SQL → Structured Query language. Page no.

† DBMS
- File Based
- Relational DBMS
- No SQL

◦ ER Model:

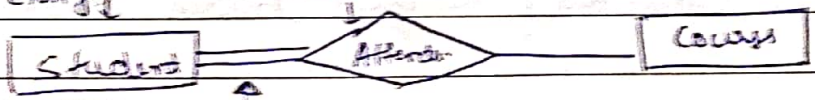
Entity Set → Main things like Student, Teacher, Course

Relationship Set → These generally connect the various entities

Attributes → Attributes of set like for student Name, Roll no, DOB ...

◦ Total Participation: Every entity of one side participates in the relationship.

Entity Set / Rly Set

eg 

eg 

◦ Weak entity: Do not have their own primary key.

eg 

Say 5 class with 2 section (A, B)

no By A and B we can't tell the class it belongs to.

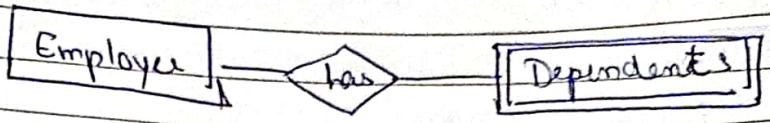
• Weak entity sets always have total participation.

• key is set of attributes which identify a particular entity.

Tep

PAPERWELL
Date :
Page no.

eg 2



eg 3

Keys in database

- Candidate key: Minimal set of attributes that derives all attributes. They can't be NULL.
  ~~eg: { Roll NO, PAN NO }~~

eg Student Table

Enroll No	Name	Address	Pan No	Contact No.

- so Ck → Enroll_No, Pan_No.
- We pick one of them as primary key (Enroll_No).
- Super key are greater than or equal to Ck

⇒ One of the candidate key is a primary key. Other keys are called Alternate key.

⇒ Ck is minimal.

We can say {Enroll_No, Name} be a key but its not a Ck, its a Super key. as Enroll_No is enough to identify uniquely.

→ Super key: Any set of attributes that identify rows / tuples in database

Minimal form of super key are candidate key.

Armstrong Axioms.

→ Reflexivity. $A \to A$

→ Transitivity if $(A \to B, B \to C)$, then $A \to C$

→ Augmentation. if $(A \to B)$ then $(AC \to BC)$

Examples:

Ex 1 R1 (A, B, C, D)

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

Soln

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

$B^+ = \{B\}$

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

$D^+ = \{D\}$

So A is a candidate key and also a primary key.

{AB, AC, AD} can be super keys.

Similary ABC, ACD, ABCD.

A itself is also a super key.

Ex 2 R2 (A, B, C, D) $AB \to CD$.

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

So AB is CK

Ex 3 R3 (A, B, C, D)

$B \to AC$, $C \to D$

$B^+ = \{B, A, C, D\}$. B is CK

PAPERWELL
Date :
Page no.

A table will always have atleast one superk, CK, PK. (consider whole ABCD) (If it doesn't have duplicates)

Theory

Database : is a collection of inter-related data which helps in retrieval, insertion and deletions of data. and organises data in form of table, views, schemas etc. . .

DBMS → Softwares, MySQL, oracle . . . .

③ → retrieval
② → Updation (insert, delete, update)
① → Defination (creation, modification)
④ → Users Admin. (registring and monitoring users, enfor. data security, integrity, concurrency control

Terms in Database :

- Redundancy
- Inconsistency (if multiple copies of same data don't match each other)
- concurrency (same data accessed by multiple users)
- Integrity (Data stored is meaning full)

Constraints
- Domain
- Key
- Entity Integrity
- Referencial Integrity

In File system DB : Redundancy of data, code is written by user, inconsistency of data, unsorted data (diff. retrievals), less security, One access / time, No backup.

Table	Fields
Records	Records → Row Fields → column (Attributes)

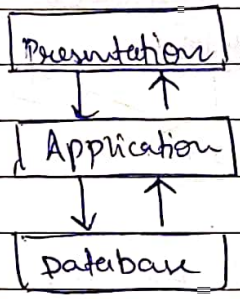
PAPERWELL
Date :
Page no.

RDBMS Advantage of DBMS:

- Min. redundancy
- Easily accesible
- Data consistency
- Data concurrency
- Back up mech.
- Data security.

→ Collection of info stored into dB at particular moment is called instance of DB.

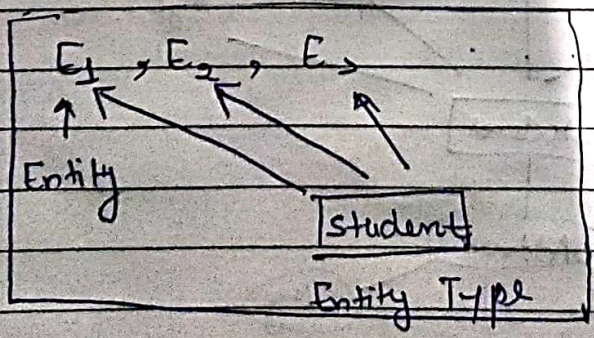
→ Schema is used to define ~~as~~ overall design of DB



- Data Independence means change of data at one lvl should not affect other levels.

ER Model Entity is object with physical existence. like a particular person, car, house, job, univ. course.

An Entity is an object of Entity Type and set of all entities is called entity set.



- **Attributes**: properties of entity type

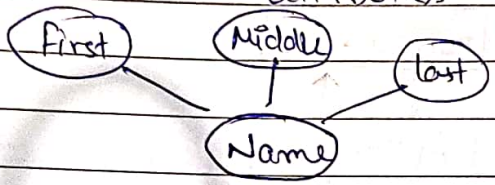
  eg → Student : name, roll no, DOB, Address

  Student → Entity Type.

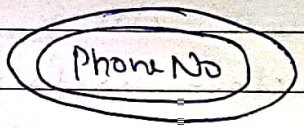
  name, roll no, DOB, Address → Attribute

(1) **Key attribute** : Unique — Roll no

(2) **Composite attribute**: An attribute composed of many attributes.

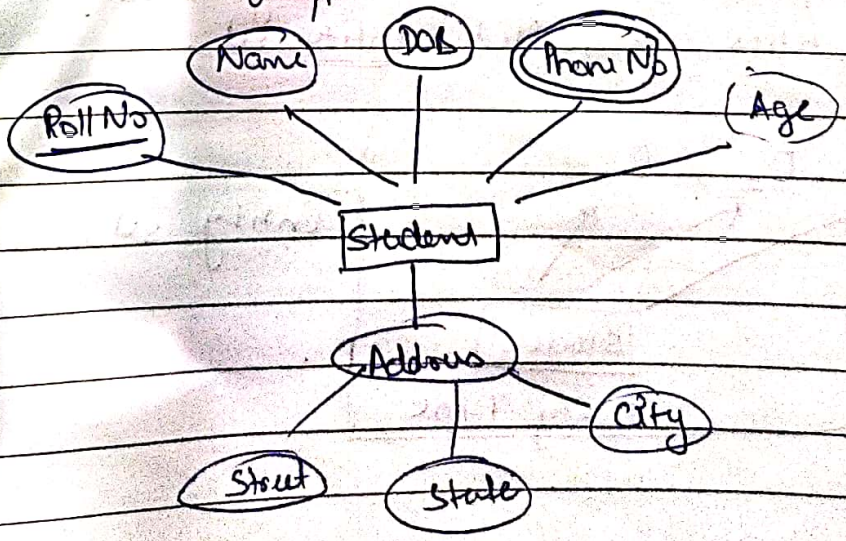


(3) **Multivalued Attribute**: An attribute consisting more than one value for a given entity.



(4) **Derived Attribute**: Age — can be derived from DOB.

So for Entity type student



Relation Model

Student

Name	Roll No	Address	Phon No
	1		
	2		
	3		
	4		

Each row is called a tuple

Cardinality : No of tuple (4)

Degree : No of attributes (5)

* Foreign Key:

- Some columns of one table are closely related to some columns of other table.

Eg

Table 1 — Order Table

Orders Id	date	code	cust_Id

Table 2 — Customer Table

Cust_ID	Name	Address

cust_Id → Foreign key. (references Cust_ID)

Foreign key can be primary key of same other table.

Creating Foreign key:

```
CREATE TABLE Order(
    order_id        INTEGER     Primary key.
    Order_book      DATE
    Cost            INTEGER
    Customer_ID     REFERENCES  customer (customer_ID)
```

Foreign statement — Table Name — Primary key (It can be set of attributes)

DDL / DML

Date :
Page no.

Refuential Integrity. If customer table is deleted / updated, the ~~prim~~ foreign key points to null.

→ ON DELETE CASCADE } certain instruction to handle this

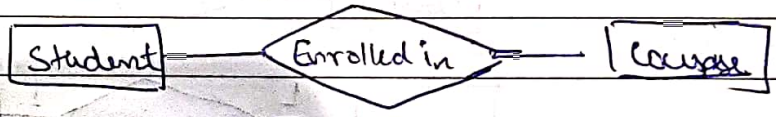
↓

Like this deletes foreign key from main table.

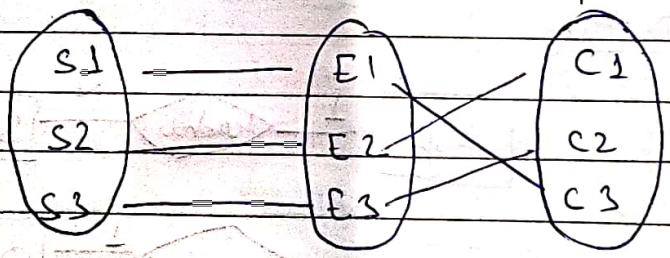
Relation Type and Relation Set — A rel^{n}p is association by entity types.

ex: Student — enrolled in — course.
Entity type — relation — entity type.

degree 2

→ 

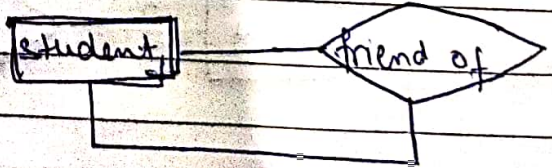
A set of relationships of same types is relation set.

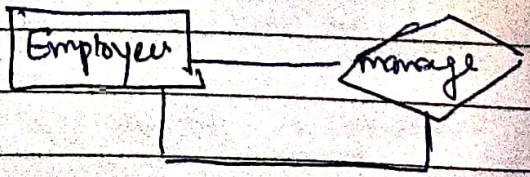


* Degree of Rln sp set: No of entitity set participating in relation.

Types

(1) Unary :

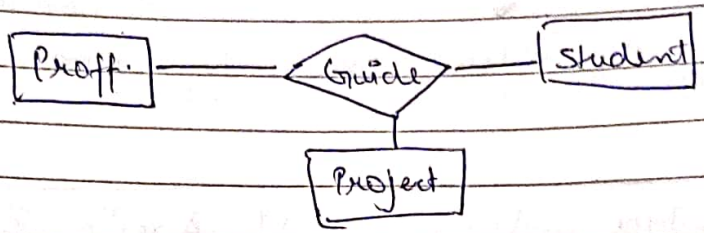




PAPERWELL
Date :
Page no.

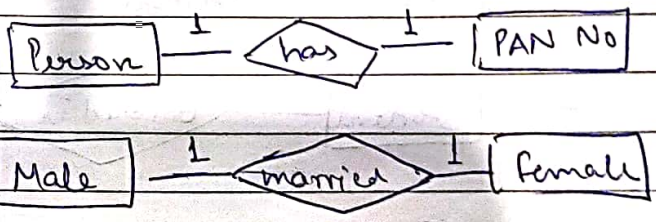
* Binary : Teacher — Teaches — Subject

n-ary when there are n entity sets in rel^n.

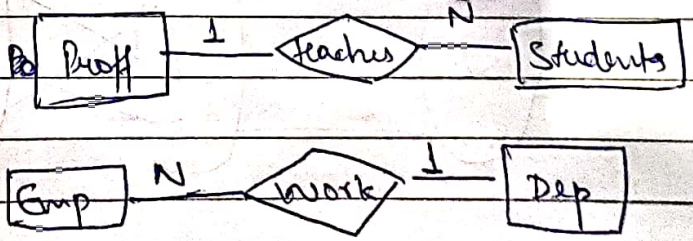


* Cardinality : no of times an entity set participates in rel^n.

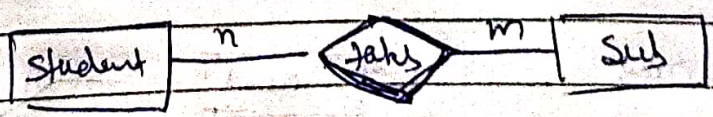
(1) One to One.



(2) One to Many

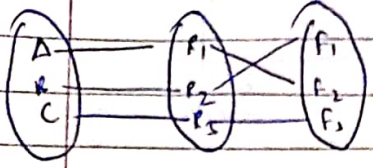


(3) Many to Many

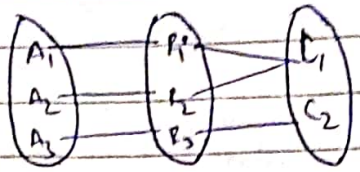


P prof
1 subj

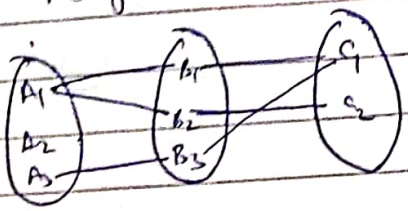
1 to 1



One to many



Many to many



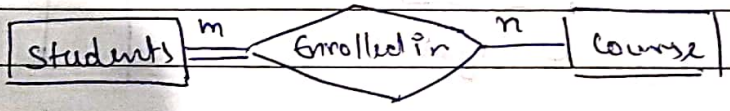
* Participation Constraints:

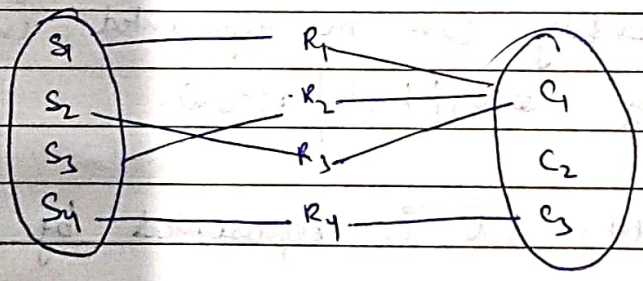
(1) Total Participation: Each entity of entity set must participate in r/n.

eg 

(2) Partial Parti: May or may not.

Example





* Weak Entity: same entity type for which attribute doesn't exist.

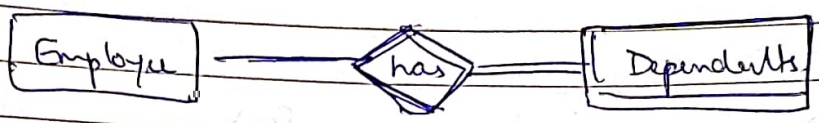
Represented by double rectangle.

The participation of Weak entity is always total.

The r/n b/w weak entity and its identifying strong entity type is called an identifying r/nship.

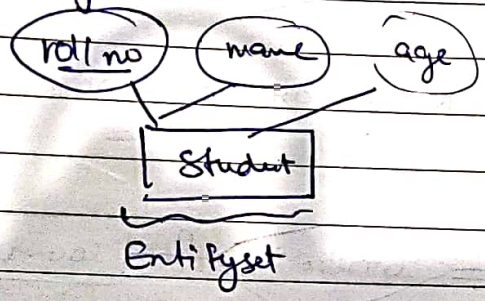
double diamond

Date :
Page no.



Entity set

ER-model



Relational node

Student

roll no	name	age
1	A	19
2	B	20
3	C	21

* Entity can be represented in a relational model by rows, tuple, record.

* Entity set is represented by table in relational model.

* Attributes are units that describe chars. of entity.

- Set of permitted values for attribute domains.

Instance → data. Schema → structure / PAPERWELL

Date :
Page no.

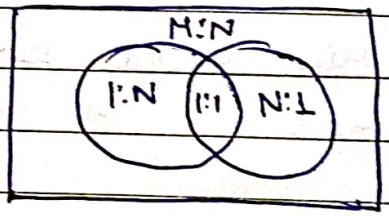
Relationship: association b/w two or more entities of same or diff. entity set.

→ Name
→ Degree
→ Cardinality / participation constraints.

Mapping / Cardinality: at most participation.

1:1 → at most 1. Person has Aadhar
1:N → Person has Mobile No.
N:M → ~~Teacher~~ Teacher teaches students.

1 many



PAPERWELL

Date :

Page no.

Relation Algebra

Basic fundamental operators :

- σ (Select)
- π (Project)
- $\cup$ (Union)
- $-$ (set diff.)
- $\times$ (cartesian product)
- ρ (Rename)

Derived :

- Natural join ($\bowtie$)
- Intersection ($\cap$)
- Division ($\div$)

* Select (σ) Select some rows from all rows.
* Projection (π) Select column from all columns.

$$\sigma_{\text{condition / predicate}} (\text{table\_name})$$

Min selected tuple can be 0.
Max " " " all rows.

$$\pi_{\text{column\_name}} (\text{table name})$$

First select than project, if both have to be done.

Set difference (-) and set union (∪) are same as in sets with constraints that they have same set of attributes.

During cartesian product many redundant tuples are generated. These are filtered further by using σ with some condn (basically same foreign key are grouped together)
Alternatively, we can use Natural join (⋈) operator to do so.

Natural join (⋈) discards tuples which don't have same tuple-value in other table (i.e) if that tuple value is present in one table its rejected. Hence loss in data.
Eg:

R_1

A	B
1	a
2	b

R_2

B	C
b	p
c	q

$R_1 \times R_2$

A	B.R_1	B.R_2	C
1	a	b	p
1	a	c	q
2	b	b	p
2	b	c	q

$R_1 \bowtie R_2$

A	B	C
2	b	p

datalos

rest values are rejected

PAPERWELL
Date :
Page no.

Inner join: Suppose there are 2 attributes matching in two column's. We can use Inner join to construct a table with only 1 single attribute as column

Outer Join: → left outer join
Right ——
Complete ——

Eg:

R_1

A	B
1	p
2	q
3	r

R_2

B	C
q	x
r	y
s	z

$R_1 \times R_2$

A	R_1.B	R_2.B	C
1	p	q	x
1	p	r	y
1	p	s	z
2	q	q	x
2	q	r	y
2	q	s	z
3	r	q	x
3	r	r	y
3	r	s	z

$R_1 \bowtie R_2$

A	B	C
2	q	x
3	r	y

Left outer join ⟕

A	B	C
1	p	NULL
2	q	x
3	r	y

Right outer join ⟖

A	B	C
2	q	x
3	r	y
NULL	s	z

Complete attr join ⟗

A	B	C
1	p	NULL
2	q	x
3	r	y
NULL	s	z

Date :
Page no.

→ Rename (ρ) is a unary operator used to rename attributes of a relation

$\rho(a/b)R$ → rename attribute 'b' of rln by 'a'

→ If 'A' has n tuple and 'B' has 'm' tuples then A × B → 'mn' tuples.

A → n attribute A × B → m+n attributes.
B → m attributes

→ If resultant relation has duplicate rows after projection they are removed by default.

→ ÷ operator A ÷ B is applied if and only if
(1) Attribute of B is proper subset of A'.
(Roll no) (2) Relation will contains tuples from (Att. 'A' - Att B)
attribute
(3) Relation will return those tuples from r/n A which are associated with every B's tuple.

Eg

A

Roll No	Sports
1	x
2	x
2	y
3	y

B

Sports
x
y

A ÷ B → Roll no.
2

loan no for each loan of amount > 1500

(tuple)

- Relational calculus:

(TRC)

$$\{ t \mid P(t) \}$$

It is a set of all tuples t such that Predicate (P) is true for t.

$$\{ t \mid t \in \text{student} \wedge t[\text{age}] \geq 15 \}$$

↳ gives all attributes of student table who has age ≥ 15

Eg:

Table: loan (loan no, branch name, amount)

Find branch name, loan no, amount for loans > ₹1500

Only selection

$$= \{ t \mid t \in \text{loans} \wedge t[\text{amount}] > 1500 \}$$

Find loan no. for each loan of amount greater than 1500

projection too

$$\{ t \mid \exists\ L \in \text{loan} \ (t[\text{loan\_no}] = L[\text{loan\_no}] \wedge L[\text{amount}] > 1500) \}$$

$\pi_{name} \left| \sigma_{\substack{cust\_no \\ \wedge\, branch\_name}} (loan \times borr) \right|$

PAPERWELL
Date :
Page no.

Table :

(1) branch (branch_name, city, assets)
(2) customer (cust_name, cust_street, cus_city)
(3) loan (loan_no, branch_name, amount)
(4) borrow (custom_name, loan_no)
(5) account (account_numbers, branch_name, balance)
(6) Depositor (cust_name, cus_no)

(4) Find names of all customer who have loan from Pune branch.

~~$\{ t \mid b \in borrower \; t[cust\_name] = b[cus\_name]$~~

$l \in loan \; (l[loan\_no] =$

$\{ t \mid \exists\, b \in borrower \; (t[cust\_name] = b[cust\_name]$

$\wedge \exists\, l \in loan \; (l[loan\_no] = b[loan\_no]$

$\wedge l[branch\_name] = 'Pune'\;))\}$

(5) Find all customers names, who have loan, account, or both at bank.

$\{ t \mid \exists\, b \in borrower \; (t[cust\_name] = b[cust\_name])$

$\vee \exists\, d \in depositor \; (t[cust\_name] = d[cust\_name])\}$

Date:
Page no.

Domain Relational calculus:

$$\{ \langle x_1, x_2, x_3, x_4 \ldots x_n \rangle \mid P(x_1, x_2 \ldots x_n) \}$$

↓ set of all domains such that these cond^n are true.

rollno, name, age

$$\{ \langle r, n, a \rangle \mid \langle r, n, a \rangle \in \text{student} \wedge a \geq 15 \}$$

Query Find loan No, branch name, and amount for loans >1500

~~$\{ \langle \text{loan\_no}, \text{branch\_name}, \text{amount} \rangle \mid \in \text{loan} \}$~~
(l, b, a)

$$\{ \langle l, b, a \rangle \mid \exists \langle l, b, a \rangle \in \text{loan} \wedge a > 1500 \}$$

Query Find all loan numbers for loan with amount >1500

$$\{ \langle l \rangle \mid \exists\, b, a \ \langle l, b, a \rangle \in \text{loan} \wedge a > 1500 \}$$

l → output domain

Query Find name of all customers who have loan from Pune branch and find the loan amount.

$$\{ \langle c, a \rangle \mid \exists\, l \ (\langle c, l \rangle \in \text{borrower} \wedge \exists\, b \ (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{"Pune"})) \}$$

Date:
Page no.

Query: Find the names of all customers who had a loan, an account or both at Pune branch.

$\{ \langle c \rangle \mid \exists\, l \,(\langle c, l \rangle \in borrower) \wedge \exists\, b, a\, (\langle l, b, a \rangle \in loan \wedge b = \text{"Pune"}) \vee \exists\, an\, (\langle c, an \rangle \in depositor) \wedge \exists\, b, bb\, (\langle an, b, bb \rangle \in account \wedge b = \text{"Pune"}) \}$

Bam

Anomalies in Database:

Redundancy: When same data is stored multiple times unnecessarily in database.

Disadv of redun: (i) Insertion / deletion / updation anomalies.
(ii) Inconsistency of data
(iii) Increase in db size and increase in time.

Insertion anomali: When certain data (attribute), can't be inserted in DB, without the presence of other data.

Deletion anomali: if we delete some data (unwanted), it causes deletion in other data.

Updation / Modif": If we want to update single piece of data, but it must be done at all its copies

Page no.

Normalization is decomposition of tables.
on basis of functional dependency until each table contain one idea.

* 1NF → Every cell should contain atomic value.

eg:

Rollno	Name	Course
101	Ash	CN
		OS

1NF →

101	Ash	CN
101	Ash	OS

* • Every cell in table contain atomic value
• Every cell in a column have same domain
•

2NF

Consider $R(A,B,C,D)$ such that

$AB \rightarrow D$

$B \rightarrow C$

Then

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

$AB \rightarrow$ candidate key

all attribute which are part of any primary key.

$A,B \in$ prime attribute

$CD \in$ non-prime attribute

So $B \rightarrow C$ C is depending only on B not on A since both A and B makes a primary key.

Date :
Page no.

So such kind of dependency
$B \rightarrow C$
is a partial dependancy

When a non-prime attribute instead of depending on whole candidate key, it is depending on partial candidate key ⟶ partial dependancy.

A table/rln is said to be in 2NF if its in 1NF and doesn't have partial dependancy.

How to translate in 2NF

eg: $R(A, B, C, D)$
$AB \rightarrow D$
$B \rightarrow C$ → // Partial dependancy.

convert →
$R_1(A, B, D)$ $R_2(B\ C)$

Here problem is with C if B is null we cant identify C as both AB contribute to primary key!

Eg $R(ABC)$
$B \rightarrow C$
CK: AB

$R_1(AB)$

A	B
a	1
b	2
a	3
c	3
d	3
e	3

$R_2(B, C)$

B	C
1	x
2	y
3	z

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

Redundancy

Page no.

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

AB → C // PD

D → E // PD

CK: A B D.

- R_1 (A B C)
- R_2 (D E)
- R_3 (A B D)

(2) R(A B C D E) ← Transitive dependency

A → B // PD

B → E

C → D // PD

CK = A C

- R_1 (A, B, E) ← Transitive but we are only seeing Partial Dep.
- R_2 (C D)
- R_3 (A C)

(3) R(A B C D E F G H I J)

PD AB → C

PD AD → GH

PD BD → EF

PD A → I

PD H → J

CK = A B D

- R_1 (A, B, C)
- R_2 (A, D, G, H, I) // Transitive Dependency
- R_3 (B, D, E, F)
- R_4 (A, I)
- R_5 (A, B, D)

~~R_1 (A, B)~~

~~R_2 (A, D)~~

Date :
Page no.

3NF

Transitive Dependency: When one non-prime start finding other non prime attributes.

∴ A F.D from $\alpha \to B$ is T.D. if $\alpha, B \in$ non prime attributes

A rln is 3NF if its (i) 2NF
(ii) not T.D

Eg: R (A B C)

$A \to B$

$B \to C$ ↙ TD.

R_1 (B C)

R_2 (A B)

Other def: Table in 3NF if from $\forall\ \alpha \to B$
either α is super key.
or B is a prime attribute.

Decomposition in 3NF.

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

× $A \to B$

$B \to E$

× $C \to D$

CL = AC

R_1 (A B E)

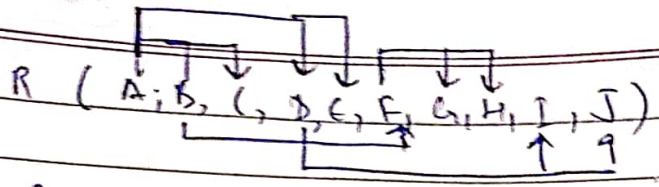
R_{11} — (A B)

R_{12} (B E)

R_2 (C D)

R_3 (A C)

Page no.



* A B → C

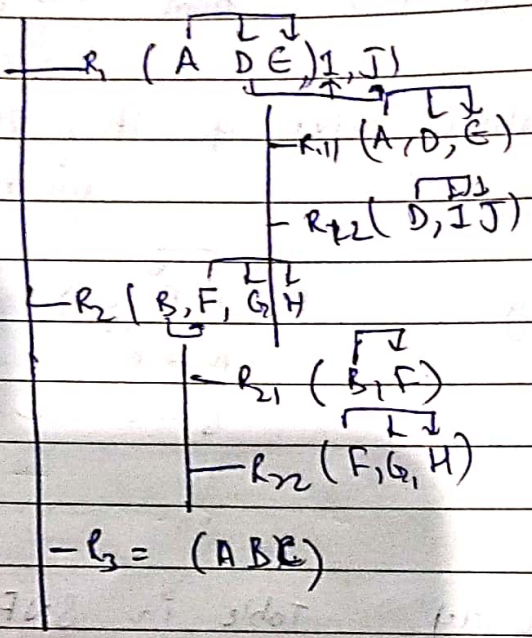
✕ A → DE

✕ B → F

F → GH

D → IJ

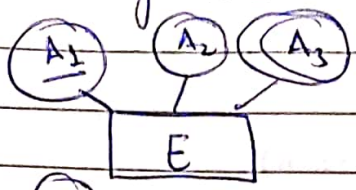
CK = AB



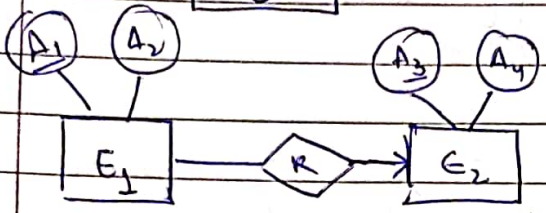
rep

PAPERWELL
Date :
Page no.

Forming schemas from ER - diagram.



→ $E_1(A_1, A_2)$, $E_2(A_1, A_3)$



Entities dependency | Cardinlity E1 : E2 | schema

× | N : 1

$E_1(\underline{A_1}, A_2)$

$E_2(\underline{A_3}, A_4)$

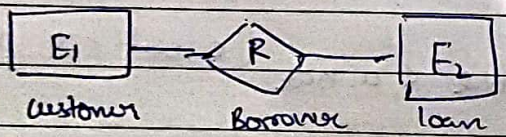
$E_3(\underline{A_1}, \underline{A_3})$

✓ | N : 1

$E_1 \cup R$, FK

$E_1(\underline{A_1}, A_2, A_3)$

$E_2(\underline{A_3}, A_4)$



✓/× | N : M

$E_1(\underline{A_1}, A_2)$

$E_2(\underline{A_3}, A_4)$

$R(\underline{A_1}, \underline{A_3})$

× | 1 : 1

$E_1(\underline{A_1}, A_2)$

$E_2(\underline{A_1}, A_4)$

$R(\underline{A_1}, \underline{A_3})$

✓ | 1 : 1

$E_1 \cup R \cup E_2$

$E_1(\underline{A_1}, A_2, \underline{A_3}, A_4)$

loan (lno, bname, amt)

Rename operation (ρ)

$\rho_x(E) \rightarrow \rho_L(loan)$

$\rho_{x(A_1, A_2, A_3 \ldots A_n)}(E) = \rho_{L(l, b, a)}(loan)$

renaming attributes.

Find out the maximum account balance in bank.

$\pi_{acc\,bal} - \pi_{acc\,bal}(\sigma_{acc.bal \leq a.bal}(account \times \rho_a account))$

BCNF

2NF :→ $\alpha \rightarrow B$.

prime ✗ NP

3NF np ✗ np.

PAPERWELL

Date :

Page no.

BCNF works

$$\alpha \longrightarrow \beta$$

P/NP P

Example

R (A B C)

AB → C

C → B.

CK : AB , AC

AB → C in 2NF , 3NF

C → B

It doesn't have P.D ($p \rightarrow np$) and T.D ($np \rightarrow np$)

But there is still a problem. its not in BCNF

$$\alpha \longrightarrow \beta$$

P/np P

ideally prime attributes derive someone and aren't derived by someone else.

* If table is in BCNF

then $\alpha \longrightarrow \beta$

↑

super key.

if α is not a super key, table is not in BCNF

Date :
Page no.

→ Select marks from student where marks between 100 and 150

≥ 100 ≤ ≥ 150

NOT BETWEEN 100 > < 150

→ Query 1 union Query 2.
↳ eliminated duplicates

Query 1 union all Query 2.
↳ no duplicates.

→ Select cust_name Cartesian Product.
From Account, Depositor
where b_name = "Newdelhi" AND Account.A_no
= Depositor.A_no
Foreign key

→ Natural join
Select cust_name
From Account natural join Depositor.
where b_name = "N.D";

→ Inner join / join: (Multiple foreign key)

$R_1(A, B, C)$ $R_2(B, C, D)$

Select A
From R_1 join R_2 using B
↓
only gp using $R_1.B = R_2.B$
not $R_1.C = R_2.C$

On

Select Cname
From loan join Borrower on L.Lno = B.Lno
and Amount >1000

Rename (as)

Select A_no, (balance *100) as New_Column_name
from Account
where balance <1000

maximum loan amount

loan A

amount
10
20
30

loan B

amount
10
20
30

```
10   10
10   20 ✓
20   30 ✓
20   10
20   20
30   10
30   20
33   30
```

(Select amount
From loan
minus
Select
From loan as A, loan as B.
where A.amount < B.amount)

→ 20,30

→ String operation

% → substring

its not case sensitive

like → % Kumar → suffix
% kumar % → substring
kumar % → prefix

like - - - -

where cname like ' ' ;

#1 order by field (dept, 'B Tech', P.HD, M.Tech')

~~dept~~ (

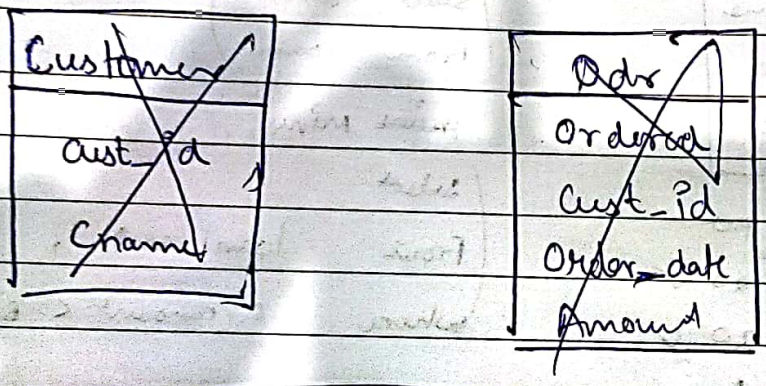
random order.

select count (*) from student → 9

group by branch;

CSE → 2 PHD — 0

IT — 6



Loseless join

It is loseless iff.

(1) $att(R_1) \cup att(R_2) = att(R)$

(2) $att(R_1) \cap att(R_2) \neq \phi$

(3) $att(R_1) \cap att(R_2) \rightarrow att(R_1)$
$\rightarrow att(R_2)$

common attribute should have unique value.

or hum prove karde that common attribute is candidate key for any one table.

BCNF

R(A B C)

$(AC)^+$

$(AB)^+$

$AB \rightarrow C$

$C \rightarrow B$

Table in 2NF and 3NF but still a problem not in BCNF

$\alpha \rightarrow \beta$

p/np p

If table in BCNF

$\alpha \rightarrow \beta$

super key

identification:

~~BCNF~~ $\alpha \rightarrow B$ table in BCNF

super key

$\alpha \rightarrow B$ table in 3NF

↑ ↑

SK/CK prime

If, then check 2NF

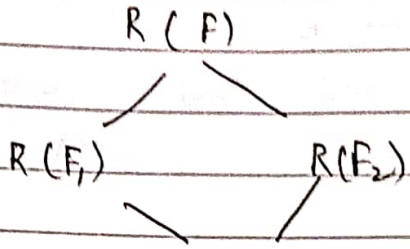
3NF = np → np

2NF = p → np

Dependency Preserving:

Def:

A relation is dependency preserving if



if this is its dependency preserving.

i.e $F_1 \leq F^+$

$F_2 \leq F^+$ then

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

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

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

$R_1(A, B)$	$R_2(B, C)$
F_1: $A \to B$	F_2: $B \to C$
$B \to A$	$C \to B$

$(F_1 \cup F_2)^+$ = $A \to B$

$B \to C$

$C^+ = C, B, A$

So $C \to A$

So

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

R(A, B, C, D)

AB → CD

D → A

$R_1(A, D)$	$R_2(B, C, D)$
F_1 : D → A	F_2 : ~~B → B~~
	~~B → B~~
	B → B ✗
	C → C ✗
	D → A ✗
	BC → BC
	CD → CD ✗
D → A	BD → BD ✓

$(F_1 \cup F_2)^+ \rightarrow D \rightarrow A$

$(AB)^+ = (A, B) \neq F$

Not dependency preserving.

R(A B C D E F G H I J)

Table in INF

AB is CK

R_1 (A, B C)

R_2 (A D E I J)

R_3 (B F G H)

R_{21} (A D E)

R_{22} (D I J)

R_{31} (B F)

R_{32} (F, G H)

2NF

(3 Tables)

(3NF)

5 Tables

~~Between~~

- Between (≤ ≥) · NOT BETWEEN (≥ ≤)
- Union, Cross join → table1 , table2.
  Natural join
- On → write condn in From
- 'as → To rename ( ) as ( )
- % suffix , % substring % , prefix (like) ← where
- IN | NOT IN → filter data in list (value1, 2, 3 . . 3. in (1, 3)
- LIMIT → no of rows to be selected.
  OFFset → starting index.
- ORDER BY → order by coln name asc | desc.
- n^{th} highest

  Select * from student where marks = (select DISTINCT (marks) from student order by marks desc limit n-1, 1);
- Order by sname desc , mark asc.
- order by field (Field name , 'order1' , 'order2') .
- Group by

Page no.

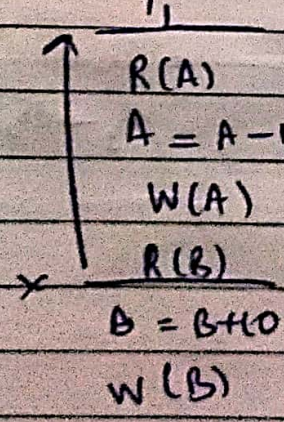
Transaction Management

Topics

- → ACID Properties
- → transaction states
- → problem with concurrency
- → conflict and view serializablity
- → Recovery and cascadelessness.

- → Transation designing protocol.
- → time stamping.
- → 2- phase locking.
- → Graph based protocols
- → Multiple ~~granularity~~ granularity.

Transcation: Atomic (✓ or ✗) set of instructions.



Either task are performed completely or not performed totally.

No concept of partial process.

Date :
Page no.

ACID Properties:

→ Atomicity (Either trans' are executed completely / or total not)
→ Consistency
→ Isolation (One Transaction is independent of other transaction)
→ Durability (Transaction should persist after its executed)

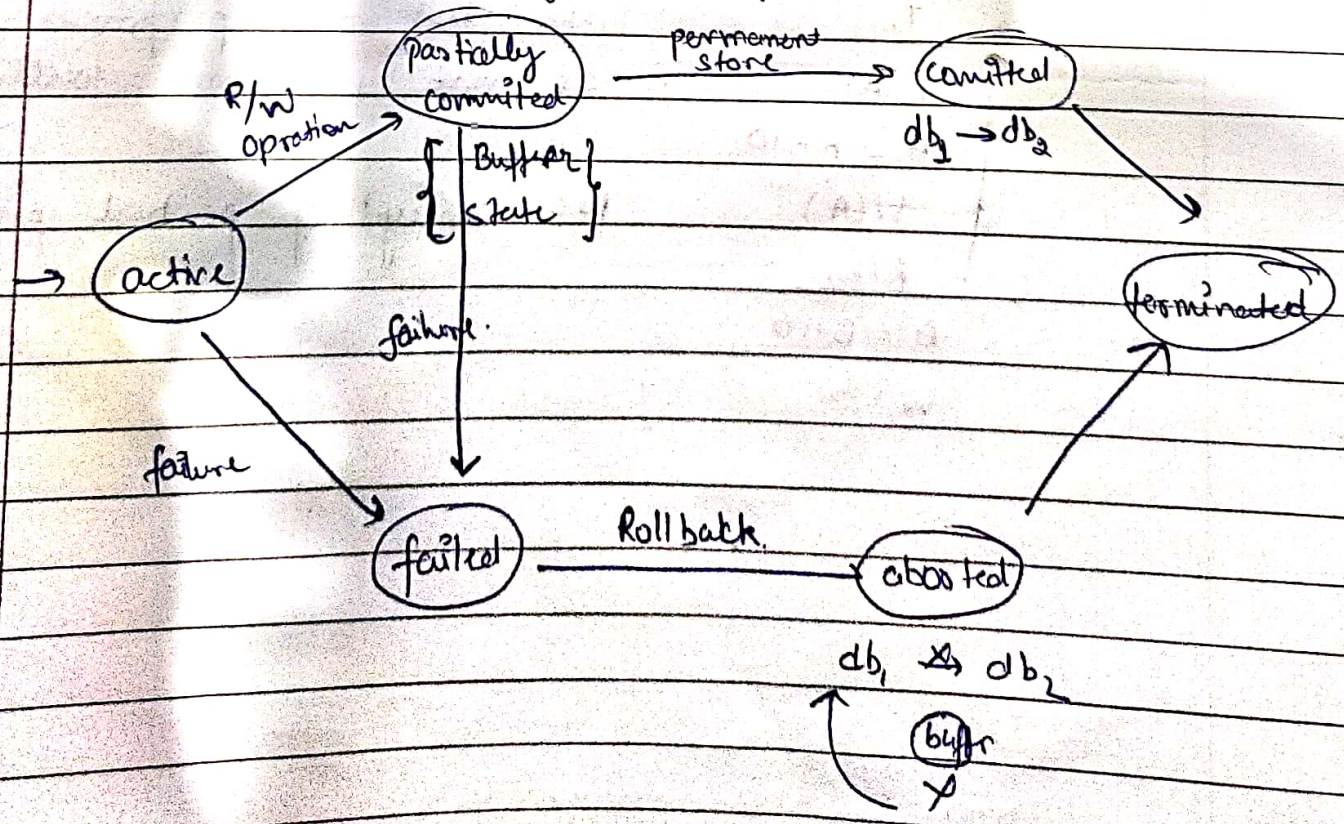
→ Transaction Managment component in dbms which ensures Atomicity

→ Concurency control component in dbms ensure Isolation.

→ Recovery Managment component ensures Durability.

Transaction State:

Transaction is done in buffer memory, and if it done completely changes are made database.



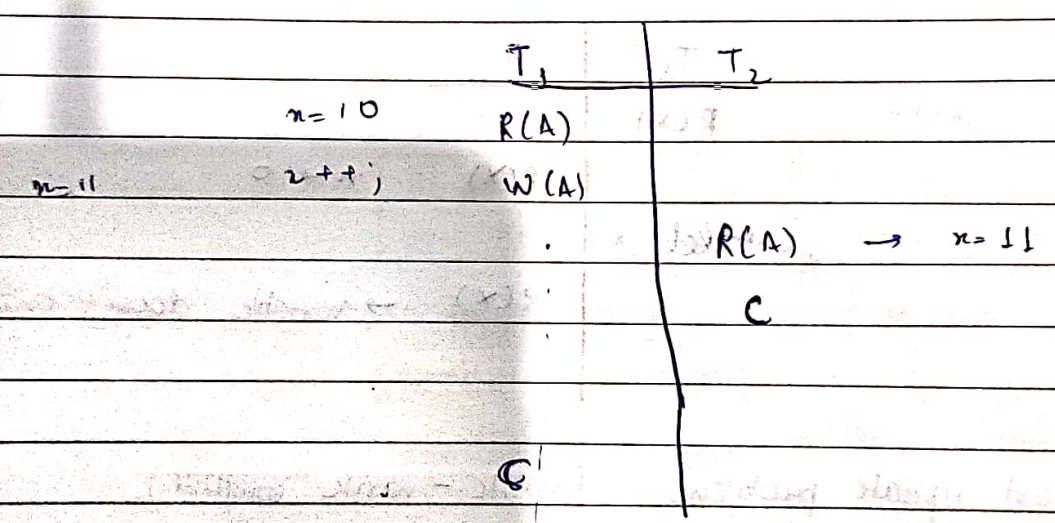
Date :
Page no.

* Advantage of concurrency:
  - When many users access / make transaction at same time.

And
  - Waiting time ↓
  - Respone time ↓
  - Resource utilization ↑
  - Efficiency ↑

But concurrency can lead to inconsistency

Dirty read Problem:



If transaction reads the value from an uncommited transaction. (i.e local buffer). If transaction ends with failure T_1 rolls back but T_2 can't as its committed.

There is always not problem, but their is possibility

→ soln to this is T_2 should commit after T_1 has commited.

Unrepeatable read problem:

	T_1	T_2	
$x=10$	R(X)		
		R(X)	$x \to 10$
$x++$	W(X)		
		R(X)	$x \to 11$

Phantom read Problem:

	T_1	T_2	
$x=10$	R(X)		
		R(X)	$x \to 10$
	Delete(X)		
		R(X)	⇒ variable doesn't exist.

Lost update problem (write - write conflict).

		T_1	T_2	
$x=10$	10	R(A)		
$x=11$	$x++$	W(A)		
$x=50$			W(A)	(blind write) $x=50$
			C	
		C		

A is updated to 50 instead of 11.

Date :
Page no.

Schedule:

Serial schedule.

T_1	T_2
R(A)	R(B)
W(A)	W(B)
R(B)	R(A)
W(B)	W(A)
n_1	n_2

$n_1 + n_2$

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	

Non serial schedule.

T_1	T_2
R(B)	
W(B)	
R(A)	
W(A)	
	R(A)
	W(A)

T_1	T_2
R(A)	
W(A)	
	R(B)
R(B)	
	W(B)
	R(A)
	W(A)
W(B)	

For n transaction, $T_1, T_2 \ldots T_n$

Example Q

$1 \ldots n$: $n_1, n_2 \ldots n_m$

$n \times (n-1) \times (n-2) \cdots 1$

n! Serial schedule are possible	$\frac{(n_1 + n_2 \cdots n_n)!}{n_1!\, n_2! \cdots n_n!} - n!$
→ no concurrency	→ concurrency.
→ guaranteed consistency	→ May be unsafe.

If we prove a non-serial schedule equal to serial schedule, then, non-serial schedule is also consistent.

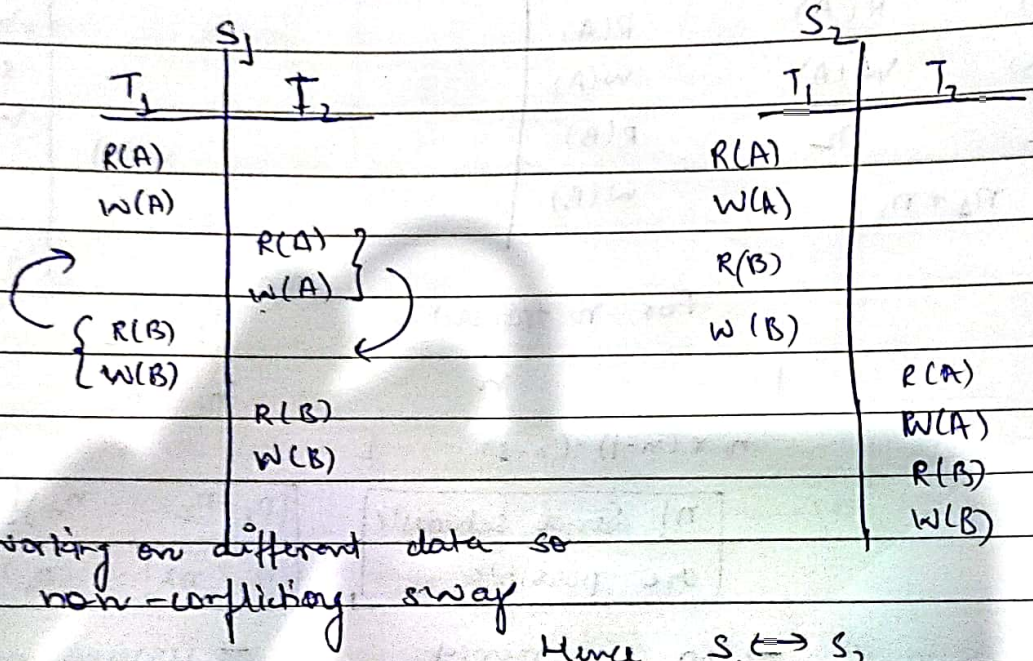
Conflict serializibility:

Swapping of non conflicting instruction.

If swapping doesn't affect any of Transaction then, its non conflicting swapping

If instruction work on same data$, then their is possibility of confliction.

If same data and one is write instruction → conflicting.



working on different data so non-conflicting sway

Hence $S_1 \leftrightarrow S_2$

S_1 is consistent

Then S_1 is called conflict serializable.

→ Converting of non-serial into serial schedule.
By swapping of non-conflicting, then the schedule is called conflict serializable and is consistent.

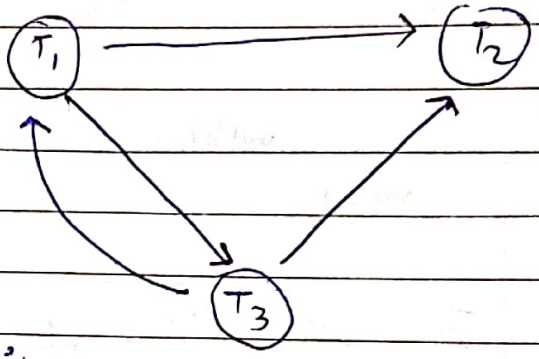
#

→ different transaction
→ same data value.
→ at least one of them should be write.

PAPERWELL
Date :
Page no.

Practice question:

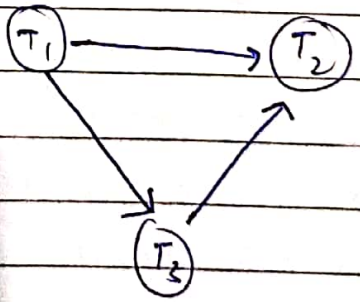
S		
T_1	T_2	T_3
R(x)		
		R(z)
		W(z)
	R(y)	
R(y)		
	W(y)	
		W(x)
	W(z)	
W(x)		



if cycle we cant convert.
so this schedule is not conflict serializable.

(2)

S		
T_1	T_2	T_3
R(x)		
	R(y)	
		R(y)
	W(y)	
W(x)		
		W(x)
	R(x)	
	W(x)	



Hence it is conflict serializable.

Order: $T_1 \rightarrow T_3 \rightarrow T_2$

T_1: no incoming edge
T_2: no outgoing edge.

(4)

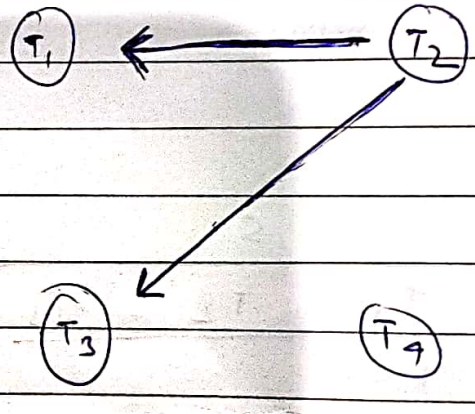
(3) S

T_1	T_2	T_3
R(a)		
	R(b)	
		R(c)
		W(c)
	W(b)	
W(a)		

conflict serialization
Order can be anything
so 3! possible schedule

(3)

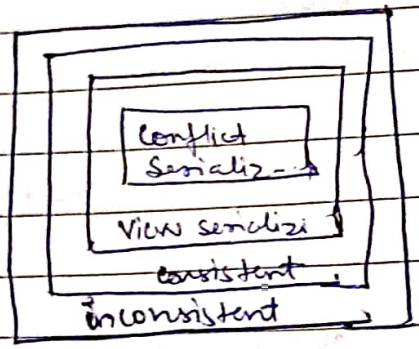
T_1	T_2	T_3	T_4
	R(x)		
		W(x)	
		C	
W(x)			
C			
	W(Y)		
	R (z)		
	C		R(X)
			R(Y)
			C

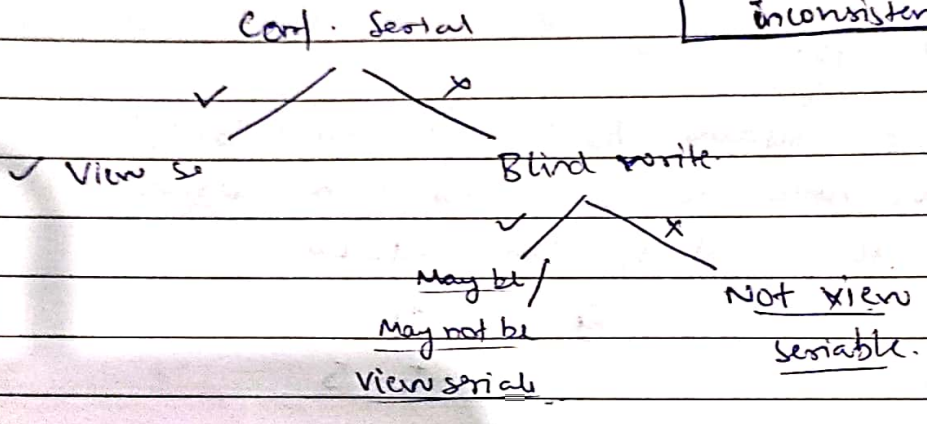


conflicting serializable.

View serializibility:

If schedule is view se... and is not conflict seri... then it must have atleast





View equivalence:

(1) Initial read for every ~~~~ data value.
same Transaction should read data value in both schedule.

(2) Final write

(3) Intermediate read. → if any transaction T_i is reading value of other Transaction, then same should happen in serial schedule.

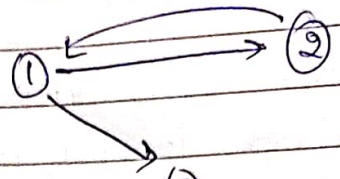
schedule
Then ~~table is~~ in view equivalent.
and table is view serializable.

We have to check with every n! possible serial schedule, if its is in view equivalent with any one its okay.

Ques

	S		
	T_1	T_2	T_3
(1)	R(a)		
(2)	→	W(a)	
(3)	W(a)		
(4)		W(a)	

First check for conflict



Not conflict serializable.

Check for view.

Yes, it contains blind write in (T_2, T_3)
So we have to check for view serializable
Check for all 3! serial schedul.

(1) S_1

T_1	T_2	T_3
	R(A)	
	W(A)	
		W(A)
		W(A)

(1) Initial read.
$S \to (T_1)$ $(S_1 \to T_1)$

(2) Final write.
$S \to (T_3)$ $(S_1 \to T_3)$

(3) ✓✓

Hence S and S_1 are view equivalent.
Hence S is view serizable schedule.

(#) Recoverable schedule:

Order of last read = Order of commit

Last read → $T_1 \to T_2 \to T_3$

T_1	T_2	T_3
R(A)		
W(A)		
	R(A)	
	W(A)	
	C	
C		
		R(A)

Now recoverable schedule

Date :
Page no.

Cascadless schedule:

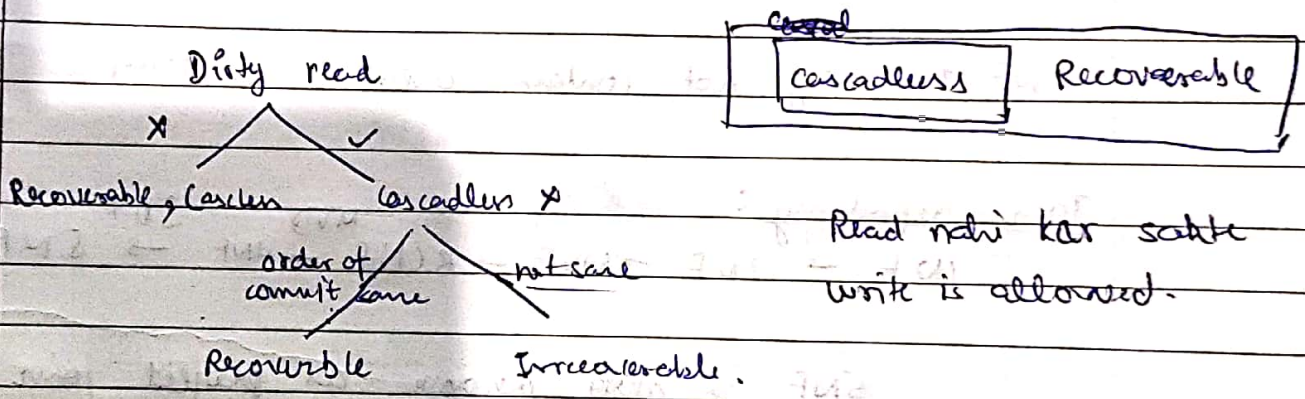
if other Transaction have to roll back due to 1 (due to dirty read) ~~that~~

If schedule doesn't have it, its called cascadless schedule

If their is no dirty read its cascadless schedule.

→ Prime aim is to remove dirty read.

Cascadless is optional.

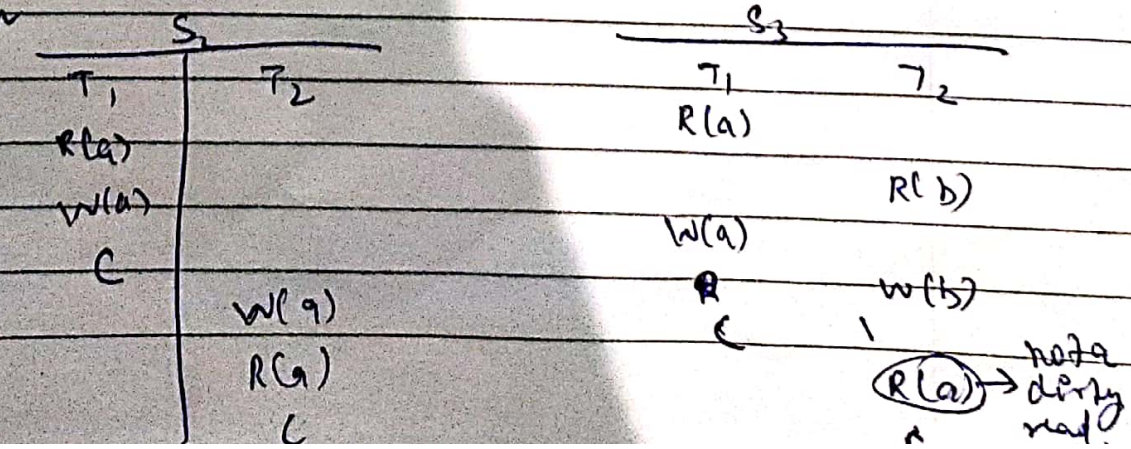


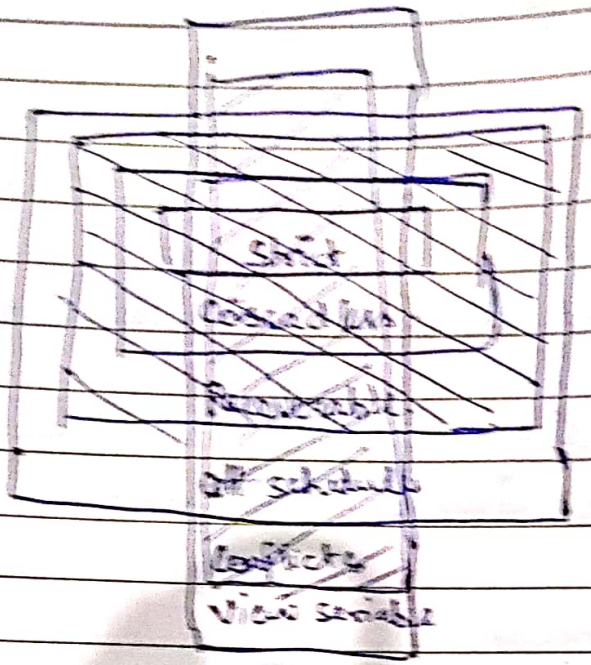
Strict schedule:

Unless one schedule.

Jab tak ek ~~schedule~~ transaction ek data item pe kaam kar raha hai and commit nahi kaste, other transaction can neither read or write on that data value.

write operation perform start karde





4NF → may not contain over 1 multivalues.

Join dependency:

1NF → 2NF → 3NF → BCNF → 4NF → 5NF
(4NF and JD)

5NF is also known as project join form.

1NF →
2NF → P.D ✗
↳ supprime → nonprime
on attribute
3NF → PD✗ TD✗
→ np → np
BCNF → $\alpha \to \beta$
↑
Super key
3NF → $\alpha \to \beta$
↑ super k ↑ prime

Lossless join
(i) $att(R_1) \cup att(R_2) = att(R)$
(ii) $att(R_1) \cap att(R_2) \neq \phi$
(iii) a common attribute should exist between both tables and that common attribute must be candidate key in one table

→ Dependency preserving → on FD

R(A)
$R_1(F_1)$ $R_2(F_2)$
$R(F_1 \cup F_2) = F$

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

lossless is mandatory
FD preserving is optional.

Transactions

Atomicity → Transaction Manager
Consistency
Isolation → Concurrency control
Durability → Recovery management.

Conflict Serializable
(1) Different Transactions
(2) same data value
(3) At least one is write opr

View serial
→ Initial read
→ Inter. W → r
→ Final write.

Conflict seri ✓ ✗
View ✓
Blind write ~ ✗
May or may not
Not view

Recoverable (mandatory)
Order of dirty read = order of commit

Strict schedule
if write is performed in one trans other trans can't read or write until first one has committed
Dirty read ✗
Blind write ✗

Cascadeless schedule (optional)

Dirty Read
W(A) → R(A)
✗ Recovery cascading
✓ Cascadeless ✗
order of commit same → Recoverable
order of commit not same → Irrecoverable

Blind write is allowed ✓

PAPERWELL
Date :
Page no.

6.20

Concurrency control Technique:

→ We study protocols for designing schedule that guarantee these properties specially (C.S.)
→ We have to manage transaction trying to access same data at same time.
→ Ways
  → Time-stamping Protocol. (priority orders to instructions before it enters system.)
  → lock-based protocol
    → 2 phase-locking (Basic, Conservative, Strict, Rigorous)
    → Graph based
  → Validation protocol

Time stamping:

- Ti request for Read(Q):

5 10

✗ T.S(Ti) < write T.S(T(Q))

means Ti needs to read value of Q earlier then write T.S., but it came later, so it rolled back, and new Time stam is assign

T.S(Ti) ≥ write (TS(T(n)(Q))

10 5

read time stamp will be

max (RTS(Q) , Ts(Ti))

Ti request for Write(Q)

if TS(Ti) < RTS(Q) roll back.

if TS(Ti) < WTS(Q) roll back

all other cases → Write time stamp will be

max (WTS(Q) , TS(Ti))

Trick: Junior (> time stamp) should work later.

USACO

CP 3 steven
cp Handbook

PAPERWELL
Date :
Page no.

625.

Properties:

T.S
CS

→ It ensures conflict serializability.
→ It ensures view serializability.
→ possibility of directy read & no ~~retraction~~ reduction on commit irrecoverable schedules. & cascading roll backs are possible.

Thomas write Rule:

Modify time-stamping protocol to ~~modify~~ make some improvements and may generate those schedules that are view serializable but not conflict serializable and provides better concurrency.

→ modify time-stamping protocol in obsolete.
write-write case where

$$T.S.(T_i) < WTS(T_x)$$
(write)

→ here T_i attempts to write obsolete value of Q so we can ignore it as later blind write hone hi hai

eg. R(Q) W(Q) Not conflict ser.

W(Q) → W(Q)

even if this is not performed & is changed later, so instead of its roll back (in case of WTS) we ignore the instruction)

So with Thomas write rule & protocol, transaction are view serializable (as bcz of blind write) but not conflict serializable.

PAPERWELL
Date :
Page no.

lock based Protocol:

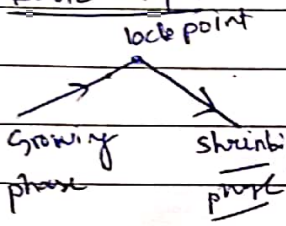
We put a lock, lock is → Shared mode → (lock-S) perform read oper only other trans. can share data item. (read-read).

Exclusive mode: (lock-X) both W/R operation can be performed, but exclusively no other trans. can't access it, even if have to write or read on it.

Alone locking is not perfect as it hampers concurrency/consistency. We need to manage both side-by-side.

→ 2phase locking → Graph based locking.

Basic 2PL



i) Growing phase can only obtain locks.
ii) Shrinking phase can only release locks.
iii) Trans can be perform w/r in both growing/shrinking phase.
iv) C.S/V.S → order is order of lock point.
v) Dont ensure freedom of deadlock.

Conservative 2P: Static 2PL locking:

i) First transaction will acquire all locks, then perform w/r operation
ii) If locks are not available, then it has to relase the locks acquired and wait.
iii) No change in shrinking phase.
iv) Must have know. of all dataitems to be used.
v) prevents deadlock (casualties).
vi) irrecoverable, roll backs.

lock point

growing point

PAPERWELL

Date :

Page no.

Rigorous 2 PL:

locks are held until transaction commits i.e no shrinking phase.

→ ensures C.S, V.S, recoverability, cascadelessness.

Improvement ↓

→ Suffer from deadlock / inefficiency. (concorrency ↓↓↓↓)

Strict 2PL

→ partial shrinking phase

→ In shrinking phase S-unlocking is allowed and X- unlocking is not allowed.

6.31

Graph Based Protocol