INDEX

Sl. No.	Date	Title	Page No.	Teacher's Sign/Remarks
		ER Diagram.		
		Conversion of E-R Diagram into Relational Model.		
		Basics of Relational model & functional dependency.		
		Idea about Keys / Types.		
		Normalization (1NF - BCNF)		
		Lossless decomposition & Dependency Preserving		
		Indexing and Physical Structure (B B^{+} tree)		
		SQL, RA, RC.		
		Transaction		
		Concurrency Control.		

Scanned with CamScanner

Scanned by TapScanner

DEBOJYOTI DAS.
1915001

Orion
PAGE:
DATE: / /

DBMS :-

DATA :-
Raw and isolated facts about an entity (recorded)
e.g text, audio, video, image, map etc

INFORMATION:
Processed, meaningfull, usable data

DATABASE :
Collection of similar / related data.

DBMS :
S/W used to create, manipulate and delete database.

Disadvantages of file System :-
1) Data redundancy.
2) Data inconsistency.
3) Difficulty in accessing data.
4) Data isolation.
5) Security Problem.
6) Atomicity Problem
7) Concurrent - access anomalies.
8) Integrity Problem

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

DBMS :-

OLAP (online analytical processing)	OLTP (online transaction processing)
• historical data.	• current data
• subject oriented.	• application oriented
• decision making	• day to day operations
• TB, PB.	• MB, GB.
• CEO, MD, GM.	• clerks, managers
• R.	• R/W.

Entity Relationship Diagram

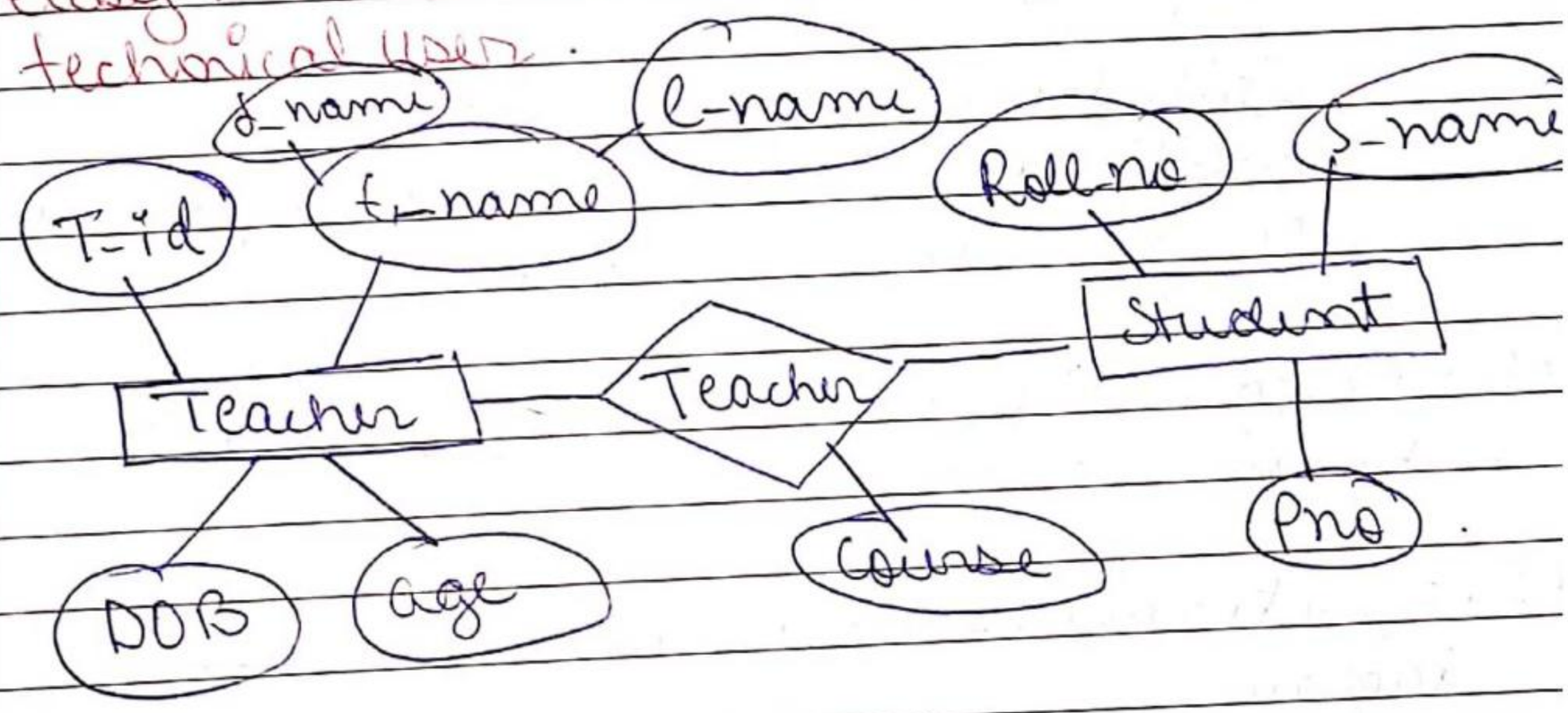
- Introduction and Basics.
- Entity and Entity Set.
- Attributes and Types.
- Relationship
- Strong and weak entity set
- Traps.
- Conclusion.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

- Introduced by Dr. Peter Chen in 1976.
- A non-technical design method works on conceptual level based on the perception of real world.
- Consists of collection of basic objects called entities and of relationships among these objects and attributes which define their properties.
- free from ambiguities and provides a standard and logical way of visualizing data.
- basically it is a diagrammatic representation easy to understand even by non-technical user.



Scanned with CamScanner
Scanned by TapScanner

GROUP → OBJECT

Orion
PAGE:
DATE: / /

Entity :-
An entity is a thing or an object in the real world that is distinguishable from other objects based on the values of the attributes it possess.

Types of Entities :-

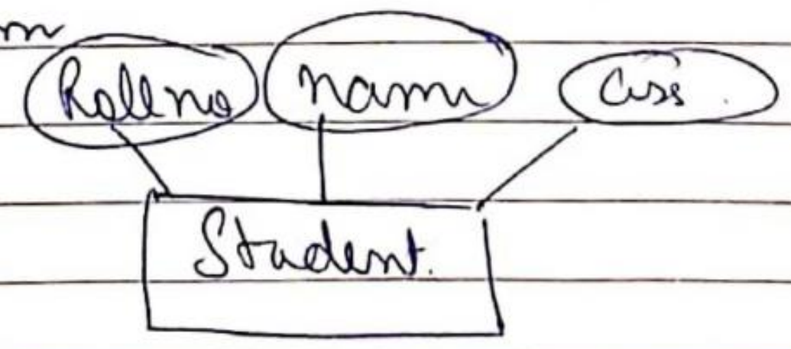
Tangible :
Entities which physically exist in real world.
e.g. car, pen, bank locker.

Intangible :
Entities which exists logically
e.g. account.

Entity Set :-
Collection / set of same types of entities i.e. that share same properties or attributes called entity set.

→ Entity cannot be represented in an ER diagram as it is instance / data.

→ Entity set is represented by rectangle in ER diagram



Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

→ Entity can be represented in an relational model by row / tuple / record

→ Entity set is represented by table in relational model.

Student

Roll no	name	age

→
→
→

Attributes :-
are the units that describe the characteristics of entities.

→ For each attribute there is a set of permitted values called domains.

→ In ER diagram represented by ellipse or oval which in relational model by a seperate column.

Types

Simple - Composite.

- Simple cannot be divided further represented by simple oval.
- Composite can be divided further in simple attribute, oval connected to a oval.

Simple - Multivalued.

- Simple can have only one value at a instance of time
- multivalued → can have more than one value at a instance of time.

Scanned with CamScanner

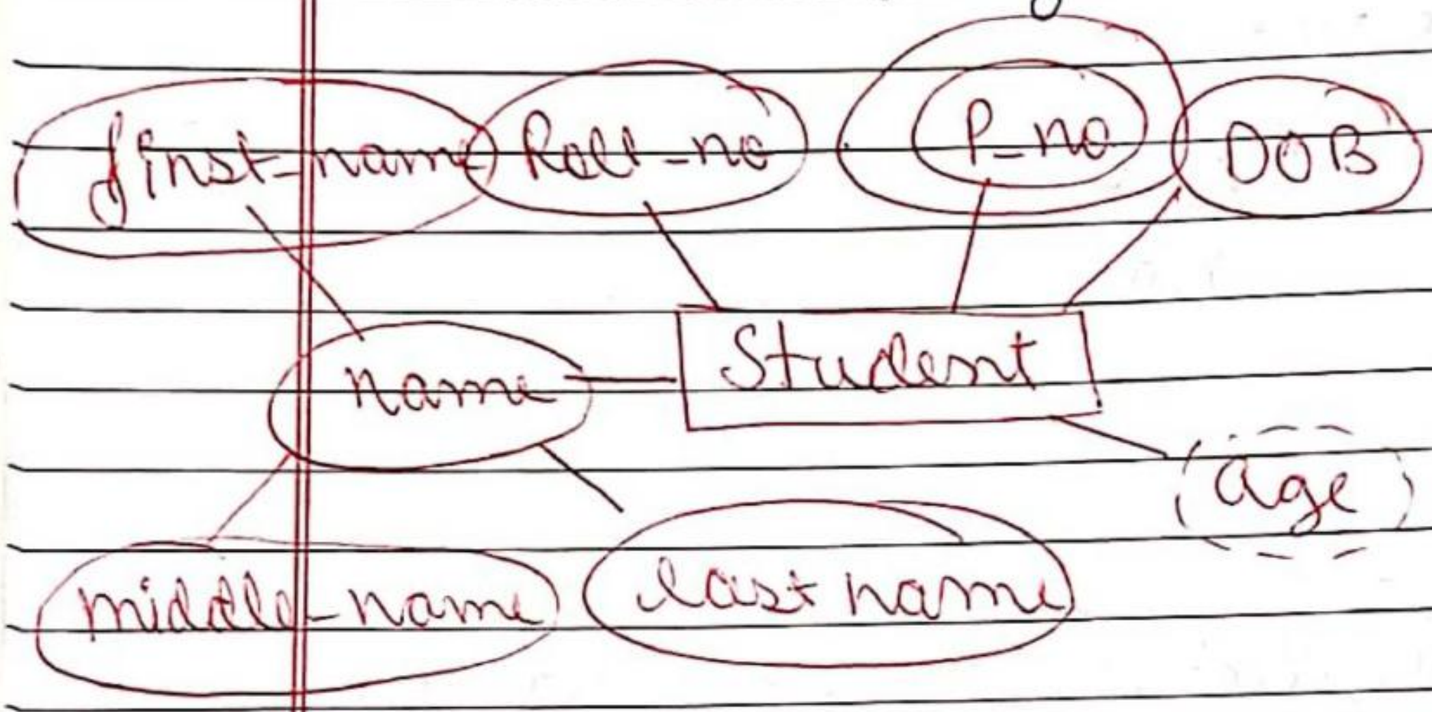
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Stored - derived.

Stored - have value is stored in the data base.

Derived - have value can be completed in run time using stored atributes.



Student

Roll no.	f_name	M_name	L_name	DOB	Age

Roll_no	P_no

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Relationship :-

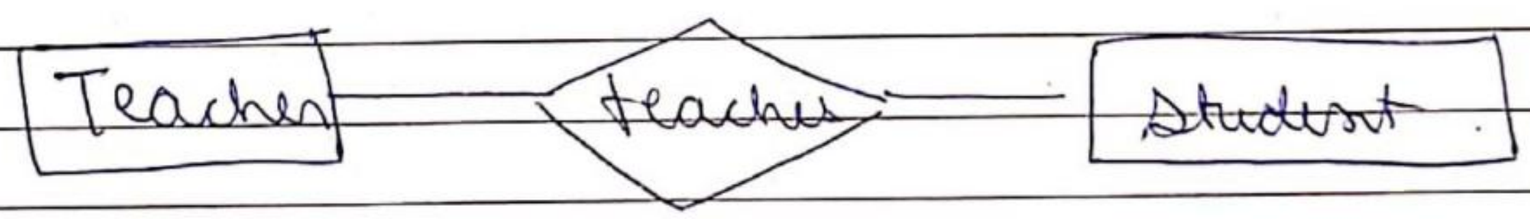
Is an association between two or more entities of same or different entity set.

- No representation in ER diagram as it is an instance or data.
- In relational model represented either using a row in a table.

Relationship type / Set :-

A set of similar type of relationship

- In ER diagram represented using a diamond.
- In relational model either by a seperate table or by seperate column (foreignkey)
- Every relationship type has three components (i) Name (ii) Degree (iii) Cardinality ratio / Participation constraints

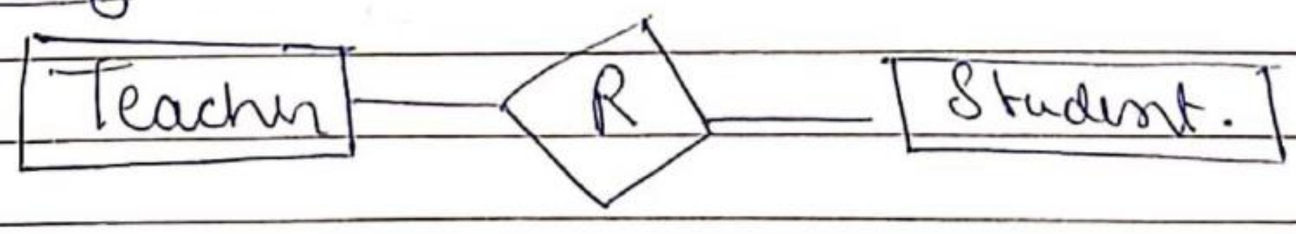


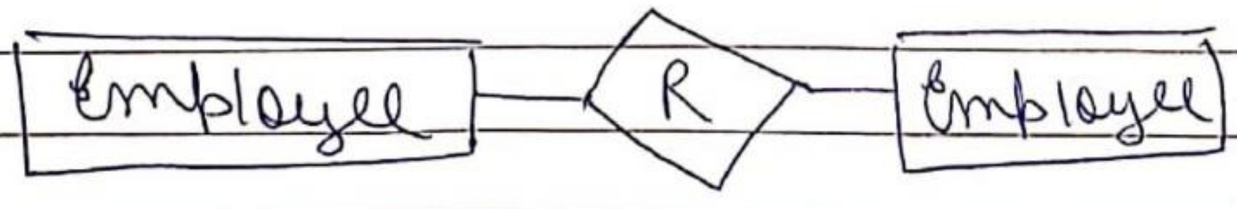
* Degree of a relationship set :-

means number of entity set associated (participated) in the relationship set.

- Most of relationship sets in the ER diagram are binary (in industry). Occasionally however relationship sets involve more than two entity sets.

Binary :-



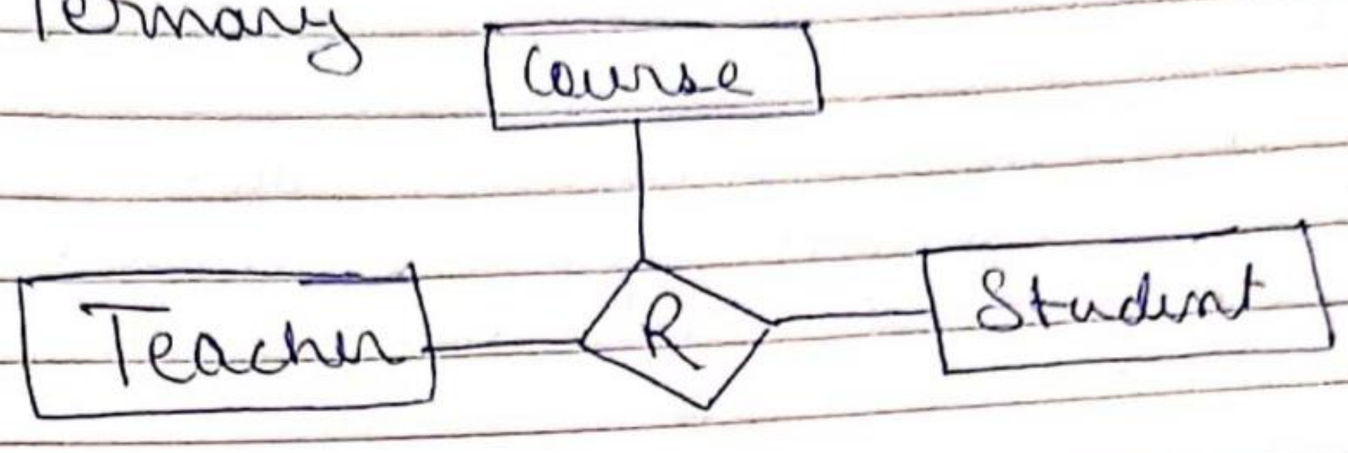


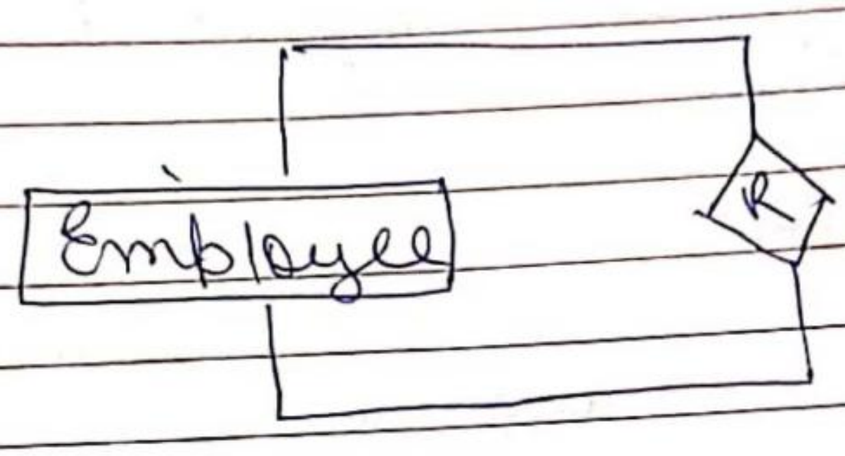
Scanned with CamScanner

Scanned by TapScanner

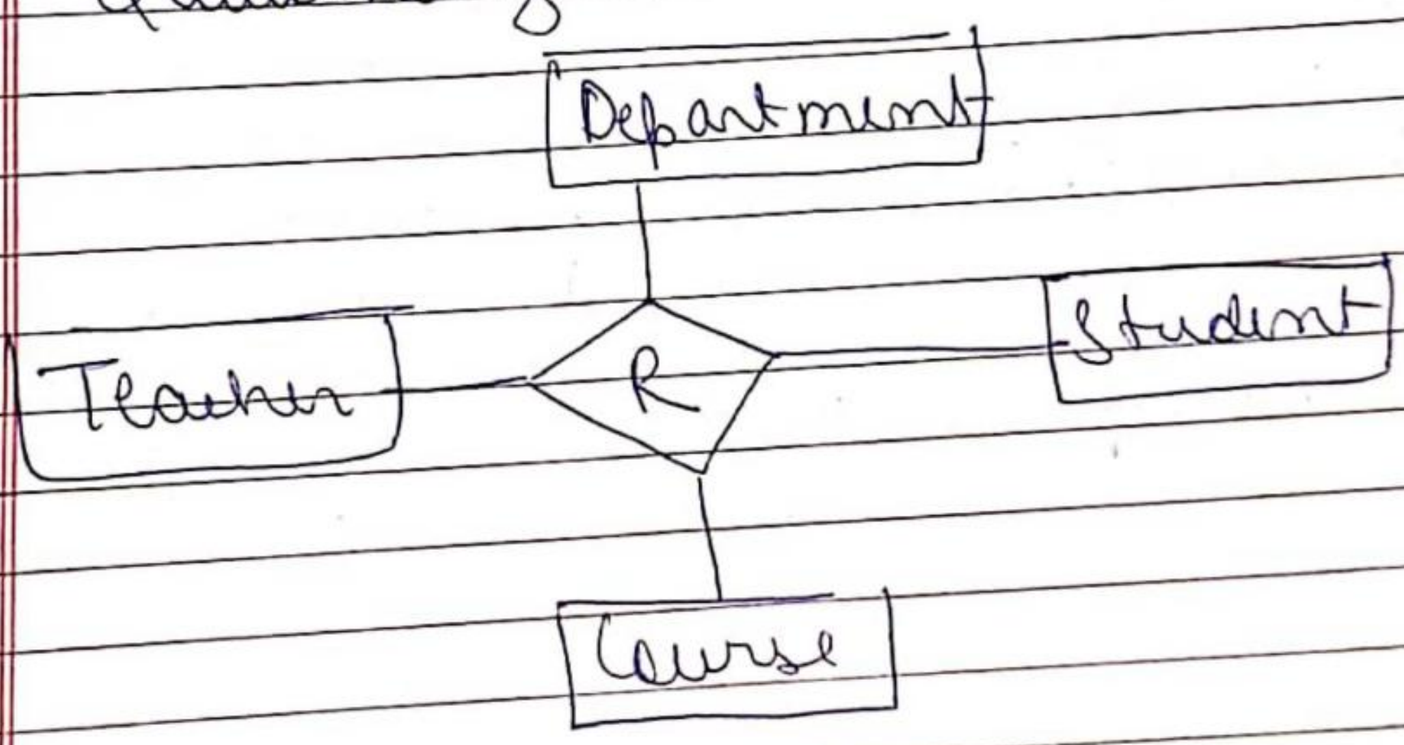
Orion
PAGE:
DATE: / /

Ternary

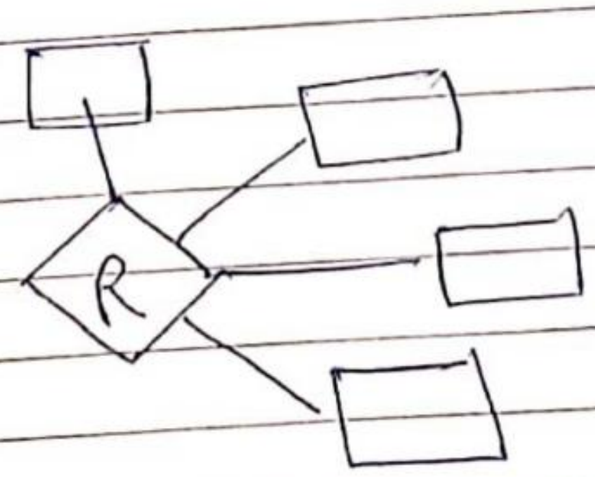




Quaternary



N-ary



Scanned with CamScanner

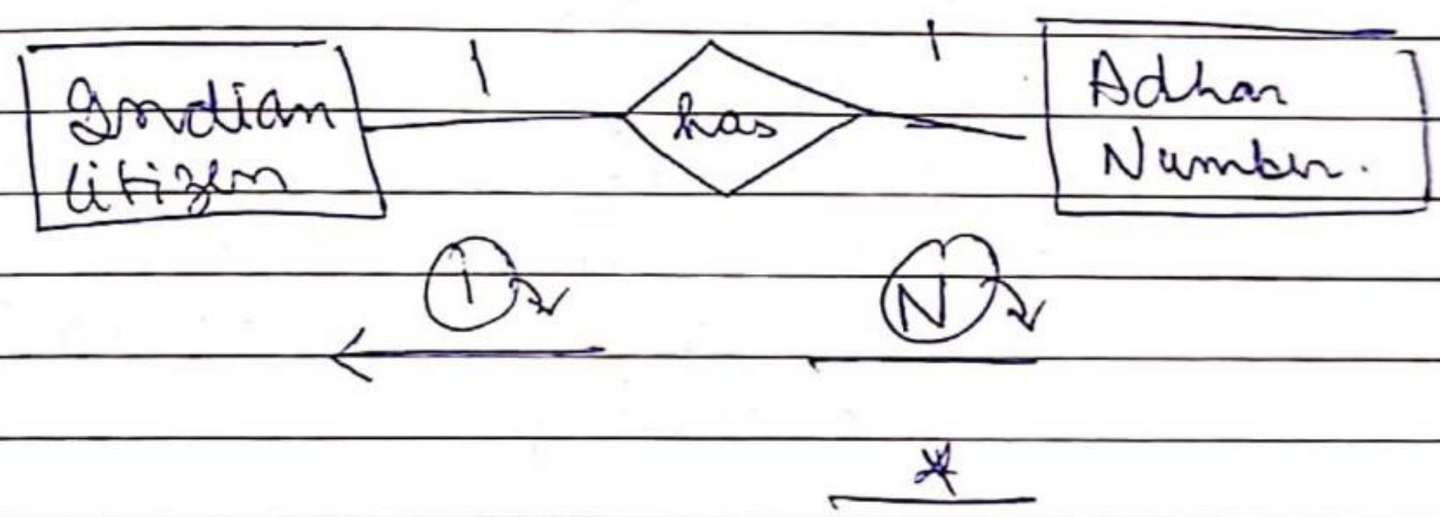
Scanned by TapScanner

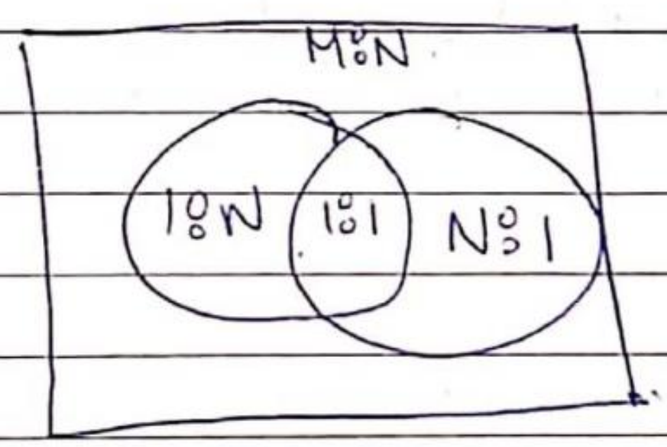
Orion
PAGE:
DATE: / /

Mapping Cardinalities / Cardinality ratio :-
Express the number of entities to which other entity can be related to a relationship.

→ can be used to in describing relationship out of any degree but is most useful in binary relationship.

→ 1:1 (one to one) → Adhar Number.
→ 1:N (one to many) → Mobile Number.
→ N:1 (many to one)
→ M:N (many to many) :→ Student.





Scanned with CamScanner
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Participation constraints:

Specifies whether the existance of an entity depends on its being related to another entity via a relationship type.

These constraints specify minimum and maximum number of relationship instances that each entity can/must participate in.

Max cardinality:

It defines the maximum number of times an entity occurance participating in a relationship.

Min cardinality:

It defines the minimum number of times an entity occurance participating in a relationship.

→ Partial participation.
→ Total participation.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

DBMS

1. Functional Dependencies, Keys.
2. Normalization.
3. Indexing, Physical structures.
4. Transaction.

$f : \alpha \rightarrow \beta$.

	α	β
$t_1 \rightarrow$	a	1
$t_2 \rightarrow$	b	~~2~~ 1
	c	3
	d	4

if we know "α" we can search "β"

For the same value of α we can't get two different values of β.

$f : \alpha \rightarrow \beta$.

$f(x) \longrightarrow y$.

✓1 $\rightarrow$ a

1 $\rightarrow$ b ✗

✓2 $\rightarrow$ a

$\alpha \subseteq R$, $\beta \subseteq R$

$\alpha \rightarrow \beta$

if $t_1[\alpha] = t_2[\alpha]$

$\hookrightarrow t_1[\beta] = t_2[\beta]$

dependent

determinant

R

	α	β
$t_1 \rightarrow$	a	b
$t_2 \rightarrow$	a	b

$\alpha \longrightarrow \beta$

trivial

$AB \rightarrow A$

if $\boxed{\beta \subseteq \alpha}$

non-trivial

$\boxed{\beta \not\subseteq \alpha}$

$AB \rightarrow ABC$

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Functional Dependencies.

a) $A \rightarrow BC$ ✓
b) $DE \rightarrow C$ ✓
c) $C \rightarrow DE$ ✗
d) $BC \rightarrow A$. ✓

R

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6.

$\alpha \rightarrow \beta$
a — 1
a — 1
b — 1

$\alpha \rightarrow \beta$
✗ { a — 1
a — 2.

- check if all "α" are unique
- checking all the values of C are same.

Attribute closure / closure on attribute set / closure of attribute set :-

An attribute set 'A' can be defined as a set of attributes which can be functionally determined from it Denoted by F^+

R(ABC)
$A \rightarrow B$
$B \rightarrow C$.

$A \rightarrow BC$

$(A)^+ = A$
AB
$(A)^+ = ABC$

finitely / infinitely.

R(ABCDEF)
$A \rightarrow B$.
$C \rightarrow DE$.
$AC \rightarrow F$.
$D \rightarrow AF$.
$E \rightarrow CF$.

R(ABCDEF)
✓ $A \rightarrow B$
$C \rightarrow DE$
$AC \rightarrow F$
✓ $D \rightarrow AF$
$E \rightarrow CF$.

$(D)^+ = D$
$= ADF$
$= \boxed{ABDF}$

$(DE)^+ = ABCDEF$

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Armstrong's axiom / rule :-

- Axiom is a statement that is taken to be true and serve as a premise or starting point for further arguments.
- Armstrong axioms holds on very relational database, can be used to generate closure set.

Primary rules (RAT) Secondary rules.

- Reflexivity :
  if $Y \subseteq X$
  then $X \rightarrow Y$

- Augmentation :
  if $X \rightarrow Y$
  then $XZ \rightarrow YZ$.
  (Augmented)

- Transitivity :
  if $X \rightarrow Y$ && $Y \rightarrow Z$
  then $X \rightarrow Y$.

- Union :
  if $X \rightarrow Y$ && $X \rightarrow Z$
  $X \rightarrow YZ$.

- Decomposition :
  if $X \rightarrow YZ$
  then $X \rightarrow Y$ && $X \rightarrow Z$.

- Pseudotransitivity
  if $X \rightarrow Y$ && $WY \rightarrow Z$
  then $WX \rightarrow Z$

- Composition
  if $X \rightarrow Y$ && $Z \rightarrow W$
  $XZ \rightarrow YW$.

$AB \rightarrow C$
$\Downarrow$
$A \rightarrow C$
$B \rightarrow C$.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Closure set of attributes :-

$R(ABC)$
$A \to B$
$B \to C$ } $A \to BC$.

$F = f_1 + f_2$

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

$A \to BC$

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

1. $R(ABCDEFG)$
$A \to B$
$BC \to DE$
$AEG \to G$.
$(AC)^+ = AC$
ABC
$\boxed{ABCDE}$

$G \to B$.

2. $R(ABCDE)$
$A \to BC$
$CD \to E$
$B \to D$
$E \to A$
$(B)^+ = B$
$\boxed{BD}$ same result

2 iterations

3. $R(ABCDEF)$
$AB \to C$
$BC \to AD$
$D \to E$
$CF \to B$.
$(AB)^+ = ABC$
$ABCD$
$ABCDE$.

4. $R(ABCDEFGH)$
$A \to BC$
$CD \to E$
$E \to C$
$D \to AEH$
$ABH \to BD$.
$DH \to BC$
$BCD \to H$?

$(BCD)^+ =$
$ABCDEH$

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Equivalence of Functional Dependency

R(ACDEH)

F: $A \rightarrow C$
$AC \rightarrow D$.
$E \rightarrow AD$.
$E \rightarrow H$.

G: $A \rightarrow CD$.
$E \rightarrow AH$.

a) $F \subseteq G$.
b) $F \supseteq G$.
c) $F = G$.
d) $F \neq G$.

$(A)^+ = ACD$
$(E)^+ = EADHC$

$(A)^+ = ACD$.
$(AC)^+ = ACD$.
$(E)^+ = EAHCD$.

R(PQRS).

X: $P \rightarrow Q$
$Q \rightarrow R$
$R \rightarrow S$

Y: $P \rightarrow QR$
$R \rightarrow S$.

$(P)^+ = PQRS$.
$(Q)^+ = Q$
$(R)^+ = RS$.

$(P)^+ = PQRS$
$(R)^+ = RS$.

a) $X \subseteq Y$
b) $Y \subseteq X$.
c) $X = Y$.
d) $X \neq Y$.

$X \not\subseteq Y$.

$Y \subseteq X$.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Irreducible set of FD (canonical form)

R(WXYZ)
$X \to W$
$WZ \to XY$.
$Y \to WXZ$.

Redundant : if your presence or absence doesn't affect the system it is called redundant.

R(WXYZ)
$X \to W$
$WZ \to XY$.
$Y \to WXZ$

$X \to W$ ✓
$WZ \to X$ ✗
$WZ \to Y$ ✓
$Y \to W$ ✗
$Y \to X$ ✓
$Y \to Z$ ✓

$X^+ = XW$
$X^+ = X$
$(WZ)^+ = WZXY$?
$(WZ)^+ = WZYX$.
Since both results are the same
∴ Redundant.

$X \to W$
$WZ \to Y$
$Y \to X$
$Y \to Z$.

$X \to W$
$WZ \to Y$
$Y \to XZ$

$(WZ)^+ = WZYX$.
$W^+ = W$
$Z^+ = Z$

∴ If they are the same it means the presence or absence of Z doesn't create any diff.

Scanned with CamScanner

Scanned by TapScanner

Key → Chabi
to extract data

R

A	B	C
1	a	p
2	b	q
3	c	q
4	c	r

The set of attributes which can uniquely identify a row or a tuple is called ..... key.

Key = A, BC

Key = A → BC.

$(A)^+ = R$.

$$\boxed{(key)^+ = R}$$

1. R(ABCD)

A → BC

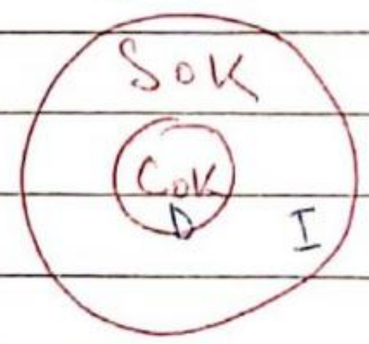
$A^+ = ABC$

$\boxed{(key)^+ = R}$

$(SoK)^+ = R$

↓

→ CoK (efficient)



2. R(ABCD)

ABC → D.

AB → CD.

A → BCD.

$(ABC)^+ = ABCD$.

$(AB)^+ = ABCD$.

subset → $(A)^+ = ABCD$.

3. R(ABCD)

B → ACD.

ACD → B.

$(B)^+ = ACDB$.

$(ACD)^+ = ACDB$

B → } Super key

ACD → } candidate key

1. R(ABCD)

A → B.

B → C

C → A.

AD, BD, CD

$(AD)^+ = ABCD$.

$(BD)^+ = ABCD$.

$(CD)^+ = ABCD$.

R(ABCD)

Find the attribute with no incomming edge.

Scanned with CamScanner

Scanned by TapScanner

Essential.

R(ABCDEF)

AB → CD
C → D.
D → B

R(A B C D)

$(A)^+ = A$ X
$(B)^+ = B$ X
$(C)^+ = C$ X
$D^+ = DB$ X

AB ✓
ACX
AD ✓
BC ✓
BD X
CD ✓

ACX
BDX

X ACB
X ACD
X BDA
X BDC

R (A B C D E F)

AB → C
C → D
D → BE
E → F.
F → A.

AX
BX
C ✓
D ✓
EX
FX

AB ✓
AE X
AF X
BE ✓
BF ✓
EF X.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Insertion, Deletion and Updation Anomalies in DBMS

Student_info.

S_id	name	age	Br_code	Br_name	HOD
1	A	18	101	CS	XYZ
2	B	19	101	CS	XYZ
3.	C	18	101	CS	XYZ
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6	F	19	103	ME	KLM

Idea:-
In the table student_info we have tried to store entire data about student.

Result:-
Entire branch data of a branch must be repeated for every student of the branch.

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

Disadvantages:-
(i) Insertion, deletion and modification anomalies.
(ii) Inconsistency (data)
(iii) Increase in database size & increase in time (slow)

Insertion anomalies:-
When certain data (attribute) can't be inserted into Database, without the presence of other data.

Deletion anomalies:-
If we delete some data (unwanted) cause deletion of some other data (wanted)

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Updation/Modification anomalies :-
when we want to update a single piece of data but it must be done at all

Student-info

S_id	name	age	Br_code	Br_name	HOD
1	A	18	101	CS	X42
2	B	19	101	CS	X42
3	C	18	101	CS	X42
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6.	F	19.	103.	ME	KLM

Kisi "Table" Ko decompose Karna

Student-info

S_id	name	age	Br_code.
1	A	18	101
2	B	19	101
3	C	18	101
4	D	21	102
5	E	20	102
6	F	19	103.

Branch-info.

Br_code	Br_name	HOD_name.
101	CS	X42
102	EC	PQR
103.	ME	KLM.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

- As one paragraph contains a single idea similarly one table must contain direct and main data about an Entity.

- Normalization (decomposition of tables) of table is done of the basis of functional dependencies.

1NF :- (atomic value).

Roll no.	name	course
101	Modi	CN OS
102	Sonia	DBMS. CO.

101	Modi	CN
101	Modi	OS
102	Sonia	DBMS
102	Sonia	CO

- every cell contains atomic values

2NF :-

R(A B C D) $(AB)^+ = ABCD$.

AB ← C.K.

A,B ← prime attribute.

C,D ← nonprime attribute.

B → C (partial dependency) ***

non-prime attribute

Only a part of candidate key.

A	B
1	—
—	2
—	—
3	4

ANS :-

R_1(A B D)

R_2(B C)

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

$R(ABC)$

$B \rightarrow C$

$(AB)^+ = ABC$.

$A, B \leftarrow P$

$C \leftarrow NoP$.

$P \rightarrow NoP$ (Partial dependency)

$B \rightarrow C$.

R

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

R_1

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

R_2

B	C
1	x
2	y
3	z.

1. $R(ABCDE)$.

$AB \rightarrow C$

$D \rightarrow E$.

$R(ABCDE)$.

$(ABD)^+ = R$.

**(Partial dependency).

$R_1(ABC)$

$R_2(DE)$

$R_3(ABD)$

2. $R(ABCDE)$

$(AC)^+ = R$

(P.D) $A \rightarrow B$

$B \rightarrow E$

(P.D). $C \rightarrow D$

$R_1(ABE)$

$R_2(CD)$

$R_3(AC)$

Scanned with CamScanner

Scanned by TapScanner

3. R(ABCDEFGHIJ)

PD $AB \rightarrow C$

PD $AD \rightarrow GH$ $\qquad (ABD)^+ = R$.

PD $BD \rightarrow EF$

PD. $A \rightarrow I$

$H \rightarrow J$.

$R_1(ABC)$

$R_2(AI)$

$R_3(ADGHJ)$

$R_4(BDEF)$.

$R_5(ABD)$.

Transitive dependency :-
A F.D from $\alpha \rightarrow \beta$ is called transitive if α, β is non-prime.

3NF :- A relation R is in 3NF if
a) it is in 2NF
b) no transitive dependency.

every dependency from $\alpha \rightarrow \beta$.
(i) Either α is superkey.
(ii) or β is a prime attribute.

F.D
$P \rightarrow NP$
T.D
$NP \rightarrow NP$

R(ABC)

$A \rightarrow B$
$B \rightarrow C$.

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

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

B	C
1	x
2	y
3	z

Scanned with CamScanner

Scanned by TapScanner

3. R(ABCDE) R(ABCDE)

AB → C × A → B.
B → D. × B → E.
D → E. ✓ C → D

$(AC)^+ = ABCDE$.

ABCDE

- R_1 (ABE)
  - R_{11} (AB)
  - R_{12} (BE)
- R_2 (CD)
- R_3 (AC)

4. R(ABCDEFGHIJ)

✓ AB → C
× A → DE. Candidate Key :- AB
B → F. ∵ $(AB)^+ = R$.
F → GH.
D → IJ.

- R_1 (ADEIJ)
  - R_{11} (ADE)
  - R_{12} (DIJ)
- R_2 (BFGH)
  - R_{21} (BF)
  - R_{22} (FGH).
- R_3 (ABC)

Scanned with CamScanner
Scanned by TapScanner

Orion
PAGE:
DATE: / /

BCNF :-
(Boyce Codd Normal Form)

2NF. $\alpha \xrightarrow{\times} B$
prime. no prime

3NF $\alpha \xrightarrow{\times} B$
no p no p

$\alpha \longrightarrow \beta$.
P/NP P

R(ABC) $(A)^+$

$AB \longrightarrow C$. $(AB)^+$ ✓
$C \longrightarrow B$. $(AC)^+$ ✓

$\alpha \longrightarrow \beta$.
P/NP P

$\alpha \longrightarrow B$
S.K.

AB
AC

R(ABC)
$AB \longrightarrow C$
$C \longrightarrow B$.
$\alpha \longrightarrow B$.

α must be a superkey

A	C
a	x
b	y
c	z
c	w
d	w
e	w

C	B
x	1
y	2
z	2
w	3

R

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

- $R_1(CB)$
- $R_2(AC)$
- ~~$R_3(AB)$~~.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

1. R(ABCDEFGH).

AB → C
A → DE.
B → F
F → GH.
(AB).

~~BCNF.~~
~~3NF.~~
~~2NF.~~
(1NF).

2. R(ABCDE).

CE → D.
D → B.
C → A
(CE).

~~BCNF.~~
~~3NF~~
~~2NF.~~
(1NF)

WF

3. R(ABCDEF)

AB → C
DC → AE.
E → F.
(ABD) (BCD).

4. R(ABCDEFGHI)

AB → C
BD → EF.
AD → GH.
A → I
(ABD).

3NF

INF.

5. R(ABCDE).

{ AB → CD.
D → A → prime attribute
BC → DE.

(AB), (BD), (BC)

6. R(ABCDE)

BC → ADE.
(D) → (B)
(BC), (CD).

Candidate key || prime attribute.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

7. R(VWXYZ)

CK1 X → Y V PA

Y → Z

Z → Y

VW → X

VW , XW

1NF

8. R(ABCDEF)

ABC → D

ABD → E

CD → F

CDF → B

BF → D

ABC , ACD

1NF

9. R(ABC)

A → B

B → C

C → A

A, B, C

BCNF

Scanned with CamScanner

Scanned by TapScanner

How to find identify normal form in dbms.

1. R(ABCDEF)
   A → BCDEF.
   BC → ADEF
   DEF → ABC.
   A, BC, DEF.
   BCNF.

2. R(ABC)
   AB → C
   C → A.
   AB, BA.
   3NF.

3. R(ABCDE)
   A → B
   BC → E.
   DE → A
   ACD, BCD, CDE
   3NF.

4. R(ABCDE).
   AB → CD.
   D → A.
   BC → DE.
   AB, BC, BD.
   3NF.

5. R(WXYZ)
   Z → W.
   Y → XZ.
   XW → Y.
   Y, XW, XZ.
   3NF.

6. R(ABCDE)
   ~~A → B~~ np.
   B → E.
   C → D.
   AE.
   1NF.

7. R(ABCDEF).
   AB → C
   DC → AE.
   E → F.
   ABD, BCD.
   1NF.

8. R(NWXYZ).
   Z → Y
   Y → Z
   X → Y → n.p.
   VW → X
   VW, XW.
   1NF.

9. R(ABCDEF)
   ABC → D.
   ~~ABD → E.~~
   CD → F np.
   CDF → B
   BF → D.
   ABC, ACD.
   1NF.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Lossless Join Decomposition / NON-additive.

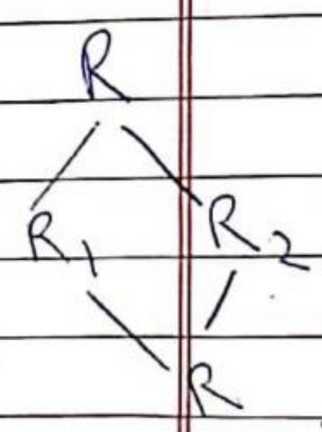
Tuple generation problem doesn't occur after decomposition.

- It is a mandatory property must always hold good.
- If a relation R is decomposed into two relations R_1 & R_2, then it will be lossless iff :-

(i) $attr(R_1) \cup attr(R_2) = attr(R)$.
(ii) $attr(R_1) \cap attr(R_2) \neq \phi$
(iii) $attr(R_1) \cap attr(R_2) \rightarrow attr(R_1)$.

or

$attr(R_1) \cap attr(R_2) \rightarrow attr(R_2)$.



R

A	B	C	D
1	a	p	x
2	b	q	y

R_1

A	B
1	a
2	b

R_2

C	D
p	x
q	y

$R_1 \times R_2$

A	B	C	D
1	a	p	x
1	a	q	y
2	b	p	x
2	b	q	y

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

R

A	B	C
1	a	β
2	b	γ
3	a	γ

R_1

A	B
1	a
2	b
3	a

R_2

B	C
a	β
b	γ
a	γ

$R_1 \times R_2$

A	B	C
1	a	β
1	a	γ
2	b	γ
3	a	β
3	a	γ

hence it is candidate key in any of the table.

Scanned with CamScanner

Scanned by TapScanner

If a table R having F.D set F, is decomposed into two tables R_1 and R_2 having F.D set F_1 and F_2

then $F_1 \leq F^+$
$F_2 \leq F^+$

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

$R(ABC)$
$A \rightarrow B$.
$B \rightarrow C$
$C \rightarrow D$

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

$R(F)$

$R_1(F_1)$ $R_2(F_2)$

$R(F_1 \cup F_2) = F$.

$R(ABCD)$
$AB \rightarrow CD$.
$D \rightarrow A$
$R_1(AD)$ $R_2(BCD)$.

$R(ABCD)$
$F: AB \rightarrow CD$
$D \rightarrow A$.

$R_1(AB)$ $R_2(BC)$
$F_1: A \rightarrow B$ | $F_2: B \rightarrow C$
$B \rightarrow A$ | $C \rightarrow B$

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

$C^+ = CBA$

$R_1(AD)$	$R_2(BCD)$
F_1: $D \rightarrow A$	F_2: $BD \rightarrow C$
$(AB)^+ = AB$.	

$R(ABCDEFGHIJ)$

$R_1(ABC)$

$R_2(ADEIJ)$

$R_{21}(ADE)$
$R_{22}(DIJ)$ } 3NF.

$R_3(BFGH)$

$R_{31}(BF)$
$R_{32}(FGH)$ } 3NF.

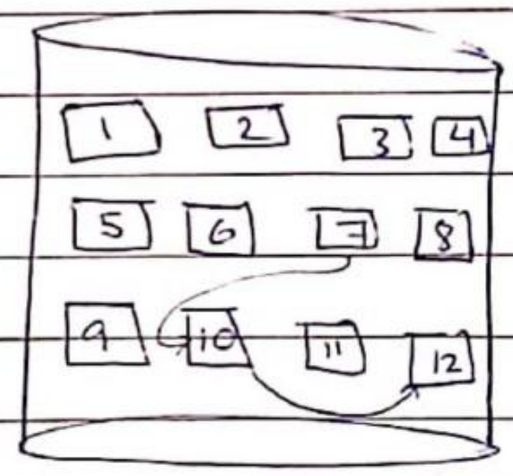
Scanned with CamScanner

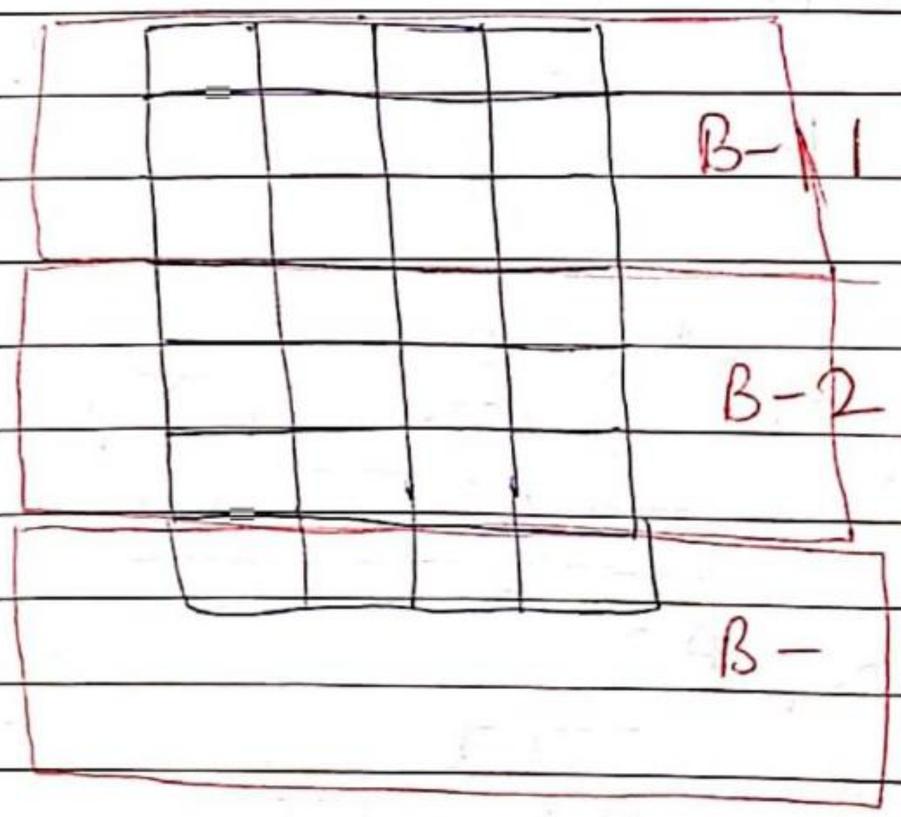
Scanned by TapScanner

File Structures

SRS
↓
ER-diagram.
↓
Relation model
↓
normalization.
↓
file structures
(underlying of physical structures)

*** File → data structure





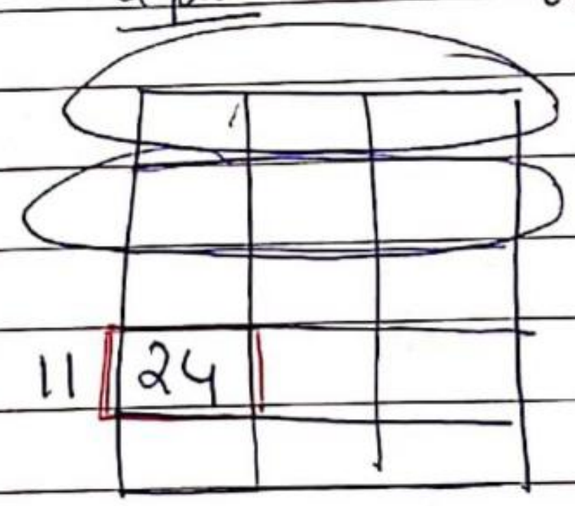
Scanned with CamScanner
Scanned by TapScanner

Orion
PAGE:
DATE: / /

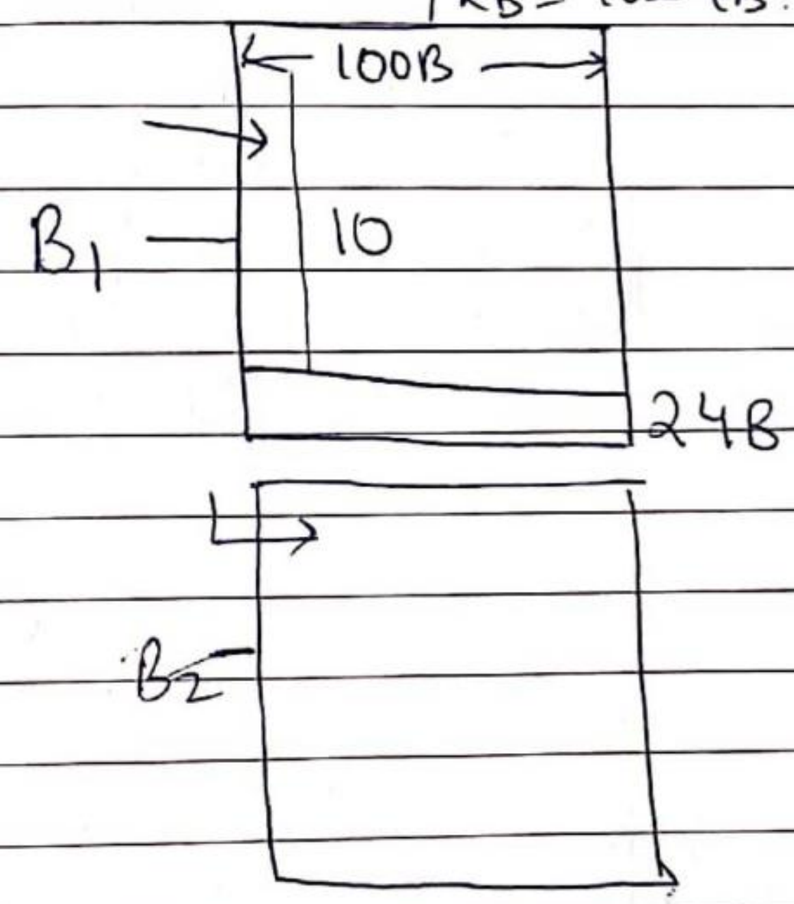
Sorted File	Unsorted File.
∘ Can be sorted acc. to one attribute (search key)	∘ Random order, any record can be placed anywhere
∘ Searching fast	∘ Searching will be slow.
∘ Insertion and deletion can be difficult.	∘ Insertion and deletion will be easy.

Spanned/Unspanned Mapping

2 part completely

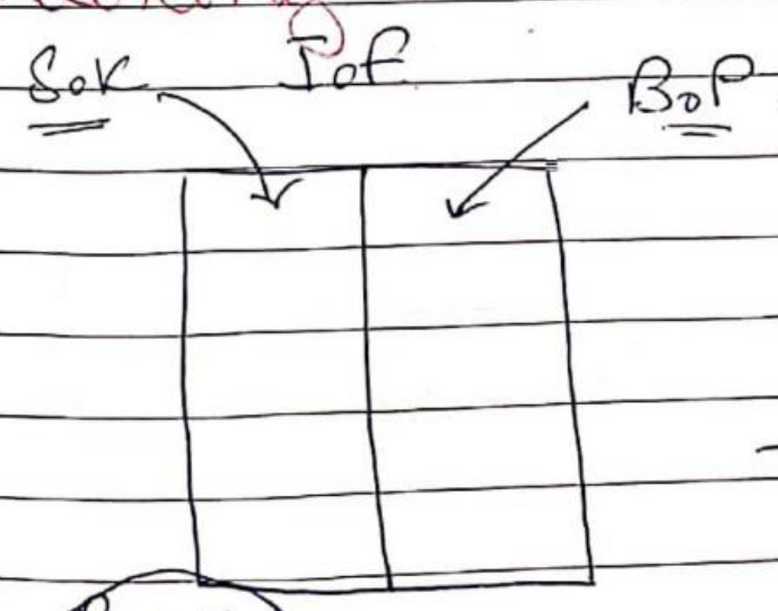


1KB = 1024B.



block → record
(ignore)

Indexing



Index → Size small
sorted/unsorted.

→ to access the file quickly.

Book
Index-
ch. - pg no

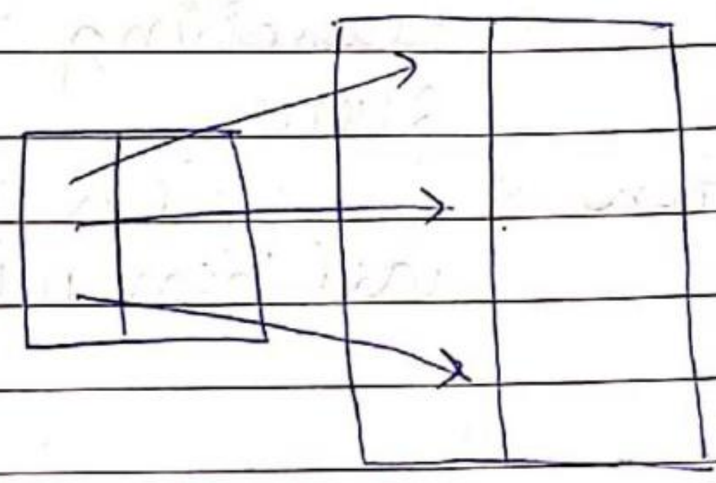
Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

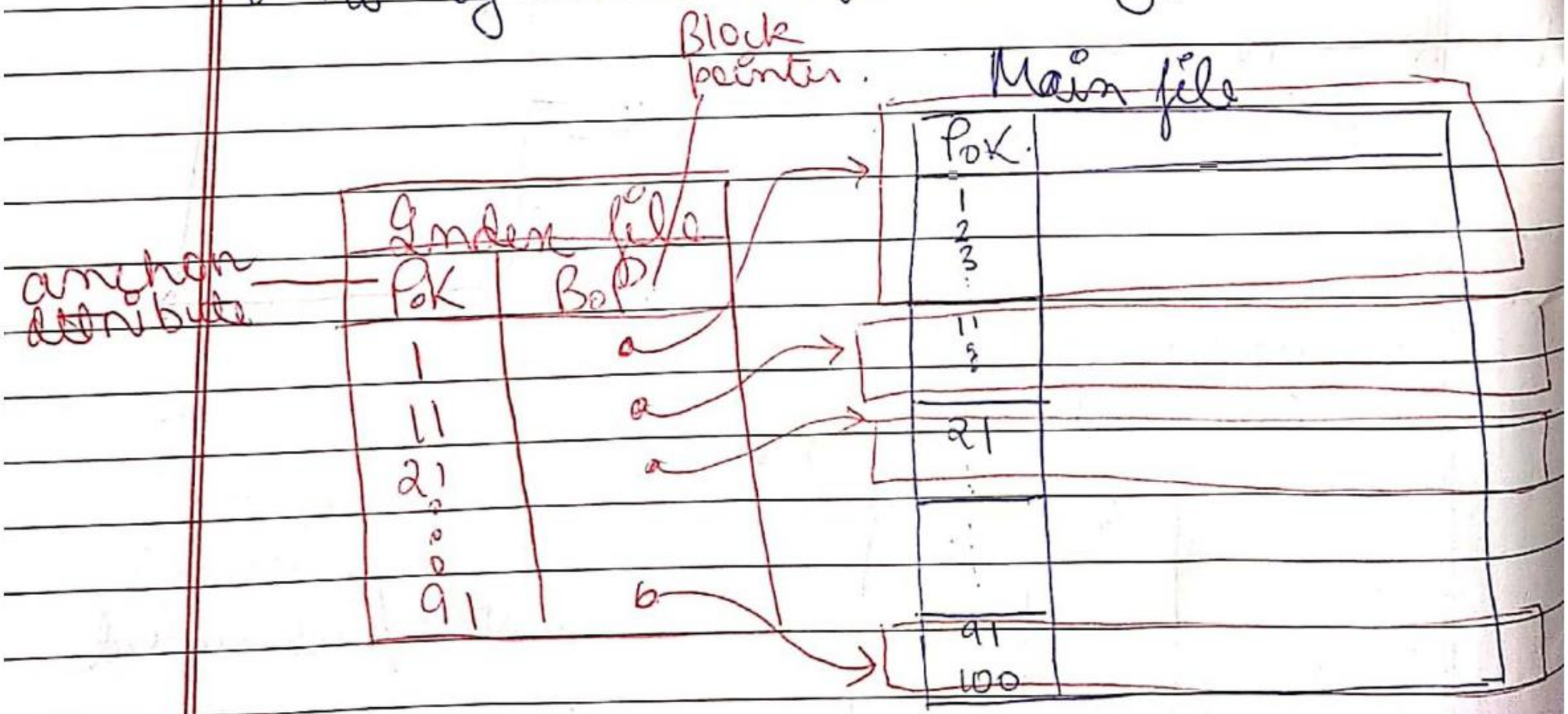
Types of indexing.

1. Primary indexing
2. Clustered indexing.
3. Secondary indexing.
4. Multilevel indexing.



Primary indexing.

- Main file sorted.
- Primary key is used as anchor attribute (Search key).
- Example of sparse indexing
- no. of entries = no. of blocks acquired in index file by the main file.
- no. of access required = $\log_2 n + 1$.



Scanned with CamScanner

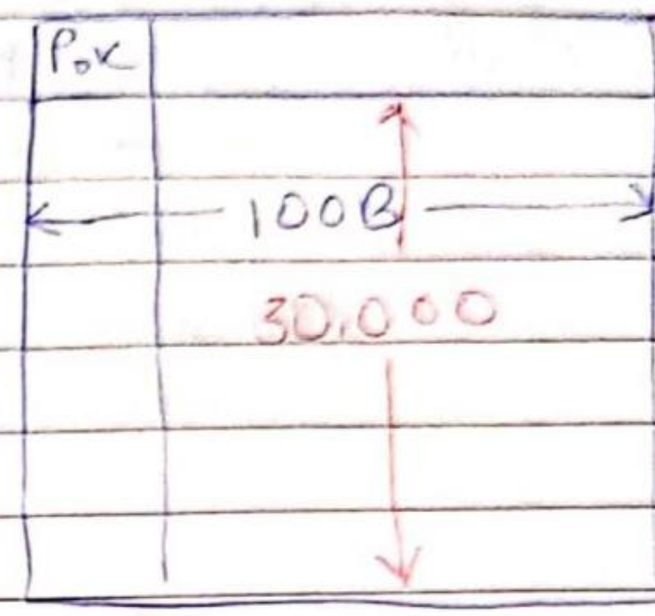
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Block size = 1024B

INDEX FILE

MAIN FILE



Blocking factor = $\left[\frac{\text{Block size}}{\text{Record size}}\right]$

no. of Blocks for main file (n) = $\left[\frac{\text{no. of records in main file}}{\text{Blocking factor}}\right]$

no. of block access required = $\log_2 n$.

Blocking factor = $\frac{1024B}{100B} = [10.24] = 10$.

no. of blocks for main file (n) = $\frac{30,000}{10} = 3000$.

no. of access reqd = $\log_2 n$
$= \log_2 3000$.
$= [11.\ldots] = 12$

INDEX FILE

Blocking factor = $\left[\frac{1024}{15}\right]$

no. of Blocks for index file = $\left[\frac{30,000}{68}\right]$
$= [44.\ldots]$
$= 45$.

no. of block access reqd = $[\log_2 n] + 1$
$= 6 + 1 = 7$.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Clustering index.

- Main file is sorted (on some non-key attribute).
- there will be one entry for each unique value of the non-key attribute.
- If number of block acquired by index file is n, then block access required will be

$$\geq \log_2 n + 1.$$

Index file

NoK	BoP
1	
2	
3	
4	
5	

Main file

NoK		
1		B1
1		
2		
3		
3		B2
3		
4		
4		
5		
.		
100		

Scanned with CamScanner

Scanned by TapScanner

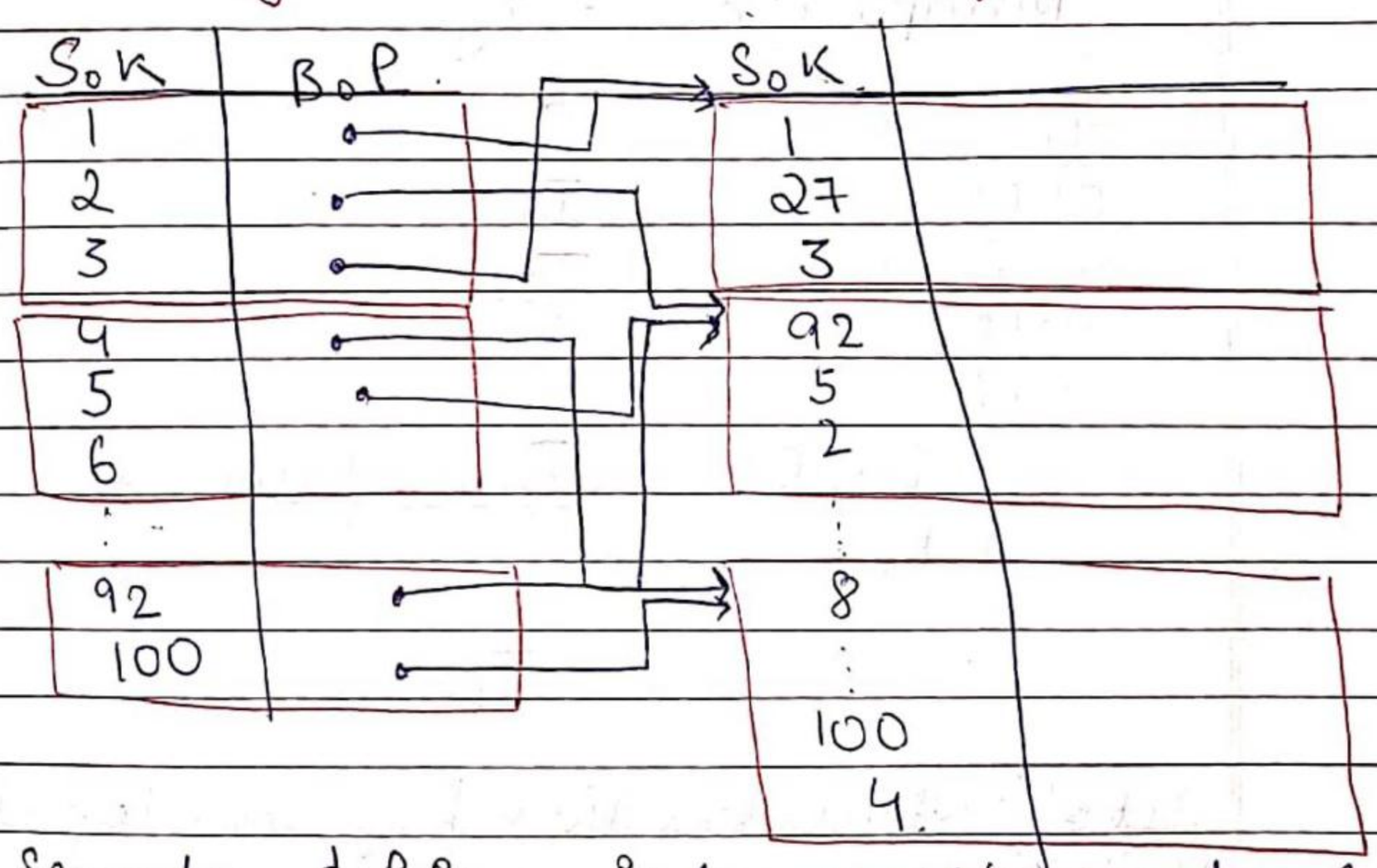
Orion
PAGE:
DATE: / /

Secondary Indexing :-

- Main file Unsorted.
- can be done on key as well as on non-key attribute.
- called secondary because normally one indexing is already done

[no of entry in index file = no of entry in main file]

- no of block access = $\lceil \log_2 n \rceil + 1$.



Secondary & Primary index can't be compared since in primary list is sorted.

Scanned with CamScanner
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Transaction.

What is transaction
What are acid properties.
Transaction states.
Problem with concurrency.
Conflict and view serializability
Recoverability and cascadlessness.
Transaction designing protocols.
- Time stamping.
- Lock based protocols.
2-phase locking.
Graph based protocol.
Multiple granularity.

T_1	P
R(A)	T_1
A = A − 10.	T_2 ←
W(A)	⋮
R(B)	⋮
B = B + 10.	T_{60} failure x
W(B).	⋮
	⋮
	T_{n100}.

Set of instructions that perform a logical unit of work - Transaction.

Transaction:-
1. Atomicity
2. Consistency.
3. Isolation.
4. Durability

T ⟶ [D.B.]

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

- DB can be managed by transaction
- Either all the instructions within a transactions will be executed or neither of them will be executed.

Atomicity → Transaction management component.

Isolation
If working of one transaction doesn't affect other transactions
Concurrency control component.

Durability
If a change has occured, the change will stay.
Recovery management component.

if these prob hold good =

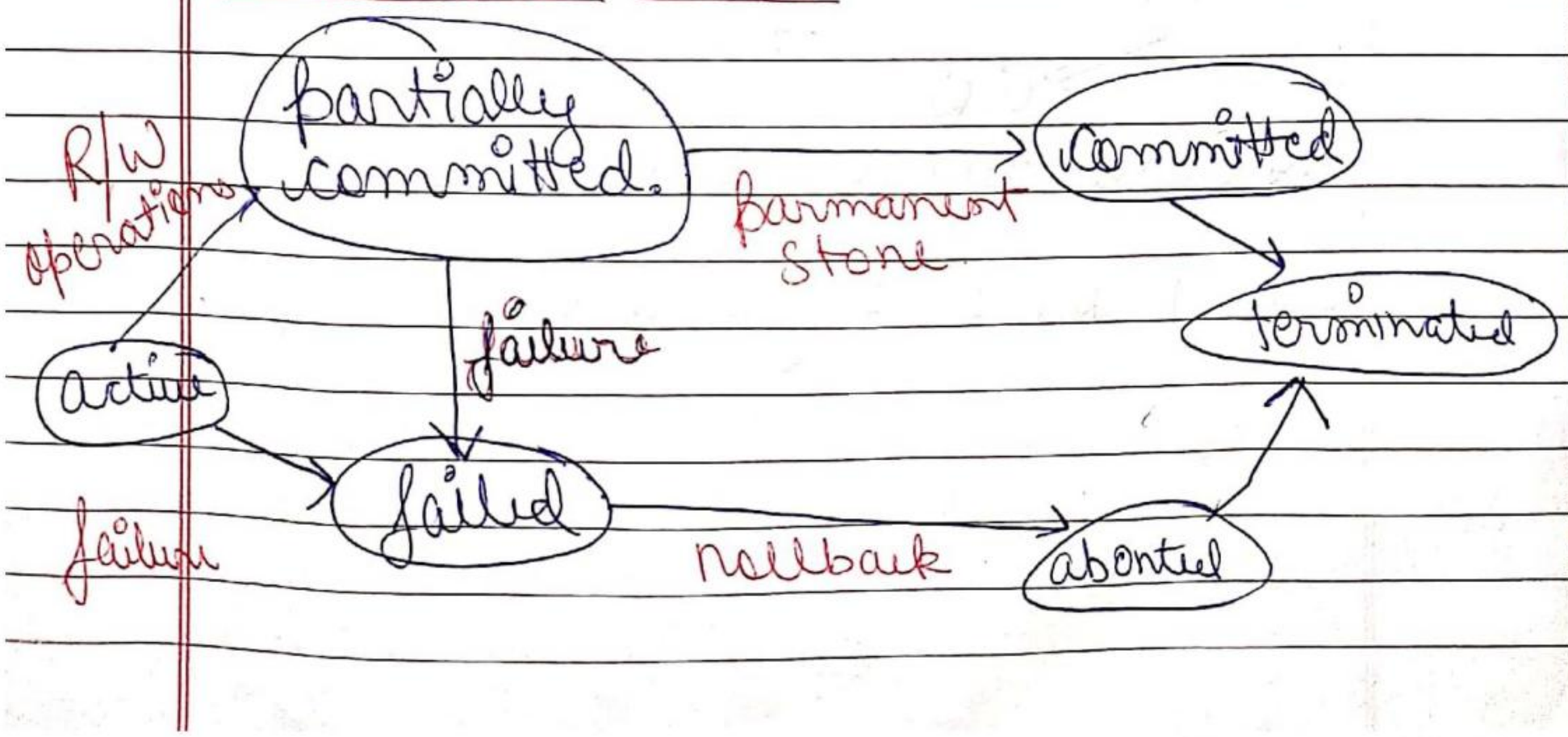
Consistency.

$DB_1 \xrightarrow{T} DB_2$

Consistent. (After transaction it will be consistent).

A+B = A+B

Transaction States.



Scanned with CamScanner

Scanned by TapScanner

e.g. form photo copy.

Initial state → active

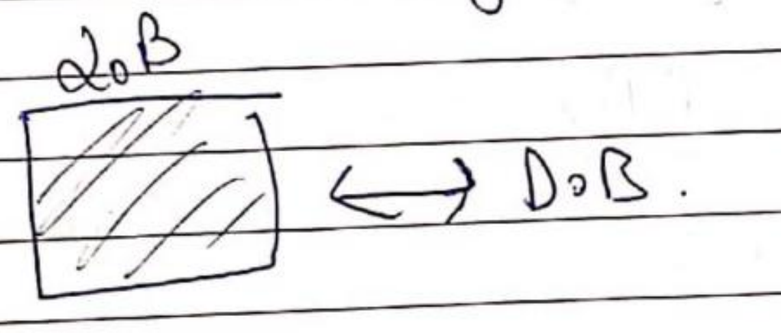
when the transaction is executing in terms of instructions $\{ I_1, I_2, \ldots, I_n \}$

when instruction is running

There may be a failure (failed state)

On we will mark the partially committed state after performing all the read/write operations.

We do the changes in a copy. (local buffer)
(because the data is not permanently stored in memory)



$$DB_1 \xrightarrow{T} DB_2$$
$$DB_1 \to LB \to DB_2$$

* consistent in both cases.

e.g. Ticket booking & cancelling.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Advantages of concurrency.
→ If we can run more than one transaction in a system (at a time) it is called concurrency.
interleaved execution of more than one transaction.

1) Waiting time. ↓
2) Response time. ↓
3) Resource utilization. ↑
4) Efficiency. ↑

Dirty Read Problem

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

→ two students giving exams & one cheating from the other one. One submitting the paper early. later the T changed ans.

Unrepeatable Read problem:-

$X = 10$

T_1	T_2
R(X)	
	R(X) — 10
X++ W(X)	
	R(X) — 11

child. bhoot
maggie
→ Nayma chawal.

Phatom Read Problem.

$X = 10$

T_1	T_2
R(X)	
	R(X)
Delete(X)	
	R(X)

Lo B $X = 10$

thief

this same story

Because a transaction is like independent doesn't depend on other transaction.
But here it does.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Lost update problem (write write conflict)

	T_1	T_2
DB A = ~~10~~ 50	R(A)	
	W(A)	
		W(A)
		C
LOB = ~~10~~ ~~11~~ 50.	C	

→ conflict b/w these two writes

* Serial schedule and non-Serial schedule in DBMS.

Schedule:

S1

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

consistent

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

$n_1 + n_2$.

S2

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

non-serial schedule.

$T_1, T_2, T_3 \ldots T_n$

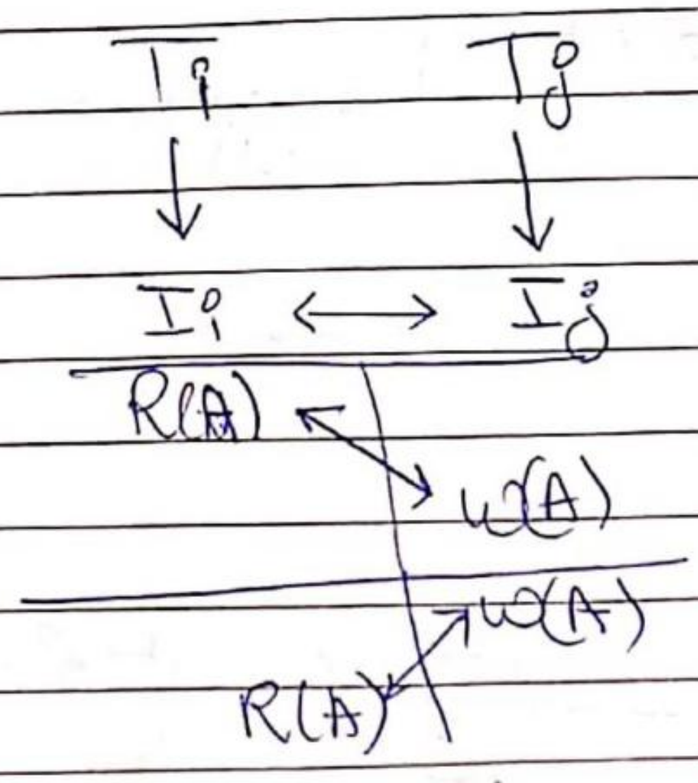
$n \times (n-1) \times (n-2) \ldots \times 1$

Total no. of Non Serial Schedule $= \dfrac{(n_1 + n_2 + \ldots + n_n)!}{n_1!\, n_2! \ldots n_n!} = n!$

Scanned with CamScanner

Scanned by TapScanner

- If we can prove non serial ≡ serial then it is consistent.
- when instruction belong to same data item conflicts happen.



no changes in TA1, TA2 & DB after swap.

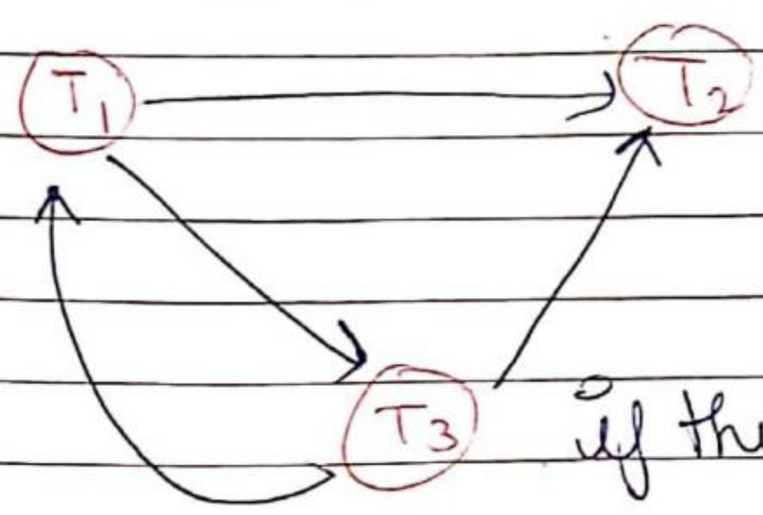
9/2/2022

Conflict Serializable Schedule

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)		

1st ⟶ 2nd



if there is a cycle

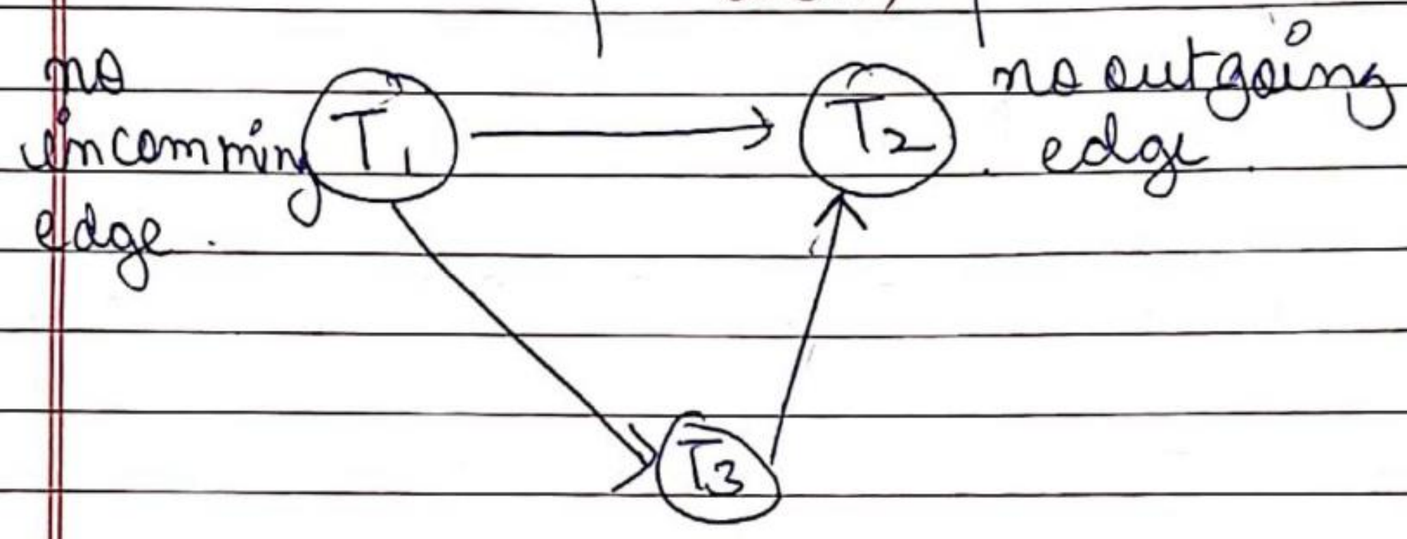
"NOT" a conflict serializable schedule

Scanned with CamScanner

Scanned by TapScanner

S

T_1	T_2	T_3
R(X)		
	R(Y)	
		R(Y).
	W(Y)	
W(X)		
		W(X).
	R(X)	
	W(X)	

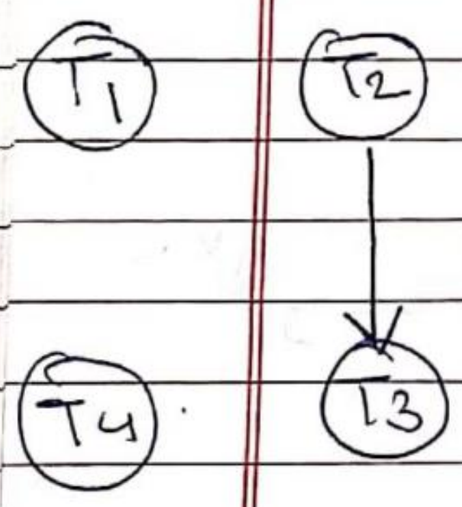


Cost of finding

$= O(n^2)$

$T_1 \rightarrow T_3 \rightarrow T_2$

$n \rightarrow$ no of transaction



S

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

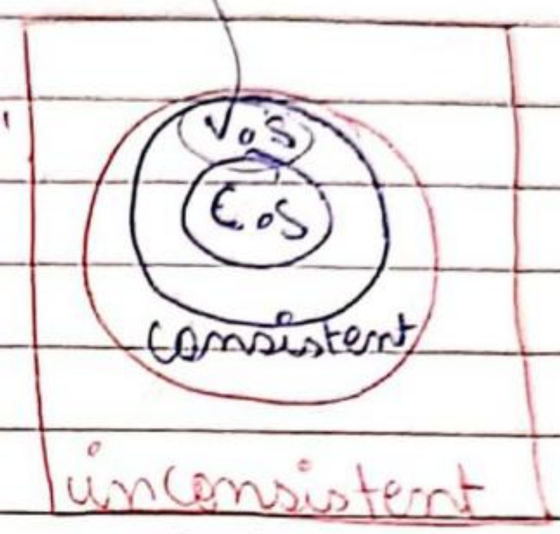
check till it is committed

when one of its view serial is view equivalent

Orion
PAGE:
DATE: / /

View Serializability

If it is not C.S
But it is V.S
it will have "Blind write"



schedules

S

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

S

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

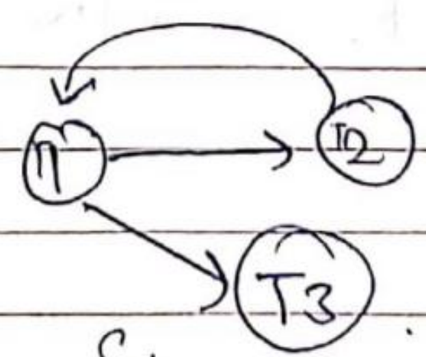
1) T_1 a T_1 } Initial
T_1 b T_1 } Read.
same

2) Final write same.

3) Intermediate read same.

S

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



S1

T_1	T_2	T_3
Ra		
W(a)	W(a)	(Wa)

S_1 123
S_2 132
213
231
S_6 312
321

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Recoverable Schedule :-

S

T_1	T_2
10 - R(A)	
A = A+10	
20 - W(A)	
	R(A) - (20)
	A = A - 5 (15)
	W(A)
C	C
C	C

D.R (Dirty Read) when a transactions reads the uncommited changes of other transaction.

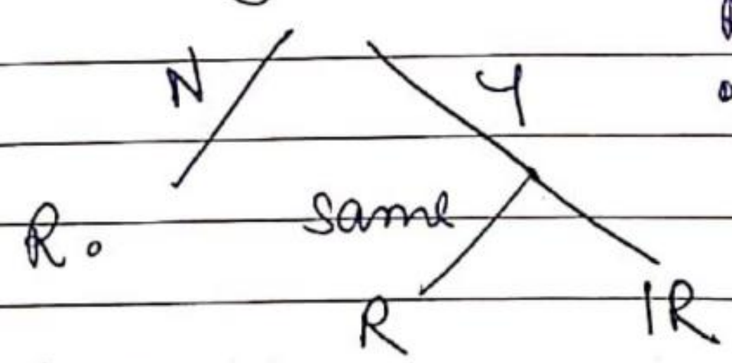
→ recoverable

A = ~~10~~ 15

R.B A = ~~10~~ 10
~~20~~
15

we roll backed T_1 but not T_2.
when T_1 was rolling back we wanted T_2 to roll back as well.

Dirty Read



Recoverable →
• the order dirty read = the order of commit.

Scanned with CamScanner
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Cascadless schedule :- (no cascading rollbacks)

S

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

non Recoverable.

$T_1 \rightarrow T_2 \rightarrow T_3$.

C
C
C

Recoverable

S

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

R
CR

D.R

N / Y

CL, R CLx

same / same x

R IR

Scanned with CamScanner

Scanned by TapScanner

Strict Schedule

S1

T_1	T2
R(a)	
W(a)	
	W(a)
C	
	R(a)
	C

S2

T_1	T2
R(a)	
W(a)	
C	
	W(a)
	R(a)
	C

S3

T_1	T_2
R(a)	
	R(b)
W(a)	
	W(b)
C	
	R(a)
	C

→ cascadeless.

Strict schedule
→ when a schedule/transaction has performed a write operation on a data item then no other transaction can perform a read/write operation on that data item.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Concurrency Control techniques:-

- Till now we already know how to check whether a schedule will maintain the consistency of DB or not (CoS, VoS, Recoverability, Cascadless, strict etc)
- Now will study protocols that guarantees to generate schedule which satisfy these properties specially (CoS).
- For CoS we must avoid conflicting instructions. We remember three conditions of conflicting instruction.
- Actual problem is different transaction trying to access data at same time.
- How to approach conflict serializability.

  - time stamping protocol (Ordering)
  - lock based protocol (lock-unlock)
    - ↳ 2PL (Basic, conservative, strict, rigorous).
    - ↳ Graph based.
  - Validation Protocol.

- Goals:-
  Concurrency, properties, time, logic.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Time Stamp Ordering Protocal :-

- Basic idea of Time stamping is to decide (the order) between the transactions by one that enters into the system. So that in case of conflict during execution, we can resolve the conflict using Ordering.
- the reason we call time-stamp not stamp because, for stamping are the values of system clock. (as it will always be unique can never repeat itself)

Two ideas of time stamping :-

- Time stamp with transaction :- with each transaction T_i, we associate a time-stamp denoted by T.S (T_i).
- It is the value of the system clock when a transaction enters into the system, so if a new transaction T_j enters after T_i then $T.S(T_i) < T.S(T_j)$ always unique, will remain fixed through the execution
- also determine serializability order if $T.S(T_i) < T.S(T_j)$ then, system ensure that in the resultant C.S.S T_i will execute first before T_j.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Time stamp with data item :-
For each data item Q protocal maintains two time-stamps.

W-timestamp(Q) :-
is the largest time-stamp of any transaction that executed write(Q) successfully.

R-time stamp(Q) :-
is the largest timestamp of any transaction that executed read(Q) successfully.

Ti request for Read(Q) :-

if ToS(Ti) < W.T.S(Q),
means Ti needs to read a value of Q that was already overwritten.
hence request must be rejected & Ti must rollback

if T.S(Ti) ≥ W.T.S(Q)
Operation can be allowed and R.T.S(Q) will be max(RTS(Q), TS(Ti))

Ti request for Write(Q)
if T.S(Ti) < RTS(Q)
means value of Q that Ti is producing was needed previously and the system assumed that the value would never be produced
hence reject & rollback.
if T.S(Ti) < WTS(Q)
Ti is attempting to write an absolute value of Q reject and Rollback
Otherwise
WTS(Q) = max(WTS(Q), TS(Ti))

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

S

$T_0 S(T_P) = 5$	$T_0 S(T_X) = 10$.
R(Q)	W(Q)
R(Q)	

$TS(T_P) = 10$	$T_0 S(T_X) = 5$.
	W(Q)
R(Q)	

junior/ professor doing somearticity outside college gate.

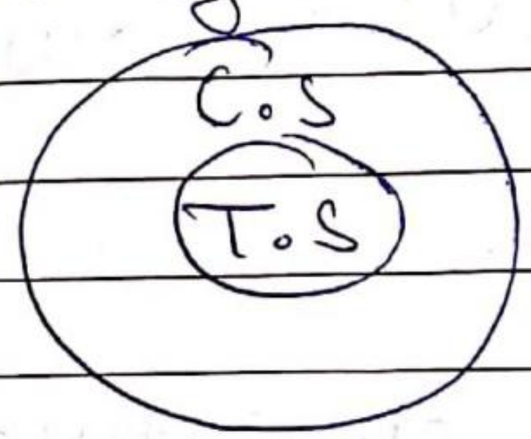
Scanned with CamScanner

Scanned by TapScanner

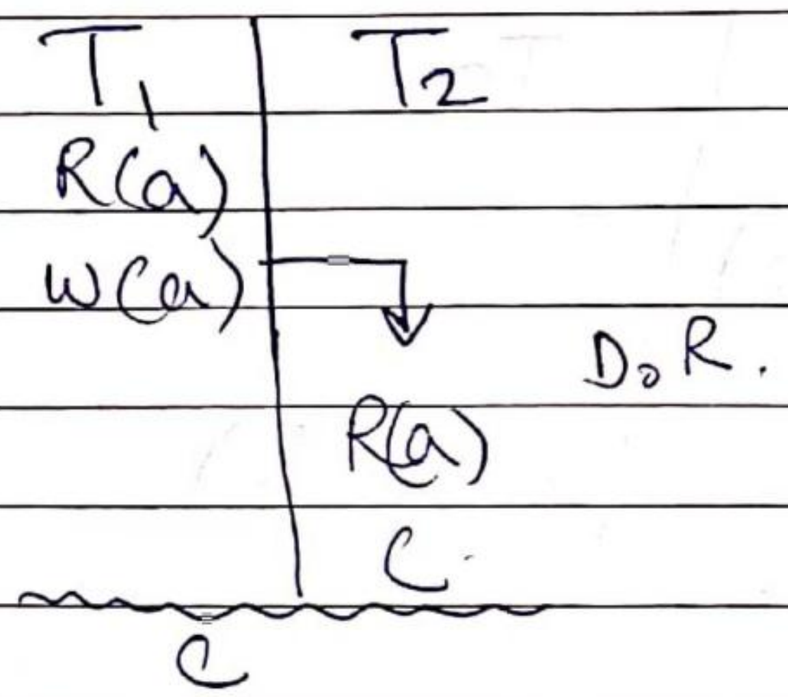
Orion

PAGE:
DATE: / /

Properties of Time Stamping Protocal :-

- It ensure conflict serializability.
- It ensure view serializability.
- possibility of dirty read no restriction can commit, irrecoverable schedule and cascading rollbacks are possible.
- here either we allow or reject so no idea of deadlock.
- 

- may suffer from starvation relatively slow.



Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Thomas Write Rule :-

Modify time stamping protocal to make some improvements and may generate those schedules that are V.S but not C.S and provide better concurrency.

- It modify time-stamping protocal in absolute write case when T_i request write (Q)
  if $TS(T_i) < WTS(Q)$.

- here T_1 attemps to write absolute value of Q rather than rolling back T_1, write operation is ignored.

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

$RTS(Q) = 1$.
$WTS(Q) = 2$.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Q = ~~10~~ 14.
Q = ~~10~~ ~~12~~ 14.
$T_i^0 = 12$,
$T_x = 14$.

$T_0 S(T_i^0) = 5$	$T_0 S(T_x) = 10$
W(Q)	
	W(Q)
~~W(Q)~~	

Lock Based Protocol :-

- To achieve consistency isolation is the most important idea. Locking is simplest idea to achieve isolation i.e. first obtain a lock on a data item then perform a desired operation and then unlock it.
- To provide better concurrency along with isolation we use different modes of locks.

Shared Mode :-

Denoted by lock S(Q) transaction can perform read operation, any other transaction can also obtain same lock on same data item at same time (so called Shared)

Exclusive mode :-

Denoted by lock - X(Q) transaction can perform both Read/write operations, any other transaction cannot obtained either Shared/exclusive mode lock.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Compatability table.

	Shared	exclusive
Shared	T	F
exclusive	F	F.

Properties of Lock Based Approach.

S

T_1	T_2
Lock-X(A)	
R(A)	
W(A)	
Unlock(A)	
	Lock S(B)
	R(B)
	Unlock(B).
Lock-X(B)	
R(B)	
W(B)	
Unlock(B)	
	Lock-S(A)
	R(A)
	Unlock(A)

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

- if we do unlocking inconsistency will arise, if we do not unlock them concurrency will be poor.
- we require than transaction follow some set of rules for locking and unlocking of data item e.g. 2PL or graph based.
- We say a schedule is legal under a protocal if it can be generated using the rules of the protocals.

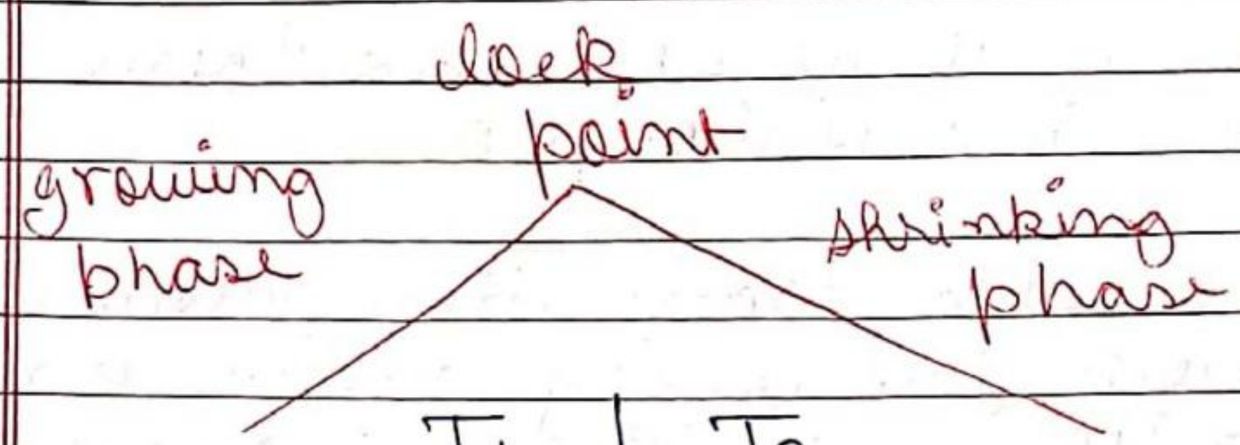
Two-Phase Locking Protocal (2PL) :-
Basic 2PL.

- This protocal requires that each transaction in a schedule will be two phased - Growing phase & shrinking phase.
- In growing phase - transaction can only obtain locks but can not release any lock.
- In shrinking phase - transaction can only release locks but can not obtain any lock.
- transaction can perform read/write operation both in growing / shrinking phase.
- Ensure CoS/VoS, the order of serialiability is the order in which transaction reaches lock point.
- Do not ensure freedom from deadlock.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /



T_1	T_2
lock-S(A)	
Read (A)	
	lock-X(B)
	Read (B)
	Write(B).
lock S(B)	
	lock X(A)

S

T_1	T_2
lock X(A)	
R(A)	
W(A)	
unlock X(A)	
	lock S(A)
	R(A)

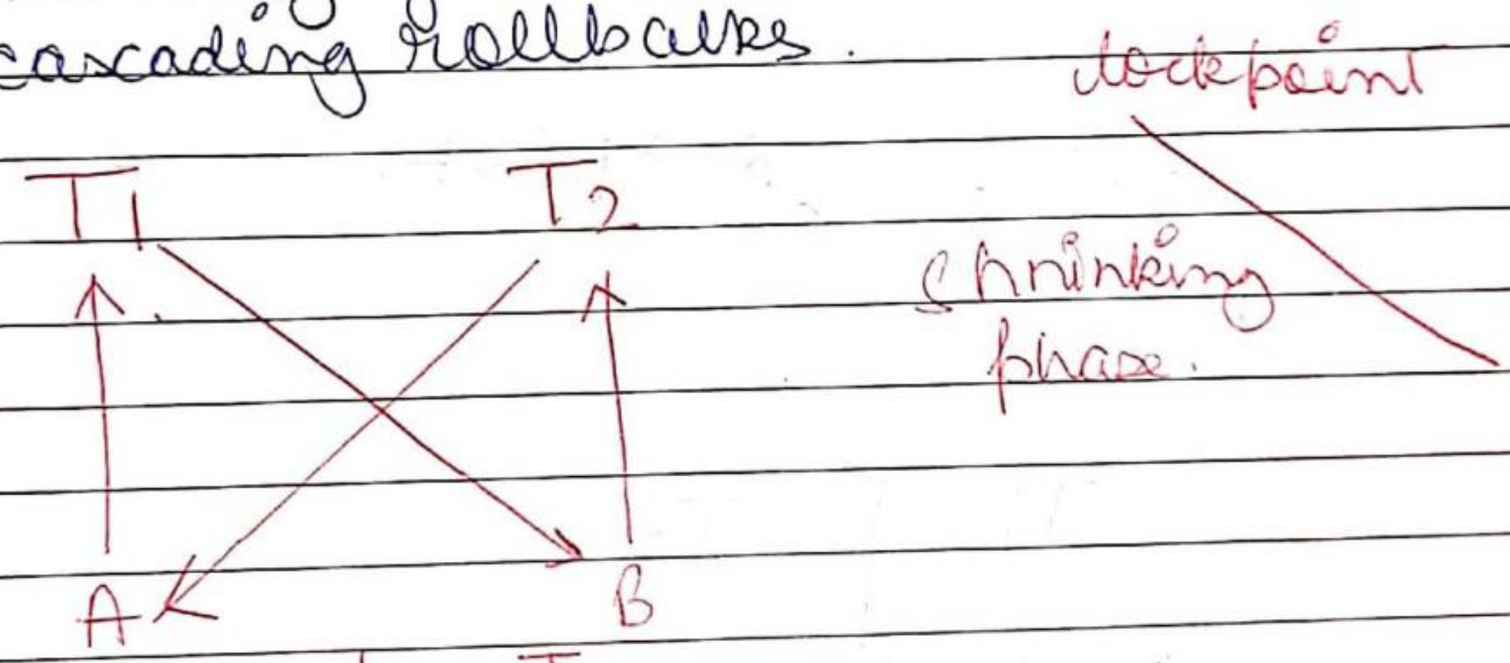
Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Conservative / Static 2PL:-

- There is no growing phase transaction first will acquire all the locks need and then directly will start from lock point.
- If all locks are not available then transaction must release the lock acquired so far and wait.
- Shrinking phase works as usual and transaction can unlock any data item.
- Here we must have all the knowledge that what data items will be required during execution.
- C.S, V.S independent from deadlock.
- possibility of irrecoverable schedules and cascading rollbacks.



T_1	T_2
lock X(A)	
R(A)	
W(A)	
	lock S(B)
	R(B).
lock X(B)	
	lock S(A).

Scanned with CamScanner

Scanned by TapScanner

PAGE:
DATE: / /

Rigorous 2PL:

It is an improvement over 2PL protocol where we try to ensure recoverability and cascadelessness.

- Rigorous 2PL requires that all the locks must be held until transaction commits. i.e. there is no shrinking phase in the life of a transaction.

- We ensure C.S, V.S, recoverability, cascadelessness
- Suffer from deadlock & inefficiency.

Growing phase — lockpoint.

T_1	T_2
lock X(A)	
R(A)	
W(A)	
unlock(A)	
	lock S(A)
	R(A)
C	

Scanned with CamScanner

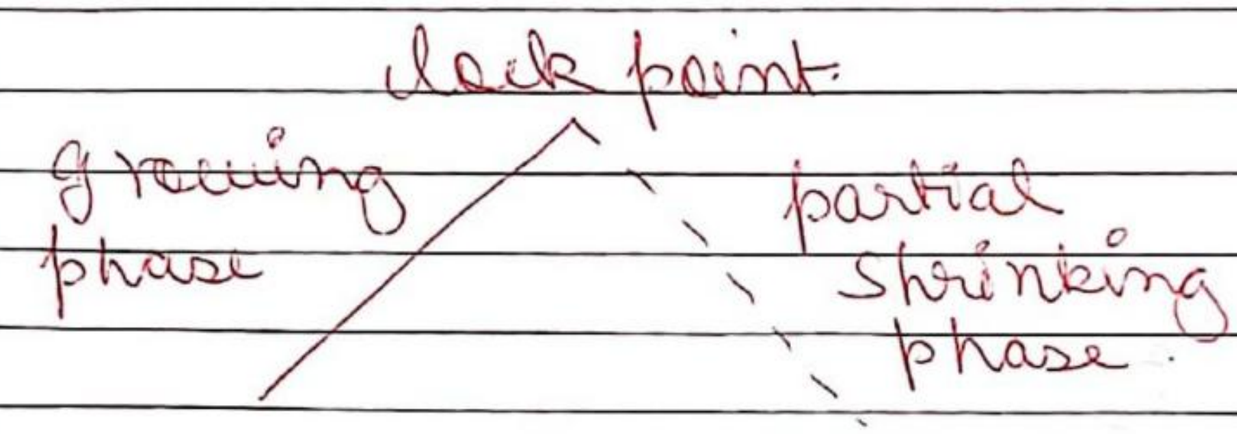
Scanned by TapScanner

Orion
PAGE:
DATE: / /

Strict 2PL :-

It is an improvement over Rigorous 2PL.

- In the shrinking phase unlocking of exclusive locks are not allowed but unlocking of shared locks can be done.
- all the properties are same as that of rigorous 2PL, but it provide better concurrency.



Graph based Protocol :-

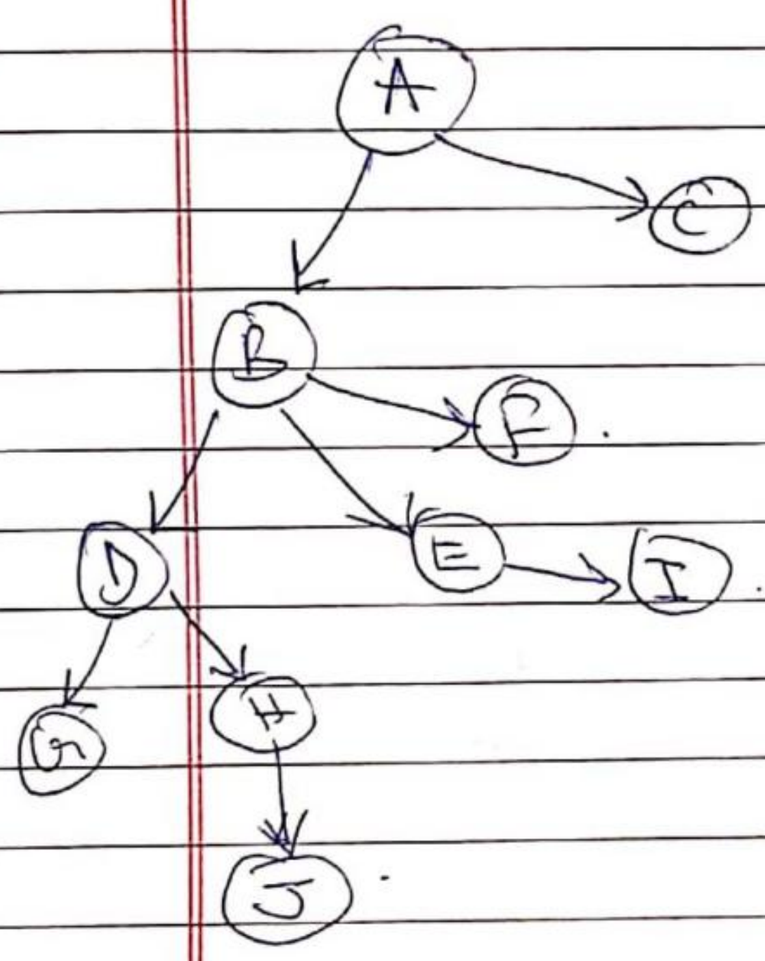
- If we wish to develop lock based protocol that are not based on 2PL, we need additional info that how each transaction will access the data.
- There are various model that can give additional information each differing in the amount of info provided.
- Our idea is to take prior knowledge about the order in which the database items will be accessed.
- We impose partial ordering - on set all data items $D = \{d_1, d_2 .. d_n\}$
if $d_i \rightarrow d_j$ then any transaction accessing both d_i and d_j must access d_i before d_j.
- P.O may be because logical or physical organisation or only because of concurrency control.

Scanned with CamScanner

Scanned by TapScanner

- aftis P.O set of all data items D will know be viewed as directed acyclic graph called database graph.

- for sake of simplicity, we follow two restrictions:-
  - ↳ will study graphs that are rooted trees.
  - ↳ will use only exclusive mode locks.



Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Tree Protocol

- Each transaction T_i can lock a data item Q will following rules:-
  - ↳ First lock by T_i may be on any data item.
  - ↳ Subsequently a data item Q can be locked by T_i only if the parent of Q is currently locked by T_i.
  - ↳ Data item may be unlocked at any time
  - ↳ Data item Q that has been locked and unlocked by T_i can not subsequently be relocked by T_i.

T_1	T_2
lockX(B)	lock(A)
lockX(D)	lock(B).
lockX(E)	lock (C)
unlock X(B).	lock (B)
lockX(I)	unlock(A).
unlock (D)	lock (D).
unlock (E)	unlock (B).
unlock (I).	lock (G).
	unlock (D).
	unlock (G)
	unlock (C)

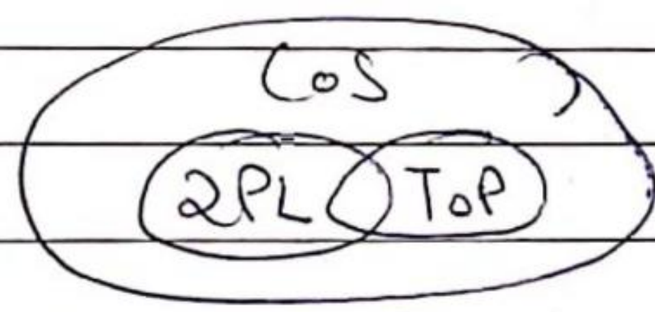
Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Properties of Tree Protocol :-

- All schedules that are legal under the tree protocol are C.S & V.S.
- Tree Protocol ensure freedom from deadlock.
- Tree protocol don't ensure recoverability and cascadelessness.
- Early unlocking is possible which leads shortest waiting time on increase concurrency
- A transaction may have to lock data items that it does not access lead to overhead waiting time and decrease in concurrency.
- transaction must know exactly what data items are to be accessed.



- Recoverability & cascadelessness can be provided by not unlocking before commit.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

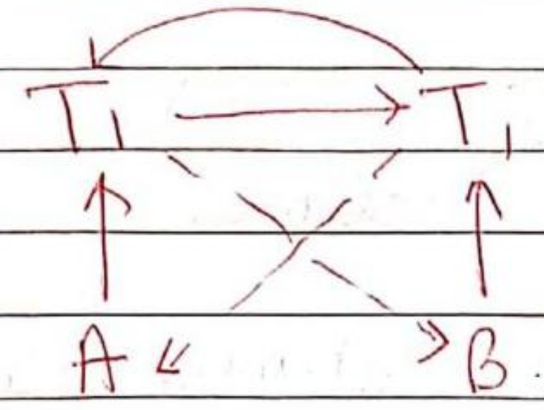
Deadlock handling

- A system is in deadlock state if there exists a set of transaction such that every transaction in the set is waiting for another transaction in the set.

- if there exists a set of waiting transaction $T_0, T_1, \ldots T_{n-1}$ such that
  $T_0 \rightarrow T_1, T_1 \rightarrow T_2, \ldots, T_{n-1} \rightarrow T_0$
  So, none transaction can progress in such situation.

- System must have a proper methods to deal with deadlocks, otherwise
  ↳ In real time system it may lead to life and memory.
  ↳ Will reduce resource utilization and increase inefficiency.

- There are two principle for dealing with deadlock problem
  - Prevention:- which ensure that system will never enter a deadlock state.
  - Detection and Recovery:-
    allow system to enter into deadlock, then try to recover.



Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Deadlock Prevention:-

- To ensure no hold & wait, each transaction locks all the data item before it begins execution eg C-2PL.

- To ensure no cyclic wait, impose an ordering of all data item and requires that a transaction look data item only in the sequence consistent with ordering. e.g tree protocal

  ↳ it is difficult to predict what data items are required.
  ↳ data item utilization will be low.
  ↳ ordering of data item may be difficult, or time taking to follow.

Wait-die: (Non-preemptive)

$T_i \rightarrow Q \leftarrow T_j$ if $TS(T_i) < TS(T_j)$ then T_i must wait (T_i is older).
if $TS(T_i) > TS(T_j)$ then T_i rollback (die).

Wound-wait: (preemptive)

$T_i \rightarrow Q \leftarrow T_j$ if $TS(T_i) > TS(T_j)$ then T_i can wait (T_i is younger).

if $TS(T_i) < TS(T_j)$ then T_j rollback.

Lock-timeouts: a time is alloted for each transaction.

Scanned with CamScanner

Scanned by TapScanner

11/02/2022

Query Language in DBMS

After designing of database, i.e. ER diagram design then converting it into relational model followed by normalization and indexing now task is how to store, retrive and modify data in database. Thought here we will be concentrating more on the query part.

Query languages:-

Languages in which user request some information from database.

Procedural query language:-

here user instruct the system to perform a sequence of operations in order to produce desired result. User tells what data to be retrieved and how to be retrieved.

Query language

- procedural (target + procedure.)
- non-procedural. (target)

Non-procedural query language:-

here user describe the desired information without giving the specific procedure for obtaining the information.

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

In practice we use RDBMS (practical implementation of relational model).

- SQL is used to write query on it.

- So Relational model is a conceptual theoritical framework and RDBMS is it's implementation.

- Relational algebra (procedural) and Relational calculus (non-procedural) are mathematical system or query language used on Relational model.

Relational Model	RDBMS.
RA, Rc.	SQL.
Algo	Code
conceptual	Reality
Theoretical.	Practical.

Query language

- procedural
  - Relational Algebra
- non-procedural
  - Relational Calculus.

↓

SQL

Structured Query Language.

Scanned with CamScanner

Scanned by TapScanner

Relational Algebra

Orion
PAGE:
DATE: / /

- Relational algebra is one of the two formal query lang. associated with relational model.

- Like any other mathematical system is defined a number of operators and use relations (tables) as operands.

- Every operator in relation algebra take one or two relations as input argument and generate a single relation as a result without a name.

- R.A do not consider dublicacy as it is based on set theory.

- In each query we describes a step by step procedure for computing the desired result. So procedural QL.

- no. use of English Keywords.

Basic / Fundamental	Derived
select (σ)	natural join ($\bowtie$)
project (π)	$\bowtie$, $\bowtie$, $\bowtie$
Union ($\cup$)	Intersection ($\cap$)
set difference ($-$)	division ($\div$)
Cartesian Product ($\times$)	
Rename (ρ)	(can be derived from fundamental)

Scanned with CamScanner

Scanned by TapScanner

12/02/2022

Orion
PAGE:
DATE: / /

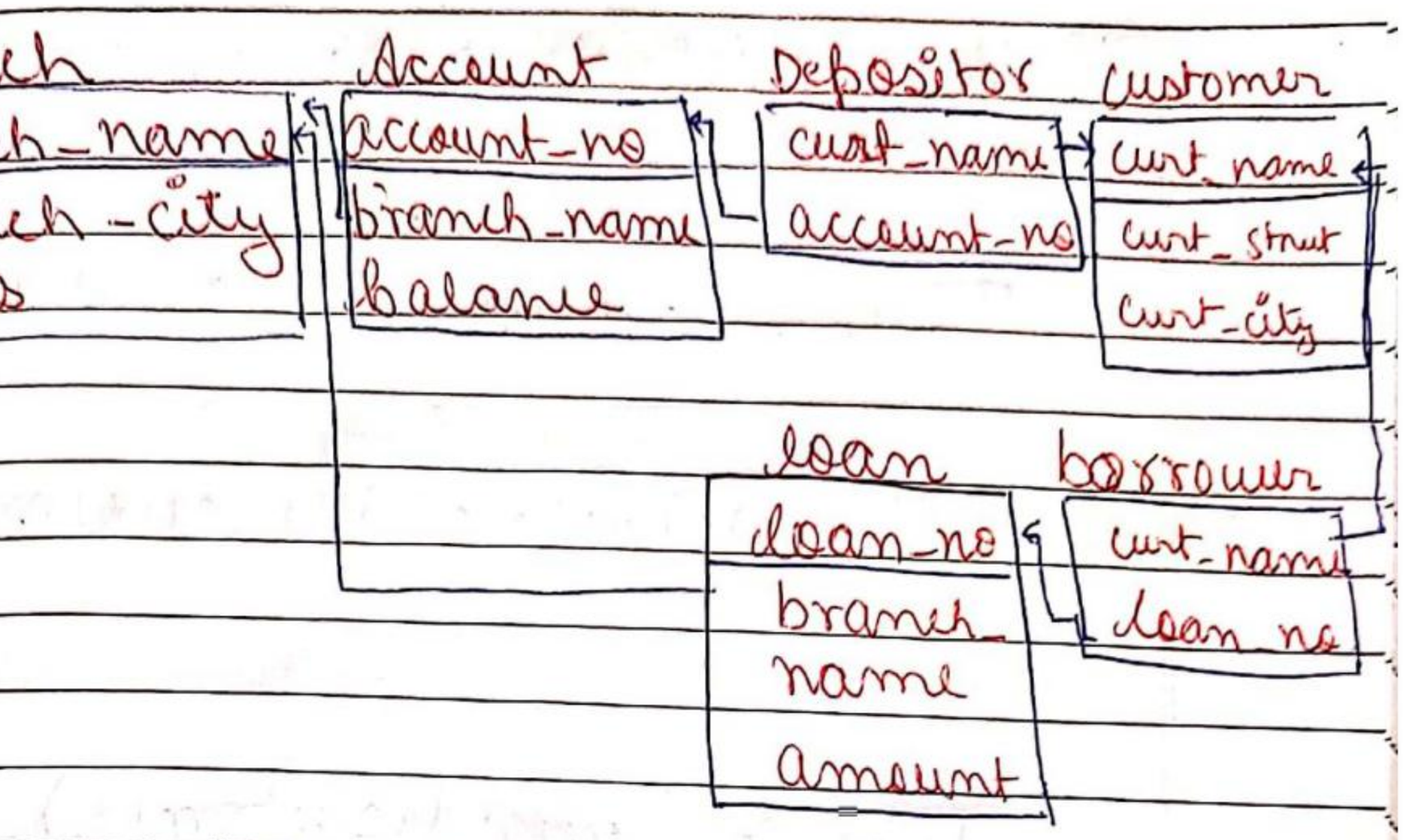
Select Operation in DBMS

projection $\Pi \rightarrow$ column
select $\sigma \rightarrow$ row

- Select is a unary operator, so can take only one table at a time.
- It is a fundamental operator.
- Main idea of select is to find those tuples/rows in a relation which satisfies a given condition.
- Syntax $\sigma_{condition/predicate}$ (table_name)
- It has same function as of where clause in SQL.
- min no of tuples selected is 0.
- max no of tuples selected is all the tuples
  [0, all]

Scanned with CamScanner

Scanned by TapScanner



Student

S_id	Name	Branch
1	A	CoS
2	B	ME
3	C	CoS
4	D	CoS
5	E	EE
6	F	EC
7	G.	EE.

$\sigma_{Branch = 'CoS'}$ (student)

Scanned with CamScanner

Scanned by TapScanner

1. Find the details of account having balance ≥ 10,000?

$\sigma_{balance \geq 10,000}$ (Account).

2. Find the details of the customer who live in delhi?

$\sigma_{cust_city = 'delhi'}$ (customer).

3. Find the details of those loans having amount ≤ 5000 and from north-delhi branch.

→ $\sigma_{branch_name = 'N_D' \wedge amount \leq 5000}$ (loan)

$\sigma_{amount \leq 5000}$ ($\sigma_{branch_name = 'N_D'}$ (loan))

or

$\sigma_{branch_name = 'N_D'}$ ($\sigma_{amount \leq 5000}$ (loan))

4. Find those branch details which are in delhi as having assets more than 10,00,000?

$\sigma_{branch_city = 'delhi' \vee assets > 10,00,000}$ (branch)

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Project (π)

- Unary operator, take one table at a time.
- Fundamental operator.
- Main idea of project is select desired columns.
- Syntax π_{column_name} (table_name)
- It works as select clause of SQL.

1. Find all branch_name of the bank?

π_{branch_name} (branch)

2. Find all account_no along with there balance?

$\pi_{account_no, balance}$ (Account)

3. Find the name of all the customers who have loan?

π_{cust_name} (borrower)

4. Find all details about branch?

(branch)

Scanned with CamScanner

Scanned by TapScanner

Orion
PAGE:
DATE: / /

Q. Find the names of all students from CoS branch?

✗ $\sigma_{branch='CoS'}(\Pi_{name}(student))$

✓ $\Pi_{name}(\sigma_{branch='CoS'}(student))$.

Q. Find those account no where balance is less than 1000?

– $\Pi_{account_no}(\sigma_{balance<1000}(Account))$

Q. Find those loan numbers which are from CoP branch with amount > 1000

– $\Pi_{loan_no}(\sigma_{branch_name='CoP' \wedge amount>1000}(loan))$

Q. Find branch_name and branch_city with assets more than 100,000?

$\Pi_{branch_name, branch_city}(\sigma_{assets>1,00,000}(branch))$

Scanned with CamScanner

Scanned by TapScanner

$A \cap B = A - (A - B)$

Orion
PAGE:
DATE: / /

Q. Find the name of customers who have a loan or an account or both?

Π_{cust_name} (Depositor)
$\cup$
Π_{cust_name} (borrower)

Q. Find the name of a branch who have accounts but not loan?

Π_{branch_name} (Account)
$- \Pi_{branch_name}$ (loan)

Q. Find the name of a customer who neither have a loan or an account?

Π_{cust_name} (customer)
$- (\Pi_{cust_name}$ (Depositor) $\cup \Pi_{cust_name}$ (borrower))

Cartestian Product (x)

- Binary operator takes two tables at a time.
- Fundamental operator.
- Allows us to combine information b/w two tables.

$A = \{1, 2\}$ $B = \{a, b, c\}$

$A \times B = \{(1,a), (1,b), (1,c), (2,a), (2,b), (2,c)\}$

R_1

A	B
1	a
2	b
3	c

R_2

C	D	F
p	101	y
q	102	z

$R_1 \times R_2$

A	B	C	D	F
1	a	p	101	y
1	a	q	102	z
2	b	p	101	y
2	b	q	102	z
3	c	p	101	y
3	c	q	102	z

Scanned with CamScanner

Scanned by TapScanner

$\Pi_{cust_name}(\sigma_{Account_no = Depost_Ano \wedge balance < 100}(Account \times Depositor))$

Q. Find those account numbers which are in Delhi?

$\Pi_{account_no}(\sigma_{branch_city='delhi' \wedge Branch.branch_name = Account.branch_name}(Branch \times Account))$

Q. Find those customer names who have a loan from a branch having assets more than 10,00,000.

$\Pi_{cust\ name}(\sigma_{assets>10,00,000 \wedge Branch.Branchname = loan.Branch_name \wedge loan.loan_no = borrower.loan_no}(Branch \times loan \times borrower))$

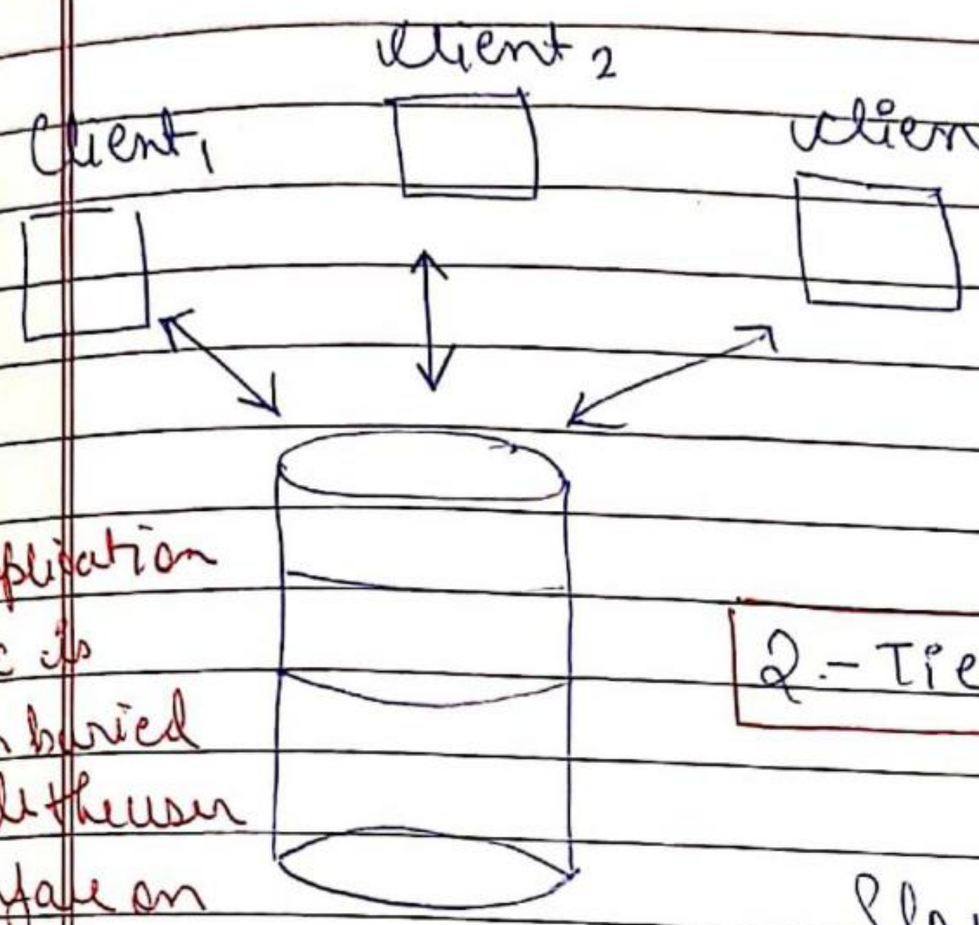
Scanned with CamScanner

Scanned by TapScanner

13/02/2022

Orion
PAGE:
DATE: / /

Difference between Two-Tier and Three-Tier database architecture.



(application logic is either buried inside the user interface on the client or within the database on the server (or both)

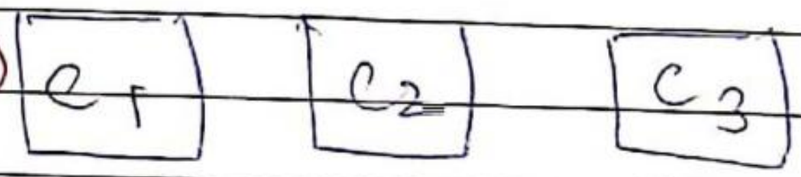
- Maintainance.

2-Tier

{ Scalability
Security.

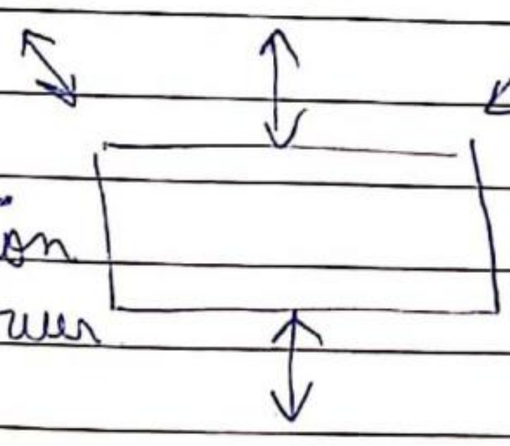
Slow

Database Server.



Client layer.

Client Application.



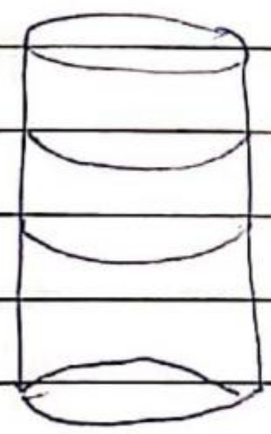
Application Server

Business layer.

{ info comes in the form of application program }

Application

{ we want that data }



Data Source.

Data Layer.

3 Tier

Past.

Scanned with CamScanner

Scanned by TapScanner

SQL

Introduction to SQL.

- Number of query languages are in use.
- The most popular and widely accepted query language is SQL (Structured Query Language).
- SQL is domain specific not general purpose, used in design and management of data held in RDBMS.
- Can do much more than just query a database, can define structure of database, modify data in database, specify security constraints and a number of other tasks.
- Based on relational algebra & tuple relational calculus.
- ~~Name IBM dates changed~~
- IBM developed original version of SQL called SEQUEL, in Project R in early 1970.
- Name dates changed to SQL ~~called~~ because of trademark issues.
- In 1986, American National Standard Institute (ANSI) and ISO publised SQL standard called SQL-86.
- next version, 1989, 1992, 1999, 2003, 2006, 2008, 2011 & most recently 2016.

Scanned with CamScanner

Scanned by TapScanner

Parts of SQL

Data Defination Language (DDL)
provide commands for defining relations schemas, deleting relations and modifing relational schemas.

CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME

Integrity
provide commands for intenity constraints on the data stored in DB.
e.g primary key can't be NULL, foreign key reference.

View Defination
Commands for defining views e.g sorting the result according to some attributes.

Authorization

Commands for access rights e.g read only, R/W grants etc.

Data Manipulation Language (DML):
provides the ability to query information from the DB and insert tuples, delete tuples and modify tuple in the database e.g insert, delete commands etc.

Scanned with CamScanner

Scanned by TapScanner

Transaction control

SQL includes commands for specifying the begining and ending of transactions e.g commit, rollback etc.

Embedded SQL and Dynamic SQL

have SQL statements can be embedded within general purpose programming language such as C, C++, Java etc.

— Z —

→ For data retrival in SQL i/p & o/p both are relations.

→ no of relation i/p to query will be at least one, but o/p will always be a single relation without any name unless specified.

→ Basic structure of SQL query consists of three clauses.

select $A_1, A_2 \ldots A_n$ (column name)
from $R_1, R_2 \ldots R_m$ (relation name)
where P (condition).

Scanned with CamScanner

Scanned by TapScanner

→ It should be noted that select and from are mandatory and if not required, then not necessary to write where.

→ SQL in general is not case sensitive.

→ SQL allows duplicates in a i/p relation as well as in the result of SQL expression.

Scanned with CamScanner

Scanned by TapScanner

Select clause :-

- is used to pick column required in result of the query out of all columns in i/p relation.

- order in which column are represented in the select clause will be same in which column will appear in the result.

- if all columns are required then * can be used.

- may also contain arithmetic expression involving operators like +, -, /, * etc

however it doesn't change the database.

select with where :-

- where clause in SQL is used to specify conditions / predictor.

- where clause can have expressions involving comparision operators.
  <, <=, >, >=, =, <> etc.

- SQL allows use of logical connectivities and, or, not, etc.

- Allows between comparision operator to simplify where clause which specify >= to some value and <= to some other value.

- Similarly now can 'not between'

Scanned with CamScanner

Scanned by TapScanner

Set Operations :-

- SQL provides set operations Union, ~~Intersect~~, Intersect, except / minus / set difference which corresponds to the mathematical set operations $\cup, \cap$ - resp.

- here union, intersect, minus automatically eliminate duplicate tuple.

- if we want to retaint duplicate tuples we write all after operator.

- Set operations are only applicable b/w two relations if they are compatible i.e having same number of columns with correspondingly same domain

e.g.: (select cname from Depositor)
union / intersect / minus.
(select cname from Borrower).

no of columns → same.

all → to retain duplication

Scanned with CamScanner

Scanned by TapScanner

Queries on multiple relations (cartesian Product)

- So far wanted on simple Query requires only one relation.
- Queries often need more than one relation for information and to do that our simple idea is cartesian product.
- C.P combine two relation into one by concatenation each tuple of first relation with every possible tuple of second relation.
- from clause by itself define C.P of relation listed.
- C.P is commutative in nature, so order doesn't matter.
- if same attribute name appears in both relation we begin the name of the relation from which attribute originally comes

Scanned with CamScanner

Scanned by TapScanner

Depositor.

C_n	A_no
C-1	X42
C-2.	PQR

Account

A_no	bal	B_n
X42	10	N-D
PQR	20	S-D

Depositor X Account.

	C_n	D.Ano	A.Acno	bal	B_n
✓	C_1	X42	X42	10	N-D
X	C_1	X42	PQR	20	S_D
X	C-2	PQR	X42	10	N_D
✓	C-2	PQR	PQR	20	S-D

redundant tuples

Select Cname from Account, Depositor
where b_name = 'N_D'
and Account.A_no = Depositor.A_no

Scanned with CamScanner

Scanned by TapScanner

Queries on multiple relations (Natural Join)

- Basically natural join work same as cartesian product but consider only those pair of tuples with the same value on those attribute that appear in the schemas of both relation.

- Notice that common attribute comes only once and prefix is also not required as we allow only same value.

- Commutative in nature.

- may lead to data loss if attributes to be match have different names, or some values occur only in one table.

Depositor

C_n	A_no.
C_1	X42
C_2	PQR.

Account.

A_no	bal	B_n
X42	10	N_D
PQR	20	S_D

D N_J A

C_n	A_no	bal	B_n
C_1	X42	10	N_D
C_2	PQR	20	S_D

```
select c_name
from Account natural join Depositor
where Bname = 'N_D'
```

Scanned with CamScanner

Scanned by TapScanner

R_1

A	B
1	a
2	b
3	c

R_2

B	C
c	p
d	q
e	r

$R_1 \times R_2$

A	R_1 B	R_2 B	C
1	a	c	p
1	a	d	q
1	a	e	r
2			

$R_1 \bowtie R_2$

A	B	C
3	c	p

inner → on
→ using

Join / inner join

- is an operation which works same as cartesian product if used alone.
- But if Join is used with keyword 'using' it provides additional functionality.
- Provides ability to explicitly change columns which must be used by join for comparison and removal of redundant tuples.
- if there are more than one tuple common between two tables but we do not want each of them to be considered then we specify which one to be considered.

```
select c_name
from Loan Join Borrower using (L_no)
where amount < 1000.
```

Join with on

- SQL join with "on" allows us use a general predicate over the relations being joined.
- This ~~predicates~~ predicate is written like a where predicate except for the use of the keyword "on" rather than "where"
- Like CoP here we must specify which attribute must be marked.
- More Powerfull as predicate of where can also be written with "On"

Scanned with CamScanner

Scanned by TapScanner

- 'On' condition can express any SQL predicate.
- improves → readibility.
- In case of outer join 'on' condition behave differently from where.

select cname
from loan Join Borrower on l.l no = B.l no.
and A < 1000.

Outer Join :-

One Problem with Natural Join is only those value that appear in both relation will manage to reach final table.
but if some data is explicitly in table one or table two then that will be lost.

→ there are three form of outer join
→ left outer join (left join)
→ right outer join (right join).
→ full outer join (full join / complete join)

(Rname → as)
Joins in SQL
- (Inner) Join :- returns records that have matching values in both tables.
- Left (Outer Join) : returns all records from the left table and matched records from the right table.
- Right (outer Join) : all right — matched left
- Full (outer Join) : returns all the records when there is a match in either left or right table.

Scanned with CamScanner

Scanned by TapScanner

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

$R_1 ⟕ R_2$

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

$R_1 ⟖ R_2$

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

$R_1 ⟗ R_2$

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

Scanned with CamScanner

Scanned by TapScanner

String Operations

- SQL specifies strings by enclosing them in single quotes 'Knowledge gate'
- Equality operation on string may or may not be case sensitive.
  e.g Most of the dbms system are case sensitive but MySQL and SQL server are not. We can change the behaviour in settings.
- ~~Mini mum~~ number of operations are available on string such as concatenation, substring, finding length of strings, converting strings to uppercase or lowercase etc.
- Exact set of operations may change from system to system.
- pattern matching can be performed on a string using "like" operator.
- two special characters -
  Percent (%) : match any substring.
  underscore (_) : matches with any character.
- matter matching is case sensitive.
- if single quote is part of a string, it must be specified by two single quotes.
- for pattern which includes special pattern we can escape character just before the special char.

Scanned with CamScanner

Scanned by TapScanner

Ordering the display of Tuples:-

- SQL gives the user some control over the order in which tuples in a relation are displayed.

- Order by clause causes the tuple in the result of a query to appear in sorted order.

- By default order by clause lists items in ascending order. We may specify asc for ascending order and desc for descending order.
- Ordering can be performed on multiple attribute i.e. if there is a tie or the first attribute we sorted according to second attribute.

- ~~find the~~

Scanned with CamScanner

Scanned by TapScanner

SQL DELETE STATEMENT

The DELETE statement is used to delete existing records in a table.

* DELETE SYNTAX *

DELETE FROM table-name where condition;

SQL GROUP BY Statement:

The GROUP BY Statement groups rows that have the same values into summary rows like "find the number of customers in each country"

The GROUP BY Statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result set by one or more columns

Scanned with CamScanner

Scanned by TapScanner

SQL HAVING Clause.

The HAVING clause was added to SQL because the WHERE keyword can't be used with aggregate functions.

SELECT column_name(s)
FROM table_name.

SQL TOP, LIMIT, FETCH FIRST or ROWNUM Clause.

The SELECT TOP clause is used to specify the number of records to return.

~~Birthday Para.~~

Scanned with CamScanner

Scanned by TapScanner

13/2/2021

SQL Joins.

A Join clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL JOINs

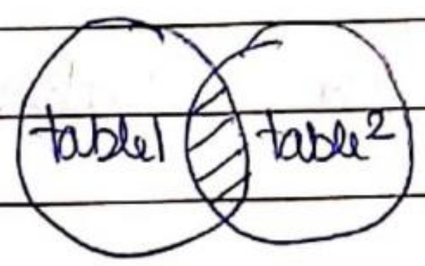
- (INNER) JOIN:-
  Returns records that have matching values in both tables.

- LEFT (OUTER) JOIN:-
  Returns all records from the left table, and the matched records from the right table

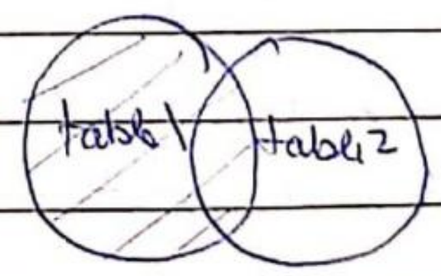
- RIGHT (OUTER) JOIN:-
  Returns all records from the right table, and the matched records from the left table.

- FULL (OUTER) JOIN:-
  Returns all records where there is a match in either left or right table.

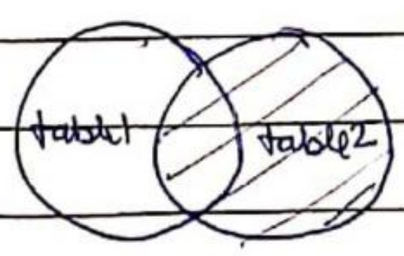
INNER JOIN



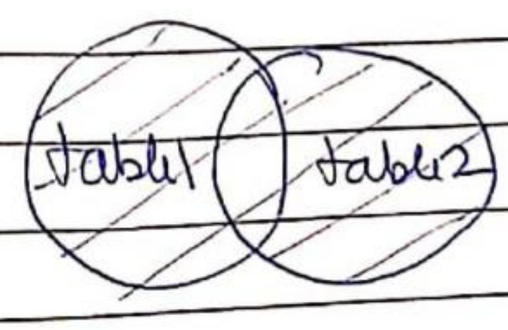
LEFT JOIN.



RIGHT JOIN.



FULL OUTER JOIN.



Scanned with CamScanner

Scanned by TapScanner

INNER JOIN EXAMPLE.

→ Select Orders.OrderID, Customers.CustomerName
From Orders
INNER JOIN Customers
ON Orders.CustomerID = Customers.CustomerID;

LEFT JOIN EXAMPLE / RIGHT JOIN

→ Select Customers.CustomerName, Orders.
OrderID
From Customers
LEFT JOIN Orders / RIGHT JOIN Orders.
ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

OUTER JOIN EXAMPLE.

→ SELECT Customers.CustomerName,
Orders.OrderID
From Customers
FULL OUTER JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

Self Join
A self join is a regular join, but the table is joined with itself.

Scanned with CamScanner

Scanned by TapScanner