Date: _____ Page No. _____

Books → (1) Korth
(2) Navathe
(3) Schaum Series

Topics →

		Page No.
(1)	SQL ***	(3 - 66)
(2)	Normalization *	(67 - 120)
(3)	Hashing and Indexing *	(121 - 166)
(4)	Transaction **	(167 - 206)
(5)	E-R Diagram	(207 - 22)

Date: Page No:

Definition: A DBMS is a software whose basic requirement is to provide facility to store data.

* Any collection of relevant data is database.

Other Requirements of DBMS:

(1) It should provide an easy language for data manipulation (SQL)

(2) It must maintain integrity (uniqueness) of data.

* There can be two types of integrity:
  (1) Entity Integrity
  (2) Referential Integrity

(3) It must provide concurrent access to data i.e. multiple users can access data simultaneously.

(4) It must provide security.

—x—

* Database is a collection of relevant or meaningful informat

Date: _____ Page No: _____

RDBMS (Relational DBMS):

basic requirement is to

* Its (A) stores data in the form of tables or relations.

eg:

RollNo	Name	Phone
1	Mayank	100
2	Aman	200
3	Neeraj	300

* Here, RollNo, Name and Phone are columns / attributes / fields

* Also, 1 Mayank 100 is a row / tuple / record

* When data is stored in the form of rows & columns or tuples & attributes or records & fields, this format is known as tabular format or relation.

* Nowadays, Object Oriented DBMS is being used.

SQL

Date: Page No:

⇒ Structured Query Language

① Create a table in database :

⇒ create table Student (RollNo number (6) primary key, name varchar(20), Phone number (10)) ;

② Insert a record in table :

⇒ insert into Student values (1, 'Mayank', 100) ;

* all non-numeric values are written in single codes.

* NULL is used wherever information is missing.

eg

⇒ insert into Student values (2, 'Aman', NULL) ;

⇒ insert into Student values (3, NULL, 300) ;

* NULL is always written without single codes.

* To insert in particular columns ;

⇒ insert into Student (RollNO, Phone) values (1, 400) ;

* remaining columns will contain NULL, but primary key must be inserted as it can't be NULL

⇒ 1 NULL 400

* where clause works row-wise.

eg update student set RollNo = 1 ; ⇒ error as RollNO is primary (has to be unique)

eg update Student set RollNo = 100 where RollNo = 1 ; ✓ no error

③ Update a record :

* it can also update primary key.

⇒ update Student set Phone = 500

* it will set phone of all the students to 500.

⇒ update Student set Phone = 500
where RollNO = 10 ;

* If there's no rollno 10 ⇒ no row updated.
* If there are 2 rollno 10 ⇒ 2 rows updated

Que Write a query to update marks of every students by +10.

Student :

RollNo	Name	Marks
1	A	10
2	B	
3	C	15

* B didn't appear in exams.

⇒ update Student set marks = marks + 10 ;

Student :

RollNo	Name	Marks
1	A	20
2	B	
3	C	25

⇒ NULL + Anything = NULL

Date: _____ Page No: ____

④ Delete a record :

⇒ delete Student or delete from Student

* All the records are deleted but the table still exists

* delete command is used to delete rows not columns.

⇒ delete Student where RollNo = 10 ;

⑤ Drop a table :

⇒ drop table Student

* entire table is deleted.

⑥ Fetch / Retrieve data from database :

⇒ select * from Student

* it will display all records of the student table.

⇒

RollNo	Name	Phone
1	A	100
2	B	200
3	C	300
4	D	400

Date _____ Page No: _____

Q4e Write a query to find name & salary of all the employees of dept. 10.

emp :

empno	ename	sal	deptno
1	A	100	10
2	B	200	10
3	C	300	20

⇒ select ename, sal from emp where deptno = 10.

⇒

ename	sal
A	100
B	200

Q4e Write a query to find name of employees of dept 10 and 20.

⇒ select ename from emp
where deptno = 10 and deptno = 20; ✗

⇒ select ename from emp
where deptno = 10 or deptno = 20; ✓

OR

⇒ select ename from emp
where deptno in (10, 20); ✓

Date: _____ Page No: _____

Que write a query to find name of employees having salary between 10000 to 20000.

⇒ select ename from emp
where sal > 10000 or sal < 20000 ✗

⇒ select ename from emp
where sal > 10000 and sal < 20000 ✓

OR

⇒ select ename from emp
where sal between 10000 and 20000 ✓

⇒ select ename from emp
where sal between 10000 to 20000 ✗

- = : equal to
- != or <> : not equals

* 'between' is usually inclusive.

* between also works with date

Que

⇒ select ename from emp
where ename between 'A' and 'J';

⇒ S1 = 'ABC'
S2 = 'x'
} which is greater?

* ASCII values are compared in strings.

ABC
65 ⇒ 120 > 65 Thus, 'x' is greater than 'ABC'
120
x

⇒ eg 'ABC'

⇒ 'ABC' 65 66 66 > 0 ⇒ 'ABC' > 'A'
'A' 65 0

⇒ 'ABC' 65 75 > 65 ⇒ 'J' > 'ABC'
'J' 75

Thus, 'A' < 'ABC' < 'J'
'ABC' will be displayed.

* It will display all the names b/w A & I & only 'J' can be displayed for J.

Date: _____ Page No. _____

⇒ eg 'JAI'

→ 'JAI' 75 75 > 65 ⇒ "JAI" > 'A'
'A' 65

⇒ 'JAI' 75 65 65 > 0 ⇒ 'JAI' > 'J'
- 'J' 75 0

* 'JAI' is greater than A but not smaller than 'J' thus, it won't be displayed.

—x—

% is a wildcard character. It includes any no. of characters including NULL character.

Date: Page No:

Que Write a query to find the name of employees whose names start from A & end with T.

* 'like' operator is used for string comparison.

⇒ select ename from emp
where ename like 'A%T';

Que Write a query to find name of employees whose names contain 2T. (i) atleast (ii) exactly

(i) ⇒ select ename from emp where
ename like '%T%T%';

(ii) ⇒ select ename from emp where
ename like '%T%T%'
and ename not like '%T%T%T%';

Que Write a query to find name of employees such that A is at 1st position, anything at 2nd, T at 3rd & then anything.

⇒ select ename from emp where
ename like 'A_T%';

% → any no. of characters
_ → exactly one character

* to search % %
⇒ /% % ✗ /%/% ✓

Date _____ Page No. ____

Que Write a query to find name of employees whose names contain exactly 3 characters.

⇒ select ename from emp where
ename like '___';

Que Write a query to find name of employees whose emailid contains an underscore(_).

⇒ select ename from emp where
emailid like '%_%'; ✗

* escape character (/) is used to search _, %, /, etc

⇒ select ename from emp where
emailid like '%/_%';

* escape character is different for different softwares.

Que write a query to find name of employees having NULL salary.

⇒ select ename from emp where
sal = NULL; ✗

- true and unknown = unknown
- false and unknown = false

true or unknown = true
false or unknown = unknown

Date: Page No:

* In SQL, along with true and false, there's o more result of comparison i.e. NULL or unknow

anything (+ or - or =) NULL ⇒ NULL

* thus, 'is' operator is used for NULL comparison

⇒ select ename from emp where sal is NULL;

* To display in order;

* Display follows order of Insertion by default.

⇒ select ename from emp order by ename;

⇒ select ename from emp order by ename desc

Order of Insertion	Ordering	ordering desc
e2	e1	e4
e4	e2	e3
e3	e3	e2
e1	e4	e1

* If 1st column is same, then order of insertion will be followed.

Date: _____ Page No: _____

* ordering can be done in more than one column.

eName	Job
e1	Manager
e1	salesman
e2	Manager
e2	Analyst

⇒ select ename, job from emp order by ename, job

eName	Job
e1	Manager
e1	Salesman
e2	Analyst
e2	manager

* if 1st column is same then ordering is done acc. to 2nd column.

* ename, job desc ⇒ ename (ascending), job (desc.)

* ename desc, job desc ⇒ both (descending)

* ename desc, job ⇒ ename (desc.), job (asc.)

Aggregate Functions / Group Value Functions

1. Sum
2. Min
3. Max
4. Count
5. Average

1. Sum :

⇒ select sum (sal) from emp;

⇒

Name	Sal
e1	10000
e2	
e3	30000
e4	40000

Sum ⇒ 80000 ✓

⇒ NULL ✗

* aggregate functions ignore NULL values.

2. Min :

⇒ select min (sal) from emp; ⇒ 10000 ✓

NULL ✗

* If there's no row in table, except 'count', all can return null.

max, min also works with date.
max → recent date
min → oldest date

Date _____ Page No. _____

③ Max :

⇒ select max(sal) from emp; ⇒ 40000

④ Count :

⇒ select count(sal) from emp; ⇒ 4 ✗
3 ✓

* It also ignores NULL value.

* select count (distinct sal) from emp;

⇒ select count (ename) from emp; ⇒ 4

⇒ count (*) ⇒ counts no. of rows in database.
* If there's no row ⇒ it returns zero (not, NULL)*

⑤ Average :

⇒ select avg(sal) from emp;

* avg(sal) = sum(sal) / count(sal) = 80000 / 3 = 26666.66

80000 / 4 ✗

* It also ignores NULL value.

* sum, max, min, avg dont work with strings.

⇒ select ename, avg (sal) from emp; ✗ error

* ename & avg (sal) are not compatible as column ename contains many values while avg has only one value.

* Thus, we cannot use any aggregate function with non-aggregate columns.

* Two or more aggregate functions are compatible with each other.

⇒

ename	sal	deptno
e1	10000	10
e2	40000	10
e3	15000	20
e4	30000	20

Que Write a query to print sum of salaries dept. wise.

⇒ select sum(sal) from emp
group by deptno;

* 2 rows will be printed.

O/P ⇒ 50000
45000

Date: Page No:

* select deptno , sum (sal) from emp ✓
group by deptno ;

* A non-aggregate column can't be selected with aggregate function unless that column is grouped.

O/P ⇒

deptno	sum(sal)
10	50000
20	45000

* groups are also formed for NULL. ie if deptno. contains a NULL, then one more row would be printed in O/P.

—x—

* select ename from emp ✗
group by deptno. ;

* If we've done grouping, then we can only use aggregate functions or grouped column (deptno. here)

—x—

* select deptno from emp group by deptno.;

⇒ 10
20

* select deptno. from emp;

⇒ 10
10
20
20

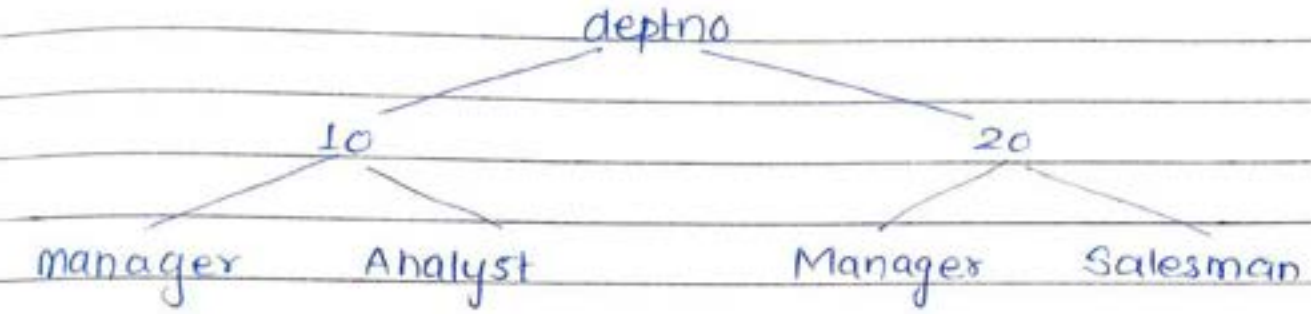
* select distinct deptno from emp;

⇒ 10
20

Date: Page No:

* Grouping can also be done on multiple columns.

⇒ select deptno, job, sum(sal) from emp
group by deptno, job;



O/P ⇒

deptno	job	sum (sal)
10	Manager	10000
10	Analyst	20000
20	Manager	30000
20	salesman	40000

* 'where' clause works row-wise.
* 'having' clause works group-wise.
thus 'having' clause comes after 'groupby' statement.

Que Write a query to find the deptno where average salary of employee is more than 5000.

⇒ select deptno from emp
where avg(sal) > 5000 ✗
group by deptno;

* aggregate functions never use 'where' clause, they use 'having' clause to check conditions.

⇒ select deptno from emp
group by deptno
having avg(sal) > 5000;

Que Write a query to find the deptno where average salary of managers is greater than 5000.

⇒ select deptno from emp where job = 'Manager'
group by deptno.
having avg(sal) > 5000;

* First, managers are selected then grouping will be done.

* 'having' clause can be used without grouping. but, then complete table will be treated as a single group. Date: Page No:
~~but we can use only aggregate functions. in select.~~

⇒ select deptno from emp ✗
group by deptno
having avg (sal) > 5000 and job = 'Manager';

* job is not unique for a group

⇒ select deptno from emp ✓
group by deptno, job
having avg(sal) > 5000 and job = 'Manager';

⇒ select deptno from emp where job = 'Manager' ✓
group by deptno, job
having avg(sal) > 5000 and job = 'Manager';

. To find deptno where avg. salary is greater than 5000.

⇒ select deptno, asal from (select deptno, avg(sal) as asal group by deptno) from emp.
where asal > 5000;

π ⇒ project ('select' in SQL)

σ ⇒ select ('where' in SQL)

Date: Page No.

Relational Algebra :

* It is a theoretical language of query writing & SQL is practical implementation of relational algebra.

⇒ select ename from emp

⇒ π_{ename} (emp)

⇒

ename		SQL	RA
e1	⇒	e1	e1
e2		e2	e2
e3		e3	e3
e3		e3	

* set notation ⇒ duplication ✗ (default)
  bag notation ⇒ duplication ✓

* Relational algebra is based on set theory.

⇒ select distinct ename from emp = π_{ename} (emp)

eg π_{ename} σ sal > 1000 (emp)

• $\pi_{\frac{sal * 10}{1000}}$ ✓

eg π_{ename} σ sal > 10000 ∧ deptno = 10 (emp)

$\sigma_{sal > 1000}$ (emp) ⇒ select * from emp where sal

Date _____ Page No. _____

Nested Query : (query within a query)

Que Write a query to find the name of employees having highest salary.

⇒ select ename from emp where
sal = (select max(sal) from emp)

* first inner query is executed.

Que Write a query to find the name of employees having salary greater than that of e1.

⇒ select ename from emp where
sal > (select sal from emp where ename = 'e1');

* if there is no e1 ⇒ inner query will return NULL. ✓

* if there are more than 1 e1 ⇒ error

⇒ Thus, to resolve this,

① ~~sal~~ > Any
② > All

③ .. sal > (select max(sal) / min(sal) from emp where ename = 'e1'

Date: Page No:

① → select ename from emp where
sal > Any (select sal from emp
where ename = 'e1');

* sal > 5000 / 10000 ⇒ sal > 5000

② ⇒ select ename from emp where
sal > All (select sal from emp
where ename = 'e1');

* sal > 5000 & 10000 ⇒ sal > 10000

Que write a query to find the name of employees having salary less than that of e1.

⇒ select ename from emp where
sal < (select sal from emp where ename = 'e1');

* if there is no e1 ⇒ inner query will return NULL v

* If there are more than 1 e1 ⇒ error

* Thus, to resolve this,

① < Any
② < All

Date: _____ Page No: _____

① ⇒ select ename from emp where
sal < Any (select sal from emp where ename = 'e1');

* sal < 5000 / 10000 ⇒ sal < 10000

② ⇒ select ename from emp where
sal < All (select sal from emp where ename = 'e1');

* sal < 5000 & 10000 ⇒ sal < 5000

---

* ⇒ select ename from emp where
sal > Any (select sal from emp where ename = 'e1');

* if there is no e1 ⇒ inner query will return NULL

⇒ sal > Any (NULL)

⇒ select ename from emp where
sal > All (select sal from emp where ename = 'e1');

* if there is no e1 ⇒ inner query will return 0 (zero).

⇒ sal > All (0)

* it will display ename of sal > 0

Date: Page No:

Que write a query to find the name of employees having salary equal to that of e1.

⇒ select ename from emp where
sal = (select sal from emp where ename = 'e1');

* if there is no e1 ⇒ inner query will return NULL ✓

* if there are more than 1 e1 ⇒ error

⇒ Thus, to resolve this,

① = Any
② = ~~All~~ in } Both are same.

① ⇒ select ename from emp where
sal = Any (select sal from emp where ename = 'e1');

② ⇒ select ename from emp where
sal in (select sal from emp where ename = 'e1');

Date: Page No:

emp:

ename	deptno
e1	10
e2	20
e3	20
e4	10

dept:

deptno	dname
10	Research
20	Accounts

Que Write a query to find the name of employees of Research dept.

⇒ select ename from emp where
deptno = (select deptno from dept where
dname = 'Research') ;

Que Write a query to find the name of employees having 2nd highest salary in emp table.

⇒ select ename from emp where
sal = (select max(sal) from emp where
sal != (select max(sal) from emp)) ; ✓

OR

⇒ select ename from emp where
sal = (select max(sal) from emp where
sal < (select max(sal) from emp)) ; ✓

Date: _____ Page No: _____

Que write a query to find 3rd highest salary - employee.

⇒ select ename from emp where
sal = (select max(sal) from emp where
sal != (select max(sal) from emp where
sal != (select max(sal) from emp))) ; ✗

✗ It will return 1st highest salary.

⇒ select ename from emp where
sal = (select max(sal) from emp where
sal < (select max(sal) from emp where
sal < (select max(sal) from emp))) ; ✓

- $\Pi_{sum(sal)}$ ✗ . For aggregate functions, we can't use project (Π).

→ $\mathcal{G}_{sum(sal)}(emp)$

- for grouping, (on deptno column) ⇒ $\mathcal{G}_{deptno/sum(sal)}(emp)$

" ← " ⇒ assignment (in R.A)

Date: Page No:

Joins

Que Write a query to print the employee name with resp. dept.

* join is used to fetch the data from more than 1 table

⇒ select ename, dname from emp, dept.

* It is cartesian product of 2 tables. Each row of emp table is 'join' to each row of dept table.

* If one table contains m rows & another contains n rows, then their cartesian product will have m*n rows.

Que Write a query to find the name of employees having salary greater than that of e1. (in R.A)

⇒ $S \leftarrow \pi_{sal}\, \sigma_{ename = 'e1'}(emp)$ // inner query

$\pi_{ename}\, \sigma_{sal >= S}(emp)$ // outer query

---

* select * from emp, dept ⇒ 8 rows & 4 columns

* equal condition on common field ⇒ Natural Join.

Date: _____ Page No: _____

* If one table contains 10 rows & other contains no rows (∅), then their cartesian product will also be ∅.

Natural Join:

⇒ select ename, dname from emp, dept where emp.deptno = dept.deptno;

* if columns are common in both the tables, then write it as: ⇒ tablename.columnname else there will be error.

eg emp.deptno, dept.deptno.

⇒ emp:

ename	deptno
e1	10
e2	20
e3	20
e4	10

dept:

deptno	dname
10	Research
20	Accounts

O/P ⇒

e1	Research
e2	Accounts
e3	Accounts
e4	Research

* This operation is called Natural Join. (equality on common column)

Date _____ Page No _____

* Alternate way :

⇒ select ename, dname from emp natural join dept;

* select * from emp natural join dept;

O/P ⇒ 3 columns (common column is displayed only once (deptno here)).

* If there is no common column, it will simply perform cartesian product.

* If there are more than 1 column, it will equate all the columns.

eg R :

A	B
a1	b1
a2	b2
a3	b2
a4	b1

S :

A	B
a1	b2
a2	b2
a3	b2
a4	b2

* select * from R natural join S.

O/P ⇒

A	B
a2	b2
a3	b2

Date: _____ Page No: _____

* But if want to equate only 1 column.

⇒ select * from R natural join S using A

O/P ⇒	A	R B	S B
	a1	b1	b2
	a2	b2	b2
	a3	b2	b2
	a4	b1	b2

---

* In Relational Algebra, cartesian product ⇒ X

eg $\pi_{ename, dname}$ (emp X dept)

* natural ~~join~~ join ⇒ * or ⋈

eg $\pi_{ename, dname}\ \sigma_{emp.deptno = dept.deptno}$ (emp X dept)

OR $\pi_{ename, dname}$ (emp * dept)

OR $\pi_{ename, dname}$ (emp ⋈ dept)

> If R & S are same schema (ie same no. of rows & columns)
then, R * S = R ∩ S

Date _____ Page No. _____

⇒ R:

A	B
a1	b1
a2	b1
a3	b2
a4	b2

S:

A	B
a1	b1
a2	b2
a3	b3
a4	b2

⇒ $\pi_A (R * S)$ ⇒ a1, a4

⇒ $\pi_A (R) * \pi_A (S)$ ⇒ a1, a2, a3, a4

Both are different.

⇒ R:

A	B
a1	b1
a2	
a3	b3

S:

B	C
b1	c1
b2	c2
	c4
b3	c3

• select * from R natural join S.

⇒ O/P ⇒

a1	b1	c1
a3	b3	c3

* In cartesian pro. all columns are display but in natural join common column is displayed once

* NULL - NULL join ✗

eg a2 c4 ∝

Date: _____ Page No: _____

Outer Joins :

* They are the extension of natural join. They displays unmatched rows also.

① Left Outer Join : displays unmatched rows of left table.

② Right Outer Join : displays unmatched rows of right table.

③ Full Outer Join : displays unmatched rows of both tables.

⇒ R.:

A	B
a1	b1
a2	
a3	b3

S:

B	C
b1	c1
b2	c2
	c4
b3	c3

⇒ Left Outer Join

A	B	C
a1	b1	c1
a3	b3	c3
a2		

⇒ Right Outer Join

A	B	C
a1	b1	c1
a3	b3	c3
	b2	c2
		c4

Date: Page No:

⇒ Full Outer Join :

⇒

A	B	C
a1	b1	c1
a3	b3	c3
a2		
	b2	c2
		c4

In SQL,

R → left table
S → right table

* select * from R left outer join S
* select * from R right outer join S
* select * from R full outer join S.

In R.A.,

* $\Pi_{A,C}$ (R ⟕ S) ⇒ left outer join
* $\Pi_{A,C}$ (R ⟖ S) ⇒ right outer join
* $\Pi_{A,C}$ (R ⟗ S) ⇒ full outer join

Date _____ Page No. _____

R ⇒ left table
S ⇒ right table

In ORACLE,

⇒ select * from R, S ⇒ natural join
where R.B = S.B

⇒ select * from R, S ⇒ left outer join
where R.B = S.B (+)

⇒ select * from R, S ⇒ right outer join
where R.B (+) = S.B

⇒ select * from R, S ⇒ full outer join
where R.B(+) = S.B (+)

Date: _____ Page No: _____

Primary key : unique & not null.

* only 1 primary key is allowed.

* All those keys which are capable of being primary are known as candidate keys.

eg rollno, enrollmentno are 2 candidate keys.

⇒ create table Student (Rollno number(6) primary key,

Enrol varchar(10) unique not null,

name varchar(10) not null,

age number(2) check (age between 21 and 28),

branch varchar(6) default 'CS');

* check constraint is checked on insertion & updation.

⇒

Rollno	courseno	grade
R1	c1	A
R1	c2	B
R2	c1	B
R2	c3	A

* Here no columns is unique, but combination of columns is unique eg. (Rollno., courseno) Thus, here, candidate key consists of 2 columns.

* candidate key is said to be composite candidate key if it is formed by more than 1 column.

⇒ create table Student (Rollno number (6),

courseno number(6), grade varchar(2),

primary key (Rollno, courseno)) ;

* Rollno. & courseno. cannot be null & (duplicate at same row) as they are part of candidate key / primary key.

* Individual attributes can be duplicate but not null.

Date _____ Page No. _____

Foreign Key :

emp:

ename	sal	deptno

dept:

deptno	dname
10	Research
20	Design
30	Testing

⇒ Can we insert (e2, 1000, 20) ✓

⇒ Can we insert (e3, 2000, 40) ✗ entire row will be rejected.

* dept table is parent table & emp is child table.

* Whenever there's insertion or updation in child table, parent table is referred. If that value is present in parent table then only it can be inserted in child table.

* Now, deptno. column of emp table is declared as foreign key.

* We can declare foreign key in child table only when that column is either primary key or unique in parent table.

—x—

* Database ensures entity integrity by giving facility to declare primary key.

* Database ensures referential integrity by giving facility to declare foreign key.

---

① create parent table :

⇒ create table dept (deptno number(2) primary key, dname varchar(20));

* deptno must be primary or unique.

② create child table :

⇒ create table emp (empname varchar(10) primary key, deptno number(2) foreign key references dept);

* foreign key can accept any no of NULL values.

* In dept table, deptno can't be NULL as it's primary key, but it can be NULL in emp table as it's foreign key in emp table.

Date: Page No:

emp:

ename	deptno
e1	10
e2	10
e3	20

dept:

deptno	dname
10	Research
20	Accounts
30	Academics

⇒ update dept set deptno = 100 where deptno = 10; ✗

* we cannot update / delete any value from parent table when that value is present in child table.

* But, if you want to delete / update, then there are 2 ways;

① on delete null : If we delete any value from dept table, that value will become NULL in emp table.

⇒ create table emp (ename varchar(10) primary key, deptno number(2) foreign key references dept on delete null);

* on update null & on update cascade also exists.

⇒ on update cascade, if there's updation in parent table, there will be updation in child table automatic-

② on delete cascade : If we delete any value from dept, that entire row will become NULL in emp table.

⇒ create table emp (ename varchar (10) primary key, deptno number (2) foreign key references dept on delete cascade);

→

* parent-child relationship can be there b/w columns within same table.

eg

empno	ename	mgr
1	e1	
2	e2	1
3	e3	4
4	e4	3

* empno ⇒ parent key

* mgr ⇒ child key

* mgr can accept those values which are present in empno

→

Que B is foreign key referring to column A of same table. On delete cascade is set.

⇒ delete from R where A = 1;

How many rows will be deleted?

Date: Page No:

R :

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

Ans : All the rows will be deleted as on delete cascade is set.

* 'with' clause is used to create temporary table.

- To display employee name drawing highest salary.

⇒ With large_sal (msal) as select max(sal) from emp ;
(table, column)

select ename from emp, large_sal. where sal = msal ;

- To display deptno whose largest salary is more than avg. salary

⇒ With large_sal (deptno, msal) as select deptno, max(sal) from emp group by deptno.

With avg_sal (asal) as select avg(sal) from emp

Select deptno from large_sal, avg_sal where msal > asal

Date: Page No:

Self Join : joining a table to itself.

emp:

empno	ename	mgrno
1	e1	
2	e2	1
3	e3	1
4	e4	2

Que write a query to print the employee name with manager name.

* This query can't be solved with one table as we want data from 2 rows. Thus, we need 2 tables.

2 copies

x_1

empno	ename	mgrno
1	e1	
2	e2	1
3	e3	1
4	e4	2

x_2

empno	ename	mgrno
1	e1	
2	e2	1
3	e3	1
4	e4	2

⇒ select x1.ename "employee", x_2.ename "Manager"
from emp as x1, emp as x2
where x1.mgrno = x2.empno;

Date: _____ Page No: _____

x_1 & x_2 are known as alias or instances or virtual copies of the table & these are valid till query is under execution.

O/P ⇒

employee	Manager
e2	e1
e3	e1
e4	e2

* This query can be solved by making only 1 copy.

⇒ select emp.ename "employee", x2.ename "Manager"
from emp, emp as x2
where emp.mgrno = x2.empno ;

In relational algebra, above query will be,

⇒ $\pi_{emp.ename, x1.ename}\ \sigma_{emp.mgrno = x1.empno}$

$(emp \times \rho(emp, x1))$!

* ρ is operator for rename.

* In above query, we rename the table but in relational algebra, we can write query of self-join also by renaming column, it can't be done in SQL.

* For natural join, 'using' is used. (Here, 'on' can also be used)
* For other joins, 'on' is used.

Date: Page No:

⇒ By renaming column ;

empno	ename	mgrno	// column names original
eno	name	mno	// in 2nd copy (table in 1st copy)

⇒ $\Pi_{ename, name}$ (emp $\underset{mgrno = eno}{\bowtie}$ ρ(eno, name, mno) emp)

---

* Equal on common columns → Natural Join
* Equal on uncommon columns ⇒ Equi Join
* Unequal, greater than, etc. on columns ⇒ Theta Join

eg $\underset{mgrno \neq eno}{\bowtie}$, $\underset{mgrno > eno}{\bowtie}$, etc.

* all joins are theta joins.

---

* All joins except outer join are inner joins.

eg
```
select x1.ename as "employee", x2.ename as "manager"
from emp as x1 join emp as x2
    on x1.mgr = x2.empno
```
↗ theta join

Date _____ Page No _____

Set Operations :

① Union :

credit:	debit:
cname	dname
c1	d1
c2	d2
c3	d3
d1	c1

Que write a query to print the names of account holders

⇒
```
select cname from credit
        union
select dname from debit;
```

O/P ⇒

cname
c1
c2
c3
d1
d2
d3

* It will display column name of first table i.e. cname

* If any value is common in both the tables then these set operations will suppress those duplicate values. (by-default) (eg c1, d1 here)

* But if we want to display duplicate values too.

⇒ select cname from credit
union all
select dname from debit ;

* types of columns must be same, their size can be different.

eg: cname varchar (20) , dname varchar (60) ✓

* The union of 2 tables can also be performed if they are union compatible to each other.

⇒ Two relations are said to be union compatible if there exists one to one correspondence between their columns. ie. if first table contains 3 columns then 2nd table must have 3 columns & the types must be like ; (size doesn't matter)

* type of a1 = type of b1
* type of a2 = type of b2
* type of a3 = type of b3

* ~~size of data type doesn't matter.~~

A:

a1	a2	a3

B:

b1	b2	b3

Date: _____ Page No: _____

eg credit :

cno	cname
1	c1
2	c2
3	d1

debit :

dno	dname
1	d1
2	d2
3	d1

⇒ select * from credit
union
select * from debit

O/P ⇒

cno	cname
1	c1
2	c2
3	d1
1	d1
2	d2

* if entire tuple is duplicate, then it will be displayed only once.

eg credit :

addr	cname
XYZ	c1
IJK	c2
123	d1

debit :

dname	addr
d1	ABC
d2	DEF
d3	GHI

⇒ select * from credit
union
select * from debit

O/P ⇒

addr	cname
XYZ	c1
IJk	c2
123	d1
d1	ABC
d2	DEF
d3	GHI

② Intersection :

⇒ select * from credit
intersect or intersection
select * from debit

eg credit :

cno	cname
1	c1
1	c1

debit :

dno	dname
1	c1
1	c1

O/P ⇒

cno	cname
1	c1

⇒ select * from credit
intersect all or intersection all
select * from debit

Date: Page No:

O/P ⇒

cno	cname	
1	c1	} twice, not
1	c1	} 4 times

③ Minus / Except :

* A − B ⇒ present in A and not in B

⇒ select * from credit
minus / except
select * from debit

eg

credit:

cno	cname
1	c1
2	c2
3	c3

debit:

dno	dname
1	d1
2	c2
3	d3

O/P ⇒

Cno	Cname
1	c1
3	c3

* In Relational algebra, for two relations R & S,

① union ⇒ R ∪ S
② intersection ⇒ R ∩ S
③ minus ⇒ R − S

Date _____ Page No. _____

Que what is the estimated size of natural join?

R : (child table)

A	B(FK)
a1	b1
a2	b1
a3	b2
a4	b3

S : (parent table)

B(PK)	C
b1	c1
b2	c1
b3	c2
b4	c3

* If table R contains n tuples & S contains m tuples then their natural join can have maximum n tuples & minimum 0 tuples as there can be NULL values in column B.

Que R:

A.	B
a1	b1
a2	b1
a3	b1
a4	b1
a5	b1
a6	b1

S:

B	C
b1	c1
b1	c2
b1	c3

* no. info is given about primary & foreign key.

* If table R contains n tuples & S contains m tuple then their natural join can have maximum m * n tuples & minimum 0 tuples.

Date: _____ Page No: _____

Correlated Subqueries :

Que write a query to find the name of employees having highest salary in their dept.

① Nested Query :

⇒ select ename from emp where sal in (select max(sal) from emp group by deptno);

ename	sal	deptno
e1	10000	10
e2	20000	10
e3	30000	20
e4	40000	20
e5	20000	20

O/P ⇒ e2
e4
e5

⇒ select ename from emp where (sal, deptno) in (select max(sal), deptno from emp group by deptno) ;

O/P → e2
e4

(2) Correlated Subquery :

* In correlated subqueries, inner query is executed whenever 'where' clause is executed & 'where' clause works row wise, thus, inner query is executed per row.

⇒ select e1.ename from emp as e1 where

```
e1.sal = ( select max(sal) from emp
           where deptno = e1.deptno );
```

* When inner query contains reference of outer table, it is correlated subquery, not nested query.

* Outer query is executed per row while the inner query is executed per entire table.

—x—

Date: _____ Page No: _____

③ Exists

* if inner query returns some rows → true

* if inner query doesn't return any row ⇒ false

⇒ select x1.ename from emp as x1 where

not exists (select * from emp where
deptno = x1.deptno and sal > x1.sal);

O/P ⇒ e2
e4

—x—

* 'exists' can also be used with nested query.

⇒ But, it will display either entire table or nothing.

as in nested queries, inner query is run only once.

ename	sal	deptno
e1	10000	10
e2	20000	10
e3	20000	20
e4	30000	20

⇒ select ename from emp exists (select * from emp where deptno = 10);

Date: Page No:

⇒ exists (TRUE) ⇒ displays entire table ; as in nested query, inner query is executed only once.

unique:

Customer (Cno, Cname) , Purchase (Cno, itemno)

* To find the name of customer who has bought only 1 items

⇒ select cname from Customer where unique (select cno from Purchase where Purchase.Cno = Customer.Cno).

Purchase:

Cno	itemno
1	I1
1	I2
2	I1

⇒ unique (1, 1) → not unique

⇒ unique (2) ⇒ unique

Customer:

Cno	Cname
1	C1
2	C2

O/P ⇒ C2

eg unique (1,2) → unique

alter table emp Add (name number(6)); // add column

* existing rows will have NULL for this new column

Date: Page No:

⇒ SQL commands are divided into :

① DDL (Data Definition Language)

* to change structure of table / database

⇒ create, drop, alter, etc.

② DML (Data Manipulation Language)

* to manipulate data.

⇒ select, insert, update, delete, etc.

③ DCL (Data Control Language) : grant, revoke

④ TCL (Transaction control Language): commit, rollback.

⇒ commit, rollback, grant, revoke

* commit :

* Whatever changes we perform, are saved in primary memory. To save it permanently in harddisk, 'commit' is used.

* Autocommit or exitting software correctly saves changes permanently.

Date: Page No:

- Rollback :

* till last commit, 'rollback' undo the changes we performed.

eg
```
commit
insert into emp values (100, _ _ );
delete t1
create table t2 ( );
update student set phone = 9933101601
    where rollno = 2 ;
select * from emp
drop table t3
Rollback
```

they're auto committed.

* DDL commands can't be rolled back. Thus, dropped tables cannot be recovered.

—x—

- Grant :

users : u1 u2
tables : t1 t2

* Users cannot access the tables of other users. If u1 wants to access t2 then u2 will run a 'grant' command to give permission to u1 to access t2.

Date: _____ Page No: _____

u2 ⇒ grant select, insert on t2 to u1

* u1 cannot run update, delete, etc. commands on t2.

u2 ⇒ grant all on t2 to u1

all ⇒ select, insert, update, delete, create

* Now if u1 wants to give permissions further to other users.

u2 ⇒ grant all on t2 to u1 with grant option.

- Revoke :

* to take given permission back, 'revoke' is used.

u2 ⇒ revoke all on t2 from u1

u2 ⇒ revoke select on t2 from u1

Date: _____ Page No: _____

View : They are virtual tables.

⇒ create view v1 as
(select ename, empno from emp)

⇒ select * from v1.

* Views are known as virtual tables because they don't have their own database, they fetch data from table on which they're defined.

⇒ In grant command, we give permission to access entire table, but if we want other users to access only some selected columns then we can create views.

* It is possible to insert/update in the original table with the help of views; provided that view satisfies following conditions;

(1) view should be single table based. (there should not be any join or nested query in view definition)

(2) view definition should not contain any aggregate functions.

(3) primary key & all 'not null' keys must be the part of view definition.

$Q_1(x,y) \div Q_2(x)$

$\Pi_{(x,y)} \div \Pi_x$

* Q_2 can project only those columns which are being 'project' by Q_1

$x \subseteq (x,y)$

Date: Page No:

⇒ It'll select those values of y in which all values of x with y are present

⇒ create table emp (empno varchar (10),

deptno number (6) check (deptno in (select deptno from dept));

* This query is not equivalent to foreign key because, in foreign key, constraint is on both tables while here constraint is only on emp table.

* There's no constraint on dept table ; if a record is updated in dept table, it won't affect emp table.

---

* "For all" type of queries, divide operator is used.

eg

Part: Pno	Pcolor	Pname
1	Red	P1
2	Green	P2
3	Red	P3
4	Blue	P4

Supplier: Sno	Pno
S1	1
S1	2
S1	3
S2	3
S2	4
S3	1
S3	3

• Query to find the supplies which supply all red color parts.

⇒ $\Pi_{sno,pno}(Supplier) \div \Pi_{pno}\, \sigma_{Pcolor='red'}(Part)$ O/p ⇒ S1, S3

R.A. is procedural but Rel. Calculus is non-procedural langu

Date: Page No:

Relational Calculus : It is also a theoretical language of query writing.

1. Tuple Calculus (Row)
2. Domain Calculus (Column)

* Relational calculus is based on predicate / first order logic.

Tuple Calculus :

⇒ $t \mid t \in emp$ ⇒ select * from emp

⇒ $t.ename \mid t \in emp$ ⇒ select ename from emp

OR

~~⇒ $t \mid u \in emp \wedge t = u[ename]$ ⇒ select ename from emp~~

⇒ $t \mid \exists d \in emp(t[ename] = d[ename])$ ⇒ from emp

⇒ $t \mid \exists U \in emp \left(t^{[ename]} = U[ename] \wedge U[sal] > 10000 \right)$

OR

⇒ $t.ename \mid t \in emp \wedge t.sal > 10000$

⇒ select ename from emp where sal > 10000

* Like R.A, R.C. also avoids duplicate values.

Date: _____ Page No. _____

Que Write a query to find the name of employees of Research dept. (in Tuple calculus).

⇒ $t.ename \mid t \in emp \wedge \exists d \in dept \wedge$

$d.dname = 'Research' \wedge d.deptno = t.deptno$

OR

⇒ $t \mid \exists u \in emp \left(t^{[ename]} = u[ename]\right) \wedge$

$\exists d \in dept \left(d[dname] = 'Research' \wedge d[deptno] = u[deptno]\right)$

* if $\exists$ is replaced by $\forall$, this query will only work if dept table has only one row & the dname of that row is Research.

—x—

* variable with any quantifier ⇒ Bounded or quantified variable
* variable without quantifier ⇒ free variable.

* A relational calculus query can have only one free variable.

eg $\underset{\text{free}}{t} \mid \underset{\text{quantified}}{\exists d \in emp} \ldots$

emp

empno	ename	sal
(a)	(e)	(s)

Date _____ Page No. _____

Domain Calculus :

⇒ $e \mid \langle a, e, s \rangle \in emp$ ⇒ select ename from emp

⇒ $\langle a, e, s \rangle \mid \langle a, e, s \rangle \in emp$ ⇒ select * from emp

⇒ $e \mid \langle a, e, s \rangle \in emp \wedge s > 10000$

⇒ select ename from emp where sal > 10000

Que Write a query to find the name of employees of Research dept. (in Domain calculus)

emp :	ename	deptno	dept :	deptno	dname
	(e)	(d)		(dno)	(n)

⇒ $e \mid \langle e, d \rangle \in emp \wedge \exists \langle dno, n \rangle \in dept \wedge$

$(n = \text{'Research'} \wedge d = dno)$

Date: Page No:

* The expressive power of Relational Algebra & Relational Calculus is equal for safe queries.

* It is not possible to write unsafe queries in Relational Algebra, but in Relational calculus.

eg ⇒ $t \mid t \notin emp$ (select from all tables except emp table)

* It's an unsafe query, & can't be written in Relational Algebra & SQL.

—x—

Scalar Subquery:

dept: deptno	dname	emp: ename	deptno
10	Research	e1	10
20	Design	e2	10
		e3	20
		e4	20
		e5	20

- To display no of employees in every dept.

→ select dname, (select count(*) from emp where emp.deptno = dept.deptno) as totalemployee from dept;

o/p ⇒

dname	totalempl-
Research	2
Design	3

(inner query)

* It should return only one value, thus, it's known as scalar.

Date: _____ Page No: ___

Supplier:

Sno	Pno
S1	P1
S1	P3
S2	P2
S2	P4
S2	P1

Part:

Pno	Pcolor
P1	red
P2	blue
P3	red
P4	green

eg $t \mid \exists s \in$ supplier $(\exists P \in$ Part $\wedge$ S.Pno = P.Pno $\wedge$ P.color = 'red' $\wedge$ t.Sno = S.Sno)

(there exists) (some red part)

eg (all red parts) (for all)

$\Rightarrow t \mid \exists s \in$ Supplier,

$(\forall P \in$ Part $\blacktriangle$ ~~(Pcolor ≠ 'red')~~

$\wedge \blacktriangledown ((P.color \neq 'red') \blacktriangledown (S.Sno = P.Sno))$

Supplier:

Sno	Sname
S1	A
S2	B
S3	C

Part:

Pno	Pcolor	Sno
P1	red	S1
P2	blue	S2
P3	red	S1
P4	green	S3

NORMALIZATION

Date: Page No:

* Functional Dependency (FD)

⇒ given a rollno, you will get only one name. Thus, it's an FD (Rollno → Name)

⇒ given a name, you will get many rollno. Thus, it's not an FD.

* FDs are decided while creating tables.

eg

X	Y	Z
x1	y1	z1
x1	y1	z2
x2	y2	z3
x2	y2	z4

Here, X → Y holds ✓

X → Z ✗

* Z → X, Z → Y ✓

* Here, Z is unique, thus it'll always decicle other attributes.

eg

X	Y	Z
x1	y1	z1
x1	y2	z2
x2	y1	z3
x2	y2	z4

⇒ XY → Z , Z → X , Z → Y , - - - - -

combination is unique.

Que

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y2	z2
x4	y1	z1

Que: what you can conclude on this database;

(a) X → Y & Y → Z (b) Y doesn't decide X ✓

⇒ On this instance,

- X → Y ✓
- X → Z ✓
- Y → X ✗
- Y → Z ✓
- Z → X ✗
- Z → Y ✓
- XY → Z ✓
- YZ → X ✗
- XZ → Y ✓

Date: _____ Page No: _____

Laws related to FD :

① Reflexivity : An attribute always decides itself.

$X \rightarrow X$

② Transitivity :

If $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$

③ Pseudo Transitivity :

If $X \rightarrow Y$ & $YZ \rightarrow W$ then $XZ \rightarrow W$

④ Augmentation :

If $X \rightarrow Y$ then $XZ \rightarrow Y$ & $XZ \rightarrow YZ$

eg

Rollno	name	add
1	n1	a1
2	n2	a2

If rollno → name
then,

rollno add → name & rollno. add → name add.

Date: Page No.

⑤ Additivity :

If $X \to Y$ & $X \to Z$ then $X \to YZ$

⑥ Projectivity :

If $X \to YZ$ then $X \to Y$ & $X \to Z$

Note : You cannot break LHS.

$AB \to C \implies A \to C$ & $B \to C$ ✗

* These rules are used to find closure of attributes.

eg $R(A, B, C, D)$

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

* closure of A contains all those attributes which are decided by A directly or indirectly.

- $\{A\}^+ = \{ABCD\}$. $\{B\}^+ = \{BCD\}$
- $\{C\}^+ = \{CD\}$. $\{D\}^+ = \{D\}$

* The attribute whose closure contains all the attributes of the relation, is known as candidate key. (A here)

R(A, B, C, D, E) ⇒ if A is candidate key, how many superkeys?

⇒ $\underset{(1)}{A} + {}^4C_2 + {}^4C_3 + \underset{(1)}{{}^4C_4} = \underline{16}$ Ans

Date: Page No:

eg R(A, B, C, D)

A → B
B → C
C → D
D → A

⇒ $\{A\}^+ = \{ABCD\}$

⇒ $\{B\}^+ = \{BCDA\}$

⇒ $\{C\}^+ = \{CDAB\}$

⇒ $\{D\}^+ = \{DABC\}$

* All 4 are candidate keys.

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

A → B
C → D

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

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

Here, candidate key must be composite.

* Those attributes which are only in L.H.S. of FDs, will compulsorily be in candidate key (A, C here)

* Those attributes which are not a part of any FD, will also compulsorily be in candidate key (E here)

⇒ $\{ACE\}^+ = \{ACEBD\}$ ⇒ <u>CK</u>

R(A, B, C, D, E) if A & B are C.K., how many super keys?

⇒ 16 + 16 − 8 = 24 (inclusion exclusion)

Date: _____ Page No. _____

eg R(S, C, D, G, V)

S → C
SD → G
CG → V

⇒ {SD}⁺ = {SDCGV} ⇒ CK

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

A → B
BC → D
E → C
D → A

⇒ {EH}⁺ = {EHC} ×

⇒ {AEH}⁺ = {AEHBCD} ⇒ CK

⇒ {BEH}⁺ = {BEHCDA} ⇒ CK

⇒ {CEH}⁺ = {CEH} ×

⇒ {DEH}⁺ = {DEHACB} ⇒ CK

(A, B, C, D, E), max. no. of super keys?

→ For max. no. of super keys, assume all 5 are C.K.

→ $^5C_1 + {}^5C_2 + {}^5C_3 + {}^5C_4 + {}^5C_5 = 2^5 - 1 (31)$ Ans

Date: Page No:

eg R(A, B, C, D, E, F, G, H)

CH → G
A → BC
B → CFH
E → A
F → EG

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

$\Rightarrow \{AD\}^+ = \{ADBCFHEG\} \Rightarrow$ CK

$\Rightarrow \{BD\}^+ = \{BDCFHEGA\} \Rightarrow$ CK

$\Rightarrow \{ED\}^+ = \{EDABCFHG\} \Rightarrow$ CK

$\Rightarrow \{FD\}^+ = \{FDEGABCH\} \Rightarrow$ CK

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

AB → CD
DE → F
C → E
F → C

$\Rightarrow \{ABG\}^+ = \{ABGCDEF\} \Rightarrow$ CK

* Trivial FD : An FD is said to be trivial if $\beta \subseteq \alpha$

$\alpha \rightarrow \beta$ * trivial FDs always hold.

eg AB → B, A → A, etc

R(A, B, C, D, E), max. no of keys that can be C.K simultaneously?

→ (A, B, C, D, E) ⇒ $\leq X$ $(^5C_1)$

→ (AB, AC, AD, AE, BC, BD, BE, CD, CE, DE) = 10 $(^5C_2)$ also $(^5C_3)$

Date: Page No.

⇒ How to find closure of set of FDs (F^+):

⇒ R (A, B, C)

$$F = \begin{Bmatrix} A \to B \\ B \to C \end{Bmatrix}$$

• ~~How many FDs can be~~ be possible.

⇒ How many FD's can be derived from given FD set.

⇒ $A^+ = \{ABC\}$ * it's closure contains 3 attributes, thus, there will be $2^3 = 8$ FD's in F^+

⇒ $B^+ = \{BC\}$ ⇒ $2^2 = 4$

⇒ $C^+ = \{C\}$ ⇒ $2^1 = 2$

⇒ $\{AB\}^+ = \{ABC\}$ ⇒ $2^3 = 8$

⇒ $\{BC\}^+ = \{BC\}$ ⇒ $2^2 = 4$

⇒ $\{AC\}^+ = \{ABC\}$ ⇒ $2^3 = 8$

⇒ $\{ABC\}^+ = \{ABC\}$ ⇒ $2^3 = 8$

∴ total FD's in F^+ = 8 + 4 + 2 + 8 + 4 + 8 + 8 = 42

+ 1 ($\emptyset \to \emptyset$)

= 43

$F^+ = \{ A \to \emptyset,\ A \to A,\ A \to B,\ A \to C,\ A \to AB,\ A \to BC,\ A \to AC,\ A \to ABC,\ \ldots\ 43 \}$

If a rel. contains 10 attributes, max. how many keys can be d.k simult....

→ max. value → $^{n}C_{(n/2)}$ → $^{10}C_5 = \underline{252}$

ANS

Date: Page No.

eg R (A, B)

$$F = \left\{ \begin{matrix} A \to B \\ B \to A \end{matrix} \right\}$$

⇒ $A^+ = \{AB\}$ ⇒ $2^2 = \underline{4}$

⇒ $B^+ = \{AB\}$ ⇒ $2^2 = \underline{4}$

⇒ $\{AB\}^+ = \{AB\}$ ⇒ $2^2 = \underline{4}$

∴ total FD's in F^+ = $4+4+4 = \underline{12} + 1\ (\phi \to \phi) = \underline{13}$

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

A → B

A → C

CD → E

B → D

E → A

Q: Which of the following FD is not implied by above FD set?

↑ Another way to ask this question.

Q: Which of the following is not a part of closure of given FD set?

(a) CD → AC (b) BD → CD

(c) BC → CD (d) AC → BC

• For this, don't find entire closure just find closure of LHS of FD & if RHS is present then it'll be part of closure else not

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

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

eg R (A, B, C, D)

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

$\Rightarrow A^+ = \{AB\} \Rightarrow 2^2 = 4$, $\Rightarrow B^+ = \{B\} \Rightarrow 2^1 = 2$

$\Rightarrow C^+ = \{CD\} \Rightarrow 2^2 = 4$, $\Rightarrow D^+ = \{D\} = 2^1 = 2$

$\Rightarrow \{AB\}^+ = \{AB\} \Rightarrow 2^2 = 4$, $\Rightarrow \{BC\}^+ = \{BCD\} \Rightarrow 2^3 = 8$

$\Rightarrow \{AC\}^+ = \{ABCD\} \Rightarrow 2^4 = 16$, $\Rightarrow \{CD\}^+ = \{CD\} \Rightarrow 2^2 = 4$

$\Rightarrow \{AD\}^+ = \{ADB\} \Rightarrow 2^3 = 8$, $\Rightarrow \{BD\}^+ = \{BD\} \Rightarrow 2^2 = 4$

$\Rightarrow \{ABC\}^+ = \{ABCD\} \Rightarrow 2^4 = 16$, $\Rightarrow \{BCD\}^+ = \{BCD\} \Rightarrow 2^3 = 8$

$\Rightarrow \{ACD\}^+ = \{ACDB\} \Rightarrow 2^4 = 16$, $\Rightarrow \{ABD\}^+ = \{ABD\} \Rightarrow 2^3 = 8$

$\Rightarrow \{ABCD\}^+ = \{ABCD\} \Rightarrow 2^4 = 16$

$\therefore$ total FD's in $F^+ = 4 + 2 + 4 + 2 + 4 + 8 + 16 + 4 + 8 + 4 + 16 + 8 + 16 + 8 + 16$

$= 120 + 1 = 121$

Date _____ Page No _____

⇒ If a table contains n attributes, atmost how many FD's are possible?

eg R(A, B, C)

$A^+ = \{ABC\} \Rightarrow 2^3 = 8$
$B^+ = \{ABC\} \Rightarrow 2^3 = 8$
$C^+ = \{ABC\} \Rightarrow 2^3 = 8$
$\{AB\}^+ = \{ABC\} \Rightarrow 2^3 = 8$
$\{BC\}^+ = \{ABC\} \Rightarrow 2^3 = 8$
$\{AC\}^+ = \{ABC\} \Rightarrow 2^3 = 8$
$\{ABC\}^+ = \{ABC\} \Rightarrow 2^3 = 8$

$\Rightarrow (2^3 - 1) * 2^3 = 56$

Also, $\emptyset \to \emptyset$, $\emptyset \to A$ --- exists. $\therefore (2^3 - 1 + 1) \times 2^3 = 64$

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

⇒ How many non trivial FD's are possible?

⇒

• $\alpha \to \beta$ is trivial if $\beta \subseteq \alpha$

Date: Page No:

Normalization :

primary key (composite)

⇒

Rollno	Name	Cno	Cname	grade	Bookname
1	n1	c1	DBMS	A	Korth
1	n1	c2	OS	B	Galvin
2	n2	c1	DBMS	B	Korth
3	n1	c3	CN	A	Tanenba

① Insertion Anomaly : If we want to add a new course to the university, we cannot because rollno can't be NULL & initially no student is enrolled in the course. Thus, we are unable to insert the data.

② Deletion Anomaly : If we've to delete rollno 3, it'll delete the details of CN Thus, on deleting some data, extra data is getting deleted.

③ Updation Anomaly : Due to data redundancy, we've to update data many times. (eg Bookname)

④ Data Redundancy : If there are lots of students, we've to insert coursename for all the students. Thus, the unnecessary repeatation of data is known as data redundancy.

* practically, the desired normal form is 3NF.

Date: _____ Page No: ____

⇒ Thus, to remove these problems, we normalize the relation. (decompose one table into many tables)

* there exists various normal forms;

(1) 1NF

* An information representation is said to be in 1NF if it is atomic.

eg

ename	job	phone
e1	programmer	9801276012
	analyst	8435501286
	leader	
e2	salesman	9009012768
	manager	7999720116
		8770401972

It's info. ⇒ not table

* It cannot be directly stored in a table. Thus, it's not atomic & thus, not in 1NF.

⇒ To convert it into 1NF(table), ~~in one row~~, per job store row for each phone no.; 12 rows will be in table.

* Every relation is in 1NF.

Date _____ Page No _____

(2) 2NF :

* A relation is said to be in 2NF if all the non-prime attributes fully functionally depends on a candidate key

OR

* A relation is said to be in 2NF if there exists no partial FD.

* **prime attribute** : An attribute is said to be prime if it is a part of candidate key but not the candidate key itself. & those which are not related to candidate key are called non-prime attributes.

eg R(A, B, C, D)

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

$A \rightarrow B$
$C \rightarrow D$

Here, AC is candidate key.
thus A & C are prime & remaining B & D are non-prime.

eg R(S, C, D, G, V)

$S \rightarrow C$
$SD \rightarrow G$
$CG \rightarrow V$

Here, SD is candidate key.
thus, prime ⇒ S, D
non-prime ⇒ C, G, V

Date: Page No:

eg R (A, B, C, D)

A → B
B → C
C → D

Here, A is candidate key.
Thus, there exists no prime attributes.
non prime ⇒ B, C, D.

* A is neither prime nor non-prime attribute.

* If the candidate key is not composite then there will be no prime attributes.

eg R (A, B, C, D)

A → B
B → C
C → D
D → A

Here, candidate key ⇒ A, B, C, D
Thus, A, B, C & D are neither prime nor non-prime attributes.

* partial FD : An FD is said to be partial if its LHS is prime & RHS is non-prime.

eg R (A, B, C, D)

A → B
C → D

Here, AC is candidate key,
prime ⇒ A, C
non-prime ⇒ B, D

⇒ Both are partial FDs. Thus, the relation is not in 2NF.

• For partial dependency $\alpha \to B$, α should be proper subset of CK, but not CK itself.

Date _____ Page No _____

eg R (S, C, D, G, V)

S → C (partial) Here, Candidate key → SD
CK. SD → G (not partial) prime ⇒ S, D
itself CG → V (not partial) non-prime ⇒ C, G, V

⇒ Not in 2NF

—x—

eg R (A, B, C, D)

A → B (not partial) Here, candidate key ⇒ A
B → C (not partial) Thus, there exists no prime attribu
C → D (not partial) non prime ⇒ B, C, D
* A is neither prime nor not-prime.

⇒ In 2NF

—x—

eg R (A, B, C, D, E, F, G, H)

CH → G (not partial) Here, candidate key ⇒ DA, DB, DE, DF

- A → BC
- B → CFH
- E → A
- F → EH

prime ⇒ D, A, B, E, F
non prime ⇒ C, G, H

↳ A → B (not partial)
A → C (partial) ⇒ Not in 2NF

Date: _____ Page No: _____

Que Part - Supplier Table

Part (Pno, pname, color, sno, sname)

Pno → pname
Pno → color
Pno → sno
sno → sname

Here, Pno is candidate key.
Thus, there exists no prime attribute.
non prime ⇒ pname, color, sno, sname

⇒ In 2NF

* Pno is neither prime nor non prime.

* If the candidate key is not composite then the relation will always be in 2NF.

—x—

* Here, all non-prime attributes fully functionally depends on candidate key.
(sno → sname, pno → sno)
transitively, Pno → sname

* There exists insertion, deletion, updation anomaly & data redundancy; all these problems exist due to transitive FDs.

eg Pno → sno
sno → sname
⇒ pno → sname

③ 3NF :

* A relation is said to be in 3NF if all non-prime attributes non-transitively depends on candidate key.

OR

* A relation is said to be in 3NF if in FD $\alpha \rightarrow \beta$, either α should be super key or β should be prime attribute.

eg R (A, B, C, D)

$A \rightarrow B$	✓	Here, candidate key ⇒ A
$B \rightarrow C$	x	non prime ⇒ B, C, D
$C \rightarrow D$	x	A is neither prime nor non-prime

* Here, AB can also be candidate key but it's not minimal.

* every candidate key is a super key, but not viceversa

* superset of candidate key is also a super key.

* minimal super key is candidate key.

⇒ Not in 3NF.

Date _____ Page No. _____

eg R (A, B, C, D)

A → B ✓
B → C ✓
C → D ✓
D → A ✓

Here, candidate key ⇒ A, B, C, D.
⇒ A, B, C, D are neither prime nor non-prime.

⇒ In 3NF

—x—

Que R (Rollno, Name, Cno, Cname, grade)

Rollno → Name ✓
Name → Rollno ✓
Cno → Cname ✓
Cname → Cno ✓
Rollno Cno → grade ✓

Here, candidate keys ⇒
Rollno Cno, Name Cname,
Rollno Cname, Name Cno.
prime ⇒ Rollno, Name, Cno, Cname
non-prime ⇒ grade

⇒ In 3NF

—x—

* If a relation is in 3NF, all anomalies will be vanished but there can be data redundancy.

* In questions, start checking from BCNF, & then further only for those FD's which are violating BCNF.

Date: _____ Page No: _____

④ BCNF :

* A relation is said to be in BCNF if the determinant (L.H.S. of FD) is super key.

* If $\alpha \to \beta$ is an FD, then α must be super key.

eg R (A, B, C, D)

$A \to B$ ✓
$B \to C$ ✓
$C \to D$ ✓
$D \to A$ ✓

Here, candidate key ⇒ A, B, C, D

⇒ In BCNF

→ —

eg R (A, B, C, D, E) ; What's the ~~min~~ max. normal form this relation is?

$A \to B$
$BC \to D$
$CD \to E$

Here, candidate key ⇒ AC
prime ⇒ A, C
non-prime ⇒ B, D, E

(a) BCNF ✗
(b) 3NF ✗
(c) 2NF ✗
(d) 1NF ✓

— ×

Date: Page No:

CS-2004

Que 50. R (Rollno, Name, Cno, grade)

Rollno Cno → grade
Name Cno → grade
Rollno → Name
Name → Rollno

Here, candidate keys ⇒ Rollno Cno, NameCno

prime ⇒ Cno, Rollno, Name
non-prime ⇒ grade

- BCNF ✗
- 3NF ✓

—x—

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

AB → CD
DE → F
C → E
F → C
B → G

Here, candidate key ⇒ AB

prime ⇒ A, B
non-prime ⇒ C, D, E, F, G

- BCNF ✗
- 3NF ✗
- 2NF ✗
- 1NF ✓

—x—

Date: Page No:

* Till BCNF, normal forms are based on FDs. But, after BCNF, normal forms are not based on FDs but on MVDs.

* If a relation is in BCNF, all the anomalies & data redundancy due to FD will be vanished.

* Normalization is done to decompose tables & denormalization to merge tables.

Date _____ Page No. _____

⇒

empno	ename	deptno	dname	dcity
1	e1	10	Research	Delhi
2	e2	10	Research	Delhi
3	e3	20	Finance	Mumbai
4	e4	20	Finance	Mumbai

* All the anomalies & data redundancy exist here. Thus, we'll normalize the table.

⇒ emp:

empno	ename
1	e1
2	e2
3	e3
4	e4

dept:

deptno	dname	dcity
10	Research	Delhi
20	Finance	Mumbai

⇒ But, which emp belongs to which dept. ✗ data-loss

* Thus, on join, we must be able to create original table.

* Such decompositions where data is lost, are known as lossy join decomposition. & where data is not lost, known as lossless join decomposition.

⇒ Above decomposition is lossy.

Date: Page No:

⇒ emp:

empno	ename	dcity
1	e1	Delhi
2	e2	Delhi
3	e3	Delhi
4	e4	Delhi

dept:

deptno	dname	dcity
10	Research	Delhi
20	Finance	Delhi

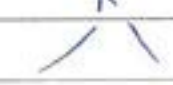
⇒ This decomposition is also lossy. • We'll get total 8 tuples on join.

* In lossy join, we get some extra tuples known as spurious tuples. (cartesian product)

—x—

⇒ Identify Lossy or Lossless Decomposition:

Method (1):



⇒ $R_1 \cap R_2 = \{A\}$

then, A must be primary key of any table for lossless.

eg $R_1 \cap R_2 \Rightarrow$ emp ∩ dept = dcity

↓

it's not primary key of any table, thus, lossy

eg $R_1 \cap R_2 \Rightarrow$ emp ∩ dept = deptno

↓

it's primary key of dept thus, lossless.

Apply method (1) for only 2 rel. decomposition. (91)

Date: _____ Page No: _____

Method (2) :

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

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

$R_1 (A, B, C)$ ⟷ $A \to B,\ B \to C$

$R_2 (B, D, E)$ ⇒ $B \to D$

⇒

	A	B	C	D	E
R1	α	✓	α	α	
R2		α	α	α	α

Step (1) : Search the column with 2 α (B here)
Step (2) : Search the column with 1 α (A, C, D, E here)
Repeat

Now, if $B \to C$ then $R_2(C) = \alpha$

& $B \to D$ then $R_1(D) = \alpha$

* If any row can be converted into α, then the decomposition is lossless else lossy.

⇒ this decomposition is lossy.

eg previous question with method ① ;

$R_1(A,B,C) \cap R_2(B,D,E) = B$

* primary key of R1 ⇒ A
* primary key of R2 ⇒ BE

it's not primary key of R1 or R2. thus, lossy.

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

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

R_1 (A, B, E)

R2 (B, C, D)

① method: $R_1 \cap R_2 = B$ → it's primary key of R2, thus, lossless

* primary key of R1 ⇒ AE
* primary key of R2 ⇒ B

For $AC \to D$, then A & C should have 2α
& D should have 1α.

Date _____ Page No. _____

(2) method :

	A	B	C	D	E
R1	α	α	α	α	α
R2		α	α	α	

- $2\alpha \Rightarrow$ B , $1\alpha \Rightarrow$ A, C, D, E

* $B \to C \Rightarrow R_1(C) = \alpha$
* $B \to D \Rightarrow R_1(D) = \alpha$
*

⇒ This decomposition is lossless.

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

$A \to B$ R_1 (A, B)
$B \to C$ R_2 (B, C, D)
$C \to D$ R_3 (A, E)
$B \to D$

(1) method : $R_1 \cap R_2 = B \Rightarrow$ its primary key of R2

$R_2 \cap R_3 = \emptyset$

$R_1 \cap R_3 = A \Rightarrow$ it's primary key of R1

Thus, it's lossless.

Date _____ Page No. _____

(2) method :

	A	B	C	D	E
R1	∝	∝	∝	∝	
R2		∝	∝	∝	
R3	∝	∝	∝	∝	∝

- $2\propto$ → A, B $1\propto$ → C, D, E

R1 & R2

* $B \to C \Rightarrow R_1(C) = \propto$
* $B \to D \Rightarrow R_1(D) = \propto$

R1 & R3

* $A \to B \Rightarrow R_3(B) = \propto$
* $B \to C \Rightarrow R_3(C) = \propto$
* $B \to D \Rightarrow R_3(D) = \propto$

⇒ This decomposition is lossless.

* A decomposition should be lossless & dependency preserving.

Date _____ Page No: ____

* A decomposition is dependency preserving if

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

* All FD's should exist after normalization.

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

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

$R_1(A, B, E)$

$R_2(B, C, D)$

$$F_1 = \{ A \to B \} \qquad F_2 = \{ B \to C,\ C \to D,\ B \to D \}$$

Here, $F^+ = F_1^+ \cup F_2^+$, thus, the decomposition is dependency preserving.

⇒ Identify decomposition is dependency preserving or not

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

① $A \to B$ ✓R1 $R_1(A, B, C)$
② $B \to C$ ✓R1
③ $C \to D$ $R_2(B, D)$
④ $B \to D$ ✓R2

$\emptyset^+ = \emptyset$

Date: _____ Page No: _____

* Thus, we've to check only for FD $C \to D$ & for this, there's an algorithm.

$\Rightarrow$ $Z = C$ (LHS of FD) $C \to D$

R3

$Z = Z \cup (R_1 \cap (R_1 \cap Z)^+)$

$Z = C \cup (ABC \cap (ABC \cap C)^+)$

$Z = C \cup (ABC \cap C^+)$

$Z = C \cup (ABC \cap CD)$

$Z = C \cup C$ Also, check for R2.

$Z = C$ $\because$ we didn't get RHS of FD, here $\Rightarrow$ this is not dependency preserving.

R2 $\Rightarrow$ $Z = C$

$Z = C \cup (BD \cap \emptyset^+)$

$Z = C$

Date: _____ Page No: ____

eg R(A, B, C, D)

A → B ✓R1 R1 (A, B, C)
B → C ✓R1
A → C ✓R1 R2 (C, D)
B → D .

R1 ⇒ Z = B

$Z = Z \cup (R1 \cap (R1 \cap Z)^+)$

$Z = B \cup (ABC \cap BCD)$

Z = BC

R2 ⇒ Z = B

$Z = B \cup (CD \cap (CD \cap B)^+)$

Z = B

R1 ⇒ Z = BC

$Z = BC \cup (ABC \cap (ABC \cap BC)^+)$

$Z = BC \cup BC$

Z = BC

* Here, value of Z is updated, thus check again for its new value.

R2 ⇒ Z = BC

$Z = BC \cup (CD \cap (CD \cap BC)^+)$

$Z = BC \cup C$

Z = BC ⇒ not dependency preserving

—x—

Date _____ Page No. _____

* Repeat this until Z achieves a constant value or when we get RHS of FD in Z.

Que R (A, B, C, D)

$A \to B$

$A \to C$ ✓R1 — R1 (A, C)

$B \to C$ ✓R2

$B \to D$ ✓R2 — R2 (B, C, D)

R1 ⇒ $Z = A$

$Z = A \cup (AC \cap (AC \cap A)^+)$

$Z = A \cup AC$

$Z = AC$

R1 ⇒ $Z = AC$

$Z = AC \cup (AC \cap (AC \cap AC)^+)$

$Z = AC \cup AC$

$Z = AC$

R2 ⇒ $Z = A$

$Z = A \cup (BCD \cap (BCD \cap A)^+)$

$Z = A$

p2 ⇒ $Z = AC \cup (BCD \cap (BCD \cap AC)^+)$

$Z = AC \cup C$

$Z = AC$

⇒ not dependency preserving

Date _____ Page No. _____

IT-2008

Que $R(A, B, C, D)$

$A \to B$ ✓R1
$B \to C$ ✓R2
$C \to D$.
$D \to B$ ✓R3

R1 (A, B)
R2 (B, C)
R3 (B, D)

R1 ⇒ $Z = BC$
$Z = BC \cup (AB \cap (AB \cap S)^+)$
$Z = BC$

R2 ⇒ $Z = C$
$Z = C \cup (BC \cap (BC \cap C)^+)$
$Z = C \cup BC$
$Z = BC$

R3 ⇒ $Z = C$
$Z = C \cup (BD \cap (BD \cap C)^+)$
$Z = C$

R3 ⇒ $Z = BC$
$Z = BC \cup (BD \cap (BD \cap BC)^+)$
$Z = BC \cup BD$
$Z = BCD$

⇒ dependency preserving.

Date: Page No:

⇒ How to find Minimal Cover / Canonical Cover of FDs set:

① Convert all FDs to simple.

* An FD is said to be simple if its RHS contains a single attribute.

eg $A \to BC \implies A \to B$ & $A \to C$

② Remove extra attributes from LHS of FDs.

eg $AB \to C \implies B \to C$ (if A is extra)

Here, A will be an extra attribute if the closure of B contains A, & vice versa.

③ Remove extra attributes from RHS of FDs
OR Remove redundant FDs.

* An FD is said to be redundant if it can be derived from other FDs.

eg $A \to B$
$B \to C$
$A \to C \implies$ redundant

eg Remove redundant FDs.

① $A \rightarrow B$
② $B \rightarrow C$
③ $A \rightarrow C$

* Find the closure of ~~LHS of~~ all FDs without considering itself.

① ⇒ $A^+ = \{AC\}$ ⇒ RHS (B) is not present thus, not redundant

② ⇒ $B^+ = \{B\}$ ⇒ RHS (C) is not present thus, not redundant

③ ⇒ $A^+ = \{ABC\}$ ⇒ RHS (C) is present. thus, redundant FD.

** Once you say that an FD is redundant, then don't consider that redundant FD while computing redundancy of other FDs.

Date _____ Page No _____

Que Find minimal cover:

① $A \rightarrow B$
② $A \rightarrow C$
③ $A \rightarrow D$
④ $A \rightarrow H$
⑤ $DH \rightarrow A$
⑥ $DH \rightarrow B$
⑦ $DH \rightarrow C$
⑧ $DH \rightarrow G$

Step ①: All FDs are simple

Step ②: There's no extra attribute on LHS of FDs.

Step ③: ① ⇒ $A^+ = \{ABCDHG\} - \{BG\}$ ⇒ redundant

② ⇒ $A^+ = \{ABDHCG\} - \{ADHBCG\}$ ⇒ redundant

③ ⇒ $A^+ = \{AH\}$ ⇒ not redundant

④ ⇒ $A^+ = \{AD\}$ ⇒ not redundant

⑤ ⇒ $(DH)^+ = \{DHBCG\}$ ⇒ not redundant

⑥ ⇒ $(DH)^+ = \{DHACG\}$ ⇒ not redundant

Date _____ Page No. _____

⑦ ⇒ $(DH)^+ = \{DHABG\}$ ⇒ not redundant

⑧ ⇒ $(DH)^+ = \{DHABC\}$ ⇒ not redundant

⇒ previous question from bottom:

⑧ ⇒ $\{DH\}^+ = \{DHABC\}$ ⇒ not redundant

⑦ ⇒ $\{DH\}^+ = \{DHABGC\}$ ⇒ redundant

⑥ ⇒ $\{DH\}^+ = \{DHAGBC\}$ ⇒ redundant

⑤ ⇒ $\{DH\}^+ = \{DHG\}$ ⇒ not redundant

④ ⇒ $A^+ = \{ABCD\}$ ⇒ not redundant

③ ⇒ $A^+ = \{ABCH\}$ ⇒ not redundant

② ⇒ $A^+ = \{ABDHG\}$ ⇒ not redundant

① ⇒ $A^+ = \{ACDHG\}$ ⇒ not redundant

* Thus, there can be multiple minimal covers.

Date _____ Page No _____

⇒ Now, these 2 minimal covers are equivalent or not?

If $F_1 \subset F_2$ & $F_2 \subset F_1$

then $F_1 \equiv F_2$

F_1	F_2
• $A \to D$ ✓	$A \to B$ ✓
• $A \to H$ ✓	$A \to C$ ✓
• $DH \to A$ ✓	$A \to D$ ✓
• $DH \to B$ ✓	$A \to H$ ✓
• $DH \to C$ ✓	$DH \to A$ ✓
• $DH \to G_1$ ✓	$DH \to G_1$ ✓

Thus, $F_1 \equiv F_2$.

---

* $F_1 \subset F_2$ ⇒ means, all the FDs of F_1 can be derived from FDs of F2.

---

* $DH \to B$ is in F1 but not in F2,

⇒ In F2, $\{DH\}^+ = \{DHAG_1BCD\}$

thus, it can be derived in F2

* Do this for all FDs.

Date _____ Page No. _____

Que Find minimal cover of following FD set;

① $CH \rightarrow G$
② $A \rightarrow BC$
③ $B \rightarrow CFH$
④ $E \rightarrow A$
⑤ $F \rightarrow EG$
⑥ $A \rightarrow CG$

Step ①: $CH \rightarrow G$
$A \rightarrow B$, $A \rightarrow C$
$B \rightarrow C$, $B \rightarrow F$, $B \rightarrow H$
$E \rightarrow A$
$F \rightarrow E$, $F \rightarrow G$
$A \rightarrow C$, $A \rightarrow G$

Step ②: there's no extra attribute on LHS of FDs

Step ③: ① $CH \rightarrow G$
② $A \rightarrow B$
③ $A \rightarrow C$
④ $B \rightarrow C$
⑤ $B \rightarrow F$
⑥ $B \rightarrow H$
⑦ $E \rightarrow A$
⑧ $F \rightarrow E$
⑨ $F \rightarrow G$
⑩ $A \rightarrow G$

Date _____ Page No _____

① ⇒ $\{CH\}^+ = \{CHG\}$ ⇒ not redundant

② ⇒ $A^+ = \{ACG\}$ ⇒ not redundant

③ ⇒ $A^+ = \{ABGCFHE\}$ ⇒ redundant

④ ⇒ $B^+ = \{BFHEGA\}$ ⇒ not redundant

⑤ ⇒ $B^+ = \{BCHG\}$ ⇒ not redundant

⑥ ⇒ $B^+ = \{BCFEGA\}$ ⇒ not redundant

⑦ ⇒ $E^+ = \{E\}$ ⇒ not redundant

⑧ ⇒ $F^+ = \{FG\}$ ⇒ not redundant

⑨ ⇒ $F^+ = \{FEAGBCH\}$ ⇒ redundant

⑩ ⇒ $A^+ = \{ABCFHEG\}$ ⇒ redundant

Que

F1	F2
$AB \rightarrow C$ ✓	$AB \rightarrow C$ ✓
$D \rightarrow E$ ✓	$D \rightarrow E$ ✓
$E \rightarrow C$ ✓	$AB \rightarrow E$
	$E \rightarrow C$ ✓

* $F_1 \subset F_2$ but $F_2 \not\subset F_1$

thus, $F_1 \not\equiv F_2$

Date _____ Page No. _____

⇒ How to convert a relation into 3NF :

Step: ① Find minimal cover of FD set.

Step ② Find all candidate keys of relation.

Step ③ Create the relations with the attributes of FD having same LHS.

eg $A \to B$ $R(A, B, C, D, E, G)$
$A \to D$
$C \to D$
$C \to E$
$D \to G$

⇒ Step ① : The FDs set is already minimal.

Step ② : candidate key ⇒ AC

Step ③ : $A \to B$, $A \to D$ | $C \to D$, $C \to E$ | $D \to G$

$R1 (A, B, D)$ $R2 (C, D, E)$ $R3 (D, G)$

* It's a lossy decomposition. thus, there's one more step to make it lossless.

→ this step is performed only when we've composite

Step (4) If candidate key is not a part of any relation then create a relation of that candidate key.

Thus, R1 (A, B, D)
R2 (C, D, E)
R3 (D, G)
R4 (A, C)

Now, it's a lossless decomposition.

* If there are more than 1 CK then create relation for each CK.

* 3NF decomposition is always lossless join & dependency preserving.

* BCNF is always lossless but may / may not be dependency preserving.

Date: Page No:

⇒ How to convert a relation into BCNF :

step ① Find the minimal cover of FD set.

step ② Identify the FDs which violates BCNF

step ③ Create a relation with attributes of FD which violates BCNF. (one by one)

eg R (A, B, C, D)

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

step ① FD set is already minimal.

step ② candidate key ⇒ A

$B \to C$ } violates BCNF
$C \to D$ }

eg
$B \to C$ }
$C \to D$ } violates BCNF
$C \to E$ }
then create a rel. for (C, D, E) not for (C, D) & (C, E)

step ③ R (A, B, ∅, ∅)

* taking $C \to D$ first :

R1 (~~A~~, ~~B~~) (C, D) * cut their RHS from original relation.

R2 (B, C) ⇒ create this rel. only if B & C are in original rel.

I ⇒ R1 (A, B) , R2 (C, D) , R3 (B, C)

* taking B → C first

R (A, ~~B~~, ~~C~~, D)

R1 (B, C)

* Now, we will not make a relation for C → D as C is not an attribute of original relation anymore.

R contain (A, B, D) & ...

But, now, candidate key of R is AD, so, we'll check again if some FDs are violating BCNF.

A → B
B → C } violates BCNF
C → D

⇒ R_2 (A, B)   & cut B from R

(II) * R_1 (A, D)   R2 (B, C)   R3 (A, B)

⇒ Both decompositions are valid & lossless but II is not dependency preserving.

* Here, R_1 (A, D) → no FD for R1 → is it in BCNF?
R_2 (B, C) → B → C ✓BCNF
R_3 (A, B) → A → B ✓BCNF

⇓ yes, if all attr make CK then it's in BCNF
OR
Also, it has 2 attr

Thus, individual rel. are in BCNF.

Date: Page No:

* Any relation with only 2 attributes will always be in BCNF.

eg R (A, B)

* FDs can be;

- $\underset{\text{super key}}{A} \rightarrow B$ OR - $\underset{\text{super}}{B} \rightarrow A$ OR - $\underset{\text{super}}{A} \rightarrow B$, $\underset{\text{super}}{B} \rightarrow A$

Here, $Ck \Rightarrow A$ — Thus, it's in BCNF

Here, $Ck \Rightarrow B$ — Thus, it's in BCNF

Here, $Ck \Rightarrow A, B$ — Thus, it's in BCNF

—x—

* A relation where no FD exists then in that case, all the attributes together form candidate key & that relation is always in BCNF.

—x—

* If a decomposition is lossless & individual relations are in BCNF then that decomposition is a valid BCNF decomposition (although not returned by our algorithm).

—x—

* In 3NF, an attribute may depend transitively on candidate key.

In 3NF, all non prime attributes non-transitively depend on candidate key but the prime attributes may or may not depend transitively on candidate key.

$\underset{\text{super key}}{\alpha} \rightarrow \underset{\text{prime attribute}}{\beta}$

Date: Page No:

* In BCNF, an attribute cannot depend transitively on candidate key.

$$\underset{\text{super key}}{\alpha} \rightarrow \beta$$

—x—

* If a relation is in 3NF & there exists only one candidate key, then the relation is also in BCNF.

⇒ If there's no composite candidate key, there will be no prime attributes, then relation will also be in BCNF.

eg R(A, B, C, D)

AC → D	~~BCNF~~	CK = AC, BC
BC → D	~~BCNF~~	prime ⇒ A, B, C
A → B	~~3NF~~	non-prime ⇒ D
B → A	~~3NF~~	Thus, it's in 3NF, but not in BC

* In $\alpha \rightarrow \beta$, if α is not super key & B is prime then α will also be prime attribute but from another candidate key. This type of FDs are possible only when we've more than 1 composite candidate key.

* Thus, if there's only 1 key, then in 3NF, all the FD's $\alpha \rightarrow \beta$, α will be super key, thus, it will always be in BCNF.

Date _____ Page No _____

4NF:

* It is based on Multivalued Dependency (MVD).
* For a given α, you can have one or more β.

$$\alpha \twoheadrightarrow \beta$$

* every FD is an MVD, but not vice-versa.
* MVD that is not FD is true or pure MVD.

⇒ Course:

Cname	Instructor	Author
DBMS	I1	Korth
DBMS	I2	Navathe
OS	I1	Tanenbaum
NW	I2	Tanenbaum
OS	I1	Galvin
NW	I3	Tanenbaum
NW	I1	Tanenbaum

* Here, all the attributes combinely form the candidate key. Thus, it's in BCNF.

⇒

Cname	Instructor	Author
DBMS	I1	Korth
DBMS	I2	Navathe
DBMS	I1	Navathe
DBMS	I2	Korth

* there exists some MVDs.;

① Cname $\twoheadrightarrow$ Author
② Cname $\twoheadrightarrow$ Instructor

* If $\alpha \twoheadrightarrow \beta$ is an MVD then $\alpha \twoheadrightarrow R - (\beta \cup \alpha)$ is also an MVD.

eg $R(A, B, C, D)$

if $A \twoheadrightarrow B$ holds
then $A \twoheadrightarrow ABCD - BA$
$A \twoheadrightarrow CD$ will also hold.

* In FDs, $A \to B$ = $A \to BC$
$A \to C$

In MVDs, $A \twoheadrightarrow B$ ≠ $A \twoheadrightarrow BC$
$A \twoheadrightarrow C$

Cname →→ Author
Cname →→ Instructor

* It means that, there's no rel. b/w author & instructor.

Date: Page No:

eg Cname →→ Author Instructor

* It means that, there's relationship b/w author & instructor.

⇒

A	B	C
a1	b1	c1
a1	b2	c2

- A →→ BC ✓
- A →→ B ×
- A →→ C ×

⇒

A	B	C
a1	b1	c1
a1	b2	c2
a1	b1	c2
a1	b2	c1

- A →→ BC ×
- A →→ B ✓
- A →→ C ✓

⇒

A	B	C
a1	b1	c1
a1	b2	c2
a1	b3	c1

- A →→ BC ✓
- A →→ B ×
- A →→ C ×

⇒

A	B	C
a1	b1	c1
a1	b2	c2
a1	b3	c1
a1	b1	c2
a1	b2	c1
a1	b3	c2

- A →→ BC ×
- A →→ B ✓
- A →→ C ✓

* MVD is also known as tuple generating dependency.

Date _____ Page No. _____

* A relation is said to be in 4NF if $\alpha \twoheadrightarrow \beta$ is not a trivial MVD then α must be super key.

or

* there should not be any true MVDs, only FDs are allowed.

$\alpha \twoheadrightarrow \beta$ * if α is super key, β will have exactly one value. i.e. it will be an FD.

* An MVD is said to be trivial, if in MVD $\alpha \twoheadrightarrow \beta$

$\beta \subseteq \alpha$ or $\alpha \cup \beta = R$

eg

Cname	Author
DBMS	Korth
DBMS	Navathe

$\Rightarrow$ Cname $\twoheadrightarrow$ Author is a trivial MVD.

- Cname $\cup$ Author = R (Cname, Author)

* Trivial MVDs always hold.

HASHING & INDEXING

Date: Page No.

* Consider a harddisk whose block size is 512 bytes & one record takes 100 bytes. in HD storage.

1	ABC	100 bytes
2	DEF	100 bytes
3	GHI	100 bytes
4	JKL	100 bytes
5	MNO	100 bytes
		12 bytes

* The file organisation which leads to wastage of storage is unspanned organisation.

* The file organisation which doesn't lead to wastage of storage is spanned organisation.

⇒ In unspanned organisation, the next record is stored in another block & those 12 bytes will be wasted.

⇒ In spanned organisation, a part of next record is stored in same block & the remaining 88 bytes in another block.

* Unspanned file organisation is generally used to provide faster access i.e. data will be accessed from one block.

* For OS, database is one file. The files i.e. tables in database are stored by database itself, it's hidden from OS.

Que Consider block size is 512 bytes, record size is 100 bytes. How many blocks are required to store the file if the file contains 4000 records & if file organisation is; ① spanned ② unspanned

⇒ ① spanned :

$$\text{no. of blocks} = \frac{\text{file size}}{\text{block size}} = \frac{4000 \times 100}{512}$$

$$= 781.25 \approx 782 \quad \text{Ans.}$$

⇒ ② unspanned :

$$\text{no. of records in a block} = \frac{512}{100} = 5.12 = 5$$

$$\text{no. of blocks} = \frac{4000}{5} = 800 \quad \text{Ans}$$

Date _____ Page No. _____

⇒ select ename from emp where empno = 100;

* If linear search is used; and there are 800 blocks & 1 us is required to read a block, then,

* in worst case (if record is stored in last block)

⇒ 800 * 1 us = 800 us is required.

⇒ Thus, linear search is not feasible in the database.

* Binary search is also not feasible because there are frequent insertions, updations & deletions in the database. (sorting is not possible).

⇒ Thus, to speedup searching within the database, indexing & hashing are introduced.

—x—

Que There are records of 200 students, block size is 512 bytes, record size is 100 bytes & file organisation is unspanned. If linear search is performed, how many blocks of harddisk database will read in worst case?

⇒ $\frac{200}{5}$ = 40 blocks Ans

—x—

Date _____ Page No. _____

Indexing :

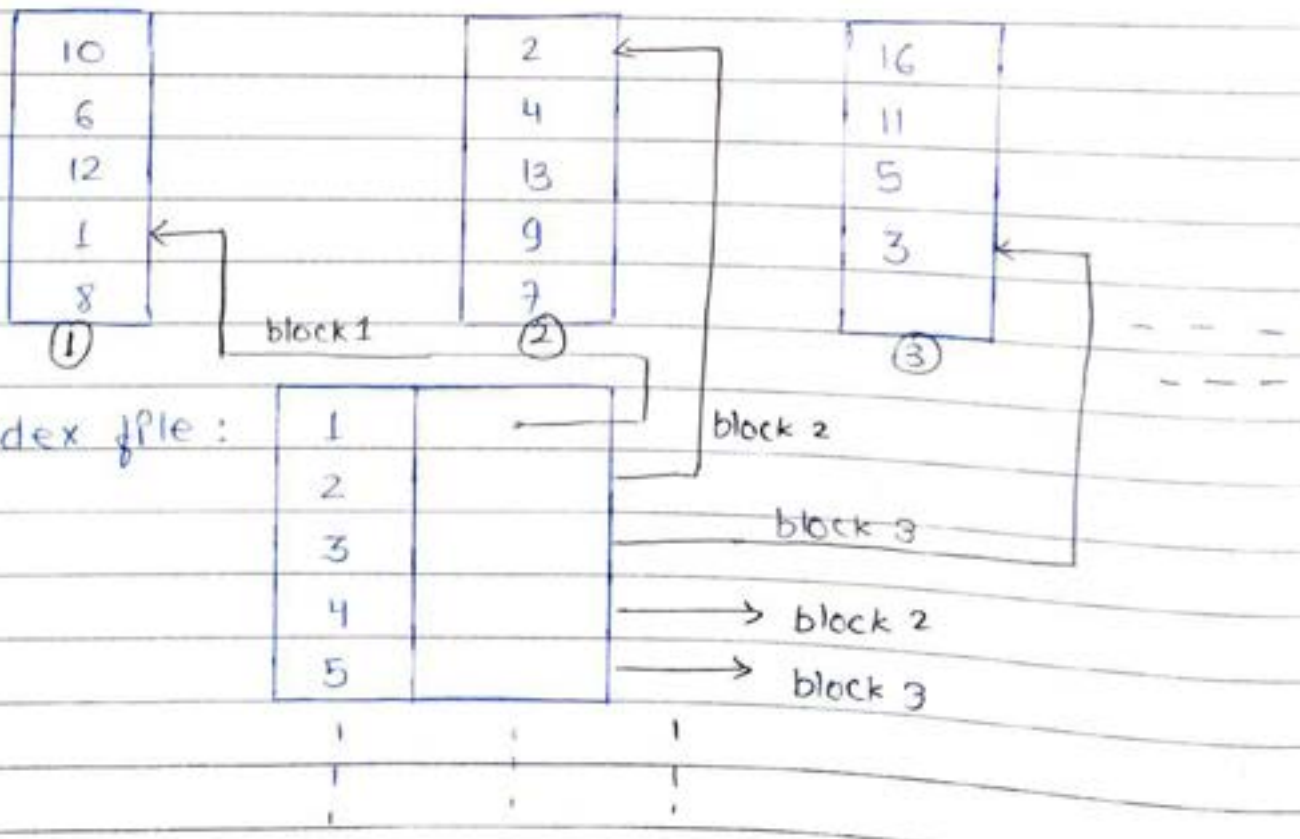
* In indexing, we create a separate file known as index file.

* index file has two columns ;

① key field
primary key

② record pointer
address of record in HD

eg



* index file is sorted on key field.

* take unspanned if not given in que.

Date _____ Page No. _____

* no. of records in index file = no. of records in database file

Que Consider a database, record size is 100 bytes, key field is 4 bytes long, record pointer is 4 bytes, unspanned file organisation. How many blocks are required to store database & index file, if block size is 512 bytes & there are 200 records in the database.

⇒ $\frac{200}{5} = 40$ blocks for database file.

⇒ $200 * (4+4) = 1600$ bytes

⇒ $\frac{1600}{512} = 3.125 \approx 4$ blocks for index file

OR

⇒ no. of records in one block (it is called blocking factor) $= \frac{\text{block size}}{\text{record size}} = \frac{512}{8} = 64$

no. of blocks $= \frac{200}{64} = 3.125 \approx 4$ blocks for index file

* In previous question,

⇒ Indexing : Best case : 2 blocks have to be rea
Worst case : 5 blocks have to be read

⇒ If indexing was not done;

Best case : 1 block has to be read
Worst case : 40 blocks have to be read

—x—

* If a ~~block~~ record is not present in harddisk i.e. Unsuccessful Search:

⇒ Indexing : Best case : 1 block has to be read.
Worst case : 4 blocks have to be read.

⇒ If indexing was not done;

Best case : 40 blocks have to be read
Worst case : 40 blocks have to be read.

—x—

* If the index file is sorted, & linear (sequentia search is used then the file org is known as indexed sequential file organisation.

Date: _____ Page No: _____

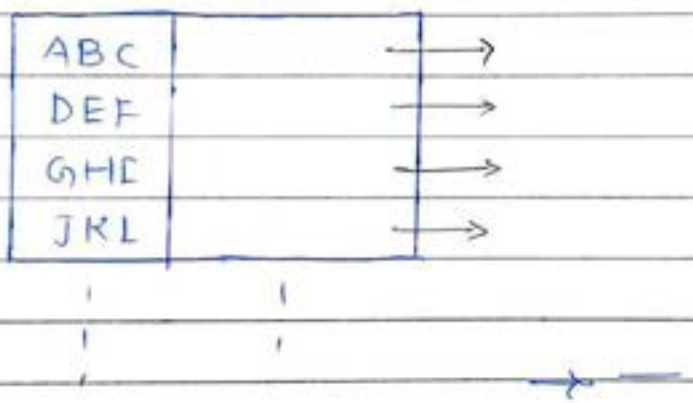
- Drawbacks of Indexing:

(1)

⇒ select rollno from Student where name = "ABC";

⇒ select name from Student where rollno = 1;

* Thus, we have to make index files for all attributes.



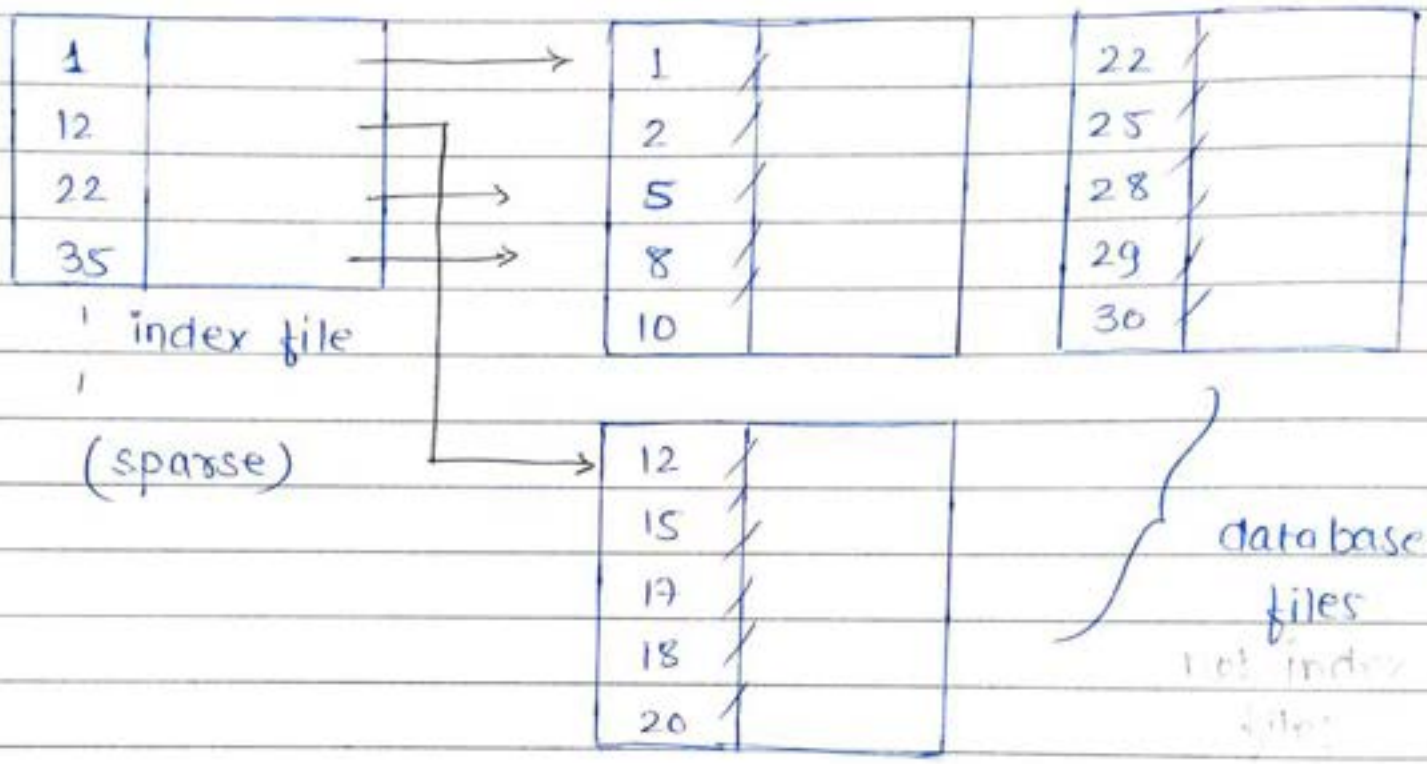
(2) If the database is very large then its index file will also contain a lot of records & will be stored in large no. of blocks & in worst case, all these blocks have to be read.

⇒ Thus, how to speedup searching in an index file

PTO

Date: _____ Page No: _____

① Minimize Index file :



* But for this, database must be sorted &

* The index file in which there's an entry for every value (rollno) is known as dense index file.

* The index file in which there's an entry for some value (rollno) is known as sparse index file.

* database must be sorted for sparse index.

Date: _____ Page No: _____

* In dense index;

no of records in index = no. of records in database

* In sparse index;

no. of records in index = no. of blocks in database is stored

* no. of dense index file = no. of attributes & their combinations

* no. of sparse index file = atmost one; as database can be sorted on any one column.

If database is not sorted, index will be secondary or non-clustered index. (always dense)

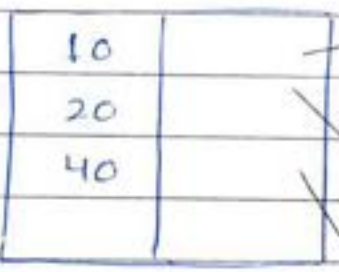
Date: _____ Page No: _____

* Primary Index : The index on a unique field on which the database is sorted is primary index (eg rollno)

* It should be a sparse index as the records are sorted.

* Cluster Index :

* If the database is sorted on a non-unique field (eg. deptno)



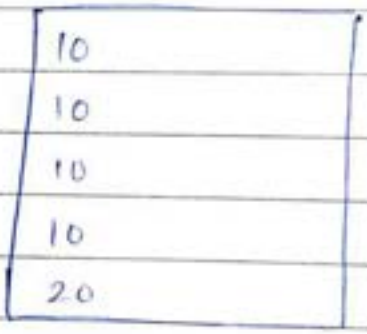
sparse

cluster index

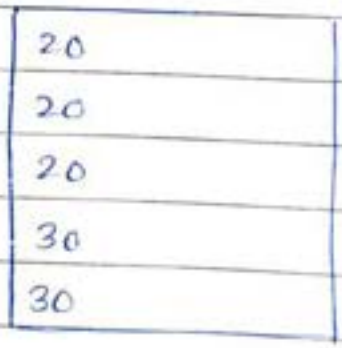
korth (dense)

Navathe (sparse)

30 is not present in Index.



block 1



block 2

block 3

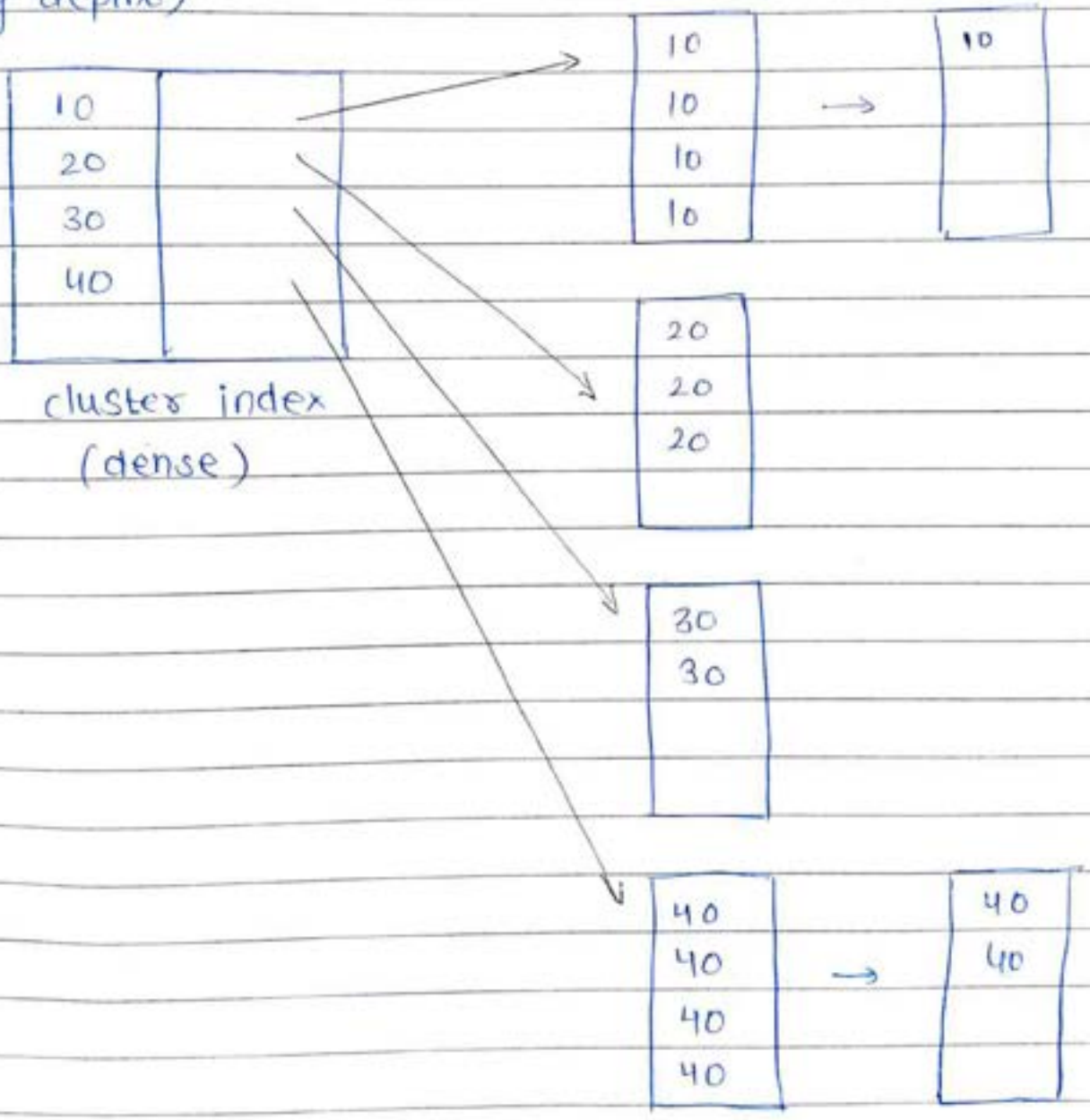
Date: _____ Page No: _____

problem ⇒ select * where deptno = 20

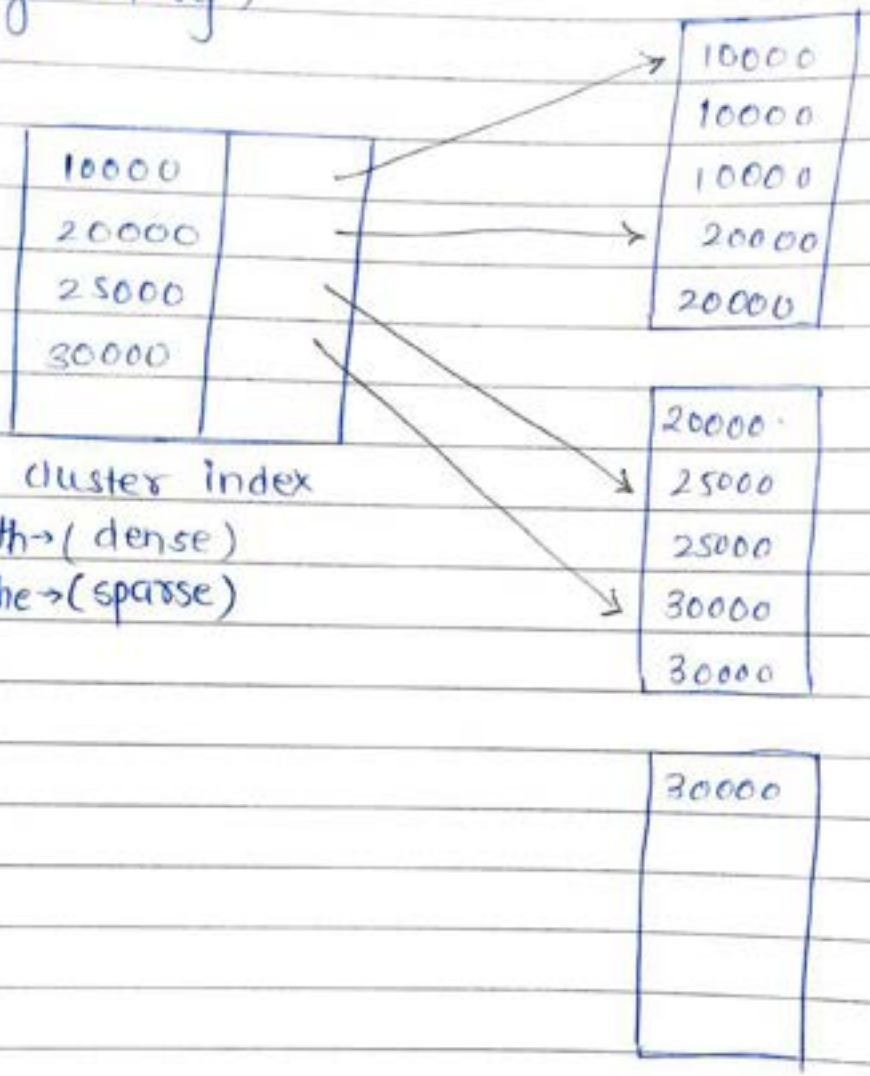
* only 3 records will be printed, block 1 is not read.

Solution:

① If there are more ~~repeation~~ repeatations in column (eg deptno)



② if there are less repeatations in column (eg salary)



✓ Kooth → (dense)
Navathe → (sparse)

*

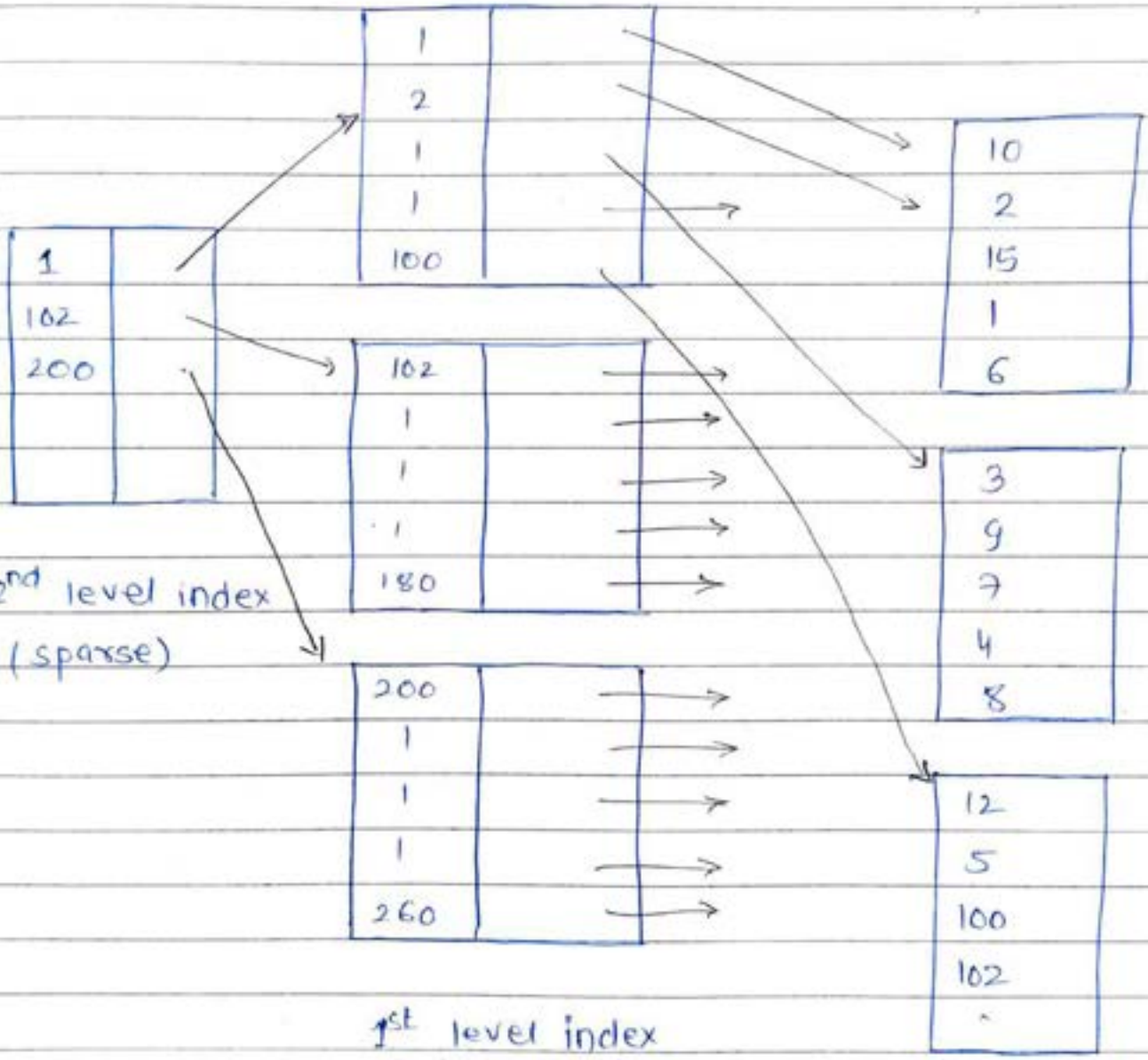
* There can be only 1 sparse index, the remaining index(s) have to be dense. Thus, how to speedup searching in dense index;

① Multilevel Index ② Tree Index

Date: _____ Page No: _____

Multilevel Index :

* First, we create dense index of database file & then we'll create index to index file.



* 2nd level index will be sparse as 1st level index is sorted.

⇒ no. of records in 2nd level index = no. of blocks in 1st level index

⇒ no. of records in 1st level index = no. of records in database

Que Consider a database file consisting of 4000 records & record size is 100 bytes, 2-level indexing is used where 1st level is dense index on rollno & size of rollno is 4 bytes, record pointer is also 4 bytes long. How many blocks are required to store 1st level & 2nd level index if block size is 512 bytes.

⇒ no. of records in a block = $\frac{512}{100} = 5$

no. of blocks req. to store database = $\frac{4000}{5} = 800$

* index record size = 4 + 4 = 8 bytes

no. of index records in one block = $\frac{512}{8} = 64$

Date _____ Page No: _____

⇒ no of blocks req. to store index of 1st level = $\frac{4000}{64} = 62.5 \approx 63$

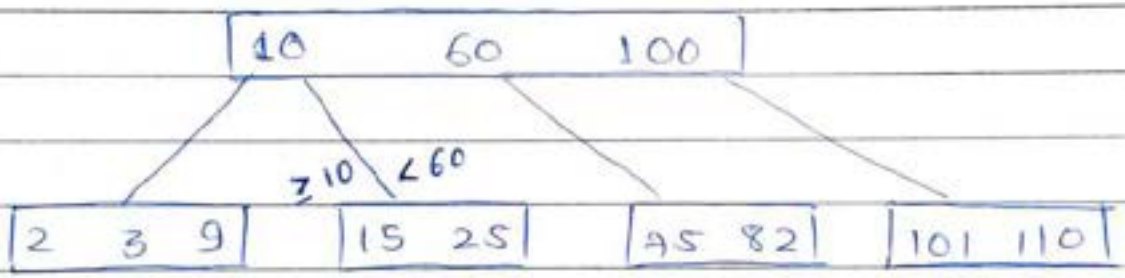
x 1 block for 2nd level index.

Tree Index :

B - trees : They are multiway search tree or n-way search tree in which per node there can be more than 1 key.

* within the nodes, keys are sorted.

eg



* Before creating tree, order of tree is fixed ie. how many children a node can have at maximum.

* if order = n, there will be atmost n children & there is n-1 keys

Date: _____ Page No: _____

* B-tree is a multiway search tree in which ;

(1) all the leaf nodes should be at same level.

(2) Every non-root node should have atleast $\lceil \frac{n}{2} \rceil$ children, where n = order &

min. keys will be $\lceil \frac{n}{2} \rceil - 1$.

eg B-tree of order = 3
max. children = 3 , min children = 2
max. keys = 3-1 = 2 , min keys = $\lceil \frac{3}{2} \rceil - 1 = 1$

eg B-tree of order = 10
max. children = 10 , min children = 5
max. keys = 10-1 = 9 , min keys = $\lceil \frac{10}{2} \rceil - 1 = 4$

Date: _____ Page No: _____

Que B-tree of order 3 (max. key = 2, min key = 1)

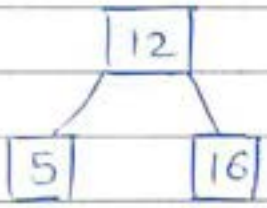
⇒ 16, 5, 12, 9, 3, 10, 2, 14, 20, 6

* Always perform insertion in leaf nodes.

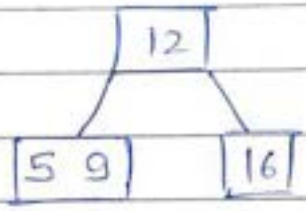
① [16]

② [5 16]

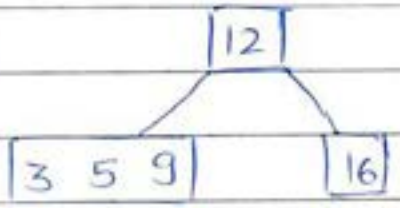
③ [5 12 16] ← split



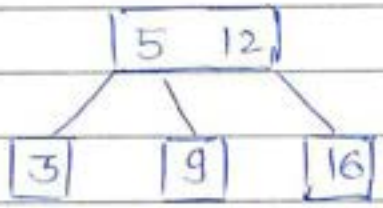
④



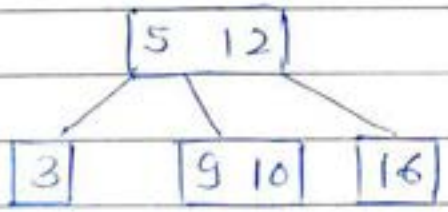
⑤



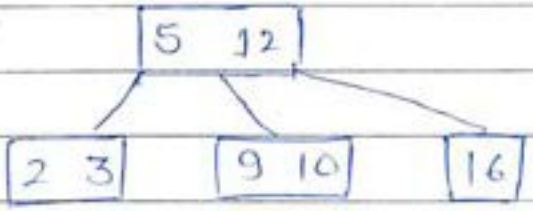
split ⟹



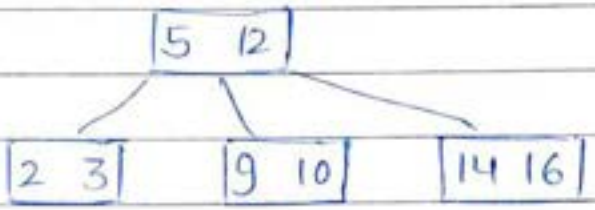
⑥



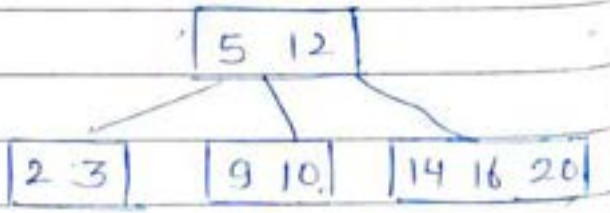
⑦



⑧

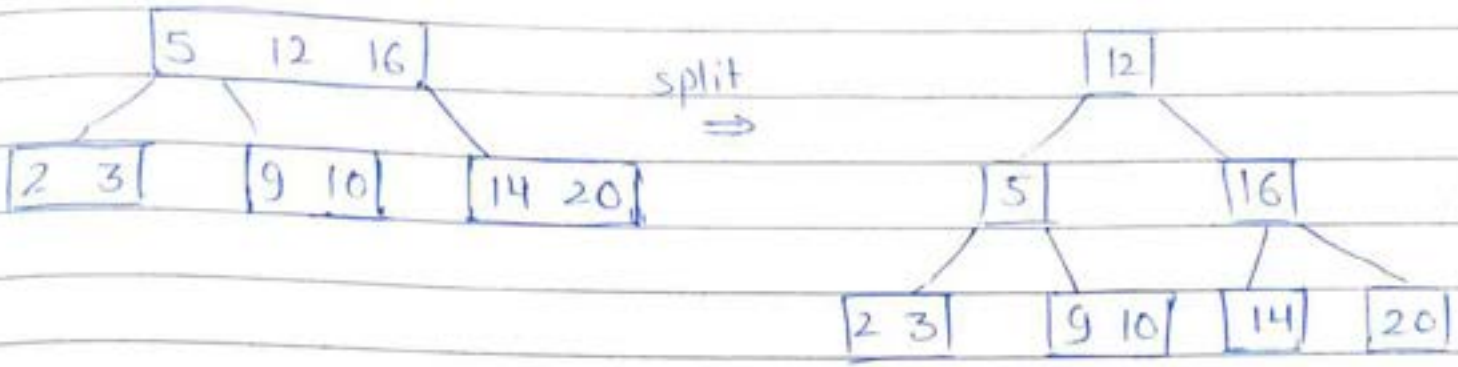


⑨

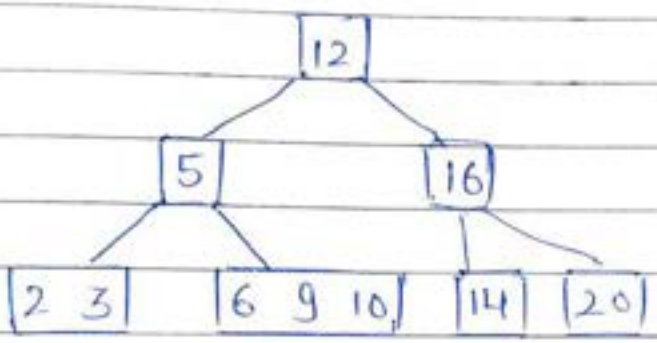


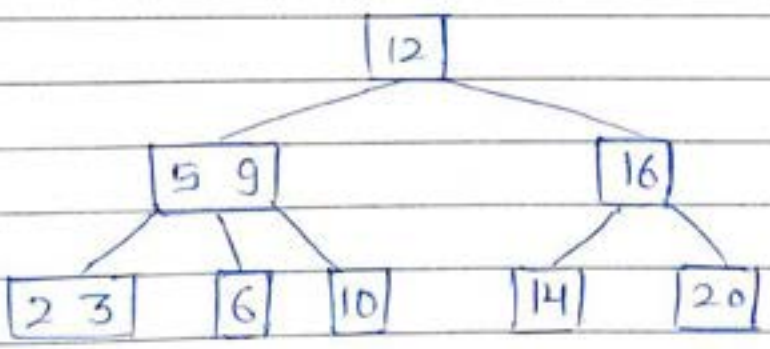
split ⟹

Date _____ Page No. _____



(10)





→—

* New node is created only when root node is splitted.

Date: _____ Page No: _____

Que B-tree of order 4

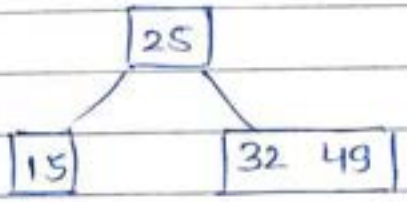
⇒ 25, 32, 15, 49, 36, 38, 42, 43, 45, 39, 37

(1) [25]

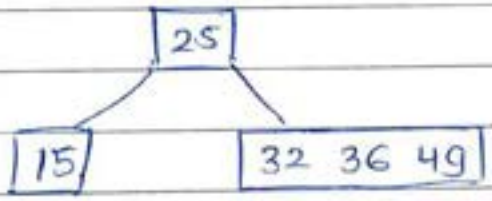
(2) [25 32]

(3) [15 25 32]

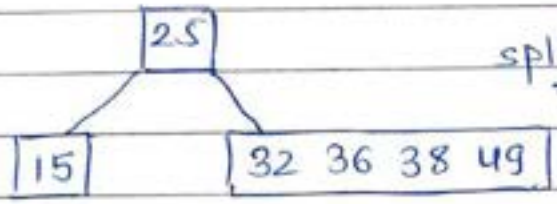
(4) [15 25 32 49] split ⇒ (either left or right)



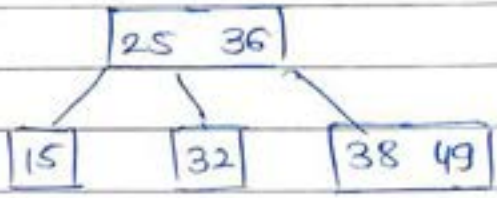
(5)



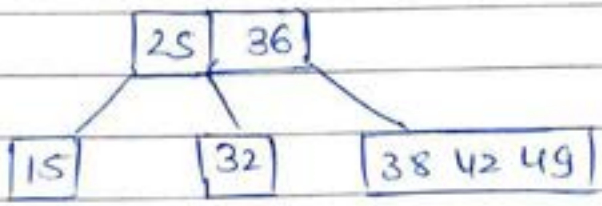
(6)



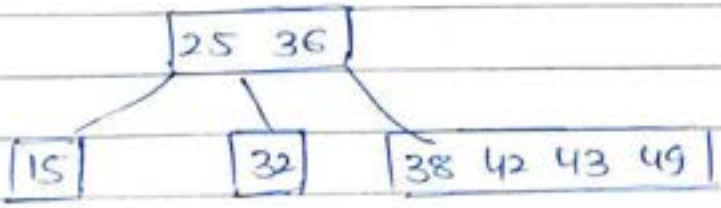
split ⇒



(7)

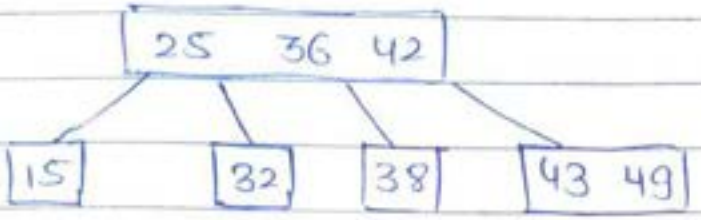


(8)

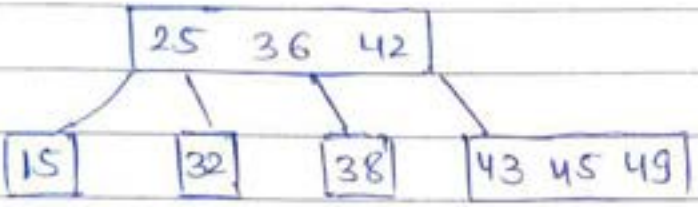


split

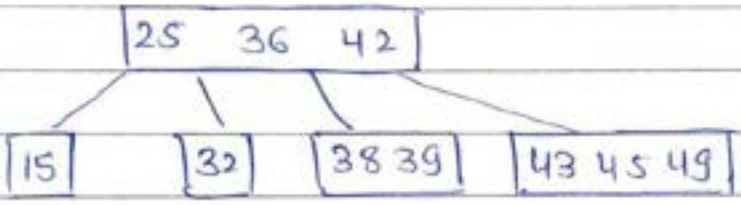
Date: _____ Page No: ____



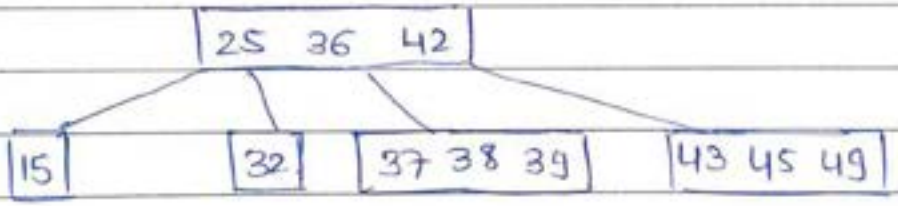
(9)



(10)



(11)



→ —

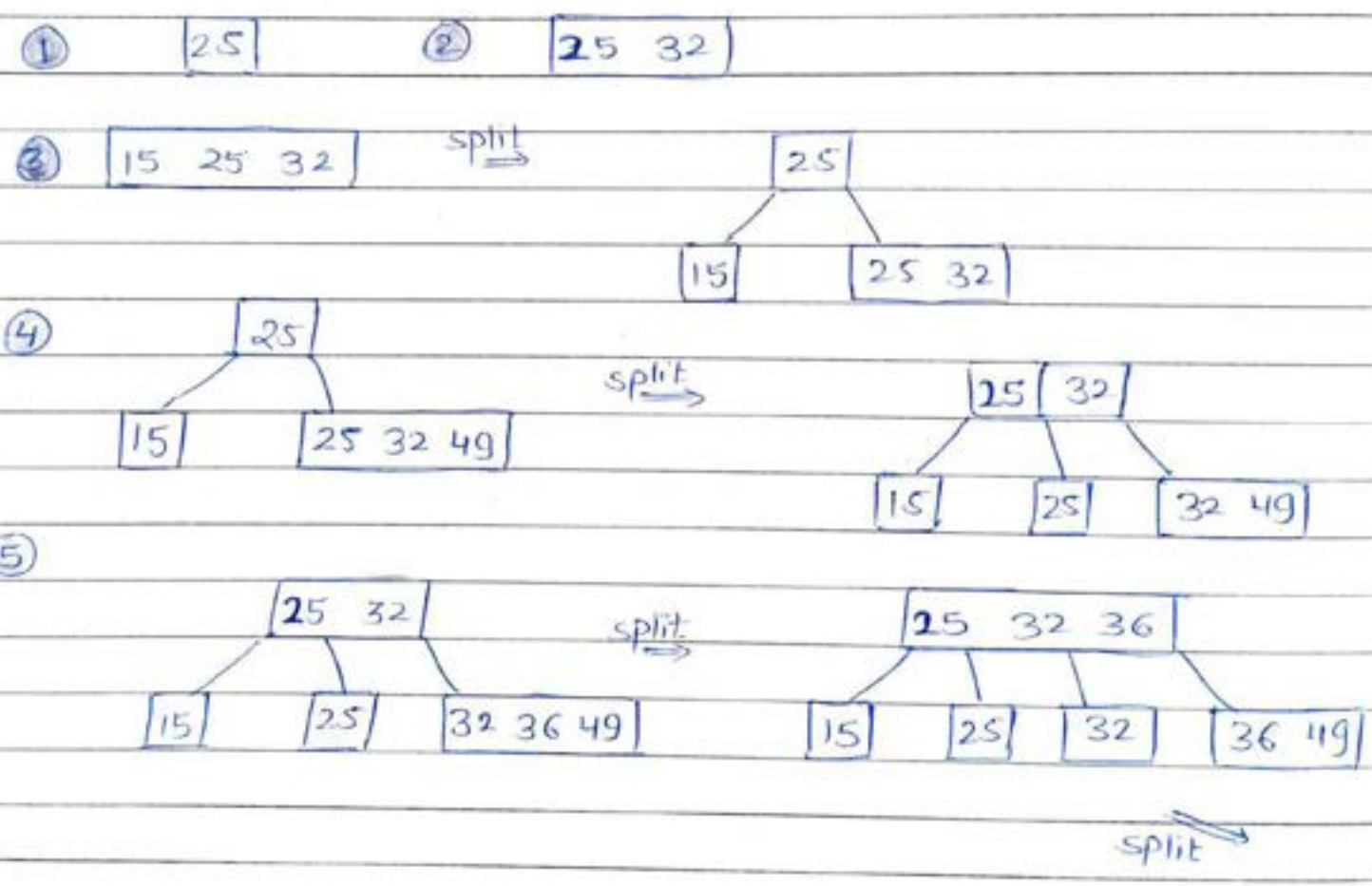
Date _____ Page No. _____

B^+ - Tree : They are B trees in which every key must present at leaf node.

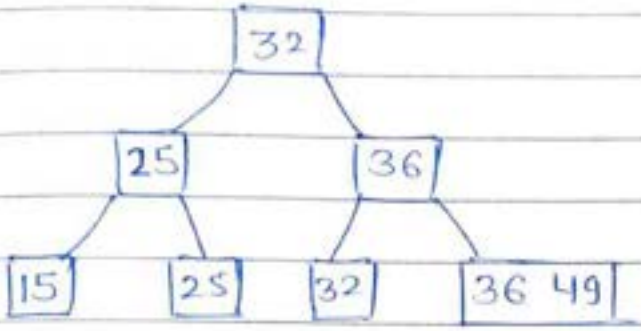
* Perform replication only when leaf is splitted, not for non-leaf node.

Que B^+ - tree of order = 3.

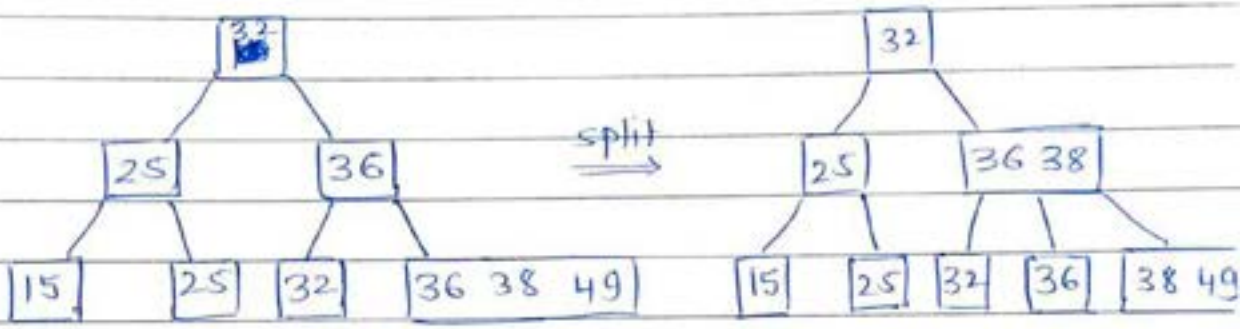
⇒ 25, 32, 15, 49, 36, 38, 42, 43, 45, 39, 37



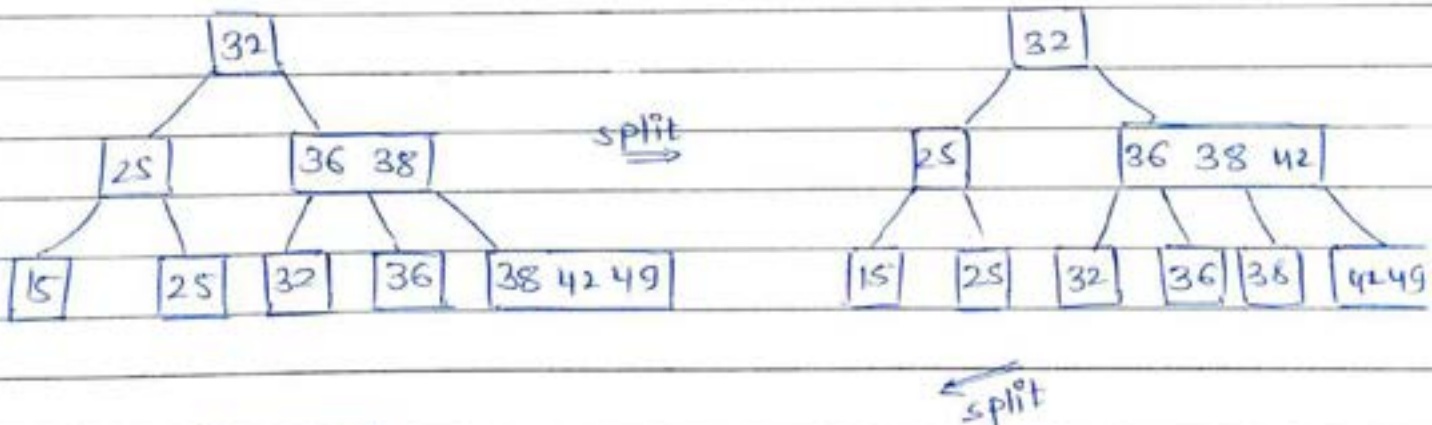
Date _____ Page No. _____

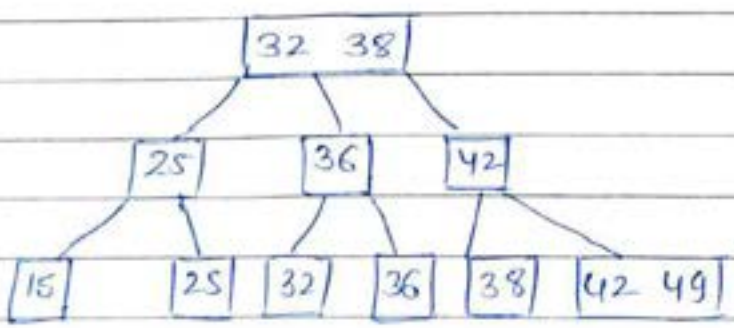


(6)

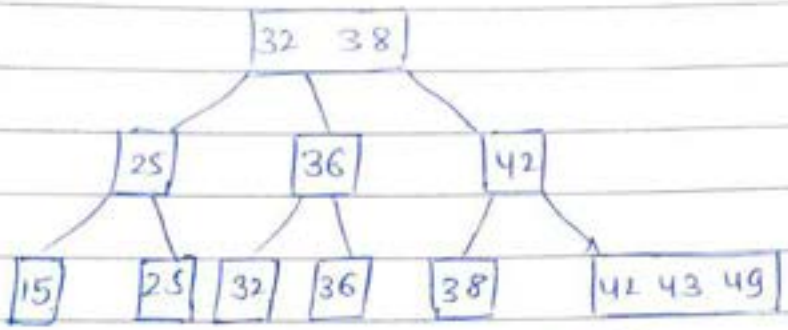


(7)

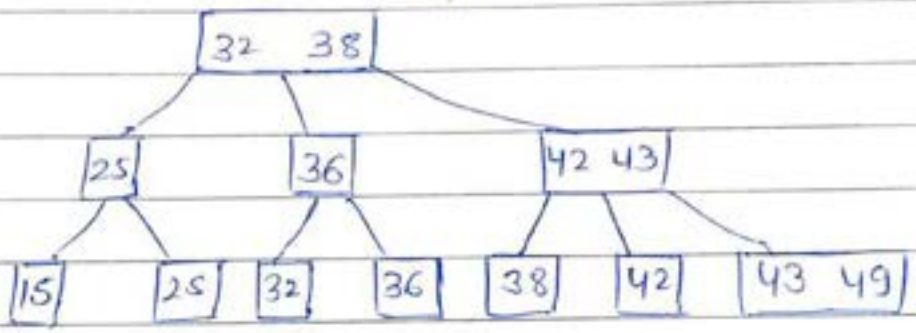




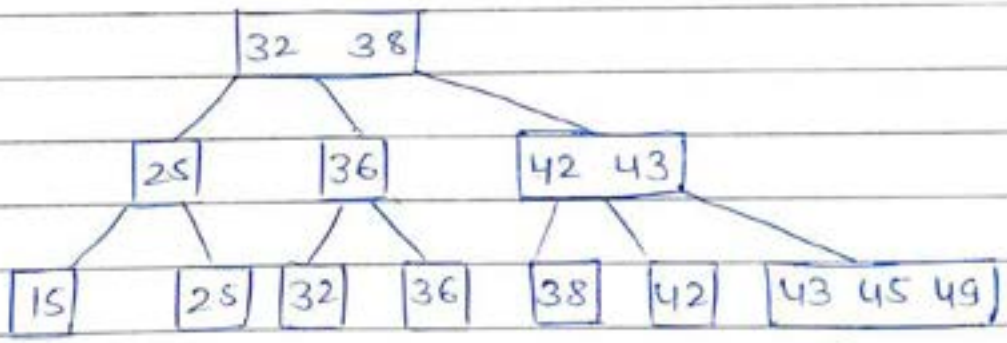
(8)



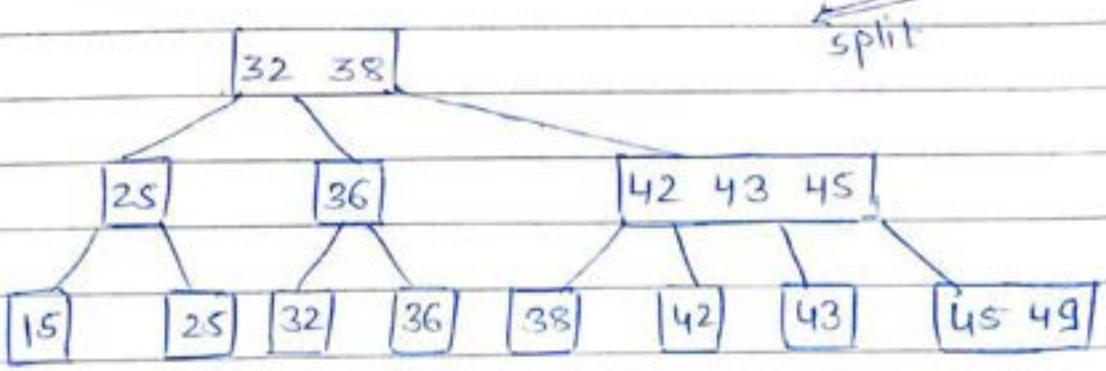
split



(9)

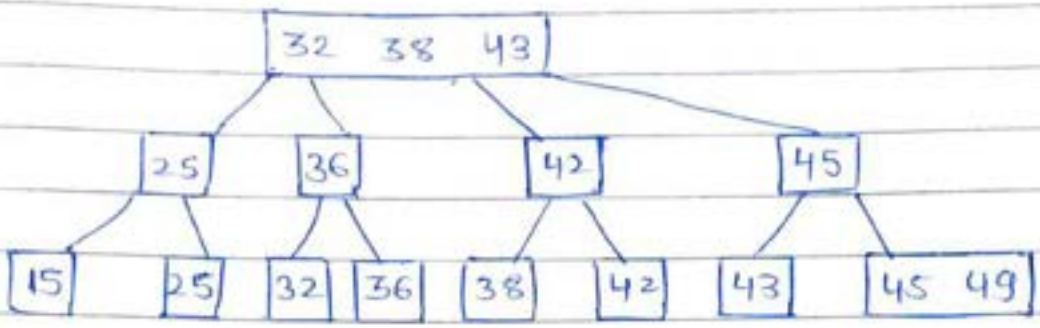


split



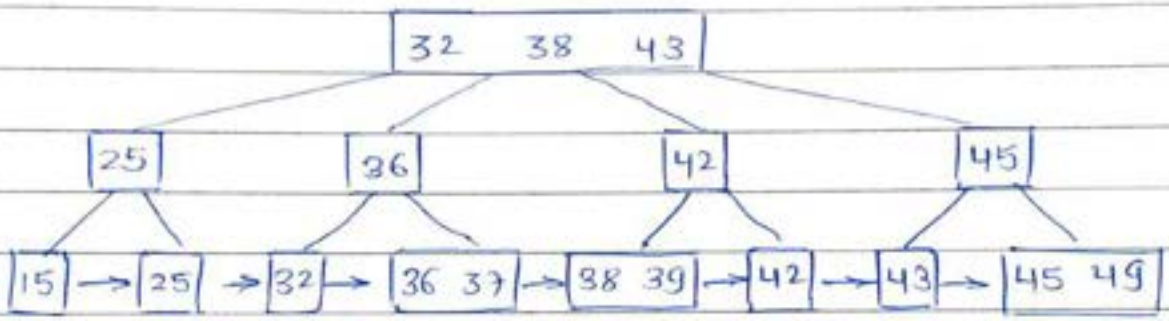
split

Date: _____ Page No. _____



(10)

(11)



* In B+ tree, always perform left biasing. [In case of even order]
* Left biasing means left subtree has more elements than right.

Date: Page No:

Que B+-tree of order 4

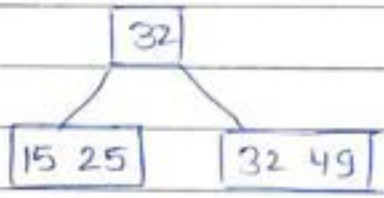
⇒ 25, 32, 15, 49, 36, 38, 42, 43, 45, 39, 37

(1) [25]

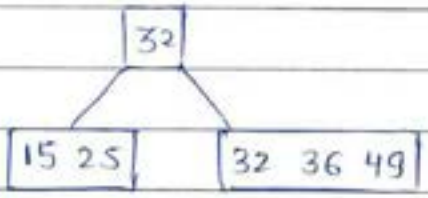
(2) [25 32]

(3) [15 25 32]

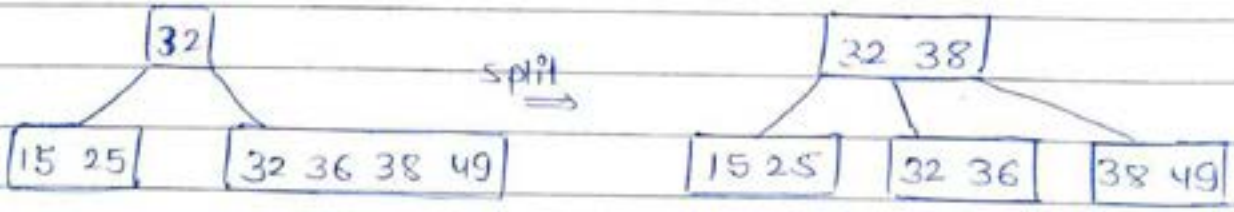
(4) [15 25 32 49] split →



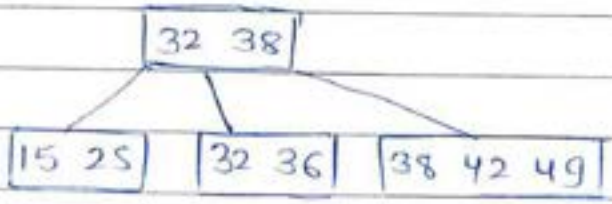
(5)



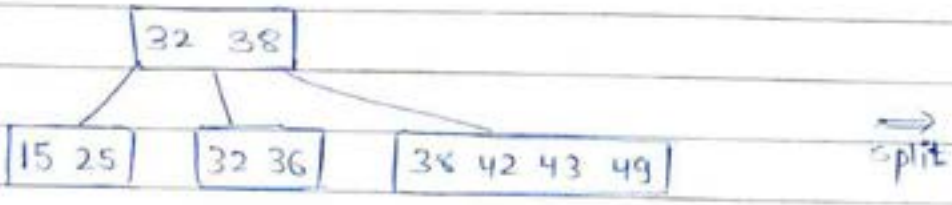
(6)



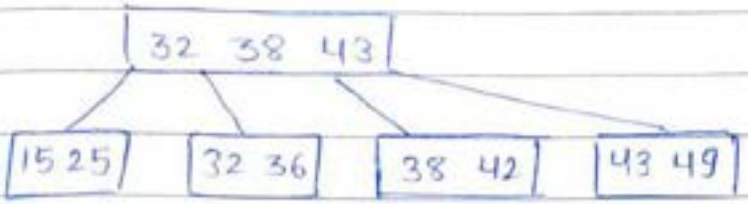
(7)



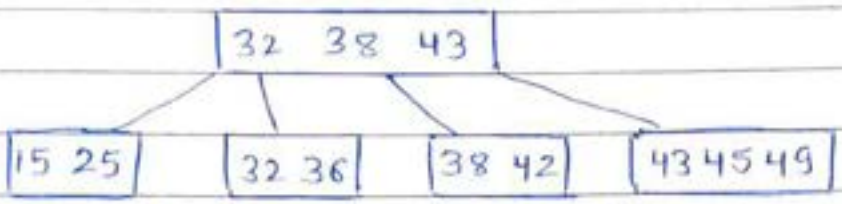
(8)



Date: _____ Page No: _____

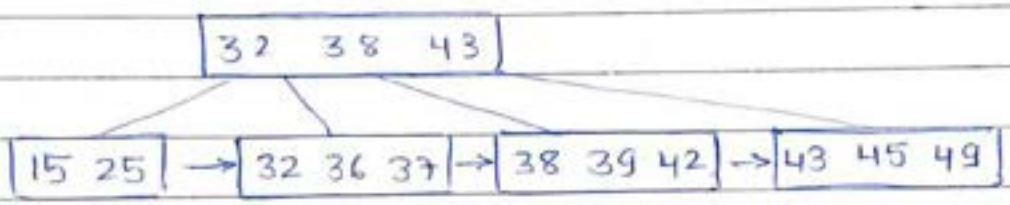


9



10

11



Date _____ Page No _____

* In B^+ tree, all the leaf nodes are connected to each other through linked lists.

* In B tree, record pointer is with key itself in every node

* In B^+ tree, record pointer is in leaf nodes thus the searching is always upto leaf nodes.

* In B tree, non leaf node contains key field, child pointer & record pointer.

* In B^+ tree, non leaf node contains key field & child pointer.

* order of tree is decided by keeping in mind that single node can be accommodated in single block of harddisk.

* In B tree, leaf node also stores child pointer. Null value is stored in their child pointers.

* In B^+ tree, leaf node doesn't store child pointer. It stores one sibling pointer.

Date: _____ Page No: _____

Que What is the order of non leaf node of B tree & B^+ tree if key field is 4 bytes, record pointer is 8 bytes, child pointer is 6 bytes & block size is 512 bytes.

⇒ If order = n, n child pointers are required in 1 node

* one child pointer is of 6 bytes, thus n child pointers will be of 6n bytes

* For non leaf or leaf in B tree: (In B tree, structure of leaf & non leaf are same)

⇒ $\underbrace{6n}_{n * c} + (n-1) * k + (n-1) * r = \text{block size}$

$$6n + 4n - 4 + 8n - 8 = 512$$
$$18n - 12 = 512$$

$$n = 29.11 \approx \underline{29} \quad \cancel{30}$$

Ans

* For non leaf in B^+ tree: (In B^+ tree, structures of leaf & nonleaf are different)

⇒ $n * c + (n-1) * k = \text{block size}$

$$6n + 4n - 4 = 512$$

$$n = 51.6 \approx \underline{51} \quad \cancel{52}$$

Ans

child pointer is also known as tree pointer

Date: _____ Page No: _____

- Sometimes, in B^+ tree, orders of leaf & nonleaf are defined separately;

* In B^+ tree, order of non leaf is defined as maximum no. of children a node can have. So,

* if order of non-leaf is n; it can have maximum n children & maximum (n-1) keys.

* For non-root & non-leaf nodes of order n;

minimum $\left\lceil \frac{n}{2} \right\rceil$ children & minimum $\left\lceil \frac{n}{2} \right\rceil - 1$ keys

—o—

* In B^+ tree, order of leaf node is defined as, maximum no. of keys a node can have.

* if order is n; leaf node can have maximum n keys, n record pointers & 1 sibling pointer & minimum $\left\lceil \frac{n}{2} \right\rceil$ keys.

* For root: minimum 2 children & 1 key

* For root & non root - non leaf; maximum n children.

—o—

Date: _____ Page No. _____

(for leaf)

Here, order = max. no. of keys, a node can have.

⇒ previous question for leaf in B^+ tree :

$$\Rightarrow n \times k + n \times r + c = \text{block size}$$

$$4n + 8n + 6 = 512$$

$$n = 42.16 \approx 42$$

Que Find max. no of nodes in B tree (or B^+ tree) of order = 10 & height = 4 (by default height starts from zero).

⇒

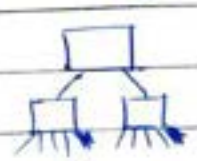
height	nodes
0	1
1	10
2	10×10
3	10^3
4	10^4

⇒ $1 + 10 + 10^2 + 10^3 + 10^4$

$= 11111$

Que Find min no. of nodes in B tree or B^+ tree of order = 10 & height = 4

- In B or B^+ tree max. or min no. of nodes are same but, max. or min. no. of (x) keys are different.

0		1
1		2
2		2×5
3		2×5×5
4		2×5×5×5

- min key for root = 1
- min key for nonroot = $\left\lceil \frac{n}{2} \right\rceil$

$= \left(\frac{10}{2}\right)$

$= 5$

$= 4$

⇒ For order = n & height = k (starting from zero)

- Max. nodes = $1 + n + n^2 + n^3 + n^4 \ldots\ldots - n^k$.
- Min. nodes = $1 + 2\left[1 + \left\lceil \frac{n}{2} \right\rceil + \left\lceil \frac{n}{2} \right\rceil^2 \ldots\ldots \left\lceil \frac{n}{2} \right\rceil^{k-1}\right]$

Date _____ Page No. _____

Que Consider a harddisk with block size 1024 bytes, key field 4 bytes, record pointer 10 bytes & child pointer 8 bytes. What should be the order of leaf & non-leaf node of B^+ tree.

⇒ For leaf:

⇒ $n * k + n * r + 1 * c = 1024$

$4n + 10n + 8 = 1024$

$n = 72.53 \approx \underline{72}$ ~~73~~

⇒ For non-leaf:

⇒ $n * c + (n-1) * k = 1024$

$8n + 4n - 4 = 1024$

$n = 85.66 \approx \underline{85}$ ~~86~~

---

* Block pointer is addr. of block
* Record pointer is addr of block & offset within the block.

* In multilevel indexing, 1st level has record pointer & 2nd level has block pointer.

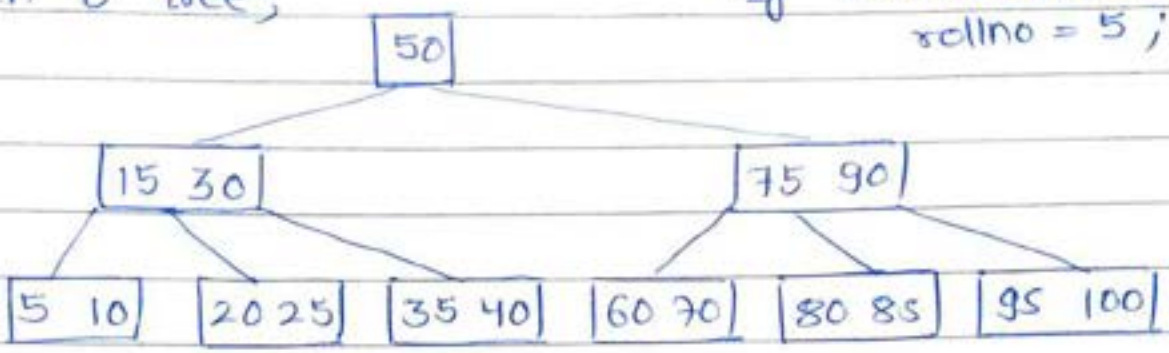
* In B/B^+ tree, child pointer is also block pointer as one node of B/B^+ tree equals one block of H.D.

Date: Page No.

⇒ 1 node of B tree = 1 block of harddisk (or B^+ tree)

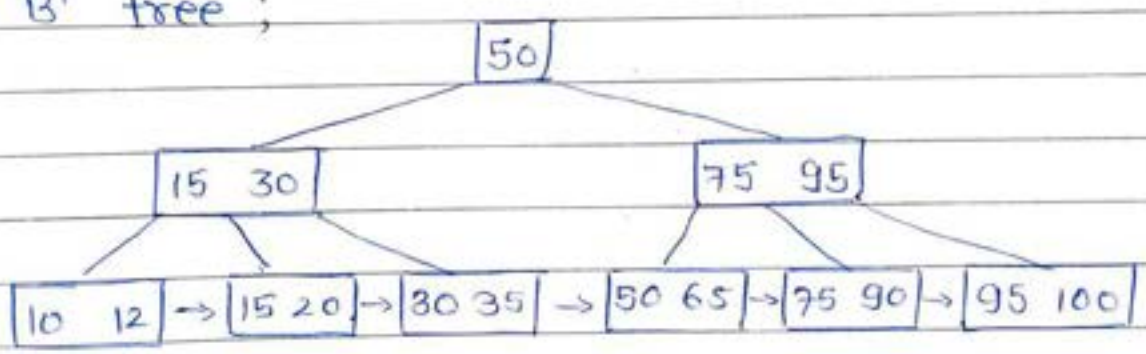
* In B tree;

eg select name where rollno = 5;



⇒ No. of blocks (read) to search 5 ⇒ 3 + 1 (database)
⇒ No. of blocks (r) to search 75 ⇒ 2 + 1 "
⇒ No. of blocks (r) to search 50 ⇒ 1 + 1 "

* In B^+ tree;



⇒ No. of blocks to search any key ⇒ 3 + 1 (database) as record pointer is stored in leaf nodes.

* But, yet B^+ tree is better because depth of B tree is more than that of B^+ tree. Thus, B^+ tree is used.

—x—

• In B^+ tree, non-leaf doesn't store record pointer, they just help to traverse to leaf nodes.

Date _____ Page No. ____

⇒ to print rollno between 10 & 100 ⇒ range query.

* In B tree, we have to search all the values between 10 & 100 whether they are present or not.

* But in B^+ tree, after searching first element, all the values can be accessed. Thus, range queries are faster ~~then~~ in B^+ trees.

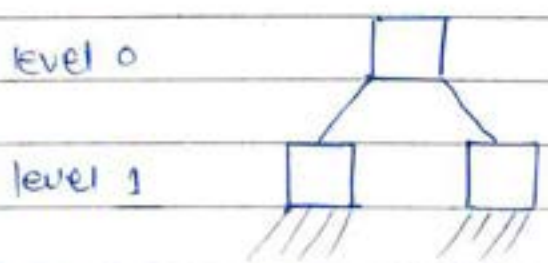
—ᵖ—

Que What is the min. no. of keys a B^+ tree of order 101 can store if level of tree is 2.

⇒ min. children = $\left\lceil \frac{101}{2} \right\rceil = 51$

min. keys = 51 − 1 = 50

Root ⇒ min. children = 2, min. key = 1



→ here, every children will have 50 keys

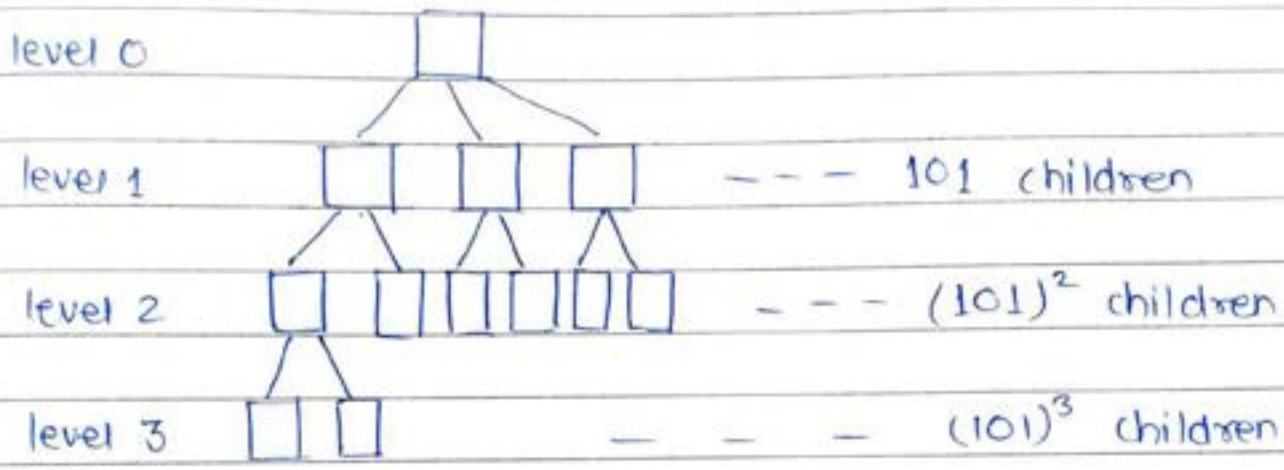
Thus, ⇒ (51 + 51) * 50 = 5100 keys (min)

* It's B^+ tree thus, we've computed for leaf nodes only.

Date _____ Page No: _____

Que What is the max. no. of keys a B^+ tree of order 101 can store if level of tree is 3.

⇒ max children = 101, max keys = 101 - 1 = 100

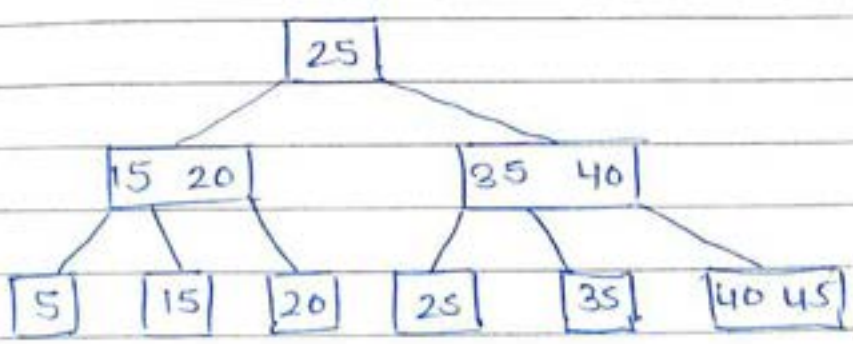


⇒ 101 * 101 * 101 = $(101)^3$ children

⇒ $(101)^3 * 100$ keys

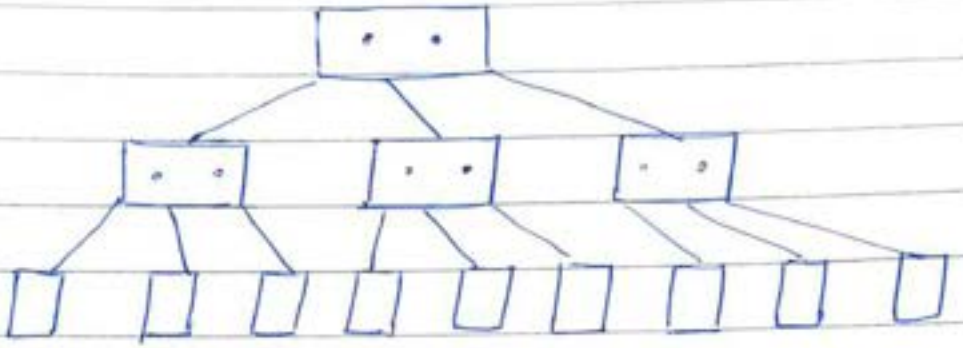
—x—

Que what is the max. no. of keys can be inserted in given B^+ tree without increasing level of tree, order = 3



* min key at root is 1, not $\left(\lceil \frac{n}{2} \rceil - 1\right)$.

Date: Page No.



* total keys at leaf nodes = 2 × 9 = 18
* keys already inserted = 7

⇒ Thus, 18 − 7 = 11 keys can be inserted.

Ques Find max. & min. no. of keys in B tree of order = 10 & height = 4.

⇒ Max keys = max. keys in 1 node × max. no. of nodes = 9 × max nodes

- Max keys ⇒ $(n-1)(1 + n + n^2 \dots n^k)$
- Min keys ⇒ $1 + \left[\left(\lceil \frac{n}{2} \rceil - 1\right)2\right]\left[1 + \lceil \frac{n}{2} \rceil + \lceil \frac{n}{2} \rceil^2 \dots \lceil \frac{n}{2} \rceil^{k-1}\right]$

Ques Find max. & min. no. of keys in B+ tree of order = n & height = k

- In B+ tree, all keys are at leaf level & max nodes at leaf level can be (n^k) & max. keys at a node is $(n-1)$

∴ Max. keys ⇒ $(n-1)(n^k)$

Min keys ⇒ $\underbrace{\left(\lceil \frac{n}{2} \rceil - 1\right)}_{\text{min keys at a node.}}\underbrace{\left(2 \times \lceil \frac{n}{2} \rceil^{k-1}\right)}_{\text{min nodes at leaf level}}$

Date _____ Page No. _____

Hashing :

* Hashing is a technique in which searching is not required.

* In hashing, we choose a hash function which when applied on a key, returns the index.

eg hash function : index = key %. 1000

- 4080 %. 1000 = 80
- 3126 %. 1000 = 126

key 4080 will be saved at index 80.

* The same hash function will be applied in reverse queries. (select name where rollno = 4080)

But, 4080 %. 1000 = 80
3080 %. 1000 = 80

* This problem is known as hash collision. If a hash function returns same value for 2 different keys.

* Thus, a hash function is chosen such that there will be less or no hash collisions.

Date: Page No:

Techniques to handle collision :

① Rehashing (Open Addressing)
↓
↳ Linear Probing
↳ Quadratic Rehashing

② Chaining

* In rehashing, we change the hash function for the key which collides.

Linear Probing :

* In linear probing, we search for free space linearly.

eg 3080 % 1000 = 80 ⇒ 3080 will be stored on 80 index.

4080 % 1000 = 80 ⇒ Now, we'll search for free space after 80.

eg 9998 % 1000 = 998 ⇒ 9998 will be stored on 998

2998 % 1000 = 998 ⇒ Now, we'll search for free space after 998.

& if there's no free space till end (999) then we'll search from the beginning.

Date _____ Page No: _____

⇒ drawback of linear probing :

① It may behave as linear search on collision if hash function is not chosen accordingly. (If the hash function returns same index for all the keys.)

② Primary clustering :

* Hash function should distribute data uniformly on array

eg Which hash function should be preferred.

H1 : k^2 mod 10 , H2 : k^3 mod 10

index	H1	H2
0	0	0
1	1/9	1
2		8
3		7
4	2/8	4
5	5	5
6	4/6	6
7		3
8		2
9	3	9

$0^2 \% 10 = 0$, $0^3 \% 10 = 0$
$2^2 \% 10 = 4$, $2^3 \% 10 = 8$
$3^2 \% 10 = 9$, $3^3 \% 10 = 7$
- - etc.

* in H1, 4 index(s) are not used while in H2, all are used.

Thus, H2 will be preferred. as it distributes uniformly.

* To remove primary clustering, quadratic rehashing is used.

Date _____ Page No _____

CS-2010
Que 52

Hash func. $h(k) = k \% 10$

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

* 6 values are stored

⇒ If linear probing is used, which order will be used?

(a) 46 42 34 52 23 33
(b) 34 42 23 52 33 46
~~(c)~~ 46 34 42 23 52 33
(d)

Also, how many orders of insertion exist using same key values so that you will get same resulting hash table.

Date: _____ Page No: _____

⇒ Given a hash table, what is load factor:

0	
1	
2	42
3	
4	34
5	
6	86
7	
8	
9	

$$\text{Load Factor} = \frac{\text{total keys in slots (occupied)}}{\text{total slots}}$$

$$= \frac{3}{10} = 0.3$$

Que Given a hash table of size 20, after how many insertions, probability of collisions will be greater than equal to 1/2.

⇒ On 1st insertion; prob = 0

⇒ On 2nd insertion; prob = $\frac{1}{20}$

⇒ On 3rd insertion; prob = $\frac{12}{20}$

Thus, on 11th insertion; prob = $\frac{10}{20} = \frac{1}{2}$

⇒ Thus, after 10 insertions, prob of collisions will be greater than or equal to 1/2.

Quadratic Rehashing:

* linear probing ⇒ +1, +2, +3 - - -

* quadratic rehashing ⇒ $+1^2, +2^2, +3^2$ - - -
⇒ +1, +4, +9, - - -

eg 80

⇒ 80, 81, 84, 89, 96, 105, 116, -

* Now, there will be no clustering but its drawback is that it stores key values only in selected index despite the array is empty.

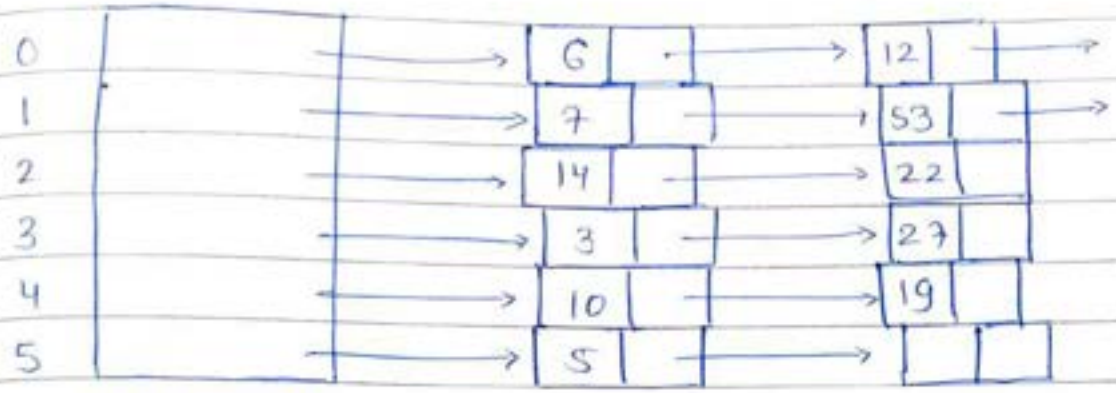
- Quadratic Rehashing.

⇒ index = Key % 1000

```
if (a[index] is not free)
    for i=1 to (n-1)
        if (a[(index + i²) %n] is free)      // Replace i² by i for linear probing
        {
            allocate the index;
            break;
        }
}
```

Date _____ Page No: _____

Chaining :



array of pointers (stores addresses)

Here, load factor = $\frac{11}{6} = 1.83$

* if all elements follow a single linked list then the complexity of searching will be O(n) (linear search). Thus, a hash function is used that distributes uniformly.

⇒ How to perform hashing in harddisk :

① Static Hashing

② Dynamic / Extendible Hashing

Date _____ Page No: _____

Static Hashing :

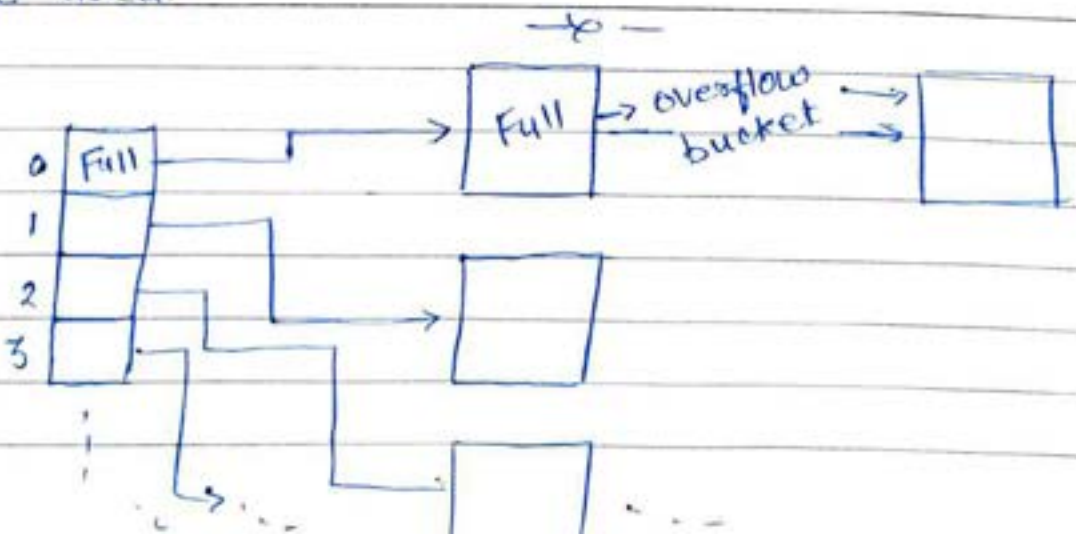
* If bucket (HD block) gets full, the database will make a new bucket known as overflow (or skew) bucket.

* When a table is created, database cannot know how many rows will be in the table. Thus, it cannot choose hash function accordingly. Thus, hash func. is chosen by some estimation / assumption.

eg key %. 4 ⇒ 4 blocks will be reserved in harddisk
if more rows are inserted in table, there will be many overflow buckets & linear search has to be appl.

key %. 1000 ⇒ 1000 blocks will be reserved in harddis
if less rows are inserted in table, it'll lead to wastage of storage.

* Static Hashing fails if the table grows dynamically. Thus, to remove this problem, dynamic hashing is used.

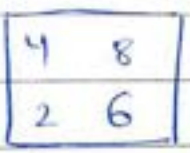


Date _____ Page No. _____

Dynamic Hashing :

* If hash function is like ; key % 10^6 , it won't reserve 10^6 blocks in HD.

⇒ First, only 2 blocks will be reserved & when they will get full, 4 blocks will be reserved then 8, 16, 32 - - -



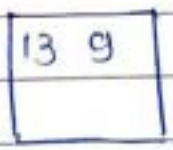
block 0

1 block

eg $13 = (1101)_2$

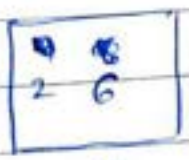
block ①

00



01

* Here, data is redistributed.



10

11

* If dynamic hashing is implemented successfully, searching will be zero.

Date _____ Page No. _____

* While using hashing, indexing is also important.

* Hashing can not be done in more than one column.

* Among with hashing on rollno, indexing will also be done on rollno.

* Range queries are better executed in indexing & equal queries are better executed in hashing.

- Dynamic Hashing (Navathe):

eg Hash func. ⇒ key % 16

block (1)

① 15 % 16 = $(1111)_2$
② 32 % 16 = $(0000)_2$
③ 7 % 16 = $(0111)_2$
④ 17 % 16 = $(0001)_2$
⑤ 16 % 16 = $(0000)_2$
⑥ 8 % 16 = $(1000)_2$
⑦ 5 % 16 = $(0101)_2$

* Initially, only 2 blocks are allocated.

assume, one block stores 5 records.

⇒ Now, 0th block gets full, add 2 blocks to it & redistribute keys 2 so on upto total 16 blocks

Here, global depth = 4 (max bits in remainder) & local depth.

TRANSACTION

Date _____ Page No. _____

* Transaction is a set of statements which either committed (executed) fully or shouldn't be committed at all.

* If it's executed partially, there will be some inconsistency.

⇒ There are 4 basic properties of transaction :

① A : Atomicity
② C : Consistency
③ I : Isolation
④ D : Durability

① Atomicity : Transaction should be atomic. ie either committed fully or nothing.

② Consistency : Transaction should lead one valid state to another valid state. eg Money transfer

③ Isolation : When a transaction is running, another transactions shouldn't run, especially when both are manipulating same data item. eg last seat reservation.

④ Durability : Once the end user is informed about successful completion of a transaction. then later on, it shouldn't inform about its failure.

* If a transaction ensures atomicity & isolation then it will definitely ensure consistency, if there's no error.

* If the transactions are executed one by one, then that execution is serial execution & all four properties are ensured in serial execution.

* But, the transactions cannot be executed serially, concurrent or parallel execution is required but the effect should be same as in serial execution i.e. all four properties must be ensured.

⇒ A concurrent schedule is serializable or valid if it is equivalent to one of the serial schedules.

PTO

Date _____ Page No. _____

eg

T1	T2
S11: update bank	S21: update bank
S12: set bal = 10000	S22: set bal = 20000
S13 where acno = 10;	S23: where acno = 10;

* if ~~enter~~ serial schedule → T1 T2 ✓
  then, bal = 20000

* if Serial schedule ⇒ T2 T1 ✓
  then, bal = 10000

* Thus, concurrent schedule should be equi. to (T1 T2) or (T2 T1)

⇒ If some concurrent schedule is given, then to check whether it is serializable or not, we've 2 tests of serializability;

(1) Conflict Serializability Test
(2) View Serializability Test

* 2 schedules can't be termed as equivalent if their outputs are same (they're just output-equivalent)

* 2 schedules are equivalent if in both schedules, the values read by T1 & T2 should be same.

Date _____ Page No. _____

Conflict Serializability Test :

Consider 3 set of operations to be conflicted;

① Read x → write x
② Write x → read x
③ Write x → write x

→ 2 operations are conflicted if: ① they belong to diff trans & they manipulate same item
② one of them must be write operation.

⇒ For this test, we draw precedence graph in which transactions are vertices & there will be an edge from T1 to T2 if T1 performs conflicting operation first & then T2 performs.

Qus Create precedence graph for given schedule;

S1 : $r_1(x)$, $r_2(x)$ $w_3(x)$ $w_1(y)$ $r_2(y)$: $w_1(x)$

* Just check for next conflicting

⇒ if precedence graph contains cycle, then the schedule doesn't satisfy conflict serializability test.

* no self loops

Date: _____ Page No: _____

Que S1 : $r_1(x)$ $w_2(y)$ $r_2(y)$ $w_1(x)$ $w_2(x)$ $w_2(y)$

⇒ no cycle, thus, the schedule is conflict serializable.

CS-2009

Que S1 : $r_1(x)$ $r_2(x)$ $r_2(y)$ $w_1(x)$ $w_1(y)$ $w_2(y)$

⇒ cycle, thus, not conflict serializable.

S2 : $r_1(x)$ $r_2(x)$ $r_2(y)$ $w_1(x)$ $w_2(y)$ $w_1(y)$

⇒ no cycle, thus, conflict serializable.

S3 : $r_1(x)$ $w_1(x)$ $r_2(x)$ $w(y)$ $r_2(y)$ $w_2(y)$

⇒ no cycle, thus, conflict serializable

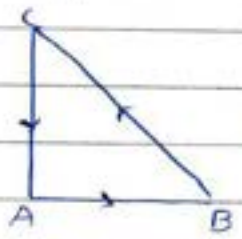
—x—

⇒ How to find order of serializability :

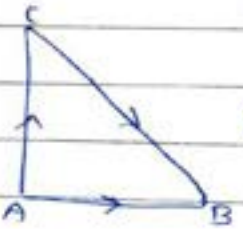
(How to find that our concurrent schedule is equivalent to which of the serial schedules)
$(T_1\ T_2)$ or $(T_2\ T_1)$

Topological Sort :

* It's a graph traverssal technique that can only be applied on acyclic graphs (without cycles).



cyclic



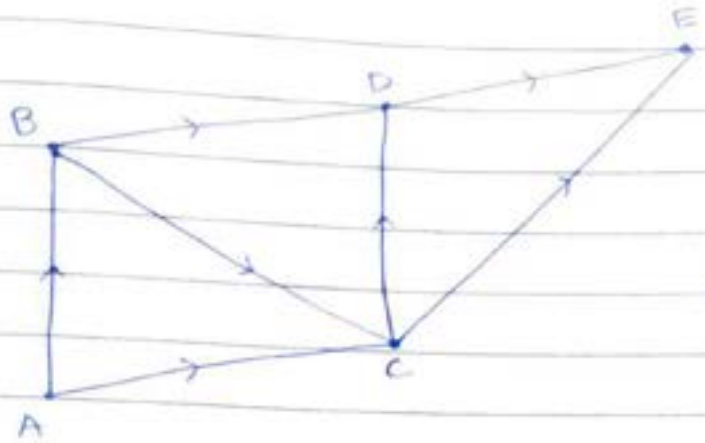
acyclic

* If a graph is acyclic, then there will be atleast one vertex with 0 indegree & atleast one with 0 outdegree.

* We can start from any 0 indegree vertex.

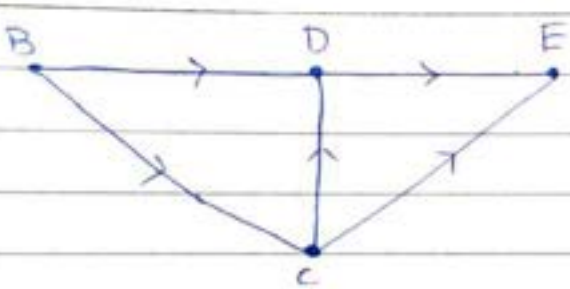
Date: _____ Page No. _____

Q9

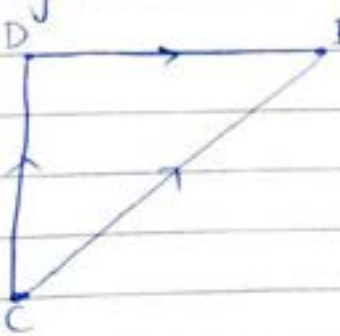


0 indegree ⇒ A vertex

⇒ traverse A & delete edges associated with A.



Now, 0 indegree vertex = B



Now, 0 indegree vertex = C

Thus, order of serializability ⇒ ABCDE

—x—

• 2 schedules are conflict-equivalent if their precedence graph is exactly same ie no. of edges should also be equ

Date _____ Page No. _____

* To find the order of serializability, perform topological sort of precedence graph.

eg 

⇒ 0 indegree ⇒ T2
Thus, order ⇒ T2 T1

eg check the serializability of given schedule;

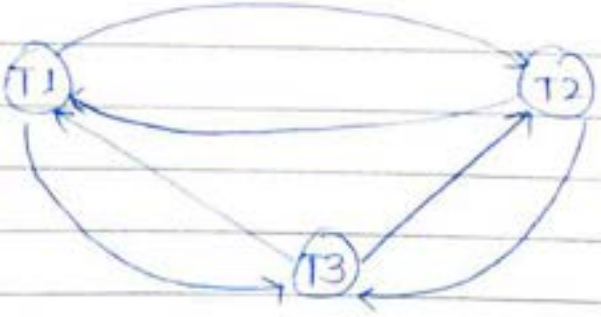
S: $r_1(x)$ $w_2(x)$ $w_3(y)$ $r_2(y)$ $w_1(x)$ $w_3(x)$ $r_1(y)$ $r_2(y)$

⇒ * Consider only 2 transactions at a time

T1,T2 ⇒ $r_1(x)$ $w_2(x)$ $r_2(y)$ $w_1(x)$ $r_1(y)$ $r_2(y)$

T1 & T3 ⇒ $r_1(x)$ $w_3(y)$ $w_1(x)$ $w_3(x)$ $r_1(y)$

T2 & T3 ⇒ $w_2(x)$ $w_3(y)$ $r_2(y)$ $w_3(x)$ $r_2(y)$



⇒ cycle, thus not serializable

* order ⇒

$2RA \Rightarrow r_2(A)$

Date _____ Page No. _____

ST-2007

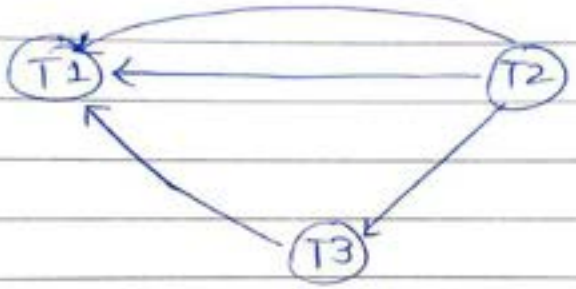
Que

S: 2RA, 2WA, 3RC, 2WB, 3WA, 3WC, 1RA, 1RB, 1WA, 1WB

* 1 & 2 ⇒ 2RA, 2WA, 2WB, 1RA, 1RB, 1WA, 1WB

* 2 & 3 ⇒ 2RA, 2WA, 3RC, 2WB, 3WA, 3WC,

* 1 & 3 ⇒ 3RC, 3WA, 3WC, 1RA, 1RB, 1WA, 1WB

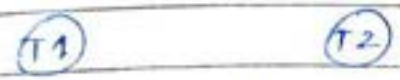


⇒ no cycles, thus, serializable.

* Order ⇒ T2, T3, T1

* Conflict Serializability is strictor definition of serializability i.e there are some schedules which are serializable but doesnt satisfy conflict serializability test.

* If there's no edge in precedence graph then the schedule is serializable & order can be T1T2 or T2T1



Date: _____ Page No: _____

View Serializability Test :

T1	T2	T3
r(A)		
	w(A)	
w(A)		
		w(A)

⇒ $r_1(A)\ w_2(A)\ w_1(A)\ w_3(A)$

- It's not conflict serializable as it can't be equivalent to any serial schedule.

* Here, write of T2 & T3 is not read by any transaction, thus their order doesn't matter.

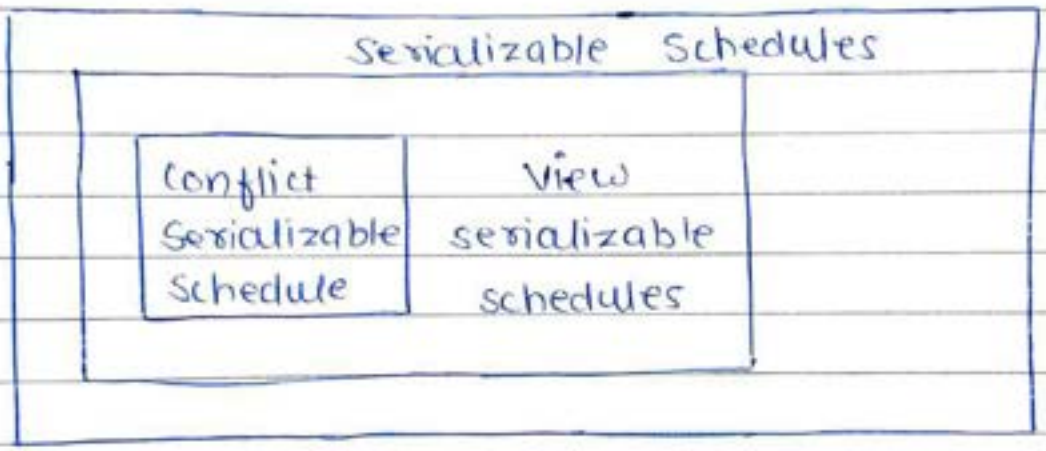
* T2 & T3 are doing blind writes here, as they are writing A without reading.

⇒ If there is blind write & the schedule is not conflict serializable, then only you've to check for view serializability test.

Date _____ Page No. _____

* Every conflict serializable is view serializable but every view serializable schedule is not necessarily a conflict serializable.

* If there is no blind write & schedule is not conflict serializable then it won't definitely be view serializable.



Que Given concurrent schedule is equi to which serial schedule?

→

T1	T2
Read A	
A = A + 50	
Write A	
	Read A
	A = A + 30
	write A
	Read B
	B = B - 30
	Write B
Read B	
B = B - 50	
Write B	

① T1 T2 ⇒ T1 reads B's initial value but there T1 reads B's updated value

② T2 T1 ⇒ T2 reads A's initial value but there T2 reads updated value

③ Both

④ None

This schedule $[r_1(A)\ w_2(A), w_1(A), w_3(A)]$ is not conflict serializable but it's view serializable as view checks for order of reads only.

Date: Page No:

⇒ How to check View Serializability :

T1	T2	T3	T_f
$r(A)$			
	$w(A)$		
$w(A)$			
		$w(A)$	
			$r(A)$

- Add one trans. T_f

* T_f will read all the data items which have been written. (only A, here)

* View Serializability checks order of read operations only. to check equivalence of schedules.

⇒ T1 reads the initial value of A.

⇒ T_f reads the value of A written by T3.

Serial schedules:

① T2 T1 T3 T_f ✗ Here, T1 reads updated value of A, not initial. ✗

② T1 T3 T2 T_f ✗ Here, T_f reads value by T2, not T3

③ T1 T2 T3 T_f ✓ this given concurrent schedule is equivalent to T1 T2 T3 T_f serial schedule.

— ⁂ —

* A schedule is view serializable if it's view equi. to any serial schedule.

Date _____ Page No. _____

* View Serializability cannot be checked by finite time algorithm. It's an NP-complete problem.

* Practically, server cannot check even for conflict serializability. Thus, there are some protocols by which servers ensure serializability;

① Two Phase Locking Protocol
② Time Stamp based Protocol

—x—

• How many schedules are possible with n transactions having $n_1, n_2 \dots n_n$ statements respectively.

⇒ $\frac{(n_1 + n_2 + n_3 \dots n_n)!}{n_1!\, n_2! \dots n_n!}$ → because internal order of a transaction can't be violated.

↓

total schedules

⇒ total schedules − $\frac{n!}{\text{serial}}$ ⇒ concurrent schedules

—x—

* for view equi, check for ① initial reads of all data items used
② final writes —— " ——
③ intermediate reads only

* If all 3 are same as any serial schedule, then schedule is view serializable.

* ~~Same trans should possess~~

Concurrency Control Protocols :

① Two Phase Locking Protocol (2PL) :

* Locking : If a transaction is manipulating a data item then another transaction should not interfere unless first transaction completes its execution.
  Thus, the first transaction will lock that data item

* There exists two types of locks ;

① Shared Lock : It's used when the data items have only to be read.

② Exclusive Lock : It's used when the data items have to be written or manipulated.

* Many shared locks on a data item can be applied at a time but only one exclusive lock can be applied.

	Unlocked	S	X
S	✓	✓	X
X	✓	X	X

Date _____ Page No. _____

* Exclusive lock can be applied only when that data item is unlocked.

⇒

T1	T2
X (A)	
R (A)	
W (A)	
unlock (A)	
	X (B)
	R (B)
	W (B)
	unlock (B)
X (B)	
R (B)	
W (B)	
unlock (B)	
	X (A)
	R (A)
	W (A)
	unlock (A)

⇒ cycle, thus not serializable.

- Here, we've performed locking-unlocking, but yet serializability is not ensured.

Date: _____ Page No: _____

* Thus, random locking & unlocking cannot return a serializable schedule. Thus, locking & unlocking should be done according to some rules known as two phase locking protocol (2PL).

⇒ 2PL protocol divides the transaction in two phases;

① Growing Phase ② Shrinking Phase

* A transaction is in growing phase initially.

* A transaction can lock a data item when it's in growing phase; not in shrinking phase.

* A transaction enters in shrinking phase when it unlocks any data item.

* Thus, before unlocking a data item, we've to lock data items to be used.

Date: Page No:

⇒

T1	T2	
X (A)		
R (A)		
W (A)		
X (B)		only one exclusive lock can be used at a time
unlock (A)		
	X (B) → Blocked	
	R (B)	
	W (B)	
	X (A)	thus, T2 will be blocked & when T1 gets chance back it executes completely. Then T2 will be executed. Thus, this schedule is no more concurrent but serial, thus, serializable.
	unlock (B)	
R (B)		
W (B)		
unlock (B)		
	R (A)	
	W (A)	
	unlock (A)	

* Schedules under 2PL always ensure conflict serializability.

* advantage of 2PL :

* A schedule returned by 2PL always satisfy conflict serializability. test.

* drawbacks of 2PL :

(1) Cascade Rollbacking
(2) Deadlock

Date: _____ Page No: _____

⇒

T1	T2
① read x	③ read y
② write x	④ write y

⇒ How many schedules are possible?

* ② cannot come before ① & ④ can't come before ③.

eg S: $r_1(x)$ $r_2(y)$ $w_1(x)$ $w_2(y)$ ✓

S: $r_1(x)$ $r_2(y)$ $w_2(y)$ $w_1(x)$ ✓

S: $w_1(x)$ $r_1(x)$ $w_2(y)$ $r_2(y)$ ✗

* If a transaction has m statements & another transaction has n statements then there can be;

$$\Rightarrow \frac{(m+n)!}{m! * n!} \text{ schedules.}$$

⇒ in above example ⇒ $\frac{(2+2)!}{2! * 2!} = 6$ schedules

* no. of transactions = 2, thus $2! = 2$ schedules will be serial & remaining 4 will be concurrent.

serial : ① ② ③ ④ , ③ ④ ① ②

* concurrent schedules = $\frac{(m+n)!}{m! \times n!}$ — total − serial $t!$

where, t = no. of transactions

* If there are 3 transactions, there will be $3! = 6$ serial schedules.

Que How many concurrent schedules are possible? which are conflict serializable?

T1	T2
read x	write x
write y	read z

⇒ order can be T1T2 or T2T1.

T1T2 ⇒ ① $r_1(x)$ $w_2(x)$ $w_1(y)$ $r_2(z)$
② $r_1(x)$ $w_2(x)$ $r_2(z)$ $w_1(y)$
③ $r_1(x)$
④ $r_1(x)$
⑤ $r_1(x)$
⑥ $r_1(x)$

$\frac{3!}{2!} = 3$

Date: _____ Page No: _____

Que How many conflict serializable schedules are possible ?

T1	T2
read(x)	write(y)
write(y)	write(x)

Cascade Rollbacking :

T1	T2
X (A)	
read (A)	
write (A)	
X (B)	
unlock (A)	
	X (A)
	read (A)
	write (A)
	unlock (A)
read (B)	
if (B < 5000)	
rollback	

⇒ Now, T1 will rollback due to which T2 will also have to rollback as T2 is reading value of A written by T1. This is called cascade rollbacking.

Date: _____ Page No: _____

T1	T2
x(A)	
read(A)	
write(A)	
x(B)	
unlock(A)	
	x(A)
	read(A)
	write(A)
	unlock(A)
	commit
read(B)	
if (B < 5000)	thus, there will be inconsistency.
rollback	

⇒ Now, as T2 has committed, it cannot rollback. Thus, this is known as irrecoverable schedule.

thus, we prefer transaction to not read the changes by the uncommitted transaction.

Date: Page No:

* Reading the changes done by uncommitted transaction ⇒ required cascade rollbacking

Que S: $r_1(x)$ $r_2(x)$ $w_1(x)$ $r_3(x)$ C2 C3 C1

→ T1 & T2 are reading initial value of x but T3 is reading value of x updated by T1.

⇒ Required Cascade Rollbacking & Non-recoverable

Que S: $w_1(x)$ $w_2(y)$ $r_1(y)$ C2 $r_3(x)$ C1 C3

⇒ Required Cascade Rollbacking & Recoverable

Que S: $r_1(x)$ $w_2(y)$ $r_2(y)$ C2 $r_1(y)$ C1 $r_3(y)$ C3

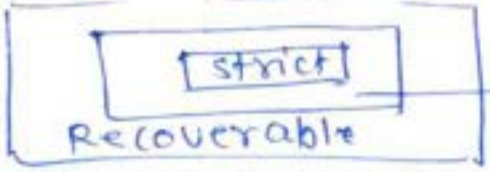
⇒ cascadeless schedule & thus recoverable

Que S: $w_1(x)$ $w_2(x)$ $r_3(y)$ C1 C2 C3

⇒ cascadeless schedule & recoverable.

∴ For recoverability, check for order of trans. commits

* the trans. performing dirty read should commit last.



Date: _____ Page No: _____

⇒ Strict Schedule: If value written by trans T_1 can't be read or overwritten by other trans until it (T1) either commits or aborts.

Ques S: $w_1(x)$ $w_2(x)$ $r_3(y)$ C1 C2 C3

⇒ Not a strict schedule

* Every strict schedule is cascadeless but not viceversa

⇒ To avoid cascade rollbacking, in basic 2PL,

* Strict 2PL : All the transactions should release their exclusive locks after 'commit' [Thus, dirty reads aren't possible here]
* But, it may make our schedule serial.

* Rigorous 2PL : All the transactions should release their exclusive & shared locks after 'commit'.

- In strict 2PL, shared lock can be upgraded into exclusive, only in growing phase.
Also, exclusive lock can be downgraded into shared & then it will move to shrinking phase.

* In strict & rigorous 2PL, dirty reads are not possible.

Date _____ Page No. _____

Deadlock :

T1	T2
x(A)	
	x(B)
x(B)	
	x(A)

* T1 is waiting for B & T2 for A. This results in deadlock & there's no solution for this, it's upto programmer.

→

- Conservative 2PL : A transaction should lock all the data items to be used, in the very beginning.

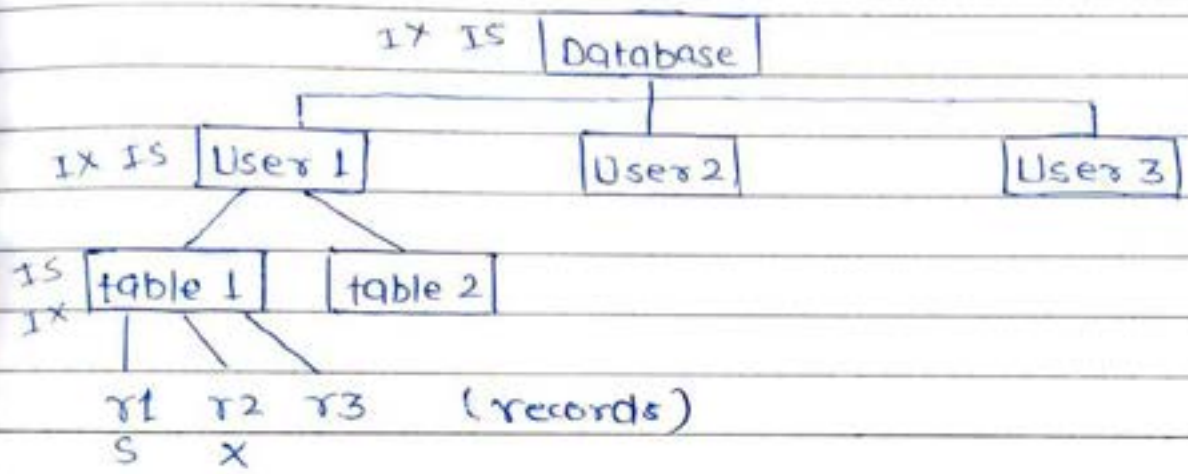
	Cascade Roll.	Deadlock
(1) Basic 2PL :	✓	✓
(2) Conser. 2PL :	✓	X
(3) Strict 2PL :	X	✓
(4) Rigorous 2PL :	X	✓

→

Date: _____ Page No: ____

Granularity of Locking:

* Database is collection of users
* User is collection of tables.
* Table is collection of records.



* If this database is to be copied somewhere, but if any record is locked, it cannot be copied.

* For a table to be locked, all the records should be unlocked. Thus, how does a table know whether its records are unlocked or locked.

* If a record is shared locked, then its parents, grandparents & so on will be intentionally shared locked.

Date: _____ Page No: ____

already locked → / ↓ to be locked	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
S	✓	X	✓	X	X
SIX	✓	X	X	X	X
X	X	X	X	X	X

* locking is pessimistic approach of concurrency control.

Date: _____ Page No: _____

(2) Time Stamp Based Protocol :

* It ensures serializability without locking.

* Every transaction has a time stamp which tells about when it was started

* No two transactions can have same time stamp

* Every data item has 2 time stamp → read time stamp
  write time stamp

eg A → read 11 ⇒ it was last read by transaction of time stamp 11
  → write 12
  ⇓
  it was last written by transaction of time stamp 12.

⇒ A ⟶ read 9
⟶ write 9

T1 ⇒ 11
read A
write A

* T1 starts on 11, thus, T1 can read changes done by transactions of time stamp less than or equal to 11.

⇒ Before reading A, T1 will first check write time stamp of A, if it is less than 11, then only it will read A else rollback itself & starts with new time stamp.

Now, A → read 9 11
→ write 9

⇒ Before writing A, T1 will check read & write time stamp to be less than 11 then only it will write A else roll back itself & starts with new time stamp

A → read 11
→ write 9 11

—x—

* If time stamps of transactions aren't given assume ;

T1 < T2 < T3 _ _ _

Date _____ Page No: _____

Que Is the given schedule possible?

⇒ S1 : $r_1(A)$ $r_2(A)$ $w_1(A)$

✓ ✓ ×

T1 ⇒ 11
T2 ⇒ 12

A ⟶ read ~~10~~ 11 12
↘ write 10

* read time stamp of A is not less than time stamp of T1, thus, it cannot write A.
Thus, the schedule is not possible.

* assume read & write time stamp of A to be less than 11 & 12; let's assume 10

⇒ S2 : $r_1(A)$ $w_2(A)$ $w_1(A)$

✓ ✓ ×

T1 ⇒ 11
T2 ⇒ 12

A ⟶ read ~~10~~ 11
↘ write ~~10~~ 12

⇒ not possible

Thomas Write Rule :

* The rule for read time stamp is same i.e. check for write time stamp.

* When some transaction is writing a data item then read time stamp of data item should be less than or equal to that of transaction and then if write time stamp of data item is less than or equal to that of transaction then only transaction should perform write operation.

serial schedule : * $r_1(A)$ $w_1(A)$ $w_2(A)$
↳ useless as no trans. is reading A before $w_2(A)$.

→
```
if (read time stamp (A) ≤ Ti)
{
    if (write time stamp (A) ≤ Ti)
    { Ti performs write }

    else
    { Ti should ignore its write }
}
else
{
    Ti should rollback
}
```

Date _____ Page No. ____

* Time stamp based protocol is deadlock free but cascade rollbacking is possible.

* Schedules under basic time stamp based protocol always satisfy conflict serializability test.

* Schedules under thomas write rule in time stamp based protocol may or may not satisfy conflict serializability test but they are always serializable.

* concurrent schedules are possible in thomas rule.

⇒

T1	T2
R(A)	
	W(A)
R(A)	

* T1 is reading 2 different values of A. Thus, this is known as unrepeatable read.

⇒

T1	T2
	W(A)
R(A)	

* T1 is reading changes done by uncommitted transaction. It's known as dirty read.

* When there's dirty read then the schedule will be required cascade rollbacking.

⇒

T1	T2
R(A)	
	W(A)
W(A)	

* Here, T2's write is useless i.e. A is updated just after by T1 transaction. It's known as lost update.

Date: _____ Page No: ____

Phantom Phenomenon :

T1	T2
select sum (sal) from emp	
	insert into emp values (100, 'M', 10000)
select sum (sal) from emp	

* row by this statement is known as phantom or ghost row.

Recovery :

* To ensure durability, database uses log based recovery.

* In log based recovery, database maintains a log file.

log file ⇒ information about operations performed

```
< T1 start >
< T1 A 100 150 >        // T1 updates A from 100 to 150
< T2 start >
< T2 B 200 210 >
< T1 C 150 50 >
< T1 commit >
• system crashes
```

* When will system update database (in HD)? on this basis database is of 2 types ;

① Immediate update : database is updated immediately.

② Deferred update : database is updated when transaction 'commits'.

Date: _____ Page No: ____

* Redo changes done by committed transaction in deferre update
* Undo changes done by uncommitted transaction. in immed. updat
* In immediate update, redo is not required.
* In deferred update, undo is not required.
* In immediate update, new values need not to be stored in log file as they've already been stored in database.

⇒ This log file is too large. thus, database inserts checkpoints between log file.

* Those who have committed till a checkpoint, there's no need to undo / redo for them.

Date _____ Page No. _____

CS-2015(II)

Que Consider simple checkpointing protocol ;

< start T4 >
< write T4, y, 2, 3 >
< start T1 >
< write T1, z, 5, 7 >
< commit T4 >
checkpoint
< start T2 >
< write T2, x, 1, 9 >
< commit T2 >
< start T3 >
< write T3, z, 7, 2 >
.

If crash happens now & the system tries to recover using undo & redo operations. What are the content of undo & redo list.

⇒ no undo/redo T4 // before checkpoint.
⇒ redo T2 // committed
⇒ undo T1 // not committed
⇒ undo T3 //

—x—

ER DIAGRAM

Date: _____ Page No: _____

* To design a database, we must know how many tables or columns will be present. For this purpose, ER diagram is made.

* Any real life object is entity (eg book)

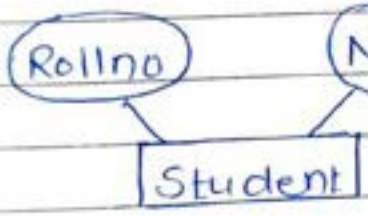
* All the books together form a book entity set

* In ER diagram, entity set is represented by rectangle.

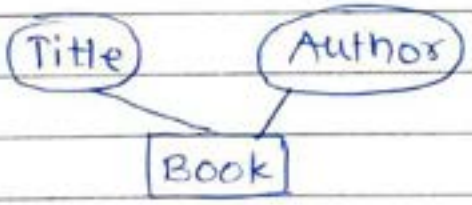
* Every entity (book) is instance of entity set.

* The attributes of entity are represented by ellipse.

eg



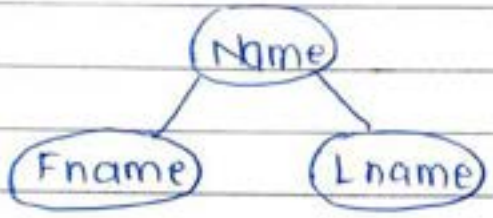
eg



Types of attributes :

① Composite attribute : An attribute which can further be divided into subattributes.

eg



② Multivalued attribute : Those attributes which can have multiple values.

eg (PhoneNo) * It's represented by double ellipse.

③ Derived attribute : Those attributes whose value can be derived from other attribute.

eg (age) * It's represented by dotted ellipse.

* age can be derived from DOB.

④ Simple attribute : Those attributes which aren't any of above three are simple.

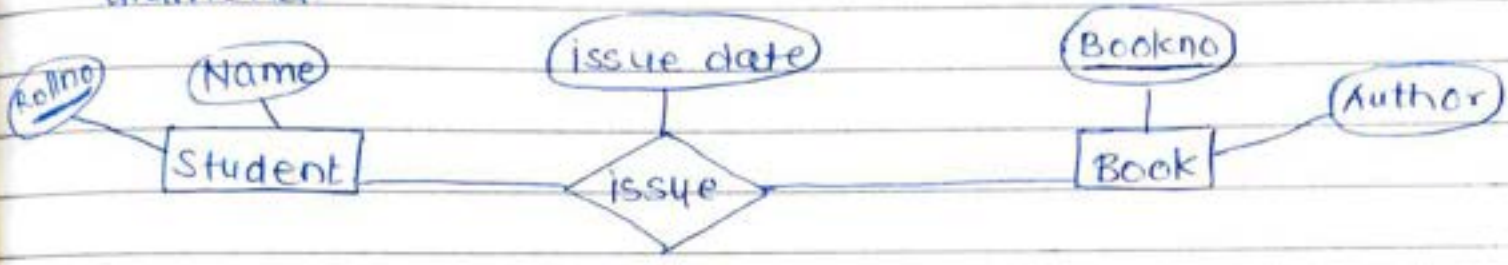
eg (marks)

Date _____ Page No. _____

* unique attribute (primary key) is represented by underline.

eg (Rollno)

* Relationship between 2 entities is represented by diamond.



* issue date is the attribute of issue relationship.

→ There can be 4 types of relationships;

① One to Many: Student ← Book

② One to One : Student ←→ Book

③ Many to One : Student → Book

④ Many to Many : Student — Book

Date _____ Page No: ____

* Total participation of an entity in relationship is represented by thick or doubled line.

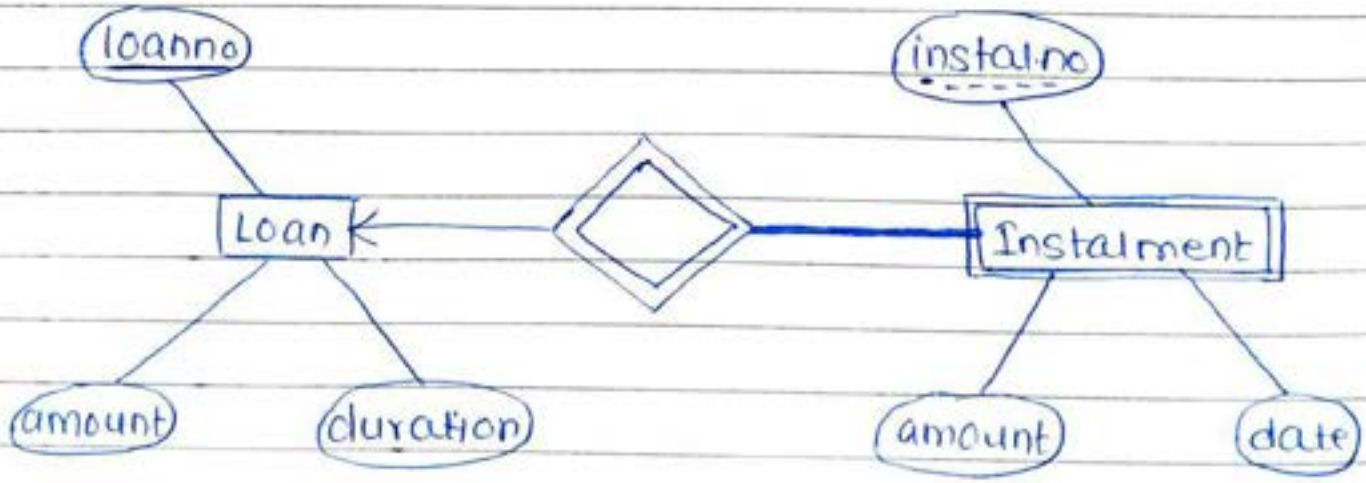
eg

* It means every book is issued to some student.

* Entities have their own primary key ⇒ Strong entity

* Entities dont have primary key ⇒ weak entity
(dont have even composite)

eg



Date: _____ Page No: _____

* There is no primary key of instalment entity. Thus, it's a weak entity & represented by double rectangle.

* Relationship between a strong & a weak entity is represented by double diamond.

* There will always be total participation of weak entity.

* There will always be one to many relationship between a strong & a weak entity.

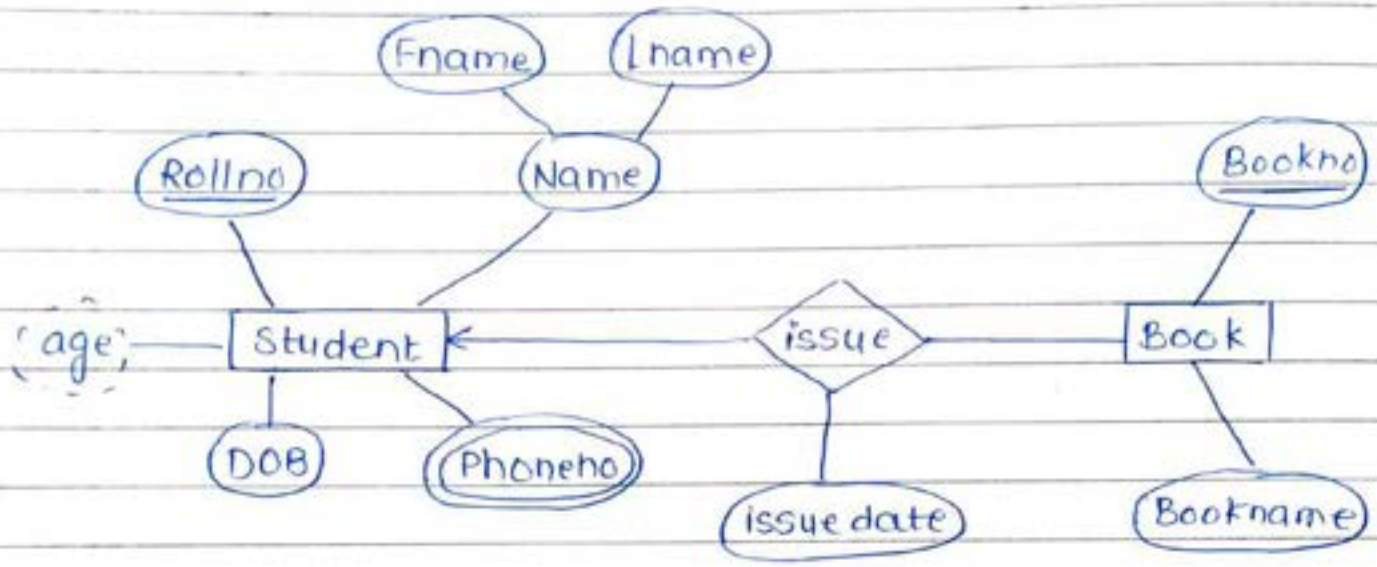
* That attribute of weak entity which when combined with primary key of strong entity creates a unique key is known as partial key or discriminator & it's represented by dotted line.

eg 

the relation returned by this method will always be in 3NF

Date: Page No:

⇒ How to map ER diagram with a table:



* there will be a table corresponding to every entity.

* For every simple attribute, there will be a column in table. (eg Rollno, DOB)

* For composite attribute, there will be no column but there will be columns for every individual attributes of composite attribute. (eg Fname, Lname)

* For derived attribute, there will be no column. (eg age)

* For multivalued attribute, there will be no column but a separate table, & in that table, primary key of main table is present along with phoneno.

eg | Rollno | Phoneno |

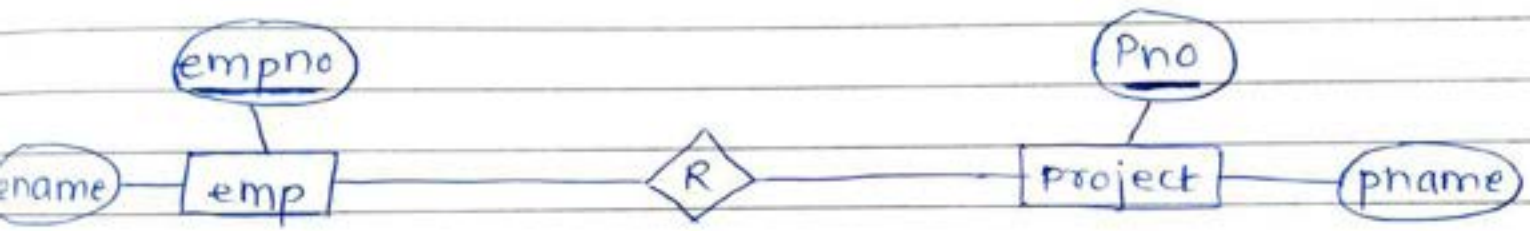
Date _____ Page No _____

* In a table of relationship, along with its attributes, primary keys of entities will be present in table

eg | Rollno | issuedate | Bookno |

⇒ When to create table for relationship :

① Many to many ⇒ always create table of relationship



* In table of relationship, primary key will be composite (empno, Pno)

eg emp:

empno	ename
1	e1
2	e2

Project :

Pno	pname
P1	ABC
P2	DEF

R:

empno	pno
1	P1
1	P2
2	P2

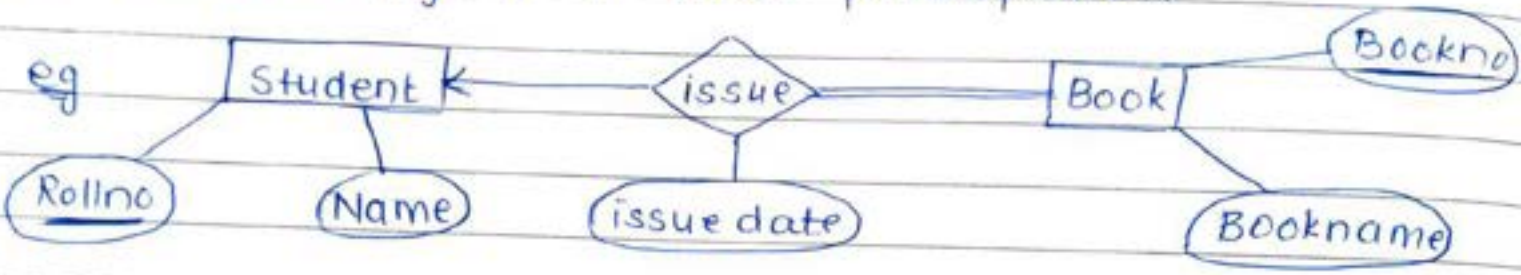
* weak entities are always in total participation, thus, the table of their relationship is never made.

Date: Page No:

② One to Many : Create table of relationship in All the cases except ;

(a) when many is in total participation.

eg



* Now, the attributes of relationship will be shifted towards many. i.e. issuedate in table of book.

* But, there will also be primary key of Student tabl

eg Book :

Bookno	Rollno	Bookname	issuedate

* If there's no total participation of many, then there will be a lot of books which are not issued to any student due to which large no. of NULL values have to be stored. Thus, we create table of relationship in this case, & don't shift the attributes of relationship towards many.

⇒ But, if question is min. no of tables, then ans. will be 3 not 3.

Date _____ Page No. _____

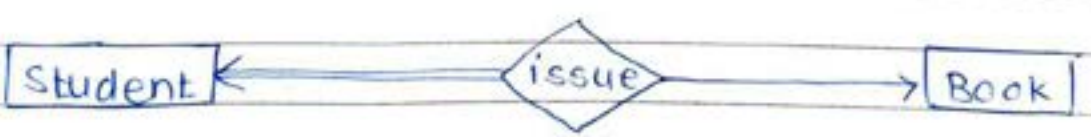
Book :

Bookno	Rollno	Bookname	issuedate
b1	1	ABC	20/2/17
b2	2	DEF	22/2/17
b3	NULL	GHI	NULL
b4	NULL	JKL	NULL
b5	NULL	MNO	NULL
⋮	⋮	⋮	⋮

* In one to many, primary key of many will be the primary key of relationship. (eg bookno)

③ One to One : Create table of relationship in all the cases except ;

(a) When any 'one' is in total participation.



* Now, ~~student~~ table the attributes of relationship will be shifted towards table of entity having total participation.

* But, there will also be primary key of Book table

eg Student :
/ Rollno / Name / issuedate / Bookno /

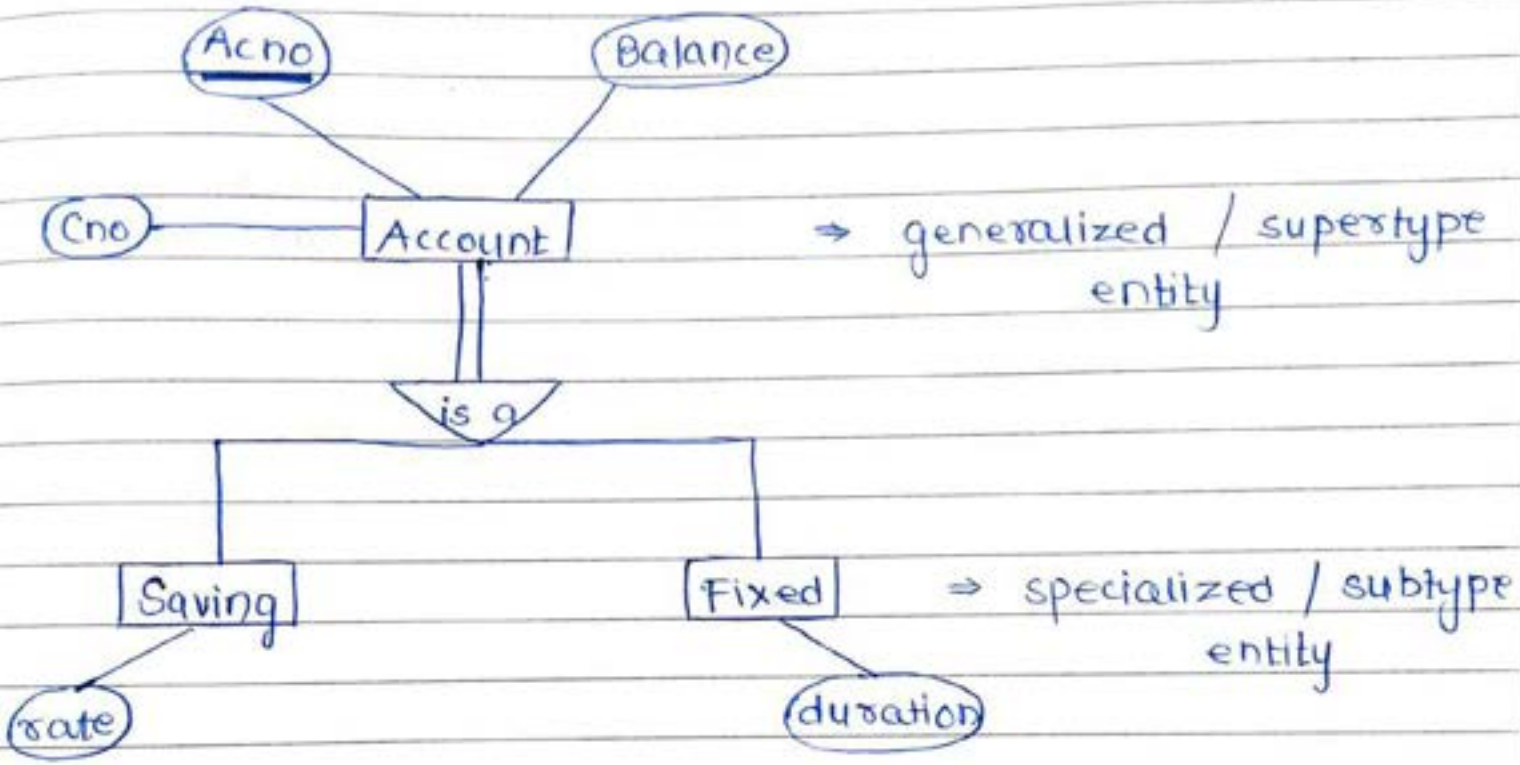
—x—

* If both have total participation, there will be single table for student & book.

—x—

Date: _____ Page No: _____

Generalisation & Specialization :



⇒ generalized / supertype entity

⇒ specialized / subtype entity

* interest rate, FD duration, etc. cannot be the attribute of every account.

* if every supertype entity belongs to a subtype entity (i.e. if every account is either saving or fixed) then this is known as total specialization.

* If there's an account which doesn't belong to subtype (eg current account) then this is known as partial specialization.

Date: Page No:

⇒ Mapping into table :

① way : Create tables for supertype & all subtype entities

* In sub type table, along with its attributes, primary key of supertype will also be there.

* every record of subtype will also be present in table of supertype.

OR

② way : Create tables for subtype entities only.

* In every table of subtype, along with its attributes, all the attributes of supertype will also be present.

* But, it fails when there's partial specialization.

eg Saving:

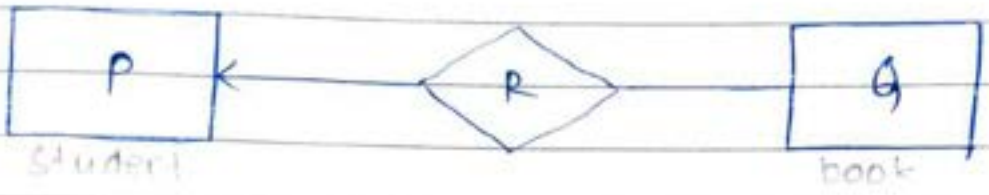
| Acno | balance | Cno | rate |

Fixed:

| Acno | balance | Cno | duration |

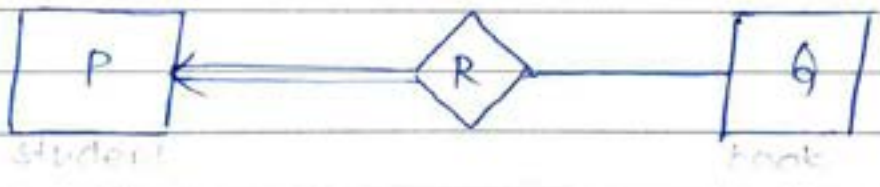
—x—

Que



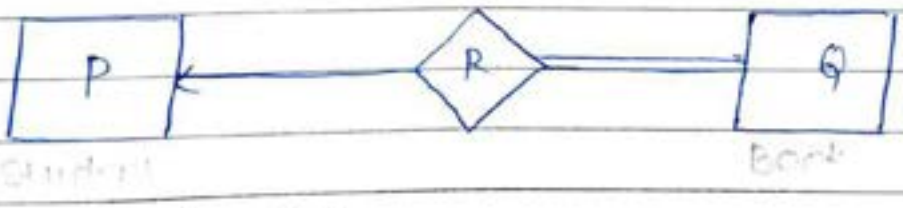
(a) $|P| = |Q|$
(b) $|P| \leq |Q|$
(c) $|P| \geq |Q|$
(d) no relationship ✓

Que



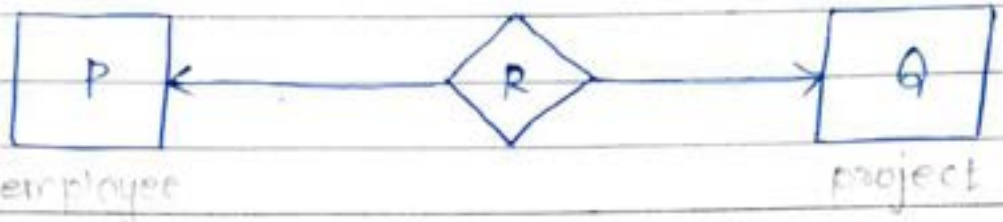
⇒ $|P| \leq |Q|$

Que



⇒ No relationship

Que what is the max & min cardinality of relationship R?

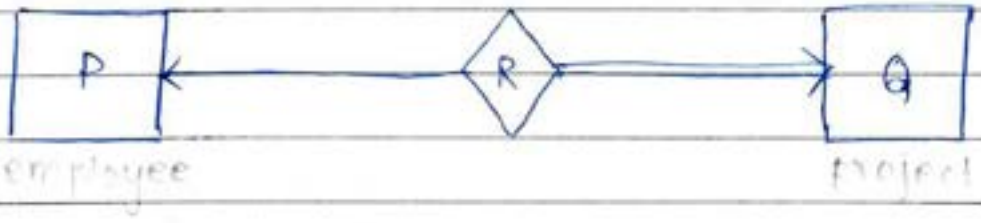


$|P| = 10$, $|Q| = 100$, $|R| = ?$

⇒ maximum ⇒ $10 * 100 = 1000$ rows in table of relationship

⇒ minimum ⇒ 0 rows as there's no total participation.

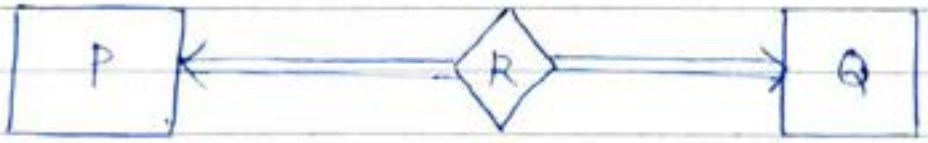
Que



$|P| = 10$, $|Q| = 100$,

maximum ⇒ $10 \times 100 = 1000$, minimum ⇒ 100 (1 emp - 10 rel)

Que



maximum ⇒ 1000, minimum ⇒ 100 (1 emp - 10 rel)

—e—