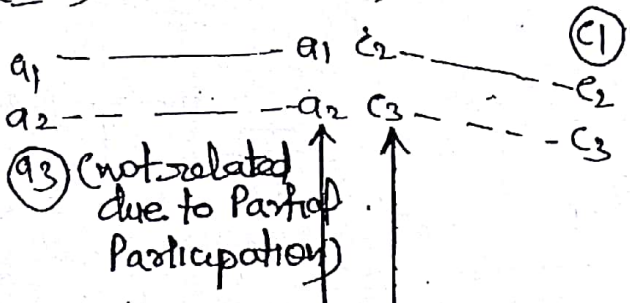
Min 3 Rel. Tables & 2 Foreign keys

c) 1:1 relationship b/w E_1 & E_2 with single valued attributes only.

i) Partial Participation at both ends



$E_1(\underline{A}\ B)$ $R(\underline{A}\ \underline{C})$ $E_2(\underline{C}, D)$



Possibility (I): $E_1R(\underline{A}\ B\ \underline{C})$ $E_2(\underline{C}\ D)$

A	B	C
a_1		c_2
a_2		c_3
a_3		NULL

C	D
c_1	d_1
c_2	d_2
c_3	d_1

⇒ Can't be merged further because in the same table A & C will become 2 diff. cand. keys.

P.K. (as unique & not NULL)

Alternative Key

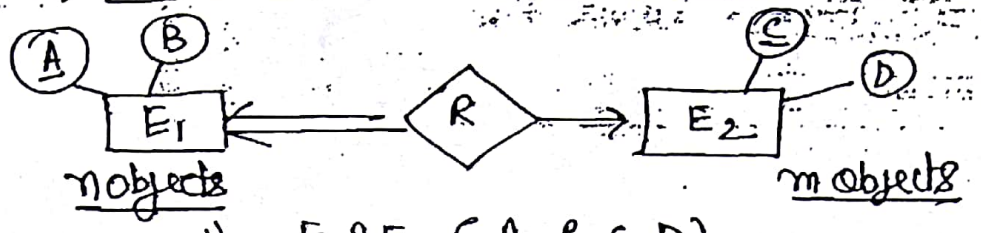
Possibility (II): $E_1(A\ B)$ $RE_2(A\ C\ D)$

Problem

A	B	C	D
a_1	–	c_2	
a_2	–	c_3	
a_3	–	NULL	NULL
NULL	NULL	c_1	

{A, C} Cand. key but no cand key with field values NOT NULL.

∴ Minimum 2 Relational Tables & 1 Foreign key



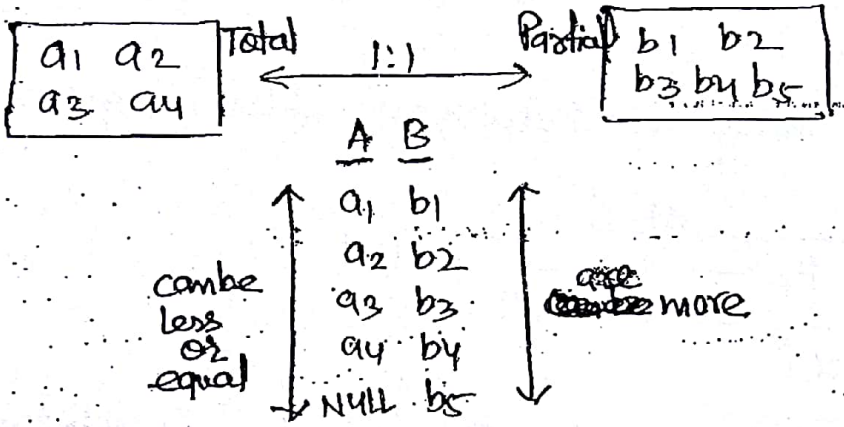
1) $E_1 R E_2$ (A B C D)

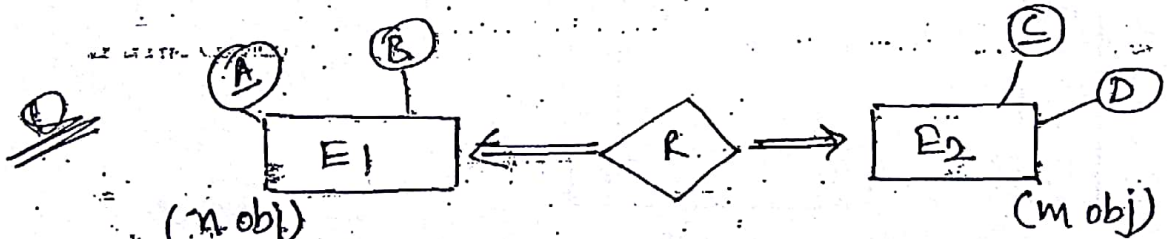
Minimum | rel. table with no FK

Alternative Key (allowed NULL)

Primary key

2) $n \leq m$





RDBMS table for ERD

$[n = m]$

$E_1 R E_2$ (A B C D)

A: CK (Not Null)

C: CK (Not NULL)

Create table $E_1 R E_2$

(A char(10) Primary key,

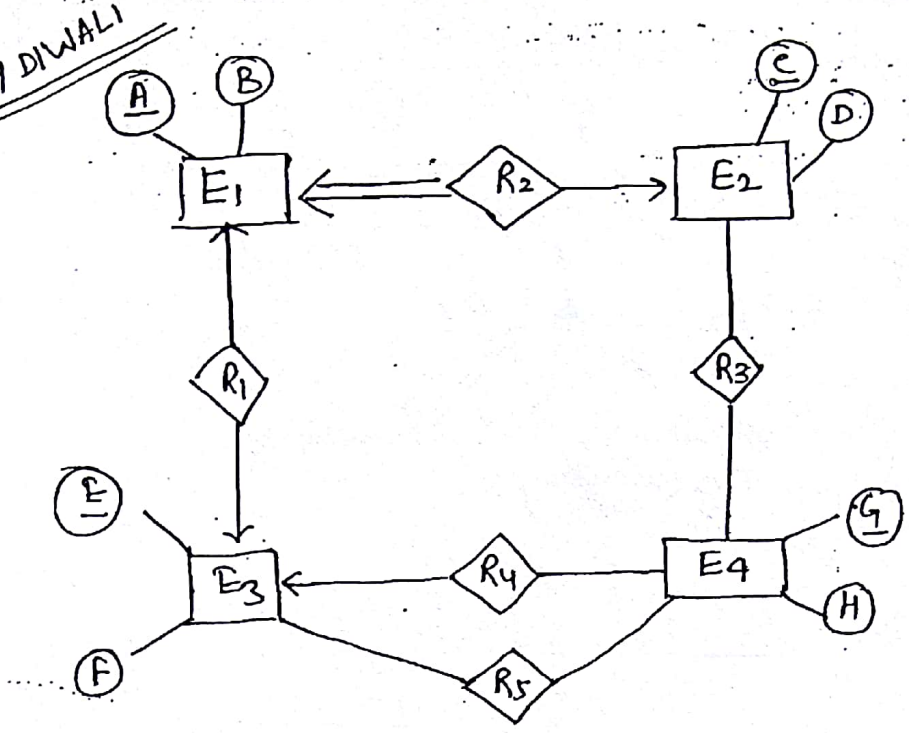
B char(3)

C char(10) UNIQUE NOTNULL,

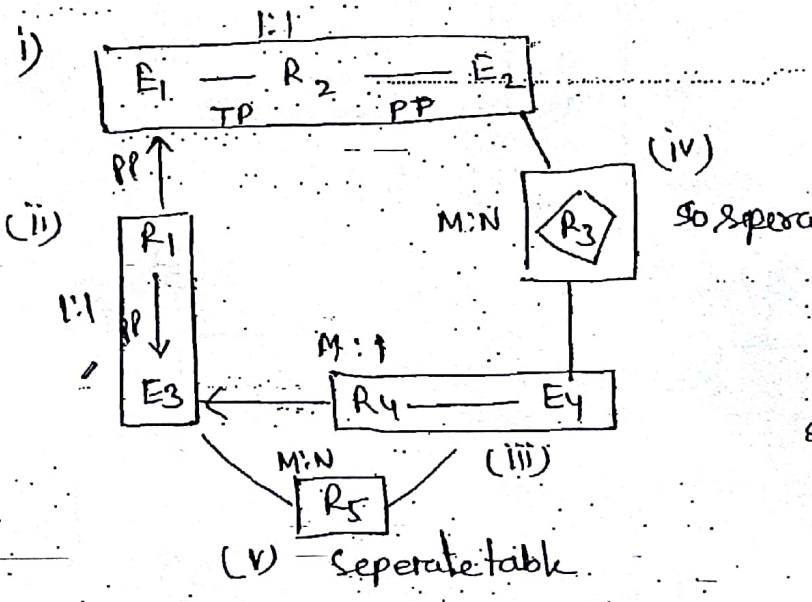
D char(3)

);

HAPPY DIWALI



1) How many min Relational tables for given ERD?



So, 5 tables.

2) How many foriegn keys required for given ERD, RDBMS design?

E1 R E2 (A B C D)

R1 E3 (E F A)

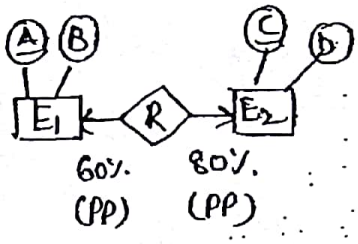
R4 E4 (G H E)

R3 (C G) #FKs=6

5) Total # of ... 4+3+3+2+2 = 14.

Q E1 & E2 are entity sets
R is relationship b/w E1 & E2 with 1:1 Mapping
80% of E2 Entities related &
60% of E1 Entities related with relationship set
which is best possible RDBMS design?

a) E1 & E2 kept seperate with FK at E1:



b)

c)

d)

The (d) option is not possible.
as (d) is $E_1RE_2(\underline{A}\ B\ \underline{C}\ D)$

The (c) option is not efficient because there is no need to have R merging both sides:

$E_1R(\underline{A}B\underline{C})$ $E_2R(\underline{C}D\underline{A})$

The option (a) says

$E_1R(\underline{A}\ B\ \underline{C})$ $E_2(\underline{C}\ D)$ (FK: C, PK: C)

A	C
a_1	c_1
a_{10}	c_5
a_{11}	NULL
	NULL

40%

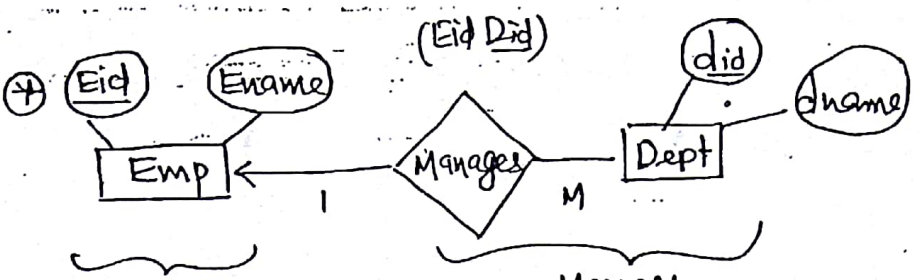
Here A is participating 60%. So, 40% C values are NULL

But in option (b): $E_1(\underline{A}\ B)$ $E_2R(\underline{C}\ D\ \underline{A})$

C	A
c_1	a_1
c_3	NULL
	NULL

20%

Less Null values are in option (b) so, (b) is correct or more preferred than (a). because 80% participation of A & C



Meages

1:M Relationship combined towards right side

Emp

Eid	Ename
E_1	A
E_2	B
E_3	B
E_4	C

Dept Manages

did	dname	Eid
d_1	X	E_1
d_2	X	E_1
d_3	Y	E_2
d_4	Y	Null

allowed NULL → partial participation b/w Manages & dept.

Eid → Ename

did → dname Eid

Here we can say that,

i) 1:M relationship satisfied

ii) No Data Redundancy

Eid → Ename (SK)

did → dname Eid (SK)

iii) Partial participation b/w manages & dept is satisfied.

1:M Relationship combined towards left (one) side

Emp Manages (Eid Ename did) PK

not SK: Eid → Ename

Did → Eid

E_1	A	d_1
E_1	A	d_2
E_2	B	d_3

Data Redundancy occurs due to Eid → Ename (not SK)

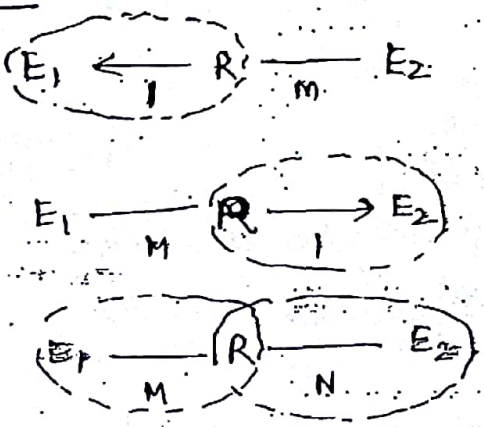
Also, partial participation between Emp & Manages is failed.

Not possible to insert emp. records who are not managers (for this did should be NULL which will be wrong)

* In this case "did" is cand key but not Eid because in 1:M mapping E_1 can manage d_1, d_2, d_3... etc Thus in the table E_i will be repeated for each dept.

∴ did is the candidate key.

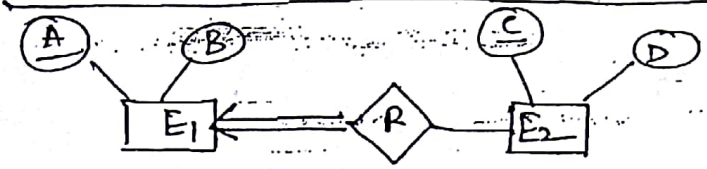
WRONG DESIGNS



These all are wrong designs because it leads to data redundancy & failed partial participation.

(*) Based on Participation & NF, the design can be changed.

Q.



a) How many Relational tables for ERD which satisfy 1NF?

a) 1 b) 2 c) 3 d) 4

Here, it is right to combine R and E_2.

But here E_1 can also be merged into RE_2 because the data redundancy is allowed in INF and ~~best~~ there is total participation from E_1 to R.

So, $E_1 R E_2$ (A B $\underset{C.K.}{\subseteq}$ D)

$C \rightarrow DA$

$A \rightarrow B$: data Redundancy (but redundancy is allowed in INF)

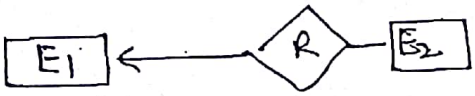
Here partial participation can't be failed as in ERD, we are given total participation b/w E_1 & R.

∴ only 1 table ∴ (a).

(*) For 2NF also, 1 table is required for same ERD. as redundancy is allowed.

However for BCNF, 2 tables will be required.

Q.



a) Min relation which satisfy INF?

Now here, partial participation can be failed if we merged as

$E_1 R E_2$

therefore we require here two tables.

E_1 $E_2 R$ → because merging with E_1 & R cause failure of partial participation.

Redundancy is allowed here as INF.

∴ 2 relational tables

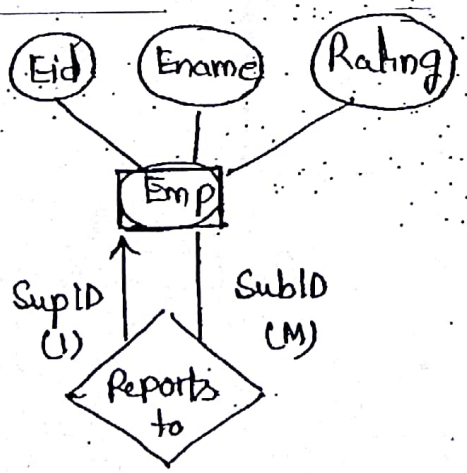
Self Referential Relationship :

Entity of entity set related to some other entity of the same entity set.

} Recursive entity set

} self - referential

eg.



Each supervisor supervises many subordinates and each subordinate can report to only one supervisor.

Emp (Eid, Ename, Rating)

Reports (SupID, SubID)

E_1	E_2
E_1	E_3
E_2	E_4

Here SubID is the key because SubID is towards the "many" end.
OR
Each SubID can have only one supervisor, so SubIDs will not repeat.

EmpReports to (Eid Ename Rating SupID)

E_1	—	—	Null
E_2	—	—	E_1
E_3	—	—	E_1
E_4	—	—	E_2

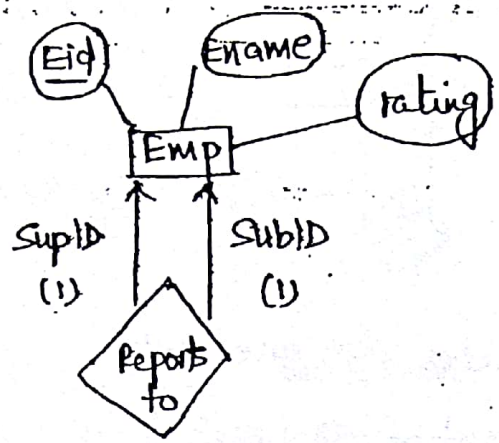
Here Eid will act as both Eid and SubID.

(each subordinate can have only one SupID)
but ~~each~~ supervisor can have many subordinates
where SubIDs are unique and thus SubID or Eid will be the cand-key, thus we can combine SubID & Eid and SupID will be kept seperate.

Therefore,

1 Relational table for ERD and 1 foriegn key.

Now, what will be RDBMS table for given ERD?



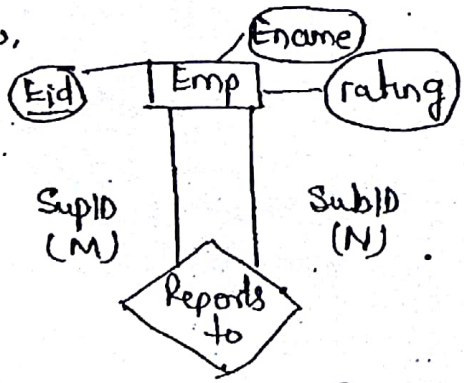
Here we can combine for only one side.

Either we can combine SupID with Eid and keep SubID seperate or can combine SubID with Eid & keep SupID seperate.

EmpReportsTo

Eid	Ename	Rating	SupID
E_1	–	–	NULL
E_2	–	–	E_1
E_3	–	–	E_2
E_4	–	–	E_3

Now,



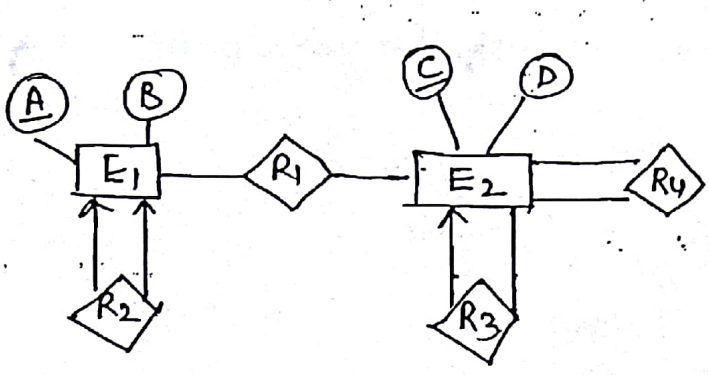
Herewe can't combine at both the table as repetitions will come.

So, Emp (Eid, Ename, Rating)

Reports to (SupID, SubID)

So, minimum 2 relational tables and 2 foreign keys.

Q:



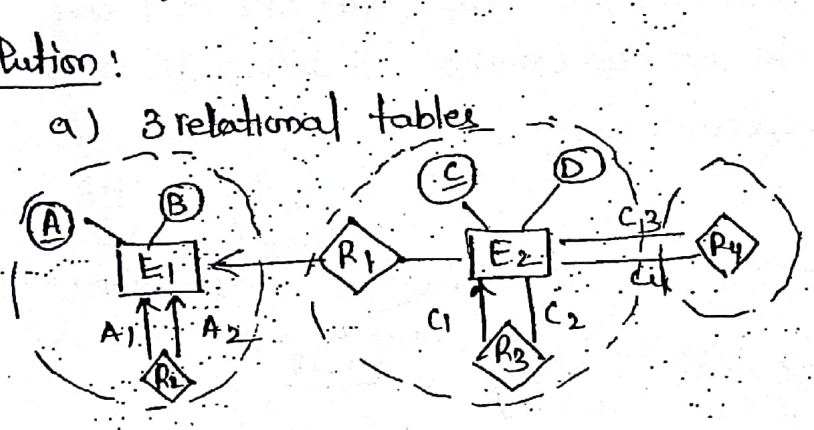
a) Min Relational tables for given ERD

b) # FK's in minimization of ERD?

c) # of total attributes in minimized design?

Solution:

a) 3 relational tables →



b) $E_1 R_2 (\underline{A}\ B\ A1)$

$E_2 R_1 R_3 (\underline{C}\ D\ A\ C1)$

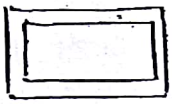
$R_4 (\underline{C3}\ C4)$

FKs = 5

c) # Attributes = 3+4+2 = 9

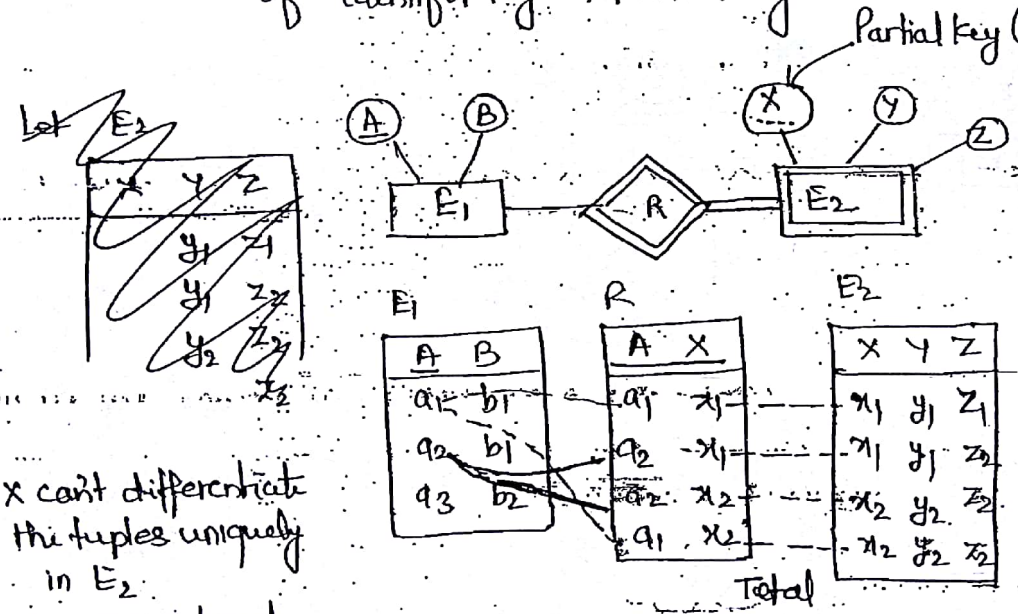
Weak Entity Set

Entity set with no key - [Attributes of weak entity set not sufficient to differentiate entities uniquely].

Denoted by 

Weak Entity set must have identifier entity set which is also called owner entity set.

Weak Entities uniquely indentified by combination of identifier key & Partial key.



Let E_2

X	Y	Z
	y_1	z_1
	y_1	z_2
	y_2	z_2

z_2

X can't differentiate the tuples uniquely in E_2 when X combined with A then it can uniquely determine values in E_2 so, X => Partial key

E_1

A	B
a_1	b_1
a_2	b_1
a_3	b_2

R

A	X
a_1	x_1
a_2	x_1
a_2	x_2
a_1	x_2

E_2

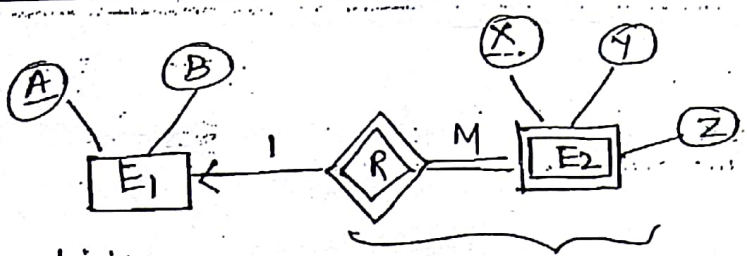
X	Y	Z
x_1	y_1	z_1
x_1	y_1	z_2
x_2	y_2	z_2
x_2	y_2	z_2

Total

(*) Relationship Set also weak becoz not related identifier weak entity properly.

(*) Here in relation every value of X in E_2 must be related to any value of A to uniquely determine its own tuple.
Also, E_2 is dependent on E_1.
∴ There is total participation of E_2

(*) 1:M Mapping b/w identifier & weak entity set.
because every weak entity set has only one owner set but each owner set can have more than one weak entity set.



Minimum 2 Relational tables for given ERD

After combining on Many side.

E_1 (A B)

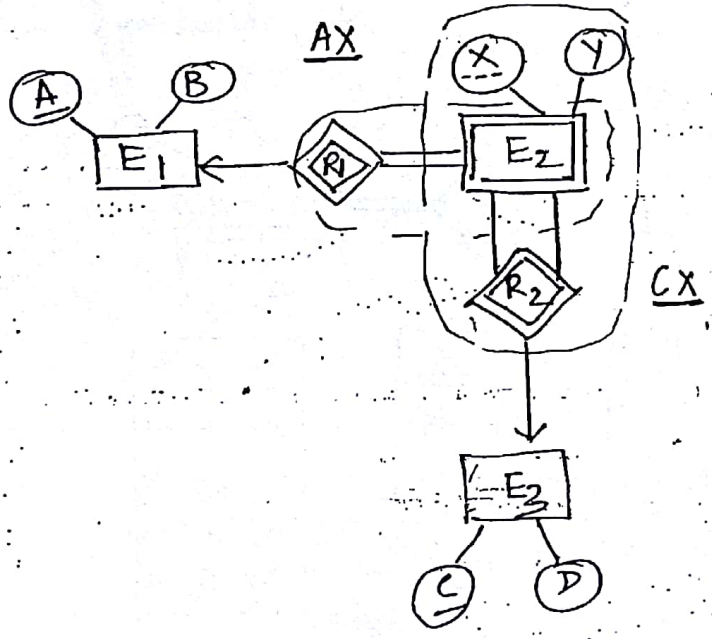
E_2R (A X Y Z)

a_1 x_1 y_1 z_1

a_2 x_1 y_1 z_2

④ Multivalued Attributes & Weak Entity set allowed to represent in ERD but not allowed in RDBMS.

Q



Min. Relational tables for given ERD?

Here we can't combine as

R_1: E_2R_2 (AX CY)

Two unique cand. keys which are overlapping

as AX data is depending on E_1 & CX data is depending on E_2] both keys will contain NULL

so, we can't merge both the relations R_1, R_2

but in any relation both cand keys are not allowed to have NULL.

Atleast one should be NOT NULL.

So, 4 relational tables required.

Ans, Now we can store R_1E_2 and E_2R_2 seperately and E_1, E_3 are stored seperately.

E_1 (A B)

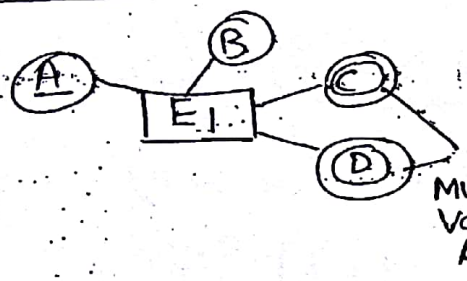
R_1E_2 (A X Y)

R_2E_2 (A X Y)

E_3 (C D)

2 foreign keys

Multivalued Attributes in ERD



{ A → B
A →→ C
A →→ D }

i) Min ~~Rel tables~~ for 1NF?

1 Relational Table E_1 (A C D B)

Proper subset of SK

A → B Non Prime ← Partial Dependency

A →→ C ✓

A →→ D ✓

So, not 2NF

ii) Min Rel tables for 2NF?

2 Relational tables { A B : SK A → B ; A C D : SK (A →→ C, A →→ D)

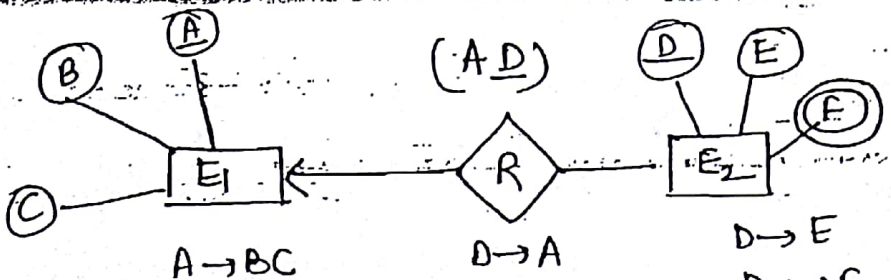
2NF ✓
3NF ✓
BCNF ✓

Not in 4NF because 2 Multivalued Attr. in one table

iii) Min Rel tables for 4NF?

3 Relational tables { A B : A → B ; A C : A →→ C ; A D : A →→ D

Q



Min Rel for 1NF (2 Relational Tables)

E_1 (A B C) RE_2 (A D E F)

A → BC

Cand key : DF

D → EA ← Partial Dependency

D →→ F

1NF ✓

Not in 2NF ← due to

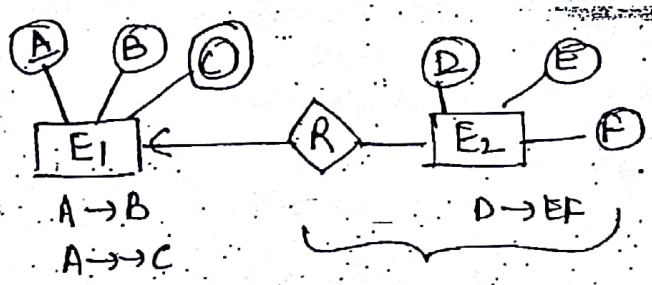
Min Rel for 2NF (3 Relational tables)

E_1 [A B C] [D E A] [D F]

A → BC D → EA D →→ F

2NF ✓
3NF ✓
BCNF ✓
4NF ✓

Q



If we combined RE_2 then Foriegn key constraint is failed as FK should refer to cand key or alternative key but A is not a Cand key in E_1.

failed

E_1 (A C B) RE_2 (D E F A)

Thus we have to maintain a table having A as a candkey

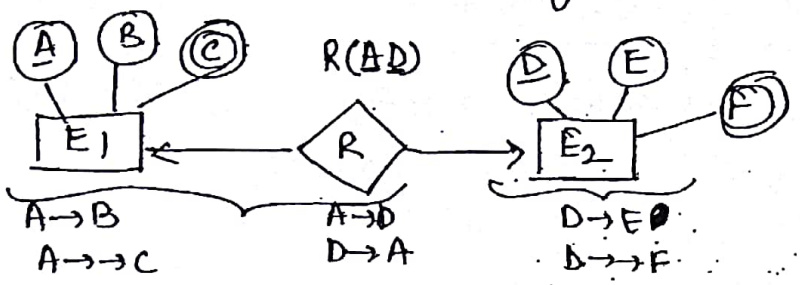
E_{11} (A B) — A → B

E_{12} (A C) — A →→ C

RE_2 (D E F A) — D → EFA

3 Relational tables for INF.

Q



Here because of 1:1 we can combine R towards left or right.

Let R is combined on left side

E_1 R(A B C D) — FK — // failed — E_2 (D E F)

A → BD
D → A
A →→ C

D → E
D →→ F

Candkey (DF)

Here we need seperate tables

So,

3 Relations for INF

E_1 R(A B C D)
Candkey (AC, DD)
as C is MV Attr.

E_{21} (D E) — D → E.

E_{22} (D F) — D →→ F

then
A → B
is partial dependency so not in 2NF

Split on basis of closure
$A^+ = \{ABD\}$

ABD — A → BD, D → A

AC — A →→ C

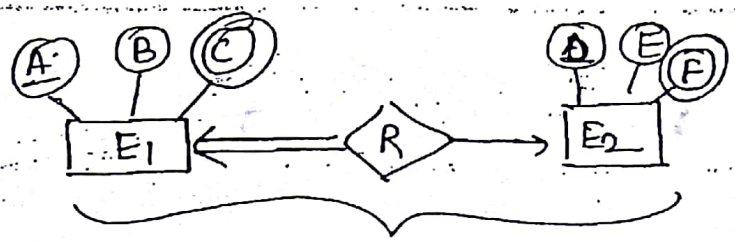
4 Relations for 2NF

DE — D → E

DF — D →→ f

Highest NF: 4NF

Q



Since 1:1
& TP.

(No requirement of FK)

Min Rel Tables for 1NF

E_1RE_2 (A B C D E F) (1 Rel. Tables)

$A \to B$ $\quad D \to E$ $\quad D \to\to F$

$A \to\to C$ $\quad D \to A$

$A \to D$

Cand keys: {ACF, DCF}

Because $A \to B$
$D \to E$ are Partial dependencies
So, Not in 2NF

Min Rel Tables for 2NF

Partial Dependency Determinant Closure is calculated.

$A+ = ABDE$

So, ABDE (2 Rel. Tables) DCF

ABDE: $A \to BD$, $D \to EA$

DCF: $D \to F$, $D \to\to C$

Highest NF
BCNF

PK: D (as total participation towards A)
AK: A

$D \to A$, $A \to\to C$
So, $D \to\to C$

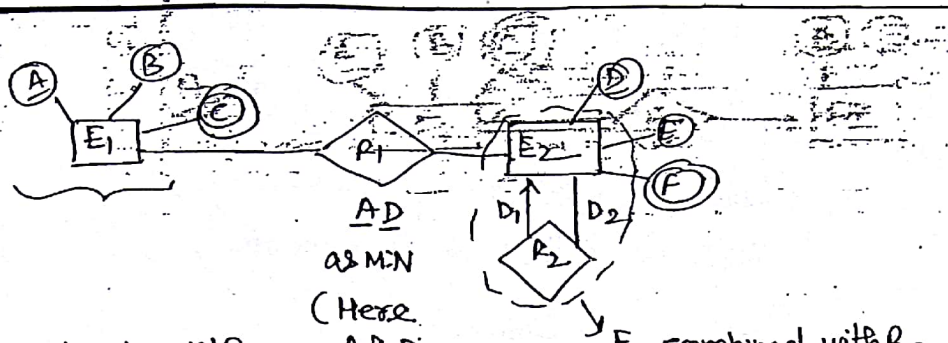
Not in 4NF

for 4NF split

DCF → DC, DF

4NF: 3 Rel Tables

Q

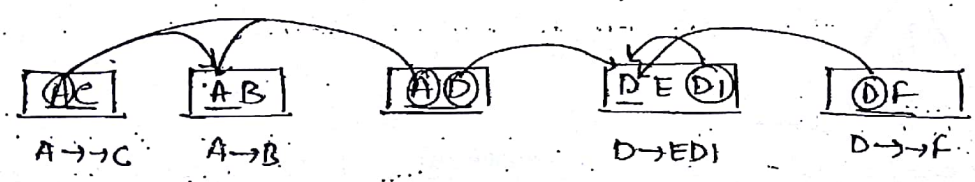


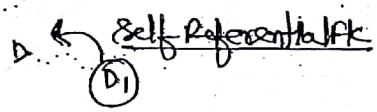
AD
as M:N

(Here A & D must be FK to E1 & E2 but in E1 AC is cand key and in E2, DF is cand key
∴ we split for AC and DF

E_2 combined with R_2

Min Rel for INF





4NF

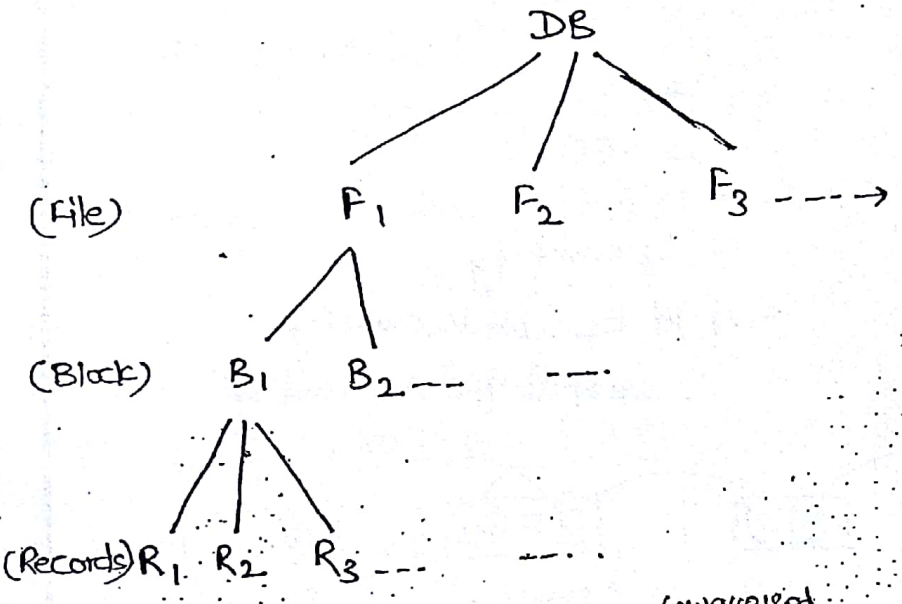
5 Min Relations
5 FKs

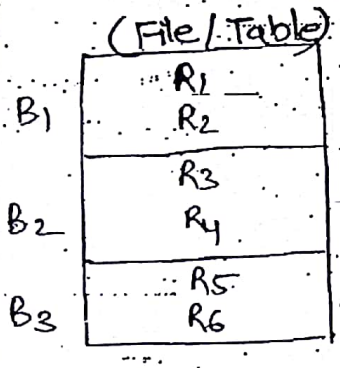
INDEXING & FILE ORGANISATION

DB is collection of files (table)

Table is collection of Blocks (Pages)

Block is collection of Records.





(or) accessed

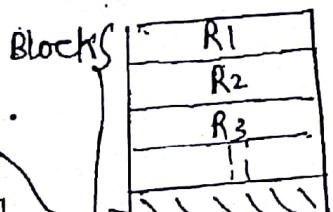
Data is transferred from secondary memory to main memory in terms of blocks.

Records in Block

Fixed Length Records

```
Create table R
( A char(100)
  B char(150)
  C char(50));
```

Record Size = 300 Bytes



Block header

[offset of table to ...]

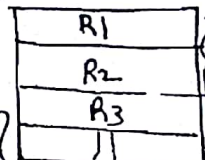
Variable length Records

```
Create tables
( D char(100),
  E char(100),
  F text);
```

record1: 100+100+200 (400B)

record2: 100+100+50 (250B)

record3: 100+100+150 (350B)



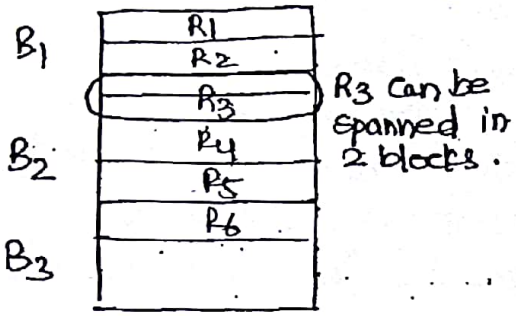
Spanned Organisation

Record can be spanned in more than one block.

Block Size: 1000B

Record Size: 400B

(Assume header is negligible)



(+) Advantage

Allocation of record without any space or memory hole in between them.

→ Allocation of file without Internal fragmentation is possible.

(-) Disadvantage - More access Cost

(As to access record R3 whole 2 block accesses will be required)

(*) Blockfactor $= \frac{B-H}{R}$

$= \frac{1000-0}{400}$

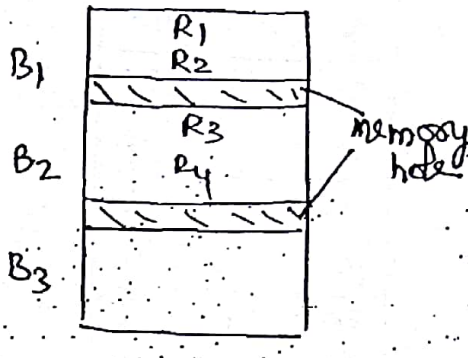
$= 2.5$ Records/Block

bytes of F is not fixed. (Record size for F is not fixed)

(+) Preferred for variable length records.

Unspanned Organisation

Complete Record must be in one Block.



Here complete record must be placed in 1 block only.

(+) Advantage - Less Access Cost

(+) Disadvantage - Internal fragmentation may not possible to avoid.

(*) Blockfactor

$= \frac{\text{Block size} - \text{Header}}{\text{Record size}}$

$= \left\lfloor \frac{B-H}{R} \right\rfloor$

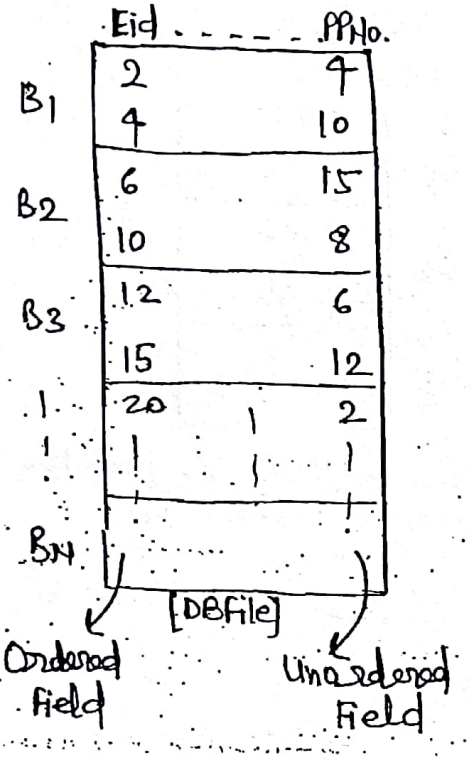
$= \left\lfloor \frac{1000-0}{400} \right\rfloor = 2$ Records/Block

(+) Preferred for fixed length records.

a) Access Cost from DB file without Indexing :-

DB Records are physically ~~recorded~~ Ordered based on Eid.

Remaining all fields are considered to be unordered.



Access Cost Based on Ordered Field

i) Select *
from Emp
where (Eid) = X;
Ordered Field

Here we don't know in which block X is present. So, here DB to access this ~~file~~ record, makes use of Ordered field (which is already known to DB).

Then he can apply an efficient search algo. (like Binary search)

$\lceil \log_2 N \rceil$ Block Access

ii) Access cost Based on Unordered Field

Select *
From Emp
Where (PNo) = Y
Unordered field

If search is based on Unordered field, then DB has to access all the N blocks to search the record.
(Linear search)

N Block Access

For First Level :- (Index of DB)

The pointer to the records in different blocks are itself stored in a block.
So, that the file maintaining

The pointer to the first record of block (or base address of each block) is maintained in a seperate file which is itself divided into blocks.

But now size of this first level table will be much smaller than size of DB table.

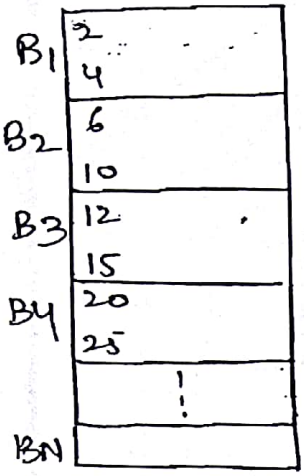
Also, in DB table, records are ordered using ordered field so search for corresponding record is based on ordered field.
So that $\lceil \log_2 N \rceil$ records to be accessed at each level of size N blocks.

Similarly for Second Level :- (Index of first level)

This table will maintain pointer to the base address of each block in first level table.

Basically second level is required in case of we have to access I/o units etc.

Similarly we can access records & index the tables upto k^{th} Levels.


B1
2
4
B2
6
10
B3
12
15
B4
20
25
!
BN

Index file :-
consist of (search key, pointer) pairs.

Entry in index file
< search key, pointer >

Attribute set which is used for index (Can be cand key of Table or can be Non key Attribute)

Block factor of Index file $\Rightarrow \left\lfloor \frac{B-H}{K+P} \right\rfloor$ Entries/Block

B : size of block
H : size of header
K : size of search key
P : size of pointer

④ Block factor of Index file > Block factor of DB file

Because, denominator of Block factor of DB file > Index file
size of (K+P) < size of R (here K only contains size of one or two attr. and P is pointer whereas R is size of whole record) ↓ all attr.

$\therefore \frac{B-H}{K+P} > \frac{B-H}{R}$

Multilevel Index

Index to the index until last level must be only one block.

Idle I/O cost to Access Record using Multilevel Index
(K+1) Blocks
K : # of levels of index

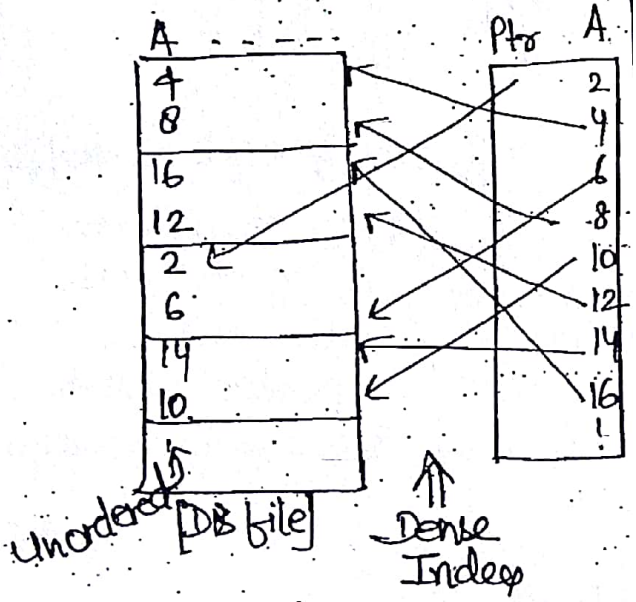
X: ordered Field

Search key(X) is cand key of DB table	Search key(X) is non key of DB table
⇓	⇓
Primary Index	Clustering Index

← Preferred index is Sparse →

DENSE INDEX :-

* Maximum possible entr
* for each database fi
*

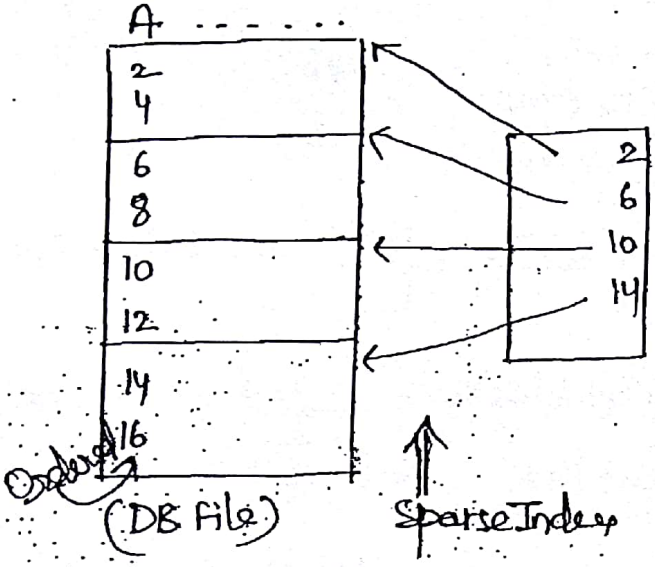


whenever there is Dense Inde

(# of entries in Index

Sparse Index

* less entries in index file
* for set of DB records, there exist entry in index file.
* possible only when search key is an ordered attribute



By default -

$$\left(\#\text{ of Index entries} = \#\text{ of DB file block}\right)$$

But it could not hold for certain cases.

Index File Size reduces means Access cost reduces.

(*) Ordered Field Index must be Sparse.

T/F

False

Q DB File Consist : 30,000 records

Record size: 100B Block size: 1024B

search key Size: 9Bytes

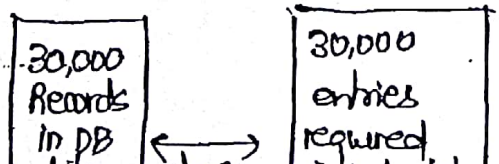
Pointer size: 6 Bytes

a) How many dense index blocks?

b) How many sparse index blocks?

Sol:- Dense Index:-

"Block" to determine size of an index file



$$\text{Block factor of Index file} = \left\lfloor \frac{B-H}{K+P} \right\rfloor = \left\lfloor \frac{1024}{15} \right\rfloor$$

default: Unspanned

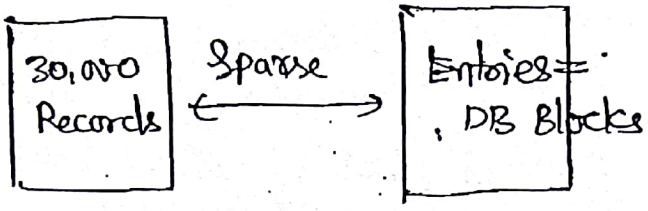
$= 68$ B

$$\#\text{ of Dense Index Blocks} = \frac{30,000}{68} = 442 \text{ Block}$$

For Spanned Organisation,

$$\Rightarrow \frac{30{,}000 * 15}{1024} = \left(\frac{3000}{\frac{B-H}{K+P}}\right)$$

b) Sparse Index



Blockfactor of DB file $\Rightarrow \left\lfloor \frac{B-H}{R} \right\rfloor = \left\lfloor \frac{1024-0}{100} \right\rfloor = 10$ Records/Blocks

# of DB blocks $= \frac{30{,}000}{10} = 3000$ blocks

∴ we require 3000 entries in index file

We can store 68 entries / block

∴ No. of blocks in Sparse index $= \left\lceil \frac{3000}{68} \right\rceil$

$= 45$ blocks.

∵ No. of blocks in Sparse Index < No. of blocks in dense Index

∴ Size of Sparse Index file < Size of Dense Index file

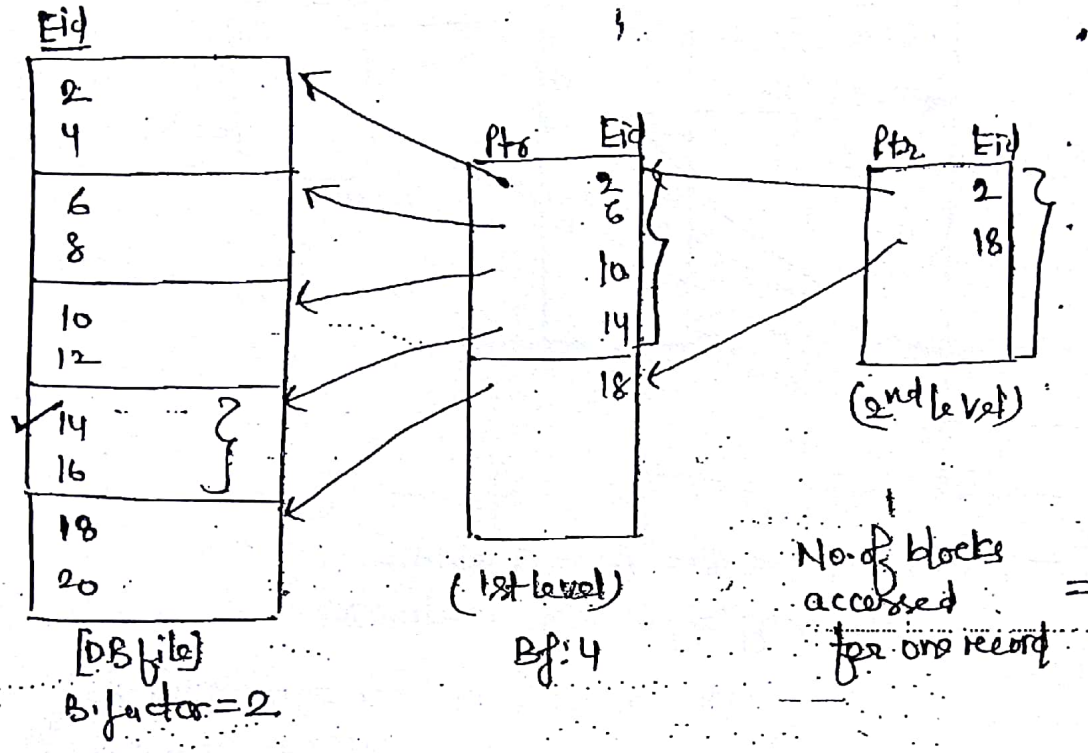
PRIMARY INDEX

Attribute set used for Index must be

- ordered field of DB file

and

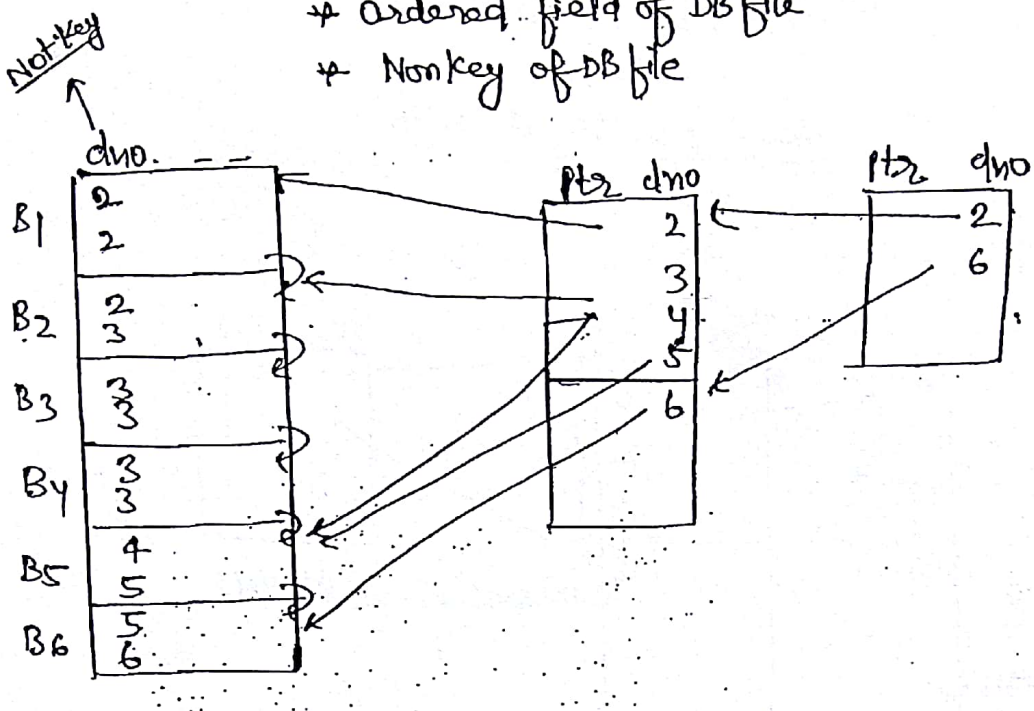
- candidate key of DB file



- I/O cost to access record using Primary Index with Multilevel Index $\Rightarrow$ (K+1) blocks.
  (K: # of levels of Index)
- Primary Index can be dense or sparse but sparse Primary Index is preferred.
- Atmost one Primary Index for any database relation is possible.
  (because ordering is possible only for 1 attribute which will have other attributes unordered and thus there can be only 1 primary index)

Clustering Index

↳ Attribute set used for Index must be
- Ordered field of DB file
- Nonkey of DB file

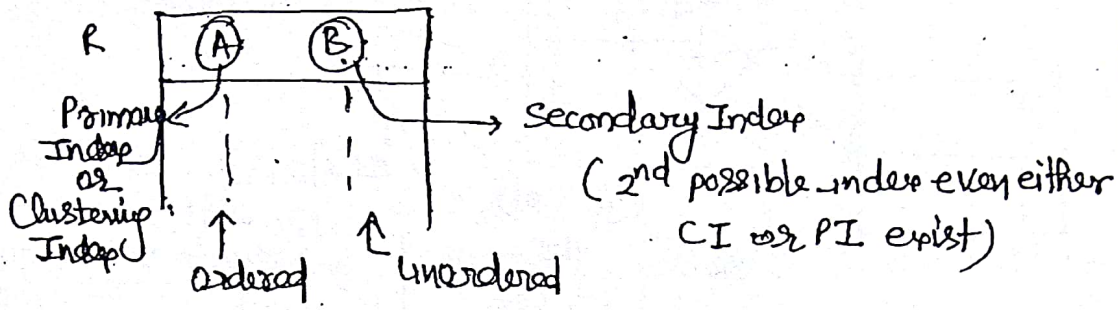


I/o cost to access record using clustering Index with multilevel
= k blocks + one or more blocks access from DB file (until next cluster begins)

- Here rounded pointers in DB files are required to access the same record sequentially.
- Here pointer will give the blocks and one block is pointing to the next block & searching will be done to access the records of the same cluster. Suppose we want to access the records of cluster 3 then we will access block B2 B3 B4 B5. B5 will be accessed because search will fail or stop for cluster-3 in this block. (Here only one more block will be accessed)
- Clustering Index mostly sparse index
- Dense clustering index may occur if all clusters consist one-one record each.
- Either clustering index or Primary index is possible for any database table but not possible for both. becoz for primary index ordering for key is required whereas for clustering index ordering for nonkey is required.

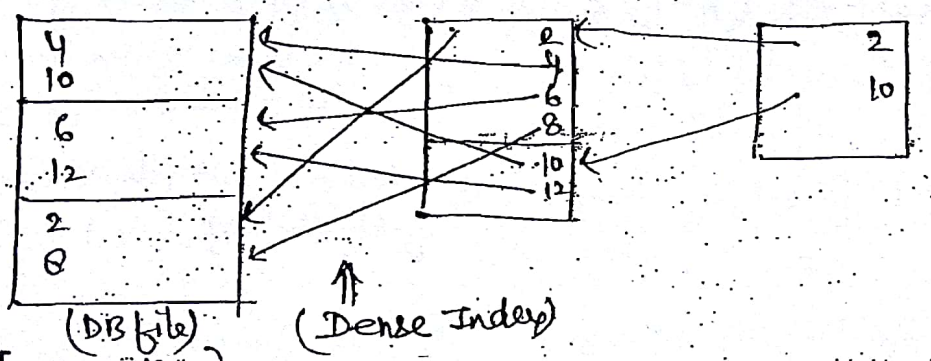
Secondary Index

Attribute set used for index is un-ordered fields of the DB file (and) Key / Non key



→ Always dense

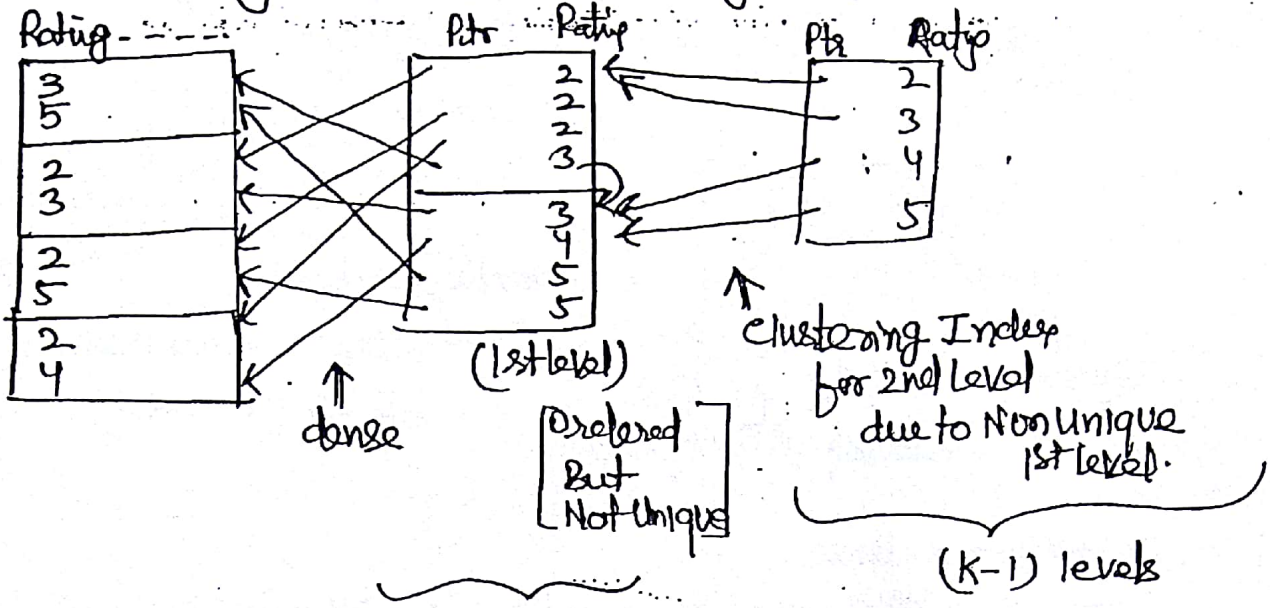
→ More than one SI possible for DB table



(*) I/o cost for secondary index (over key) using multilevel index = (K+1) blocks

(*) It is same as Primary Index but the difference between Primary & secondary index is that the indexing is sparse at 1st level whereas the indexing is dense at 1st level.

Secondary Index over Non Key :-



I/O cost ⇒ (K-1) blocks + (one or more blocks from 1st level Index until next cluster begins)

+ (m blocks of DB file access if cluster size : m)

Workbook

Pg 63
Q.11

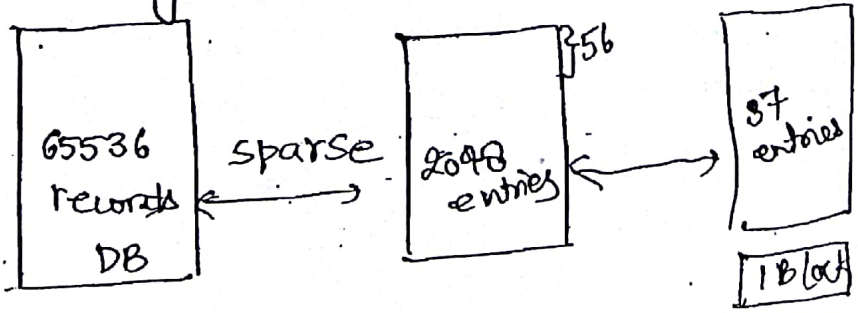
a) secondary Index over key

b) Primary Index

c) clustering Index (Cluster size = 16)

d) secondary Index over non key (Cluster size = 16)
(every 16 records whose non key value same)

11 (b). Primary Index



Block factor of DB file $= \left\lfloor \frac{B-H}{R} \right\rfloor = \frac{1024}{32} = 32$ Records/Blo

# of DB Block $= \left\lceil \frac{65536}{32} \right\rceil = \underline{2048 \text{ blocks}}$

# of entries in Index file = 2048

Block factor of Index $= \left\lfloor \frac{B-H}{R} \right\rfloor = \left\lfloor \frac{1024}{18} \right\rfloor = 56$ entries/block

No. of blocks at 1st level $= \left\lceil \frac{2048}{56} \right\rceil = \underline{37 \text{ blocks}}$.

Now since this level has more than 1 block, we go for next level until we get only one block at that level.

No. of entries in 2nd Level = No. of blocks at 1st leve
= 37.

∵ Block factor of Index = 56 entries/block

$37 < 56$

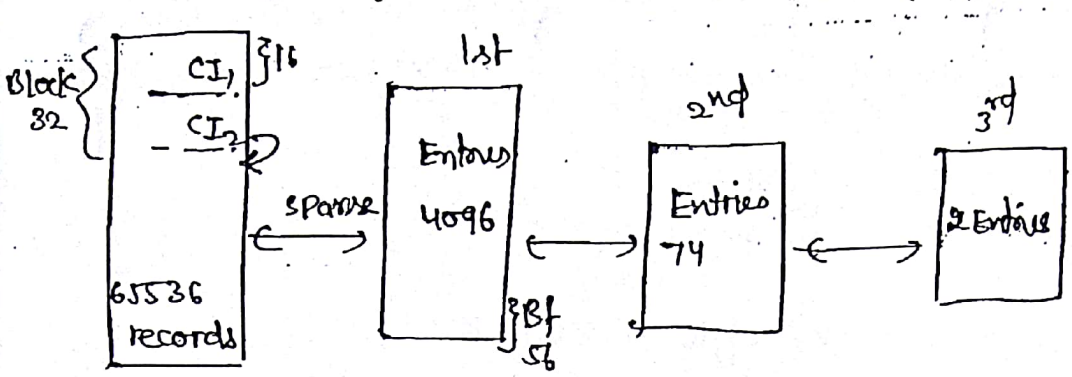
so, only one block at second level.

i) # of index blocks = 38

ii) # of levels = 2

iii) I/O cost $(K+1) = 3$ blocks.

11. c]



$$\#\text{ of clusters} = \left\lceil \frac{65536}{16} \right\rceil$$

$$= 4096 \text{ clusters.}$$

So, #entries at 1st level = 4096

Block factor = 56 entries/block of Index file

$$\#\text{ of blocks at 1st level} = \left\lceil \frac{4096}{56} \right\rceil = 74 \text{ blocks}$$

# of entries at 2nd level = 74

At second level,

Block factor of Index file = 56 entries/block

total entries = 74

$$\#\text{ block at 2nd level} = \left\lceil \frac{74}{56} \right\rceil = 2 \text{ blocks}$$

At third level,

we require 1 block for 2 entries.

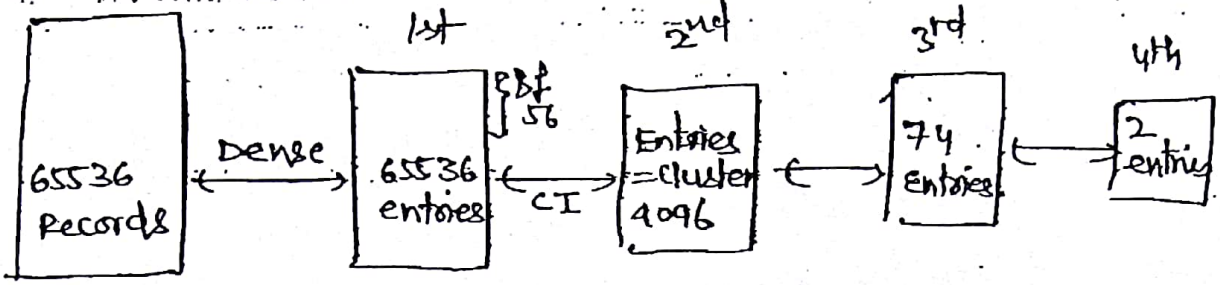
# of index blocks = 77 blocks

# of levels of index = 3

I/O cost = 3 + (2) = 5 blocks.

↳ because we have to access 2 blocks to confirm that cluster is completed.

11. a. Secondary Index Non Key



Since Dense Index,

So, # of entries in index file = 65536 (same as # records in DB

Block factor of index file = 56 entries / block

# of blocks at 1st level = $\lceil \frac{65536}{56} \rceil$ = 1171 blocks

At 2nd level, clustering is applied

# of clusters = $\frac{\text{total entries}}{\text{Cluster size}}$ = $\lceil \frac{65536}{16} \rceil$ = 4096

So, at 2nd level, # entries = # of clusters = 4096

# of blocks at 2nd level = $\lceil \frac{4096}{56} \rceil$ = 74 blocks

Block factor

Then Dense index at 3rd level.

# of entries at 3rd level = 74.

# of blocks = $\lceil \frac{74}{56} \rceil$ = 2 blocks

At 4th level, # of entries = 2

# of blocks = 1

i) # of levels = 4

ii) # of index blocks = 1171 + 74 + 2 + 1 = 1248 blocks

iii) I/O cost = (X-1) + one or more blocks of first level + from DB # of blocks = cluster

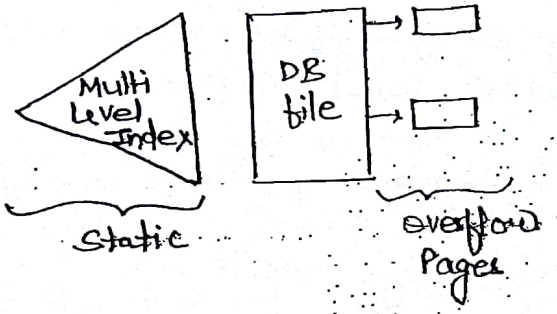
copyright©theorypoint.com The Best Gateway Of Success

Multilevel Index :-

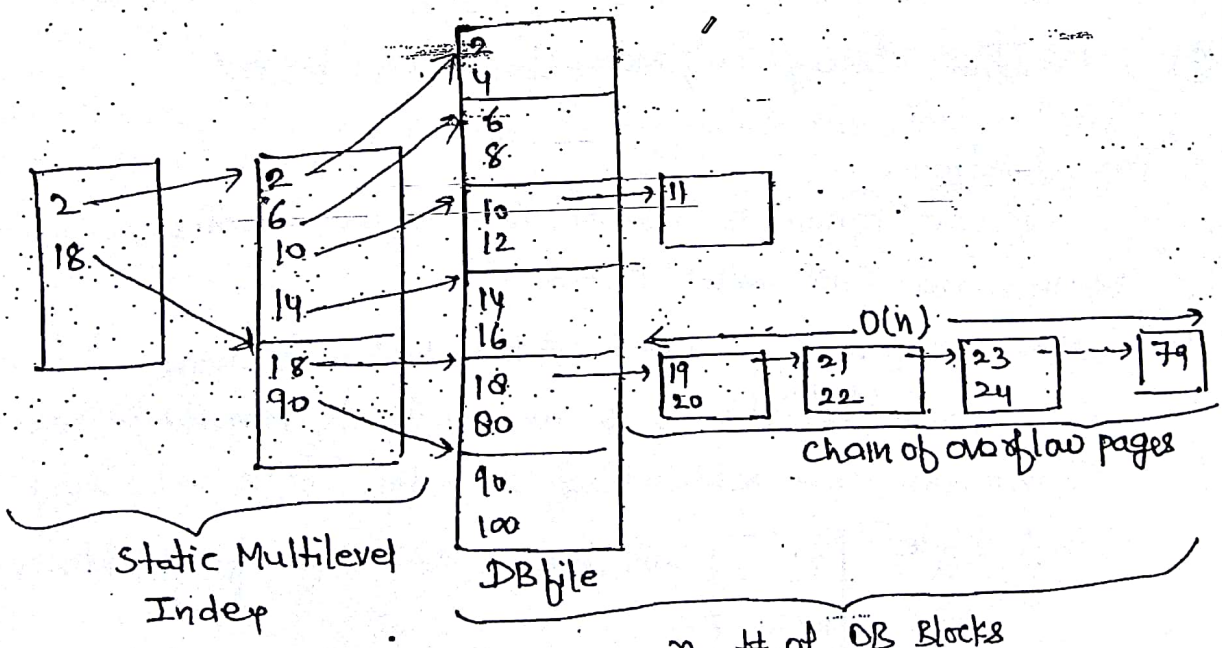
1) Static Multilevel Index
2) Dynamic Multilevel Index

Static Multilevel Index

Index for given DB records is static [Becoz of insertion / deletion of DB records not allowed to modify index].



The new records to be inserted in DB files are inserted in the overflow pages.



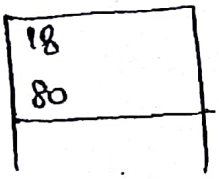
Disadvantage

For insertion operation,

we can insert any additional record eg. record 11 by a pointer given to record 11 from record 10 at overflow pages.

Similarly other records can be entered.

suppose record no. b/w two block entry is very large then, many no. of overflow pages can be inserted after 18 until 79 page #.



or

It could reach the complexity of O(n).

But disadvantage here is that,

to access record # 79, it will require to access the index blocks and then DB file block plus, searching in the chain of overflow pages.

① Worst case access cost = O(n)

where n = no. of DB blocks.

② Minimum usage of index blocks can be 0%.

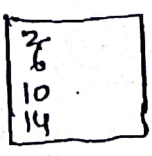
↓

for deletion,

if one record is deleted, then its index will be maintained (not much problem)

However no. of records deleted is much more

eg. from record 2 to 16 are deleted, then also their indexes are maintained and the block at 1st level index file will have 0% usage of index blocks.



and so we won't know from the index file whether the data

Dynamic Multilevel Index

→ Index file must be modified because of data records insertion / deletion

Goals:
- overflow pages are not allowed
- Minimum usage of index blocks should be 50%

Standard Data Structures are used for dynamic multilevel index are –

1) BTree
2) B^+ Tree

BTree / B^+ Tree (Balanced Search tree)

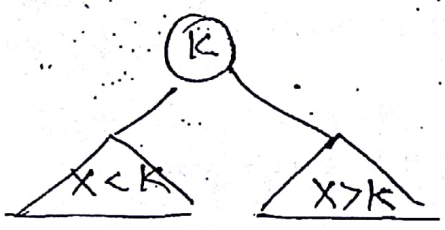
→ comes under dictionary data structure

→ Data Structure which supports efficient way to
- → search element
- → Insertion / deletion of element

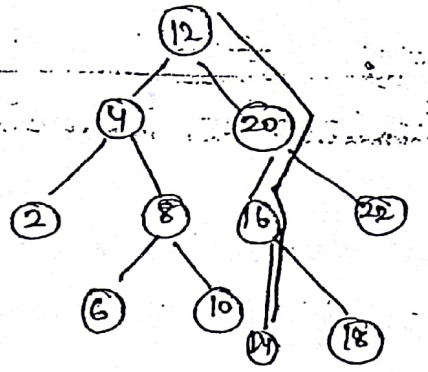
eg
- Balanced Search Tree – O(logn)
- Hashing – O(1)

Search Tree

Parent key must be greater than all the keys used on left subtree & Parent key must be lesser than all the keys used on the right subtree.



eg



Probe sequence for X=14 : 12, 20, 16, 14 → Probe sequence

result: 14 is found

Probe sequence for X=17 : 12, 20, 16, 18 → Probe sequence

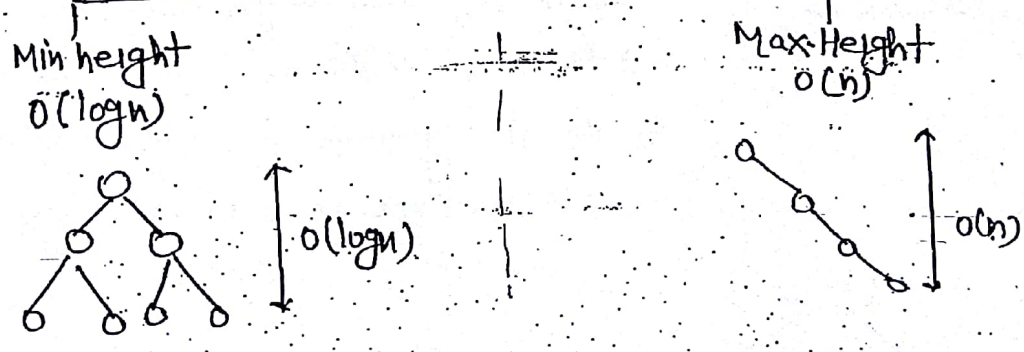
result: 17 is not found

Thus we are accessing one node/one key at each level.

one such example of B-Tree is Binary search tree.

Binary search Tree

for n distinct keys



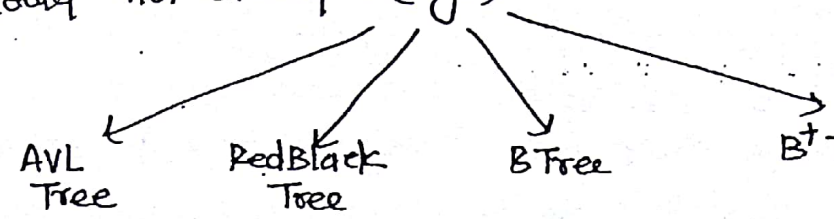
⇒ worst case search cost : O(n) for n keys

(This is the drawback of normal search tree and ∴ we use balanced search tree)

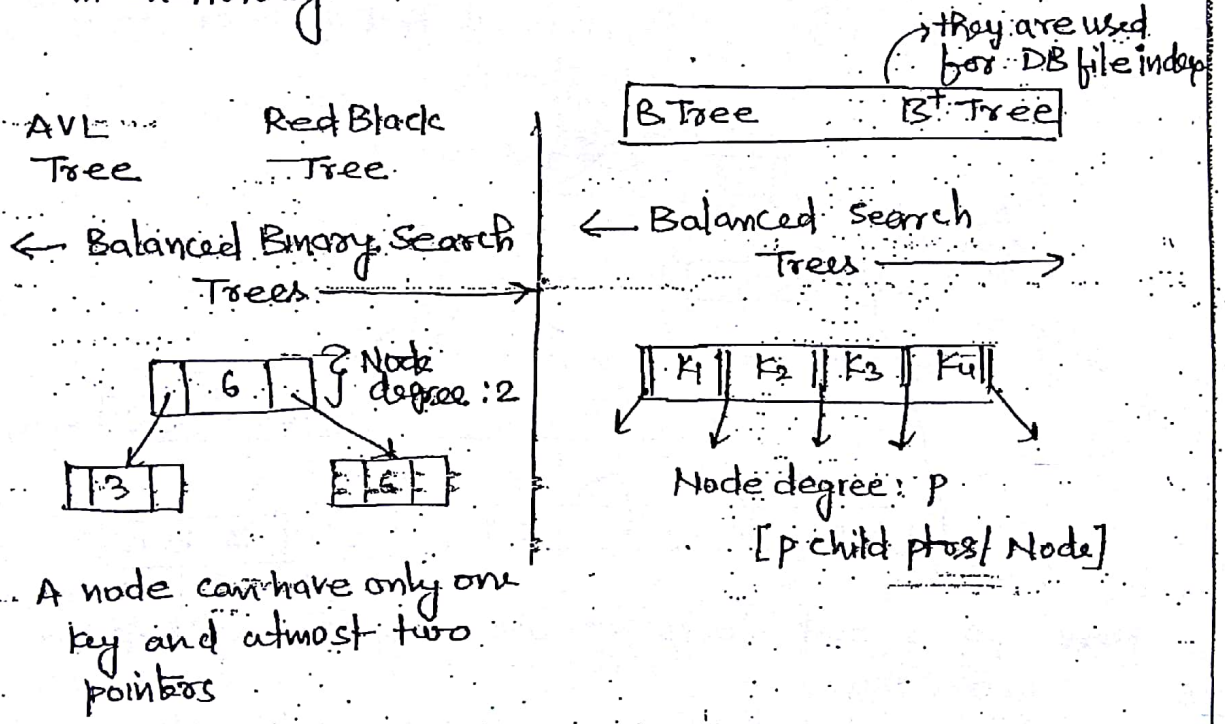
Balanced Search tree

(Height Restricted Search tree)

Height of Balanced search tree for n distinct keys should not exceed O(logn)



Because of this, search cost never exceeds O(logn) and so we can use them for search/insertion/updation in dictionary.



Key factor for using B/B+ trees for DB Index rather than Balanced BST:

1) Index for SM blocks (or DB Index as DB file is stored in Sec. memory) so that Index file also must be in Secondary Memory

Data access from secondary memory in terms of Blocks.

2) If AVL Tree is used for index:-

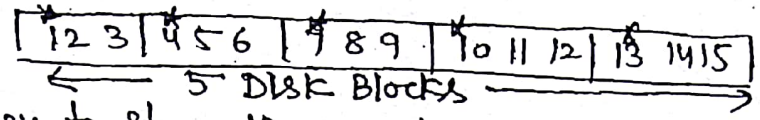
Here at one node only one key can be stored (in AVL Tree)

So, we know that Index file is stored in sec. memory

Here each ~~data~~ node of AVL tree is stored in a seperate Disk Block:

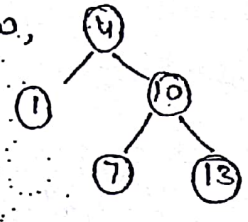
for eg.

DB file is



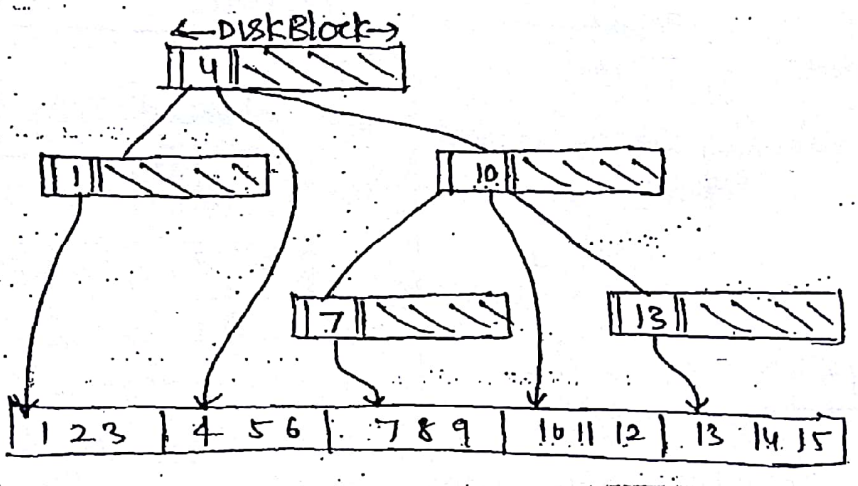
then to store their index using AVL tree, we choose {1, 4, 7, 10, 13} as key values. for seperate Disk Blocks

Now,



Can be one such AVL tree.

Each node of tree require a seperate disk block to be stored as shown,



Here for 5 disk blocks we are using 5 more blocks to access them.

OR we have 5 index blocks for 5 disk (data) blocks.

that is not efficient.

Suppose, block size = 4 KB

and AVL Tree Node size = [2 Child Pointers + key + Data Pointer + Height]

≃ 100 Bytes (approx. randomly taken)

Thus, out of 4 KB we are using 100 Bytes only.

Therefore 99% of the disk block ...
to wastage of Secondary Memory space.

Also, to access 1 record we have to access 4KB disk block and therefore if we want to access record 7 then we have to access 3 disk blocks (: 4KB x3) i.e. access cost or I/O cost becomes too high.

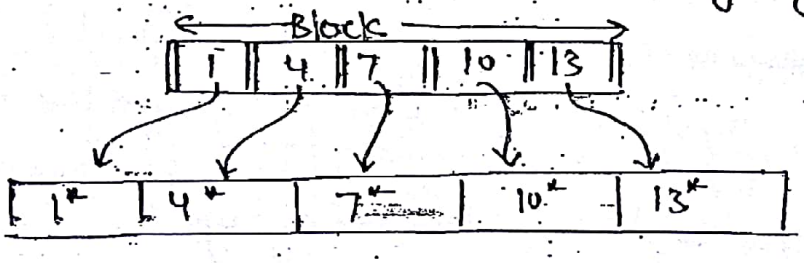
or we had to transfer 100 B only but now we are transferring 4KB for that one record so access cost is high.

Disadvantages

* wastage of SM space.
* Index file too large in size.
* wastage of I/O cost too high.

3] If B/B+ Tree Index is used -

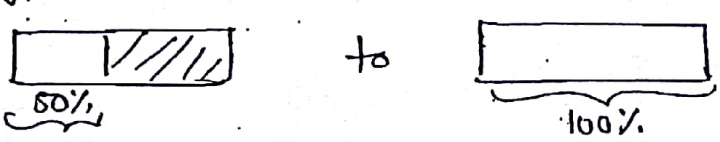
* Here each node can store many keys.



(If one disk block is completely used, then the remaining keys are stored in another block).

→ Entire disk block should be allocated for B Tree Node.

a) Every Index Node should be min. 50% occupied.



Advantages

* less wastage of SM space
* less # of index blocks
* less Access cost
* less wastage of access cost

Q Store 10 lakh keys in Main Memory and used Data Structure for searching. which is preferred Data structure?

a) AVL Tree Used
- 10 lakh nodes
- 20 lakh pointer (Child Pointers)
- More Height

b) B Tree Used [preferred]
- less Nodes
- less ptrs.
- less height.

Q store 100 keys in main memory and used data structure for searching. which is preferred Data structure?

In this case, AVL Tree should be used because of less no. of keys in the main memory.

For indexing in the secondary memory, the data structure used is B/B+ Tree if the no. of keys are less or more.

AVL Tree Used → less no. of keys + Indexing in Main Memory

B/B+ Tree Used when → i) more no of keys in Main Mem.
ii) less/more no. of keys in Sec. Mem.

B-Tree Definition

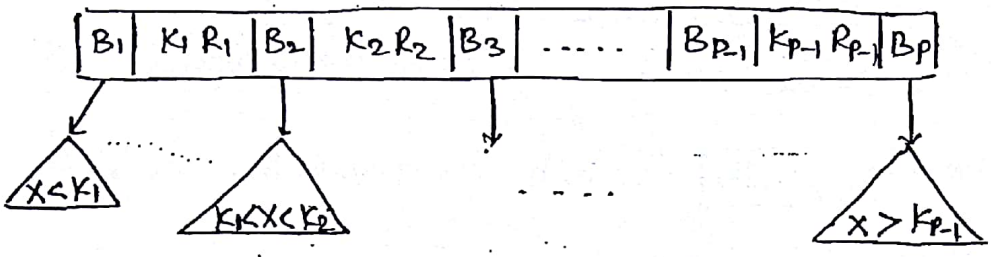
order P : Max Possible C_p (Child Pointer) per Node

a) Node Structure

If there are p child pointers in a node
then there will be (p-1) [Key, R_p] in the node

↳ Data Pointer or Record Pointer

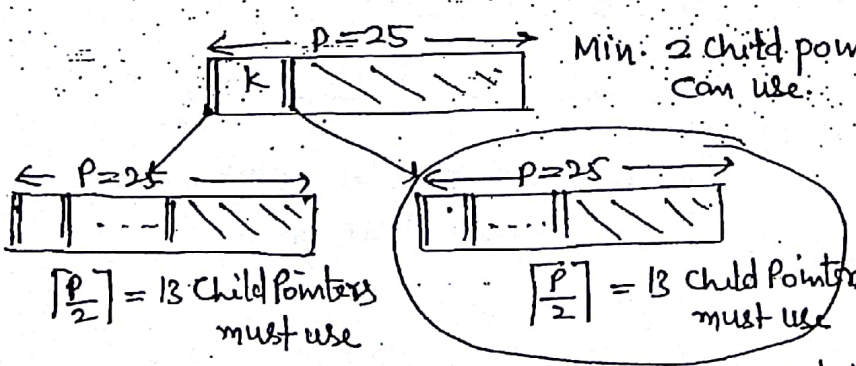
eg



Child pointer points to next index block
Data pointer points to the database block.

b) Every internal node except Root, atleast $\lceil \frac{p}{2} \rceil$ child pointers & $(\lceil \frac{p}{2} \rceil - 1)$ keys must be used.
and atmost p child pointers & (p-1) keys must be used.

c) Every Root Node can be
atleast 2 child pointers and 1 key
atmost p child pointers & (p-1) keys can be used.

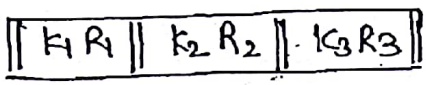


d) All leaf Nodes must be at same level.

Balancing Conditions – (b) (c) (d)

eg order $P = 4$ [Max. child pointers per Node]
↳ 3 keys Possible

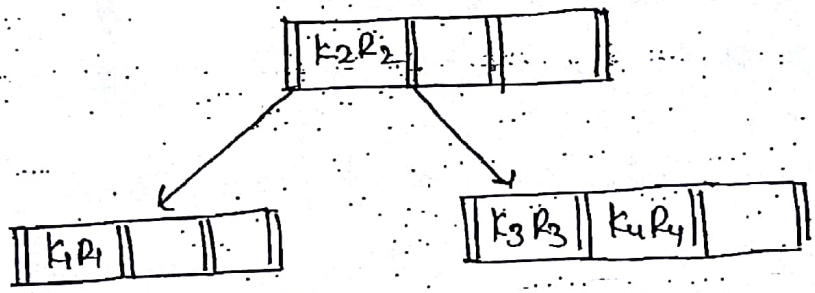
$K_1 < K_2 < K_3 < K_4$



Here Block is full. as almost 3 keys possible

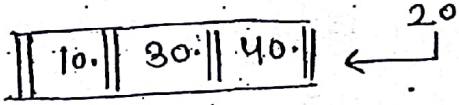
Now if $K_4 R_4$ is to be inserted in the same node

Node Splitting takes place ⇒



Q. Construct B Tree with order $P = 4$ following sequence of keys:
10, 30, 40, 20, 15, 25, 50

10, 30, 40.



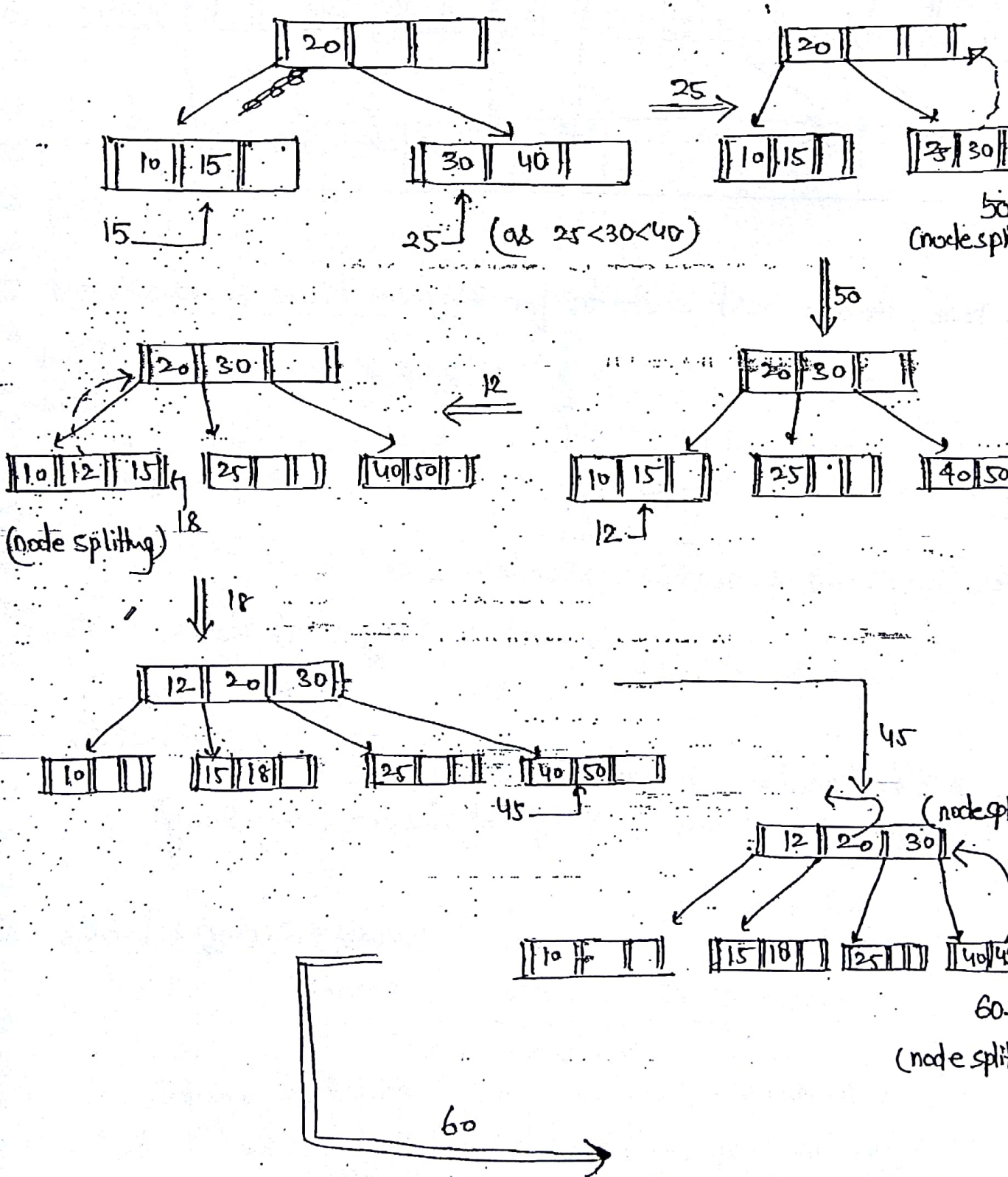
$10 < 20 < 30 < 40$

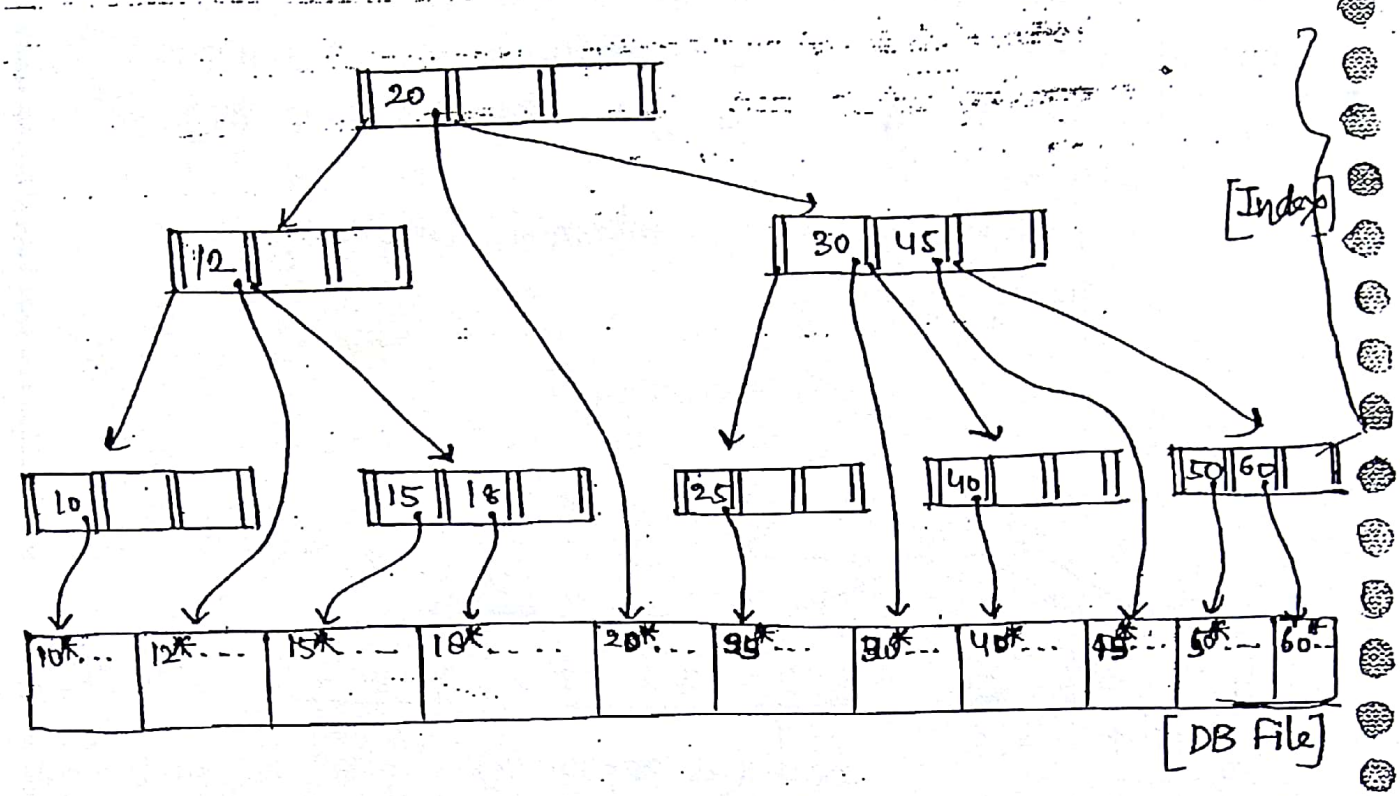
20 & 30 both can be middle keys & thus we can have 2 B-Trees

Insertion should ...

if in case leaf node is full then node splitting will take place.

So, key can be placed in internal node only by node splitting.





(+) B Tree index best suitable for Random Access of Some Record

```
select *
From R
where X=25;
```

Height of B Tree: $\lceil \log_p n \rceil$ (same as no of levels in multilevel Index)

n: distinct keys

p: degree of Node

For these kind of queries, B Tree is best as

I/O Cost $\approx (\log_p n + 1)$ blocks

($[=k+1]$ blocks in multilevel Index)

(-) B Tree Index not best suitable for Sequential Access of Range of Record. [Range Query]

```
select *
From R
where A≥15 and A≤40
```

X Blocks

I/O cost $\approx X [\log_p n + 1]$ blocks

15 ⟶ 4 block
18 ⟶ 4 block
20 ⟶ 2 block access required
25 ⟶ 4 block
30 ⟶ 3 block
40

So, I/O cost ∝ $X[\log_p n + 1]$ blocks (too la

X : No. of blocks in the range.

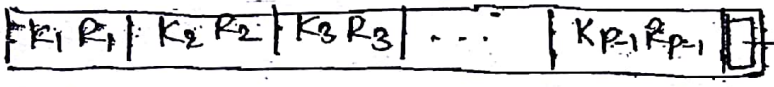
So, we go for B+ Tree.

B+ Tree Definition

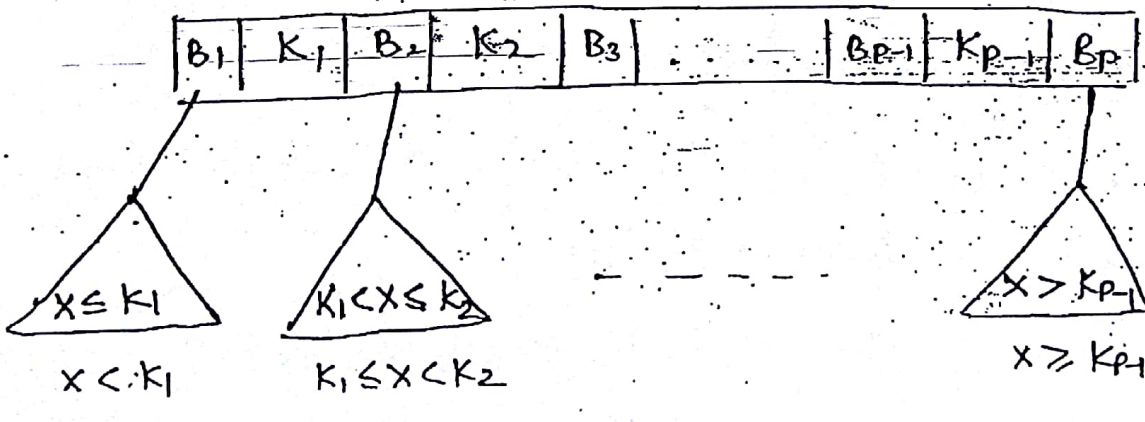
order p : Max. Pointer per Node

a) Node Structure

i) Leaf node consists of {set of (K_y, P_t) pairs and one Block ptr



ii) Internal Node { only child Pointers & keys }



Left Biasing → Parent node is maintained in left child node

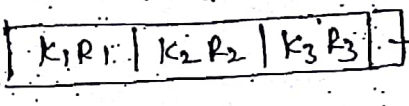
Right Biasing → Parent node is maintained in right child node

(A) Balancing conditions of B^+ Trees is same as that of B Trees (b), (c), (d).

Construction of B^+ Trees

a) **Leaf Node Splitting** :- order P: 4 (max. pointers per node)
out of 4 pointers, 1 pointer is used for block pointer and remaining pointers are used as remaining pointers.
∴ 3 (Key, Rp) pairs and 1 block Pointer.

Suppose, one leaf node is →

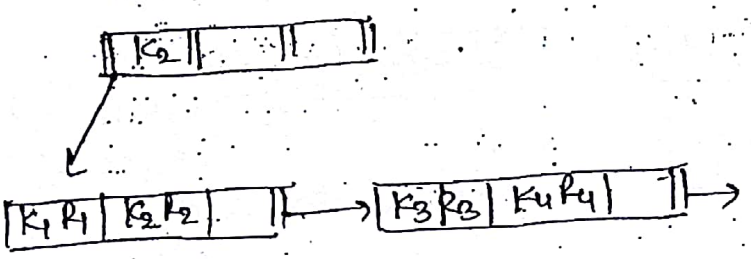
(full) 

Insert ($K_4 R_4$)

$K_1 < K_2 < K_3 < K_4$

↳ Now 4 B^+ Trees possible.
as K_2 and K_3 are middle nodes
For both left biasing & right biasing possible.

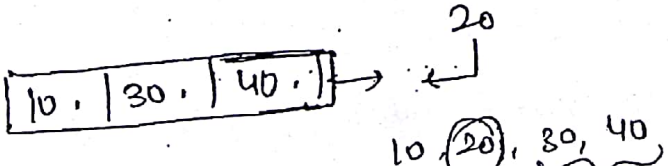
Left Biasing →



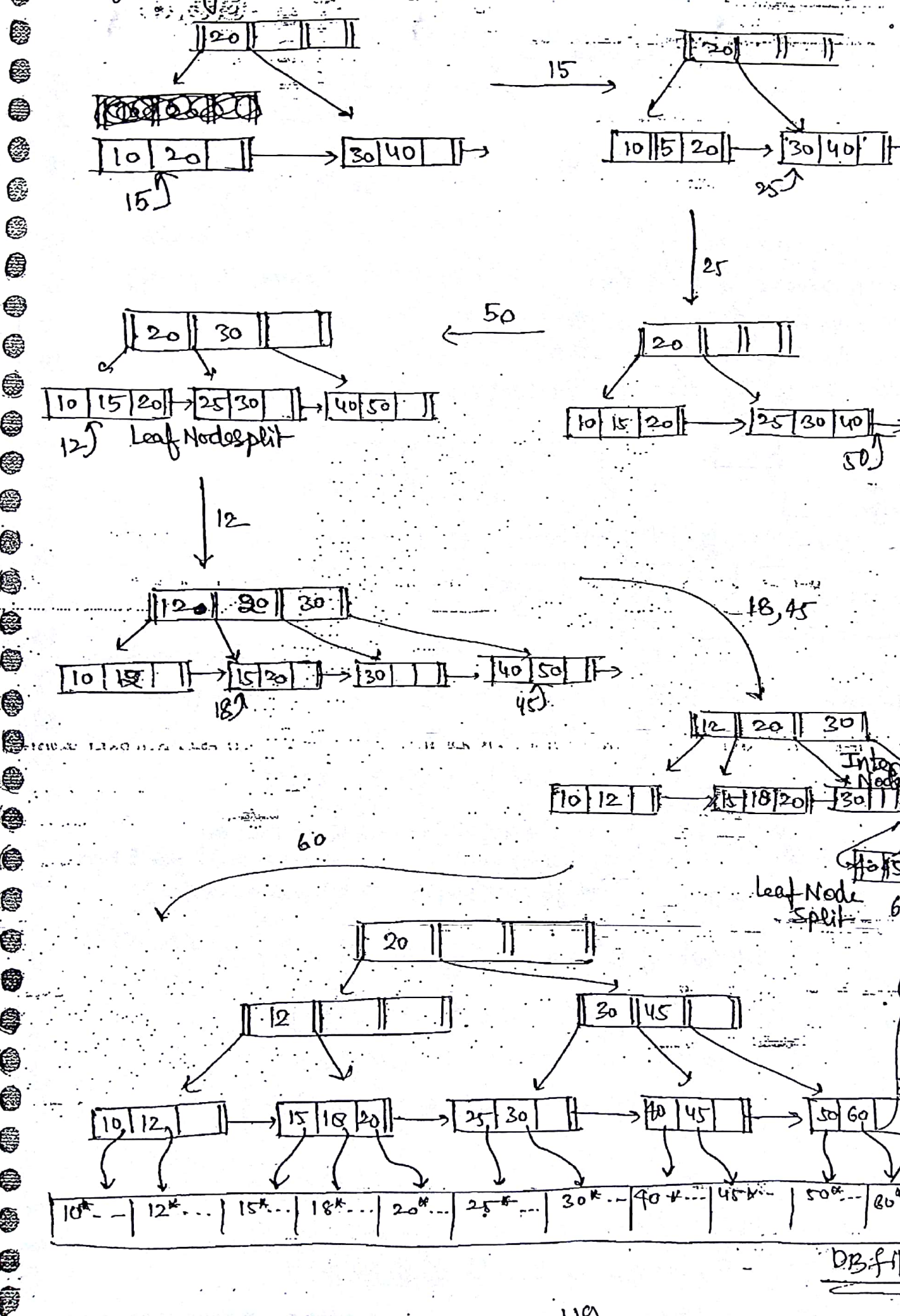
b) **Internal Node Splitting** :- same as B Tree.

Q Construct B^+ tree with order P = 4 (max ptr per node) following sequence of keys

10, 30, 40, 20, 15, 25, 50, 12, 18, 45, 60



10 (20), 30, 40


20
10 20
30 40
15
15
20
10 5 20
30 40
25
25
50
20 30
10 15 20
25 30
40 50
12
Leaf Node split
20
10 15 20
25 30 40
50
12
12 20 30
10 12
15 18 20
30
18, 45
Leaf Node Split
60
20
12
30 45
10 12
15 18 20
25 30
40 45
50 60
10* 12* 15* 18* 20* 25* 30* 40* 45* 50* 60*

B+ Tree is best suitable for Random Access of some record

or Sequential Access of Range Query.

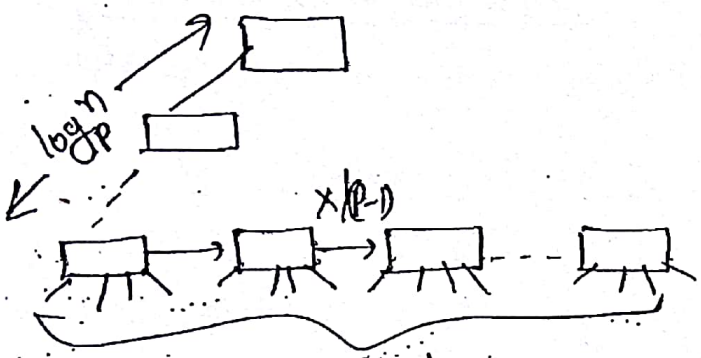
```
select *
from R
where A >= 15 and A <= 45
```

Here we first reach the leaf blocks 15 and then we can directly access next records sequentially on other leaf node by block pointer.

```
select *
from R
where A = 15
```

I/O cost $\approx \log_p n + 1$ blocks

(same as B tree)



I/O cost $= \left(\log_p n + \frac{X}{P-1} + X\right)$ blocks.

- $\log_p n$ → to reach first leaf node
- $\frac{X}{P-1}$ → to access from 15, 16, 17 ... 45 (X)
  - to access X leaf nodes (blocks) we have to access through block pointer
  - at most $(P-1)$ keys = 3 keys in each block there are
  - so # blocks accessed $= \frac{X}{(P-1)}$ from 15 to 45
- X → there are X blocks in DB file

I/O cost of B+ tree << I/O cost of B Tree

$(X + \log_p n)$ << $(X \cdot \log_p n)$

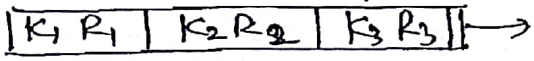
(4) in B+ Tree Index

i) every key used to index must be at leaf level.

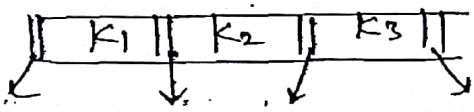
ii) all data pointers stored at leaf ~~no~~ level.

iii) each leaf node pointed to next leaf node.

iv) leaf node consist set of (Key, Rp) pairs and only one Block pointer which is pointed next leaf Node.



v) Internal node consist only keys & child ptrs.



vi) Entire tree must completely be left biased or right ~~biased~~ but ~~not~~ be mixed.

Becoz in that case, to search for parent node, both left & right child has to be accessed for that node which is not the property of a search tree. Either we should go left or right for searching (Probe sequence of a key should not be ambiguous) a key.

vii) ~~structure~~ of Leaf Node and Internal Node is different in B+tree which is same in B Tree

viii) If left Biasing used:-

Max key of each Leaf Node except Last (Rightmost) Leaf Node is stored as Internal Node keys.
(no. of keys stored in internal nodes = (# of leaf nodes - 1)

For Right Biasing

Min key of each Leaf Node except First (Leftmost) Leaf Node is stored as Internal Node keys.

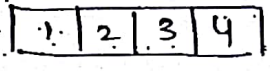
iv) To insert key into level (l) B/B+ Tree,

Min no. of Nodes required = 0
(if free space is available)

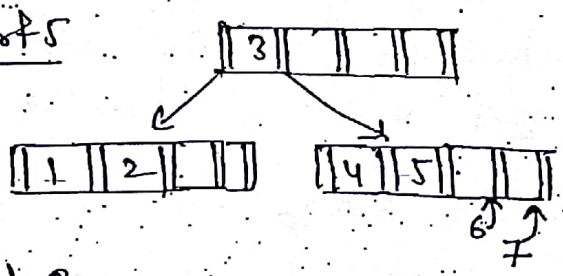
Max. no. of Nodes required = (l+1)
(if at each level, node splitting is done. so,
at each level → 1 new node
so l new nodes
+1 new node at root
= l+1 new nodes)

Q order P=5 [Max Cp per BTree)
if keys inserted into empty BTree with sequence
1, 2, 3 - - -
which is first key that leads to level-4 BTree?

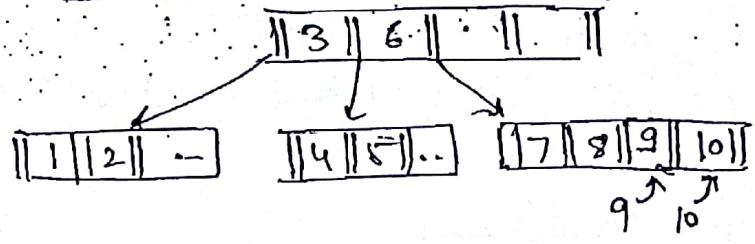
Sol:- Node splitting take place when node is full.

i) 

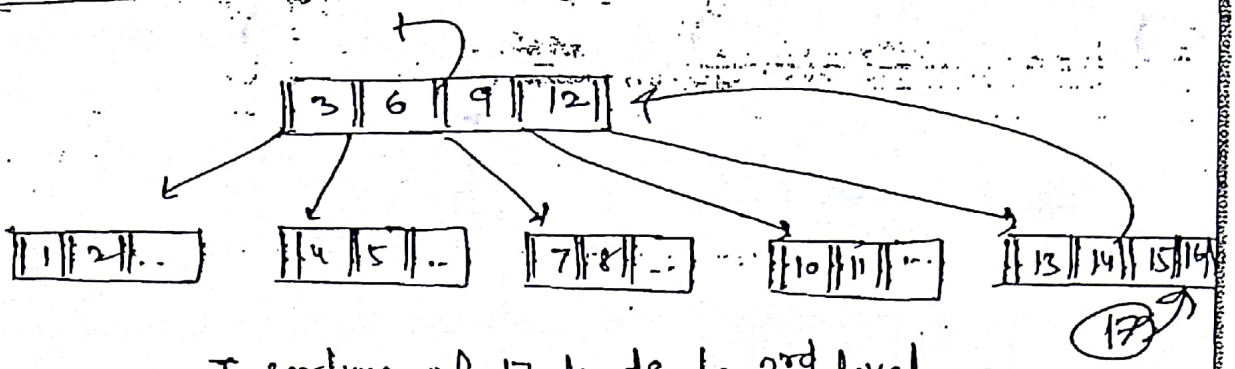
Insert 5



ii) Insert 8



Insert 9, 12 node splitting occured.



Insertion of 17 leads to 3rd level.

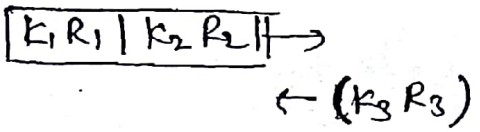
b) which is first key insertion leads to B+ tree at 4th level

a) by using left biased

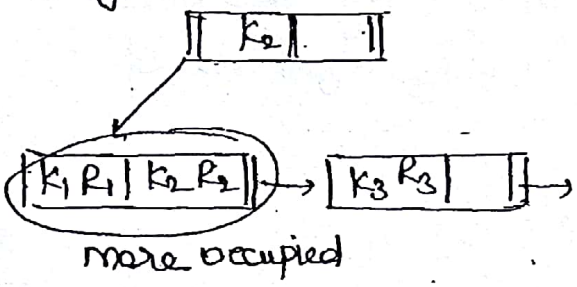
b) using Right biased.

(*) Left Biasing leads to more Node splitting if sequence of keys is in descending order.

eg order P : 3 [B+ Tree]



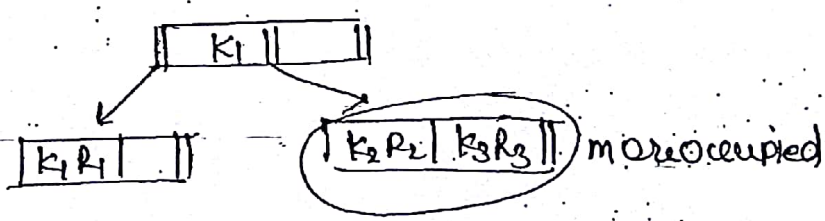
Left Biasing



10, 8, 6, 4, 3, 1 →

further node always entered in left leaf and which is more occupied, so splitting will be more

(*) Right Biasing leads to more Node splitting if sequence of keys is in ascending order.



1, 3, 4, 6, 8, 10 →

always right side will be entered new key so more splitting results.

** Q: Find order of B/B+ Tree Node for given Block size, Child Block Pointer (Bp), Record Pointer (Rp), Key Size.

Q Block size: 1024 bytes, Bp: 7 Bytes

Search key size: 8 Bytes Rp: 9 Bytes

Definition of order could be changed

order p: max possible child pointer in B Tree Node

what is best possible order of BTree Node?

Sol: one disk block is allocated for 1 BTree Node

Block

$$p*B_p + (p-1)[K_y + R_p]$$

$$p * B_p + (p-1)[K + R_p] \leq \text{BlockSize}$$

$$p * 7 + (p-1)[8+9] \leq 1024$$

$$24p \leq 1041$$

$$p = \left\lfloor \frac{1041}{24} \right\rfloor = 43$$

max. child ptr per Node

Q order p: Max possible keys per B Tree Node

what is best possible order of B Tree Node?

Sol: p = Max keys per Node

$$p[K+R] + (p+1)*B_p$$

Block $(p+1)*7 + p[8+9] \leq 1024$

$$p = \left\lfloor \frac{1017}{24} \right\rfloor = 42$$

$p = 42$

Not always, no. of B_p = 1 + (no. of keys)

for right ans. we should calculate as per given cond^{n}

Q. Order p is { b/w 1 to (2p) keys for Root Node
b/w p to (2p) keys for other Nodes.
best.

What is possible order of B Tree?

Allocation of space depends on max. limit always.
(or) max. occupancy

So, we take 2p keys.

$2p(K+P_p) + (2p+1)B_p \leq$ Block Size.

If
p: { 2 to 2p Bp
p to 2p Bp
from
2p (Bp) +
(2p-1)(K+Pp) ≤ Blocksize

Q

Block Size: 1024 B

Search Key: 8 B Pp: 9B

Bp: 7 bytes

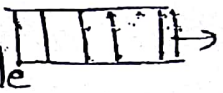
order p: max possible ptrs in B+Tree.

what is best possible order of B+ Tree

a) Leaf node

b) Internal node.

Sol:- a) Leaf node



p: Max possible ptrs. per node

$(p-1)[K+P_p] + B_p \leq$ Block size

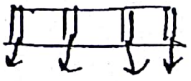
$(p-1)[K+P_p]+B_p$ — Block

$(p-1)(17) + 7 \leq 1024$

$17p \leq 1034$

$p = \left\lfloor \frac{1034}{17} \right\rfloor = 60$

b) Internal node



p: max possible ptrs. per node

$p * B_p + (p-1)$ keys — Block

$p * B_p + (p-1)\,key \le$ Block size

$p*7 + (p-1)\,8 \le 1024$

$15p \le 1032$

$p = \left\lfloor \frac{1032}{15} \right\rfloor \approx \underline{68}$.

Q In which case, possible order for leaf Node will be same as that of Internal Node.

Leaf Node	Internal Node
$(p-1)[K+R_p] + B_p$	$p*B_p + (p-1)\,Keys$
$(p-1)\,keys + (p-1)R_p + B_p$	$(p-1)\,keys + p*B_p$
$(p-1)R_p + B_p$	$= p*B_p$
$(p-1)R_p$	$= (p-1)B_p$
R_p	$= B_p$

So, if size of (B_p) = size of (R_p)

Then B^+ Tree Internal Node order and Leaf Node order is equal.

Workbook

Pg65 Q27 Block size = 1KB, $R_p = 7B$, Key = 9B, $B_p = 6B$

order of leaf node ?

B^+ Tree

Order: max (value, R_p)

$(p)[K+R_p] + B_p \le 1KB$ (∵ order p: no. of (key, R_p) pair)

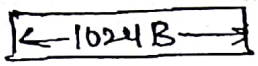
$p(9+7) + 6 \le 1024$

$p = \left\lfloor \frac{1018}{16} \right\rfloor = 63.$

④ if Disk-block Allocated for

1) B Tree Index

K=8
Bp=7
Rp=9



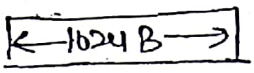
order P = 43

Min: $\lceil \frac{P}{2} \rceil$ = 22 Bp && 21 Key stored in one node.

Max: P = 43 Bp && 42 Keys stored in one node.

2) B+ Tree Index

K=8
Bp=7
Rp=9



Internal Node P = 68

Min: $\lceil \frac{P}{2} \rceil$ = 34 Bp && 33 keys in one node

Max: P = 68 Bp && 67 Keys in one node

Leaf Node P = 60

Min keys = 29 keys
Max keys = 59 keys

∴ For given disk blocks,

order of B Tree Node < order of B+ tree Node

so that # of index blocks for n keys using B Tree > # of index blocks for n keys using B+ Tree

⇒ I/o cost of B Tree > I/o cost of B+ tree.

⑤ For same order, B Tree is more better than B+ Tree

⑥ For same disk block size, B+ tree is more better than B Tree. (as true order varies) (discussed above)

min/max keys/Nodes in B/B+ Tree

for given level :- order p : max ℘ per Node

Height	Min. Nodes	Min Bp	Min key	Max Nodes	Max Bp	Max keys
0	1	2	1	1	p	p−1
1	2	$2\lceil \frac{p}{2} \rceil$	$2(\lceil \frac{p}{2} \rceil - 1)$	p	p^2	$p(p-1)$
2	$2\lceil \frac{p}{2} \rceil$	$2\lceil \frac{p}{2} \rceil^2$	$2\lceil \frac{p}{2} \rceil (\lceil \frac{p}{2} \rceil - 1)$	p^2	p^3	$p^2(p-1)$
3	$2\lceil \frac{p}{2} \rceil^2$	$2\lceil \frac{p}{2} \rceil^3$	$2\lceil \frac{p}{2} \rceil^2 (\lceil \frac{p}{2} \rceil - 1)$	p^3	p^4	$p^3(p-1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
h	$2\lceil \frac{p}{2} \rceil^{h-1}$	$2\lceil \frac{p}{2} \rceil^{h}$	$2\lceil \frac{p}{2} \rceil^{h-1} (\lceil \frac{p}{2} \rceil - 1)$	p^h	—	$p^h(p-1)$

(iv) Min. no. of Nodes in B/B+ tree with order p & height h

$$= 1 + 2 + 2\lceil \tfrac{p}{2} \rceil + \cdots 2\lceil \tfrac{p}{2} \rceil^{h-1}$$

$$= \boxed{1 + \frac{2\left(\lceil \frac{p}{2} \rceil^{h} - 1\right)}{\lceil \frac{p}{2} \rceil - 1}}$$

(v) Min no. of keys in B tree with order p & height h

$$= 1 + 2(\lceil \tfrac{p}{2} \rceil - 1) + 2\lceil \tfrac{p}{2} \rceil (\lceil \tfrac{p}{2} \rceil - 1) - \cdots - 2\lceil \tfrac{p}{2} \rceil^{h-1} (\lceil \tfrac{p}{2} \rceil - 1)$$

$$= \boxed{1 + 2\left(\lceil \tfrac{p}{2} \rceil^{h} - 1\right)} \times \left(\lceil \tfrac{p}{2} \rceil - 1\right)$$

(vi) Min no. of keys in B+ Tree with order p & Height h

$$= 2\lceil \tfrac{p}{2} \rceil^{h-1} \left(\lceil \tfrac{p}{2} \rceil - 1\right)$$

(Consider only last level)

Max Nodes in B/B+ Nodes

$$(p^{h+1}-1)/(p-1)$$

Max Keys in BTree

$$(p^{h+1}-1) \times (p-1)$$

Max Keys in B+ Tree

$$p^h (p-1)$$

B Tree :-

p : max cp per Node
h : 0, 1, ---

$$\left\{2\left\lceil\frac{p}{2}\right\rceil^h - 1\right\} \leq \text{keys} \leq \{p^{h+1}-1\}$$

$\Theta\left(\frac{p}{2}\right)^h$ ... $\Theta(p^h)$

p^h : p

$$\log_p^{(n+1)-1} \leq \text{height} \leq \log_{\frac{p}{2}}^{\left(\frac{n+1}{2}\right)}$$

$\Theta(\log_p n)$ ... $\Theta\left(\log_{p/2}^{n/2}\right)$

Height of BTree for n keys and order p :-

a) Min Height :- [max keys per Node] / Max possible fill factor

$$n = p^{h+1} - 1$$

$$h = \log_p^{(n+1)-1} \approx \Theta(\log_p n)$$

b) MaxHeight :- [Min keys per Node]

$$n = 2\left\lceil \frac{p}{2} \right\rceil^{h} - 1$$

$$h = \log_{\lceil p/2 \rceil}\left(\frac{n+1}{2}\right) \approx \mathcal{O}(\log_{p/2} n/2)$$

Q How many min/max keys and nodes in BTree/B+Tree with order P [Max Possible keys per Node] is 5 and Level (L): 4?

fill factor:-

Root	Min	Max
Bp	2	6
Keys	1	p=5

OtherNode	Min	Max
Bp	$\lceil \frac{p}{2} \rceil = \lceil \frac{6}{2} \rceil = 3$ ← p+1=6	
Keys	2	p=5

level	Min nodes	min Bp	Min keys
Root (1)	1	2	1
2	2	2*3	2*2
3	6	6*3	6*2
4	18	—	18*2

=27 Min Nodes In B/B+ Tree

= 53 Min keys in BTree

= 36 Min keys in B+Tree (only last level for B+tree)

map Nodes	Map Bp	Map Key
1	6	5
6	6*6	6*5
36	36*6	36*5
216	—	216*5

=259 Max Nodes in B/B+ tree

= 1295 Map keys in B Tree

1080 Map keys in B+ tree

Q How many min/max keys and nodes are in B/B+ tree with order p : b/w 1 to 2p keys for Root
b/w p to 2p keys for other nodes
order p : 5 and level : 4 tree?

Sol:

Root	min	max
Bp	2	11
keys	1	2p=10 (p=5)

Others	min	max
Bp	6	11
keys	p=5	2p=10

Level	min Node	min Bp	min keys	max Node	max Bp	max keys
(Root) 1	1	2	1	1	11	10
2	2	2*6	1*5	11	11*11	11*10
3	12	12*6	12*5	121	121*11	121*10
4	72	72*6	72*5	1331	1331*11	1331*10

Workbook
Q30, 31

Bulk Loading B+ Tree

[construction of B+ Tree in Bottom Up Approach]

Leaf Node to Root Node.

Procedure-

1) Sort the keys which are required for the index of B+ Tree.

2) Design leaf nodes
(Distribute keys (we already know keys) into leaf Nodes either • min keys per nodes or • max keys per node.)

[# of leaf Nodes]

for min. no. of index nodes.

for Max no. of index nodes

[Nodes filled as least as possible
" " as most as possible

Initially given conditions

3) design internal nodes

if x nodes at level l (decided at time of step(2) design)

then level (l-1) must store x child pointers

distribute x Cp into Nodes

either Min Cp per Node or

Max Cp per Node

until Root

Eg Design B+ Tree for 25 keys with order p=5 [Max ptrs per Node]

→ what is min. no. of indexnodes.
→ what is min. levels of index.
→ what is min. I/o cost.

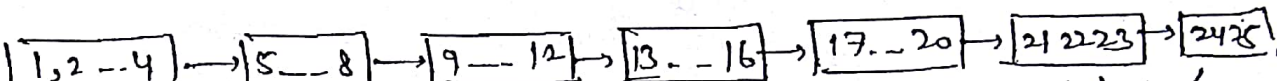
Sol- For this we have to get the maximum filled factor.

Max fill factor [Max Bp = 5 (Given) → Max keys = 4]

We can store max 4 keys in a node and min $= \lceil \frac{p}{2} \rceil - 1$

$= \lceil \frac{5}{2} \rceil - 1$

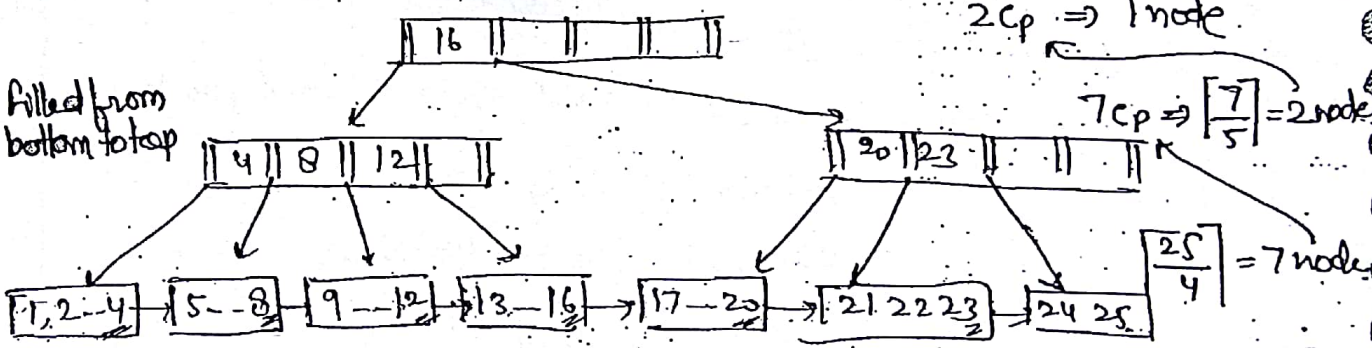
$= 3 - 1 = 2$ keys



because here we can't keep only 1 key because min key in a node can be 2 keys

No. of leaf nodes = $\lceil \frac{25}{4} \rceil$ = 7 nodes

Max 4 keys in each node



with $p = 5$ [Max ptrs per Node]

→ Max index Nodes ?

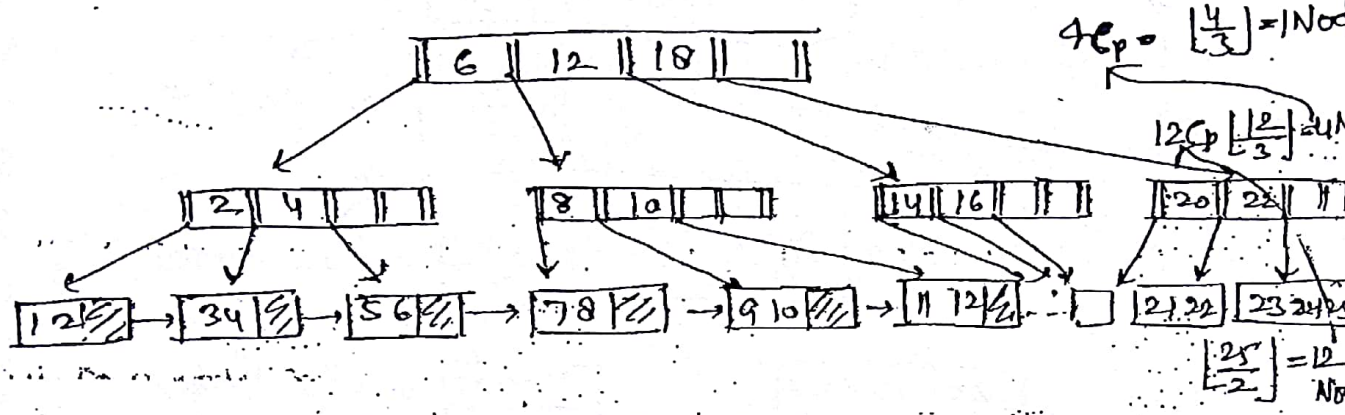
→ Max Levels of index?

→ Max I/O cost?

Sol:-

Min filled factor — Min B_p : $\lceil \frac{p}{2} \rceil = 3$ → Min keys = 2

Max B_p : $p = 5$ → Max keys = 4



Workbook

Q.32

20,000 records [dense B^+ Tree index]

→ Means now we have 20,000 keys to be stored as #keys = #records

Block size: 1000B

ptr: 10B

search key: 40B

a). Nodes filled as most as possible

$p * Bp + (p-1)keys \leq Block\ size$

$p * 10 + (p-1)\,40 \leq 1000\ B$

$50p \leq 1040\ B$

$p = \left\lfloor \frac{1040}{50} \right\rfloor = 20$

# of levels of index: 4

Min index Nodes: 1053 + 53 + 3 + 1

= 1110

Min I/O Cost = 5 block.

3 Cp — 1 Node

53 Cp — $\lceil \frac{53}{20} \rceil$ = 3 Nodes

1053 Cp — $\lceil \frac{1053}{20} \rceil$ = 53 Nodes

20,000 key — $\lceil \frac{20000}{19} \rceil$ = 1053

b) Nodes filled as least as possible

$\lceil \frac{P}{I} \rceil$ = 10 ptrs or 10 Bp (min.)

① 2 BP — 1 Node

② 22 Bp — $\lceil \frac{22}{10} \rceil$ = 2 Nodes

③ 222 Bp — $\lceil \frac{222}{10} \rceil$ = 22 Nodes

④ 2222 Bp — $\lceil \frac{2222}{10} \rceil$ = 222 Nodes

⑤ 20,000 keys — $\lceil \frac{20000}{9} \rceil$ = 2222 Nodes

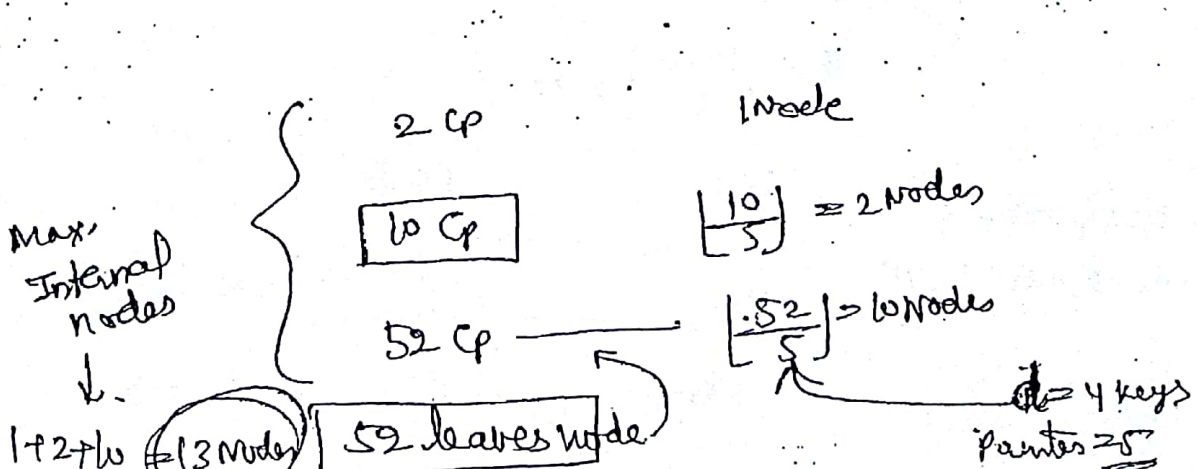
Max levels : 5

Max index blocks = 2222 + 222 + 22 + 2 + 1

= 2469

Max I/O cost : 6 Blocks access.

Q33 52 leaves and order = 4



MODEL QUESTIONS

1) Find best possible order (P) of B/B+ Tree Node?
   given blocksize, Bp, Rp, key

2) if # of levels given
   find min/max keys/Nodes in B/B+ Tree.
   [Root to Leaf] [Top to Bottom design]

3) if # of keys given
   find min/max Levels/Nodes of B+ Tree
   [Bottom Up Approach] (Bulk loading B+ tree)

4) if # of keys given to find min/max levels
   of B Tree Construct Top Down Structure.

Q # of keys: 600
How many minimum levels of B Tree Index with order p is 5? (p: Max Cp)

Level	Max Nodes	Max Bp	Max keys
1	1	5	4
2	5	5*5	5*4 → 24 keys over
3	25	25*5	25*4 → 124 " "
4	125	125*5	125*4 → 624 " "

max keys for 3 levels

Stop as we have 600 keys

max keys for 4 levels

Any no of keys greater than 124 keys & less than 624 keys, it will lead to 4th level.

So Min 4 levels must for given 600 keys.

Q # of keys = 600

How many max levels of BTree with order P=5 (P is max Cp)

min $\lceil \frac{P}{2} \rceil$ = 3 Bp. or 2 keys

level	min nodes	min Bp	min keys
1	1	2	1
2	2	2*3	2*2
3	6	6*3	6*2
4	18	18*3	18*2
5	54	54*3	54*2
6	162	162*3	162*2 → 486 min
7	486	486*3	486*2

Min keys required for 7 levels = 1457 keys

for 600 key Max Possible level of BTree = 6 levels.

~~even if we have 1456 keys then also we will not go for 7th level~~ we must be at 6th level.

∴ for 600 keys, max possible level of Btree = 6

JOIN Algo

a) Nested Loop Join Algo

b) Block Nested Loop Join Algo

Nested Loop Join Algo:

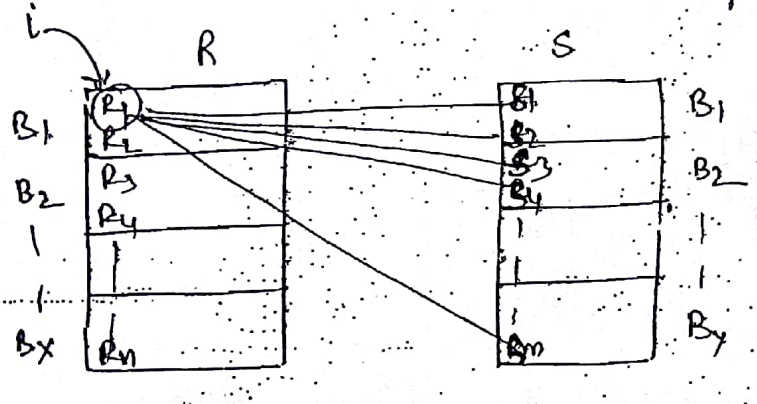
Relation R with n tuples occupies X blocks

Relation S with m tuples occupies Y blocks

R ⋈ S ⇒ for (i=0; i≤n ; i++)

for (j=1; j≤m ; j++)

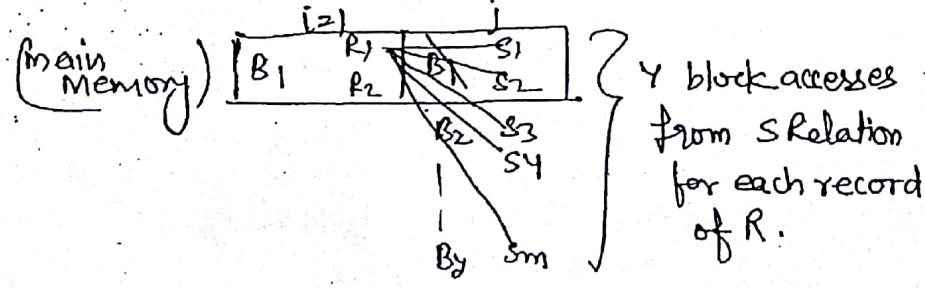
compare (R_i, S_j)



When all js from R_1 to R_m are completed for $i = R_1$ then only $i = R_2$ will start.

(*) More i/o cost required if Main Memory space allocated for join is very less.

i] Assume 2 Blocks of MM allocated for join [At any time one block of Records of R & one block of records of S can store in MM]



Now for each block of R_2 all y blocks of S are brought in MM.

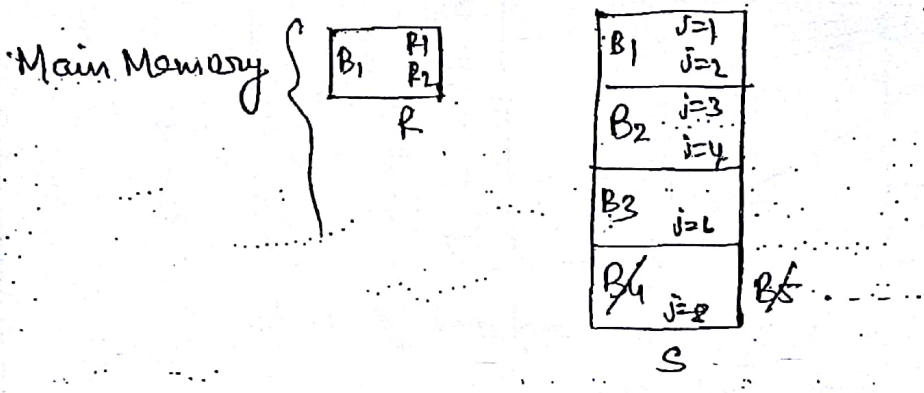
So, Y blocks are called n times

X blocks of R are called one time each.

So, R ⋈ S access cost : $(X + n * Y)$ blocks.

similarly, S ⋈ R access cost : $(y + m * x)$ blocks.

2] Assume 5 blocks of Main Memory used to store R & S data for join



for first block of R, All blocks of S has to be accessed or Y blocks of S accessed

but for every next block of R, we will already have first 3 blocks in mm.

So,

R ⋈ S Access Cost : $(X + Y + (n-1)(Y-3))$ blocks

X blocks of R

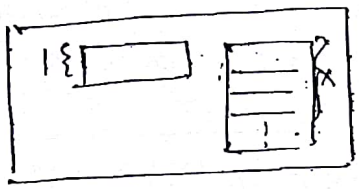
Y blocks of S for first block of R

Y-3 blocks of S for remaining (n-1) blocks of R.

S ⋈ R Access Cost : $(Y + X + (m-1)(X-3))$ Blocks

3] How many min MM blocks required so that can join R & S by accessing all blocks of R & S only once from Secondary Mem to Main Mem?

$R \bowtie S$ / $S \bowtie R$



Min Main Memory space $[\min(X,Y)+1]$ block

In this case, No disadvantage of Nested Loop Join Algo

Here we will put that relation from R & S which has min. tuples, in the main memory and other table will be accessed one by one.

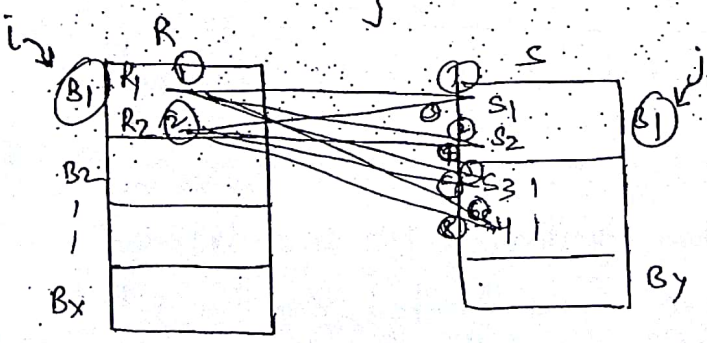
Thus i/o cost will be X+Y blocks.

Block Nested Loop Join

Relation R occupied in X blocks

Relation S occupied in Y blocks.

```
R ⋈ S ⇒ for (i=1; i≤X; i++)
        {
            for (j=1; j≤Y; j++)
            { Join (ith Block of R Records,
                    jth Block of S Records)
            }
        }
```



i=1

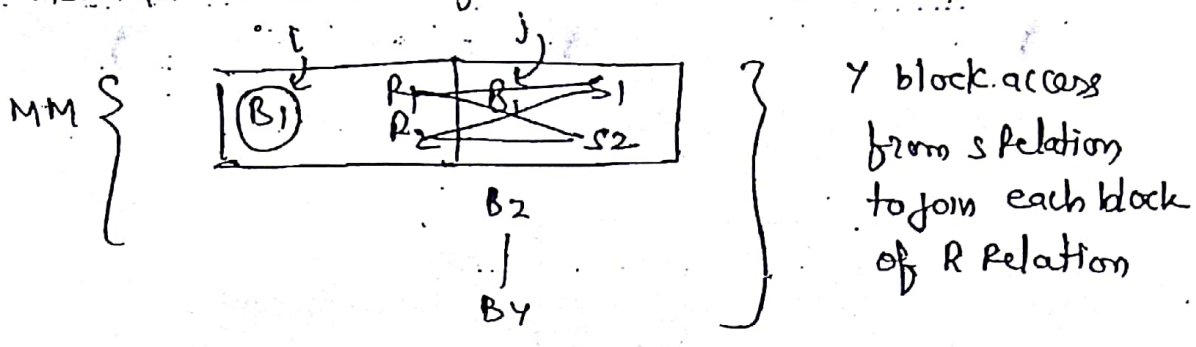
Firstly, R_1 — S_1, S_2

then R_2 — S_1, S_2

then i becomes 2

So, block is joined with block instead of single single records from R & S as in prev. case.

1]. Assume 2 Blocks of MM Allocated for Join -



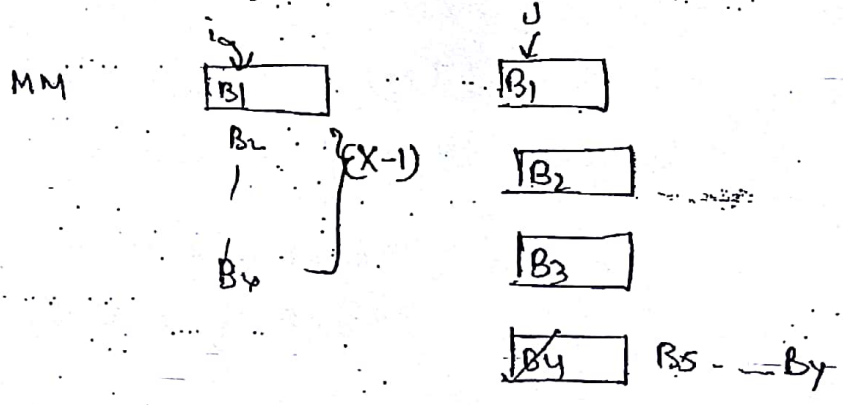
R ⋈ S Access Cost = $(X + X*Y)$ blocks

S ⋈ R Access Cost = $(Y + Y*X)$ blocks.

(*) So, In both the Algo, execution time will remain same i.e. $O(n*m)$. Only Access cost will differ

The CPU time or no. of comparisons will still remain the same.

2] Assume 5 blocks of MM allocated to store R&S data for Join



R ⋈ S Access Cost : $[X + Y + (X-1)(Y-3)]$ blocks

Advantage :- Less Access cost compared to Nested Loop Join Algorithm if MM space is Limited

(If MM space is not limited, then there will be no difference b/w the two Algos)

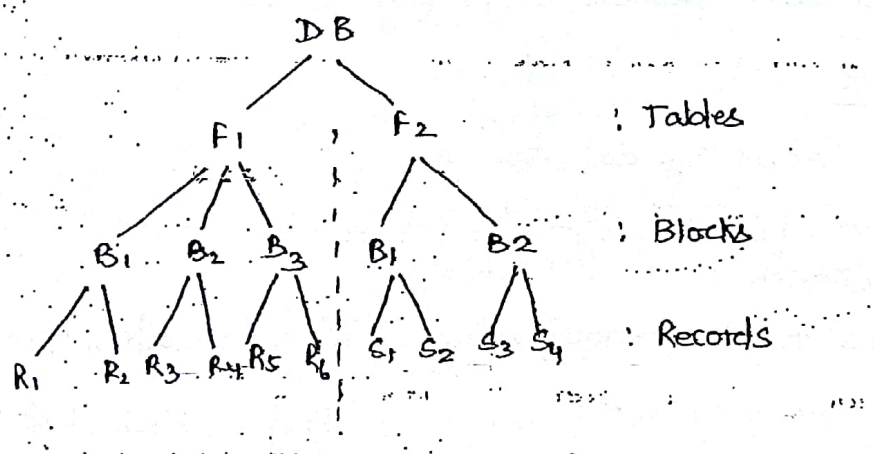
TRANSACTION & CONCURRENCY CONTROL

TRANSACTION: set of logically related operations to perform unit of task.

In OS terminology ⇒ ~~OS~~ Process
In DBMS " ⇒ Transaction
} Both are logically equivalent

OS ⇒ Data item

DBMS ⇒ Shared Resource
(database element required to access multiple transactions is called data item)



DBMS Concurrency Control over record level.
Here advantage is More degree of concurrency.

OS concurrency control over file level.
Drawback: Less degree of concurrency.

For accessing a file for using we had to block all the records (lit 1000 records in one file), so at record level concurrency control, more difficult to handle resources.

At file level, easy to handle resources.

→ There are more no. of records and we have to apply locks at each record, that's why it is difficult to handle resources at the record level.

⊛ DBMS allows multilevel Concurrency Control

Consider different transactions -

i) T_1: Update R_2 ⇒ more advantageous to lock(R_2)

ii) T_2: Update S_1 S_2 S_3 S_4 lock(R_2)

iii) T_3: Update R_3 R_4 lock(R_2 of R_1)

Goal of DBMS Concurrency controller:
(More degree of concurrency with easy to handle resources)

→ To avoid the disadvantages of using only file level or record level concurrency control,

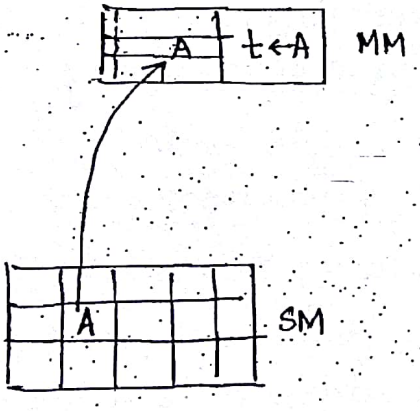
DBMS provides multilevel concurrency control.

Main Operations

Read(A): Accessing data item "A"

→ From DB file to programmed Variable
(Sec. Mem.) (Main Mem.)

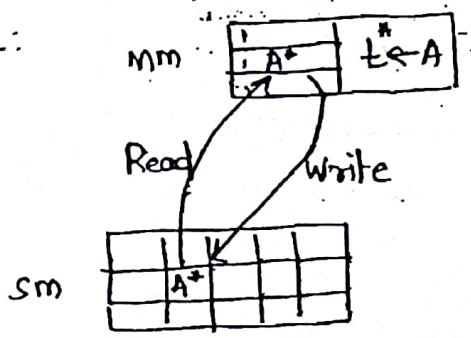
inorder to use previous value of "A" in transaction logic.



1. Find SM block which contain data item A.
2. Data item A is brought into Main Memory stored in a block where other data items are also stored.
3. Find address of data item A & copy into programmed Variable (i.e. t)

→ t is now used for transaction logic and original value of A is stored in Sec Memory.

Write (A): Modification of data item A into DB file (SM block)



Here also first item A is brought from SM to MM

1. } of Read(A) are
2. } followed.
3. }
4. Modified block of MM is replaced into SM.

(A) SQL Query converted into transaction (user point of view)

```
update table R set A = A+10
        ⇓
SQL Compiler && Parser
        ⇓
Transaction(Ti)
{ Read(A)
  A = A+10
  Write(A)
  Commit : [Transaction executed successfully]
```

copyright©theorypoint.com The Best Gateway Of Success

(A) consider query:

```
Update table R set A=10;
        ⇓
Transaction(Tj)
{ W(A)
  Commit
```

Here we not require previous value of A. So no requirement of Read(A) operation.

Access cost ← remains same as that of prev. case because read operation will be performed while writing

BLIND WRITE

Trans Tj writes "A" without R(A) previously by Tj

④ eg Blind Write

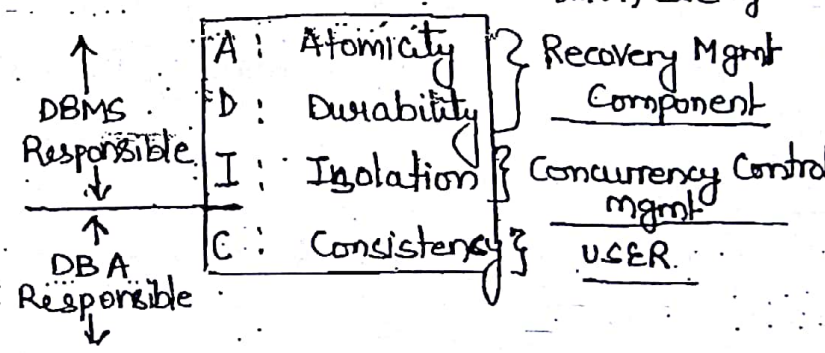
T_i	T_j
R(A)	
	W(A) ← Blind Write
	R(B)
	W(B)
Blind Write → W(B)	

→ Read value of Ti and Tj are independent. Both have their own set of local data on which processing is done. Both are different processes.

ACID Properties:

To preserve integrity (correctness) of the transaction, it should satisfy ACID Properties (or Rules):

④ To prove transaction is successful, it should satisfy all ACID rules. Even if one rule is not followed, it will not be considered correct. (failure of integrity ⇒ incorrect result ⇒ error)



Atomicity

[Execute all operations of transaction including commit]

(OR)

[Execute None of the operations of transaction]

④ Atomicity fails when some (but not all) of opers are executed before termination

eg $T_i \Rightarrow$ [Trans 500Rs from A to B]

```
begin Ti
    R(A)
    A = A - 500
    W(A)
failed ──→
    R(B)
    B = B + 500
    W(B)
    Commit
```

(DB file)

A = ~~2000~~	1500
B = 3000	

Incorrect Data in DB file becoz of Atomicity failure.
(value of A is updated but value of B is not)

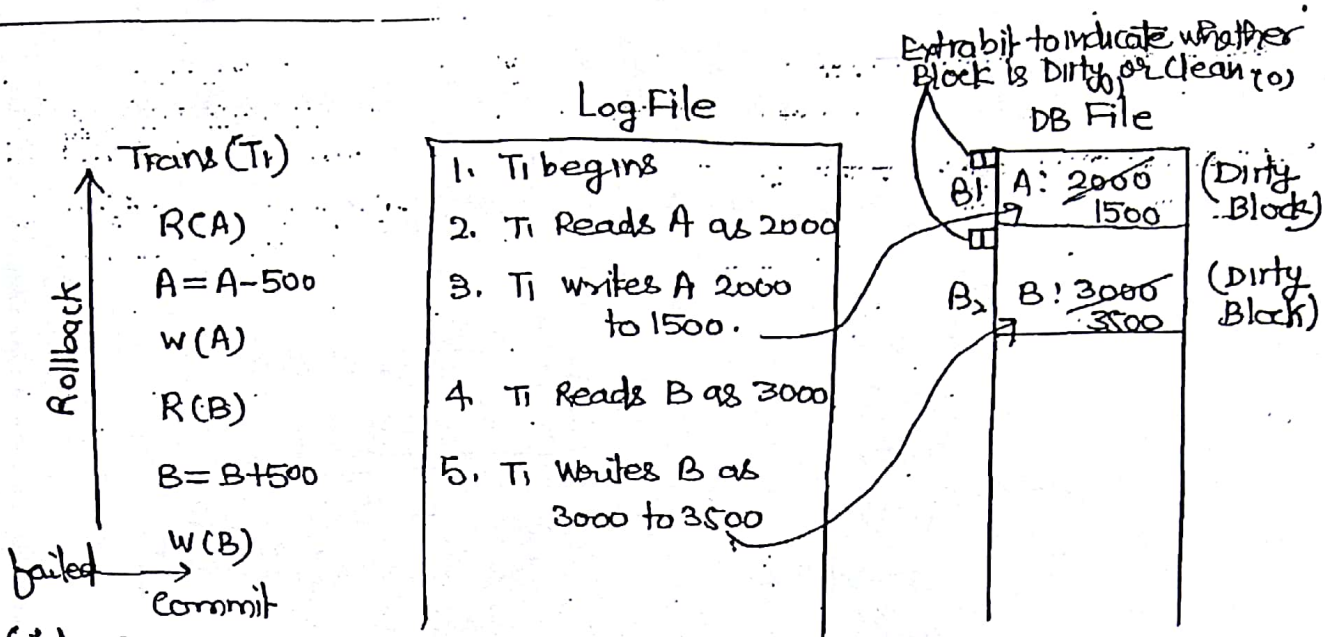
⑤ Recovery Mgmt. Component (RMC) takes care of Atomicity.
It Rollbacks the transaction if Transaction failed anywhere before commit.

Rollback [Abort]: Undo modifications of the DB File which are done by failure transaction.

A = 1500	2000 (original value restored)
B = 3000	

So, here previous state of DB is achieved.
or No operation is executed before termination
(is same as)
Rollback before termination.

⑥ To restore old (original) value of data item, ~~some variable~~ this old value needs to be stored somewhere!
Transaction Log :- DB file (log file) maintained in Disk which records all activities of Transaction.

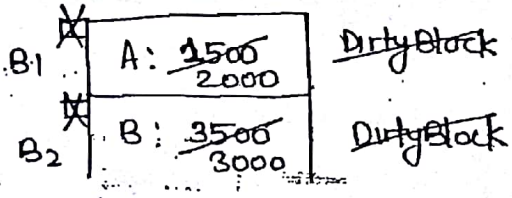


(if trans failed here then also the trans will be rolled back.)

Dirty Block:
DB Block which is modified by an uncommitted trans.

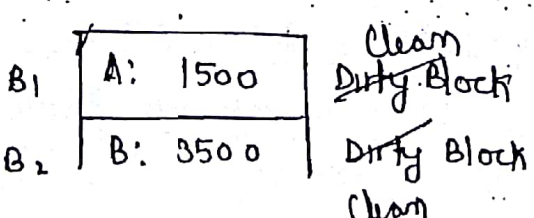
Changes occurred when trans failed before being committed :-

Undo operations: DB File modifications becoz of Trans Rollback.



Suppose, trans. committed successfully. Then changes occurred here also →

Redo operations: DB File modifications becoz of trans Commit



④ Transaction execution is a task of less than 1 second.

T_1 ... Commit → Redo

T_2 ... Commit → Redo

T_3 Commit → Redo

⋮

→

Cost of Redo is very high.

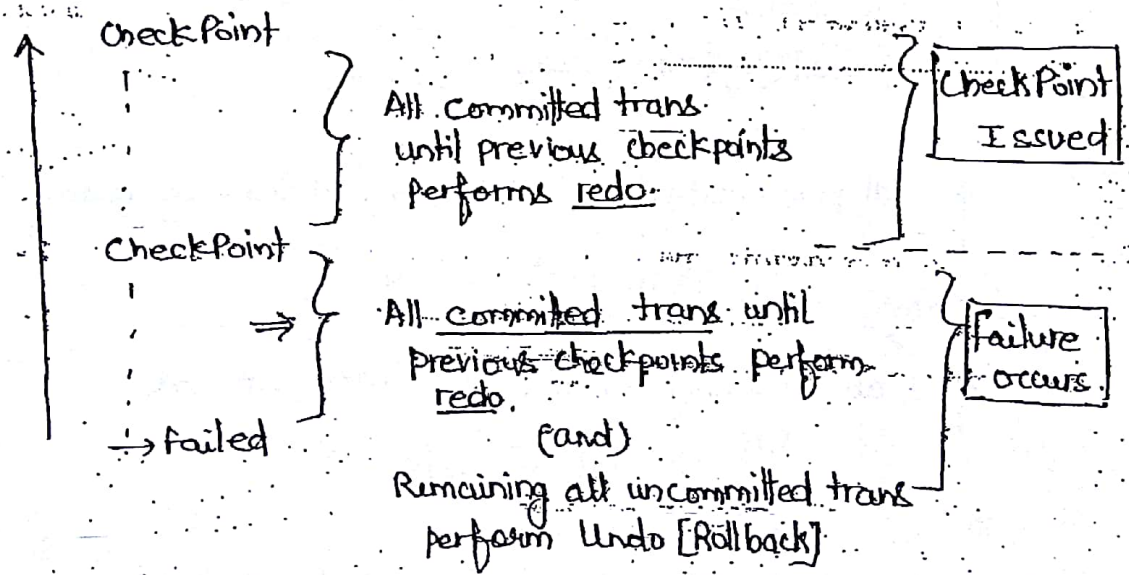
If after every commit we perform Redo operation, then it becomes overhead. as overall cost due to redo op^n becomes much more

For this, DBMS implements Checkpoints.

CheckPoint:

[Trans. consistency checkpoint]

runs some interval of Time



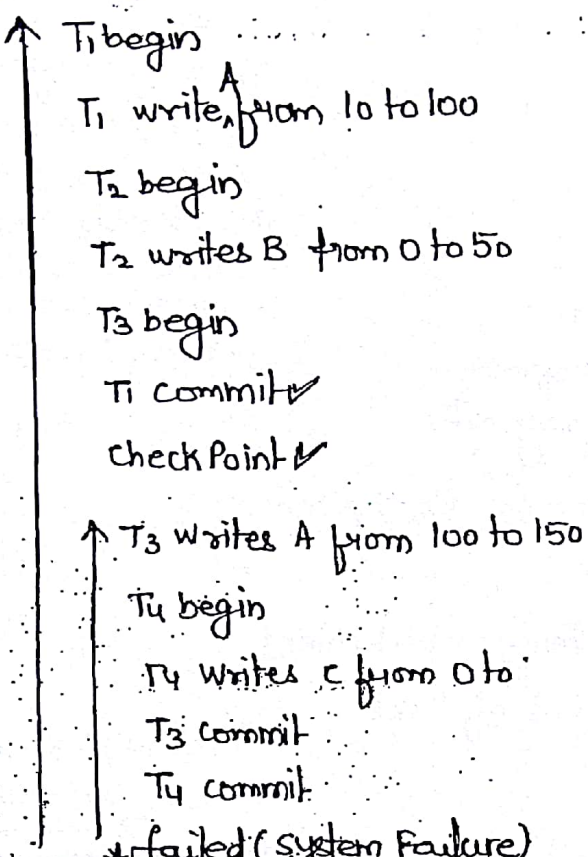
Q What will happen when i) CheckPoint is issued (or) ii) Failure occurs.

(i) All committed trans until previous checkpoints paforms redo.

(ii) All committed trans until previous checkpoints perform redo.

Remaining all uncommitted trans perform Undo [Rollback].

Q. Translog:



what operations will be executed because of failure?

Sol:

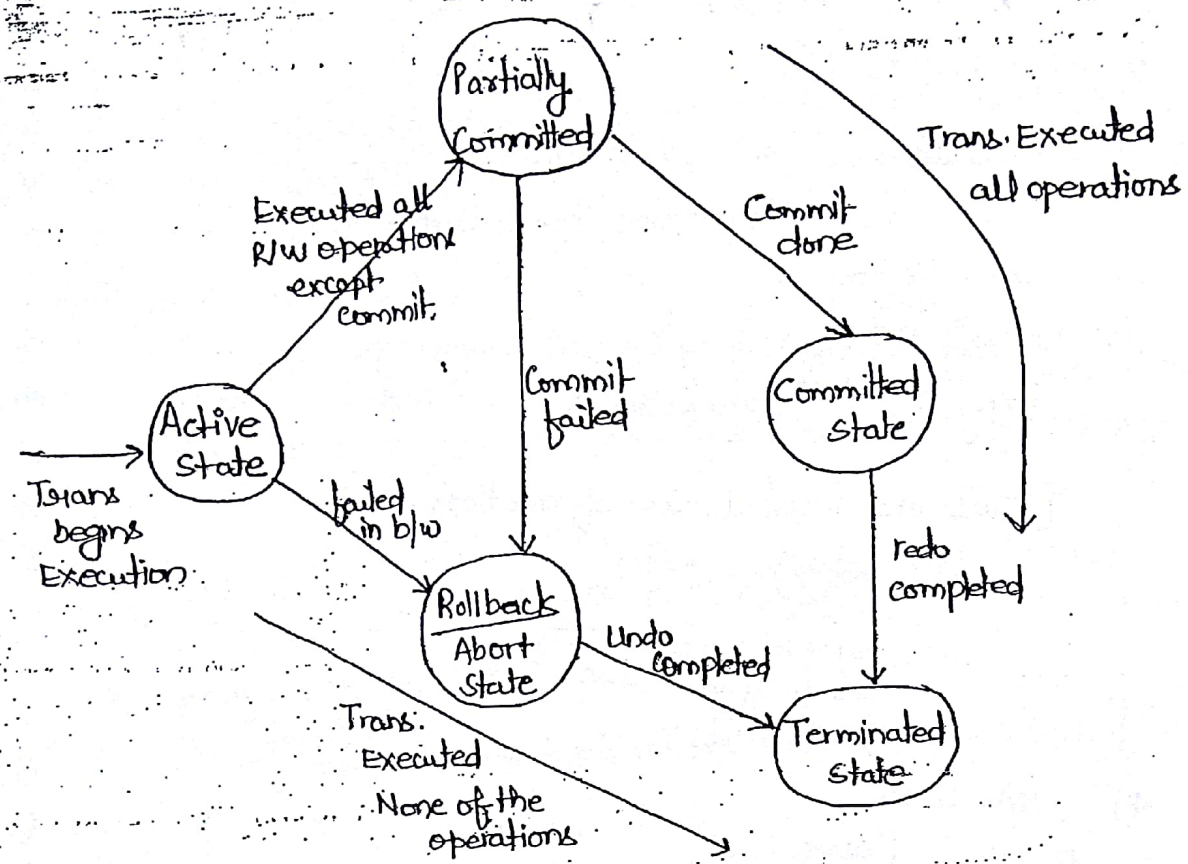
↑ till prev. checkpoint, all committed trans are redo.

So, T_3 T_4 perfoms redo.

↑ all remaining uncommitted trans are undo.

So, T_2 perfoms undo

* T_1 is already redo at the checkpoint.

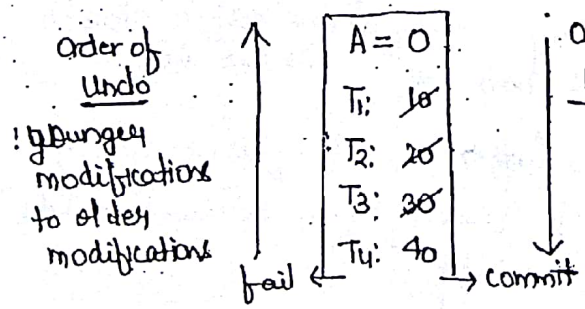


Durability [Recovery Mgmt Component responsible for Durability]

Trans should able to recover under any case of failure (and)

DB updations must be persistent.

There is proper way to perform updations or undo/Redo



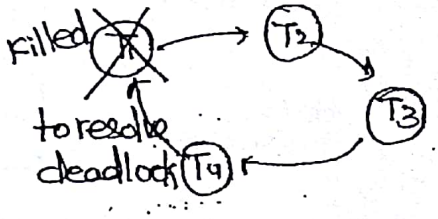
Possibility of Trans Failure :-

1) Power Failure

2) S/w Crash
   a) OS Restarted
   b) DBMS s/w Restarted

3) OS/DBMS concurrency control components may kill the transaction.

[Trans. are killed in case of deadlock.



4) H/w Crash
   [Disk Crash]

(*) Concurrency Control: The DB should remain consistent in case of concurrent trans.

To maintain consistency, the system must be able to recover from any failure.

(*) Power Failure
    S/w Crash
    OS/DBMS Concurrency control kills transaction

↕ Can be recovered from log file stored in the disk.

But if disk crashes (where all log files are stored), then it is the duty of Recovery Mgmt. to maintain data (This comes under Durability).

For recovery in case of Disk failures, Durability is assured through RAID Architecture of Disk Design.

RAID (Redundant Array of Independent Disks)

Here we have diff. types of RAID —
- RAID 0 → only one disk
- RAID-1 → 2 independent disk

(*) In RAID Architecture, we have independent disks & if we write on one disk, then we have to write the data on all the independent disks. Now, if any disk fails, then all the other independent disks would still be working

ISOLATION [***Concurrency Control Component]

Concurrency Controller comes into picture when

[Two or more transactions executing concurrently]

then

[it should avoid concurrent execution which leads to incorrect result]

Schedules :- order or (Sequence) of operations of two or more transactions to be executed.

↳ Time order sequence of two or more transactions.

The concurrent transactions execute in interleaved fashion.

eg.

	T_1	T_2	T_3
1.	$r_1(A)$		
2.		$w_2(A)$	
3.	$w_1(A)$		$w_3(A)$
4.			
5.		$w_2(B)$	

S : $r_1(A)\ w_2(A)\ w_1(A)\ w_3(A)\ w_2(B)$

Serial Schedule

After termination of current trans. then only allowed to begin other trans.

eg $T_1 \Rightarrow$ Transfer 500 from A to B $[R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)]$

$T_2 \Rightarrow$ disp total balance of A $[R_2(A)\ R_2(B)\ display(A+B)]$

	T_1	T_2
2000:	$R_1(A)$	
1500:	$W_1(A)$	
3000:	$R_1(B)$	
3500:	$W_1(B)$	
		$R_2(A)$: 1500
		$R_2(B)$: 3500
		disp(A+B): 5000 (correct)

$T_1 \rightarrow T_2$ serial schedule

	T_1	T_2
		$R_2(A)$: 2000
		$R_2(B)$: 3000
		disp(A+B): 5000 (correct)
2000:	$R_1(A)$	
1500:	$W_1(A)$	
3000:	$R_1(B)$	
3500:	$W_1(B)$	

$T_2 \rightarrow T_1$ serial schedule

Advantage

Result of every serial schedule is correct output (i.e. no inconsistency possible if transactions executing serially) because at a time only one transaction is accessing all the resources.

Disadvantage - Less degree of concurrency

Less throughput (No. of transactions executed per unit time)

To avoid these disadv., we go for concurrent schedules.

Concurrent Schedules -

Trans allowed to execute in non serial order

[Simultaneous / concurrent / Interleaved order]

(A) Every serial schedule is also concurrent schedule

Drawback: Concurrent Execution may cause inconsistency

T_1	T_2
2000: $R_1(A)$	
1500: $W_1(A)$	
	$R_2(A)$: 1500
	$R_2(B)$: 3000
	Disp(A+B): 4500 Incorrect
3000 $R_1(B)$	
3500 $W_1(B)$	

Since it does not satisfy Consistency criteria.
∴ Incorrect Schedule

T_1	T_2
2000: $R_1(A)$	
1500: $W_1(A)$	
	$R_2(A)$ 1500
3000 $R_1(B)$	
3500 $W_2(B)$	
	$R_2(B)$ 3500
	Disp(A+B) 5000 Correct

Since it satisfies consistency criteria
∴ Correct Schedule

Initial:

A	2000
B	3000

After T_1 & T_2 execution, expected result:

A	1500
B	3500
disp(A+B)	5000

(Correct Result)

Duty of Concurrency Controller is to not allow the execution of this schedule.

Concurrency controller should allow the execution of this schedule.

But to decide that a solution is correct / Incorrect it should have to compare that result with the correct result.

Consistency Criteria

(A) disp(A+B) before exec of T_1 & T_2 = disp(A+B) after T_1 & T_2

We know that the result of serial schedule is always correct. So compare result of concurrent schedule with serial schedule.
Here arises concept of **Isolation**

Concurrent Execution b/w two or more trans. result must be equal to result of any serial schedule.

④ In order to check that result of serial schedule = result of concurrent schedule, & compare we observe, the order of Read & Write operations of each transaction.

Serial Schedule

	T_1	T_2
2000	$R_1(A)$	
1500	$W_1(A)$	
3000	$R_2(A)$	
3500	$W_2(A)$	
		$R_2(A)$ 1500
		$R_2(B)$ 3500
		disp(A+B) 5000

T_1	T_2	
	$R_2(A)$	2000
	$R_2(B)$	3000
	disp(A+B)	5000
$R_1(A)$		2000
$W_1(A)$		1500
$R_2(A)$		3000
$W_2(A)$		3500

Concurrent Schedule①

T_1	T_2	
2000	$R_1(A)$	
1500	$W_1(A)$	
	$R_2(A)$	1500
	$R_2(B)$	3000
	disp(A+B)	4500
3000	$R_1(B)$	
3500	$W_1(B)$	

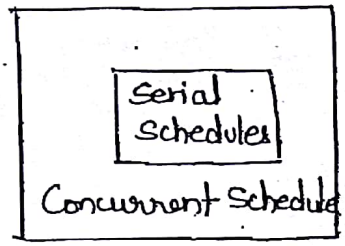
Serializable Schedule

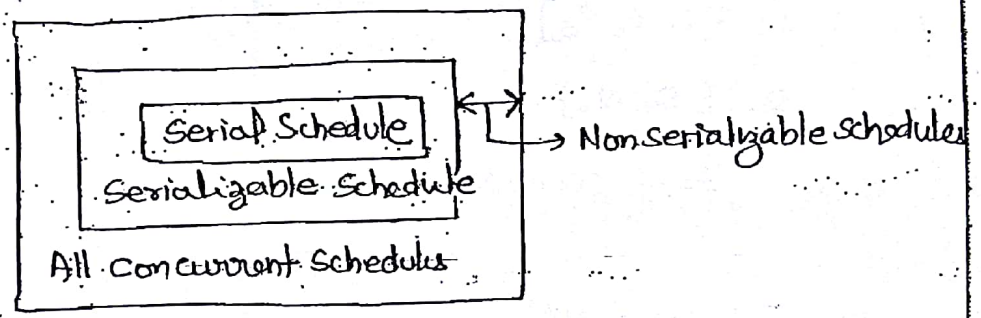
Schedule for which concurrent execution of 2 or more transactions is equal to result of any serial schedule are called searializable schedules.

Isolation ≡ Serializability

All serial schedules are serializable.

Concurrent schedules may/ may not be serializable





Q. $T_1, T_2, T_3 \ldots T_n$ are n Trans. # of possible serial schedules:

n!

Q. T_1: $R_1(A)$ $W_1(A)$ $R_1(B)$ $W_1(B)$

T_2: $R_2(A)$ $R_2(B)$.

# of concurrent schedules b/w T_1 & T_2 : ${}^6C_4 * {}^2C_2 = 15$

13 Non Serial + 2 Serial

Both T_1 & T_2 are allowed to execute in interleaved fashion. but operations of T_1 only or T_2 only must be executed in same order only.

$R_1(A)$ — $W_1(A)$ — $R_1(B)$ $W_1(B)$

# of ways in which 4 operations can be placed in 4 positions out of 6 = ${}^6C_4$

# of ways 2 operations can be placed in remaining 2 positions = ${}^2C_2$

136

Q T1 & T2

# of Concurrent Schedules = $^{n+m}C_n = \frac{(n+m)!}{n!\,m!}$

Q T_1, T_2, T_3 trans with n, m, p operations respectively

# of Concurrent schedule = $^{n+m+p}C_n \cdot {}^{m+p}C_n$

$= \frac{(n+m+p)!}{n! \cdot m! \cdot p!}$

Includes Serial schedules also.

Q (i) P: $[P_1, P_2, P_3]$

Q: $[Q_1, Q_2]$

# of Concurrent Execution: $^5C_2$

(ii) A: $a_1 a_2 a_3$

B: $b_1 b_2 b_3 b_4 b_5$

In how many ways we can merge A & B in a sorted array? (same as concurrent schedule)

order of $a_1 a_2 a_3$ or $b_1 b_2 b_3 b_4 b_5$ should be same

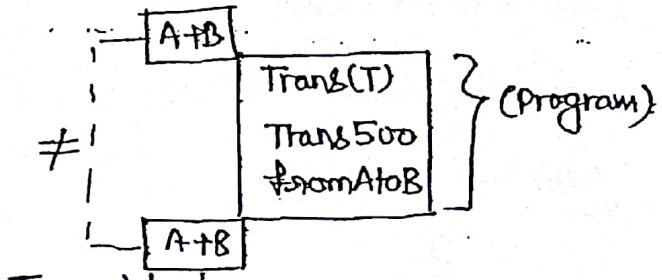
but they can come in interleaved manner.

So, # ways = $^8C_5 \cdot {}^3C_3$.

CONSISTENCY

[User] is responsible for consistency.

DB operations requested by user must be logically correct.



In Algo ⇒ Validity of an algo

In DBMS ⇒ Consistency of trans.

Inconsistent

A = 5000

T_1 (A)	T_2
5000 R(A)	
	R(A) 5000
	A = A − 1000
	W(A) 4000
A = A − 1000	
4000 W(A)	

since two times withdraw is taking place so result should be A = 3000

but the result came A = 4000

so, Incorrect Result.

But the result is not due to inconsistency

but result is incorrect becoz of isolation failure

① ACID Properties

② Problems due to Concurrent Execution

RW Problem

WR Problem

WW Problem

Irrecoverable Problem

Cascading Rollback Problem

Lost Update Problem

③ Schedule Classification :-

(a) Serializability :- ***

1] Conflict Serializable

2] View "

(b) Recoverability :- **

1] Recoverable ~~used~~ sched.

2] Cascadeless Rollback Recoverable.

3] Strict Recoverable

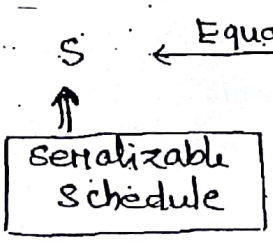
④ Concurrency Control Protocols

**1] Locking Protocols

2] Timestamp ordering Protocols

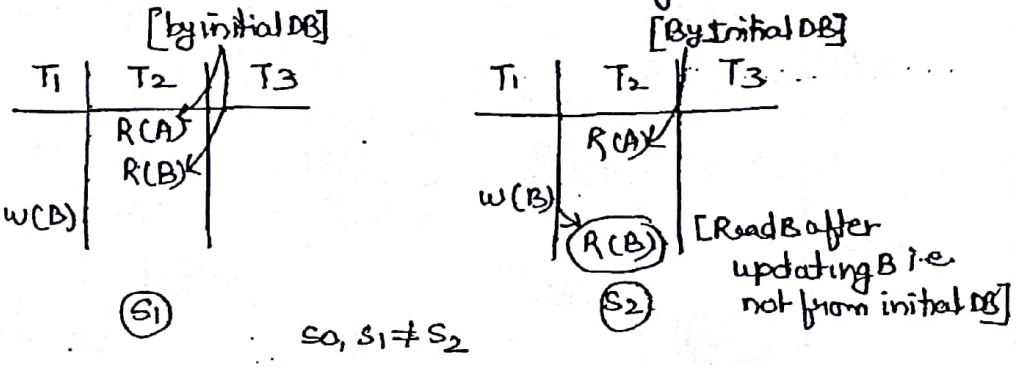
Serializable Schedule :-

Schedule S is serializable iff some serial schedule S' must be equal to schedule (S).



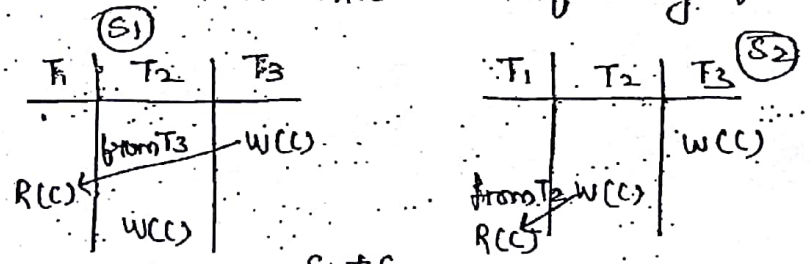
(*) S_1 & S_2 are equal schedules iff [View Equal]

1] Initial Read :- If T_i reads 'A' from initial DB in S_1 Then in S_2 also T_i should read 'A' from initial DB only.



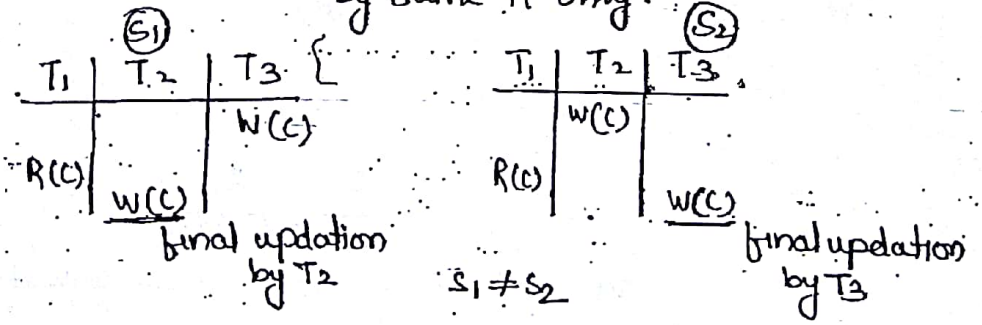
So, $S_1 \neq S_2$

2] Updated Reads :- If T_i reads A which is updated by T_j in S_1 then in S_2 also T_i must read 'A' which is modified by T_j only.



$S_1 \neq S_2$

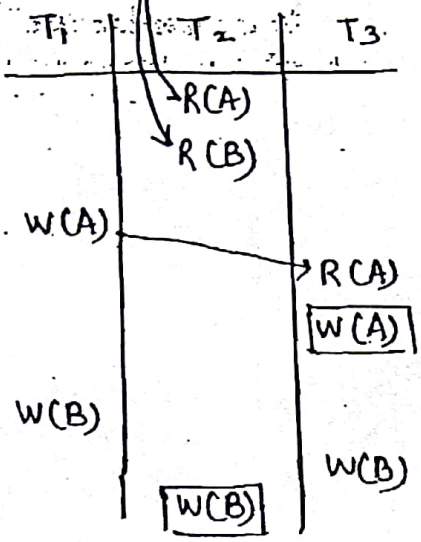
3] Final Write :- If T_i updated A finally in S_1 then in S_2 also A must be finally updated by same T_i only.

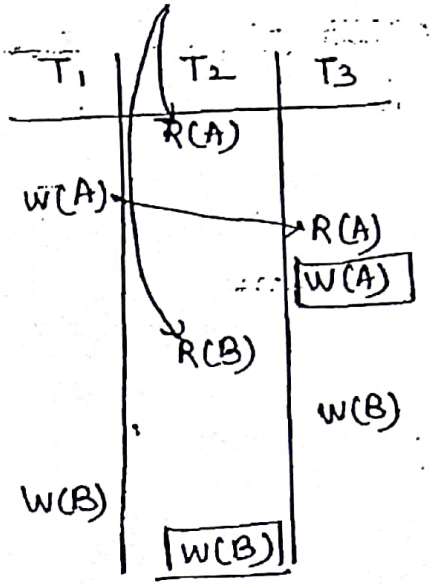


$S_1 \neq S_2$

All 3 conditions must be satisfied.

Q Test S_1 & S_2 equal or not.





Initial Read :-

S_1: R(A), R(B)] from initial DB

S_2: R(A), R(B)] from initial DB

Updated Read :-

S_1: T1 W(A) → T3 R(A)

S_2: T1 W(A) → T3 R(A)

Final Write :-

S_1: T2 W(B); T3 W(A)

S_2: T2 W(B); T3 W(A)

$S_1 \equiv S_2$

or S_1 & S_2 are view equal schedules.

Q = Test given schedule is Serializable or not?

To Test this, compare given schedule with serial schedule., if atleast one serial schedule is same as given sched. then it will be serializable.

1)

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

Not equal to $T_1 \to T_2$ serial
Not equal to $T_2 \to T_1$ serial
[Non Serializable Schedule]

T_1	T_2
R(A)	
W(A)	reads modified value of A
	R(A)
	W(A) final W(A) by T_2

$T_1 \to T_2$ serial

T_1	T_2
	R(A)
	W(A)
R(A)	not initial DB
W(A)	

$T_2 \to T_1$ serial

2)

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

A is finally updated by T_1

Not equal to $T_1 \to T_2$ serial
Equal to $T_2 \to T_1$ serial
[Serializable]

T_1	T_2
W(A)	
W(A)	
	W(A) final W(A) by T_2
	W(B)

$T_1 \to T_2$ serial

T_1	T_2
	W(A)
	W(B)
W(A)	not from initial DB
W(A)	

$T_2 \to T_1$ serial

3)

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

Not equal to $T_1 \to T_2$
Not equal to $T_2 \to T_1$
[Not Serializable]

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

$T_1 \to T_2$ serial

T_1	T_2
	W(A)
	W(B)
R(A)	not from initial DB (becomes updated read)
W(A)	
R(B)	

$T_2 \to T_1$ serial

Problems because of Concurrent Execution

WR Problem
RW Problem
WW Problem

} Result of schedule can be incorrect becoz simultaneous Read-Write/ Write-Read/ Write-Write operations over same data from different transaction.

To avoid inconsistency or incorrect result because of simultaneous Read-Write/ Write-Read / Write-Write conflict, operations over same data by different transactions, Concurrency controller should avoid/restrict non-serializable schedule.

(becoz WR, RW, WW simultaneously are allowed under serializable schedules:
Even if simultaneous WR, RW, WW exist in schedule, then also no problem if schedule is serializable)

(A) (Other method to avoid the problems is to don't allow these simultaneous RW/WR/WW operation in the transactions.
But this is not the efficient way as we want more degree of concurrency, so we opt for above method)

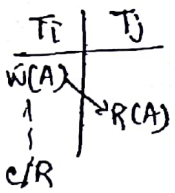
(A) To check whether WR / RW / WW Problem exist in the schedule or not, we have to prove that the schedule is non-serializable.

WR Problem

a) Non serializable schedule

(and)

b) simultaneous WR exists

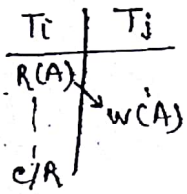


RW Problem

a) Non serializable schedule

(and)

b) simultaneous RW exists

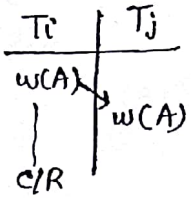


WW Problem

a) Non serializable schedule

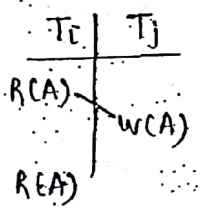
(and)

b) simultaneous WW exists



Unrepeatable Read Problem (RW Problem)

Some transaction Ti reading the same data two or more times In blw, some other transaction Tj, updated that data item leads to inconsistent / incorrect result.



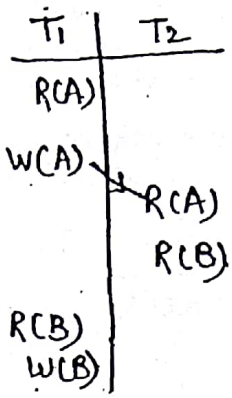
It is called RW Problem becoz -

There is possibility that Tj commit after W(A) & before R(A) then the problem definetely exist.

Also there is possibility that Tj commit after R(A) in Ti then there will be no problem.

Q. Test whether the given schedule ~~contains what~~ suffers from which problem ?

i)



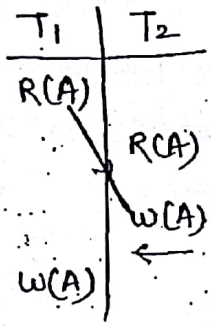
I) Not serializable

II) Simultaneous WR Problem exist

⇓

WR Problem

ii)



I) Not serializable

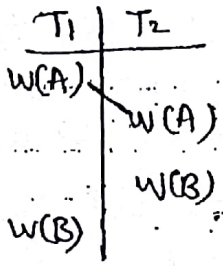
II) RW conflict

⇓

(RW Problem)

← Not W-W problem as T2 can commit here

iii)



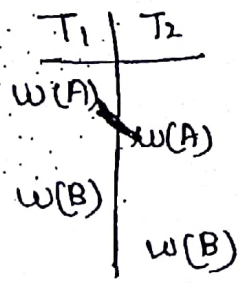
I) Not serializable

II) WW conflict

⇓

WW Problem

iv)



1) Serializable

2) WW conflict

⇓

No Problem

[result of schedule is correct]

Classification of Schedule - Recoverability

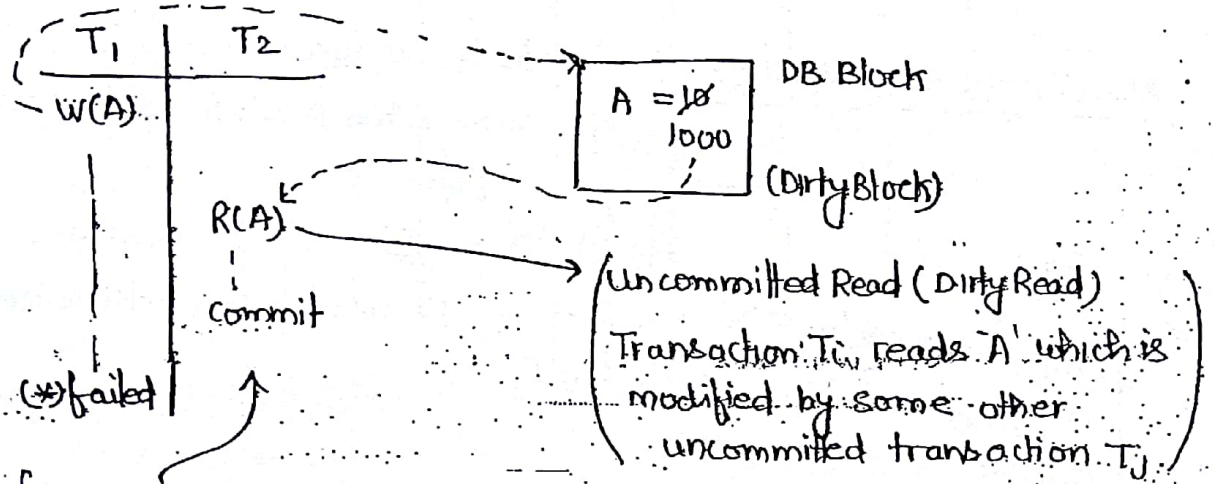
Recoverability classification goal to avoid:

- IRRECOVERABLE SCHEDULE
- Cascading Rollback Problem
- Lost Update Problem

} Can occur even if the schedule is serializable.

IRRECOVERABLE SCHEDULE

(Rollbacking of Committed Trans.)

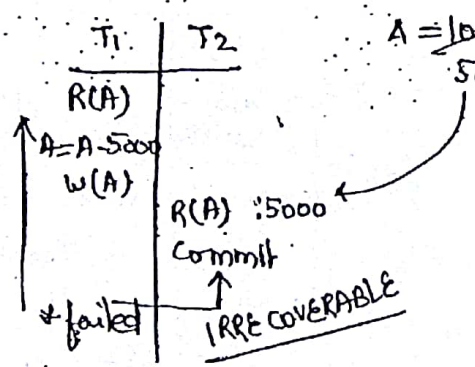


[Rollback of T_2 is required] which is not possible (IRRECOVERABLE SCHEDULE)

Suppose after commit of T_2, T_1 is failed then T_1 will be rolled back. In that case, the value read by T_2 becomes incorrect, so the result produced by T_2 will be incorrect (Inconsistency occurs). So, we also need to rollback T_2 but it can't be rolled back as T_2 is already committed (not possible).

Since o/p of T_2 is wrong so, inconsistent.

eg. A: Balance of Account



IRRECOVERABLE SCHEDULE:- Transaction T_i Reads 'A' which is modified by some other Trans T_j and T_i commit before commit / rollback of T_j.

T_j	T_i
W(A)	
|	R(A)
|	commit
C/R	

RECOVERABLE SCHEDULE:- schedule (S) is recoverable iff

a) No Uncommitted Reads in schedule (S)

(or)

b) If T_i reads A which is modified by T_j then commit of T_i must delay until commit or rollback of T_j.

T_j	T_i
W(A)	
	R(A): Dirty Read
C/R	(wait)
	commit

Q Test whether the given schedule is recoverable or not.

i)

T_1	T_2
W(A)	
	R(A)
	R_2 (rollback)
C_1 / R_1	

(Recoverable)

ii)

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A)	
	C_2
C_1	

Non serializable (RW Problem)

No dirty reads (Recoverable)

(iii)

T_1	T_2
W(A)	
	W(A)
	W(B)
W(B)	
	C_2
C_1	

Non Serializable (WW Problem)

(No Uncommitted Read) Recoverable

iv)

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	
C_1	
	C_2

Non serializable (WR Problem)

Dirty Read

wait for commit

Recoverable

v)

T_1	T_2
W(A)	
	W(A)
	R(A) ← No dirty read
	C_2
C_1	

Recoverable

Cascading Rollback Problem

Becoz of feilure some trans forced to Rollback some set of other transactions.

vi)

T_1	T_2	T_3	T_4	T_5	T_6
W(A)					
	R(A)				
	W(A)				
		W(A)	R(A)	R(A)	
				W(B)	R(B)
✝ failed					

Thus recoverable schedule does not avoid R-W/W-R/W-W Cascading Rollback Lost update Problem.

According to recoverability,

T_2 should wait for T_1

T_4 T_5 should wait for T_2

T_6 should wait for T_5

Suppose T_1 fails at given instance,

T_1 rollbacks cause T_2 rollback

T_2 rollback cause T_4, T_5 rollback

T_5 rollback cause T_6 rollback

T_3 is independent, so it is not rollbacked.

No. of rollbacks = 5.

No. of cascading rollbacks = 4

Disadvantage of Cascading Rollback

Wastage of CPU execution time and I/O Access Cost.

To avoid cascading rollbacks, the schedules used are -

Cascadeless Rollback Recoverable Schedule.

[Uncommitted Reads Not Allowed] → ∵ rollback was due to dirty Reads

T_i	T_j
$W_i(a)$	
↓R	$R_j(A)$

If trans T_i writes A, then some other trans T_j read(A) must be delayed until T_i commit / rollback.

Q. Test given schedule Cascadeless / Rollback Recoverable or not

1)

T_1	T_2
W(A)	
	W(A)
	W(B)
W(B)	
C1	
	C2

(W-W Problem)

2)

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A)	
	C2
C1	

(RW Problem)

3)

T_1	T_2
W(A)	
	W(A)
	R(A)
	C2
C1	

T_2 reads the T_2 modified data so no dirty read

4)

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	
C1	
	C2

Cascadeless Rollback Recoverable ← → Not Cascadeless Recoverable schedule

Cascadeless Rollback Recoverable schedule condition avoids irrecoverable schedules cascadding rollbacks and W-R Problem but not avoids R-W, W-W, lost update problem.

Lost Update Problem

This problem occurs if two or more trans writing same data simultaneously.

T_i	T_j
W(A)	
|	W(A)
|	
C/R	

If T_i writes (A)
Then T_j also writes A
before T_i C/R. Then
Lost Update Problem can occur.

eg

T_1	T_2
$W_1(A)$	
	$W_2(A)$
↯ failed	

A = 0
100 (by T_1)
200 (by T_2) ← Lost Updation
T_2 update is lost after T_1 failure

Final value A = 0

Strict Recoverable Schedule

[Cascadeless Rollback Recoverable schedule] and [No lost update problem]

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

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

~~If T_i writes "A" then some other transaction R(A)/W(A) of T_j must be delayed until commit~~

T_i	T_j
W(A)	
!	
C/R	
	R(A)/W(A)

Strict Recoverable schedule:-

if T_i writes data item "A" then some other trans T_j R(A)/W(A) of T_j must be delayed until commit/rollback of T_i.

Q. Test given schedule is strict Recoverable or not :-

1)

T_1	T_2
R(A)	
	R(A)
	W(A)
	Commit
W(A)	
Commit	

Strict Recoverable
⇒ Non Serializable (RW Prob. exists)

2)

T_1	T_2	T_3
R(A)		
	W(A)	
	Commit	
W(A)		
Commit		
		W(A)
		Commit

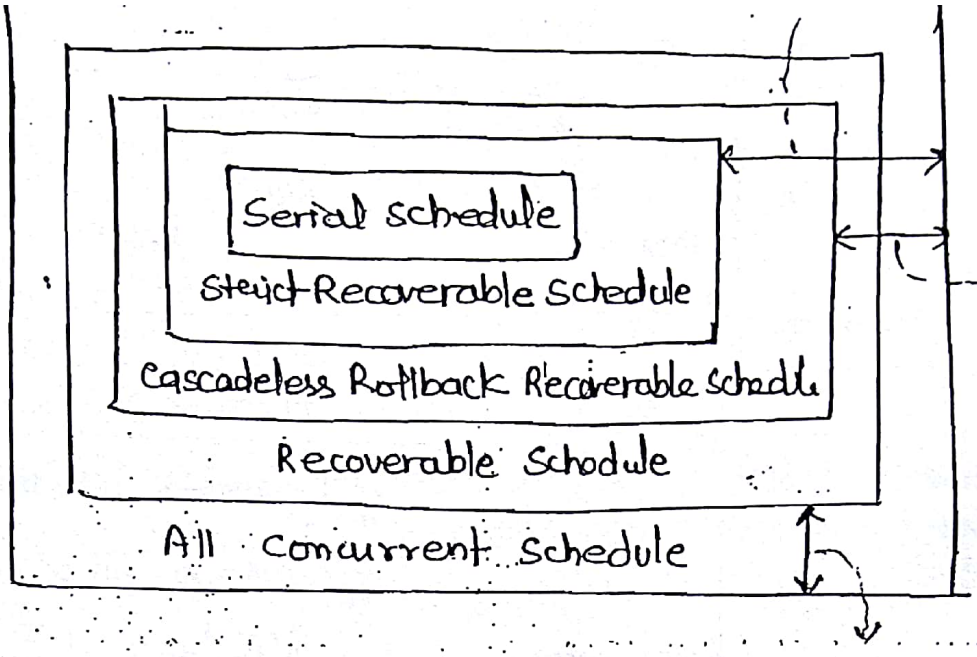
Strict Recoverable (No Conflict)

3)

T_1	T_2	T_3
R(A)		
	W(A)	
	Commit	
W(A)		
		W(A)
		commit
commit		

Not strict Recoverable

Strict Recoverable ~~conditi~~ schedule may be non serializable schedule which may consist R-W Problem but it is free from W-W, W-R, Irrecoverability Problem., Cascading Rollback & Lost Update Problem.



Q) Schedule is cascadeless but not strict recoverable, then which problem may arise?
Lost update problem.

Q) Schedule is recoverable but not cascadeless then which problem may arise?
Cascading rollback, Lost update

S: $r_1(A)$ $r_2(B)$ $r_3(C)$ $r_4(D)$ $r_2(C)$ $r_1(B)$ $w_2(A)$ $w_3(B)$ $w_1(C)$ $w_4(D)$ c_2 $r_3(A)$ $w_3(B)$ $r_3(B)$ c_4 $r_1(D)$ c_3 c_1

Which is true?

a) Strict Rec.
b) CLR Rec but not strict
c) Rec sched but not CLR Rec.
d) IRRecoverable

T_1	T_2	T_3	T_4
$r_1(A)$			
	$r_2(B)$		
		$r_3(C)$	
			$r_4(D)$
	$r_2(C)$		
$r_1(B)$			
	$w_2(A)$		
		$w_3(B)$	
$w_1(C)$			
			$w_4(D)$
	C_2		
		$r_3(A)$	
		$w_3(B)$	
		$r_3(B)$	
			C_4
		$r_1(D)$	
		C_3	
C_1			

First check for strict recoverable.
If it fails then check for CLR recov. & so on.

T_1	T_2
$w(A)$	
C/R	
	$R(A)/W(A)$

(Strict)

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

(CLR Recoverable)

T_1	T_2
$w(A)$	
	$R(A)$
C/R_1	
	C_2

(Recoverable)

Strict Recoverable

Discard all initial read op's as problem starts by write operation.

Now we will check for $w_2(A)$, whether it is written or read by any other trans before C/R,

similarity check for $w(B)$, $w(C)$ and $w(D)$

In $w_3(B)$, B is written before commit but the write op' is also done by trans T_3

∴ strict recoverable satisfied

∴ (a) is correct

Schedule Classification based on Serializability :-

a) Conflict serializable schedule
b) View serializable schedule

a) Conflict Serializable Schedule :

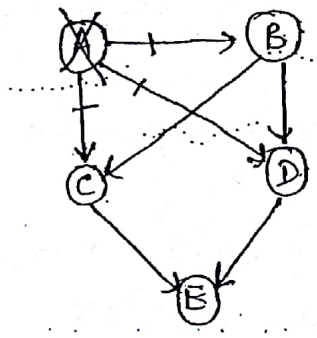
(*) **Topological Order** : [Graph Traversal Algorithm]
only for Directed Acyclic Graphs

Procedure -

i) Visit vertex V whose indegree zero and delete vertex V from graph.
ii) Repeat (i) for all vertices (n times)
↳ for n vertices in a graph

(*) Vertices visiting order is called Topological order.

eg.



A : B : C : D : E
A : B : D : C : E
(2 Topological orders)

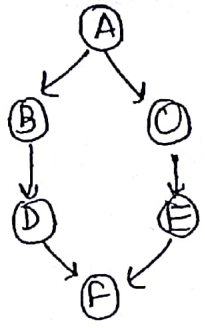
After removal of A :-
B becomes indegree 0

After removal of B :-
a) C becomes indegree 0
b) D " " "

a) After C removal:
i) visit D
then visit E

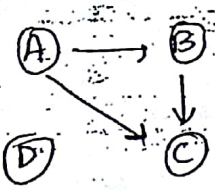
b) After D removal:
visit C then visit E

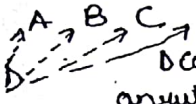
Q. How many topological orders are possible for given graph -



A :
- B :
  - C :
    - D : E : F
    - E : D : F
  - D : C : E : F
- C :
  - B :
    - E : D : F
    - D : E : F
  - E : B : D : F

b)



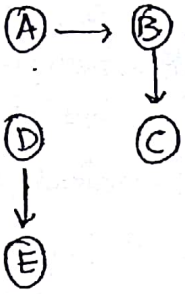


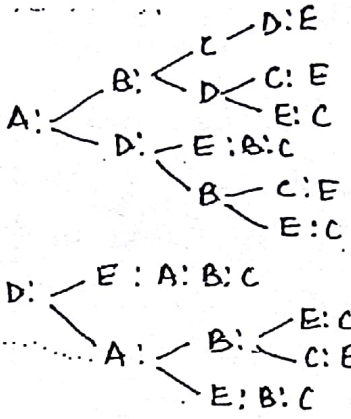
D can be placed anywhere b/w A, B, C
So total 4 topological orders.

D : A : B : C

4

c)





A _ _ B C .

10

A should be before B & B before C
we can place them in $5C_3$ ways
Now remaining positions filled with D or E in $^2C_2$ ways

$5C_3 \times {}^2C_2 = 10$ Topological orders.

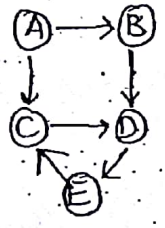
d) Null graph

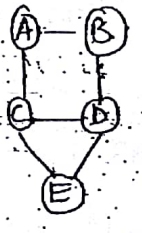
Graph with n vertices and 0 edges

# of Topological Orders: n!

every possible order will become a topological order.

e) Cyclic Graph / undirected Graph

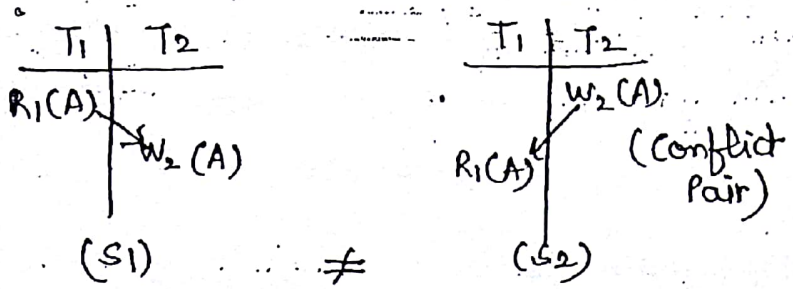




No topological order.

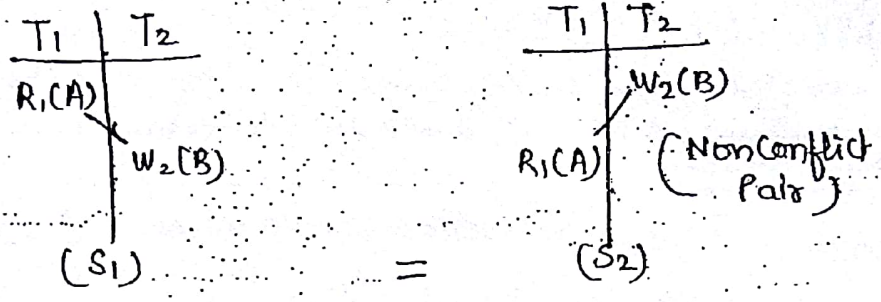
⑧ Conflict Pair

a)



Conflict Pair are those pairs of ~~instruction~~ operation which when interchanged in the schedule (S1) results in another schedule (S2) such that ~~and~~ $S_1 \neq S_2$ due to interchange of operation,

b)



Non conflict pairs are those pair of operations whose interchange has no effect such that S_1 still equal to S_2.

⑧ Remember :- The operations in the pair belong to different transactions.

Now,

1) S: $r_1(A)$ .... $r_2(A)$

2) S: $r_1(A)/w_1(A)$ .... $r_2(B)/w_2(B)$ — Non conflict Pairs

3) S: $r_1(A)$ ... $w_2(A)$

4) S: $w_1(A)$ ... $r_2(A)$

5) S: $w_1(A)$ ... $w_2(A)$ — Conflict Pairs

Conflict Pair

1. atleast one write
   and
2. on same data item
   and
3. from different transactions

T_1

$R_1(A)$
$R_1(B)$
$W_1(B)$

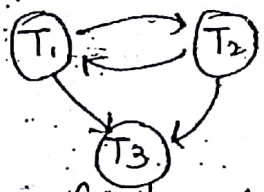
Transaction (T_i) operations must execute same order in any schedule.

The pair of operations, both belonging to same trans, are not considered to have conflict. so no need to consider them for test.

(A) S: $R_1(A)$ $W_2(A)$ $W_1(A)$ $R_1(B)$ $W_3(A)$ $R_3(B)$ $W_3(B)$

Construct Precedence graph.

Here there are 3 transactions. so 3 vertices represent trans 1, 2, 3.



Precedence Graph

Vertices (V): Trans in schedule

Edges (E): Conflict Pair Precedences in schedule

Edges must be constructed from T_i to T_j if there is conflict btw operations from i to j.

Remember:
Conflict will be for:
i) same data item
ii) Diff. transactions

i) $R_1(A)$ → $W_2(A)$ $T_1 \rightarrow T_2$
→ $W_3(A)$ $T_1 \rightarrow T_3$
~~$W_3(B)$~~

ii) $W_2(A)$ → ~~$W_1(A)$~~ $T_2 \rightarrow T_1$
~~$R_1(B)$~~
$W_3(A)$ $T_2 \rightarrow T_3$
~~$R_3(B)$~~
~~$W_3(B)$~~

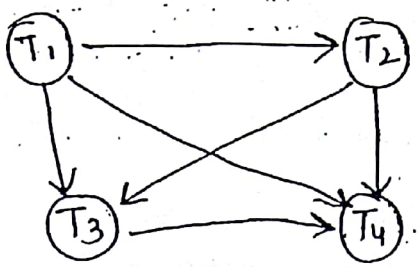
iii) $W_1(A)$ → $W_3(A)$ $T_1 \rightarrow T_3$
~~$R_3(B)$~~
~~$W_3(B)$~~

iv) $R_1(B)$ → $W_3(B)$ $T_1 \rightarrow T_3$

v) $W_3(A)$ ×

vi) $R_3(B)$ ~~×~~ vii) $W_3(B)$ ×

Q S: $R_1(A)$ $R_2(A)$ $R_3(A)$ $R_4(A)$ $W_1(B)$ $W_2(B)$ $W_3(B)$ $W_4(B)$



No cycle

Conflict Equal Schedules

S_1 & S_2 schedules are conflict equal iff S_2 schedule derived by exchanging of consecutive non conflict pairs of schedule (S_1).

$[S_1]$ ⟷ (by exchanging Consecutive Non Conflict Pairs) $[S_2]$

(V) The non consecutive pairs can't be exchanged as then order of S_1 or S_2 could be disturbed.

eg. if $R_1(A)$ and $R_2(A)$ are exchanged then order of S_1 is disturbed.

eg

	T_1	T_2
1.	$R_1(A)$	
2.	$W_1(A)$	
3.		$R_2(A)$
4.	$R_1(B)$	
5.	$W_1(B)$	
6.		$R_2(B)$

$T_1 \to T_2$

(S_1)

⇒

T_1	T_2
$R_1(A)$	
$W_1(A)$	
$R_1(B)$	
	$R_2(A)$
$W_1(B)$	
	$R_2(B)$

$T_1 \to T_2$

(S_2)

Conflict Pair must maintain same order whereas Non Conflict Pair order can be changed.

T_1	T_2
$R_1(A)$	
$W_1(A)$	
$R_1(B)$	
$W_1(B)$	
	$R_2(A)$
	$R_2(B)$

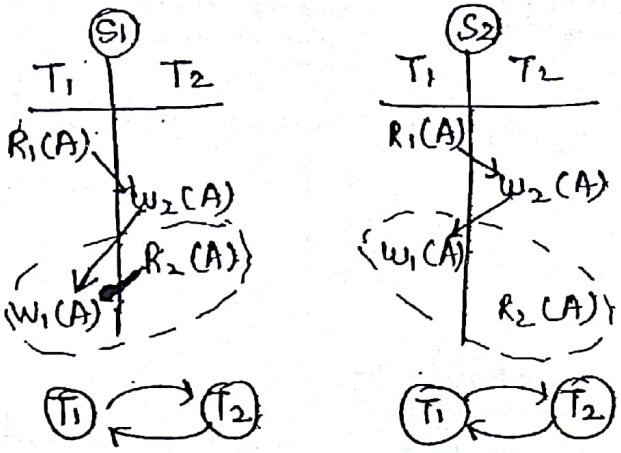
(S_3) $T_1 \to T_2$

• S_1, S_2, S_3 are conflict equivalent.

(*) if S_1 and S_2 are conflict equal schedule then Precedence graph of S_1 & S_2 must be same.

But
if S_1 & S_2 consist same precedence graph then S_1 & S_2 may / maynot be conflict equal.

eg

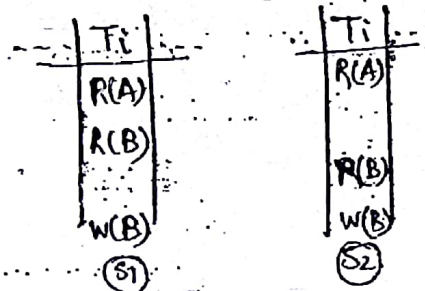


S_1 & S_2 are not conflict equal.

These are not conflict equal schedules but precedence graph of both schedules is same.

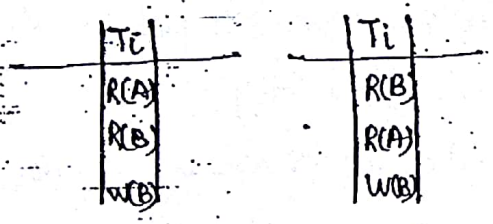
(*) S_1 & S_2 are conflict equal iff

a) every Trans of S_1 schedule must be same trans in S_2.



are conflict equal

Ti in both schedules should have same no. of operations, same operations & in same order.



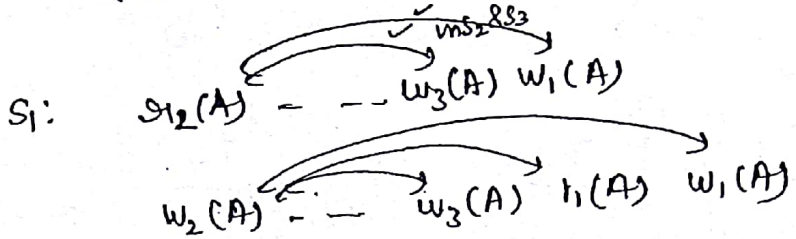
not conflict equal.

(and) b) every conflict precedence of S_1 must be same precedence in S_2 also.

Workbook
Pg57 Q.7

i) First check if order of every trans. in each schedule is same.

ii) Then check for every conflict pair that exist in S_1 also exist in S_2 and S_3 or not.



Conflict Serializable Schedule :-

Schedule (S) is conflict serializable schedule iff there exists some serial schedule (S') that must be conflict equal to schedule (S).

[Acyclic Precedence Graph] (S) ⟷ conflict equal ⟷ S' (serial schedule) [Acyclic Precedence Graph]

then conflict serializable schedule.

If we construct precedence graph of any serial schedule ~~that is always Acyclic~~ then it will always be acyclic.

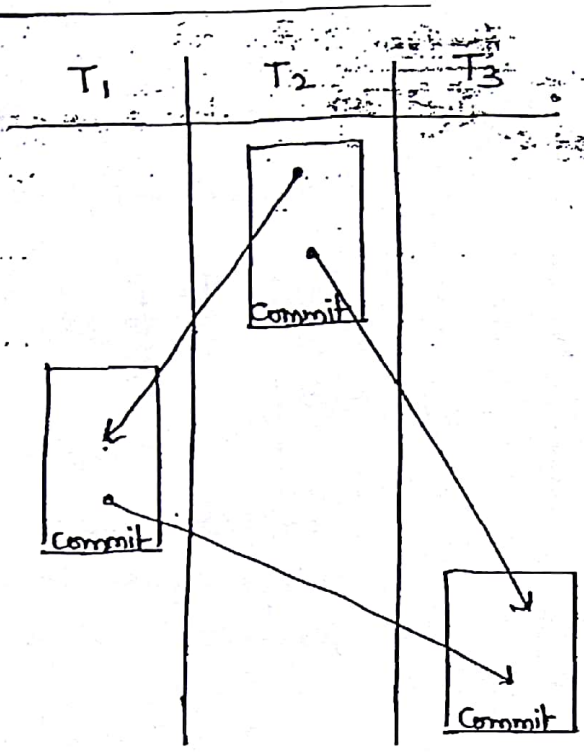
(*) Precedence graph of any serial schedule is always Acyclic.

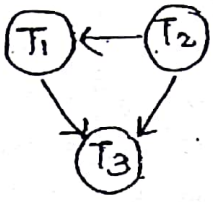
Consider given schedule, it is a serial schedule.

$T_2 : T_1 : T_3$

If there exist conflict pairs, then precedence will be

$T_2 \rightarrow T_1$ $T_2 \rightarrow T_3$ & $T_1 \rightarrow T_3$.





"Precedence Graph"
(Acyclic)

In this way, if we construct any precedence graph of any serial schedule, it will always be acyclic.

(i) If S' (serial schedule) has acyclic precedence graph then S will always have acyclic precedence graph iff S is conflict equal to S'

(ii) If precedence graph of schedule (S) is cyclic then schedules is not conflict serialisable schedule

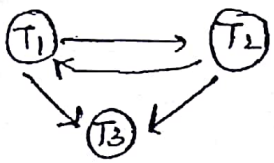
(iii) Schedule (S) is conflict serializable schedule iff Precedence Graph of schedule (S) is Acyclic.

(iv) Conflict Equal Serial Schedules of schedule (S) is Topological orders of Precedence Graph

~~workbook~~

Q Check for conflict serializability?

T_1	T_2	T_3
$R_1(A)$		
	$W_2(A)$	
$W_1(A)$		
		$W_3(A)$



cyclic precedence graph

If S is Non Conflict serializable then there is a Non serial schedule, that is Conflict Equal to Schedule (S).

S': $T_1 \to T_2 \to T_3$ serial

T_1	T_2	T_3
$R_1(A)$		
$W_1(A)$		
	$W_2(A)$	
		$W_3(A)$

∵ conflict precedence is not equal. Thus given schedule S is not conflict equal to serial schedule.

(1) ∴ S & S' are not Conflict equal.

Now, checking for view equal,

T_1	T_2	T_3
$R_1(A)$		
	$W_2(A)$	
$W_1(A)$		
		$W_3(A)$

(S)

T_1	T_2	T_3
$R_1(A)$		
$W_1(A)$		
	$W_2(A)$	
		$W_3(A)$

(S')

(2) So, S and S' are view equal

(3) S is non conflict serializable sched.

(4) S is serializable sched.

[S1, S2 conflict Equal] ⟹ Implied ⟹ [S1 S2 View Equal]

[S is conflict serializable sched] ⟹ Implied ⟹ [S is view serializable sched]

S_1 & S_2 Conflict Equal

1] Every R-W conflict pair and Every W-R conflict pair of S_1 & S_2 must be of same precedence

2] Every ~~right~~ W-W conflict pair of S_1 & S_2 must be same precedence.

as in prev. ex.

S_1 S_2 View Equal

1] Every initial read and Every updated Read of S_1 & S_2 must be the same.

Double Implied

2] Final writes of S_1 & S_2 must be same

Implied

S & S' are not conflict equal as $W_2(A) \rightarrow W_1(A)$ in S

whereas $W_1(A) \rightarrow W_2(A)$ in S'

but as final writes were same so, S & S' were view equal.

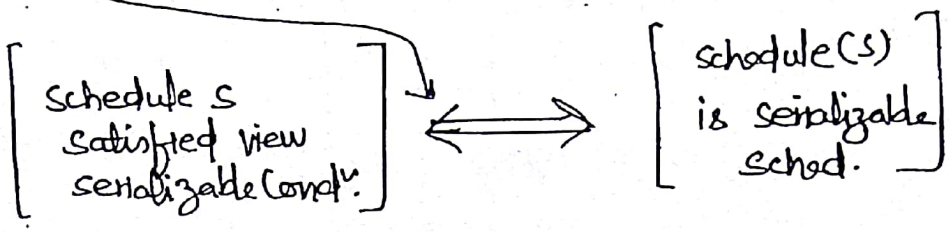
(4) Conflict serializable schedule testing condition only sufficient but not necessary condition for Serializability testing.

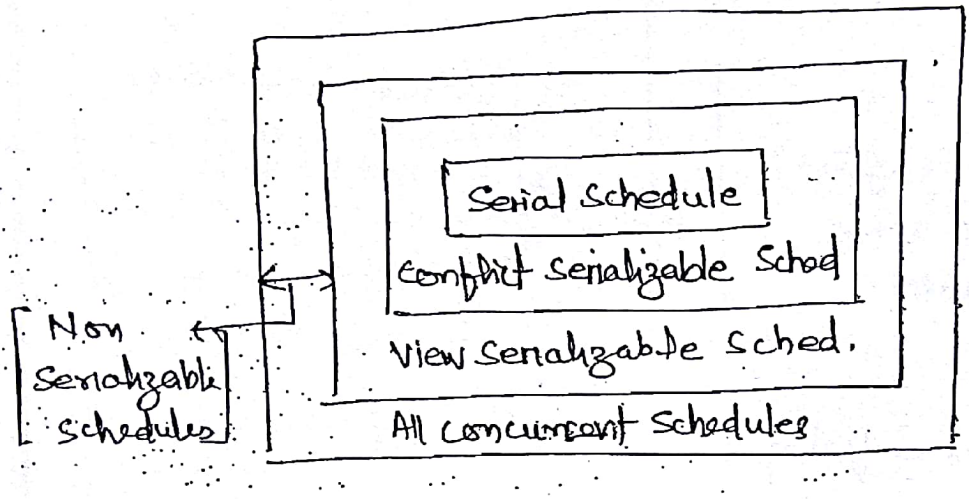
[Precedence Graph of Schedule (S) Acyclic] ⟹ Implied [Schedule (S) is serializable]

But this condⁿ not necessarily proves that s is not serializable

but is sufficient to it ~~can~~ say that s is serializable

→ View serializable schedule testing condition is both sufficient & necessary for serializability testing





View Serializable Schedule :-

Schedule (S) is view serializable schedule iff some serial schedule (S') is view equal to schedule(S).

Test given schedule is serializable or Not?

→ For this we have to test for view serializability.

(bcoz every non conflict schedule may/may not be serializable)

T_2	T_3
R(A)	
R(B)	
W(A)	
	R(A)
W(B)	
	W(B)

(S)

For view serializability, we have to find a serial schedule (S') which will be view equal to S.

Finding view equal S' (serial sched) :-

Due to $R_2(A)$

$W_1(A)$

in serial schedule — $T_2 : T_1$

also

(S)	serial schedule (S')
T_1 Final write	$T_2 \rightarrow T_1$ (for this T_3 can be anywhere)
T_2 (T_3) Final write	$(T_1, T_2) \rightarrow T_3$ (T_3 must be final)
Initial Write Read	
T_i ... T_j	$T_i \rightarrow T_j$

3] Updated Read	$W_i(A) \rightarrow R_j(A)$ T_k : other trans $[T_2]$ W_A	$T_i \xrightarrow{[T_2]} T_j$ [T_i before T_3 and T_2 not b/w T_i & T_3]

All the 3 cond^n are satisfied by serial schedule

$T_2 \rightarrow T_1 \rightarrow T_3$

so, serial schedule (S') $\equiv$ $T_2 : T_1 : T_3$ is view equal to S

Q

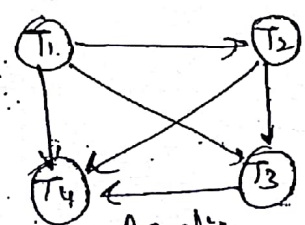
S: $r_1(A)$ $r_2(A)$ $r_3(A)$ $r_4(A)$

$w_1(B)$ $w_2(B)$ $w_3(B)$ $w_4(B)$

a) How many serial schedules conflict equal to schedule (S).

This can be given as

No. of conflict equal serial schedules	=	No. of topological orders



Acyclic

$T_1 : T_2 : T_3 : T_4$

[one topological order] only

$\Rightarrow$ one serial sched. conflict equal to schedule S.

b) How many serial schedules view equal to schedule S:

	Given Schedule (S)	Serial schedule (S')
Final write	B: T_1 T_2 T_3 (T_4)	$(T_1\ T_2\ T_3) \rightarrow T_4$

Initial Read

Data Item	Initial Read	Write	
X	T_i	–	–
X	–	T_j	–

(if only Initial Read but no write then don't bother for any schedule.

(if no write but only initial read then also initial read don't bother.

Initial Read: —

Updated Read: — (No update read)

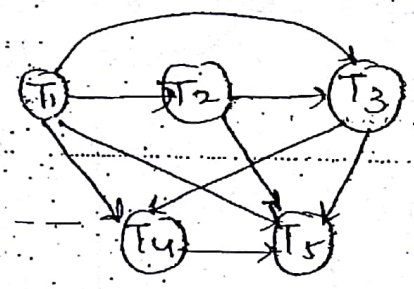
So only one cond^n: $(T_1, T_2, T_3) \rightarrow T_4$

$3! = 6$ serial schedules view equal to S.

Q

S: $r_1(A)\ r_3(D)\ w_1(B)\ r_2(B)\ w_3(B)\ r_4(B)\ w_2(C)\ r_5(C)\ w_4(E)\ r_5(E)\ w_5(B)$

a) How many serial schedules conflict equal to (S).



$T_1\ T_2\ T_3\ T_4\ T_5$

∴ only one topological order

So, only one serial sched. conflict equal to sched (S)

b) How many serial schedules view equal to (S).

	Given schedule (S)	Serial schedule (S')
1) (Final Write)	B: T_1 T_3 T_5 (final write) if no transaction or only one transaction writes a data item then ignore it. A, C, D, E ✗	$(T_1, T_3) \rightarrow T_5$

2) (Initial Read)

Data Item	Initial Read	Write	
A	T_1	—	—
D	T_3	—	—
B	—	✗	—
C	—	✗	—
E	—	✗	—

3) (Updated Read)

	TIs	
$W_1(B) \rightarrow R_2(B)$	T_3, T_5	$T_1 \xrightarrow{(T_3 T_5)} T_2$
$W_3(B) \rightarrow R_4(B)$	T_1, T_5	$T_3 \xrightarrow{(T_1 T_5)} T_4$

$W_2(C) \rightarrow R_5(C)$ — IIC — $T_2 \rightarrow T_5$

$W_4(E) \rightarrow R_5(E)$ — $T_4 \rightarrow T_5$

Cond^{n}s

$(T_1\ T_3) \rightarrow T_5$

$T_1 \xrightarrow{(T_3\ T_5)} T_2$

$T_3 \xrightarrow{(T_1\ T_5)} T_4$

$T_2 \rightarrow T_5$

$T_4 \rightarrow T_5$

$T_1 - T_2 - T_3 - T_4 - T_5$

Start with T_i's having some order

$T_1 - T_2 - T_5$, $T_3 - T_4 - T_5$

T_3 ↕ T_2 X T_3 ↕ T_4 X

T_i: $T_1\ T_3\ T_4\ T_5$

$T_3\ T_4\ T_1\ T_5$

$T_3\ T_4\ T_5\ T_1$

T_4: $T_3\ T_1\ T_2\ T_5$; $T_3\ T_4\ T_1\ T_2\ T_5$

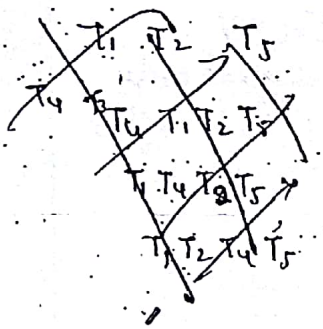
$T_1\ T_2\ T_3\ T_5$; $T_1\ T_2\ T_3\ T_4\ T_5$

T_2: $T_1\ T_2\ T_3\ T_4\ T_5$

$T_3\ T_4\ T_1\ T_2\ T_5$

$T_3\ T_4\ T_5\ T_1\ T_2$

$T_2 \rightarrow T_5$ ✗



So, only 2 serial schedules:

$T_3\ T_4\ T_1\ T_2\ T_5$

$T_1\ T_2\ T_3\ T_4\ T_5$

are view equal to S.

T_3 — S_1, S_2, S_3 } IRREC ✓, Rec ✓, CLR ✓, Strict ✓ — order

Type II

T_3

1) 2) 3) 4) 5) 6) 7) 8)

For CSS, VSS } Ignore Commit

For IRREC, CLR, Strict, IRRec } Commit order matters

eg: T_1 | T_2

$R(A)$

$R(A)$

$W(A)$ $W(A)$

Ignore as W-W conflict still considered

$T_1 \rightleftarrows T_2$

still not conflict serializable if commit or not.

eg

	T_1	T_2
1.	$R_1(x)$	
2.		$R_2(x)$
3.	$W_1(x)$	
4.		$W_2(x)$
5.	C_1	C_2
6.	C_1	C_2

Rec ✓

CLR ✓

Strict ✗ as Lost Update may occur

either C_1 before or C_2 before

eg

T_1	T_2
$W(A)$	$R(A)$
$W(B)$	$R(B)$
C_1	
	C_2

Updated Read so,

Not strict

Not ELR free

May/May not Rec

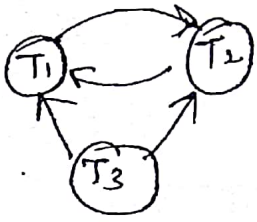
when commit is not given then commit can follow any order

$C_1 \rightarrow C_2$ or $C_2 \rightarrow C_1$

(*) If (schedule is not conflict serializable schedule and No Blind Write) Then (schedule is non-serializable schedule)

eg

T_1	T_2	T_3
$R_1(A)$		
	$R_2(A)$	
		$R_3(A)$
$W(A)$		
	$W(A)$	



Non Conflict serializable schedule and also no blind writes

⇓

Non serializable schedule (even not view serializable)

(*) If schedule is not conflict serializable but it is view serializable schedule then there will be surely Atleast one Blind write in sched(s).

Reverse is not true i.e.

If there is Blind write, it not guarantees to be not conflict serializable.

Q21 Workbook Pg 39 (Type III)

T_1: $r_1(A)$ $r_1(B)$ $w_1(B)$

T_2: $r_2(A)$ $r_2(B)$ $w_2(B)$

# of non serial concurrent schedules b/w T_1 & T_2 is serializable?

Sol:

# of concurrent schedules = $6C_3 \cdot 3C_3 = 20$.

# of serial schedules = 2

# of non serial = $20 - 2 = 18$

# of non serial serializable $\Rightarrow$ (out of 18) $\Rightarrow$

(*) if concurrent schedule is serializable then result of this schedule must be same as result of any serial schedule $T_1 \to T_2$ or $T_2 \to T_1$

(A) Every Initial Read
Updated Read
Final write
must be same in concurrent & serial sched.

a) # of schedules serializable as $T_1 \to T_2$ serial

$[R_1(A)\ R_1(B)\ W_1(B)]\ [R_2(A)\ R_2(B)\ W_2(B)]$

(T_1) $\longleftrightarrow$ (T_2)

Always start like this order. T_1 from last & T_2 from beg $\int$ In case $T_1 \to T_2$ serial

then check for updated read conflict conditions

we see

$W_1(B) : R_2(B)$

to get same view as $T_1 \to T_2$ serial.

Now,

$R_1(A)$ & $R_1(B)$ will always be before $W_1(B)$ as order will have to be followed in same transaction

so,

$R_1(A)\ R_1(B)\ W_1(B) : R_2(B)\ W_2(B)$ (same reason)

Now,

$R_2(A)$ can be placed anywhere before $R_2(B)$ but check if any $W(A)$ exist then $R_2(A)$ can't go before it. So possible schedules -

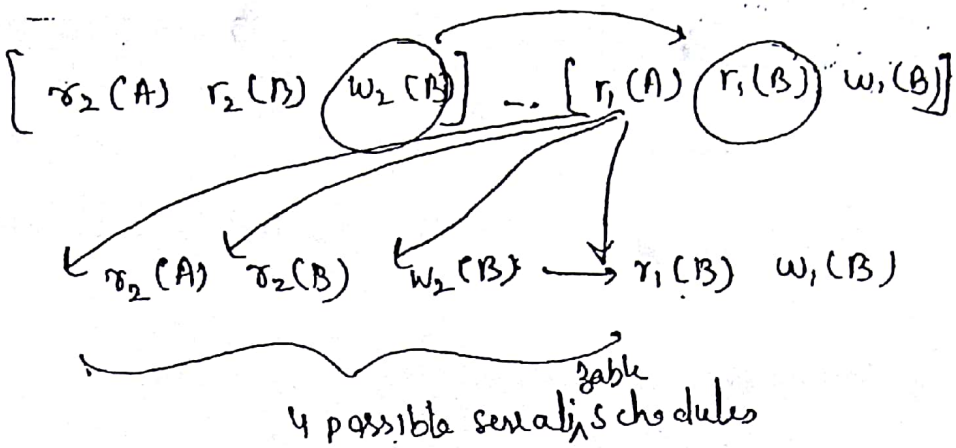
$r_2(A)\ r_1(A)\ r_1(B)\ w_1(B)\ r_2(B)\ w_2(B)$

$r_1(A)\ r_2(A)\ r_1(B)\ w_1(B)\ r_2(B)\ w_2(B)$

$r_1(A)\ r_1(B)\ r_2(A)\ w_1(B)\ r_2(B)\ w_2(B)$

$r_1(A)\ r_1(B)\ w_1(B)\ r_2(A)\ r_2(B)\ w_2(B)$

b) # of schedules serializable as $T_2 \rightarrow T_1$ serial



So, total 8 serializable schedules possible out of 18 concurrent schedules (serial + non serial)

$^6C_3$: 20 Concurrent sched

- 8 Serializable
- 12 Non Serializables

8 Serializable:
- 2 serial
- 6 serializable sched which are non serial. — Ans

Q20 T_1: $r_1(A)$ $W_1(A)$ $r_1(B)$ $W_1(B)$

T_2: $W_2(B)$ $W_2(A)$

a) Serialisable schedule as $T_1 \rightarrow T_2$ serial

[$r_1(A)$ $W_1(A)$ $r_1(B)$ $W_1(B)$] [$W_2(B)$ $W_2(A)$]

Conflict

So, $W_1(B) : W_2(B)$

Only possibility left is put all as it is as order of operations in a trans can't disturb.

1 serializable schedule: $r_1(A)$ $W_1(A)$ $r_1(B)$ $W_1(B)$: $W_2(B)$ $W_2(A)$

b) serializable schedule as $T_2 \rightarrow T_1$ read

$[W_2(B) \quad W_2(A)] \; [r_1(A) \; W_1(A) \; r_1(B) \; W_1(B)]$

conflict

(same case)

$W_2(B) \quad W_2(A) : r_1(A) \quad W_1(A) \; r_1(B) \; W_1(B)$

only 1 serializable schedule

$6C_4 = 15$ Concurrent

2 serializable — 13 Non serializable

2 serial — 0 non serial serializable — Ans

Type III

Qns S: $r_1(A) \; W_1(B) \; r_2(A) \; W_2(B) \; r_3(A) \; W_3(B)$

i) How many schedules conflict equal to sched (S)

Sol: Topological order can't work here because [Topological orders ⇒ serial schedules order which are conflict equal]

but it does not find non serial schedules.

So, this method fails.

Method II: Max 3! or 6 schedules are possible which are serial

Now for S to be conflict equal with other schedule S',

i) transaction set must be same

ii) conflict pairs in S & S' must be same precedence

Schedule S

S: $r_1(A) \; W_1(B) \; r_2(A) \; W_2(B) \; r_3(A) \; W_3(B)$

conflict pairs

conflict equal

Schedule S'

$r_1(A) \; W_1(B) \; \ldots \; W_2(B) \ldots W_3(B)$

$r_2(A)$ (must be before $W_2(B)$) [3 position]

$r_3(A)$ [5 position]

total orders ⇒ $5 \times 3 = 15$

Placing $r_2(A)$

$r_2(A)$ $r_1(A)$ $w_1(B)$ $w_2(B)$ $w_3(B)$

$r_1(A)$ $r_2(A)$ $w_1(B)$ $w_2(B)$ $w_3(B)$

$r_1(A)$ $w_1(B)$ $r_2(A)$ $w_2(B)$ $w_3(B)$

Placing $r_3(A)$ ⟶ At all 15 positions.

So, 15 Conflict equal schedules possible
[Serial + Non Serial]

ii). How many sched View Equal to Sched (S).
↳ Here also we require all schedules [Serial + Non Serial]

Sol:-

Here Initial Reads have no violations
Updated Reads " " "

because only A is read but it is not write and only B is write

Consider only Final write Conflict

$r_1(A)$ $r_2(A)$ $r_3(A)$

$w_1(B)$ $w_2(B)$ $w_3(B)$ ← 15 schedules (as was done before) in (i)

2 types →

$(w_1 w_2) \rightarrow w_3$

$w_2(B)$ $w_1(B)$ $w_3(B)$ ← 15 schedules

30 View equal sched.

Concurrency Control Protocol

Goal: Protocol should not allowed to execute
a) Non serializable schedule
b) Non strict Recoverable schedule

Problems	Serializability: Conflict S.S.	Serializability: View S.S.	Recoverability: Recoverable sched.	Recoverability: Cascadeless Rollback sched.	Recoverability: Strict Rec
1) RW	No	No	YES	YES	YE
2) WR	No (AVOID)	No	YES	No	No
3) WW	No	No	YES	YES	No
4) IRRECOVERABLE	YES	YES	No	No	No
5) CASCADING ROLLBACK	YES (Not Avoid)	YES	YES	No	No
6) LOST UPDATE	YES	YES	YES	YES	No

(*) schedule (S) results correct o/p only if
a) S is serializable schedule
(and) b) S is strict Recoverable Schedule
} implemented through concurr Control

(*)

LOCKING PROTOCOLS

Lock: Variable used to identify status of data item

Trans (T1)

lock(A) ← if lock(A) is granted by Concurrency controller to Trans T_1 then it can execute,
R(A)
W(A)
R(A) [on data item A]
or W(A)

lock(B) ← if lock(B) is denied by concurrency controller then T_1 can't execute any operation on data item B & will wait for lock

lock(B) ← lock(B) granted by concurrency controller when T_1 tried again for B.
R(B)
W(B)
U(A)] Unlock (A) and Unlock (B)
U(B)

⊛ **Legal Transaction** – Transaction use data item only if lock request of data item is granted by concurrency controller.

⊛ **Single Mode Lock**

T_1	T_2
lock(A)	
R(A)	
	lock(A) ← denied
	R(A) ← not be executed because T_1 has already locked data item A.

⊛ **Multiple Lock Mode**

Shared Exclusive Locks

shared Lock[S] : read only lock

exclusive Lock [X] : read/write lock

Trans T_1

S(A)
R(A)
X(B)
R(B)
W(B)

Here degree of concurrency can be increased as now more than one trans is allowed to read the same data item at a time

⊛ As the no. of locks increases, the degree of concurrency will also increase.

Compatibility Table

A	S	X
S	Yes	No
X	NO	NO

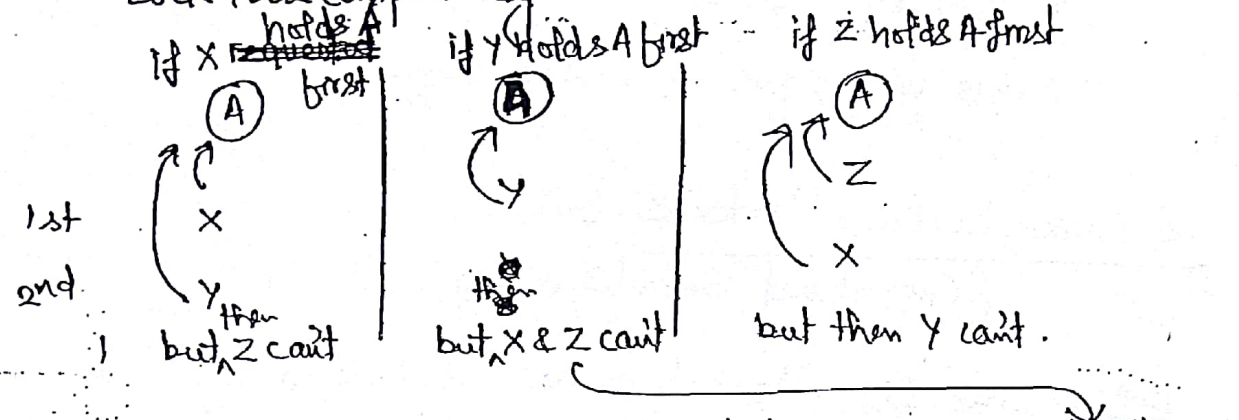
Yes → Read Read Access is Allowed

Q

Request \ Hold	X	Y	Z
X	Yes	No	Yes
Y	Yes	Yes	No
Z	No	No	Yes

How many max. diff. locks (modes) can possible for data item 'A'?

Lock table compatibility



Max 2 locks are allowed & atmost 1 lock allowed

So, for 3 lock variables we have 6 cases to check

for more lock variables, it will become more difficult.

(A)

T_1	T_2
$W_1(A)$	
	$R_2(A)$
	$R_2(B)$
$W_1(B)$	
	C_2
C_1	

This schedule is not serializable as well as not strict recoverable but still can be executed through S/X locking.

Applying S/X locks →

T_1	T_2
$X(A)$	
$W_1(A)$	
$U(A)$	
	$S(A)$
	$R_2(A)$
	$U(A)$
	$S(B)$
	$R_2(B)$
	$U(B)$
$X(B)$	
$W_1(B)$	
$U(B)$	
	C_2
C_1	

∴ S/X lock compatible table and legal trans conditions are not sufficient to avoid Nonserializability or Not strict Recoverability

Thus we saw that, even after applying S/X locks & compatibility conditions,

it is not possible to avoid execution of Non Serializable and Not strict Recoverable schedules

So, goal of locks is not satisfied

For this we go for 2 Phase Locking.

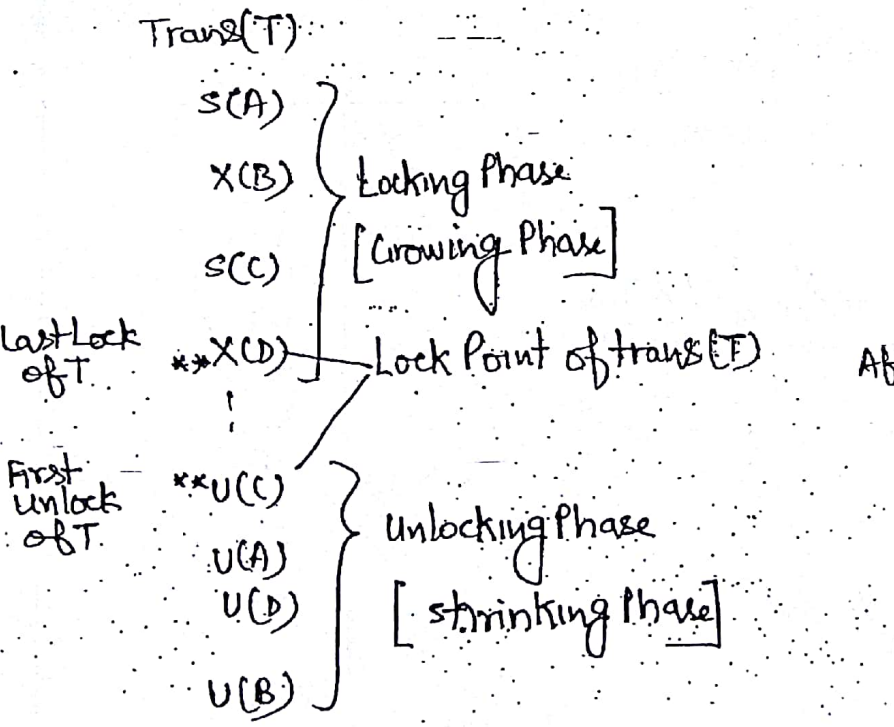
2 Phase Locking Protocol [2PL]

(Guranteed Serializability)

Def:- Trans (T) allowed to request lock on any data item only if none of unlocks done by trans T

(OR)

Lock Requests are not allowed in unlocking phase



After this first unlock request of C, we can't perform any lock request over data item C after U(C) but we can perform oper^n over other data items A, B, D. but lock request wait be allowed after first unlock.

Trans(T)

S(A)
X(B)
R(A)
S(C)
X(D)

} B,C,D are held } Locking Phase

After first unlocks, no lock request will be allowed

U(A)
W(B)
U(B)
R(C)
U(C)
W(D)
U(D)

} unlocking phase.

(*) $T_1 \rightarrow T_2$ conflict pair may allowed to execute if T_1 lock point before T_2 lock point.

(*) cyclic Precedence blw Trans is not allowed by 2PL.

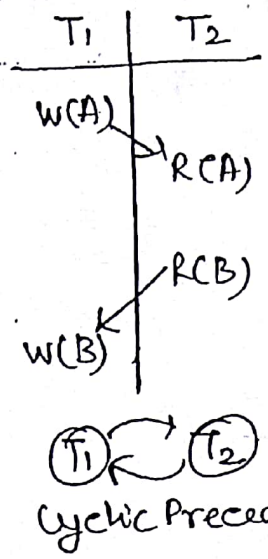
because if T_1 lock point is before T_2 lock point

then conflict pair $T_1 \longrightarrow T_2$ is allowed

but $T_2 \longrightarrow T_1$ will not be allowed by 2PL

so, no cyclic precedence could exist.

eg consider following schedule,



Cyclic Precedence

This schedule can't be executed by locks, explained as follows -

① for W(A), T_1 applied X(A)

② for R(A), T_2 applied S(A).

③ But T_2 can apply S(A) only when T_1 has applied U(A) ↳ unlock.

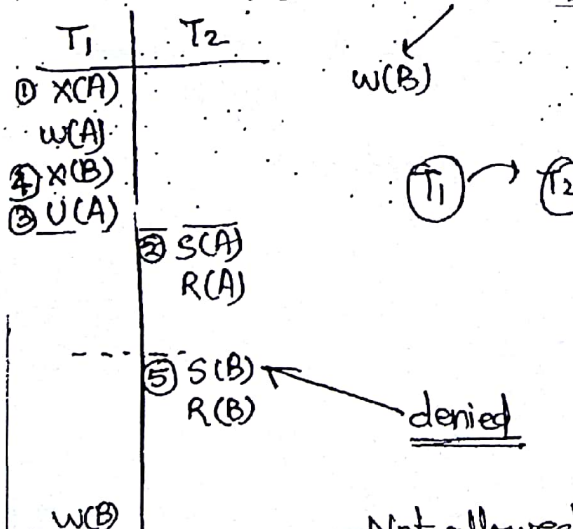
Also, if T_1 executed U(A) then T_1 enters unlocking phase.

But to execute W(B) by T_1 it can't lock B. so, lock B should be done before U(A).

④ for R(B), T_2 has to do S(B) or X(B) let say S(B).

⑤ But this S(B) ~~will~~ request will be denied by concurrency controller because T_1 has already done X(B), so ~~before~~ until T_1 execute U(B), T_2 can't do S(B).

Thus we can see that if W(A) → R(A) is allowed than R(B) is not allowed



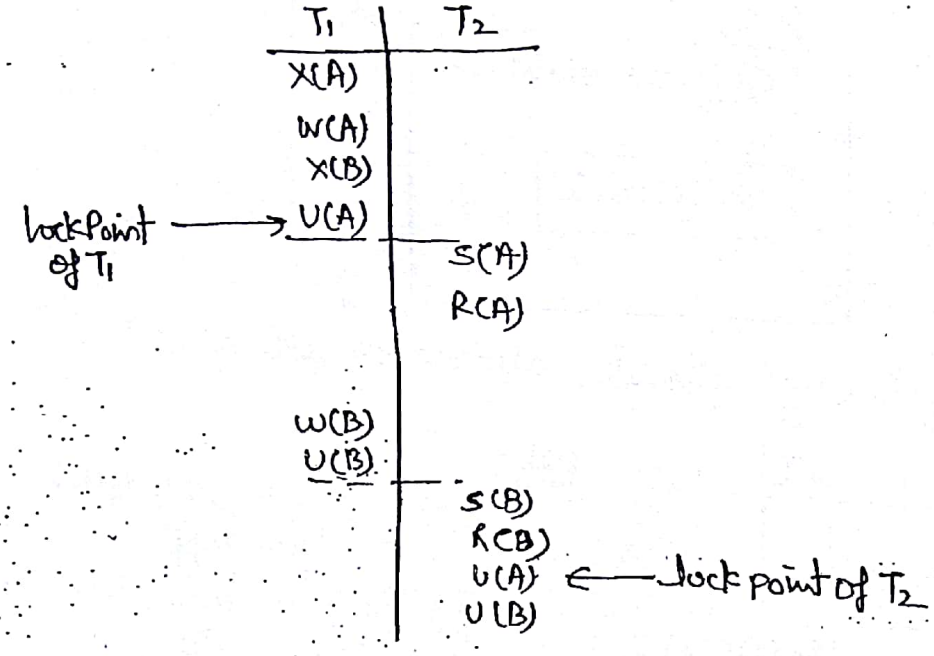
W(B) → Not allowed to execute by 2PL.

(OR)

$T_1 \rightarrow T_2$ allowed but $T_2 \rightarrow T_1$ is not allowed by 2PL.

Thus cyclic precedence is not allowed by 2PL.

For R(B) to be executed by T_2, it has to wait until T_1 executed U(B). So, modified schedule →



Here $T_1 \rightarrow T_2$ "Acyclic precedence"

This will be allowed to be executed by 2PL.

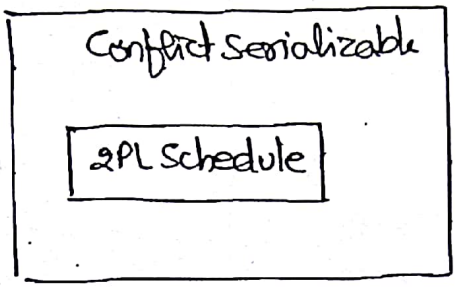
(*) 2PL doesn't allow cyclic precedence graph ⇒ 2PL doesn't allow to execute non-conflict serializable sched.

(*) [Non Conflict Serializable Schedule] ⟹ [Not allowed to Execute by 2PL]

(*) [Schedule executed by 2PL] ⟹ Schedule is conflict serializable schedule (and) Conflict Equal serial sched. is based on lock points order of transactions.

⇒ eg in above schedule executed by 2PL,
∵ Lock Point of T_1 comes before Lock Point of T_2
So, this schedule is conflict equal to serial schedule $T_1 \rightarrow T_2$

(A) [Schedule is conflict serializable] ⟶ [May / May not allowed to execute by 2PL]

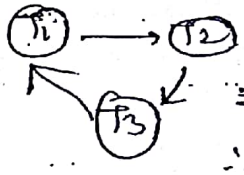


Q. Test the following schedule allowed by 2PL or not?

(I)

T_1	T_2	T_3
$R_1(A)$		
	$W_2(A)$	
	$W_2(B)$	
		$R_3(B)$
		$R_3(C)$
$W_1(C)$		

Sol:- (I)
First check for conflict serializability
If the precedence graph is cyclic then the schedule will not be conflict serializable sched then we can directly say that the sched is not allowed by 2PL.



≡ cyclic
∴ Not Conflict serializable
∴ Not allowed by 2PL.

(II)

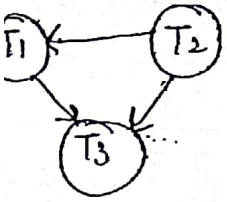
OR

T_1	T_2	T_3
S(A)		
$R_1(A)$		
X(C)		
U(A)		
	X(A)	
	$W_2(A)$	
	X(B)	
	$W_2(B)$	
	U(B)	
		S(B)
		$R_3(B)$
		S(C) ← denied
		$R_3(C)$
$W_1(C)$		
U(C)		

So, if $R_3(C)$ is after $W_1(C)$ then $T_3 \to T_1$ will be replaced by $T_1 \to T_3$ and thus the schedule will be allowed by 2PL.

	T_2	T_3
	W(A)	
	W(B)	
		W(A)
		W(B)

T_1	T_2	T_3
	X(A)	
	W(A)	
	X(B)	
	U(A)	
X(A)		
W(A)		
X(B)		
U(A)		
W(B)	not allowed before $W_1(B)$	
U(B)		
	W(B)	



conflict serializable
-gurantees that
be executed by 2PL.

b)

T_1	T_2	T_3
	X(A)	
	W(A)	
	X(B)	
	U(A)	
X(A)		
W(A)		
	W(B)	
	U(B)	
X(B)		
Lock Point U(A)		
		X(A)
		W(A)
W(B)		
U(B)		
		X(B)
		W(B)
		U(B)
		U(A)

so, allowed by 2PL

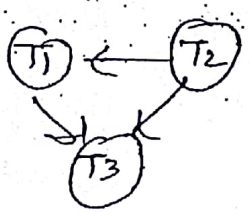
nderstand
ke either first
or last locks

Point Order

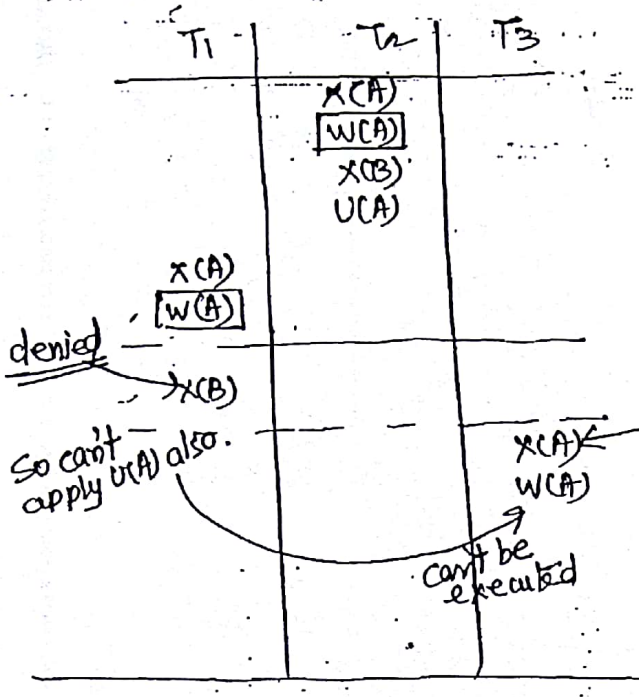
: T_1 : T_3
one of
gives the topological
s or the serial
edule which is conflict equal
this schedule.

T_2	T_3
W(A)	
	W(A)
W(B)	
	W(B)

a)



conflict serializable.



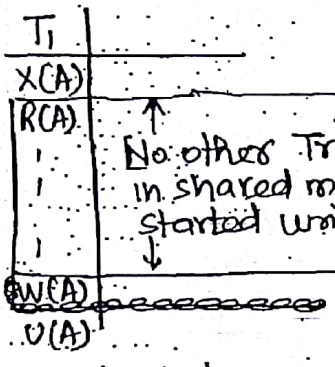
Precedence graph — cycle — Not conflict serializable — Not allowed in 2PL

Acyclic — conflict serializable — may/may not allowed in 2PL

Therefore not allowed by 2PL.

even if it is conflict serializable.

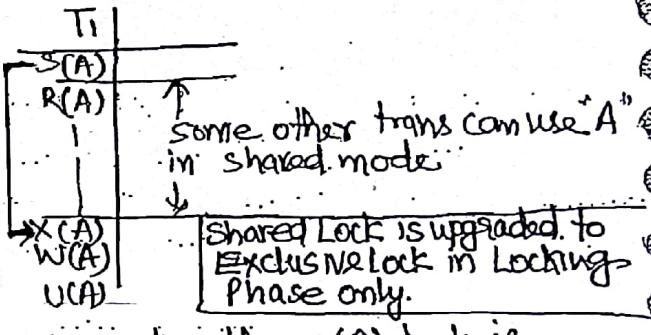
2PL without Lock upgrading



If T_i has to execute R(A) then W(A), then T_i has to apply X(A) before R(A).

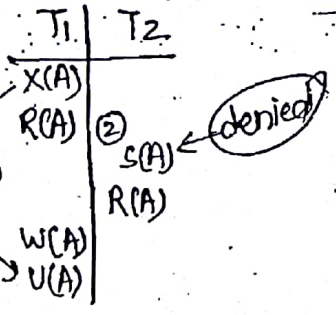
In the marked region, even if no operation is done then also until W(A) is executed, no other trans can lock A even in shared mode

2PL with lock upgrading



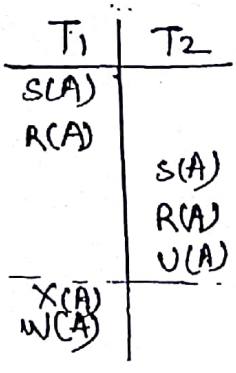
Here firstly S(A) lock is applied and when T_i has to W(A), before that S(A) is upgraded to X(A) such that in marked region, before X(A) some other trans can use "A" in shared mode.

Disadv: Less degree of concurrency



Not allowed by 2PL without lock upgrading

Adv: More degree of concurrency



This schedule is allowed by the 2PL with lock upgrading.

③ Adv: Less Possible Deadlocks

Here although there may be possibility that lock access is denied but deadlock will not occur then.

T_1	T_2
$X(A)$	
$R_1(A)$	
	⑤ $X(A)$ ← denied
①	$R_2(A)$
	$W_2(A)$
$W_1(A)$	
$U(A)$	

ALSO, "No Deadlock" as T_2 has to only wait for T_1 to do $U(A)$ so that $X(A)$ is granted to T_2

③ Disadv: More Possible Deadlocks

Here there may be conds when operation is also denied and there is possibility of Deadlock.

T_1	T_2
① $S(A)$	
$R_1(A)$	
	② $S(A)$
	$R_2(A)$
	③ $X(A)$ ← denied because lock is acquired by T_1
	$W_2(A)$
④ $X(A)$	
$W_1(A)$	

denied because lock is acquired by T_2

∴ T_2 is waiting for T_1 and T_1 is waiting for T_2 so, deadlock occurs

2PL with lock Upgrading

We can get [More degree of concurrency and Less Possible Deadlocks] by introducing one more lock

copyright©theorypoint.com The Best Gateway Of Success

S: shared lock not allowed upgrade.

SS: shared lock allowed to upgrade

X: Exclusive mode

(i)

T_1	T_2
$SS(A)$	
	$S(A)$ ← granted

(ii)

T_1	T_2
$S(A)$	
	$SS(A)$ ← granted

(iii)

T_1	T_2
$SS(A)$	
	$SS(A)$ ← denied

(iv)

T_1	T_2
$S(A)$	
	$S(A)$ ← granted

Remember-

By default, 2PL means 2PL with lock Upgrading.
(S, X)

Q Check whether the following schedule is allowed by 2PL/not?

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

(A) If there is any delay or deadlock then it is not allowed by 2PL.

means 2PL with lock Upgrading (S, X)

Here we don't take SS lock

T_1	T_2
S(A)	
R(A)	
	S(A)
	R(A)
	S(B)
	R(B)
	X(B)
	W(B)
	U(B) U(A)
X(B)	
W(B)	
X(A)	
W(A)	

⟹ No Problem in applying locks.

⟹ Allowed by 2PL with lock upgrading.

Q

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

a) T_1 ⇄ T_2

Not Conflict Serializable

b) Sol:-

T_1	T_2
S(A)	
R(A)	
X(A)	
W(A)	
S(B)	
X(B)	
U(A)	
	S(A)
	R(A)
	S(B) ← denied
	R(B)
R(B)	
W(B)	

So that W(B) can be executed

T_1	T_2
S(A)	
R(A)	
X(A)	
W(A)	
	S(A) ← denied
	R(A)
	S(B)
	R(B)
S(B)	
X(B)	
U(A)	
R(B)	
W(B)	
U(B)	

OR

So not allowed by 2PL also.

Limitations of 2PL:-

1) 2PL condition may leads to deadlock.

eg: T_1 : $r_1(A)$ $r_1(B)$ [S(A) S(B) U(A) U(B)]

T_2 : $w_2(B)$ $w_2(A)$ [X(A) X(B) U(A) U(B)]

T_1	T_2
S(A)	
$r_1(A)$	
	X(B)
	$W_2(B)$
denied → S(B)	
$r_1(B)$	
	X(A) ← denied
	$W_2(A)$

let this be one possible concurrent schedule then, locks are applied as shown.

... Here we see that,

S(B) is denied for T_1

and X(A) is denied for T_2

(OR)

T_1 is waiting for T_2 to do U(B)

& T_2 " " " T_1 to do U(A)

So, deadlock occurs.

The deadlock can occur for any arrangement of these locks.

Q T_1: $r_1(A)$ $r_1(B)$ [S(A) S(B) U(A) U(B)]

T_2: $w_2(A)$ $w_2(B)$ [X(A) X(B) U(A) U(B)]

which is true?

a) every concurrent exec^n of T_1 & T_2 is serializable.

b) every concurrent exec^n of T_1 & T_2 is not serializable

c) every concurrent exec^n of T_1 & T_2 is free from deadlock

d) every concurrent execution of T_1 & T_2 cause deadlock.

Sol:

T_1	T_2

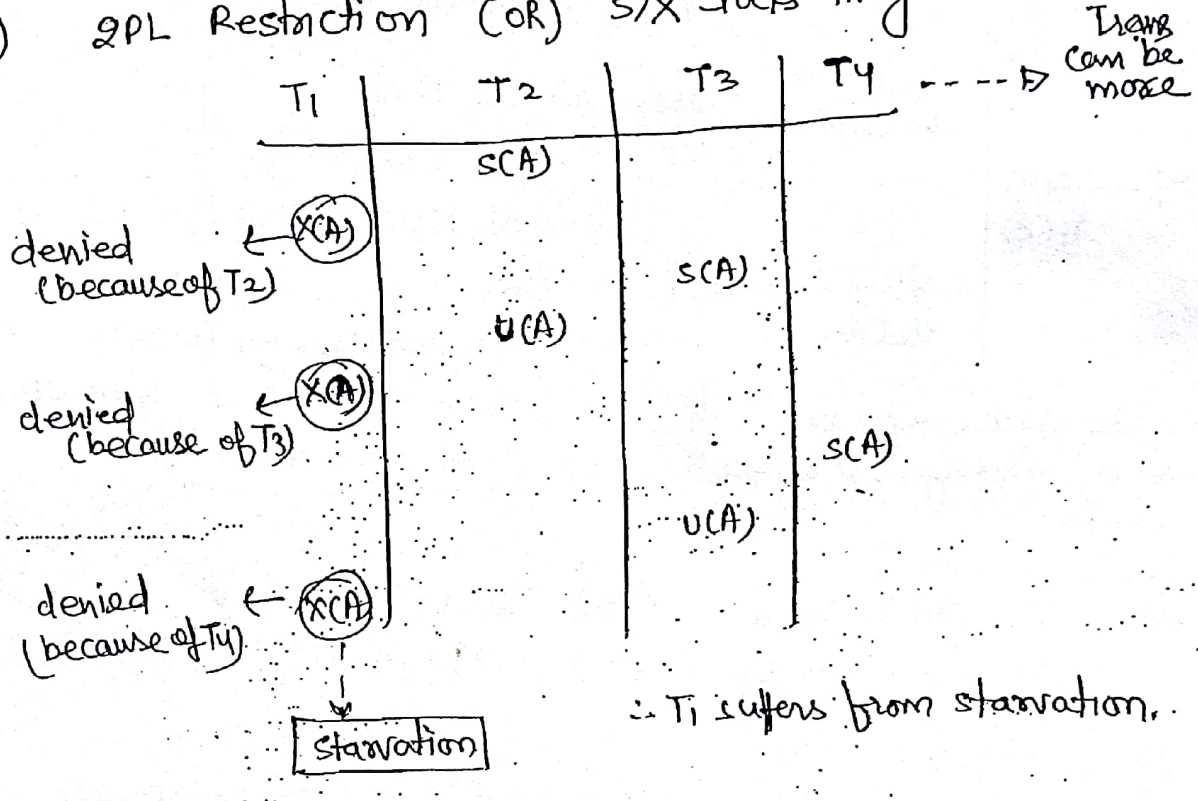
If T_1 starts executing then T_1 acquires lock S(A) on A and T_2 wait for T_1 to unlock (A) to acquire X(A). Then T_2 wait for T_1 to unlock (B) to acquire X(B).

So, after T_1 completion T_2 will definately execute, so no deadlock.

Similarly; If T_2 starts executing then T_2 acquires X(A) on A & then T_1 will be waiting for T_2 to release lock on A.
After that T_1 will wait for T_2 to release lock on B.
So, after T_2 completion T_1 will be executed definately
So, no deadlock.

∴ (C) is correct

2) 2PL Restriction (OR) S/X locks may cause starvation



∴ T_1 suffers from starvation.

3) 2PL condition not sufficient to Avoid
   i) IRRECOVERABLE SCHED
   ii) cascading Rollbacks
   iii) Lost Update Problem

Allowed by 2PL ⇓ serializable sched. But not recoverable schedule.

T_1	T_2
X(A)	
W(A)	
X(B) U(A)	
	S(A)
	R(A)
W(B)	
U(B)	
	X(B)
	W(B)
	U(A) U(B)

No Problem but still non recoverable.

Strict Recoverable schedule

T_1	T_2
X(A)	
w(A)	
|	
|	
C/R	
U(A)	
	R(A)/W(A)

Exclusive locks must be hold until commit / rollback of transaction.

T_2 can execute R(A) or W(A) after unlock of A by T_1

Strict 2PL Protocol

Basic 2PL : Lock Requests of Trans(T) not allowed in unlocking phase of transaction (T)

(and)

Strict Recoverable : All exclusive Locks of Trans(T) must be hold until commit/rollback of Trans(T).

eg:

Trans(T)

S(A)
X(B)
S(C)
X(D)
|
|

} Locking Phase

Here A & C can be unlocked before commit because A & C are shared locks whereas B & D are exclusive locks.

U(A)
U(C)
commit
U(B)
U(D)

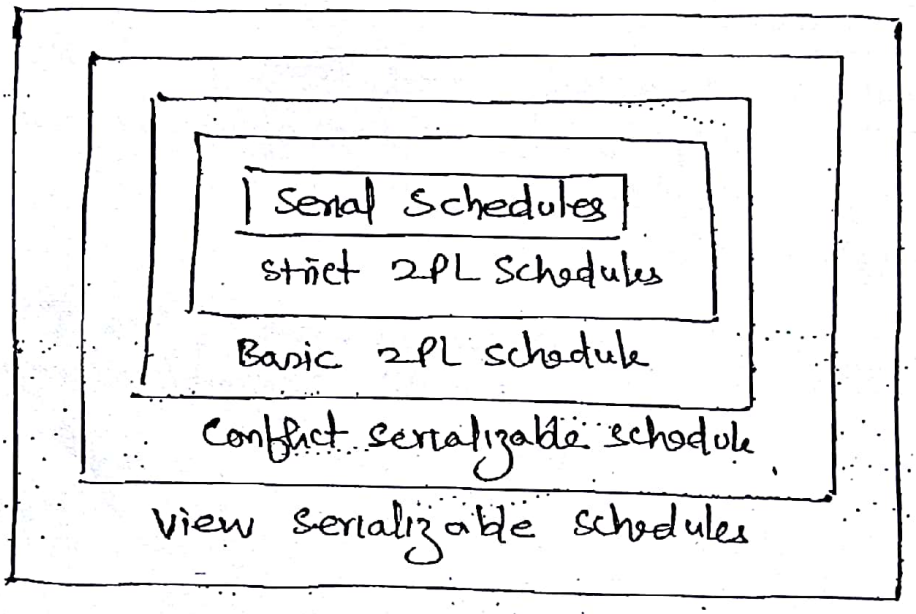
} Unlocking Phase

Strict 2PL :-

① Conflict Serializable schedule equal serial schedule based on lock points order.

② Strict Recoverable schedule

③ Deadlock, Starvation may occur.



copyright©theorypoint.com
The Best Gateway Of Success