

1. ER-MODEL

(1)

1. INTRODUCTION TO DBMS

Data: Any fact that could be Recorded and stored (Text, Numbers, Image)

Database: collection of Related data.

⇒ The database that contains text and Numbers is called Traditional Database.

⇒ Real-time Databases: Supermarkets, Firms.

⇒ "Datawarehouse" contains large amount of Data, the data is going to be historical.

⇒ Now a days the Databases are computerized and there must be some software that Defines, Constructs, Manipulate the Databases.

Now, the software that performs above operations on the Database is called "Database Management Systems".

⇒ $DB + DBMS = DATA BASE SYSTEMS$

2. MODELS IN DBMS

⇒ The various models that are used when designing the database is

1. High Level or Conceptual Models ⇒ NAIVE USERS ⇒ Diagrams
↓
ER-model.
2. Representational/ Implementation Model ⇒ used by programmers.
(Tables).
3. Logical Level/ physical data models ⇒ Structure, Datatype.

3. INTRODUCTION TO ER-MODEL

ER Model = Entity - Relationship Model. (Entity, Attributes, Relationships)

ENTITY: Any object in our Database.

ATTRIBUTES: The things that describe the Entities are called Attributes.
(properties that are used to describe Entities better).

RELATIONSHIPS: Association among entities.

⇒ Entity type = Schema = Heading = Intension = PERSON(Age, Name, Add)

⇒ Entity = (36, Raja, ...) = Extension

⇒ In the ER-Model we use Entity types but not the Entities.

4. ATTRIBUTES

⇒ Attributes are useful in order to describe the entities better.

The Attributes are mainly classified into

1> Simple Attributes (vs) Composite Attributes.

2> Single valued (vs) Multi valued Attributes.

3> Stored (vs) Derived Attributes.

4> Complex Attributes.

PERSON

Name = SurName, FirstName, MiddleName, LastName (Composite Attribute)

Age = Single valued Attribute

PNO = Multi valued Attribute

DOB = Stored Attribute

Age = Derived Attribute

Address = Complex Attribute — Composite Attribute
Multi valued Attribute

Derives.

5. RELATIONSHIPS (1-M)

⇒ Relationship is nothing but Association among entities.

Requirement Analysis: Every employee works for a Dep and a Dep can have many employees exactly

2. New department need not have any employees.

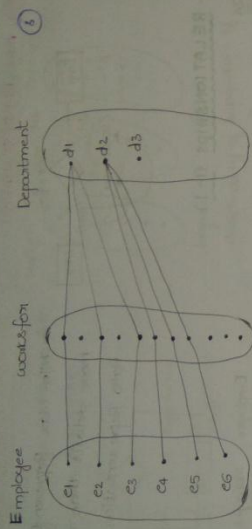
Name, Add.

(2)

Entities.

etion.

ie Attribute)



Degree: How many entities are participating in every Relationship

deg = (2) {for above diagram} = Binary Relationship

cardinality Ratio: what is the maximum no. of Relationships that an entity can participate.

⇒ cardinality of employee = 1

⇒ cardinality of Department = Many = (M) / (N)

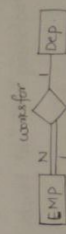
Participation or Existence: Min no. of Relationships that an entity can participate

participation = Min. cardinality

→ participation (Employee) = 1

→ participation (Department) = 0 (Newly formed department cannot have Employees).

cardinality Ratio } Structural constraints - participation/Existence



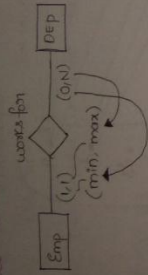
Entities: □

Relationships: ◇

Total participation (All the entities of the set are participating in Relationship)

CARDINALITY RATIO REPRESENTATION

MIN MAX REPRESENTATION



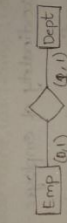
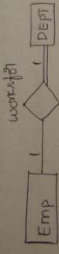
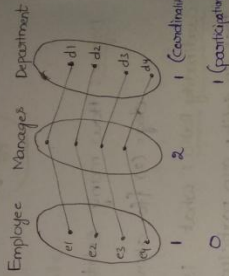
→ Min-Max Representation gives more details than Cardinality Ratio Representation.

6. RELATIONSHIPS (1-1)

RA:

Every department should have a

manager and only one manager manages a department and an employee can manage only one department (Emp should work only in 1 department)



7. RELATIONSHIPS M-M EXAMPLES

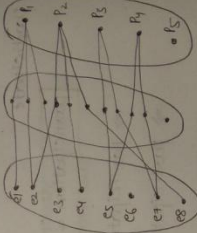
R1: Every employee is supposed to work at least on one project and he can work on many projects as well as every project is supposed to have many employees and is supposed to have at least one employee.

P1: e1, e2, e3

P2: e2, e4, e8

P3: e5, e7

P4: e1, e5, e6

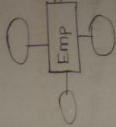


8. RECURSIVE

RA: Every entity has exactly one



9. ATTRIBUTE



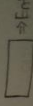
10. WEAK

⇒ whenever entity is related to the

Identif

⇒ v. Emp

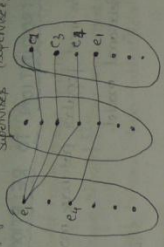
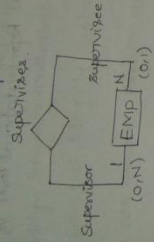
11. ER-DIA



8. RECURSIVE RELATIONSHIPS

RM: Every employee is supposed to have exactly one supervisor

Employee (supervisor)
Supervisor (supervisee)



Degree = 2

Cardinality of supervisor = N

Supervisee = 1

participation of supervisor = 0

of supervisee = 0

9. ATTRIBUTES TO RELATIONSHIPS

Station date

Station

DEPT

work shift

Emp

1

N

can shift on any side

Station date

Station

DEPT

work shift

Emp

1

N

can shift on any side

Station date

Station

DEPT

work shift

Emp

1

N

can shift on any side

Station date

Station

DEPT

work shift

Emp

1

N

10. WEAK ENTITY

whenver any entity is not having any key attribute then such an entity is called weak entity, so the weak entity should be associated to the strong entity with a relationship called as "Identifying Relationship".

Identifying Relationship:

Weak entity:

=> V. Imp point: Always the participation of weak entity in an identifying relationship should be "Total participation".

11. ER-DIAGRAM NOTATIONS

Entity

Weak Entity

Relationship

Identifying Rel.

Attribute

Key Attribute

Multi valued Attribute

Composite Attribute

Derived Attribute

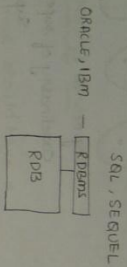
Cardinality Ratio

Total participation

2. RELATIONAL DB MODEL

1. INTRODUCTION TO RELATIONAL DATABASE

⇒ The most popular database model used at Representational level is Relational model.



2. TERMINOLOGY OF RELATIONAL DATABASE

1. Relation: Table / Relation Extension.
2. Tuple: Row (contains entities)
3. Attribute: Column
4. Domain: Set of values (associated with attributes)
5. Relational schema: Heading of the table = $R(A_1, A_2, A_3, A_4, A_5)$ / Relation
6. Degree of Relation: The no of Attributes

3. TUPLE, TUPLE VALUES AND NULL

⇒ whenever we store a Relation in a memory then it's stored in particular order.

⇒ In a Relation no two tuples can have the same values in all the attributes. (No duplicate values).

⇒ Some values of the attributes are not specified/present then we use "NULL" in that field.

⇒ For Example: Name Comprises of first name, Middle Name, last name and now a person might not have middle name so middle name = NULL

1	1	2003	2
2	1	2003	2
3	1	2003	2
4	1	2003	2
5	1	2003	2
6	2	2004	NULL



Duplicate values

NULL values

②

4. CONSTRAINTS

1. Domain constraints
 2. Key constraints
 3. Entity-Set constraints
 4. Referential constraints
- ⇒ We can view

5. CONSTRAINTS

⇒ Key - constraints

(S.NO, S)

(1, R)

(1, Ravi)

⇒ SUPER KEY

same value

⇒ Any Mini

⇒ Every Rel

⇒ Super Key

⇒ any sup

⇒ If A1 is

attribute

4. CONSTRAINTS ON RELATIONAL DB SCHEMA - DOMAIN CONSTRAINTS

- 1) Domain constraints $\Rightarrow$ Entire schema should be atomic. (7)
 - 2) Key constraints $\Rightarrow$ No two tuples should have same value.
 - 3) Entity-Integrity constraints $\Rightarrow$ Entire tuple should follow some constraints.
 - 4) Referential Integrity constraints $\Rightarrow$ applied between two tables (relations).
- $\Rightarrow$ We can view a Relation as "Flat file structure".

X		Name X		
SNO		FN	mn	LN
SNO				

Should be Atomic
Composite, Multivalued
attributes are not allowed
in Relations.

5. CONSTRAINTS ON RELATIONAL DB SCHEMA - KEY CONSTRAINTS

$\Rightarrow$ Key-constraints are also called "Uniqueness Constraints".

(SNO, SName, marks) $\Rightarrow$ SUPER KEY

(1, Ravindra, 100)

Student		Marks	
t ₁		100	
t ₂		100	
t ₁		100	
t ₂		100	

$\Rightarrow$ SUPER KEY $\subseteq$ Attributes for which no two tuples have the

same values in all the attributes.

$\Rightarrow$ Any Minimal superkey is a "Key": (SNO)

KEY = MINIMAL SUPERKEY

$\Rightarrow$ Every Relation is going to have a superkey by default and that

superkey is "set of all Attributes".

$\Rightarrow$ Any superset of a key is a superkey.

$\Rightarrow$ IF A1 is a key, and the set/Relation/table contains (A1, A2, A3, A4) attributes then the NO of superkeys that can be formed is

A1	A2	A3	A4
①	②	③	④

= $1 \times 2 \times 2 \times 2 = 8$ superkeys are possible

⇒ If we are going to have two keys for a Relation then they are going to be candidate keys. (or) If we have two minimal super keys for a Relation then they are called "candidate keys". (3)

⇒ one of the keys of candidate keys is chosen and it is going to play some important role while we insert some numbers and that key is called "Primary KEY".

$(A_1 A_2 A_3 A_4) - SK$

$(A_2 A_3 A_4) - SK$

$(A_3 A_4) - SK \text{ and Key}$

$(A_1 A_2 A_3 A_4)$

$(A_1 A_2 A_3 A_4) - SK \text{ and Key}$

Keys = $(A_3 A_4, A_1 A_2 A_4) \Rightarrow \text{candidate keys}$.

Minimal superkeys

⇒ "primary key" doesn't allow "null" values.

6. ENTITY AND REFERENCE INTEGRITY CONSTRAINTS

⇒ Entity Integrity says that no prime attribute should have null value

Employee

ENO	END	DNO	PNO
1	a	1	4

Department

DNO	NAME
1	
2	
3	

"PNO" should not be Primary key in Department Relation

PNO

1			
2			
3			

⇒ Foreign Key

1. ACTIONS UPON

⇒ The Actions

⇒ If the above

to select

⇒ while "delete"

and the

1.

2.

3.

3. COUNTING

⇒ Given a

Then the

they are
super keys

going to
and that key

(8)

1. ACTIONS UPON CONSTRAINT VIOLATIONS

→ The actions that are performed on the database are

- i) Insertion
- ii) Deletion
- iii) Update

→ If the above actions violate the constraints the default action is to reject such actions which are resulting in violation.

→ while "deleting" the constraint that get violated is "Referential Integrity" and the actions that must be taken are

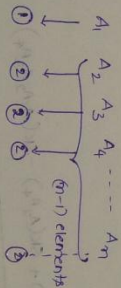
1. Ignore it (Reject the action)
2. Cascade (Delete the tuple and also delete the tuples which are being referenced by the above tuple, they should be deleted).
3. Set NULL or some other value

8. COUNTING THE NO. OF POSSIBLE - EXAMPLE - 1

→ Given a Relation R(A₁ A₂ A₃ ... A_n)

candidate key = {A₁}

Then the no. of possible super keys are



A ₁	A ₂
1	a
2	a
3	b
4	b
5	c

$$\therefore \text{Total no. of keys} = 1 \times 2 \times 2 \times 2 \times \dots (n-1) \text{ times}$$

$$= 1 \times 2^{n-1}$$

$$\boxed{\text{Total no. of SKs} = 2^n - 1}$$

usually will
primary key
dependent
element

ve null value

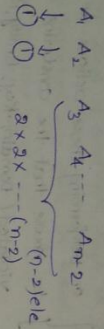
delete keys.

SK and key

9. COUNTING THE NO. OF SETS POSSIBLE - EXAMPLE 2

Relation $R = (A_1, A_2, A_3, \dots, A_n)$

Candidate key = $\{A_1, A_2\}$



$= 1 \times 1 \times 2 \times 2 \dots (n-2) \text{ times}$

$$SK = 2^{n-2} \text{ keys}$$

Now, if candidate keys = $\{A_1, A_2\}$

The no. of super keys = $SK(A_1) + SK(A_2) + SK(A_1, A_2)$

$$SK = 2^{n-1} + 2^{n-1} + 2^{n-2}$$



10. COUNTING THE NO. OF SETS POSSIBLE - EXAMPLE 3

Relation $R = (A_1, A_2, A_3, \dots, A_n)$

$CK = \{A_1, A_2, A_3\}$

No. of SKs = $SK(A_1) + SK(A_2, A_3) - SK(A_1, A_2, A_3)$

$$SK = 2^{n-1} + 2^{n-2} - 2^{n-3}$$

$CK = \{A_1, A_2, A_3, A_4\}$

No. of SKs = $SK(A_1, A_2) + SK(A_3, A_4) - SK(A_1, A_2, A_3, A_4)$

$$SK = 2^{n-2} + 2^{n-2} - 2^{n-4}$$

$CK = \{A_1, A_2, A_1, A_3\} \Rightarrow$ No. of SKs = $SK(A_1, A_2) + SK(A_1, A_3) - SK(A_1, A_2, A_3)$

$$SKs = 2^{n-2} + 2^{n-2} - 2^{n-3}$$

11. COUNTING THE NO. OF SKS POSSIBLE - Example - 4

Relation $R = (A_1, A_2, A_3, \dots, A_n)$

(11)

$$CR = (A_1, A_2, A_3)$$

$$\text{No. of SKs} = SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1A_2) - SK(A_2A_3) - SK(A_1A_3)$$

$$+ SK(A_1A_2A_3)$$

$$SK = 2^{0-1} + 2^{0-1} + 2^{0-1} - 2^{0-2} - 2^{0-2} - 2^{0-2} + 2^{0-3}$$

$$\Rightarrow R = (A, B, C, D) \quad \left\{ \begin{array}{l} \text{No. of SKs} = SK(A) + SK(B, C) - SK(ABC) \\ CR = (A, BC) \end{array} \right.$$

$$= 2^3 + 2^2 - 2^1$$

$$SK = 8 + 4 - 2 = 10$$

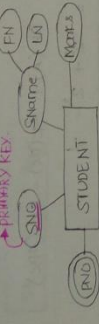
3. CONVERSION OF ER MODEL TO RELATIONAL MODEL

1. STEP 1

⇒ There are 7 steps to convert ER model (which is designed at conceptual level) to Relational model and RDBMS can be applied appropriately.

⇒ FOR EVERY ENTITY IN ER-MODEL WE HAVE TO COME UP WITH A RELATION IN RELATIONAL MODEL.

→ PRIMARY KEY



Now the Relation for this ER-diagram will be

Student	SNO	SNAME	FN	LA	MORS
---------	-----	-------	----	----	------

⇒ Every simple attribute is represented in the table.

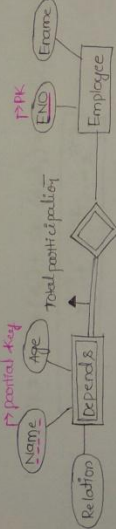
⇒ Composite attribute is further divided and atomic parts are represented in the table.

⇒ Multi valued entities are not represented in the Relation.

⇒ Represent the primary key in the ER-model in the Relation also (underline)

2. STEP 2

⇒ CONVERTING THE WEAK ENTITIES INTO RELATIONS.



⇒ Create a table/Relation for the weak entity and add all the simple Attributes

⇒ Identify the partial key in the weak entity, and also identify the primary key in the owner entity (strong entity).

⇒ Now Add primary key of the owner entity (ENO) as the foreign key of the weak entity, and make partial key of weak entity and the PK of strong entity as the PK of weak entity.

MODEL

(12)

of cascading
priority.

with A

this

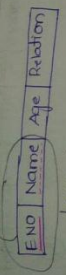
LN

es ented

of (ordered)

depends Relation

Employee Relation

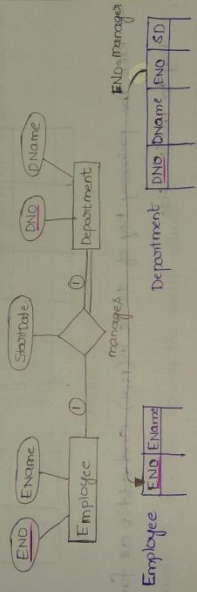


(13)

⇒ The delete operation here is "CASCADE" Delete which means if you delete any row in the owner entity with some particular "ENO" then you should delete all the entries associated with that ENO in the weak entity also.

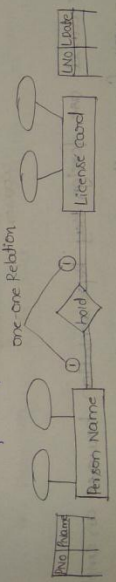
3. STEP - 3

⇒ CONVERSION OF RELATIONSHIPS INTO RELATIONS (TABLES)



⇒ Now Add the primary key of one side (one table) as the foreign key of another table, generally add the PK of (non-total participation) as the foreign key of the total participation table.

⇒ If the Relationship itself contains attributes add the attributes on the total participation side.

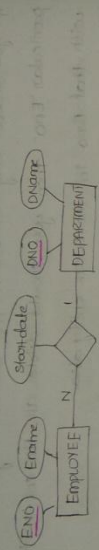


⇒ when you have total participation on both the sides and one-one Relationship exists on both sides the In both the Relations the no. of tuples will be equal and that case there is no need of adding

the primary key of one side onto the other side we can just combine the two relations into a single relation. $\{PNO, PNAME, PNO, PNO\}$ and it also takes less space. (14)

4. STEP-4

⇒ CONVERSION OF 1:N RELATIONSHIP INTO A RELATION.



Employee

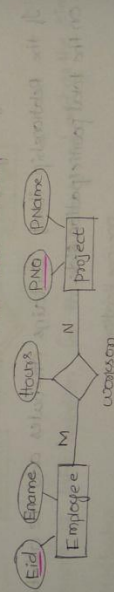
Department			
Eid	Ename	DNO	Start date

⇒ Take the primary key of 1 side in (1:N) and add it as foreign key to the 'N' side

⇒ The Attributes on the Relationships are shifted to the 'N' side.

5. STEP-5

⇒ CONVERSION OF MANY-TO-MANY RELATIONSHIP



⇒ we cannot use foreign key Representation because an employee can work for more than one project and a project can have more than one employee.
 Employee 1 workson 3
 Project 1 has more than one employee X

⇒ The solution to attributes as

Eid

$\{Eid + PNO\}$

⇒ Now the attribute Newly formed

6. STEP-6

⇒ DEALING WITH

⇒ From the multi

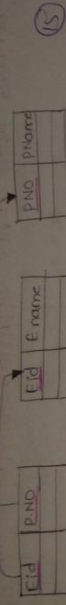
⇒ Create a

key of the

PNO	SNO
P1	1
P2	1

∴ PK = SNO

Combine $\Rightarrow$ The solution to the above problem is create a new table having and attributes as the primary keys of the participating entities.



Eid + PNO = PK of New table

$\Rightarrow$ Now the attribute on the Relationship should be added to the Newly formed table.

Eid	PNO	Hours

6-STEP-6

$\Rightarrow$ DEALING WITH MULTIVALUED ATTRIBUTES

$\Rightarrow$ For the multivalued attribute we are going to create a new table



$\Rightarrow$ Create a Relation having the multivalued attribute and primary

Key of the entity (SNO, PNO) $\Rightarrow$ SNO + PNO = PK for this table

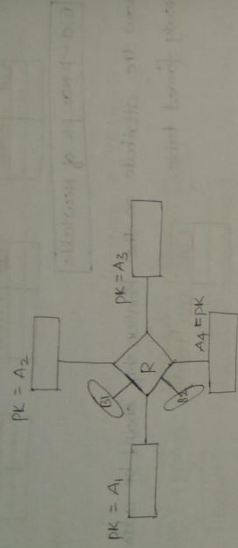
SNO	PNO

PNO	SNO	FN	LN
P1	1	a	b
P2	1		

$\therefore$ PK = SNO + PNO

7. STEP-1

⇒ DEALING WITH N-ARY RELATIONSHIPS (MORE THAN 3 ENTITIES)



⇒ Now create a new table having the attributes as primary keys of all the entities.

A1	A2	A3	A4	A5

8. SUMMARY OF ER TO RDB CONVERSION

ER-Model

Relational Model

- Entity type ————— "Entity" Relation
- 1:1 and 1:N Relationship type ————— Foreign Key (or Relation)
- M:N Relationship type ————— Relationship "relation" + 2 FK's
- N-ary Relationship type ————— Relationship "relation" + n FK's
- Simple Attribute ————— Attribute
- Composite Attribute ————— set of simple component Attributes
- Multivalued Attribute ————— Relation and Foreign Key
- value set ————— Domain
- Key Attribute ————— primary key.

9. GATE 11

Hotel Room

Lodging is made by person (x) attribute to Hotel by (y)

Sol: Let us see Rent

⇒ Let us say

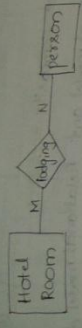
⇒ In case of the PK

the Att

10. GATE 12

Given the base
Incorrect
a) An attribute
b) An attribute
c) An attribute

9. GATE IT 2005 QUESTION ON ER-DIAGRAMS (17)



Lodging is many-many Relationship. Rent, payment to be made by person(s) occupying different hotel rooms should be added as an attribute to
 a) Hotel b) Lodging c) person d) None.

Sol: Let us say Hotel Room contains HNO, HName and if we add

HNO	HName	Rent
1	a	
2	b	

→ Many people can reside and we cannot put all the Rents in single tuple → option A X

→ Let us say the person table has pid, Pname, Rent

pid	Pname	Rent
1	a	

→ He may stay at many hotels and each hotel has its own Rent. So we cannot parallelize Rents here, ⇒ option C X

⇒ In case of many-many Relationship we create a new table having

the PK of participating entities.

pid	HNO	Rent
1	1	10000
2	2	50000

Table of the Relationship Lodging.

∴ The Attribute Rent should be on Lodging.

10. GATE 12 QUESTION ON CONVERTING ER TO RDB

Given the basic ER and Relational models, which of the following is

Incorrect

- a) An attribute of an entity can have more than one value
- b) An attribute of an entity can be composite.
- c) In a row of Relational table, an attribute can have more than one value
- d) In a row of Relational table, an attribute can have exactly one value or NULL

(15)

Primary Keys

ation)

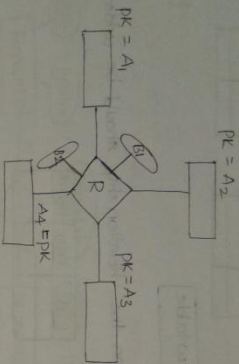
1 + 2 FK's

1 + n FK's

Attributes

1. STEP-1

⇒ DEALING WITH N-ARY RELATIONSHIPS (MORE THAN 4 ENTITIES)



⇒ Now create a new table having the attributes as primary keys of all the entities.

A1	A2	A3	A4	B1	B2
----	----	----	----	----	----

8. SUMMARY OF ER TO RDB CONVERSION

ER-Model

Relational model

Entity type ————— "Entity" Relation

1:1 and 1:N Relationship type ————— Foreign Key (or Relation)

Min Relationship type ————— Relationship "relation" + 2FK's

N-ary Relationship type ————— Relationship "relation" + nFK's

Simple Attribute ————— Attribute

Composite Attribute ————— Set of Simple component Attributes

Multi-valued Attribute ————— Relation and Foreign Key.

Value set ————— Domain

Key Attribute ————— Primary Key.

9. GATE II

Hotel Room

Lodging is a by person's attribute to Hotel of

Sol: Let us

Rent

⇒ Let us say

⇒ In case

the PK

∴ The

10. GATE I

Given the

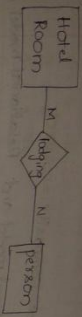
Incorrect

or An attrib

by An attrib

SS An attrib

Q. GATE IT 2005 QUESTION ON ER-DIAGRAMS



lodging is many-many Relationship. Rent, payment to be made by person(s) occupying different hotel rooms should be added as an attribute to

a) Hotel b) lodging c) person d) None.

Sol: Let us say hotel room contains HNO, HName, and if we add

Rent

HNO	HName	Rent
1	a	
2	b	

→ Many people can reside and we cannot put all the rents in single tuple → option A X

→ Let us say the person table has pid, Pname, Rent

pid	Pname	Rent
1	a	

→ He may stay at many hotels and each hotel has its own Rent. So we cannot put all the Rents here ⇒ option X

⇒ In case of many-many Relationship we create a new table having the PK of participating entities

pid	HNO	Rent
1	1	10000
2	2	5000

✓ ⇒ Table of the Relationship lodging

∴ The attribute Rent should be on lodging.

10. GATE 12 QUESTION ON CONVERTING ER TO RDB

Given the basic ER and Relational models, which of the following is

Incorrect

- a) An attribute of an entity can have more than one value
- b) An attribute of an entity can be composites
- c) An attribute of a relational table can have more than one value
- d) An attribute of a relational table can have exactly one value or NULL

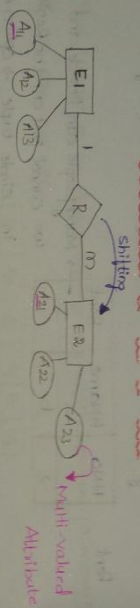
option a talks about ER model and multivalued attributes in ER-model = CORRECT

in 5th ER model attributes can be composite = CORRECT

option c talks about Relational model and Relational model doesn't allow multivalued attributes and Composite attributes = incorrect

option d is correct since there should be exactly one value in each field and can have NULL values = CORRECT

11. GATE IT Q4 CONVERSION OF ER TO RDB



Min no of tables: —

$E_1(A_1, A_2, A_3)$
 $E_2(A_2, A_3, A_4)$
 $E_3(A_2, A_1, A_3)$ } 3 tables.

12. GATE OS ON CASCADE DELETE IN CASE OF FOREIGN KEYS

The following table has two attributes A and C where A is the primary key separating with a on-delete cascade.

The set of all tuples that must be additionally deleted to preserve Referential integrity when the tuple (2,4) is deleted is:

a) (3,4) and (6,4) (b) (5,2) (7,2) (c) (5,2) (7,2) (9,5) (d) (3,4) (4,3) (6,4)

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

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

14. GATE OS QUES

F₁, F₂ - two entities

R₁, R₂ - two relations

and F₂ is many to one. what is the

pk = A₁ E₁

15. GATE IT QUES

Let R(a,b,c) and that refers to R and S:

- Shunt into R which of the following
- None of a
- All of a,b,c
- Both a,b
- Both b,c

(18)

Model demand = INCORRECT due to

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

The tuples that must be additionally deleted are (5,2) (9,2) (9,5) (7)

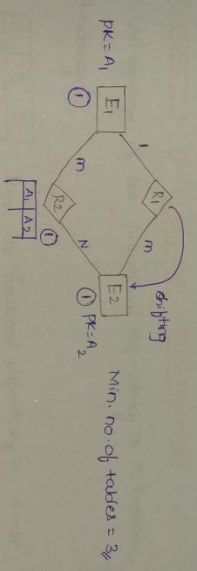
$\Rightarrow$ (2,4) is deleted $\Rightarrow$ delete the rows containing two(2)

$\Rightarrow$ Now on deleting we are deleting the rows (5,2) (9,2) because they contain
 $\rightarrow$ Now delete the rows containing (5)(9) (7)
 = (9,5) deleted

14. GATE QB QUESTION ON CONVERTING ER TO RDB

E_1, E_2 - two entities

R_1, R_2 are two relationships between E_1 and E_2 . R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 don't have any attributes of their own. What is the min no. of tables required?



KEYS
 primary:
 PK is FK
 PK is FK

15. GATE QB QUESTION ON REFERENTIAL INTEGRITY

Let $R(a,b,c)$ and $S(d,e,f)$ be two relations in which 'd' is the FK of 'a' that refers to the primary key of R. Consider the following four operations R and S:

- Insert into S
 - Delete from R
 - Delete from S
 - Insert into R
- Which of the following is true about the referential integrity constraint above?
- None of a,b,c,d can cause its violation
 - All of a,b,c,d can cause violation
 - Both a,b can cause its violation
 - Both b,c can cause its violation

801

		S	
R		d	e
a	1	1	
b	2	1	
c	3	2	
d	4	2	
e	5	3	
f	6	4	
g	10	6	

V.V. Strip

⇒ d' is depending on a' not a'

Violation (Inserting into 'S')

⇒ Inserting into 'S' and Deletion from 'R' are going to cause violations. (may cause we are not sure about it).

(20)

1. INT

⇒ If

Spe

⇒ Les

⇒ vac

⇒ Split

not

"NOE"

⇒ In

Def

2. INT

⇒ The

"Rec

4. NORMALISATION

(21)

1. INTRODUCTION TO NORMALISATION

⇒ If we hold the entire data in a single table it will take more space.

⇒ Less Redundancy

⇒ Various Anomalies will occur

Insert Anomalies
Deletion Anomalies
Update Anomalies

⇒ Splitting the tables into small tables such that our design will not contain all the above anomalies and redundancy is called "NORMALISATION".

⇒ In order to do Normalization we use the concept of Functional Dependencies (FDs) and the concept of candidate keys.

2. INTRODUCTION TO FUNCTIONAL DEPENDENCIES

⇒ The advantage of the functional dependency is it reduces "Redundancy."

A	B	C
1	2	a
2	a	b
3		
4	2	a
5	a	b

$t_1 \rightarrow$

$t_2 \rightarrow$

Here the functional dependency is $A \rightarrow BC$

⇒ for a value of 'A' you can get/derive the values of 'B' and 'C' uniquely

$A \rightarrow BC$
 $2 \rightarrow ab$

if (2, a) then
(ab) = (ab) ✓

(or) if $t_1(A) = t_2(A)$ then
 $t_1(BC) = t_2(BC)$

⇒ Initially we see that the table is in 1st Normal form.

⇒ Next 2nd NF

⇒ Next 3NF

⇒ Next BCNF

(20)

is depending on a not a dependency (inserting into) to cause

possible only
 $\Rightarrow Y$
 $2^n \Rightarrow 2^2 \times 2^2$
 $= 2^4$

of FDs are
 leaving FDs is

Now find $G_1 \supseteq F$ then, (5)
 $F: \{A \rightarrow C, AC \rightarrow D, E \rightarrow D, E \rightarrow H\}$ check whether these are derivable
 from F:

$G_1: \{A, C, D\}$
 $B \rightarrow \{A, C, D\}$
 $E \rightarrow \{E, H, I\}$
 $\therefore$ all the functional dependencies in F are covered by G_1
 $\therefore$ Both the FDs F and G_1 are Equivalent.

29. EQUIVALENCE OF FDs Example-1.

$F: \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$ check whether these two FDs are Equivalent or not?
 $G: \{A \rightarrow B, C \rightarrow D\}$

Sol
 $F \not\supseteq G \Rightarrow$ Take Each FD of G and check whether it is derivable from F:

$\Rightarrow A \rightarrow B \Rightarrow$ Now Take A^+ from F and check B is present in it
 $A^+ = \{A, B, C, D\} \therefore A \rightarrow B$ holds

$\Rightarrow C \rightarrow D \Rightarrow C^+ = \{D, C\} \rightarrow$ holds $\therefore F \supseteq G$

$G \not\supseteq F \Rightarrow$ The FDs of F are $A \rightarrow B, B \rightarrow C, C \rightarrow D$ check if they are covered by G and

in $G: A^+ = \{A, B, C\}$ $\{A \rightarrow B$ holds
 $B^+ = \{B\}$ $\{B \rightarrow C$ is not covered by $G\}$
 $\therefore G \not\supseteq F$

$\therefore$ Both the functional dependencies are not Equivalent.

TRUE

30. EQUIVALENCE OF TWO FDS EXAMPLE - 2

① $F: \{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\}$ $G: \{A \rightarrow BC, D \rightarrow AB\}$
 ② $F: \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
 $G: \{A \rightarrow BC, B \rightarrow A, C \rightarrow A\}$

③ $F \supseteq G \Rightarrow A^+ \text{ in } F = \{A, B, C\}$ ✓ $F \supseteq G$

$\Rightarrow D^+ \text{ in } F = \{D, E, A, B, C\}$ ✓

$G \supseteq F \Rightarrow A^+ \text{ in } G = \{A, B, C\}$ $A \rightarrow B$ ✓

$\Rightarrow (AB)^+ = \{A, B, C\}$ $AB \rightarrow C$ ✓

$\Rightarrow D^+ = \{A, B, D\}$ $D \rightarrow AC$ ✓

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

$\therefore G \not\supseteq F$

$\therefore$ These two FDS are not Equivalent

④ $F \supseteq G \Rightarrow A^+ \text{ in } F = \{A, B, C\}$ $A \rightarrow BC$ ✓

$\Rightarrow B^+ \text{ in } F = \{B, C, A\}$ $B \rightarrow A$ holds ✓

$\Rightarrow C^+ \text{ in } F = \{A, B, C\}$ $C \rightarrow A$ holds ✓

$\therefore F \supseteq G$

$G \supseteq F \Rightarrow \text{Now, } A^+ \text{ in } G = \{A, B, C\}$ $A \rightarrow B$ ✓

$\text{Now, } B^+ \text{ in } G = \{B, A, C\}$ $B \rightarrow C$ ✓

$\text{Now, } C^+ \text{ in } G = \{C, A, B\}$ $C \rightarrow A$ holds ✓

$\therefore G \supseteq F$

$\therefore$ Both the functional dependencies are Equivalent.

31. MINIMAL COVER

If we have a set of functional dependencies 'F' and if we could minimise it to other set of functional dependencies 'G' such that 'G' covers 'F' and 'F' covers 'G' and 'G' is minimal, then 'G' is called Minimal cover of 'F'.

PROCEDURE TO

1. Split the F

Ex: $A \rightarrow BC$

2. Find the Red

Ex: $\{A \rightarrow B, A$

3. Find the Red

Ex: $AB \rightarrow C$

4. Minimise $\{A \rightarrow$

① $A \rightarrow C, A$

② $A \rightarrow C, A$

32. MINIMAL

Minimise $\{A$

① $A \rightarrow B$

Now, $G \supseteq$

$\Rightarrow H\}$
 $C \Rightarrow H\}$

(86)

PROCEDURE TO FIND MINIMAL SET

1. split the FDs such that LHS contain single attribute.

Ex: $A \rightarrow BC, A \rightarrow B, A \rightarrow C$

2. Find the Redundant FDs and delete them from the set.

Ex: $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\} \Rightarrow \{A \rightarrow B, B \rightarrow C\}$

3. Find the Redundant attributes on LHS and delete them.

Ex: $AB \rightarrow C, A \text{ can be deleted if } B^+ \text{ contains 'A' then } B \rightarrow C$
 $B \text{ can be deleted if } A^+ \text{ contains 'B' } \Rightarrow A \rightarrow C$

4. Minimise $\{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$

① $A \rightarrow C, AC \rightarrow D, E \rightarrow A, \boxed{E \rightarrow D}, E \rightarrow H$

This production is useless (remove/delete)

This production and try to find E^+ in the remaining if it contains 'D' then " $E \rightarrow D$ " is Redundant FD.

$E^+ = \{A, E, H, C, D\} \Rightarrow "E \rightarrow D"$ is Redundant.

$\Rightarrow G\}$

② $A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H$

$\rightarrow$ This will be Redundant if C^+ contains A

$C^+ = \{C\}$

$\rightarrow$ This will be Redundant if A^+ contains C then,

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

$\therefore A \rightarrow C, A \rightarrow D, E \rightarrow A, E \rightarrow H, E \rightarrow D$

$\Rightarrow A \rightarrow CD, E \rightarrow AH$

32. MINIMAL COVER EXAMPLE - 1

Minimise $\{A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D\}$

① $A \rightarrow B, C \rightarrow B, D \rightarrow A, \cancel{D \rightarrow B}, AC \rightarrow D$

Redundant $\Rightarrow D^+ = \{D, A, B\} \Rightarrow D \rightarrow B$ is derivable

Now, $(AC)^+ = \{A, C, B\} \Rightarrow AC \rightarrow D$ is not Redundant.

e, could
such that
in 'G' is

$\Rightarrow F\}$

② $A \rightarrow B, C \rightarrow B, D \rightarrow C, AC \rightarrow D$

Now, $AC \rightarrow D$ can be deleted if A contains C (3) C contains A

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

$AC \rightarrow D$ cannot be deleted.

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

33. GATE-2013 ON MINIMAL COVER

If $\{AB \rightarrow C, D \rightarrow E, E \rightarrow C\}$ is the minimal cover of $\{AB \rightarrow C, D \rightarrow E, AB \rightarrow E\}$

$E \rightarrow C$

801

Step 1:

$AB \rightarrow C$

$D \rightarrow E$

$AB \rightarrow E$

$E \rightarrow C$

$AB^+ = \{A, B, C\}$ $\therefore$ This cannot be deleted

and must be in the minimal set

$\therefore \{AB \rightarrow C, D \rightarrow E, E \rightarrow C\}$ is not the minimal cover, The minimal cover should be $\{D \rightarrow E, AB \rightarrow E, E \rightarrow C\}$

34. LOSSLESS DECOMPOSITION

$\Rightarrow$ Mainly Normalisation is about splitting the tables i.e. decomposing the tables, so while decomposing we should see some property. One such property is "lossless decomposition".

$\Rightarrow$ when we decompose a Relation we should check that there is a common attribute in both of them, if not check what happens in below example.

<u>A</u>	<u>B</u>	<u>C</u>
a ₁	b ₁	c ₁
a ₂	b ₁	c ₁
a ₁	b ₂	c ₂



<u>A</u>	<u>B</u>	<u>C</u>
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

$\Rightarrow$ cross product

<u>A</u>	<u>B</u>	<u>C</u>
a ₁	b ₁	c ₁
a ₁	b ₂	c ₂
a ₂	b ₁	c ₁
a ₂	b ₂	c ₂

$\rightarrow$ New Tuple.

$\rightarrow$ Spurious tuple

$\therefore$ For LO



Now, if 1 ch

Now, if I choose a common Alphabet Rand only.

(5)

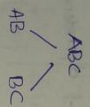


<u>AB</u>		<u>AC</u>	
a_1	b_1	a_1	c_1
a_2	b_1	a_2	c_1
a_1	b_2	a_1	c_2

Spurious Triples.

<u>A</u>	<u>B</u>	<u>C</u>
a_1	b_1	c_1
a_2	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2

LOSSY DECOMPOSITION
New Tuples that are formed but they are not present in the given table.

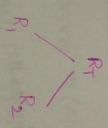


<u>AB</u>		<u>BC</u>	
a_1	b_1	b_1	c_1
a_2	b_1	b_2	c_2
a_1	b_2		

<u>A</u>	<u>B</u>	<u>C</u>
a_1	b_1	c_1
a_2	b_1	c_1
a_1	b_2	c_2

LOSSY DECOMPOSITION
⇒ The decomposition is lossy when the common attribute is key to one of the tables (A/B or B/C)

∴ For lossless decomposition.

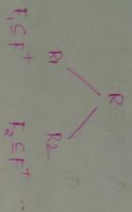


$(R1 \cap R2) \rightarrow R1$
 $(R1 \cap R2) \rightarrow R2$
The common attribute should be a key in any one of the relation.

35. FD PRESERVING

$R(A_1, A_2, A_3, \dots, A_n)$

$FD \rightarrow F^+$ [set of all functional dependencies that are applicable on R]



Now, dependency preserving means

$\Rightarrow$ The dependency preserving is not a mandatory thing

36. DECOMPOSITION - EXAMPLE 4

$R(ABC)$, $FD: \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

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

$\Rightarrow$ Now we need to check the lossless decomposition and dependency preservation

$\Rightarrow R_1(A, B)$ $R_2(B, C) \Rightarrow$ This decomposition is lossless because the common variable in $R_1, R_2 = B$ and 'B' is a key attribute in $R_2(BC)$ [$\because B \rightarrow C$].

Now, $R_1(AB)$

$R_2(BC)$

$\checkmark A \rightarrow B$

$\checkmark B \rightarrow A$

Non-Trivial Dependencies

$\checkmark B \rightarrow C$ $\checkmark C \rightarrow B$ are the Non-trivial Dependencies.

$B^+ = \{BCA\}$ $(B \rightarrow C)$

$C^+ = \{CAB\}$ $(C \rightarrow B)$

$F_1 \cup F_2 =$

$A \rightarrow B$ $B \rightarrow A$ $B \rightarrow C$ $C \rightarrow B$
 $A \rightarrow B$ $B \rightarrow A$ $A \rightarrow C$ $C \rightarrow B$
 $A \rightarrow B$ $B \rightarrow C$ $C \rightarrow A$ $\underbrace{\hspace{1cm}}_{\text{Redundant}}$

$\therefore$ The decomposition is lossless and Dependency preserving.

37. DECOMPOSITION

$R(ABCD)$

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

$D = \{AB, BC, CD\}$

$\Rightarrow$ Given $D =$

LOSSLESS decomposition

Now coming

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

$\checkmark A \rightarrow B$ $\checkmark B \rightarrow A$

$\checkmark B \rightarrow C$ $\checkmark C \rightarrow B$

$\checkmark B \rightarrow C$ $\checkmark C \rightarrow B$ $\checkmark A \rightarrow C$ $\checkmark C \rightarrow A$

Now

38. DECOMPOSITION

$R(ABCD)$

$F = \{AB \rightarrow CD\}$

$D = \{AD, BC\}$

$\Rightarrow$ The decomposition is lossless and Dependency preserving.

$\therefore$ The

37. DECOMPOSITION Example 2

$R(ABCD)$

$F: (A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A)$

$P: (AB, BC, CD)$

Let Given $D = (AB, BC, CD)$

LOSE LESS

DECOMPOSITION

$(ABC) \rightarrow (CD)$

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

$(ABCD)$

(ABC)

(CD)

Now coming to functional dependencies

$R_1(AB) \mid R_2(BC)$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

Now $F \cup F_2 \cup F_3 = A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A$ [indirectly covered]

$F_1 \cup F_2 \cup F_3 = F$

$F.D.$ is preserved (Both)

38. DECOMPOSITION Example 3

$R(ABCD)$

$F: (AB \rightarrow CD, D \rightarrow A)$

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

The common attribute is 'D' now 'D' should be a key in any one of the Relations $R_1(A, D)$ $R_2(B, C, D)$

Now, $D = \{D, A\} \xrightarrow{D \rightarrow A} A \therefore D$ is a key attribute in R_1 .

$\therefore$ The decomposition is lossless decomposition

Now,

$R_1(AD)$

$A \rightarrow D \times$

$D \rightarrow A \checkmark$

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

$R_2(BCD)$

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

$C^+ = \{C\} \times$

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

$\therefore \boxed{BD \rightarrow C}$

$(BC)^+ = (BC) \times$

$(BD)^+ = C \ BDA \ C \checkmark$

$(CD)^+ = (CDA) \times$

Now, $F_1 = D \rightarrow A$

$F_2 = BD \rightarrow C$

$F_1 \cup F_2 =$

$D \rightarrow A, BD \rightarrow C \times \left((AB)^+ = (AB) \Rightarrow \therefore \text{The FD } AB \rightarrow CD \text{ is not preserved} \right)$

$\therefore$ The decomposition is lossless and non-FD preserving.

39. DECOMPOSITION Example 4

$R(ABCDEFG_1)$

$F: \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G_1\}$

$D: (ABC, ABDE, EG_1)$

$D: \begin{matrix} \text{ABC} & \text{ABDE} & \text{EG}_1 \\ \swarrow & \searrow & \downarrow \\ \text{ABCDE} & & \text{EG}_1 \\ \swarrow & \searrow & \\ \text{ABCDEG}_1 & & \end{matrix}$

$\therefore AB \rightarrow C$ is possible $\Rightarrow$

The decomposition is lossless

Now

$R_1(ABC)$

$A \rightarrow A \times$

$B \rightarrow D \times$ (D not in table ABC)

$C \rightarrow C \times$ (C not in table ABC)

$(AB)^+ = ABCDEG_1 \Rightarrow AB \rightarrow C$

$BC \rightarrow A$

$AC \rightarrow B$

$R_2(ABDE)$

$B \rightarrow D$

$AB \rightarrow DE$

$AD \rightarrow E$

$(BE)^+ \rightarrow D$

$ABD \rightarrow E$

$ABE \rightarrow D$

$ABD \rightarrow E$

$R_3(EG_1)$

$E \rightarrow G_1$

$A^+ = A$

$B^+ = BD$

$C^+ = C$

$(AB)^+ = ABCDEG_1$

$(BC)^+ = BCDAEG_1$

$(AC)^+ = ACBDEG_1$

40. DECOMPOSITION

$R(ABCDE)$

$F: \{A \rightarrow BC, C \rightarrow D\}$

$R_1(ABCD) \ R_2(BCDE)$

Now, Given

$\therefore ABCD$

$A \rightarrow BCD \checkmark$

$B \rightarrow B \times$

$C \rightarrow D \checkmark$

$D \rightarrow D \times$

$(BC)^+ \rightarrow D$

Now $A^+ = ABCD$

$B^+ = B$

$C^+ = CD$

$D^+ = D$

$(BC)^+ =$

$(BD)^+ =$

$(CD)^+ =$

$(AB)^+ =$

$(AC)^+ =$

$(AD)^+ =$

(42)

∴ The decomposition is lossless and dependency preserving.

40. DECOMPOSITION EXAMPLE 5

R(ABCDE)

F: (A → BC, C → DE, D → E)

R₁: (ABCD) R₂: (DE).

Now, Given

(ABCD)

(DE)

Now, D⁺ = {E, D}

∴ ABCD

A → BC ✓

B → BX

C → D ✓

D → DX

(BC) → D.

Now A⁺ = ABCDE

B⁺ = B

C⁺ = CDE

D⁺ = D, E

(BC)⁺ = BCDE

(BD)⁺ = BDE

(CD)⁺ = CDE

(AB)⁺

(AC)⁺

(AD)⁺

I will not include because A itself is going to derive every thing the obviously AB, AC, AD will derive all Attributes and they will become SK's.

∴ The decomposition is lossless and dependency preserving.

(43)

R₂ (DE)

D → E ✓

E → D X E⁺ = {E}

∴ FD's =

A → BC

C → D

BC → D

D → E

= F₁ ∪ F₂ = F

(AB) ⇒ ∴ The
AB → CD is not
preserved)

{E, G₁}

A

BD

C

AB CDEG₁

BCDAEG₁

ACBDEG₁

41. DECOMPOSITION EXAMPLE-6

R(ABCDEG)

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

D: (AB, BC, ABDE, EG)

D: (AB, BC, ABDE, EG)

ABC

Now $B^+ = \{B, D\}$ B is not the key of AC

$\therefore$ The decomposition is lossy decomposition.

42. FIRST NORMAL FORM

$\Rightarrow$ The process of removing the redundancy is Normalization

A	B	C	D	E	F
1	2	3	4	5	6

$\Rightarrow$ our ultimate aim is to reduce table to BCNF, it's known that by the time you convert the table into BCNF there won't be any Redundancy (Redundancy = 0%)

$\Rightarrow$ 1NF $\rightarrow$ 2NF $\rightarrow$ 3NF $\rightarrow$ BCNF

$\Rightarrow$ 1NF says that the table has to be flat

$\Rightarrow$ By default Every Relational Database is in First Normal form



$\Rightarrow$

Student			
SNO	FN	LN	

phone no

SNO	PNO
-----	-----

$\Rightarrow$ No

43. SECOND

$\Rightarrow$ Let us see the func

$\Rightarrow$ Now, A

(AB)⁺ =

Now,

A
1
2
1
2
1
2
3
4
5

$\downarrow$
This is the

(B $\rightarrow$ C)

phone no

Find B

A

(44)

→ No multivalued and Composite values are allowed in 4NF

2nd SECOND NORMAL FORM INTRODUCTION

Let us assume we have a table having attributes ABC and the functional dependencies that are allowed are $AB \rightarrow C, B \rightarrow C$

Now, $AB \rightarrow C \left\{ \begin{array}{l} A^+ = A \\ B^+ = BC \end{array} \right\}$ with one attribute, key is not possible
 $C^+ = C \therefore$ Try with 2 attributes.

$(AB)^+ = (AB)^+ \therefore (AB)$ has the capacity to become Candidate Key.

Now, $AB \rightarrow C \rightarrow$ say that AB (combination) can determine 'C'
 $B \rightarrow C \rightarrow$ say that B (itself) can determine 'C' uniquely

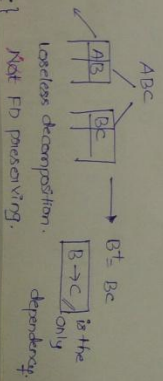
A	B	C
1	a	c1
2	a	c1
1	b	c2
2	b	c2
1	c	c3
2	c	c3
3	c	c3
4	c	c3
5	c	c3

$\Rightarrow$ The 2NF says that if your candidate key is containing more than one attribute then a part of the key should not determine anything else.

$\Rightarrow$ 2NF is based on Full Functional Dependency
 $\Rightarrow$ partial dependency is allowed (A part of the key determines some thing)

This table is not in 2NF because there is a partial dependency
 $(B \rightarrow C)$ in the candidate key $(AB \rightarrow C)$ so we should eliminate partial dependency.

Find $B^+ = \{BC\}$
 $A^+ = \{A\}$
 $\times A \nrightarrow B$
 $\times B \nrightarrow A$
 $\times A \nrightarrow C$
 $\times B \nrightarrow C$
 $\times AB \nrightarrow BC$
Not FD preserving.



form

known
one want

tion

0	1
1	0

44. 2NF Example 1

$R(ABCD)$

FD: $\{AB \rightarrow C, B \rightarrow D\}$ what is the highest Normal Form satisfied?

⇒ By default Every Relation will be in 1NF

⇒ Now find all the candidate keys

$$A^+ = A$$

$$B^+ = BD$$

$$AB^+ = ABCD$$

∴ (AB) is the candidate key.

Now, the three attribute candidate keys are possible and they should not contain (AB), if they include then it will become SuperKey.

$$(ACD)^+ = (ACD)$$

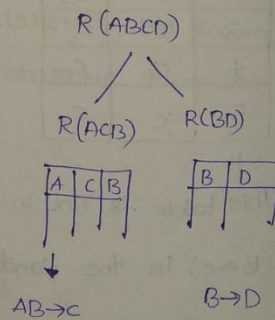
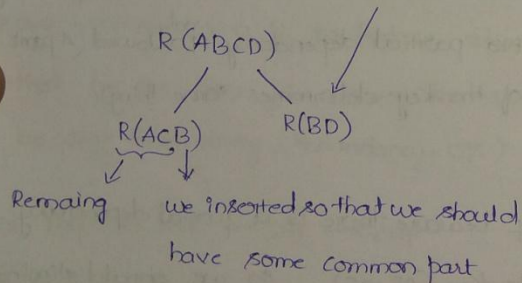
$$(BCD)^+ = (BCD)$$

$$\therefore (AB) \rightarrow ABCD$$

$B \rightarrow D$ → partial dependency

Now, find $A^+ = A$

$B^+ = BD$



FD preserving too.

Losses decomposition.

45. 2NF

$R(AB CDE)$

$$AB \rightarrow C$$

$$BD \rightarrow EF$$

$$AD \rightarrow GH$$

$$A \rightarrow I$$

$$H \rightarrow J$$

Now, find of left to all the

$$(AB)^+ =$$

$$R(AB CDE)$$

$$AB \rightarrow C$$

$$A^+ = (A, I)$$

$$A \rightarrow I$$

$$A \rightarrow I$$

46. 2NF

$R(ABCDE)$

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

The part

Now,

$$R(AB CDE)$$

46

45. 2NF EXAMPLE 2

$R(AB CDEFGHIJ)$

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$H \rightarrow J$

candidate Key = ABD.

$(ABD) \rightarrow (ABCDEFGHIJ)$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$AB \rightarrow C$

} partial Dependencies

and

some

Now, find all the closures of all partial Dependencies (find closures of left hand side) and try to create a new table by taking away all the attributes which are determined by such dependencies

$(AB)^+ = (ABC)$

$(BD)^+ = (BDEF)$

$(AD)^+ = (AD, GH, I, J)$

$R(AB \cancel{C} DEFGHIJ)$

$R(A \cancel{B} \cancel{C} DEFGHIJ)$

$R(A \cancel{B} \cancel{C} DEFGHIJ)$

$AB \rightarrow C$

A	B	C
---	---	---

$BD \rightarrow EF$

B	D	E	F
---	---	---	---

$AD \rightarrow GH$
 $H \rightarrow J$

A	D	G	H	I	J
---	---	---	---	---	---

$A^+ = (A, I)$

ABD

A	B	D
---	---	---

A	I
---	---

$A \rightarrow I$

$R(BD)$

B	D
---	---

$B \rightarrow D$

ving too.
decomposition

46. 2NF EXAMPLE 3

$R(ABCDE)$

candidate Key = (AC)

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

The partial dependencies are $A \rightarrow B$
 $C \rightarrow D$

Now, $A^+ = \{A, B, E\}$

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

$R(A \cancel{B} \cancel{C} DE)$

$R(A \cancel{B} \cancel{C} DE)$

$A \rightarrow B$

$B \rightarrow E$

A	B	E
---	---	---

$C \rightarrow D$

C	D
---	---

AC

A	C
---	---

Now, if the candidate key for a table is a single attribute and that is the only candidate key then the relation is in 2nd NF.

(18)

47. THIRD NORMAL FORM INTRODUCTION

3NF: NO Transitive Dependencies

Transitive Dependency: Non prime attribute Transitivity depending on the key.

R(ABC)

FD: $\{A \rightarrow B, B \rightarrow C\}$ CK = $\{A\} \Rightarrow$ No partial Dependencies.

Prime attribute: The attribute which is a part of Key.

Non-prime attribute: The attribute which is not a part of the key.

$A \rightarrow B$

Transitive Dependency

Now, $B \in \{B, C\}$

R(A, B, C)

$A \rightarrow B$

$B \rightarrow C$

Lossless Decomposition

FD preserving

48. 3NF Example 1

R(ABCDE)

FD: $\{AB \rightarrow C, B \rightarrow D, D \rightarrow E\}$

CK = $\{AB\}$

Now, partial dependencies are $B \rightarrow D$

$B \in \{B, D\}$

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

B	D	E
---	---	---

$B \rightarrow D$

$D \rightarrow E$

A	B	C
---	---	---

$AB \rightarrow C$

3NF

Here $D \rightarrow E$ is in 3NF.

$D \rightarrow E$ is

$\Rightarrow$ Now $D \in \{D, E\}$

R(B)

B

$B \rightarrow D$

49. 3NF Example

R(ABC)

FD: $\{AB \rightarrow C, C \rightarrow$

V.V.V.N. Imp.

partial depend

Here $C \rightarrow A$

$\therefore C \rightarrow A$ is

Ex

R(C)

CK

$A \rightarrow$

$A \rightarrow$

$A \rightarrow$

$B \rightarrow$

attribute and NF. The table (BDE) is not in 3NF.

$D \rightarrow E$ is the Transitive Dependency present in table BDE

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

$R(BDE)$

B	D	E	A	B	C
---	---	---	---	---	---

$B \rightarrow D$ $D \rightarrow E$ $AB \rightarrow C$ 3NF

49. 3NF EXAMPLE 2

$R(ABC)$

FD: $\{AB \rightarrow C, C \rightarrow A\}$

Now, Candidate Key = $(B) = (AC)$

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

$(AB)^+ = \{ABC\} \checkmark$ $\therefore$ Candidate Keys

$(BC)^+ = \{BCA\} \checkmark$ $= (AB)(BC)$

$\therefore$ All are prime attributes

V.V.V. Imp.

partial dependency = part of key $\rightarrow$ Non-prime attribute $\Rightarrow$ prime attribute $\rightarrow$ Non-prime attribute

Here $C \rightarrow A$ is not partial dependency because $C \rightarrow$ prime attribute (A)

$\therefore C \rightarrow A$ is not partial dependency.

Ex:

$R(ABCDEF)$

CK: ABC

$A \rightarrow D$ = partial dependency

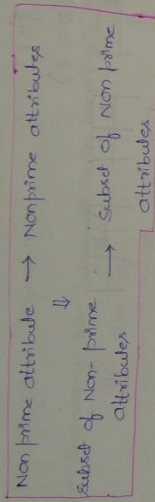
$A \rightarrow B$ $A \rightarrow C$ $B \rightarrow C$ } = not partial dependencies

$C \rightarrow E$

3NF

In the above table there are no partial dependencies $\therefore$ The Relation is in 3NF

Transitive dependency



$R(ABCEFG)$ $\Rightarrow$ $D \rightarrow E$ } Transitive Dependencies
 $CK = ABC$ $E \rightarrow F$

$B \rightarrow C$ } Not Transitive dependency
 $D \rightarrow C$

$\therefore$ The above Relation does not contain Transitive dependency

$\Rightarrow \therefore$ The Relation is in 3NF

50. FORMAL DEFINITION OF 3NF

A Relational schema R is in 3NF only if every Non-trivial FD $X \rightarrow Y$

either 1) X is a superkey

or 2) Y is a prime attribute

Which of the following is allowed in 3NF?

- proper subset of CK $\rightarrow$ Non prime (partial dependency)
- Non-prime $\rightarrow$ Non-prime (Transitive dependency)
- proper subset of CK + Non-prime $\rightarrow$ Non-prime (Transitive dependency)
- proper subset of CK $\rightarrow$ proper subset of other CK (Not a PP, PD) ✓

51. 3NF E

$R(ABCEFG)$

Now, candidate

Now, (A

Now,

Now, A^+

$A \rightarrow C$

$C \rightarrow D$

Transitive Dependency

Now, C

52. BCNF

A Relational

Functional d

of R , i.e. d

superkeys.

$R(ABC)$ FD

$\Rightarrow$ Candidate

Q. The Relation is in 2NF

3NF Example 3

R(ABCDE) F: $\{A \rightarrow C, C \rightarrow D, B \rightarrow E\}$

Now, candidate key = $(AB)^\pm$ $(ABCE)^\pm$

Now, $(AB)^\pm$ $(ABFCDE)^\pm$ $\therefore (AB)$ is the candidate key

Now, $A \rightarrow C$
 $B \rightarrow E$ are the partial dependencies

Now, $A^+ = \{A\}$
 $F \rightarrow D$

A	F	C	D
---	---	---	---

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

Transitive Dependency.

B	E	AB
---	---	----

$B \rightarrow E$

$\therefore$ The Relation is in 2NF

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

A	C	F
---	---	---

$A \rightarrow C$

C	D
---	---

$C \rightarrow D$

B	E
---	---

$B \rightarrow E$

A	B
---	---

2NF

52. BCNF INTRODUCTION

A Relational schema 'R' is in BCNF if whenever a Non-trivial functional dependency $X \rightarrow Y$ holds in R, then 'X' is a superkey of R. i.e. determinants of all functional dependencies should be superkeys.

R(ABC) FD $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

$\Rightarrow$ candidate keys = $\{A, B, C\}$

A	B
---	---

$A \rightarrow B$

B	C
---	---

$B \rightarrow C$

C	A
---	---

$C \rightarrow A$

BCNF

53. BCNF Example 1

$R(ABC) \quad F: \{AB \rightarrow C, C \rightarrow B\}$

$\Rightarrow$ The candidate key = $(A)^+$ $(ABC)^+$

$A^+ = \{A\}$

$\therefore (AB)^+ = \{ABC\}$
 $(AC)^+ = \{ACB\}$

$\therefore (AB) (AC)$ are candidate keys $\Rightarrow A, B, C$ are prime attributes

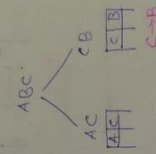
$\therefore$ There are no partial dependencies in the table

$\therefore$ The table is in 3NF because the candidate key contains all the attributes (all the attributes in the relation are prime attributes)

Now, Acc to the BCNF the LHS should be a superkey

$\Rightarrow (AB)^+ = \{ABC\}$

$C^+ = \{C, B\}$ Not superkey.



$\Rightarrow$ Lossless Decomposition
 $\Rightarrow$ Not FD preserving
 $(AB \rightarrow C)$ is lost

BCNF

54. BCNF Example 2

$R(ABCDEFGH I J)$

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

candidate key = $(AB)^+$ $(ABCDEFGHIJ)^+$

Now, $(AB)^+ = \{ABC, F, GH, I, J\} \Rightarrow A, B$ are only the prime attributes

Now, the partial dependencies are $\begin{cases} A \rightarrow DE \\ B \rightarrow F \end{cases}$

$A^+ = \{AB, DE, I, J\}$ $B^+ = \{B, F, GH\}$

Now, the Relation

$R_1(ABC) \quad (ABE)$
 $F: \{A \rightarrow DE\}$
 $F^+ = \{A, DE\}$

$\Rightarrow$ To check BCNF on a table

$\therefore$ The Relation

$R_1(ABC) \quad (ABE)$
 $A \rightarrow DE$
 $A^+ = \{ADE\}$
 $A = SK$

55. BCNF Ex 3

$R(ABCDEFGH I J)$

candidate key

Now, $(AB)^+$

$\Rightarrow$ Now $B \rightarrow F$
 $A \rightarrow$

Now,

Transitive

52

Now the Relation

$R(ABCDEF GHIJ)$

53

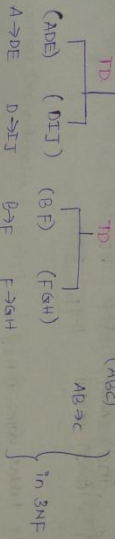
$R_1(ABDEIJ)$ $R_2(GFIH)$ $R_3(ABC) \Rightarrow$ NO PDA $\Rightarrow$ 2NF

$A \rightarrow DE$ $B \rightarrow F$ $AB \rightarrow C$

$\Rightarrow A, B, C$

are attributes

$D \rightarrow IJ$ $F \rightarrow GH$



$\Rightarrow$ To check BCNF definition every LHS of the dependency applicable on a table should be a Superkey.

$\therefore$ The Relations that we get are

$R_1(ABDEIJ)$	$R_2(GFIH)$	$R_3(ABC)$
$A \rightarrow DE$	$B \rightarrow F$	$AB \rightarrow C$
$D \rightarrow IJ$	$F \rightarrow GH$	
$A^+ = \{ABE\}$	$D^+ = \{DIJ\}$	$F^+ = \{FGH\}$
$A = SK$	$D = SK$	$F = SK$
		$AB = SK$

55. BCNF Example

$R(ABCODEFGHIJ)$ FD: $\{ AB \rightarrow C, AB \rightarrow D, D \rightarrow EF, A \rightarrow GH, H \rightarrow IJ \}$

Candidate key = $(AB)^+$ $(ABCODEFGHIJ)$

Now $(AB)^+ = (ABCODEFGHIJ)$ (AB) is the Candidate key

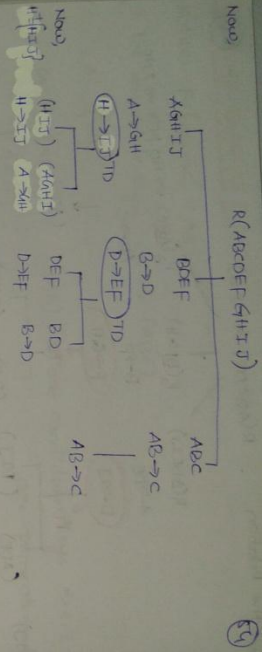
$\Rightarrow$ All prime attributes $\Rightarrow A^+ = \{ABCDEFGHIJ\}$ $B^+ = \{ABCDEFGHIJ\}$

$\Rightarrow$ Now $B \rightarrow D$, is the partial dependency $\Rightarrow B^+ \{B, D, E, F\}$ $B \rightarrow D$ is possible

Now, $(ABCODEFGHIJ)$

$R_1(BCDEF)$	$R_2(A GHIJ)$	$R_3(ABC)$	Both tables and FD preserving
$B \rightarrow D$	$A \rightarrow GH$	$AB \rightarrow C$	
Transitive $D \rightarrow EF$	$H \rightarrow IJ$ TD.		

Now,



The tables are 1) H, I, J

- 2) A, G, H, I
- 3) D, E, F
- 4) B, D
- 5) A, B, C

56. BCNF Example 4

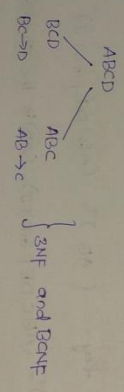
R(ABCD) FD: $\{A \rightarrow B, C \rightarrow D\}$

candidate key: $(AB)^+$ $(AC)^+$ $(BC)^+$

$(AB)^+$ $\{A, B, C, D\}$ $(AC)^+$ $\{A, B, C, D\}$ is the candidate key

Now, There are no partial dependencies

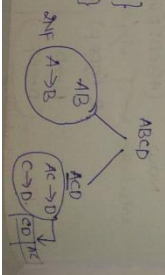
Now, $BC \rightarrow D$ is the Transitive dependency, BC is not a superkey



57. BCNF Example 5

R(ABCD) FD: $\{A \rightarrow B, C \rightarrow D\}$ CK: $\{A\}$ $\{C\}$ are prime attributes

partial dependency: $A \rightarrow B$ $C \rightarrow D$ $AC \rightarrow D$ $C^+ = \{C, D\}$



if a table is in BC

58. BCNF Example

R(AB, CD, E, F)

candidate key

Now, partial

59. BCNF

R(AB, CDE)

Now, To check dependency

- $A \rightarrow BC$
- $CD \rightarrow E$
- $B \rightarrow D$
- $E \rightarrow A$

Q. If a table contains only 2 attributes then the Relation will definitely be in BCNF.

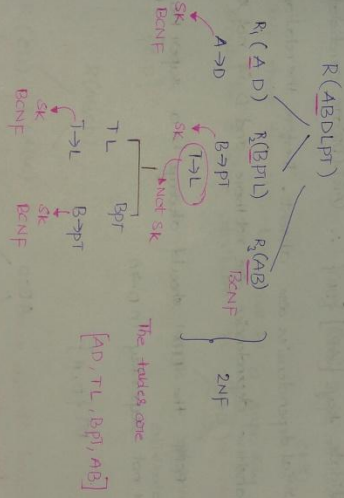
BCNF Example - 6

$R(ABDLP)$ FD: $B \rightarrow PT, T \rightarrow L, A \rightarrow D$

Candidate key = $(AB)^+$ $(ABDLP)^+ \rightarrow (AB)^+ = (ABDLP)^+$

$[CK=AB]$

Now, partial dependencies = $B \rightarrow PT$
 $A \rightarrow D$ } $B^+ (B, P, T)$
 $A^+ (A, D)$



5th BCNF Example 3

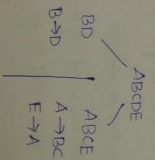
$R(ABCE)$ FD: $A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A$ Candidate key = $\{A, E, C, D\}$

All the attributes are prime

$\Rightarrow$ 3NF

Now, To check whether the Relation is in BCNF check the functional dependencies and find whether the LHS is a super key or not

- $A \rightarrow BC \Rightarrow A^+ = (ABCE) \Rightarrow A$ is SK
- $CD \rightarrow E \Rightarrow (CD)^+ = (CDEAB) \Rightarrow CD$ is SK
- $B \rightarrow D \Rightarrow (B)^+ = (B, D) \Rightarrow$ Not SK
- $E \rightarrow A \Rightarrow (E)^+ = (E, A, B, C, D) = E$ is SK



upon key

63. BCNF

R(ABCD)

The cand

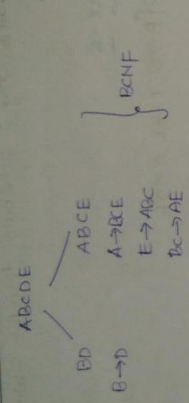
Now

64. BCNF

R(ABCD)

The cand

Now



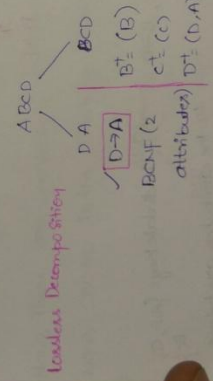
61. BCNF EXAMPLE 8 - PART 1 AND 62. PART 2

R(ABCD)

FD: $\{AB \rightarrow CD, D \rightarrow A\}$

- $\Rightarrow$ candidate key = $\{AB\} \{DE\} \therefore (AB)(DE)$ are candidate keys.
- $\Rightarrow$ partial dependencies are absent. Therefore the relation is in 2NF
- $\Rightarrow$ Transitive dependencies are not there and 'D' is not superkey, but A is prime attribute.
- $\Rightarrow$ For BCNF the LHS should always be a superkey, and here 'D' is not superkey in $D \rightarrow A$

$D \rightarrow \{D, A\}$



Lossless Decomposition

$\checkmark D \rightarrow A$
 BCNF (2 attributes)
 $B \rightarrow D$
 $D \rightarrow B$
 $(BC)^+ = (BC)$
 $(BD)^+ = (BDAC) \Rightarrow BD \rightarrow C \checkmark$
 $(CD)^+ = (CDA)$

Now, find if $AB \rightarrow CD$ is preserved using $D \rightarrow A$ and $BD \rightarrow C$
 $\Rightarrow (AB)^+ = (AB)$ $AB \rightarrow CD$ is not preserved.

$\therefore$ Dependency preserving failed.

63. BCNF Example 9

$R(ABCD) \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow D\}$

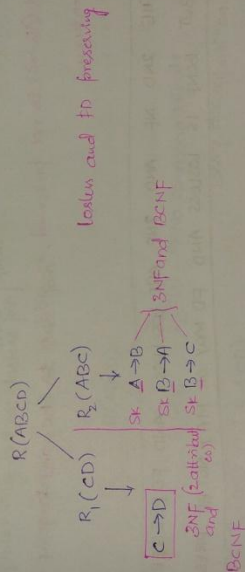
The candidate key = $()^+ = (ABCD)$

$\therefore$ Now, $A^+ = \{A, B, C, D\}$ } candidate keys
 $B^+ = \{B, A, C, D\}$ }

$C^+ = \{C, D\}$ $(CD)^+ = (CD) \times$
 $D^+ = \{D\}$

Now, A, B are prime attributes $\Rightarrow$ Now, partial dependencies are not present (only single attribute CKs).

Now for 3NF check the RHS, LHS must be a super key or the RHS should be a prime attribute in $(C \rightarrow D) \Rightarrow$ violating 3NF
 Not SK Not prime attribute
 Now, $C^+ = \{C, D\}$



64. BCNF - Example 10 PART 1 AND 65. PART 2

$R(ABCDE) \text{ FD: } \{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A\}$

candidate key = $(B)^+ = (ABCDE)$

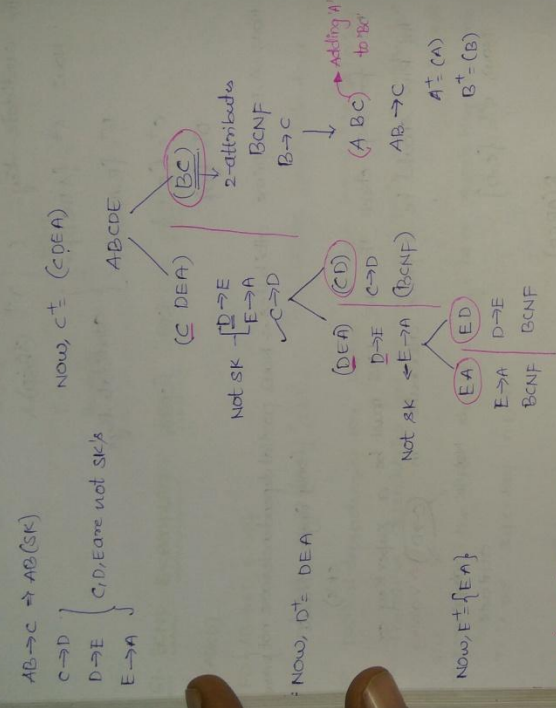
$(B)^+ = \{B\}$ Now $(AB)^+ = \{A, B, C, D, E\}$ } All the attributes are prime attributes $\Rightarrow R$ is in 3NF
 $(CD)^+ = \{B, C, D, E, A\}$
 $(DE)^+ = \{B, D, E, A, C\}$
 $(EB)^+ = \{B, E, A, C, D\}$

65

keys.
 is in 2NF
 super key but
 A is prime attr.
 f, and here

$BD \rightarrow C$ ✓
 and $BD \rightarrow E$

For BCNF, every LHS of the functional dependency should be a superkey



But $(AB \rightarrow C)$ is not preserved, so add 'A' to 'BC' and preserve the dependency.

THE 2ND NF AND 3NF ARE LOSELESS, AND Fd PRESERVING
BUT BCNF IS LOSELESS AND Fd MAY OR MAY NOT BE PRESERVED.

66. GATE QUESTION ON NORMALISATION 1

GATE-94

State True or False with Reason. There is always a decomposition into BCNF that is lossless and dependency preserving.

Ans: FALSE (Refer above Note) (we cannot guarantee dependency preserving)

GATE-98
which NF
database
a) 2NF
GATE-99
Let R =
 $C \rightarrow F, E$
RABC
GATE-01
consider
 $C \rightarrow D, T$
a) FD P
b) lossless
c) FD P
d) Not

F₁: C

57

Q1E-98

which Normal form is considered adequate for Normal Relational Database design?

- a) 2NF b) 3NF c) 4NF d) 3NF (Actual ans is BCNF)

Q1E-99

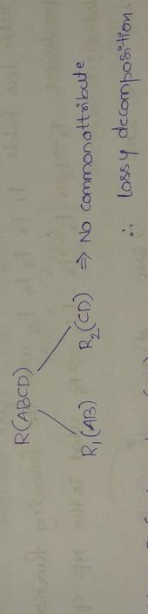
Let $R = (ABCDE)$ be a relation scheme with the following dependency $C \rightarrow F, E \rightarrow A, EC \rightarrow D, A \rightarrow B$ what is the key of R ?

$R(ABCDE) \Rightarrow (EC)^+ = (ABCDE)^+ \Rightarrow (EC)^+ = \{E, C, A, B, D\}$
 $\therefore EC$ is the key.

Q1E-01

consider the schema $R(ABCD)$ and functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Then the decomposition of R into $R_1(AB)$ and $R_2(CD)$ is

- a) FD preserving and lossless
 b) lossless but FD preserving fails
 c) FD preserving but not lossless join
 d) Not FD preserving and not lossless join



Now, $R_1(AB)$ $R_2(CD)$
 $A^+ = (A, B)$ $C^+ = (C, D)$
 $B^+ = (B)$ $D^+ = (D)$
 $F_1: (A \rightarrow B)$ $F_2: (C \rightarrow D)$

$F_1 \cup F_2 = F$ $\therefore$ Dependency preserving

Q1E-02

58

Let $R = (ABCD)$ be a relation scheme with the following functional dependencies $A \rightarrow B, C \rightarrow D$. Then the decomposition of R into $R_1(AB)$ and $R_2(CD)$ is

a) lossless and FD preserving

b) lossy and FD preserving

c) lossless and not FD preserving

d) lossy and not FD preserving

Ans: c

Adding A to B^+

$\rightarrow C$

$A^+ = (A)$

$B^+ = (B)$

reverse the

PRESERVING
BE PRESERVED

is a decomposition
eg. dependency
is preserved

GATE-02

Relation R with an associated set of FDs (F) is decomposed into BCNF. The Redundancy (consisting out of functional dependency) in the resulting set of Relations is

- a) Zero BCNF = (0% Redundancy)
- b) More than zero but less than of 3NF decomposition
- c) Proportional to the size of F^+
- d) Indeterminate

67. GATE QUESTION ON NORMALISATION 2

GATE-05

which one of the following statements about Normal forms is false?

- (a) BCNF is stricter than 3NF (Left hand side should be a SK in BCNF) (TRUE)
- (b) lossless, FD preserving into 3NF is always possible (TRUE)
- (c) lossless, " " " BCNF " " " (WE CANNOT GUARANTEE)
- (d) Any Relation with 2 attributes is in BCNF. (TRUE) LHS = Key (SK)

GATE-11-05

A table has fields F_1, F_2, F_3, F_4, F_5 with the following functional dependencies $F_1 \rightarrow F_3, F_2 \rightarrow F_4, F_1, F_2 \rightarrow F_5$ what is the NF of the Relation?

- a) 1NF b) 2NF c) 3NF d) None

Now, candidate key = $(F_1, F_2, F_3, F_4, F_5)$

$(F_1, F_2)^+ = (F_1, F_2, F_3, F_4, F_5) \therefore F_1, F_2$ is candidate key.

Now, ~~partial dependencies are present~~ $\therefore$ 2NF $\times$ $\left(\begin{matrix} F_1 \rightarrow F_3 \\ F_2 \rightarrow F_4 \end{matrix} \right)$

Now, this should be super key for 3NF $\Rightarrow F_1, F_2$ } one SK $\times$
 F_1, F_2 } = 3NF

GATE-12

which of the

- a) Every
- b) A Relation functional
- c) Every
- d) NO Relat

68. GATE

GATE-14

A prime

- a) In all
- b) In some
- c) In a
- d) Only in

GATE-05
Consider

$\therefore$ Now,

GATE-07

$R(a, b, c, d)$

FD: $a \rightarrow c$

Now

GATE-12

which of the following is True?

- Every relation in 3NF is also in BCNF (Not always)
- A Relation 'R' is in 3NF if every non-prime attribute of 'R' is fully functionally dependent on every key of R
- Every relation in BCNF is in 3NF
- No Relation can be in both BCNF and 3NF (FALSE)

(61)

68. GATE QUESTION ON NORMALIZATION 3

GATE-14

A prime attribute of a relation scheme R is an attribute that appears

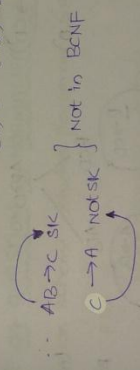
- In all candidate keys of R
- In some candidate key of R
- In a foreign key of R
- Only in the primary key of R

GATE-95

Consider $R(ABC)$ FD: $\{AB \rightarrow C, C \rightarrow A\}$ Show that R is in 3NF but not BCNF

$\therefore$ Now, candidate key = $(B)^+ = (ABC)^+$

$(AB)^+ = (ABC)^+$ } All are prime attributes
 $(C)^+ = (ABC)^+$ } $\therefore$ the relation is in 3NF



GATE-97

$R(a,b,c,d)$, a,b,c,d contains atomic values (No composite and multivalued)

FD: $\{a \rightarrow c, b \rightarrow d\}$

Now, candidate key = $(ab)^+ = (abcd)^+$

$\Rightarrow ab$ is the CK

$\Rightarrow a \rightarrow c$
 $b \rightarrow d$ } one partial dependency.

- 1NF but not 2NF
- 2NF but not 3NF
- IN 3NF
- None of the above

(62)

used into key in the

is False?

SK in BCNF (TRUE)

NOT QUANTIFIED

$S = \text{Key}(\text{SK})$

$S =$

functional of the

IF $\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$

one SK

= 3NF

9. GATE QUESTIONS ON NORMALISATION 4

GATE-99

$R(Stuv); Fd\{S \rightarrow T, T \rightarrow U, U \rightarrow V, V \rightarrow S\}$ and let $(R_1 \text{ and } R_2) = R$.
be a decomposition such that $R \cap R_2 \neq \emptyset$. The decomposition is:

- a) Not in 2NF
- b) In 2NF but not in 3NF
- c) In 3NF but not in 2NF
- d) Both 2NF and 3NF

$$CK = (S \rightarrow T) = (Stuv)$$

$$(T) \rightarrow (TuvS) \text{ all one}$$

$$U \rightarrow (UvST) \text{ prime}$$

$$V \rightarrow (VSTU) \rightarrow NF=3NF$$

10. GATE 2001 QUESTION ON BCNF

$R(ABCD)$ is a relation. which of the following does not have a lossless-join dependency preserving BCNF decomposition?

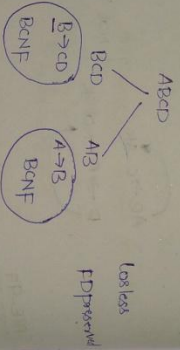
- a) $A \rightarrow B, B \rightarrow CD$
- b) $A \rightarrow B, B \rightarrow C, C \rightarrow D$
- c) $AB \rightarrow C, C \rightarrow AD$
- d) $A \rightarrow BCD$

option A: $A \rightarrow B, B \rightarrow CD \quad R(ABCD)$

$$CK = (A) \rightarrow (ABCD) \text{ No partial dependencies } = 2NF$$

Transitive dependency = $(B \rightarrow CD)$

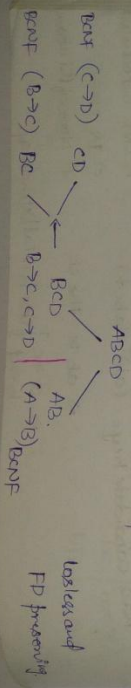
$$B \rightarrow (B, C, D)$$



option B: $A \rightarrow B, B \rightarrow C, C \rightarrow D$

$$CK = A$$

$(A) \rightarrow (ABCD) \Rightarrow$ No partial dependencies



option C: $AB \rightarrow$

$$C \rightarrow AD \text{ is } C \rightarrow \{C, A, D\}$$

11. GATE 2000

Relation 'R' is is decomposed into BCNF. To make query should be

- a) Dependency
- b) lossless join
- c) BCNF defn
- d) 3NF defn

12. GATE 04

The Relation has the FD

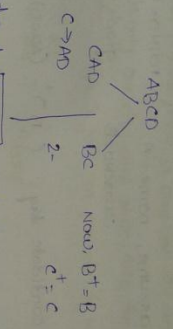
The highest n
a) 2NF b)

Q2

option c: $AB \rightarrow C$ $C \rightarrow AD$ $C_k = (AB)(CB)$ one candidate keys

$c \rightarrow AD$ is the partial dependency
 $c \in \{C, A, D\}$

Q3



The dependency $AB \rightarrow C$ is lost

all one prime $\Rightarrow NF = 3NF$

11. GATE 2002 QUESTION

Relation 'R' is decomposed using a set of FD's 'F' and Relation 'S' is decomposed using another set of FD's 'G'. One decomposition is in BCNF and other is in 3NF, but is not known which should be used on the decomposition?

- a) Dependency-preservation $\Rightarrow$ BCNF may not have dependency preserving
- b) Lossless join $\Rightarrow$ Both have lossless but we cannot differentiate them
- c) BCNF definition $\Rightarrow$ Enough to prove BCNF table and other automatically
- d) 3NF definition $\Rightarrow$ Not Enough, BCNF cannot be identified.

12. GATE 04 QUESTION ON NORMALISATION

The Relation scheme student performance (name, courseNo, rollNo, grade) has the FDs

- Name, courseNo $\rightarrow$ grade
- RollNo, courseNo $\rightarrow$ grade
- Name $\rightarrow$ rollNo
- RollNo $\rightarrow$ name

The highest normal form of this Relation scheme is

- a) 2NF
- b) 3NF
- c) BCNF
- d) 4NF

es
 teaches
 lossless and
 FD preserved

Candidate key = $(CNO)^+$ = (name, rollNo, courseNo, Grade) (2)

$$(CNO)^+ = (CNO)$$

Let us assume, name = A

courseNo = B

rollNo = C | FD's
grade = D
AB → D
CB → D
A → C
C → A

Now, candidate key = $(B)^+ = (ABC)^+$

$$B^+ = B$$

$$(AB)^+ = (ABCB)$$

$$(CB)^+ = (ACBC)$$

A, B, C

$$(DB)^+ = (CDB)$$

Now, partial dependencies are absent ∴ 2NF ✓

Now 3NF check

AB = SK

CB = SK

A → C ⇒ LHS is not SK but RHS is prime attribute } 3NF ✓

C → A ⇒ LHS is not SK but RHS is prime attribute

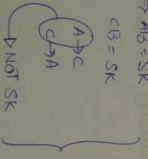
Now BCNF check

⇒ LHS should be Superkey

⇒ AB = SK

CB = SK

BCNF X



∴ AB = 3NF

13. 5A

A Rel

name,

Also, F

given

terms

a) 1NF

Let en

Given

Now,

⇒ Note

24. 5A

Which

a) Any

b) Any

c) A prim

d)

15. GATE 2008 QUESTION ON NORMALISATION

Consider the Relation Schemes of Library DB

Book (Title, Author, catalog-no, publisher, year, price)

Collection (Title, Author, catalog no) with the dependencies

I. Title Author $\rightarrow$ catalog-no

II. catalog-no $\rightarrow$ Title, Author, publisher, year

III. publisher, year title $\rightarrow$ price

Assume (Author, Title) is the Key for both schemes. Which of the following statements are True?

- a) Both Book and collection are in BCNF
- b) Both Book and collection are in 3NF only
- c) Book is in 2NF and collection is in 3NF
- d) Both Book and collection are in 2NF only

Sol: Book (ABCDEF)

collection (ABC)

Now, (ABC)

AB $\rightarrow$ C

C $\rightarrow$ ABDE

DAE $\rightarrow$ F

Transitive Dependency

DAE $\rightarrow$ F

3NF

Book = 2NF

Collection = 3NF

16. GATE 17-08

Let R(ABCDEF)

functional dependencies

$p \rightarrow c$ and $B \rightarrow G$

a) BCNF b) 3NF

Now, Candidate

Now, ...

Now, the func

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

16. GATE IT-08 AND 2013 QUESTION ON NORMALISATION (67)

Let $R(ABCDEP)$ be a Relational schema in which the following functional dependencies are known to hold $AB \rightarrow CD, DE \rightarrow P, C \rightarrow E, P \rightarrow C$ and $B \rightarrow G$. The Relation schema 'R' is in

- a) BCNF b) 3NF but not in BCNF c) 2NF but not in 3NF d) Not in 2NF

Now, Candidate Key = $(AB)^+$ = $(ABCD \checkmark E \checkmark P \checkmark G)$

Now, $(AB)^+ = (ABCD E P G) \therefore (AB)$ is the candidate key

Now, the functional Dependencies are

$AB \rightarrow CD$
 $DE \rightarrow P$
 $C \rightarrow E$
 $P \rightarrow C$
 $B \rightarrow G$

Transitive dependency

partial dependency

$\therefore$ Not in 2NF

5. RELATIONAL ALGEBRA

1. INTRODUCTION TO RELATIONAL ALGEBRA

⇒ Every data model is supposed to provide a way to manipulate the data.

⇒ Relational model → Relational Algebra (Simp for GRM)
 Relational calculus

⇒ Relational Algebra is a procedural language (what we want & how to get it)
 Relational calculus is a declarative language (what we want)

⇒ Relational Algebra → Operations (smallest unit of work that we can perform on any Relation)

⇒ RA = RC + Safe operations

Operations

π : Projection (selecting columns)

σ : Selection (selecting Rows)

$\times$: Cross product (Between two tables) (combine 2 tables)

(By using cross product we can work on more tuples simultaneously)

$\cup$: Union (same as union of sets)

$-$: Minus (set difference)

ρ : Rename (change name of the attributes)

Relational Algebra Expressions

→ Sequence of operations

⇒ $\cap$: Intersection $A \cap B = A - (A - B)$

⇒ $\bowtie$: Join ($\sigma \times \sigma$)

⇒ $\div$: Division ($\pi, \times, -$)



2. SELECTION

⇒ This operation

⇒ Selection / filter

Emp

ENO	ENAME	DNO

From this it is that selection operation is commutative

⇒ Select operation

⇒ Let us see

in the

The syntax

SELECTION OPERATION (σ):

→ This operation is used to choose a subset of Tuples (Rows).

→ Selection / Horizontal partition

Emp

Emp	Emp	Emp	Emp	Emp
ENO	Emp	DNO	Salary	

$\Rightarrow \sigma_{DNO=4}(Emp) \Rightarrow$ gives list of all Employees who belong to DNO=4

$\Rightarrow \sigma_{Sal > 10,000}(Emp) \Rightarrow$ gives list of all Employees whose salary is $> 10,000$.

From this it is clear that selection operation is commutative

$\Rightarrow \sigma_{Sal > 10,000 \text{ AND } DNO=4}(Emp) \Rightarrow$

→ Select operation works on only one tuple at a time.

→ Let us consider a Relation 'R' and |R| represents no. of tuples in the Relation 'R' then

$$|R| \geq \left| \left(\sigma_c |R| \right) \right|$$

max |R| min 0 tuples

The syntax of selection operation is

$\sigma_{\langle \text{selection condition} \rangle}(\text{Relation name})$
 ↓
 Boolean Expression
 (Attribute name) < Comparison operator > (constant) (Attribute name)

→ can be Relational Algebra Expression also

3. PROJECTION OPERATION (π):

→ used to choose a subset of columns, and the output will always be a Relation.

→ Projection/Vertical partitioning

→ Degree of a table = No. of attributes in that table

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

Now, $\pi_A(R)$:

A
1
2
3

Degree = 1

$\pi_{AB}(R)$:

A	B
1	a
2	c
3	e

Degree = 2

→ General syntax of projection operation is

$\pi_{\langle \text{attribute list} \rangle}(R)$ → create Relational Algebra Expression
 ↓
 subset of attributes present in R.

→ π operation eliminates Duplicates (Duplicate elimination)

A	B	C
1	a	c
1	b	c
2	a	c
2	b	c

(AB) - Candidate Key

$\pi_C(R)$:

C
c

$\pi_A(R)$:

A
1
2

$\pi_B(R)$:

B
a
b

$\pi_{AB}(R)$:

A	B
1	a
1	b
2	a
2	b

$\pi_{ABC}(R)$: Entire table

→ Consider a Relation R' and the tuples be represented by |R| then,

$$|R| \geq |\pi_{\langle \text{attribute list} \rangle}(R)|$$

$\pi_{\langle \text{attribute list} \rangle}(R) < R$, when $\langle \text{attribute list} \rangle$ is not superkey.

$\pi_{\langle \text{attribute list} \rangle}(R) = R$, when $\langle \text{attribute list} \rangle$ is a superkey.

(70)

→ The projection

Let R (ABC)

A	B	C
---	---	---

Let R be a Rel

→ The projection

4. RENAME

Now, consider

A	B	C
---	---	---

Representation

→ sometimes + attributes of

→ sometimes if then the s

→ Renaming

70

always let $R(ABC)$

A	B	C
1	a	1
2	c	2
3	e	3

A	B
1	a
2	c
3	e

the projection operation is not "commutative"

71

A	B
1	a
2	c
3	e

Now, $\pi_A(\pi_{AB}(R)) =$

Now, $\pi_{AB}(\pi_A(R)) =$

AB not possible

Let R be a Relation, then

$\pi_{A_2}(\pi_{A_1}(R))$ is valid only when $A_2 \subseteq A_1$

when this condition satisfy then we can directly write as $\pi_{A_2}(R)$.

The "projection" operation is same as "select" operation in SQL with distinct

4. RENAME OPERATION (R)

Now, consider a Relation $R(ABC)$

A	B	C
1	a	1
2	c	2
3	e	3

Now, if i need to get the AB columns satisfying a condition 'x' then the query is $\pi_{AB}(\sigma_x(R))$, what some people do is instead of writing in single line (single Representation they represent in different table.

Temp $\leftarrow \sigma_x(R)$

Answer $\leftarrow \pi_{AB}(\text{Temp})$

Sometimes there might be a need that you want to Rename the attributes of a table, then the syntax is $\rho_{S(X,Y,Z)}(R)$ and table name from R to S' $[R(ABC) \rightarrow S(X,Y,Z)]$

Sometimes you might want to Rename the table not the attributes then the syntax is $\rho_S(R)$
Renaming operator in SQL is "AS".

A	B
1	a
2	c
3	e

able

by R1

super key

5. GATE 98 QUESTION ON SELECTION AND PROJECTION

Which of the query transformations (ie replacing the RHS expression by the RHS expression) is incorrect? R_1 and R_2 are relations C_1, C_2 are selection conditions and A_1, A_2 are attributes of R_1

- $\sigma_{C_1}(\sigma_{C_2}(R_1)) \rightarrow \sigma_{C_2}(\sigma_{C_1}(R_1)) = \text{False}$
- $\sigma_{C_1}(\pi_{A_1}(R_1)) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1)) = \text{True}$
- $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \cup \sigma_{C_1}(R_2) \rightarrow \text{TRUE}$
- $\pi_{A_2}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_2}(R_1)) \rightarrow \text{FALSE}$

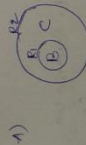
if C_1 condition is in such a way that it contains A_1, A_2 columns, we are extracting A_2 A_1 will be missed out

6. GATE 2012 QUESTION ON PROJECTION

Suppose $R_1(AB)$ and $R_2(BC)$ are two relation schemas. Let r_1 and r_2 be the corresponding values. B is the Foreign key that to C in R_1 . If data in R_1 and R_2 satisfy Referential integrity constraints, which of the following is always True?

- $\pi_B(r_1) - \pi_C(r_2) = \emptyset$
- $\pi_C(r_2) - \pi_B(r_1) = \emptyset$
- $\pi_B(r_1) = \pi_C(r_2)$
- $\pi_B(r_1) - \pi_C(r_2) \neq \emptyset$

Sol: B is the foreign key of C in $R_2 \Rightarrow$ Every value of B will be in C .



$$\therefore \pi_B(r_1) - \pi_C(r_2) = \emptyset$$

cases but not always



$$\pi_C(r_2) - \pi_B(r_1) \neq \emptyset$$

$$\pi_B(r_1) - \pi_C(r_2) \neq \emptyset$$

contradiction of option 1

$\rightarrow$ B is FK that refers to C means, B is depending on C and every value of B will be present in C .

7. SET

$\Rightarrow$ The next $\Rightarrow$ when $\Rightarrow$ should

$R(A_1, A_2)$
 $S(B_1, B_2)$

$NAB, (RUS)$

$\therefore$ The Union

$\therefore$ The Slice

$\therefore$ The Difference

$\Rightarrow$ Intersection

8. CASES

(i) Binary of

(ii) Relations

Emp

A	B	C

$\Rightarrow$ If the tab

no of attri

$\Rightarrow$ If R_1 has

1. SET OPERATIONS

- ① The next set of operations are Union ($\cup$), Intersection ($\cap$), Minus ($-$)
- ② when we apply the above operations on Relations then they should be "union compatible" or "Type compatibility".

$R(A_1, A_2, \dots, A_p)$
 $S(B_1, B_2, \dots, B_n)$

we can perform RUS iff

- 1) have same degree
- 2) corresponding elements domain must be same.

Now, $(R \cup S) = R(A_1, A_2, \dots, A_n)$

we use the this set name of $R \cup S = R(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_n)$

- ∴ The Union operator is commutative - $R \cup S = S \cup R$ [No duplicates]
- ∴ The Intersection operator is commutative - $R \cap S = S \cap R$ [No duplicate values]
- ∴ The Difference operator is not commutative - $(R - S) \neq (S - R)$ [No duplicates]

$$R \cap S = (R \cup S) - (R - S) = (S - R)$$

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

2. CARTESIAN PRODUCT

- 1) Binary operation
- 2) Relations need not be union compatible

Emp

A	B	C
---	---	---

Dependent

D	E
---	---

⇒ Now
Emp x dependent

A	B	C	D	E
---	---	---	---	---

⇒ If the table R_1 contains 'n' attributes and table R_2 contains 'm' attributes then $R_1 \times R_2$ contain $(n+m)$ attributes (columns)

⇒ If R_1 has 'n' tuples and R_2 has 'm' tuples then $R_1 \times R_2$ has $(n \times m)$ tuples

Employee

A	B	C
1	b1	c1
2	b2	c2
3	b3	c3

(m) rows = 3

(n) columns = 3

Dependent

D	E
1	a1
2	a2

(p) rows = 2

(q) columns = 2

Employee x Dependent

A	B	C	D	E
1	b1	c1	1	a1
1	b1	c1	2	a2
2	b2	c2	1	a1
2	b2	c2	2	a2
3	b3	c3	1	a1
3	b3	c3	2	a2

⇒ using a cross product alone is meaningless, using it with some condition

$$\Rightarrow \sigma_{A=D} (\text{Employee} \times \text{Dependent}) \Rightarrow$$

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

∴ A new operation called join $\bowtie$ is developed to specify the cross product with a condition

∴ Join is a combination of ($\times$ and σ).

9. JOIN AND NATURAL JOIN ($\bowtie$)

⇒ "Join" operation is used to combine two operations / Relations into Related tuples form

a single larger tuple

Ex:

A	B	C
1	b1	c1
2	b2	c2

(R $\bowtie$ S)

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

⇒ some of the questions that will be asked in Gate exam if you join two tables what are the no. of tuples in the resulting new Relation.

Ex:

R (ABC)

A	B	C
1	a	1
2	b	1
3	c	1

The syntax

Now, some operation are Automatically performed on

R

A	B	C
1	a	1
2	b	1
3	c	1

10. DIVISION

R

A	B
1	a
2	b

74

E	a1	a2	a1	a2	a1	a2

(m1 * n1) rows

(m2 * n2) columns

are Condition

75

Ex:

R(ABC)			S(DE)			(R ⋈ S) (theta)		
A	B	C	D	E	A	B	C	D
1	a	d	1	a	1	a	d	1
2	b	e	2	b	2	b	e	2
3	c	f	2	c	2	c	e	2

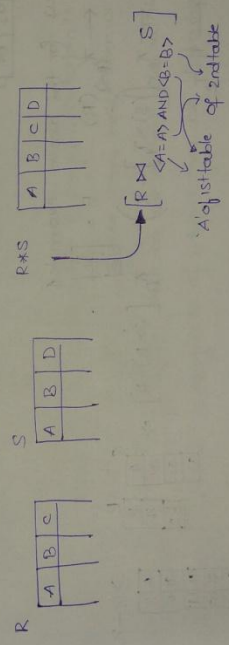
The syntax of Join is $[R \bowtie_{\text{join condition}} S]$

$\downarrow$ θ AND θ_j $\Rightarrow$ This join is called "Theta" join (No comparison against with values, comparison only between attributes)

ity the

Now, some conditions are frequently used while performing join operation and so we don't specify the condition and they are automatically derived from the Relations such kind of join is called Natural join denoted by $(*)$. The join operation is automatically performed on matching names.

relations into



10. DIVISION OPERATION ($\div$)

$R \div S \Rightarrow$ what ever attributes have in 's' you subtract them from 'r' and write what is remaining in R. $\Rightarrow$ The attributes of $(R \div S)$ will be in $R \div S$.

R

A	B
1	a
2	b

S

A
1
2

$R \div S (R-S)$

B
a
b

if you are getting new

Ex 1:

R

A	B
a1	b1
a2	b1
a1	b2

R ÷ S

A
a1
a2

B
b1

→ This table contains the values of B that are associated with the corresponding values of S.

→ Here 'b1' is associated with all the attributes (a1, a2) of S so it will be present in (R ÷ S)

Ex:

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b4
a2	b4
a3	b4

R ÷ S

A
a1
a2
a3

B
b1
b4

These are the values associated with attributes of S.

Now, for the above example 1

$$T_1 \leftarrow \pi_{(R)}(R) = B = \begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

$$T_2 \leftarrow \pi_{(R \div S)}(R \div S) = \pi_B[(S \times B) - R]$$

$$T \leftarrow T_1 - T_2$$

B
b1
b2

B
b1

S × B - R

A	B
a1	b1
a2	b1
a1	b2
a3	b2

S

A
a1
a2
a3

⇒

B
b1
b2

⇒ By performing a join on left side B

(76)

∴ The

A
a1
a2

Attribute

⇒ If R is a Relation

U. Aggarwal

The Agg

COUNT

⇒ Another

mainly

perform

are miss

join are

(76)

Ans the values associated with values of S:

of S solution

are the values d with $a_1, a_2, a_3 =$ of S:

SXB-R

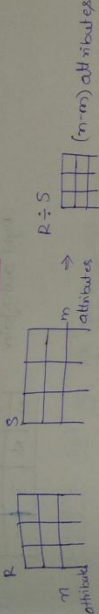
a	b
a1	b1
a2	b2
a3	b3
a4	b4

a	b
a1	b1
a2	b2

∴ The implementation of division using the basic operations is

(77)

$$\begin{aligned} T_1 &\leftarrow \pi_{(R,S)}(R) \\ T_2 &\leftarrow \pi_{(R,S)}[(\sigma_{X \neq Y} T_1) \rightarrow R] \\ T &\leftarrow T_1 \div T_2 \end{aligned}$$



Attributes in S $\subseteq$ Attributes in R

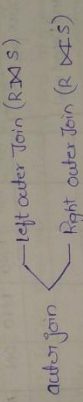
⇒ If R contains X attributes, S contains Y attributes then the Relation $Z = (R \div S)$ contains X-Y attributes

$$Z = (X) - (Y) \Rightarrow X = Z + Y$$

II. AGGREGATE FUNCTIONS AND OUTER JOINS

The Aggregate functions are SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT.

⇒ Another additional function provided is outer join. Outer join is mainly introduced because we may miss some tuples when we perform Natural join (tuples which are not satisfying a condition are missed). The outer join is of 2 types they are Left outer join and Right outer join.



⇒ By performing left outer join the output contains all the matching tuples in 'R' and 'S' and also the non matching tuples in the left side Relation.

Ex:

R	A	C	D
	1	1	-
	2	-	-

S	B	D
	1	-
	3	-

New, (R ⋈ A S)
A=B

A	C	B	D
1	-	1	-
2	-	N	N

New, (R ⋈ X S)
A=B
Right outer join

A	C	B	D
1	-	1	-
N	N	3	-

Ex:

Emp	ID	EN	ENO

Department	DNO	DN	MID

Manager ID.

Emp ⋈ Department =
<ID=mid>

ID	EN	ENO	DNO	DN	MID
			N	N	N

which does not match with MID but present in Emp table

Emp ⋈ Department =
<mid=ID>

ID	EN	ENO	DNO	DN	MID
N	N	N			
N	N	N			

Tuples matching with EID
Not matching and present in Department table

Emp ⋈ Department = Full Outer Join.

Not Equal to Cross product of Employee and Department

ID	EN	ENO	DNO	DN	MID
1	Ravi	55	N	N	N
N	N	N	9	Ravan	509

Tuples matching
Not matching

12. COMPLETE
The operations

New

13. GATE 94

GATE 94

Given a Relational Algebra expression
Find the result of the expression
[R ⋈ S] - (R ⋈ S)

GATE 99

Consider the following two relations R and S. R has 100 tuples and S has 100 tuples. R and S have 100 tuples in common. Find the number of tuples in the result of the following expression: (R ⋈ S) - (R ⋈ S)

Imp:

	Min	Max	
$\times$	(mn)	(mn)	$R = n$ tuples
$\bowtie$	0	(mn)	$S = m$ tuples
$\Join$	n	(mn)	
$\bowtie$	m	(mn)	
$\bowtie$	$\max(m, n)$	(mn)	

12. COMPLETE SET OF RA OPERATIONS

The operations $\{\sigma, \pi, \cup, \cap, \times\}$ = Complete set

$\{\bowtie, \cap, \div, \Join, \bowtie, \bowtie\}$ Can be derived from complete set.

Now,

	Min	Max	
$\cup$	$\max(m, n)$	(mn)	$R = m$ tuples
$\cap$	0	$\min(m, n)$	$S = n$ tuples
$-$	0	m	

13. GATE QUESTIONS ON RA

GATE-94

Given a Relational Algebra Expression Using only the minimum no of operators from $\{\cup, -\}$ what is equivalent to $(R \cap S)$?

Q. $(R \cap S) = R - (R - S)$

$[(R \cup S) - (R - S)] - (S - R) = (R \cap S)$

GATE-99

Consider the join of a relation R with relation S. If R has 'm' tuples and 'S' has 'n' tuples, then the maximum and minimum sizes of the join respectively are

mn and 0

mn and $(m+n)$ (d) mn and mn

es where matches in EID but present

Tuples matching with EID not matching and present in parent table

Tuples matching

Not matching

GATE - 99

The Relational Algebra expression equivalent to the following tuple calculus expression is:

Expression $\{t / t \in r \wedge (t[A] = 10 \wedge t[B] = 20)\}$ is:

a) $\sigma_{\{A=10\} \cap \{B=20\}}(r)$ OR but we need AND $\sigma_{\{A=10\}}(r) \cap \sigma_{\{B=20\}}(r)$

b) $\sigma_{\{A=10\}}(r) \cup \sigma_{\{B=20\}}(r)$ OR $\sigma_{\{A=10\}}(r) - \sigma_{\{B=20\}}(r)$

GATE - 2000

Given the Relations

Employee (name, salary, deptno) and

Department (deptno, deptname, address)

which of the following queries cannot be expressed using the basic

Relational Algebra operations ($\sigma, \pi, \times, \bowtie, \cup, \cap, -$)?

- a) Department address of every employee $\Rightarrow$ Using join Emp $\bowtie$ Dept. $\rightarrow$ This will give deptno = deptno, get address
- b) Employees whose name is same as Dept. name $\Rightarrow$ $\sigma_{e.name = d.name}$ get address
- c) The sum of all employee salaries. = AGGREGATE FUNCTION
- d) All employees of a given department = $\sigma_{deptname = 'dname'}(r)$ get dept name $\rightarrow$ select employee corresponding to that particular dept number

14. GATE 2003 QUESTION ON CONVERTING SQL TO RA

Consider the following SQL query

select distinct $a_1, a_2, \dots, a_n$

from $r_1, r_2, \dots, r_m$

where P.

For an arbitrary predicate 'P', this query is equivalent to which of the following Relational Algebra expressions

a) $\pi_{a_1, a_2, \dots, a_n} \sigma_P$

b) $\pi_{a_1, a_2, \dots, a_n}$

select distinct

where = σ_P

c) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

d) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

e) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

f) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

g) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

h) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

i) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

j) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

k) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

l) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

m) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

n) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

o) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

p) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

q) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

r) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

s) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

t) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

u) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

v) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

w) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

x) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

y) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

z) $\pi_{a_1, a_2, \dots, a_n} \times \pi_{a_1, a_2, \dots, a_n}$

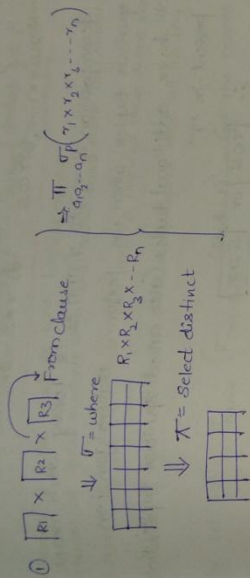
80) the calculus

a) $\pi_{a_1, a_2, \dots, a_n} \sigma_p(x_1, x_2, \dots, x_m)$ ⑧

b) $\pi_{a_1, a_2, \dots, a_n} \sigma_p(x_1, x_2, \dots, x_n)$ ⑨

select distinct = projection
where = Selection

From = Cross product



the basic

get dept name
get dept address
name

get dept name
Corresponding
dept number

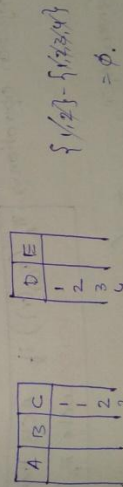
lent to

IS-GATE Q4 QUESTION ON FOREIGN KEY AND RELATIONAL ALGEBRA

Let $R_1(ABC)$ and $R_2(DE)$ be two Relation schema, where the primary keys are shown underlined, and let C be a foreign key in R_1 referring to R_2 . Suppose there is no violation of the above referential integrity constraint. In the corresponding relational algebra expressions would necessarily produce following Relational Algebra expressions would necessarily produce and empty relation?

a) $\pi_D(x_2) - \pi_C(x_1)$ ① $\pi_C(x_1) - \pi_D(x_2)$ ② $\pi_D(x_1) \bowtie \pi_C(x_2)$ ③ $\pi_C(x_1) \bowtie \pi_D(x_2)$ ④ $\pi_C(x_1) \bowtie \pi_D(x_2)$

ANSWER: The values present in the 'C' will definitely be present in 'D'.



$\therefore \pi_C(x_1) - \pi_D(x_2) = \emptyset$

16. GATE 05 QUESTION ON DECOMPOSITION AND JOIN

Let 'r' be a relation instance with schema R(ABC). we define $r_1 = \pi_{AB}(r)$ and $r_2 = \pi_{AC}(r)$. Let $S = r_1 \bowtie r_2$ where $\bowtie$ denotes natural join. Given that the decomposition of r into r_1 and r_2 is lossy, which one of the following is True?

- a) $S \subset r$ b) $r \cup S = r$ c) $r \subset S$ d) $r \bowtie S = S$.

Given the decomposition of r into r_1 and r_2 is lossy which means we get spurious tuples when we perform Natural join them. $\therefore S$ will have additional tuples (also called spurious tuples) that are not present in 'r'.

$\therefore \boxed{r \subset S}$ or $\boxed{S \supset r}$

17. GATE 2014 QUESTION ON CASCADE SELECTION AND JOIN

What is the optimised version of the Relation Algebra expression $\pi_{A_1}(\pi_{A_2}(\sigma_{F_1}(\sigma_{F_2}(r))))$ where A_1 and A_2 are sets of attributes in 'r' with $A \subset A_2$ and F_1, F_2 are Boolean expressions based on the attribute in R?

- a) $\pi_{A_1}(\sigma_{F_1 \wedge F_2}(r))$ b) $\pi_{A_1}(\sigma_{F_1 \vee F_2}(r))$ c) $\pi_{A_2}(\sigma_{F_1 \wedge F_2}(r))$ d) $\pi_{A_2}(\sigma_{F_1 \vee F_2}(r))$

$\Rightarrow$ selection operation is commutative and we can cascade it

$\therefore \sigma_{F_1}(\sigma_{F_2}(r)) \equiv \sigma_{F_1 \wedge F_2}(r)$

$\Rightarrow$ Now projection operation $\Rightarrow \pi_A(\pi_B(r))$ if $A \subset B$ then we write $\pi_A(r)$

Here Given, $A_1 \subset A_2$ $\pi_{A_1}(\pi_{A_2}(\sigma_{F_1 \wedge F_2}(r)))$
 $= \pi_{A_1}(\sigma_{F_1 \wedge F_2}(r))$
 $= \text{option a.}$

18. GATE 04 QUESTION

Consider the relation is shown underli atleast one boy are expression produ

Name	Sex
------	-----

- a) Names of girls
- b) Names of girls
- c) Names of girl
- d) Names of girl

20. Let $A = \pi_n$

$B = \pi_n$

Name	Sex
------	-----

Now $A \bowtie B =$

Q2)

we define
des natural
lossy which

which means
join them
as tuples) that

AND JOIN
expression
attributes
based on the

$$\pi_{A_2}(\sigma_{A_1 \neq A_2}(r))$$

ade it

then we
 $\pi_A(r)$

Q4 QUESTION ON INTERPRETING AN RA EXPRESSION

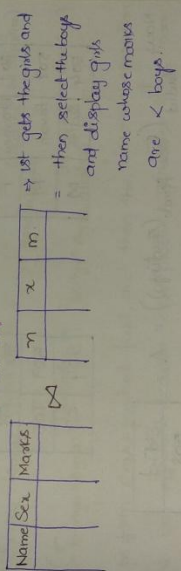
Consider the relation Student (name, sex, marks) where the primary key is shown underlined, pertaining to students in a class that has atleast one boy and one girl. what does the following relational algebra expression produce? (Note: ρ is the Rename operator).

$$\pi_{name}(\sigma_{sex=female}(Student)) - \pi_{name}(\sigma_{sex=female, marks < 50}(Student))$$

- a) Names of girls students with highest marks
- b) Names of girls students with more marks than some boy student.
- c) Names of girl students with marks not less than some boy student
- d) Names of girl students with more marks than all the boy students

20) Let $A = \pi_{name}(\sigma_{sex=female}(Student))$ = Get the names of all girls (sex = female)

$$B = \pi_{name}(\sigma_{sex=female, marks < 50}(Student))$$



Now $A - B = \{ \text{Names of all girls} \} - \{ \text{Names of girls whose marks are} < 50 \}$

$\Rightarrow \{ \text{Names of all girls whose marks are more than all the boy students} \}$

- OPTION D

19. STATE QT QUESTION ON INTERPRETING THE RA EXPRESSION

Information about a collection of students is given by the relation Stud Info (studId, name, sex). The Relation enroll (studId, courseId) gives which student has enrolled for what course(s). Assume that every course is taken by one male and atleast one female student. what does the following Algebraic Expression Represent.

$$\pi_{\text{courseId}} \left(\left(\pi_{\text{studId}} (\sigma_{\text{sex} = \text{female}} (\text{studInfo})) \times \pi_{\text{courseId}} (\text{enroll}) \right) - \text{enroll} \right)$$

- courses in which all the female students are enrolled
- courses in which a proper subset of female students are enrolled
- courses in which only male students are enrolled
- None of the above

Stud Info

studId	Name	Sex
508	Armanjan	F
507	Chavara	F
509	Pavan	M
504	Sujith	M

Enroll

studId	courseId
508	DBMS
507	CN
509	TOC
504	CD

$$\text{Now, } \pi_{\text{studId}} (\sigma_{\text{sex} = \text{female}} (\text{studInfo})) = A =$$

studId
508
507

$$\text{Now } B = \pi_{\text{courseId}} (\text{enroll}) =$$

courseId
DBMS
CN
TOC
CD

$$\text{Now, } A \times B =$$

studId	courseId
508	DBMS
508	CN
508	TOC
508	CD
507	DBMS
507	CN
507	TOC
507	CD

$$\text{Now, } \pi_{\text{courseId}} =$$

courseId	Subset of female
DBMS	of female
CN	male
TOC	male
CD	Registered

∴ OPTION B

20. STATE QT

consider the f

b- schema =

a- schema =

d- schema =

Let Branch, acc

schema. Assume

bigger than the

consider the +

which one of the

above query?

or $\pi_{\text{cname}} (\sigma_{\text{b, acc}}$

by $\pi_{\text{cname}} (\sigma_{\text{b, acc}}$

or $\pi_{\text{cname}} (\sigma_{\text{b, acc}}$

or $\pi_{\text{cname}} (\sigma_{\text{b, acc}}$

⇒ If you perform

it won't be

no of tuples

⇒ 20 first appl

product for

∴ 1st selec

2nd sele

Now, acc

Now, acc

$\pi_{\text{cname}} (\sigma_{\text{b, acc}}$

QUESTION 4

relation

Id gives
e that
student.

20. GATE Q1 IT ON OPTIMISATION OF RA EXPRESSION

consider the following relation schemas

b- schema = (b-name, b-city, assets)

a- schema = (a-num, b-name, bal)

d- schema = (c-name, a-number)

Let Branch, account and depositor be respective instances of above

schema. Assume that account and depositor relations are much

bigger than the branch relation.

consider the following query. Π ($\sigma_{c-name = 'Agra' \wedge bal < 0}$ (branch) $\bowtie$ (account) $\bowtie$ depositor))

which one of the following queries is the most efficient version of

above query?

a) $\Pi_{ename} (\sigma_{bal < 0} (\sigma_{b-city = 'Agra'} (branch) \bowtie (account) \bowtie depositor))$

b) $\Pi_{ename} (\sigma_{b-city = 'Agra'} (branch) \bowtie (\sigma_{bal < 0} (account \bowtie depositor)))$

c) $\Pi_{ename} ((\sigma_{b-city = 'Agra'} (branch) \bowtie (\sigma_{bal < 0} (account \bowtie depositor)))$

d) $\Pi_{ename} (\sigma_{b-city = 'Agra'} (branch) \bowtie (\sigma_{bal < 0} (account \bowtie depositor)))$

$\Rightarrow$ if you perform cross product first, and then the selection operation

it won't be efficient because cross product generates large/noise

no of tuples.

$\Rightarrow$ so first apply the selection condition and then apply the cross

product from the obtained two smaller tables.

$\therefore$ 1st select Agra city from b-schema

and select the tuples with bal < 0 from a-schema

Now, cross product the account with depositor

Now, cross product the Result with b-schema

$\Pi_{ename} (\sigma_{b-city = 'Agra'} (\sigma_{bal < 0} (account) \bowtie depositor) \bowtie (branch))$

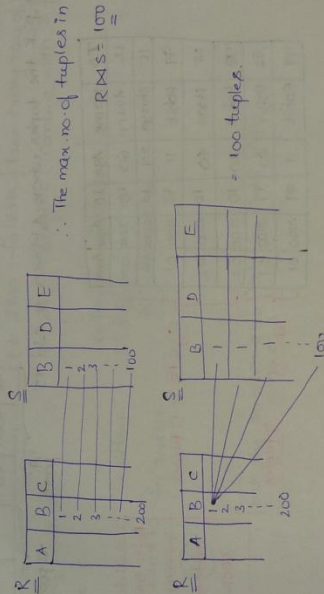
= option B.

subset
of female
male
registered

22. GATE 2010 QUESTION ON NATURAL JOIN ON PRIMARY KEY

The following FO's hold for Relations $R(ABC)$ $S(DEF)$ $\{A \rightarrow X, A \rightarrow Y\}$
 The relation 'R' contains 200 tuples, and 'S' contains 100 tuples. What
 is the max no. of tuples possible in natural join of R, S i.e. $R \bowtie S$?
 a) 100 b) 200 c) 300 d) 2000

Now, $B \rightarrow A, A \rightarrow C \Rightarrow B \rightarrow \{B, A, C\} \Rightarrow B$ is CK in $R(ABC) \Rightarrow B$
 contains distinct values in $R(ABC)$



23. GATE 12 QUESTION ON UNION AND NATURAL JOIN

consider the following relations A, B, C

A			B			C		
ID	Name	Age	ID	Name	Age	ID	Phone	Area
12	Arun	60	15	Shreya	24	16	2200	02
15	Shreya	24	25	Hani	40	99	2100	01
99	Rohit	11	98	Rohit	20			

How many tuples does the result of the following relational Algebra
 Expression contain? Assume that the schema of $A \cup B$ is the same
 as that of A. $(A \cup B) \bowtie C$

$A \cup B = \{A, B\}$
 $A \cup B = \{A, B\}$

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

$A \cup B =$

ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11
25	Haari	40
98	Rohit	20

$\bowtie C$

ID	phone	Area
10	2200	02
99	2100	01

$\Rightarrow$

(AUB) \bowtie C			ID	phone	Area
12	Arun	60	10	2200	02
12	Arun	60	99	2100	01
15	Shreya	24	10	2200	02
15	Shreya	24	99	2100	01
99	Rohit	11	10	2200	02
99	Rohit	11	99	2100	01
25	Haari	40	10	2200	02
25	Haari	40	99	2100	01
98	Rohit	20	10	2200	02
98	Rohit	20	99	2100	01

Now, pick the tuples where $A_{id} > 40 \vee C_{id} < 15$

ID	Name	Age	ID	phone	Area
12	Arun	60	10	2200	02
15	Shreya	24	10	2200	02
99	Rohit	11	10	2200	02
25	Haari	40	10	2200	02
98	Rohit	20	10	2200	02
98	Rohit	20	99	2100	01
99	Rohit	11	99	2100	01

$C_{id} < 15 = 5 \text{ tuples}$

$A_{id} > 40 = 2 \text{ tuples}$

$= 7 \text{ tuples}$

Refer 547 page
3.85(a) in
made easy
book

The nested loop
we are going to

$r(R)$
 $\downarrow$
20 tuples
 $\downarrow$
2 Blocks

In this case the
No. of Block access
are

$$1 + 10 \times 10 \text{ - for } r(R)$$

$$1 + 10 \times 10 \text{ - for } s(S)$$

$$\underline{\hspace{1cm}}$$

202

24. GATE 14 QUESTION ON NESTED EVALUATION OF JOIN

Consider a join between relation $r(R)$ and $s(S)$ using the nested loop method. There are 3 buffers each of size equal to disk block size, out of which one buffer is reserved for intermediate results. Assuming size $(r(R) < \text{size } s(S))$, the join will have fewer no. of disk blocks access if

- relation $r(R)$ is in the outer loop
- relation $s(S)$ is in the outer loop
- join selection factor between $r(R)$ and $s(S)$ is more than 0.5
- join selection factor between $r(R)$ and $s(S)$ is less than 0.5

25. GATE 201

in the dep
dependent C
relational A

ID	phone	Area
10	2200	02
99	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01

C. point

Refer 547 page
3.85(a) in
made easy
book

The nested loop method means for every tuple in one relation ^{other} we are going to take entire table and we are joining them. (87)

$r(R)$
↓
20 tuples
↓
2 Blocks

In this case the
No. of Block access
are

$1 + 10 \times 10$ - for 1st block
 $1 + 10 \times 10$ - for 2nd blk

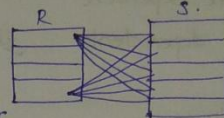
(202)

$s(S)$
↓
100 tuples
↓
10 Blocks

$1 + 2 \times 10$ - for 1st blk
 $1 + 2 \times 10$ - for 2nd blk
! Take every row of R and
combine with 'S'

10 times
 $1 + 2 \times 10$ - for 10th blk
210 Block Access

Assuming,
size of Each block = 10 tuples
and
nested looping means



∴ put the smaller size table in outer loop.

→ How did we write this is for the 1st block, we have to access

it so (1+) and for every block we have to link with all the

10 blocks in 'S' (so $1 + 10 \times 10$)

1st block

Accessing all the blocks in

'S' = 10 blocks and size of each block = 10 tuples

= $10 \times 10 = 100$ tuples.

25. GATE 2014 QUESTION ON INTERPRETATION OF RA EXPRESSION

consider the Relational schema given below, where "eId" of the relation dependent is a foreign key referring to "empId" of the relation employee

Assume that every employee has atleast one associated dependent in the dependent relation. Employee (empId, empName, empAge) dependent (depId, eId, depName, depAge) consider the following Relational Algebra Query.

$$\pi_{\text{empid}}(\text{employee}) - \pi_{\text{empid}}(\text{employee} \bowtie (\text{dependent}))$$

$$\downarrow$$

$$\langle \text{empid} = \text{eid} \rangle \wedge (\text{empAge} \leq \text{depAge})$$

The above query evaluates to the set of empid's of employees whose age is greater than that of

- a) some dependent c) some his/her dependents
b) All dependents d) All of his/her dependents.

Employee

ID	Age
1	40
2	30
3	20

Dependent

did	Age
1	30
1	20
2	40
3	40

Now, $\pi_{\text{empid}}(\text{employee} \bowtie (\text{dependent}))$

$$\langle \text{empid} = \text{eid} \rangle \wedge (\text{empAge} \leq \text{depAge})$$

⇒ It returns the empids of employees whose Age is less than or equal to their dependents.

Now, from employee table, subtract the above relation then we will get

⇒ empid's of employees whose age is greater than his/her dependents.

∴ Option D.

26. INTRODUCTION TO RELATIONAL CALCULUS

On the Relational model two formal languages are proposed they are

1) Relational Algebra

2) Relational calculus

- Tuple Relation calculus (TRC)
- Domain Relational calculus (DRC)

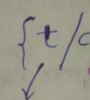
90

A language
express



27. TRC

The most



Tuple
variable

Ex:-

student

FN

{t.FN/stu

The gener

{t, A,

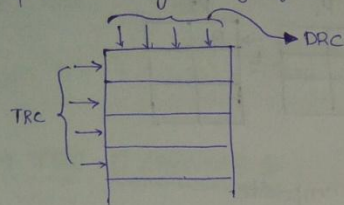
28. FREE

General
in the

<1> R(t)

90

A language is said to be Relationally complete if it has the capacity to express every query of Relational Algebra.



27. TRC SYNTAX

The most general form of TRC is

$$\{t / \text{cond}(t)\}$$

Tuple variable

condition

Ex:-

Student

FN	LN	Marks

Now if i want to find the student names whose marks are greater than 50.

$$\{t / \text{student}(t) \text{ AND } t.\text{marks} > 50\}$$

Tuple variable to range on table

Name of the table

that the tuple variable should Range

$$\{t.\text{FN} / \text{student}(t) \text{ AND } t.\text{marks} > 50\}$$

↳ Gives FN of students whose marks > 50.

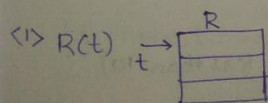
The general form is

$$\{t_1.A_1, t_2.A_2, \dots, t_n.A_n / \text{cond}(t_1, \dots, t_{n+m})\}$$

28. FREE AND BOUNDED VARIABLES

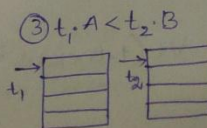
Generally we use atomic expression while representing conditions

in the TRC. The basic Atomic expressions will be



(2) $t.A \theta \langle \text{constant} \rangle$

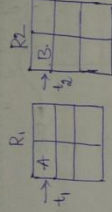
Diagram showing a table with a tuple variable 't' pointing to a row, and a constant value '50'.



Now,

Composite Expressions

$$1) R_1(t_1) \wedge R_2(t_2) \wedge t_1 < t_2.B$$



2) By default every Atomic expression is composite

3) $F_1 \wedge F_2, F_1 \vee F_2, \neg F_2$ are Composite Expressions $\{F_1, F_2\}$ = Atomic Expressions

4) $\forall t(F), \exists t(F)$ are composite Expressions.

we have 2 types of variables they are:

1) Free variables: The variables to which the Quantifiers are not applied.

2) Bounded variables: The variables to which the Quantifiers are applied

$\Rightarrow$ Free variables appear only on the left side of $\rightarrow$ in TRC Representation
In the Right side Bounded variables occur.

$$TRC = \left\{ \frac{\text{Free variables}}{\text{Bounded variables}} \right\}$$

Now,

R R(t)

A	B	C
10	1	20
20	2	30
30	3	40
40	4	50
50	5	60

$\exists t (t.A > 50) \Rightarrow$ If specifying this condition then it returns false because the table contains the tuples satisfying the condition

Now, $\forall t (t.A > 50) \Rightarrow$ Returns false because some of the values of 'A' are less than 50.

Now, $\forall t (t.A < 10) \Rightarrow$ True (All the values are less than 10)

92

Now, $\sim \exists t (t.A < 10)$

$t.C < 10$
Now, $\forall t (t.C$

29. TRC EXA

List the names
"Research" dept

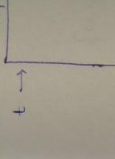
Employee (Frame)
Department (Dept)
Dept - location

Project (proj)

works-on (E)

Dependent (Dep)

sol Emp



$\Rightarrow$ Now, after variables for should be should be Variable for

$\Rightarrow$ The cross for TRC

$\{t, Frame, t$

Now, $\sim \exists t (t < 10) \Rightarrow$ There doesn't exist atleast one value of t such that $t < 10$.
 $t < 10 \therefore \text{TRUE}$

Now, $\forall t (t < 10) = \text{TRUE}$ (All values are greater than 10).

29. TRC EXAMPLE 1

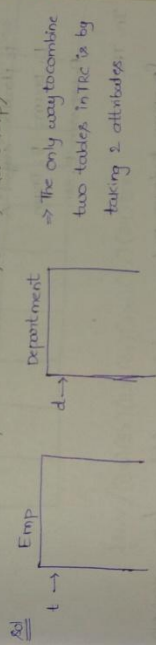
List the names and Address of all employees who work for the "Research" department. The database schema is:

Employee (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary, Super-SSN, Department) (Fname, Dnumbert, Mgr-SSN, Mgr-Startdate) (DNO)
 Dept-locations (Dnumbert, Dlocation)

Project (Pname, Pnumbert, Plocation)

works-on (E-SSN, Pno, Plocno)

Dependent (E-SSN, Dependent-name, sex, Bdate, Relationship)



$\Rightarrow$ Now, after choosing the variables we are going to make one of the variables free, in general we free the variable whose values should be printed. Now, [Fname, Address] are the values that should be printed and they are present in Emp table $\Rightarrow$ free the variable pointing to Employee table $\Rightarrow$ free the variable t .

$\Rightarrow$ The cross product in Relational Algebra is equal to $\exists$ in the TRC

$\{t: \text{Fname}, t: \text{Minit}, t: \text{Lname}, t: \text{Address} / \text{Employee}(t) \text{ AND } (\exists d) (\text{Department}(d) \text{ AND } d: \text{Dname} = \text{"Research"}) \text{ AND } d: \text{Dnumbert} = t: \text{DNO} \}$

30. TRC Example 2

For every project located in "Stafford" list the project number, the controlling department number, and the department managers' lastname, birthdate, and address. The database schema is same as in the previous problem.

First of all we want to find the project location from "project table".

Project		
P →	location	no
	no	ssn

Department	
d →	no
	no-ssn

Employee		
m →	ln	birthdate
	ssn	address

In Relational Algebra

$$\sigma^{p.location = "Stafford"}(P \times D \times E)$$

$$P.location = "Stafford" \wedge$$

$$P.no = D.no \wedge$$

$$D.no-ssn = E.ssn$$

In TRC

{P.no, P.location, m.lastname, m.birthdate, m.address / PROJECT(P) AND

EMPLOYEE(m) AND P.location = "Stafford" AND

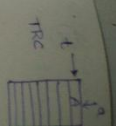
((∃d) (DEPARTMENT(d) AND P.no = d.no

AND d.no-ssn = m.ssn)}

38. DOMAIN RELATIONAL CALCULUS INTRODUCTION

The main difference between the TRC and DRC is the way in which we use the variables

⇒ In case of Domain Relational Calculus we need more variables than the TRC



SQL = $\exists$
 IGM = $\exists$

39. DRC - E

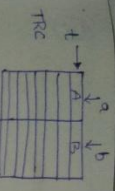
List the

'JOHN B

Now the present in

Emp

79



$t.A = 10$ in TRC
 $a = 10$ in DRC

Q5

SQL based on (TRC and RA)

IGM = QBE (Query By Example) - based in DRC.

39. DRC - Example 1

List the Birth date and the address of the employee whose name is 'JOHN B SMITH'

Now the details like Birth date, address of John-b-Smith are present in employee table.

Employee	q	r	s	t	u	v	w	x	y	z
FN	MN	LN	SSN	Bdate	Add	Sex	Sal	Supr-sin	Dno	

$\{q, v / (aq)(az)(as)(at)(aw)(ax)(ay)(az)\}$

(EMPLOYEE(aqrstuvwxyz) AND

$q = \text{'JOHN'}$ AND $r = \text{'B'}$ AND $s = \text{'SMITH'}$)}

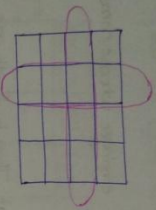
TRC - DRC - NOT much needed for GATE only
Simple Question will be asked

1. INTRODUCTION TO SQL

SQL is based on Relation Algebra, TRC

SEQUEL = prior version = Sequential English Query language

Table - Row - Column



CREATE SCHEMA Company AUTHORIZATION

2. CREATING A TABLE AND CONSTRAINTS ON IT

CREATE TABLE EMPLOYEE

(NAME VARCHAR(15) NOT NULL,

SSN CHAR(9) NOT NULL,

DATE DATE

Super-SSN CHAR(9)

PRIMARY KEY(SSN)

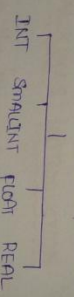
FOREIGN KEY(Super-SSN) REFERENCES EMPLOYEE(SSN);

);

primary key allows NOT NULL by default

Foreign key allows NULL by default.

The various datatypes that are used is NUMERIC, CHAR(n), VARCHAR(n)



Correlation operator etc || xyz = abxyz

NUM, DNO INT NOT NULL CHECK (DNO > 0 AND DNO < 20) => checks the vacancies within range

(8)

=> Create Con

Drave

3. REFEREN

=> whenever

the default

Emplo

FOREIGN

=> Now, inst

they are

=> The vac

{SETNULL

when we pr

Employee

=> If i delete

=> If it is

=> If SET

replaced

=> If the r

=> If Relo

(78)

write is
see in result

5. ALIASING

for each employee, retrieve the employee's first and lastname and the first and lastname of his immediate supervisor.

Employee (Fname, Minit, Lname, Ssn, Bdate, Address, sex, salary, super-ssn,
Employee (Dno)
Department (Dname, Dnumber, Mgr-ssn, Mgr-start date) → Supervisor

Dept-locations (Dnumber, Dlocation)

Project (pname, pnumber, plocation)

works-on (Essn, pno, hours)

Dependent (Essn, Dependent-name, sex, Bdate, Relationship).

Aliasing Employee as E

E.DNO

(Employee)
<ssn=10>

Employee

ESSN	FN	LN	SSN
1	A	B	10

S

SSSN	FN	LN	ESSN
10	C	D	

(we are doing
cross product on
the same table
Rename it as S)

select:

work for

SELECT E.FN, E.LN, S.Fname, S.Lname

FROM Employee E, Employee AS S

WHERE E.ESSN = S.SSN;

6. DUPLICATE TUPLES AND SET OPERATIONS

⇒ we can use the keyword DISTINCT to eliminate the duplicates.

SELECT DISTINCT Fname

FROM Employee

WHERE DNO = '4';

⇒ All the Fnames of employee
who work for dept no '4' (No
duplications)

Now,

(SELECT DISTINCT FROM Employee WHERE DNO=4) UNION
(SELECT DISTINCT FROM Employee WHERE DNO=10).
All the
whose
DNO=4/10
No Repetitions.

UNION - Eliminates duplicates + SELECT DISTINCT

UNION ALL - Does not Eliminate duplicates + SELECT ALL

R	S
a1	a1
a2	a3
a3	a5

R UNION S
a1
a2
a3
a5

R UNION ALL S
a1
a2
a3
a1
a5

1. PATTERN MATCHING AND AGGREGATE OPERATORS

The 'LIKE' is used for string comparing / comparing

Now, if I want to find all the employees whose names contains "Ravi"

⇒ The 2 operators used are

% = no. of '0' or more characters

'_' = Single character

SELECT
FROM Employee
WHERE

Frame LIKE 'Ravi%'

% Ravi % ⇒ Before Ravi there can be any
no. of characters
After Ravi there can be
any no. of characters

⇒ Now if

→

⇒

⇒ Now if

{ SELECT
FROM
BETW

2. ORDER BY

Retrieve a list
ordered by de
by last name, +

⇒ Suppose if

{ A
B
A
C
B

⇒ ORDER BY

⇒ "Order By" clause
select query.

⇒ Now if I want employees whose third letter in frame is 'v', then

→ Frame LIKE ' _ _ v _ % ' ⁽¹⁰⁾ After that there can be anything

1st character → 2nd character → 3rd character

⇒ Frame || Name ⇒ Outputs Concatenated Names.

⇒ Now if I want to increase the salary of all the employees by 10% then

```

{ SELECT Frame, 1.1 * SALARY
  FROM EMPLOYEE;
  BETWEEN ⇒
  SELECT *
  FROM EMPLOYEE
  WHERE SALARY BETWEEN 10,000 AND 20,000;
  OR 10,000 ≤ SALARY ≤ 20,000.
  
```

8. ORDER BY

Retrieve a list of employees and the projects they are working on, ordered by department and within each department ordered alphabetically by lastname, then first name.

⇒ Suppose if I have a Relation like

A	a	b
B	a	c
A	b	c
C	c	b
B	c	b

output ⇒

A	a	b
A	b	c
B	a	c
B	c	b
C	c	b

first order by dept. name, if they are same then order by lastname and then by first name.

⇒ ORDER BY ⇒ used to order the output

⇒ "OrderBy" clause can be applied on the attributes appearing in Select Query.

(100) All employees whose DNO = 4/10

one can be null
can be
can be
can be

(162)

SELECT D.Dname, E.Lname, E.Fname, P.projectname

FROM EmployeeE, Workson W, Project P, Department D

WHERE D.Dnumber = E.Dno AND E.ssn = W.ssn AND W.pno = P.pno

ORDER BY D.Dname, E.Lname, E.Fname;

⇒ The default of ORDER BY is Ascending Asc

Descending 'desc' = Not default

9. EXAMPLE ON ORDER BY

R

⇒ select ABC FROM R ORDER BY A,B,C

A	B	C
1	2	3
1	2	1
2	1	3
2	1	1
3	5	4
3	4	3

A	B	C
1	2	1
1	2	3
2	1	1
2	1	3
3	4	3
3	5	4

⇒ SELECT ABC FROM R ORDER BY A DESC, B, C.

A	B	C
3	4	3
3	5	4
2	1	1
2	1	3
1	2	1
1	2	3

10. INSERT

The comm

→ INSERT

→ INSERT

⇒ Now, if

values

R
1
2

11. DELETE

Now, if

then,

→ DELETE

→ DROP

→ UPDATE

12. DELETE

→ The probl

it is

Now, if

→ Now

10. INSERT

The command used to insert a tuple in the relation is INSERT

→ INSERT INTO EMPLOYEE VALUES ('Ravi', ('Ravada', '1234', '4'));

→ INSERT INTO EMPLOYEE ('Name', 'LN', 'SSN', 'DNO') VALUES

('Ravi', 'Ravada', '1234', '4');

⇒ Now, if i want to insert the values in a table in which are the

values present in another table then

R	C	D	E
1	2	3	4
5	6	7	8
9	10	11	12

S	C	D	E
1	2	3	4
5	6	7	8
9	10	11	12

INSERT INTO R (A,B) SELECT C,D/E
FROM S (C,D,E) WHERE

11. DELETE AND UPDATE

Now, if i want to delete all the employees whose lastname is 'Ravi' then,

→ DELETE FROM EMPLOYEE WHERE LN = 'Ravi';

→ DROP TABLE EMPLOYEE

→ UPDATE EMPLOYEE SET salary = salary * 1.1, WHERE DNO=5;

updates the salaries of all Employees whose DNO=5

12. DEALING WITH NULL VALUES

⇒ The problem with the NULL values is we don't know how to interpret it when we are comparing especially in Boolean variables the the null values can be TRUE/FALSE

⇒ Now if i have to do (x < y)

AND	T	F	UK
T	T	F	UK
F	F	F	F
UK	UK	F	UK

OR	T	F	UK	NOT	A
T	T	T	T	T	F
F	F	T	T	F	T
UK	T	UK	UK	UK	UK

→ In SQL every NULL is considered to be distinct. So just to deal with them two new operators "IS" AND "ISNOT" are introduced.

13. IN

I want to find out all the Frames, Addresses of employees who work for dept = {2, 3, 4, 5}

```
SELECT Frame, Address
FROM EMPLOYEE
WHERE Dno IN (1, 2, 3, 4);
```

(or)

DNO=1 V DNO=2 V DNO=3 V DNO=4

Find Frame, Address of employees who work for department location in 'Skatford'

```
SELECT Frame, Address
FROM EMPLOYEE
WHERE DNO IN (SELECT DNO
```

FROM DEPT-LOCATIONS

WHERE DLOCATION='Skatford');

Frame Add	DNO
A	3
C	D
E	F
	3

DNO	Block
1	Skatford
2	Skatford
3	Skatford

14. ANY, ALL, SOME

```
SELECT DISTINCT Essn
FROM WORKS-ON
WHERE (Fno, HOURS) IN (SELECT Fno, HOURS FROM WORKS-ON
WHERE Essn='10');
```

E	
1	
2	
1	
10	
10	
10	

ON S
ON

Find Name of all emp

```
SELECT F
FROM E
WHERE
```

15. NESTED

RETRIEVE the name SELECTED FROM EMP WHERE

in both on Query

(104)

E	P	H
1	10	10
2	20	10
1	30	10
10	1	10
10	20	10
10	30	10

⇒ The given query will give,

(1, 10)
(20, 10)
(30, 10)

(105)

⇒ The outer query produces (2)

ON Some
ON ANY

↳ The query is find out all the employees who have worked on some project on which employee no. 10 has worked for the same no. of hours.

Find Name of all employee whose salary is greater than the salary of all employee in department no. 5

```
SELECT Fname
FROM EMPLOYEE
WHERE SALARY > ALL (SELECT SALARY
FROM EMPLOYEE
WHERE DNO=5);
```

15. NESTED CORRELATION QUERY

RETRIEVE the name of each employee who has a dependent with the same name and is the same sex as the employee.

```
SELECTED (E.Fname)
FROM EMPLOYEE AS E
WHERE E.SSN IN (SELECT FROM DEPENDENT AS D
WHERE (E.Fname) = D.Dependent_name
AND E.Sex = D.Sex);
```

If the same attribute is present in both outer and inner loops then that query is called co-related query.

Employee

Frame	SSN	Sex
A	501	F
B	502	M
C	503	M

Dependent

Dependent Name	Sex	SSN
A	F	501
B	M	502
F	M	503

Inner Query produces

ESsn
501
502

Outer Query produces

A
B

16. EXISTS AND NOT EXISTS

SELECT Frame
FROM EMPLOYEE AS E
WHERE EXISTS (SELECT *

FROM DEPENDENT AS D
WHERE E.Ssn = D.EsSn AND
E.Sex = D.Sex AND
E.Frame = D.DepartmentName);

=> If the Internal Query returns some value the EXISTS will return TRUE and if the internal query doesn't return anything the EXISTS will return FALSE.

Retrieve the Frames of employees who have no dependents.

SELECT Frame
FROM EMPLOYEE

WHERE NOT EXISTS (SELECT * FROM DEPENDENT WHERE Ssn = EsSn);

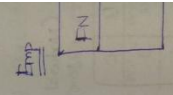
-> If this query returns something then NOT EXISTS returns FALSE
-> If this query returns nothing then NOT EXISTS returns TRUE.

17. EXISTS

List the

SELECT
FROM
WHERE
AND
EXISTS

SELECT
FROM
WHERE
AND



18. EXISTS

Retrieve
Controlled

SELECT
FROM E
WHERE

Emp

SSN
1
2
3
4

17. EXISTS Example 1

List the frames of managers who have at least one dependent

SELECT frame

FROM Employee

WHERE EXISTS (SELECT * FROM Department WHERE SSN = E.SSN)

AND

EXISTS (SELECT * FROM Department WHERE SSN = Mgr.SSN)

SELECT frame

FROM Employee

WHERE EXISTS (SELECT * FROM Department WHERE SSN = E.SSN)

AND EXISTS (SELECT * FROM Department WHERE SSN = Mgr.SSN)

Emp

FN	SSN
1	10
2	20
3	30
4	40

Department

SSN	Dept
10	1
20	2
30	3
40	4

Department

SSN	Dept
10	1
20	2
30	3
40	4

for (i = 1 to n)
{
 for (j = 1 to n)
 for k = 1 to n
 }
}

18. EXISTS Example 2

Retrieve the frame of each employee who works on all the projects controlled by department number 5.

SELECT frame

FROM Employee

WHERE NOT EXISTS (SELECT Project Number
FROM Project
WHERE Dept = 5) EXCEPT (SELECT Proj FROM
WORKS-ON WHERE SSN = E.SSN)

RETURNS TRUE

Emp

SSN	FN
1	Ravi
2	
3	
4	

Project

Proj	Dept
1	5
2	5
3	5
10	1
20	2
30	4

PNO	ESSN
10	1
1	1
2	1
3	1
1	2
2	2

Ravi will be printed.

19. JOINS

108

SELECT Frame, LName, Address FROM (EMPLOYEE JOIN DEPARTMENT
ON DNO=DNUMBER)

WHERE Dname = 'Research';

SELECT Frame, LName, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMENT
AS DEPT (Dname, Dno, Mgrname, Mgrdate))

WHERE Dname = 'Research';

SELECT Ename, AS Employee-name, S.LName AS Supervisor-name
FROM (EMPLOYEE AS E LEFT OUTER JOIN
EMPLOYEE AS ON E.Super_ssn = S.ssn);

Employee

Eno	A
1	A
2	B
3	C
4	D
5	E
6	F

R

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
4	d	7	j
5	e	8	k
6	f	9	l

→ For Natural join

R ⋈ S (Natural join)

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i

(R ⋈ S) (Left outer join)
A=C

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
4	d	N	N
5	e	N	N
6	f	N	N

(R ⋈ S) (Right outer join)
A=C

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
N	N	7	j
N	N	8	k
N	N	9	l

- ⇒ while
- 3) SELECT
- 4) SELECT
- ⇒ min, max
- ⇒ Select Sum
- ⇒ Avg, Std
- data
- ⇒ SELECT
- ⇒ Select
- ⇒ Select
- ⇒ Select

10. AGGREGATE FUNCTIONS 1

The Aggregate functions are Avg, min, max, sum, count

Employee

ENO	ENAME	SALARY	HRA
1	A	1	2
2	B	2	4
3	C	4	6
4	D	4	8
5	E	4	NULL
6	F	NULL	6

1) SELECT sum(salary), avg(salary)
FROM EMPLOYEE

Sum	Avg
15	3

2) SELECT sum(salary) AS Total, avg(salary)
AS Average FROM EMPLOYEE

Total	Average
15	3

=> while calculating the Average NULL values are not considered

3) SELECT (max(salary) - min(salary)) as diff FROM EMPLOYEE:

Diff
3

4) SELECT COUNT(*) as Total FROM EMPLOYEE:

Total
6

=> min, max are applied on Numbers, Strings, Dates

=> SELECT max(Name) FROM EMPLOYEE:

max
F

=> Select sum (name) FROM Employee X Not valid

=> Avg, Sum are applied only on Numerical Data not on any other data types.

(1, 2, 4, 4/4)

Sum
5

=> SELECT COUNT(salary) FROM Employee

(1, 2, 4)

Count
3

=> Select COUNT (salary) FROM Employee
distinct

Count
4

=> Select COUNT (salary + HRA) FROM Employee

Count
3

=> Select SUM(salary + HRA) FROM Employee

(1, 2) = 3
(2, 4) = 6
(4, 6) = 10
(4, 8) = 12
(NULL) = NULL
(NULL) = NULL
(3, 6+10+12)

21. AGGREGATE FUNCTIONS 2

$$\text{COUNT}(\text{salary}) = 5$$

$$\text{COUNT}(\text{DISTINCT salary}) = 3$$

$$\text{SUM}(\text{salary}) = 15$$

$$\text{SUM}(\text{DISTINCT salary}) = 7$$

$$\text{AVG}(\text{salary}) = \frac{\text{SUM}(\text{salary})}{\text{COUNT}(\text{salary})} = \frac{15}{5} = 3$$

22. GROUP BY

R

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

1) select * from R GROUP BY A

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

2) select * from R GROUP BY A, B

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

3) select A, B, COUNT(*) from R

GROUP BY A

A	B
1	2
2	1
2	2

Whenever you have some attributes in Group By clause then they should always appear in select clause

4) For each department, retrieve the no. of employees in the department and their average salary

SELECT DNO, COUNT(*), AVG(salary),

FROM EMPLOYEE

GROUP BY DNO

⇒ Every NULL value will be treated as a separate group. (v.v. imp)

7. TRANS

Transaction

A Transaction

logical unit

Transaction

R(A)

A = A

W(N)

R(B)

B = C

W(B)

Let us co

Let T₁ Re

T₂ R

R(A) =

A = 450

(50)

W(A) = 4

Transaction

1) Atomicity

2) Consistency

3) Isolation

4) Durability

110

$$\rho = \frac{\text{Sum of Distinct } a_i}{\text{Count(Distinct } a_i)}$$

$$= 1/3$$

$$K(R) = \frac{1}{4} = \frac{1}{n(R) + 1}$$

$$SRL(R) = 4$$

1. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

Transaction:

A Transaction is a collection of operations that forms a single logical unit of work.

Transferring money

$$R(A) \rightarrow I/O$$

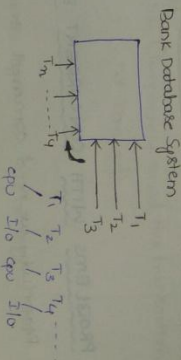
$$A = A - 50; -90$$

$$W(A); \rightarrow I/O$$

$$R(B);$$

$$B = B + 50$$

What's the sequence of steps?



Let us consider a problem of Transactions.

Let T_1 Represent transferring money from A to B

T_2 Represent withdrawing money from A

$$T_1 \quad T_2$$

$$R(A) = 500$$

$$A = 450$$

$$(50 \text{ £ withdrawn})$$

$$R(A) = 500$$

$$A = 400 \text{ (100 £ withdrawn)}$$

$$W(A) = 450$$

Transaction Properties (ACID PROPERTIES)

1) Atomicity: All or None should happen $\rightarrow$ Transaction Manager

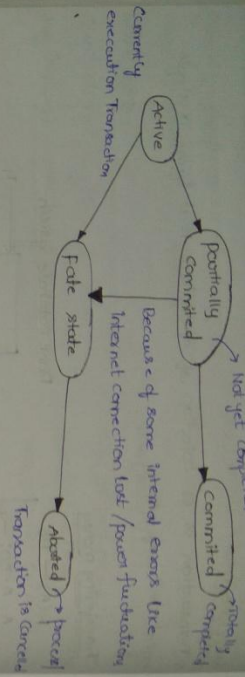
2) Consistency: Correctness $\rightarrow$ user/application programmer takes care

3) Isolation: Each Transaction must be executed without knowing what is happening with others - Concurrency Control Manager

4) Durability: All updates done by a Transaction must become permanent - Recovery Manager takes care

4p. (V2.50p)

States of Transaction / Transaction States



2. PROBLEMS WITH CONCURRENT EXECUTION

- The Reason why we need concurrent execution is
- For avoiding long waiting time
- Transaction consists of multiple steps. Some involve I/O activities and other involve CPU activities. In a Computer System CPU and I/O operations can be done in parallel. Therefore I/O activities can be done in parallel with processing at the CPU.
- The processor and Disk utilization increases

Schedule:

It represents the order in which instructions of a transactions are executed.

Loss update problem: (w-w Conflict)

Initial value of A = 100	
$A = 100$ $A \rightarrow B$	$A = 50$ $A \rightarrow A$
Read (A)	
$A = 50$	
Read (A)	$A = 100$
$X \rightarrow 0.0441 \rightarrow A = 100 * 100$	
$A = A + X \rightarrow A = 104$	
Write (A)	$A = 104$
Write (A)	$A = 104$
Read (B)	
$B = B + 50$	
Write (B)	

Test: Two write operations of diff Transactions and in this there is no Read then and write operation overwrites the write operation

Dirty Read Problem

Roll Back	
$A = 100$ Read (A)	$A = 50$ Write (A)
$A = 50$	
Read (B)	
$B = B + 50$	
Write (B)	

Roll Back
Because of Roll back $\Rightarrow A = 100$ but the not considered

Unrepeatable Read

T_1	
Read (A)	
Read (A)	
Write (A)	

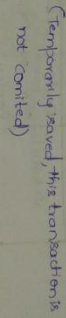
Phantom Tuple

T_1	
Read (A)	
Read (A)	
Write (A)	

1	Change	500	500
2	Change	5000	5000
3	Change	4000	4000
4	Change	3500	3500

113

process



not considered

(a) Non-Repeatable Read Problem (R-w)

called un-separable reads.

$\cos(e(A))$	Fault
--------------	-------

Phantom
Tupel (phantom phenomenon)

	T_1	T_2
<u>CNO</u> <u>Game</u> <u>Scholarship</u> 1 A 5000 3 C 4000 Select * from Emp where Salary > 3000		
Insert into Emp values (4,0,3500);		

QID	exam	sal	select #
1	A	5000	from Emp
3	C	4000	where
4	O	3500	sal > 3000

$$50x \frac{1}{100} = 2.1$$

$$+ x = 50 + 2 = 52$$

A = 52

Incorrect Summary Problem

114

T1	T2
Sum = 0	Sum = 0
Read(K)	Let K = 50
Sum = Sum + K	Sum = Sum + K
Sum = 50	Sum = 50
Read(X)	Sum = 50
X = 600	Sum = 50
Write(X)	Sum = 50
Read(Y)	Sum = 50
Y = Y + 200	Sum = 50
Write(Y)	Sum = 50

Be cause of this operation
y will be changed and
as a result sum must
be changed but if we retrieve
sum it gives 850

Before T1	After T1
K = 50	50
X = 100	600
Y = 200	400
Sum = 850	1050

wrong summary

3. NO-OF SCHEDULES POSSIBLE

Let us say there are two operations associated with a particular transaction and there are two transactions T1 and T2. Now, what are the diff ways to combine these two together.

T1	T2	S1	S2	S3	S4	S5	S6
a1	b1	a1	b1	a1	b1	a1	b1
a2	b2	a2	b2	a1	b1	a1	b1
		b1	a1	a2	b2	a2	b2
		b2	a2	a2	b2	b2	a2

⇒ In these schedules the order should be consistent which means a1 should be executed only before a2 and a2 should not be executed before a1
⇒ All these schedules will not give us correct results (not consistent)
⇒ we need to find the correct schedule that guarantee the consistency

4. TYPES OF

Serial schedule

Transactions T1

T1	T2
R(A)	
W(A)	
R(A)	

constant

⇒ serial schedule

⇒ when Transaction

state

⇒ If there are

Complete sched

⇒ At the end

Present then

T1	T2
R(A)	R(A)
W(A)	
Commit	
W(A)	
Abort	

Recoverable

Roll Back

T1	T2
R(A)	R(A)
W(A)	
Commit	
W(B)	
Commit	

database.

5. TYPES OF SCHEDULES

Serial schedule: Serial schedule means one after the other. If we have two transactions T_1 and T_2 , then T_1 and T_2 are called serial schedules.

$T_1 \rightarrow T_2$

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A+100
	W(A)

$T_2 \rightarrow T_1$

T_1	T_2
	R(A)
	A=A+50
	W(A)
R(A)	
A=A+100	
W(A)	

Not serial schedule

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A+100
	W(A)

constant states of operation

Serial schedules are the ones that doesn't interleave the actions of operations of different transaction.

When transactions are executing serially then they ensure a consistent state.

If there are n transactions then there are $n!$ serial schedules.

Complete schedule

At the end of every transaction if the lines commit and abort are present then the schedule is called complete schedule.

T_1	T_2
R(A)	
A=A+50	
W(A)	
Commit	
	R(A)
	A=A+100
	W(A)
	Abort

Recoverable schedule

Non-

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	X=A*100
	W(A)
	Commit
R(B)	
B=B+50	
W(B)	
Commit	

Roll Back is not possible because it is committed.

Non-Recoverable schedule

Recoverable schedule

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	X=A*100
	W(A)
	Commit
R(B)	
B=B+50	
W(B)	
Commit	

Roll Back

CASCADING SCHEDULE

116

T1	T2	T3
R(A)		
w(A)	R(A)	
	w(A)	R(A)
Commit	Commit	Commit

Roll Back

Roll Back

⇒ CASCADING
ABORT

T1	T2	T3
R(A)		
w(A)		
Commit	R(A)	
	w(A)	
	Commit	R(A)
		w(A)
		Commit

Every cascadeless is Recoverable

CASCADELESS SCHEDULE

STRICT SCHEDULE

T1	T2
R(A)	
w(A)	
Commit	
	w(A)

⇒ 2nd Read/Write until the other one commits



S- STRICT SCHEDULE

C- CASCADELESS SCHEDULE

R- RECOVERABLE SCHEDULE

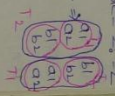
26
T1
T2

of database.

Now let us say there are transactions $T_1, T_2, T_3, \dots, T_m$ and no of operations in each transaction is $n_1, n_2, n_3, \dots, n_m$ then the no. of schedules that are possible are

$$\text{No. of schedules possible} = \frac{(n_1 + n_2 + n_3 + \dots + n_m)!}{(n_1!)(n_2!)(n_3!)\dots(n_m!)}$$

$\Rightarrow$ Now if there is no interleaving (No Transaction is supposed to enter in middle when a particular transaction is executing) then the no. of serial schedules possible are $m!$

In the above example the no. of serial schedules possible = $2! = 2$
 $\Rightarrow \frac{T_1}{T_2} \mid \frac{T_2}{T_1}$


5. RESULT EQUIVALENT SCHEDULES

Two schedules are said to be Equivalent if they follow same rules they are

1. Two schedules are said to be Result Equivalent if they produce same final database state for a given initial state of database.

S1	S2
R(A)	R(A)
A = 100	A = 100
A = 110	A = 110
W(A)	W(A)
A = 110	A = 110

→ check whether these two schedules are Equivalent / not

S1		S2	
T1	T2	X Y	X Y
R(x)		2 5	2 5
X = x+5		3 10	6 10
w(x)		11 10	
R(y)			
y = y+5			
w(y)			

6. CONFLICT EQUIVALENT AND CONFLICT SERIALIZABLE

Conflict Operations

T_i
 $R(A)$ $w(A)$ } Conflict operations
 $w(A)$ $R(A)$ } accessing the same data item
 $w(A)$ $w(A)$ } and one of the operations is write
 $R(A)$ $R(A)$ } then it is a conflict operation
 Operating on diff data } Non-conflict operations

Now, Two schedules are said to be conflict Equivalent if all the conflicting operations in both the schedules must be executed in the same Order.

T1	T2
R(A)	w(R)
w(A)	R(A)
R(B)	w(B)
w(B)	

Conflicts (R-w)

T1	T2
R(A)	w(R)
w(A)	R(A)
R(B)	w(B)
w(B)	

w-w

∴ These 2 are Conflict

1) Test which

S1: R1(A) R2
S2: R2(B)

T1	
R(A)	
w(A)	

No-Conf

2) S1: R1(A) w1

S2: R2(B) w2

T1	
R(A)	
w(A)	

Conflict

w2(B)

CONFLICT
A schedule Equivalent

7. CONFLICT

Now, check not

1) Test which of the following are conflict serializable

S1: $R_1(A)$ $R_2(B)$ $w_1(A)$ $w_2(B)$

S2: $R_2(B)$ $R_1(A)$ $w_2(B)$ $w_1(A)$

S1

T1	T2
$R_1(A)$	
	$R_2(B)$
$w_1(A)$	$w_2(B)$

No conflicting actions

S2

T1	T2
	$R_2(B)$
$R_1(A)$	
$w_1(A)$	$w_2(B)$

No conflicting actions

→ S1 and S2 are not conflicting

2)

S1: $R_1(A)$ $w_1(A)$ $R_2(B)$ $w_2(B)$ $R_1(B)$ $R_2(A)$

S2: $R_2(B)$ $w_2(B)$ $R_1(B)$ $R_2(A)$ $R_1(A)$ $w_1(A)$

S1

T1	T2
$R_1(A)$	
$w_1(A)$	
	$R_2(B)$
	$w_2(B)$
	$R_1(B)$
	$R_2(A)$

Conflicting operations

$w_2(B) \rightarrow R_1(B)$

WR $\neq$ RW

S2

T1	T2
$R_2(B)$	
$w_2(B)$	
	$R_1(B)$
	$R_2(A)$
	$R_1(A)$
	$w_1(A)$

Conflicting operations

$R_1(B) \rightarrow w_2(B)$

NOT conflict Serializable

CONFLICT SERIALIZABLE

A schedule is said to be conflict serializable if it is conflict

Equivalent to a serial schedule

3. CONFLICT SERIALIZABLE EXAMPLE

Now, check whether these two schedules are conflict serializable or not.

S1: T1, T2 (not serial)
S2: T1, T2 (conflict free)
S3: T2, T1 $\Rightarrow$ legal/serial

S1

T1	T2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

$R_1(A) \rightarrow w_2(A)$
 $w_1(A) \rightarrow R_2(A)$
 $w_1(A) \rightarrow w_2(A)$

$R_1(B) \rightarrow w_2(B)$
 $w_1(B) \rightarrow R_2(B)$
 $w_1(B) \rightarrow w_2(B)$

S2

T1	T2
R(A)	
w(A)	
R(B)	
w(B)	
	R(A)
	w(A)
	R(B)
	w(B)

$R_1(A) \rightarrow w_2(A)$
 $w_1(A) \rightarrow R_2(A)$
 $w_1(A) \rightarrow w_2(A)$
 $R_1(B) \rightarrow w_2(B)$
 $w_1(B) \rightarrow R_2(B)$
 $w_1(B) \rightarrow w_2(B)$

Conflict
Serializable

PROCEDURE

- Construct a transaction operation.
- If the Dine is not conflict
- If the graph conflict ser

Ex

T1	T2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

2

S1

T1	T2
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(B)
w(B)	
	w(B)

$R_1(A) \rightarrow w_2(A)$
 $w_1(A) \rightarrow R_2(A)$
 $w_1(A) \rightarrow w_2(A)$
 $R_1(B) \rightarrow w_2(B)$
 $w_1(B) \rightarrow R_2(B) \rightarrow w_1(B)$
 $w_1(B) \rightarrow w_2(B)$

S2

T1	T2
R(A)	
w(A)	
R(B)	
w(B)	
	R(A)
	w(A)
	R(B)
	w(B)

$R_1(A) \rightarrow w_2(A) \rightarrow w_1(A) \rightarrow R_2(A)$
 $w_1(A) \rightarrow w_2(A)$
 $R_2(B) \rightarrow w_2(B)$
 $w_1(B) \rightarrow w_2(B)$
 $w_1(B) \rightarrow R_2(B)$

T1	T2
	R(A)
	w(A)
	R(B)
	w(B)
R(A)	
w(A)	
R(B)	
w(B)	

No
 $R_1 \rightarrow w_2$

Not
Conflict Equivalent.
No $R_2(B) \rightarrow w_1(B)$

Ex :

T1	T2
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(B)
w(B)	
	w(B)

Ex :

T1	T2
	R(A)
	w(A)
	R(B)
	w(B)
w(A)	
	R(A)
	w(A)
	R(B)
	w(B)

The Relation S1 is not Equivalent to any of the serializable schedule

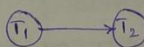
$\therefore S_1$ is not conflict serializable.

PROCEDURE FOR CONFLICT SERIALIZABILITY USING PRECEDENCE GRAPH

- ① Construct a directed graph where each vertex corresponds to a transaction and each directed edge represents a conflicting operation. (Read-write / write-write / write-read)
- ② If the Directed Graph contains cycles then the concurrent schedule is not conflict serializable.
- ③ If the graph contains no cycles then the schedule is called conflict serializable.

Ex:

T ₁	T ₂
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

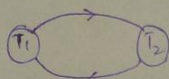


⇒ In the precedence Graph there is no cycle and therefore this schedule is conflict serializable.

⇒ The serial schedule for which this schedule is equivalent to is Topological order of the Graph = T₁T₂.

Ex:

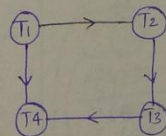
T ₁	T ₂
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(B)
w(B)	
	w(B)



⇒ In the precedence Graph there is a cycle and hence the schedule is not conflict serializable.

Ex:

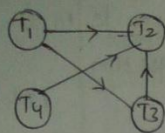
T ₁	T ₂	T ₃	T ₄
	R(x)		
		w(x)	
w(y)			
	R(y)		
	w(z)		
			R(x)
			R(y)



There is no cycle and therefore the schedule is conflict serializable.

⇒ The serial schedule to which the given schedule is T₁T₂T₃T₄ (Topological sort)

Ex



(A) Not conflict serializable

(B) $T_3 T_4 T_1 T_2$

(C) $T_1 T_4 T_3 T_2$

(D) $T_2 T_3 T_1 T_4$

$\Rightarrow$ No cycles in Graph $\Rightarrow$ Conflict serializable

TRICK (By me) (Not By RBR) : Start the sequence which does not have any incoming edge and proceed further.

8. NO. OF CONFLICT SERIALIZABLE SCHEDULES

$T_1: R_1(A) \ w_1(A) \ R_1(B) \ w_1(B)$

$T_2: R_2(A) \ w_2(A) \ R_2(B) \ w_2(B)$

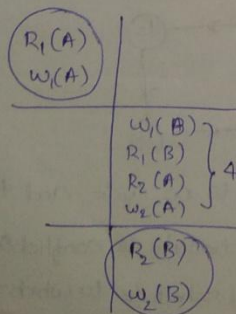
Find the total no. of conflict serializable schedules that can be formed by T_1 and T_2 ?

Now, Try to find the schedules that are Equivalent to $(T_1 \rightarrow T_2)$ or $(T_2 \rightarrow T_1)$ (The schedules equivalent to $T_1 \rightarrow T_2$ / $T_2 \rightarrow T_1$ are called conflict serializable)

Now, try to find out the schedules that are conflict Equivalent to the serial schedule $(T_1 \rightarrow T_2)$

Now, Among $R_1(A) \ w_1(A) \ R_1(B) \ w_1(B)$ and $R_2(A) \ w_2(A) \ R_2(B) \ w_2(B)$ $R_1(A)$ should come first. I cannot write $R_2(A)$ 1st because if I write I get $T_2 \rightarrow T_1$ (but my aim is $T_1 \rightarrow T_2$)

$\therefore$ $R_1(A) \ w_1(A)$ and $R_2(B) \ w_2(B)$ should come in end



4! ways but $R_1 w_1$ and $R_2 w_2$ should be executed as group

$\therefore$ No. of conflict serializable schedules

$$= \frac{4!}{2!2!} = 6$$

Now $(T_2 \rightarrow T_1)$

$R_1(A)$	
$w_2(A)$	

9. VIEW E

Two sched following 3

1) For each in sched read the

$\Rightarrow$ If Trans produced S' also

3) For each of 'A' of in S'

$\Rightarrow$ In summary same order

Ex:

T_1
$R(A)$
$w(A)$
T_2
$R(B)$
$w(B)$

h $\Rightarrow$ Conflict
serializable

s not have

Now ($T_2 \rightarrow T_1$)

	$R_2(A)$	
	$w_2(A)$	
$R_1(B)$	$R_1(B)$	} $\Rightarrow \frac{4!}{2!2!} \text{ ways} = 6$
$w_1(B)$	$w_1(B)$	
$R_1(A)$		
$w_1(A)$		

9. VIEW EQUIVALENT AND SERIALIZABLE

Two schedules 'S' and 'S'' are said to be view Equivalent if the following 3-Conditions are met for each data item (say A)

1> For each data item A if Transaction T_i reads the initial value of A in schedule 'S' then transaction T_i must schedule in 'S' also read this initial value of A.

$\Rightarrow$ If Transaction T_i executes Read - $R(A)$ in schedule 'S' and that was produced by Transaction T_j (if any) then the transaction must in schedule 'S' also read the value of 'A' that was produced by T_j

3> For each data item the transaction (if any) that performs final write of 'A' operation in 'S' must also perform the final write of 'A' operation in 'S'.

$\Rightarrow$ In summary All Read write sequences to be maintained in the same order in both S and S'

Ex:

S1	
T_1	T_2
$R(A)$	
$w(A)$	
	$R(A)$
	$w(A)$
$R(B)$	
$w(B)$	
	$R(B)$
	$w(B)$

S2	
T_1	T_2
$R(A)$	
$w(A)$	
$R(B)$	
$w(B)$	
	$R(A)$
	$w(A)$
	$R(B)$
	$w(B)$

1> Are S1 and S2 View

Equivalent

Initial Read on 'B' in S1 is by T_1

Initial Read on 'B' in S2 is by T_2

Final write on B in S1

Initial Read on 'A' in S1 is performed by T_1 .

Initial Read on 'A' in S2 is performed by T_2 .

Final write on 'A' in 'S2' is done by T_2

Final write on 'A' in 'S1' is done by T_1 .

ne first. 4
(A) 1st because
 $T_2 \rightarrow T_1$ (but my

uted as group

laizable schedules

Now, check for producers and consumers. (write and Read)

124

In S_1

T1	T2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

producer ← w(B) → Consuming

In S_2

T1	T2
R(A)	
w(A)	
R(B)	
w(B)	
	R(A)
	w(A)
	R(B)
	w(B)

producer ← w(A) → consumer

∴ These two schedules are view Equivalent

Ex:

S_1

T1	T2
R(A)	
	w(A)
w(A)	

S_2

T1	T2
R(A)	
w(A)	
	w(A)

S_3

T1	T2
	w(A)
R(A)	
w(A)	

Final write here is by T_1

Final write here is by T_2

∴ $T_1 \neq T_2$ (Not View Equivalent)

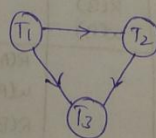
∴ $S_1 \neq S_2$ $S_2 \neq S_3$ (final write of A's are diff)

$S_1 \neq S_3$ (No producer consumer in S_1)

10. VIEW SERIALISABLE EXAMPLES

The method that we use to check view serialisability is by using "poly graphs."

T1	T2	T3
R(A)		
	w(A)	
R(A)		
		w(A)



⇒ No cycle ⇒ The schedule is view serialisable

⇒ Here T_1 should start first which means T_2 T_3 should be executed only after T_1 , so $T_1 \rightarrow T_2 \rightarrow T_3$

⇒ Now, coming to T_2 and T_3 T_2 should start first so $T_1 \rightarrow T_2 \rightarrow T_3$

Ex

T1
R(A)
w(A)
R(B)
w(B)

Ex

T1
R(A)
w(A)

⇒ If A is but a v

Blind write:

⇒ If a sche then there sh

11. SERIALISABLE

⇒ A schedule or view is

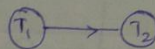
⇒ Now, const

d)

124

Ex

T ₁	T ₂
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)



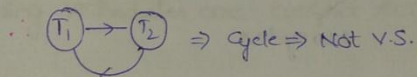
∴ The Graph contains No cycle and so the schedule is view serializable and conflict serializable

Ex

T ₁	T ₂
R(A)	
	w(A)
w(A)	

Says that T₁ should happen first and then T₂

This Says that T₁ should occur last which means T₂ and then T₁



⇒ If A schedule is Conflict serializable then it is view serializable but a view serializable schedule need not be conflict serializable



Blind write: Write without Read is called Blind write

In S₁)

⇒ If a schedule should be view serializable but not conflict serializable then there should be at least one blind operation.

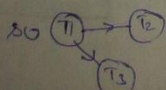
II. SERIALIZABLE SCHEDULES

⇒ A schedule is serializable if it is either conflict serializable or view serializable

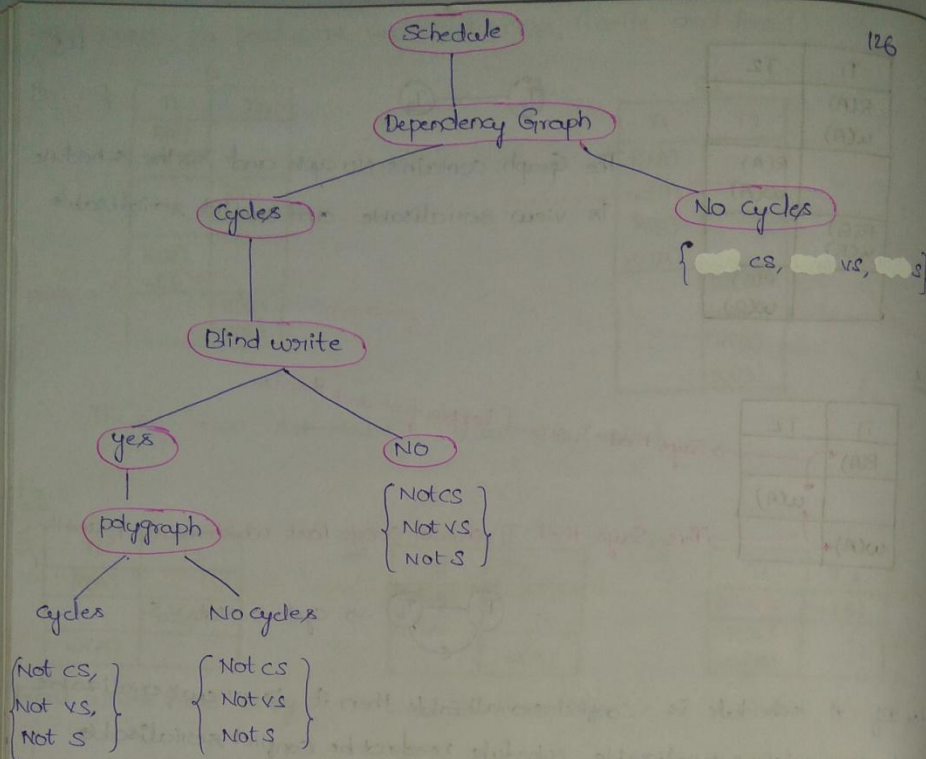
⇒ Now, consider the following tree

⇒ The is View serializable

which means T₂ T₃



should start



How many concurrent schedules can be formed over two Transactions using two operations each?

T_1
 $R_1(A)$
 $w_1(A)$

T_2
 $R_2(A)$
 $w_2(A)$

$$\text{No. of schedules possible} = \frac{(2+2)!}{2!2!} \frac{(n_1+n_2+n_m)!}{n_1!n_2!n_m!}$$

6

$$\text{Non serial schedules} = \text{Total schedules} - \text{No. of serial schedules}$$

$$= 6 - (2!) (\text{No. of Transactions})$$

4

⇒ The no. of concurrent schedules that can be formed over two transactions having n_1 and n_2 operations respectively are

$$= \frac{(n_1+n_2)!}{n_1!n_2!}$$

The no. of

⇒ The no. of having n_1, n_2

⇒ The no. of

Determine whether or not?

$S_1: R_1(A) w_1(A)$

$S_2: R_1(A) R_1(B)$

$S_3: R_1(A) R_2(A)$

$S_4: w_3(A) R_2$

S_1

T1
R(A)
w(A)
R(B)
w(B)

S_3

T1	T2
R(A)	
	R(A)
R1(B)	
	R(B)
w(A)	
	w(B)

The no. of serial schedules possible are $\left[\frac{(n_1 + n_2)!}{n_1! n_2!} \right] - 2$ 127

⇒ The no. of concurrent schedules that can be formed over m -transactions having $n_1, n_2, \dots, n_m$ operations respectively are

$$\left[\frac{(n_1 + n_2 + n_3 + \dots + n_m)!}{n_1! n_2! n_3! \dots n_m!} \right]$$

⇒ The no. of serial schedules are $\left[\frac{(n_1 + n_2 + \dots + n_m)!}{n_1! n_2! n_3! \dots n_m!} \right] - (m!)$

Determine whether the following schedules are conflict serializable or not?

S_1 : $R_1(A) \ w_1(A) \ R_2(B) \ w_2(B) \ R_1(B) \ w_1(B)$

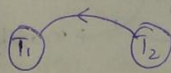
S_2 : $R_1(A) \ R_1(B) \ w_2(A) \ R_3(A) \ w_1(B) \ w_3(A) \ R_2(B) \ w_2(B)$

S_3 : $R_1(A) \ R_2(A) \ R_1(B) \ R_2(B) \ R_3(B) \ w_1(A) \ w_2(B)$

S_4 : $w_3(A) \ R_2(A) \ w_1(B) \ R_2(B) \ w_2(C) \ R_3(C)$

S_1

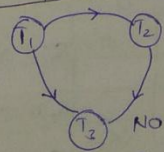
T1	T2
$R(A)$	
$w(A)$	
	$R(B)$
	$w(B)$
$R(B)$	
$w(B)$	



∴ conflict serializable

S_2

T1	T2	T3
$R(A)$		
$R(B)$		
	$w(A)$	
		$R(A)$
$w(B)$		
		$w(A)$
		$R(B)$
		$w_2(B)$

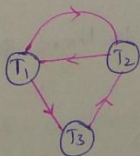


No cycles

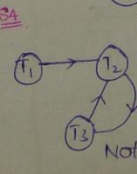
∴ conflict serializable

S_3

T1	T2	T3
$R(A)$		
	$R(A)$	
$R_1(B)$		
	$R(B)$	
		$R_3(B)$
$w(A)$		
	$w(B)$	



Not conflict serializable



Not conflict serializable

12. EXAMPLES ON TYPES OF SCHEDULES

128

Two Transactions T_1 and T_2 are given as follows

$T_1: R_1(A) w_1(A) R_1(B) w_1(B)$

$T_2: R_2(B) w_2(B) R_2(A) w_2(A)$ Now how many serial schedules are possible?

$\Rightarrow$ No. of serial schedules = $m! = 2! = 2$ [$m = \text{No. of Transactions}$]
($T_1 \rightarrow T_2$) ($T_2 \rightarrow T_1$)

$\Rightarrow$ Consider the following schedules

$S_1: \underline{R_2(x)} R_2(y) R_1(x) R_1(y) w_1(x) \underline{R_2(x)}$ ✓

$S_2: R_2(x) R_2(y) w_2(x) R_1(x) R_1(y)$

$S_3: R_2(x) R_2(y) R_3(x) R_2(y) w_2(x) w_2(x)$

which of the above schedules are having un-repeatable Read problem?

$\Rightarrow$ un Repeatable Read problem means if a transaction tries to read the same data two times and in b/w this two reads if some other transaction changes the value then it is called un-repeatable Read

$\Rightarrow$ Now $S_1: R_2(x) \dots \dots w_1(x) R_2(x)$
 $x = 100$ $x = \text{NOT } 100$
 modified the value of x

which of the above schedules is having lost update problem.

$\Rightarrow$ lost update problem means one transaction will write a value and immediately another transaction writes a value without Reading it (one writes over)

S_1 : Only one write is there so no loss update

S_2 : " " " " " " " " " " " "

$S_3: w_1(x) w_2(x)$

one write (w_2) overwrites the other

write $\therefore$ lost update problem

which of the
 $\Rightarrow$ Dirty Read
someone else

$S_2: R_2(x)$

$S_1: w_1(x)$

which of the
strict.

$S_1: R_1(x) R_2(x)$

T
RC
WC
com

$S_2: R_1(x) w_2(x)$

T
R
w
R
co

$S_3: R_3(x) R_1(x)$

Recoverable:

Cascade less

which of the above schedules is having dirty Read problem.

⇒ Dirty Read problem is if someone will write it and before writing someone else will read it this is called Dirty Read problem.

$S_1: R_1(x) \dots w_2(x)(R_1(x))$

T_2 is writing the value of 'x' and R_1 is reading it before it got committed. so it is dirty Read.

$S_1: w_1(x) R_2(x) = \text{Dirty Read problem.}$

Which of the following schedules are Cascadeless, Recoverable and strict.

$S_1: R_1(x) R_2(x) w_1(x) w_2(x) \text{Commit}(C_2) \text{Commit}(C_1)$

T_1	T_2
R(x)	
	R(x)
w(x)	
	w(x) Commit
commit	

⇒ Recoverable : consumer should not commit before the producer commit

⇒ There are no producer and consumer.

The schedule is Recoverable, cascadeless

⇒ Strict says if someone writes a value you should not Read/write it until the previous one commits. ∴ Not strict schedule

$S_2: R_1(x) w_2(x) w_1(x) R_1(x) \text{Commit}(2) \text{Commit}(1)$

T_1	T_2
R(x)	
	w(x)
w(x)	
R(x)	
	Commit
commit	

⇒ No producer consumer ⇒ cascadeless,
⇒ Not strict schedule Recoverable

$S_3: R_3(x) R_1(x) R_2(x) w_1(y) R_2(y) w_2(y) C_3 C_2$

Recoverable : says that producer should commit before the consumer. ∴ Recoverable (T_1 commits before T_2)

Cascadeless : says that producer should commit first then only you have to read it. NOT CASCADELESS ∴ NOT STRICT

T_1	T_2	T_3
		$R_3(x)$
$R_1(x)$		
	$R_2(x)$	
$w_1(y)$	$R_2(y)$	
	$R_3(x)$	
		Commit
commit	commit	

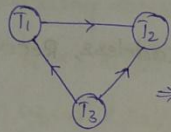
13. EXAMPLES ON CONFLICT SERIALIZABLES

130

which of the following schedules is conflict serializable? For each serializable schedule, determine the equivalent serial schedules?

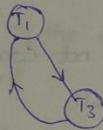
- $r_1(x), r_3(x), w_1(x), r_2(x), w_2(x)$
- $r_1(x), r_3(x), w_3(x), w_1(x), r_2(x)$
- $r_3(x), r_2(x), w_3(x), r_1(x), w_1(x)$
- $r_3(x), r_2(x), r_1(x), w_3(x), w_1(x)$

Now, i)



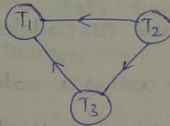
$\therefore$ The schedule is conflict serializable
 $\Rightarrow$ Start with the node that doesn't have any incoming edge = T_3, T_1, T_2

ii)



$\Rightarrow$ Not conflict serializable

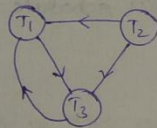
iii)



$\Rightarrow$ Conflict serializable

$\Rightarrow$ serial schedule is T_2, T_3, T_1

iv)



$\Rightarrow$ Not conflict serializable

14. EXAMPLES ON CONFLICTS

Given

$T_1: R(x) R(y) w(x)$

$T_2: R(x) R(y) w(x) w(y)$

Now form the schedules that result in
 WR-conflict, RW-conflict, ww-conflict

WR

T_1	T_2
$R(x)$	
$R(y)$	
$w(x)$	
	$R(x)$
	$R(y)$
	$w(x)$
	$w(y)$

RW

T_1	T_2
$R(x)$	
	$R(x)$
	$R(y)$
	$w(x)$
	$w(y)$
$R(x)$	
$R(y)$	

NW

T_1	T_2
$R(x)$	
$R(y)$	
$w(x)$	
	$w(x)$
	$w(y)$
	$R(x)$
	$R(y)$

15. POLYGR

T_k	T_i
	$w(x)$
	$R(x)$

$S: r_1(A) w_2$

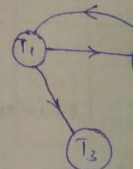
NOTE: $\Rightarrow$ If

is view

$\Rightarrow$ If

write

not



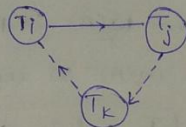
Now, Insert two
 trans actions T_k

T_k	T_1	T_2	T_3
$w(x)$			
	$R(x)$		
		$w(x)$	
			$w(y)$

15. POLYGRAPHS FOR VIEW SERIALIZABLE -- Example 1

T_k	T_i	T_j
	$w(A)$	$R(A)$

⇒ Suppose there are two transactions T_i and T_j at some point of time T_i performs write operation and then T_j Reads it, Now another Transaction T_k executes write operation - the it should be before T_i and before T_j so the polygraph will be

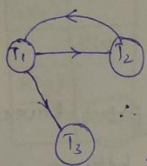


$S: r_1(A) \ w_2(A) \ w_3(A) \ w_3(A)$

NOTE: ⇒ If a schedule is conflict serializable then the schedule is view serializable, so the 1st check for VS should be CS.

⇒ If the schedule is not CS, now check if there are Blind writes or not, in case if there are Blind writes and it is not CS then only we should check for VS.

CS X	CS X	CS ✓	CS = Conflict serializable
Bw X	Bw ✓	VS ✓	Bw = Blind writes
VS X	VS ✓		VS = View serializable

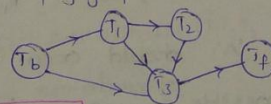


∴ Not CS X ∴ Now check for Blind write (write without Read)
Transaction 2 is writing without Reading (∴ Bw ✓)

Now, Insert two dummy transactions T_b and T_f

CS X
Bw ✓
VS ✓

Now draw poly graph



No cycles in poly graph
The schedule is view serializable

~~$T_1 T_2 T_3$~~
 $T_1 T_2 T_3$

We Assume

T_b = writes data items initially
 T_f = Read all data items

T_b	T_1	T_2	T_3	T_f
$w(A)$				
$R(A)$				
	$w(A)$			
		$w(A)$		
			$w(A)$	
				$R(A)$

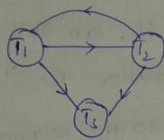
T_b, T_f are dummy transactions

16. POLY GRAPHS FOR VIEW SERIALISABLE EXAMPLE 2

152

$S = r_1(A) w_2(A) r_3(A) w_1(A) w_3(A) \Rightarrow$ find if it is view serialisable / not

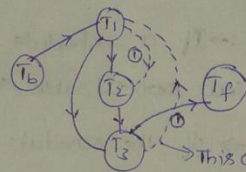
$\Rightarrow$ first check whether it is conflict serialisable / not



$\Rightarrow$ Not conflict serialisable (cycle is present)

$\Rightarrow$ Now, check for Blind writes. $w_2(A)$ is the Blind write because it is not reading 'A' before $w_2(A)$ so $w_2(A)$ is Blind write $\Rightarrow$ Now check for vs. by Adding (T_b and T_f as dummy Transactions)

T_b	T_1	T_2	T_3	T_f
$w(A)$				
	$R(A)$			
		$w(A)$		
			$R(A)$	
	$w(A)$			
			$w(A)$	
				$R(A)$



This creates cycle so remove it
No cycle is present

$\therefore$ The schedule is view serializable and the serial schedule is

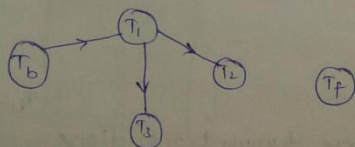
$$\cancel{T_b T_1 T_2 T_3} = \underline{T_1 T_2 T_3}$$

$\Rightarrow$ procedure to draw the above poly graph

1) check for write and Read's of the same object between two different transactions. The first WR is between

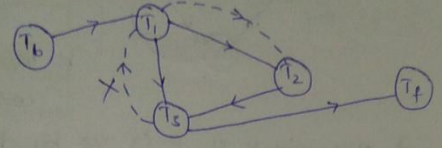
$\Rightarrow w_b(A) R_1(A)$ Now, all the writes of the data item

'A' should occur before T_b or after T_1 , now before T_b is not possible because T_b is the 1st transaction $\therefore$ The write $w_2(A)$, $w_3(A)$ should come after T_1 , and therefore there will edges from T_1 to T_2 and T_3



152
ble (not

⇒ Now again check for write Reads we find $w_2(A), R_3(A)$ so all the writes of 'A' should occur before T_2 and after T_3



133

e is present)

⇒ Now, one write is before T_2 i.e. $w_1(A)$ so there exists an edge from T_1 to T_2 and $w_1(A)$ occurs between T_3 . ∴ there exists an edge from T_3 to T_1 . Since it forms cycle Remove it

use it

⇒ Now

ctions)

⇒ Next WR is between T_3 and T_4 so there exists an edge from T_3 to T_4

17. EXAMPLES ON VERIFYING THE SCHEDULES - I

T_4

This creates cycle so Remove it

schedule

T_1	T_2
$R(x)$	
	$R(x)$
$w(x)$	
	$w(x)$



⇒ Not conflict serializable

⇒ No Blind write (Not vs)

⇒ Recoverable ✓

⇒ Not serializable

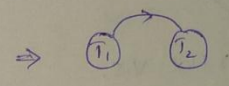
⇒ Cascadeless

⇒ No (producer-consumer)

⇒ Not strict

(Read after writes)
 $w_2(x)$ should be done after $w_1(x)$ commits

T_1	T_2
$w(x)$	
	$R(y)$
$R(y)$	
	$R(x)$



⇒ Conflict serializable

⇒ Recoverable (cannot be said (No commit))

⇒ View serializable

⇒ cascading schedule

⇒ Strict

→ Dirty Read.

between

een

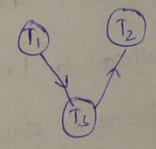
data item

re T_6 is not

write $w_2(A)$,

from T_1 to T_2 and T_3

T_1	T_2	T_3
$R(x)$		
	$R(y)$	
		$w(x)$
	$R(x)$	



⇒ conflict serializable ($T_1 T_3 T_2$)

⇒ View serializable

⇒ Recoverable cannot be decided

⇒ Cascading schedule

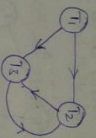
⇒ Not strict.

→ producer-consumer

18. EXAMPLES ON VERIFYING THE SCHEDULES - 2

154

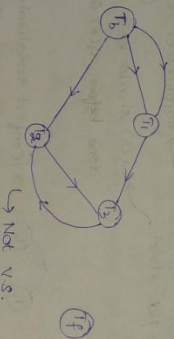
T_1	T_2	T_3
$R(x)$		
$R(y)$		
$w(x)$	$R(y)$	
$w(x)$		$w(y)$
	$R(y)$	



Not conflict
Serialisable
 $\Rightarrow w_2(y)$ is the Blind
write

→ Now check if it is vs by drawing polygons

T_0	T_1	T_2	T_3	T_4
$\omega(x)$	$R(x)$ $R(y)$ $R(z)$	$R(y)$		
$\omega(y)$			$\omega(y)$	
$\omega(z)$				$R(x)$ $R(y)$



→ Not CS

→ Not VS

→ Recoverability

Not detectable

⇒ cascading

⇒ NOT Strict

(If)

19. EXAMPLES ON VERIFYING THE STATE RULES - 3

π_1	π_2
$R(x)$	
	$u(x)$
$u(x)$	
	Abot
Commit	

Roll Back →

$\Rightarrow$ when a Transaction says About then
 that means i should Roll back
 $\Rightarrow$ when a transaction says About then
 there is only one transaction and
 here it will be CS
 $\Rightarrow$ Recoverable (No Commit on 12)
 $\Rightarrow$ Cascadeless (Independent)
 $\Rightarrow$ Not Strict $w_2(x)$ has not Read x

20. Example

T_1	
$ux(x)$	
$ux(x)$	
Comment	

π	$R(x)$	$w(x)$	Commit

T_1	$u(x)$	$(u(x))$	Comit
	R		Co

• Should act as before T_1 another
 • $T_2 \gg T_1$, edge
 will be preserved

154

conflict
serializable
is the Blind

T1	T2
w(x)	R(x)
w(x)	Abort
commit	

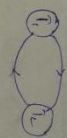
PC problem
→ Portback
→ Reading before
w(x) committed

CS ✓
VS ✓
S ✓

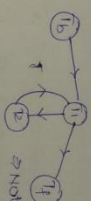
→ Producer-Consumer
→ Recoverable
(T₂ is aborting)
→ cascading schedule

155

Tb	T1	T2	Tf
R(x)			
w(x)		Commit	
commit			



⇒ Not CS
⇒ w(x) is the Blindwrite
⇒ Recoverable



⇒ Not VS, NOT STRICT

20 EXAMPLES ON VERIFYING SCHEDULES 4

T1	T2
w(x)	w(x)
	R(x)
	commit

Not CS,



⇒ Not CS
⇒ B_{all} present w₂(x)
⇒ Not VS.
⇒ Not Recoverable

Tb	T1	T2	Tf
w(x)			
	w(x)		
		R(x)	
		commit	



About then
on and

about then

reliability
& decidability
caching
Strict

should occur
before T₁
writes
→ edge
will be present

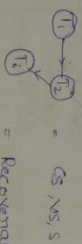
T_1	T_2
$w(x)$	
$w(x)$	$R(x)$
$w(x)$	commit
Abort	

$\Rightarrow$ when T_1 aborts there will be Roll back
 and there will be only one transaction left
 $\therefore$ The schedule is Conflict Serializable
 $\Rightarrow$ VS Recoverable - X
 $\Rightarrow$ SV $\therefore$ cascading - V (Not Cascadable)

$\Rightarrow$ Not strict

21. EXAMPLES ON VERIFYING SCHEDULE - 5

T_1	T_2	T_3
$R(x)$		
$w(y)$	$w(x)$	commit
commit		
$R(y)$	$w(z)$	
commit		

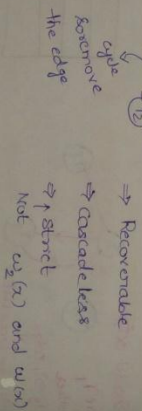


$\Rightarrow$ GS, NS, S
 $=$ Recoverable
 $=$ Cascadable
 $=$ Strict

T_1	T_2	T_3
$R(x)$		
$w(x)$	$w(x)$	
commit		
		$R(x)$
		commit



$\Rightarrow$ NOTES
 $\Rightarrow$ BWR
 $\Rightarrow$ New chaw poly graph
 $\Rightarrow$ Not VS



$\Rightarrow$ Recoverable
 $\Rightarrow$ Cascadable
 $\Rightarrow$ Strict
 Not $w_2(x)$ and $w(x)$

24. Prob

Loc - E
 $\Rightarrow$ This p
 mutual
 called

23. INT
 $\Rightarrow$ The
 that
 die

23. INTRODUCTION TO LOCKING

→ The concurrency manager present in the OS make sure that the transactions that are being executed are serializable, by using some rules they are

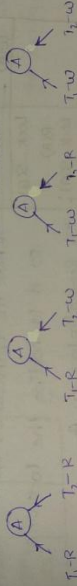
- Lock-based protocol
- Time stamp based protocols
- Graph based protocols.

Lock-Based Protocol

→ This protocol says that all the data items should be accessed in a mutually exclusive manner, to achieve this we use two locks called 1) shared lock (only Read) (Lock-S)

2) Exclusive lock (Both Read and write) (Lock-X)

→ If a transaction has shared lock on particular data item then another transaction may also acquire the shared lock on the same data item so share on share is allowed.



24. PROBLEMS WITH LOCKING

T1: A → B

Read(A)

A = A-50

Write(A)

Unlock(A)

Lock-X(B)

Read(B)

B = B+50

Write(B)

Unlock(B)

T2: Display(B+A)

Lock-SCB

Read(B)

Unlock(B)

Lock-S(A)

Read(A)

Unlock(A)

Display(B+A)

Now if it is only Reading then go for shared lock

→ Now if the Transaction are interleaved (cannot execution) then there occur some conflicts.

T ₁	T ₂
Lock(A) Read(A) A = A - 50 write(A) unlock(A)	Lock(SC(A)) Read(B) unlock(B) Lock(SC(A)) Read(A) unlock(A) Display(B+A)
Lock(X(B)) Read(C(B)) B = B + 50 write(B) unlock(B)	

$$\frac{A}{100} = \frac{B}{200} = \frac{A+B}{300} = 200$$

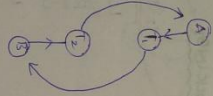
$$50 + 200 = 250$$

Because of concurrent execution we get 250.

⇒ Therefore Simple locking protocol doesn't give appropriate Result.

T ₁	T ₂
Lock-X(A) R(A) A = A - 50 write(A)	Lock-SC(B) R(B) Lock-SC(A) R(A) Display(B+A) commit unlock(B) unlock(A)
Lock-X(B) R(B) Lock-SC(B) B = B + 50 write(B) commit unlock(B) unlock(A)	

→ Here the transaction has not unlocked A
 so it is holding the lock on A
 → T₂ Requesting lock on A



There is cycle, one transaction waits for the other to unlock
 ∴ DEADLOCK

∴ Simple locking might produce Deadlock
 Simple locking might not guarantee Serializability

25. Two PHASE

since the 2-phase lock: Growing phase, Shrinking phase

⇒ Growing phase
 ⇒ Shrinking phase

⇒ The point is

⇒ with the help

T ₁	T ₂
L(X(A)) Locking point unlock(A)	L(X(B)) Locking point unlock(B)
L(X(B)) Locking point unlock(B)	L(X(A)) Locking point unlock(A)

⇒ The 2-phase overcomes serial

26. Strict 2PL

Some modification cascading 2PL
 Strict 2PL
 Which requires made locks commits. (u)

Lx-
 lock
 com
 unlock

25. TWO PHASE LOCKING PROTOCOL

Since the simple transaction has some disadvantages we come for 2-phase locking protocol. There are 2 phases in 2PL they are Growing phase, Shrinking phase

⇒ Growing phase - Obtain the locks } No one can come in middle
 ⇒ Shrinking phase - Release all the locks. } and solve serializability in 2PL

⇒ The point where the transaction has the final lock is called locking point.

⇒ with the help of locking points we can find serial schedule

T ₁	T ₂	T ₃
	L.S(A)	
		L.S(A)
	L.X(B)	
	U(A)	
		L.X(C)
	U(B)	
L.S(B)		
		U(A)
		U(C)
L.X(A)		
U(A)		
U(B)		

locking point for T₁

locking point for T₂

locking point for T₃ (point where the final lock appears)

∴ Serial schedule = $\boxed{T_2 T_3 T_1}$ = order in which we get locking points

Sometimes

⇒ The 2-phase locking has [cascading Rollbacks and Deadlocks] but it assures serializability.

26. STRICT RIGOROUS AND CONSERVATIVE 2PL

Some modifications are made for the above 2PL so that it can prevent cascading Roll backs

Strict 2PL

Which Requires that in addition to locking being 2PL all exclusive mode locks taken by a transaction must be held until the transaction commits. (unlock only after a particular transaction commits).

LX-(A)

W(A)

commit

U(x) → unlock only after transaction commits.

Strict 2PL are

cascadeless.

Recoverable,

Deadlock possible

Rigorous 2PL:

which requires that in addition to locking being 2-phase all locks must be held until the transaction commits.

⇒ can Avoid cascading Roll backs.

⇒ Deadlock cannot be avoided (Deadlocks are possible because of hold and wait)

Conservative 2PL

⇒ which Requires the transaction to update all the locks before it starts and release all the locks after it commits.

⇒ Avoids cascading Rollbacks and Deadlocks.

27. Examples ON 2PL

Which of the following schedules (Transactions) are in Strict 2PL?

a) Lock-SCA

R(A)

Lock-X(B)

R(B)

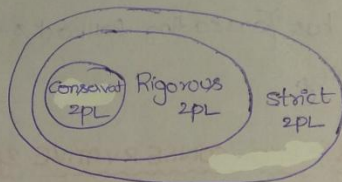
unlock(A)

w(B)

unlock(B)

⇒ There is Growing phase and shrinking phase
therefore it is 2PL

⇒ Now Consider the exclusive locks, Lock-X(B)
this should be unlocked only after 'B' commits
But it is unlocked before 'B' committed ∴ NOT
STRICT 2PL



b) Lock-SCA

R(A)

Lock-X(B)

unlock(A)

R(B)

w(B)

commit

unlock(B)

⇒ There is Growth and shrink ⇒ 2PL ✓

⇒ Consider Exclusive locks lock X-(B) it

should be unlocked only after 'B' commits

unlock done before commit ∴ **Strict 2PL**

⇒ Rigorous says that either it is shared lock or exclusive lock they should be unlocked only after commit, here shared lock is released before commit

∴ **NOT RIGOROUS 2PL**

c) Lock-SCA
R(A)
Lock-X(B)
R(B)
w(B)
commit
unlock(A)
unlock(B)

d) Lock-SCA
Lock-X(B)
R(B)
w(B)
R(A)
commit
unlock(A)
unlock(B)

e) Lock-SCA
R(A)
unlock(A)
Lock-X(B)
R(B)
w(B)
unlock(B)
unlock(A)
Commit

⇒ In the C

⇒ In the

all locks

increase of hold

before it

strict 2PL?

phase

lock - X(B)

B' commits

ted $\therefore$ NOT

STRICT 2PL

2PL ✓

- (B) it

B' commits

it is shared

should be

shared lock

c)

lock - S(A)

R(A)

lock - X(B)

R(B)

w(B)

commit

unlock(A)

unlock(B)

 $\Rightarrow$ Growth and shrink $\Rightarrow$ 2PL

141

 $\Rightarrow$ unlock (B) is done after commit $\therefore$ Strict 2PL $\Rightarrow$ unlock (A) is done after commit $\therefore$ Rigorous 2PL $\Rightarrow$ Conservative 2PL says that you have to acquire the locks before starting any operation $\therefore$ The given schedule started R(A) before acquiring the lock on B $\therefore$ NOT conservative 2PL

d) lock - S(A)

lock - X(B)

R(B)

w(B)

R(A)

commit

unlock(A)

unlock(B)

 $\Rightarrow$ Growth and shrink $\Rightarrow$ 2PL $\Rightarrow$ unlock (B) done after commit = Strict 2PL $\Rightarrow$ Both A, B are unlocked after commit = Rigorous 2PL $\Rightarrow$ All the locks are acquired before proceeding the operations $\Rightarrow$ Conservative 2PL

e) lock - S(A)

R(A)

unlock(A)

lock - X(B)

R(B)

w(B)

unlock(B)

unlock(A)

Commit

 $\Rightarrow$ 2PL X $\Rightarrow$ not strict 2PL $\therefore$ unlock(B) done before commit

NOT 2PL

NOT 2PL

 $\Rightarrow$ In the Growing phase locks are acquired and they can be ^{up}graded $\Rightarrow$ shared lock $\rightarrow$ upgraded to Exclusive lock $\Rightarrow$ In the Shrinking phase locks are Released and locks can be degraded $\rightarrow$ Exclusive lock $\rightarrow$ degraded to shared lock

28. GRAPH BASED PROTOCOL

⇒ One of the alternative to 2PL is Graph Based Protocol

⇒ we can avoid Deadlocks

Rules

1) In tree protocol the only lock instruction allowed is lock-x on each transaction T_i can lock a data item almost once and must observe the following rules.

- i) The 1st lock by T_i may be on any data item.
- ii) Subsequently a data item can be locked by T_i only if the parent of the data item is currently locked by T_i .
- iii) Data items may be unlocked at any time.
- iv) A Data item that has been locked and unlocked by T_i cannot subsequently be locked by T_i .

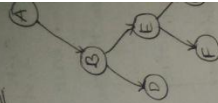
⇒ Now consider the Transactions
one-by-one T_1 and then T_2 and then T_3

T_1	T_2	T_3
lock-x(B)		
	lock-x(D)	
	lock-x(H)	
	unlock(D)	
lock-x(E)		
lock-x(D)		
unlock(B)		
unlock(E)		
		lock-x(B)
		lock-x(E)
	unlock(H)	
lock-x(G)		
unlock(G)		

Now if you want to see serializability order, replace all the locks with write statements



Ex



∴ The given scheduled

Advantages

- ⇒ Early un
- ⇒ Ensures
- ⇒ Deadlock

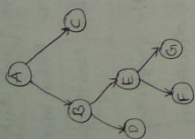
Drawbacks

- ⇒ you shoo
- ⇒ Unrecess

29. TIME S

- ⇒ Time shoo
- ⇒ Concurrency
- ⇒ It Requ
- ⇒ in advan
- ⇒ Each Tran
- ⇒ If any +
- ⇒ the Time

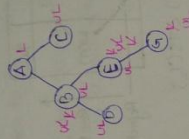
Ex



(143)

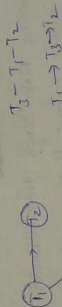
check whether this schedule is acc to that Graph or not?

T ₁	T ₂	T ₃
L(A)		
L(B)		
L(D)		
U(B)		
L(C)	L(B)	
U(D)		L(E)
		L(G)
		U(E)
		U(G)
U(C)		



∴ The given schedule is according to the Graph given and serial

schedule is



Advantages

- ⇒ Early unlocks causes increase in concurrency
- ⇒ Ensures conflict serializability
- ⇒ Deadlock-free protocol

Drawbacks

- ⇒ you should know what is the data item to pick first
- ⇒ unnecessary locking overheads

29. TIME STAMP ORDERING PROTOCOL

- ⇒ Time stamp protocol is used by the concurrency manager to provide concurrency and sees that the resulting schedules are always serializable
- ⇒ It requires that ordering among the transaction is determined in advance based on their time stamps.
- ⇒ Each transaction is given unique fixed time stamp denoted by $TS(T_i)$
- ⇒ If any transaction T_j that enters after T_i the relation between their time stamps be $TS(T_1) < TS(T_2)$ which means that the producing schedule must be equivalent to a serial schedule $T_1 \rightarrow T_2$

31. EXAMPLS

check which o

TI	R(S)	R(A)	Display(A)	A = u	Dis
----	------	------	------------	-------	-----

T_1	$R(A)$	2	T_2	T_1

T_1	T_2
$\omega(x)$	$\omega(x)$
	$R(x)$

↑
5
the
ab

$$T_2 \quad T_3 \quad T_4$$

1-2-3-4.

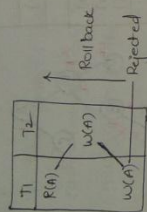
⇒ Hole $3_2(x)$

$\therefore$ This is

→ Now, Thom

here the con

3) In Time Stamp ordering protocol ensures that any conflicting read and write operations are executed in time stamp order if not such an operation is Rejected and the Transaction will be Rolled back. The rolled back transaction will be restarted with a New timestamp.



$\Rightarrow$ these $R_i(A) \cup_2(A)$ is the conflicting action which means T_1 should occur first and then T_2 .

Rejected

$w(n)$

$TS(n) < TS(n_2)$

$\therefore$ Conflict serialisable to T_1-T_2

$\Rightarrow$ Again $w_2(n)$ and $w_1(n)$ is the conflicting action but η should occur first then T_2 so Rejected the action. Rejected.

$\therefore$ Conflict + Ser

30. THOMAS WRITE RULE

Thomas write Rule says that obsolete writers can be ignored
obsolete write means if the writers are late and before that
some other write has happened which is supposed to happen after it
then you can clearly ignore the write and make the transaction
stay there before rolling back.

	Not Allowed	Allowed
$R_1(n)$	$\omega_2^2(n)$	$R_2^2(n)$
$\omega_2^2(n)$	$R_1^2(n)$	$R_1(n) \cdot R_2^2(n)$
$\omega_2(n)$	$\omega_2^2(n)$	$\omega_2^2(n) \cdot R_1(n)$
		$\omega_2^2(n) \cdot \omega_2(n)$
	$R_1(n)$	$R_1(n) \cdot R_2^2(n)$
	$\omega_2^2(n)$	$\omega_2(n) \cdot \omega_2^2(n)$

Top Group ← Top
 Bottom Group ← Bottom
 Ordering protocols
 Top rule
 Bottom rule

Reject or Rollback

Time Stamp ← Top
ordering
protocol

fibrous

If $T_2 \rightarrow T_1$ then $R_2(A)$ must be present

order and there is $R(A)$ is present.

31. EXAMPLES ON TIME STAMP ORDERING PROTOCOL

check which of the following schedules can appear under TOP.

- ⇒ Here T_1 is starting first $\Rightarrow TS(T_1) < TS(T_2)$
- ⇒ There are 2 conflicts $\Rightarrow T_1$ should come first acc to Time stamps.
- ⇒ In both the conflicts T_1 has occurred first before T_2 $\therefore$ The schedule is according to TOP

T1	T2
R(B)	R(B)
	0-B-50
	w(B)
R(A)	R(A)
Display(A)	
	A=A+50
	w(A)
	Display(A+5)

⇒ $TS(T_1) < TS(T_2)$

⇒ Not according to TOP

T1	T2
R(A)	
	w(A)
w(A)	

S1: The above schedule is possible under TOP

S2: The above schedule is possible under Thomas Write Rule

which of the above statements is true about the schedule given?

T1	T2	T3	T4
w(x)			
	w(x)		
		w(x)	
	R(x)		w(x)

⇒ In the above schedule T_1 arrived first and then followed by T_2, T_3, T_4 . $\therefore$ The order in which we should serialise them is 1-2-3-4.

⇒ Here $w_1(x) R_2(x) \Rightarrow$ '3' should occur first and then '2' (Conflict).

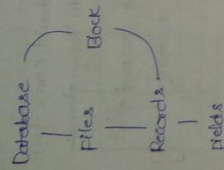
acc to 1-2-3-4

This is not possible acc to TOP

⇒ Now, Thomas write rule saves us in write-write conflict but here the conflict is (w-r) so Not acc to TOP

8. FILE STRUCTURES

1. FILE ORGANISATION



Database is collection of files
 Each file is a collection of Records
 Each Record is a sequence of field.

Blocking Factor : It is the average no. of records per block

Strategies for storing file of records into block

① Spanned strategy : It allows, partial part of Record can be stored in a block

Adv : No wastage of memory, suitable for variable length record.
Disadv : Block Access increases.

Unspanned strategy

No Record can be stored in more than 1 block

Disadv : wastage of memory

Adv : Block Access reduced, suitable for fixed length records

Organisation of Records in a file

① Ordered file Organisation

All files of Records are ordered based on some search key value

Searching : Binary search, B-databases to access a record

$$\text{Avg no of Block access} = \lceil \log_2(B) \rceil \times \text{blocks}$$

Adv : Searching is efficient

Dis : Insertion is expensive

146

Unordered file
 All the file usually at the

Searching :

Avg. no. of b

Adv : Simple

Disadv : Search

2. INDEXING

Databble

1	10	20
B1	50	70
B2	80	90
B3	100	200

B4		
----	--	--

→ Avg No of Block

⇒ For every index file the

⇒ For every index file

3. STRUCTURE

Index

⇒ Indexes are used

⇒ Index is a grid

⇒ Index is an

searching : D

Unordered file Organisation

All the file records are inserted at where ever the place is available usually at the end of the file

Searching: Linear search

$$\text{Avg. no. of block access} = \frac{B}{2}$$

Adv: Insertion is easy

Disadv: Searching is inefficient.

2. INDEXING

Datatable

1	10	20	50	70	80	90	100	200
B1	B1	B1	B2	B2	B2	B3	B3	B3

Index file

1	B1
10	B1
20	B1
50	B2
70	B2
80	B2
90	B3
100	B3

B1
B1
B1
B2
B2
B2
B3
B3
B3

→ Avg no of Block Access $(\log_2 B)$
(Binary search)

B1
B1
B1
B2
B2
B2
B3
B3
B3

For searching in index file
Avg. Block Access $= (\log_2 B) + 1$
B_i = size of index file

⇒ For every record in the datatable if we have a record/entry in the index file then that index is called Dense index

⇒ For every block in the datatable if we have a entry in the index file then that index is called Sparse index.

3. STRUCTURE OF INDEX FILES

Index:

⇒ Indexes are used to improve the search efficiency

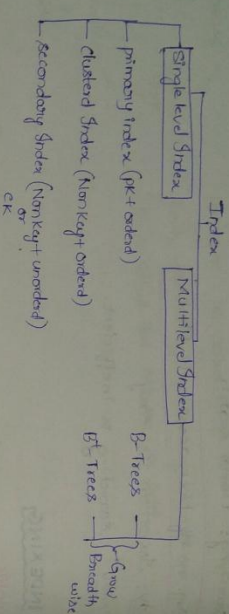
⇒ Index is a record that consists of two fields. Key Block pointer

⇒ Index is an ordered file

⇒ Searching: Binary search

⇒ To access a record using the index, the avg no. block access = $(\log_2 B_1 + 1)$

4. TYPES OF INDEXING



5. DENSE AND SPARSE INDEXING

Dense Index: If an index entry is created for every search key value then that index is called dense index.

Sparse Index: If an index entry is created only for some search key values then it is called sparse index.

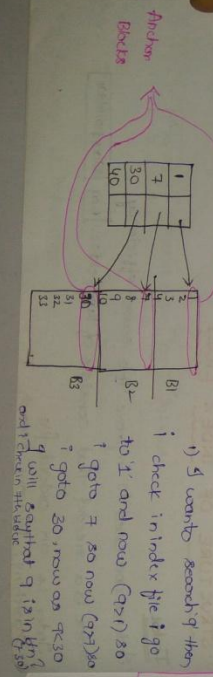
Primary Index (PK + optional)

A primary index is an ordered file whose records are of fixed length with two fields the first field is same as primary key of datafile and the second field is a pointer to data block where the key is available.

⇒ Index entry is created for 1st record of each block called "block anchor".

⇒ No. of Index entries = No. of entries in datablocks.

⇒ Avg no. of block access = $\log_2 (B_1 + 1)$ (B_1 = Index blocks)



6. EXAMPLE

Suppose that we have a block size 10 and a primary index created using a pointer of size 4.

sol. Records = 30

Block size = 10

Record size = 10

key + pointer = 10

⇒ Data records = 30

⇒ Total no. of blocks = 3

⇒ Now, without index

⇒ Size of index = 30

⇒ No. of blocks = 3

⇒ Total No. of blocks = 6

28 + 1)

(148)

6. Example on Primary Indexing

Suppose that we have an ordered file of 30,000 records on a disk. with block size 1024 bytes. File records are of fixed length and are unspanned of size 100 bytes and suppose that we have created a primary index on the key field of the file of size 9 bytes and a block pointer of size 6 bytes. Then find the avg no of blocks to search for a record using with and without index?

Sol. Records = 30,000

Block size = 1024 Bytes.

Record size = 100 Bytes

Key + pointer = 6 + 9 = 15 Bytes

⇒ Data records that can fit in 1 block = $\frac{\text{Block Size}}{\text{Record Size}} = \frac{1024}{100} = 10.24$

= 10

∴ 10 records can be placed in one block

⇒ Total no of Blocks =

1 Block — 10 Records

x Blocks — 30,000 Records

$$\Rightarrow \frac{x}{10} = \frac{30,000}{10} \Rightarrow x = 3000 \text{ Blocks}$$

⇒ Now without indexing no of block access = $\lceil \log_2(3000) \rceil = 12$ Block Access

⇒ Size of index Record = 15 Bytes

No of index Records per block = $\left\lfloor \frac{1024}{15} \right\rfloor = 68$ index Records.

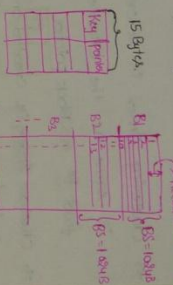
$$\Rightarrow \text{No. of Block Access} = \lceil \log_2(45) + 1 \rceil = \lceil \log_2(2^{5.5}) + 1 \rceil = \lceil 5.5 + 1 \rceil = 7$$

Total no of index Record = No of Data Blocks × 3000

1 Block (index file) → 68 Records

→ 3000 Records

$$\Rightarrow \frac{3000}{68} = 44.1176 \approx 45$$



1. CLUSTERED INDEXING (NK + ordered)

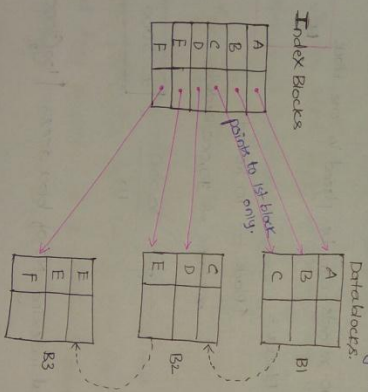
ISO

⇒ clustering index is a combination of both dense and sparse indexes.

⇒ clustering index is used when the files are sorted acc to a Non-Key value. (Not primary, Not candidate)

⇒ clustering index is a ordered file (with two fields, the first field is same as the clustering field is called non-key and the 2nd field is a block pointer)

⇒ Clustered index is created on database whose records are physically ordered on a non-key field which doesn't have a distinct value for each record that field is called clustering field.



Searching: Binary Search

Avg. no. of block reads = $\log_2(B_i) + 1$

⇒ Index entry is created for each distinct value of a clustering field

⇒ The block pointers point to first block in which the key is available

⇒ Type of index is Dense / sparse

2. SECOND

⇒ Index file

⇒ Record does

for which

⇒ Secondary

⇒ Index or

⇒ No. of ind

⇒ Type of

Index file

Avg. file access

= $\log_2(B_i) + 1$

⇒ Binary Search

Consider a

previous or

a record us

⇒ R = 30,000

⇒ No. of dat

⇒ No. of blo

⇒ size of file

⇒ No. of Reco

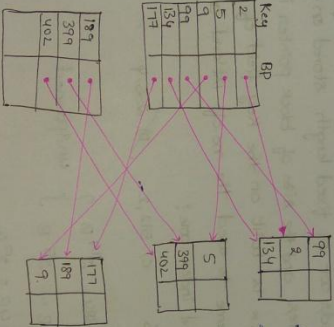
⇒ No. of Inde

1 block

B. SECONDARY INDEXING (Nk / Lk + unordered)

- ⇒ Index file should always be an ordered file
- ⇒ Secondary Index provides a secondary means of accessing file for which some primary access already exists.
- ⇒ Secondary Index maybe a key or non key
- ⇒ Index entry is created for every record of data file
- ⇒ No. of index entries = No. of records.
- ⇒ Type of Secondary Index is dense.

Index file
Avg. blk access
 $= \log_2(R) + 1$
⇒ Binary Search



Consider a Secondary index is built on the key field of the file of previous question. the find max. avg. no. of block access to access a record using with and without index

- ⇒ $R = 30,000$ (unsorted)
- ⇒ No. of data blocks = 300 blocks
- ⇒ No. of block access required without indexing = $R/2 = 1500$
- ⇒ Size of Index record = 15 bytes.
- ⇒ No. of Records we can place in 1 block = $\left\lfloor \frac{1024}{15} \right\rfloor = 68$
- ⇒ No. of Index Records = No. of Data Records
= 30,000
- 1 block $\rightarrow$ 68 Records
- 2 — 30,000 Records

$$\therefore BA = \left\lceil \log_2 \left(\frac{30000}{68} \right) \right\rceil + 1$$

$$BA = 10$$

9. EXAMPLE ON MULTILEVEL INDEXING

112

As single level index is an ordered file we can create a primary index itself. In this case the original file is called 1st level index and the index to index is called 2nd level index

⇒ If there are n levels in multilevel index then no of block are to search for a record = $n+1$ (one blk at each level and 1 data blk finally)

Consider a file of 16384 records, each record is of size 32-bytes and key field is of size 6 bytes and the file organization is unspanned and records are of fixed length stored on a disk with block size 1024 bytes and the size of block pointer is 10 bytes. If the secondary index is built on the key field of the file and multilevel index scheme is used the no of 1st level and 2nd level blocks in multilevel index are?

- A) 8, 10 B) 128, 6 C) 512, 2 D) 256, 4

$$\text{Data records} = 16384 = 2^{14}$$

$$\text{Record size} = 32 \text{ B} = 2^5 \text{ B} \quad \left. \vphantom{\text{Record size}} \right\} \text{unspanned}$$

$$\text{Block size} = 1024 \text{ B} = 2^{10} \text{ B}$$

$$\text{Key field} = 6 \text{ B}, \text{ bp} = 10 \text{ B}$$

$$\text{Index Record size} = \text{Key} + \text{Block pointer} = 16 \text{ B}$$

⇒ 1st level Index

$$\text{No of Index Records} = \frac{\text{No of Data Records}}{\text{No of Index Records per block}} = \frac{2^{14}}{2^{10}} = 2^4 = 16$$

$$\therefore \text{No of blocks in Index} = \frac{2^{14}}{2^6} = 2^8 = 256$$

⇒ 2nd level Index



$$\text{No of blocks} = \frac{256}{2^6} = 4$$

10. INTRODUCTION TO BTREES AND B⁺ TREES

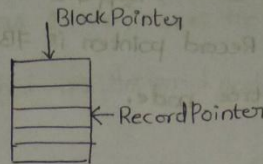
⇒ BTree and B⁺Trees are Dynamic

⇒ Node pointer and Block pointer

⇒ Record pointer

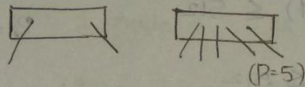
⇒ All the databases today use B⁺ Trees.

⇒ The advantage of B⁺ Trees is they grow vertically

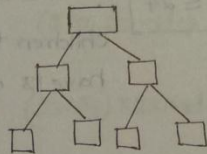


11. PROPERTIES OF BTREES

1) Root :- Between 2 and 'p' children where (p = Order of the tree)
(Order is the max amount of children that can present to a particular node.)



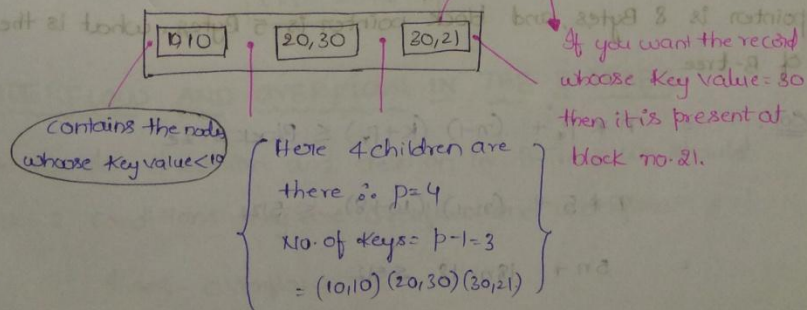
2) Internal nodes:



⇒ Internal nodes can store upto (p-1) keys.

⇒ Have between $\lceil p/2 \rceil$ and p children

The general Representation is $\langle p, \langle k_1, p_1 \rangle, p_2, \langle k_2, p_2 \rangle, \dots, p_{q-1}, \langle k_{q-1}, p_{q-1} \rangle \rangle$



3) Leaf node :- stores between $\lceil (p-1)/2 \rceil$ and p-1 keys

- All nodes have same depth.
↓
leaf

12. EXAMPLE ON BTREE

In a Btree, suppose search key is 9 Bytes long, disk block size is 512B, Record pointer is 7B, Block pointer is 6B, then calculate the order of B-tree node.

⇒ Every node in Btree is a disk block

$$P(K, P_r) \quad P(K, P_r) \quad \dots \quad P$$

⇒ The formula that every node should satisfy is

$$n * p + (n-1)(k + p_r) \leq \text{Block size}$$

n = order of the tree

p = Block pointer size

P_r = Record pointer

$$= n * 6 + (n-1)(9 + 7) \leq 512$$

$$= 6n + 16n - 16 \leq 512$$

$$= 22n \leq 528 \Rightarrow n \leq 24$$

∴ The max amount of children that this tree can have is 24.

13. EXAMPLE ON BTREE

Consider a B-tree with key size 10 Bytes, Block size 512B, data pointer is 8 Bytes, and block pointer is 5 Bytes. what is the order of B-tree

Sol: $n * p + (n-1)(k + P_r) \leq \text{Block size}$

$$= n * 5 + (n-1)(10 + 8) \leq 512$$

$$= 5n + 18n - 18 \leq 512$$

$$= 23n \leq 540$$

$$= n \leq \lfloor 23.478 \rfloor$$

$$= n = 23$$

14. EXA

Suppose

will be

$$\Rightarrow n = 23$$

L4

At level 1

At level 2

At level 3

At level 4

15. UNDER

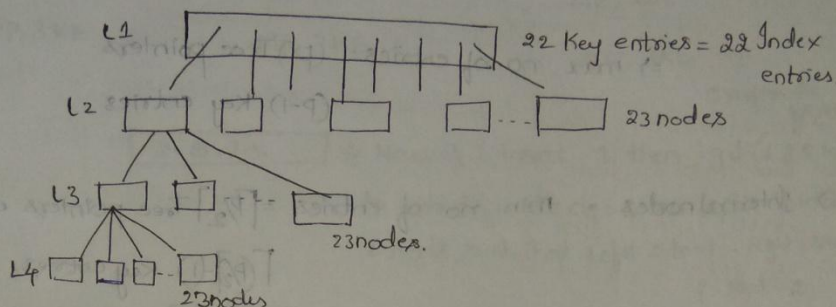
⇒ while w

check :

14. EXAMPLE IN B-TREE-3

Suppose that the order of Btree is 23. Then how many index records will be stored in 4 level (including root as 1 level) across the B-Tree.

$\Rightarrow n = 23$ (max amount of children that an internal node have)



At level 1 : 1 node 22 Index Records 23 disk/node pointers

At level 2 : (23) nodes 23×22 Index Records 23×23 node ptrs.

At level 3 : $(23) \times 23$ nodes $(23 \times 23) \times 22$ IR $(23 \times 23) \times 23$ NP

At level 4 : $(23 \times 23) \times 23$ nodes $(23 \times 23 \times 23) \times 22$ IR $\times$ (leaf nodes)

$$\text{No. of IR} = 23 \times 23 \times 23 \times 22$$

$$\text{IRs} = 267674$$

15. UNDERFLOW AND OVERFLOW IN THE B-TREES

$\Rightarrow$ while we do Insertion and deletion in B-Trees we should

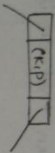
check 2 conditions they are overflow and underflow

Insert - overflow

Delete - Underflow.

Order of B-Tree = p then,

i) Root node $\Rightarrow$ min no of entries = 2 Tree pointers
1 key entry.



$\Rightarrow$ max no of entries = (p) Tree pointers
 $(p-1)$ Key entries

ii) Internal nodes = min no of entries = $\lceil \frac{p}{2} \rceil$ Tree pointers and
 $\lceil \frac{p}{2} \rceil - 1$ Key entries

= max no of entries = p Tree pointers
 $(p-1)$ Key entries

iii) Leaf nodes = min no of key entries = $\lceil \frac{p}{2} \rceil - 1$ Key entries
max no of key entries = $(p-1)$.

$p = 5$

i) Root = min = $2TP$ max = $5TP$
1KE 4KE

ii) Internal nodes = min = $\lceil \frac{p}{2} \rceil TP = \lceil \frac{5}{2} \rceil = 3TP$ max = $5TP$
2KE 4KE

iii) Leaf nodes = min = $\lceil \frac{p}{2} \rceil - 1$ max = 4
= 2KE

16. INSE

Insert

Keys: 2

Min: 2

Max: 4

$\therefore$ Find

16. INSERTION INTO B-TREES - Example - 1

Insert the following keys in B-tree of order-4.

Keys: 2, 5, 10, 1, 6, 9, 4, 3, 12, 18, 20, 25.

Root

Min: 2TP, 1KE

Intermediate nodes

12TP, 1KE

Leaves

2TP, 1KE

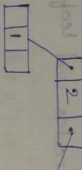
Max: 4TP, 3KE

4TP, 3KE

6TP, 3KE

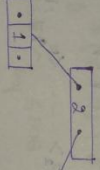
[2, 5, 10]

⇒ Now, if i insert 1 then get (1, 2, 5, 10) in which middle ele = 2, so split it into 2 nodes such that left side = 1, Right = 2



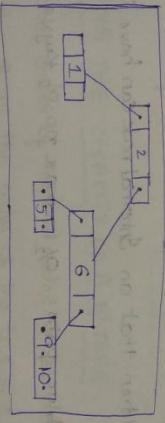
[5, 10]

⇒ Now inserting '6' (6 > 2) so move in RST



[5, 6, 10]

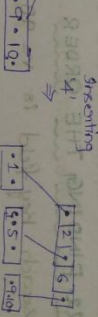
⇒ If 9 is inserted it becomes (5, 6, 9, 10) overflow. Split at mid point = 6.



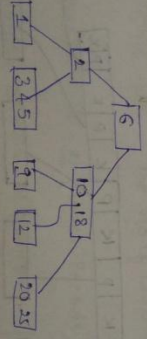
X



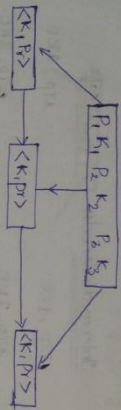
Inserting



∴ Final tree =



21. B⁺-TREES STRUCTURE OF LEAF AND INTERNAL NODES



⇒ The information needed to access the data is present in the leaves
 ⇒ The intermediate nodes don't contain the pointers that point to data instead they guide to reach the leaves for which it points to the Required data block.

⇒ The structure of the internal nodes and leaf nodes are different

Internal node : $\langle P_1, K_1, P_2, K_2, \dots, P_{q-1}, K_{q-1} \rangle$

leaf node : $\langle \langle K_1, P_1 \rangle, \langle K_2, P_2 \rangle, \dots, \langle K_{q-1}, P_{q-1} \rangle, P_{next} \rangle$

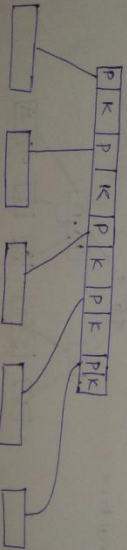
Let, Q_{leaf} be the order of leaf node

⇒ The min-no of children that an internal node can have is $\lceil \frac{Q_{leaf}}{2} \rceil$ to Q_{leaf}

⇒ min-no of key value pairs = $\frac{Q_{leaf}}{2}$, max-no of key value pairs = Q_{leaf}

22. FINDING THE ORDER OF B⁺ TREES

Search key field is $V = 9B$, Bk size = $512B$, Record ptr = $P_2 = 4B$, Bk ptr = $P = 6B$. what is the order of internal node and leaf node?



leaf node

Let the c

⇒

⇒

⇒

⇒

23. FIND

In a B⁺

node is

many size

0 and 1

⇒ then

⇒ leaf node

⇒ At Root

⇒ At leaf

$$\Rightarrow \alpha(p) + (0-1)k \leq \text{Block size}$$

O = order

$$= 0(6) + (0-1)9 \leq 512$$

P = Block position

$$6(0) + 9(6) - 9 \leq 512$$

K = key

$$= 15(6) - 9 \leq 512 \Rightarrow 15(6) \leq 521$$

$$\Rightarrow 0 \leq [34, 3] \Rightarrow [0, 34]$$

Order = 34

leaf node: $\boxed{[k, p] \mid [k, p] \mid [k, p] \mid \dots \mid [k, p] \mid \text{next}} \rightarrow$

Let the order of leaf node = $Q_{\text{leaf}} = (\text{max no of } \boxed{\text{key ptr}} \text{ pairs we can have in the leaf node})$

$$\Rightarrow Q_{\text{leaf}} * (k + p) + p \leq \text{Block size}$$

$$\Rightarrow Q_{\text{leaf}} * (7 + 9) + 6 \leq 512$$

$$\Rightarrow 16 * Q_{\text{leaf}} \leq 506$$

$$\Rightarrow Q_{\text{leaf}} \leq 31.2 \Rightarrow \boxed{Q_{\text{leaf}} = 31} \therefore \text{Order of leaf node} = 31$$

8. FINDING THE CAPACITY OF A B+ TREE

In a B+ tree, order of internal nodes is 34 and order of the leaf node is 31. If all the nodes of the tree are 69% full, then how many records will be present in 4-level B+ tree with root at level 0 and leaf level 3.

$$\Rightarrow \text{Given node} = 69\% \text{ full} = \text{The no of children} = \frac{69}{100} \times 34 = 23.46 \approx 23 \text{ keys}$$

$$\Rightarrow \text{leaf node} = 0.69 \times 31 = 21.39 \approx 21 \text{ (Index Records) (key, data ptr pairs)}$$

$$\Rightarrow \text{At Root} \quad 1 \text{ node} \times 23 \text{ KE} = 23 \text{ pointers} \quad \mid \quad \text{At L2: } 23 \text{ nodes} \times (31 \times 21) \text{ KE} = 11558 \text{ KE}$$

$$\Rightarrow \text{At level 4} \quad 23 \times 23 = 529 \text{ KE} \quad \mid \quad 23 \times 23 = 529 \text{ ptrs} \quad \mid \quad \text{No of ptrs} = 529 \times 21 = 11109$$

13: 12,167 nodes

12,167 x 81 = 985,507 (index records that file has)

is able to store)

(160)

Imp points

1) Secondary index is built on non-ordering field. Hence if of the exists in attributes then the no. of secondary indices is $2^n - 1$

2) clustered index = Non-key + ordered $\Rightarrow$ clustering index is used when we have clusters of similar database entries for particular attribute.

3) Indexing is done by ordering all the instances of attributes and clustering the same values, so index entry for only the first item is needed only.

This attribute has Repeated values so it is non-key.

FD = $X \rightarrow Y$

4. GATE 2000 QUESTION ON FD's

Given the following Relation Instance

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

which of the following functional dependencies are satisfied by the Instance?

(a) $XY \rightarrow Z$ and $Z \rightarrow Y$ (b) $YZ \rightarrow X$ and $Y \rightarrow Z$
 (c) $YZ \rightarrow X$ and $X \rightarrow Z$ (d) $XZ \rightarrow Y$ and $Y \rightarrow X$

a) $XY \rightarrow Z$ and $Z \rightarrow Y$
 ↓
 Holds

Not hold {3→5}
 {3→6}

b) $YZ \rightarrow X$ and $Y \rightarrow Z$
 ↓ ↓
 Holds Holds

c) $YZ \rightarrow X$ and $X \rightarrow Z$
 Holds

not hold {1→2}
 {1→3}

d) $XZ \rightarrow Y$ and $Y \rightarrow X$
 ↓ ↓
 Not hold Holds
 {1,3→5}
 {1,3→6}

5. GATE 2002 QUESTION ON FD's

From the following Instance of a relation schema R(A,B,C), we can conclude that

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

a) A functionally determines B and B functionally determines C
 $A \rightarrow B$ and $B \rightarrow C$

b) $A \rightarrow B$ and $B \nrightarrow C$

c) $B \nrightarrow C$

d) $A \nrightarrow B$ and $B \nrightarrow C$

a) $A \rightarrow B$ $B \rightarrow C$
 ↓ ↓
 holds Not holds

(b) $A \rightarrow B$ $B \nrightarrow C$
 ↓ ↓
 holds holds

(c) $B \nrightarrow C$
 holds

(d) $A \nrightarrow B$ $B \nrightarrow C$
 ↓ ↓
 Not holds holds

for this Instance but fail when insertion → NOT HOLDS.

B X
 C X
 A X
 B X
 A } Holds but
 C } not sure
 → X ✓
 → Z X
 → Y X
 → Z ✓
 → Y X

6. FORMAL DEFINITION OF FUNCTIONAL DEPENDENCY

	A	B
t_1		
t_2		
	If t_1 and t_2 then they must agree here	
	If t_1 and t_2 then they may disagree here	

	A	B	
Agree	1	a	Agree
	1	a	
Disagree	2	b	Agree or may not Agree.
	3	b	

7. VARIOUS USAGES OF FD'S

- ⇒ Identify Additional Functional Dependencies
- ⇒ Identifying Keys
- ⇒ Identifying Equivalences of FD's
- ⇒ finding minimal FD set.

In order to perform all the above activities we have 2 methods

- 1) Inference Rules ⇒ Error prone
- 2) Closure set of Attributes ⇒ Easy and less Error prone

Inference Rules

- a) Reflexive : $A \rightarrow B$ if $B \subseteq A$
- b) Transitive : $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$
- c) Decomposition : $A \rightarrow BC$ then $A \rightarrow B$ $A \rightarrow C$
- d) Augmentation : $A \rightarrow B$ then $AC \rightarrow BC$
- e) Union : $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$
- d) Composition : $A \rightarrow B$ and $C \rightarrow D$, then $AC \rightarrow BD$.

8. CLOSURE

functional

Here, A^+

B^+

(C^+)

9. GATE

The follo

$AB \rightarrow CD$

$AF \rightarrow D$

$DE \rightarrow F$

$C \rightarrow G$

$F \rightarrow E$

$G \rightarrow A$

Sol) a) {C

b) {

c) {

d)

10. DE

Given

The no

⇒ For

(24)

8. CLOSURE SET OF ATTRIBUTESfunctional Dependencies: $A \rightarrow B$ $B \rightarrow D$ $C \rightarrow DE$ $CD \rightarrow AB$

The closure set of a set 'A' is

denoted by A^+ and closure set says

that what are the other attributes

that you can uniquely identify

using A

Here, $A^+ = \{A, B, D\}$ $B^+ = \{B, D\}$ $C^+ = \{D, E, C, A, B\}$ $D^+ = \{D\}$ $E^+ = \{E\}$ $(CD)^+ = \{C, D, E, A, B\}$ $(AD)^+ = \{A, D, B\}$ $(ABD)^+ = \{A, B, D\}$ 9. GATE 2006 QUESTION ON CLOSURE SET OF ATTRIBUTES

The following functional dependencies are given:

 $AB \rightarrow CD$

which one of the following options is false?

 $AF \rightarrow D$ $DE \rightarrow F$ $C \rightarrow G$ $F \rightarrow E$ $G \rightarrow A$ A) $(F)^+ = \{A, C, D, E, F, G\}$ $(AF)^+ = \{A, C, D, E, F, G\}$ B) $(BG)^+ = \{A, B, C, D, G\}$ C) $(AB)^+ = \{A, B, C, D, G\}$ So A) $\{CF\}^+ = \{C, F, G, A, E, D\} = \{A, C, D, E, F, G\} = \text{TRUE}$ B) $\{BG\}^+ = \{B, G, A, C, D\} = \{A, B, C, D, G\} = \text{TRUE}$ C) $\{AF\}^+ = \{A, F, E, D\} = \text{FALSE}$ D) $\{AB\}^+ = \{A, B, C, D, G\} = \text{TRUE}$ 10. DETERMINING CANDIDATE KEYSGiven Relation $R(A, B, C, D)$ and the FD's are $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A$ The no. of candidate Keys that can be formed = $2^n - 1$ (n = no. of attributes in the Relation) $\therefore \text{No. of CK's} = 2^4 - 1$ $\Rightarrow$ For the above example No. of candidate Keys that should be examined is $2^4 - 1 = 15$, at maximum.

Now, $R(ABCD)$ the CK of length 4 = $(ABCD) = ①$

(26)

CK of length 3 = $(BCD)(ABC)(ACD)(ABD) = ④$

CK of length 2 = $(AB)(BC)(CD)(DA)(AC)(BD) = ⑥$

CK of length 1 = $(A)(B)(C)(D) = ④$

(15)

Now, perform Bottom to Top Approach

Now, let's check if 'A' is candidate key $\Rightarrow A^+ = \{A, B, C, D\}$

$\therefore$ A is candidate key

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

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

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

$\Rightarrow \{A, B, C, D\}$ are candidate keys

$\therefore$ No. of CK's = 4

Note: If 'A' is a candidate key then anything that contains A like $(AB)(CA)(AD)(ABC)(ABD)$ will not be CK's they will be SK's.

11. GATE 1999 QUESTION ON CANDIDATE KEYS

GATE - 99

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

functional dependencies are $(C \rightarrow F, E \rightarrow A, EC \rightarrow D, A \rightarrow B)$ then CK = ?

A) CD B) EC C) AE D) AC

Sol

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

$\{A, B, D, F\}$ are derivable from the RHS but there is no way to determine C, E so the only way to derive them is by having $(CE)^+ = \{C, E\}$
 $\therefore$ 'CE'

$\therefore (CE)^+ = (C, E, F, A, D, B)$

CK = EC

or check by option

12. GATE 20

consider the of functional $\{K\} \rightarrow \{M\}$,

① $\{E, F\}$

30) Given F

Now,

Now, (E

13. GATE

Consider a functional candidate

① AE, BE

30) The f

12. GATE 2014 QUESTION ON CANDIDATE KEYS

Consider the Relation Scheme $R = (E, F, G, H, I, J, K, L, M, N)$ and the set of functional dependencies $\{EF \rightarrow G, F \rightarrow I, J, EH \rightarrow K, L \rightarrow M, K \rightarrow M, L \rightarrow N\}$ on R . What is the key for R ?

- Ⓐ $\{E, F\}$ Ⓑ $\{E, F, H\}$ Ⓒ $\{E, F, H, K, L\}$ Ⓓ $\{E\}$

sol) Given FDs =

$\{E, F\} \rightarrow \{G\}$	$\Rightarrow (EF) \rightarrow (G)$
$\{F\} \rightarrow \{I, J\}$	$\Rightarrow (F) \rightarrow (IJ)$
$\{E, H\} \rightarrow \{K, L\} \rightarrow \{M\}$	$\Rightarrow (EH) \rightarrow (KL) \rightarrow (M)$
$\{K\} \rightarrow \{M\}$	$\Rightarrow (K) \rightarrow (M)$
$\{L\} \rightarrow \{N\}$	$\Rightarrow (L) \rightarrow (N)$

Now, $(EFH)^+ = (E, F, G, I, J, K, L, M, N)$ The attributes that can be derived from EFH are (G, I, J, K, L, M, N) and there is no way I could get $\underline{EFH}$.

Now, $(EFH)^+ = \{E, F, H, G, I, J, K, L, M, N\}$

$\therefore \{EFH\}$ is the candidate key and $\{E, F, H, K, L\}$ is SK.

13. GATE 05 QUESTION ON CANDIDATE KEY

Consider a Relation Scheme $R = (A, B, C, D, E, H)$ on which the following functional dependencies hold: $\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$. What are the candidate keys of R ?

- Ⓐ AE, BE Ⓑ AE, BE, DE Ⓒ AEH, BEH, DEH Ⓓ AEH, BEH, DEH

sol) The functional dependencies =

$A \rightarrow B$	$\therefore$ candidate keys should contain EH .
$BC \rightarrow D$	
$E \rightarrow C$	
$D \rightarrow A$	

$\Rightarrow (EH)^+ = (A, B, C, D, E, H)$ Now, $(EH)^+ = (E, H, C)$

Now, $(EH)^+ = \{E, H, c\}$

$(A EH)^+ = \{A, B, C, D, E, H\} \rightarrow 8 \text{ SKS}$ [Total 6 attributes already known]

$(BEH)^+ = \{A, B, C, D, E, H\} \rightarrow 8 \text{ SKS}$ [Total 6 attributes already known]

$(CEH)^+ = \{C, E, H\}$

$(DEH)^+ = \{A, B, C, D, E, H\} \rightarrow 8 \text{ SKS}$

14. GATE 2013 QUESTION ON CANDIDATE KEY

Relation R has 8 attributes ABCDEFGH. Fields of R contain only atomic values. $F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow H, F \rightarrow EG\}$ is a set of functional dependencies so that F^+ is exactly the set of FDs that hold for R. How many C's does R have?

$CH \rightarrow G$

$A \rightarrow BC$

$B \rightarrow CFH$

$E \rightarrow H$

$F \rightarrow EG$

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

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

$\Rightarrow (AD)^+ = \{A, B, C, D, E, F, G, H\} \checkmark$

$\Rightarrow (BD)^+ = \{A, B, C, D, E, F, G, H\} \checkmark$

$\Rightarrow (CD)^+ = \{C, D\} \times$

$\Rightarrow (ED)^+ = \{A, B, C, D, E, F, G, H\} \checkmark$

$\therefore$ No. of C's possible are = 4's = $(AD)(BD)(ED)(FD)$

15. Examples on Candidate Keys-1

1) $R = (ABCDE)$

$FD = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$

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

Now, $(AB)^+ = \{A, B, C, D, E\}$ $\therefore$ "AB" is the candidate key $\Rightarrow 2 \times 3 \times$

2) $R(ABCE)$

$\Rightarrow (C)$

Now (B)

3) $R(ABC)$

$\Rightarrow (C)$

$\Rightarrow (A)$

4) $R = AB$

$\Rightarrow (C)$

Now (A)

$\times (C)$

16. EXAM

$R = (ABCE)$

$FD = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$

Now, A

B

C

D

Let's already say
Total keys = {A, B, C, D, E, F}
EH are CKs

only atomic
set of
D's that had

(28)

② R(ABCDE) FD = {A → B, C → D, B → E}

⇒ (B)⁺ = {A, B, C, D, E}

Now (B)⁺ = {B, E, A, C, D} ∴ B is the candidate key

∴ No of superkeys = 2⁴ = 16

③ R(BCDEF) FD = {A → B, C → D, E → F}

⇒ (ACE)⁺ = {A, B, C, D, E, F}

⇒ (ACE)⁺ = {A, C, E, B, D, F} ⇒ ACE is the CK and 2³ = 8 SK are possible

④ R = ABCD, FD = {A → C, D → A}

⇒ (B)⁺ = {A, B, C, D} ⇒ Now (B)⁺ = {B}

Now (AB)⁺ = {A, B, C, D} ✓

(CB)⁺ = {C, B} ×

(DB)⁺ = {D, B, A, C} ✓

(AB) and (DB) are candidate keys

16. EXAMPLES ON CANDIDATE KEY - 2

R = (ABCD)

FD = {A → B, C → D, D → B}

Now, A⁺ = {A}

Two attribute keys = (AB)⁺ = (ABCD) ✓

B⁺ = {B}

(AD)⁺ = (ABCD) ✓

C⁺ = {C, D}

(BC)⁺ = (BCD) ✓

D⁺ = {D}

(BD)⁺ = (BD) ✓

(CD)⁺ = (CD) ✓

AB, AD, BC, CD are CKs.

CKs are: AB, AD, BC, CD.

EXAMPLES ON CANDIDATE KEYS - 3

$$R = (\overline{A} \overline{B} \overline{C} \overline{D} \overline{E} \overline{F})$$

$$FD = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow A\}$$

$$\text{Now, } B^+ = \{B\}$$

$$\Rightarrow (AB)^+ = (ABCDEF)$$

$$\Rightarrow (CB)^+ = (ABCDEF)$$

$$\Rightarrow (DB)^+ = (ABCDEF)$$

$$\Rightarrow (EB)^+ = (ABCDEF)$$

$$\Rightarrow (FB)^+ = (ABCDEF)$$

$$\therefore CKs = (AB)(CB)(DB)(EB)(FB)$$

$$= 5CKs$$

18. CANDIDATE KEY- Example - 4

$$R = (\overline{A} \overline{B} \overline{C} \overline{D} \overline{E} \overline{F})$$

$$FD = \{AB \rightarrow C, C \rightarrow D, D \rightarrow EB, E \rightarrow F, F \rightarrow A\}$$

Now,

$$A^+ = A$$

$$B^+ = B$$

$$C^+ = (CDEBFA) \checkmark$$

$$D^+ = (DEBFAC) \checkmark$$

$$E^+ = EFA$$

$$F^+ = FA$$

Now,

$$(AB)^+ = (ABCDEF)$$

$$(AE)^+ = (AEF)$$

$$(AF)^+ = (AF)$$

$$(EF)^+ = EFA$$

$$(BE)^+ = (BEFACD) \checkmark$$

$$(BF)^+ = (BFACDE) \checkmark$$

Now,

$$(\underline{BEF})^+ = (BEF) \times$$

$$(\underline{AEF})^+ = (AEF) \times$$

$$(\underline{ABF})^+ = (ABF) \times$$

$$(\underline{ABE})^+ = (ABE) \times$$

Now,

$$(\underline{ABEF})^+ = (ABEF) \times$$

20. ExAM

$$R = (\overline{A} \overline{B} \overline{C} \overline{D} \overline{E})$$

$$FD = \{A \rightarrow B\}$$

Now,

$$(E)^+ = \{E\}$$

21. ExAM

$$R(\overline{A} \overline{B} \overline{C} \overline{D} \overline{E})$$

$$A \rightarrow BC$$

$$CD \rightarrow E$$

$$B \rightarrow D$$

$$E \rightarrow A$$

$\therefore C$ and D are candidate keys

AB and BE, BF are candidate keys.

(30)

19. EXAMPLES ON CANDIDATE KEYS - 5

R(ABCDEF)

 $A \rightarrow BCDEF$ $BC \rightarrow ADEF$ $DEF \rightarrow ABC$

Now,

 $A^+ = (ABCDEF) \checkmark$ $B^+ = (B) \times$ $C^+ = X$ $D^+ = X$ $E^+ = X$ $F^+ = X$

Now,

 $(BC)^+ = ABCDEF$

Now,

 $(DEF)^+ = (DEFABC)$ $\therefore$ Candidate keys = A, BC, DEF

(31)

20. EXAMPLE ON CANDIDATE KEY - 6

R(ABCDE)

Now,

 $(ABEF) \times$ FD = $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Now,

 $(E)^+ = \{E\}$

Now,

 $(AE)^+ = (AEB CD) \checkmark$ $(BE)^+ = (BEC DA) \checkmark$ $(CE)^+ = (CED AB) \checkmark$ $(DE)^+ = (DEABC) \checkmark$ $\therefore$ The CK's are: AE, BE, CE, DE

No. of CK's = 4

21. EXAMPLE ON CANDIDATE KEY - 7

R(ABCDE)

Now,

 $A \rightarrow BC$ $CD \rightarrow E$ $B \rightarrow D$ $E \rightarrow A$ $A^+ = (ABCDE) \checkmark$ $B^+ = BD$ $C^+ = C$ $D^+ = D$ $E^+ = (EABCD) \checkmark$ $(BC)^+ = (ABCDE) \checkmark$ $(BD)^+ = (BD)$ $(CD)^+ = (ABCDE) \checkmark$ $\therefore$ CK's are A, E, BC, CD

22. CANDIDATE KEY FOR SUB RELATION - Example - 1

$R(ABCD)$

$\{AB \rightarrow CD, D \rightarrow A\}$ what are the candidate keys of subrelation $R_1(BCD)$?

Now, Given Relation $R_1(BCD)$

$$\begin{aligned} B^+ &= B & (BC)^+ &= BC \\ C^+ &= C & (BD)^+ &= \overline{CDAB} \\ D^+ &= DA & (CD)^+ &= \overline{CDA} \end{aligned} \quad \therefore \text{candidate key} = \underline{BD}$$

23. CANDIDATE KEY FOR SUB RELATION Example 2

$R(ABCDEFG)$

FD's: $\{AB \rightarrow C, B \rightarrow D, AD \rightarrow F, C \rightarrow D, D \rightarrow E, E \rightarrow F, E \rightarrow D\}$

Find candidate keys for $R_1(DEF)$.

Given Sub Relation $R_1(DEF)$

$$D^+ = \{\overline{DEF}\} \checkmark$$

$$E^+ = \{\overline{EDF}\} \checkmark$$

$$F^+ = \{F\}$$

$\therefore$ The candidate keys are $\underline{DE}$

24. CANDIDATE KEYS FOR SUB RELATION - Example 3

$R(ABCDE)$

FD's $\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

what are the candidate keys of $R_1(ABCE)$

Given sub Relation $R_1(ABCE)$

$$A^+ = \overline{ABCDE}$$

$$B^+ = BD$$

$$C^+ = C$$

$$E^+ = \overline{EABCD}$$

Now,

$$BC = \overline{BCDEA} \checkmark$$

$$BD \times (\text{D not intable})$$

$$CD \times (\text{D not intable})$$

$\therefore$ candidate keys are,

$$\underline{A/E, BC}$$

25. CHECKING

$\Rightarrow$ Given the set

functional Dep

Inference Rule

$R(AB)$ and the

$$\underline{X} \rightarrow \underline{Y}$$

$$\phi \rightarrow \phi$$

$$A \rightarrow A \Rightarrow$$

$$B \rightarrow B$$

$$AB \rightarrow AB$$

The no. of Add

$$\checkmark \phi \rightarrow \phi \quad \checkmark$$

$$\times \phi \rightarrow A \quad \checkmark$$

$$\times \phi \rightarrow B \quad \checkmark$$

$$\times \phi \rightarrow AB \quad \checkmark$$

The no. of FDs

26. CHECKING

$R(ABC)$

FD's: $\{A \rightarrow B, B \rightarrow$

$$\underline{X} \rightarrow \underline{Y}$$

$$\downarrow \quad \downarrow$$

$$2^3 \times 2^3 = 8 \times 8$$

$$\phi \rightarrow \phi \text{ (only)}$$

$$A \rightarrow \{ABC\} =$$

$$B \rightarrow \{BC\} =$$

$$C \rightarrow \{C\} =$$

$$AB \rightarrow (AB)^+ = \{A$$

$$(B)^+$$

25. CHECKING ADDITIONAL FD's - Example 1

⇒ Given the semantics of the Database we can derive more functional Dependencies from the existing FD's by applying Inference Rules.

$R(AB)$ and the functional dependencies are $\{A \rightarrow B, B \rightarrow A\}$

$\underline{X} \rightarrow \underline{Y}$

$\phi \quad \phi$

A A

B B

AB AB

⇒ No. of functional dependencies are $4 \times 4 = 16$

The no. of additional FD's possible are (along with given FD's $A \rightarrow B, B \rightarrow A$)

$\checkmark \phi \rightarrow \phi \quad \checkmark A \rightarrow \phi \quad \checkmark B \rightarrow \phi \quad \checkmark AB \rightarrow \phi$

$\times \phi \rightarrow A \quad \checkmark A \rightarrow A \quad \checkmark B \rightarrow A \quad \checkmark AB \rightarrow A$

$\times \phi \rightarrow B \quad \checkmark A \rightarrow B \quad \checkmark B \rightarrow B \quad \checkmark AB \rightarrow B$

$\times \phi \rightarrow AB \quad \checkmark A \rightarrow AB \quad \checkmark B \rightarrow AB \quad \checkmark AB \rightarrow AB$

$B^+ = \{BA\}$: possible

$A^+ = \{AB\}$: possible.

The no. of FD's possible on the table are $(\checkmark) 13$

26. CHECKING ADDITIONAL FD's - Example 2

$R(ABC)$

FD's: $\{A \rightarrow B, B \rightarrow C\}$

$\underline{X} \rightarrow \underline{Y}$

$\downarrow \quad \downarrow$

$2^3 \times 2^3 = 8 \times 8 = 64$ FD's

$\phi \rightarrow \phi$ (only one FD possible with ϕ) = 1 FD

$A \rightarrow \{ABC\}$ = 8 FD's are possible with LHS = 'A' because $A^+ = \{ABC\} = 2^3$

$B \rightarrow \{BC\}$ = 4 FD's are possible with LHS = 'B' because $B^+ = \{BC\} = 2^2$

$C \rightarrow \{C\}$ = 2 FD are possible with LHS = 'C' because $C^+ = \{C\} = 2^1 = 2$
($C \rightarrow \phi$, $C \rightarrow C$)

$AB \rightarrow (AB)^+ = \{ABC\} = 8$ FD

$AC \rightarrow (AC)^+ = \{ABC\} = 8$ FD's

$(BC)^+ = (BC)^+ = \{BC\} = 4$ FD

$(ABC)^+ = \{ABC\} = 8$ FD's

⇒ Total valid FD's = $1 + 8 + 4 + 2 + 8 + 4 + 8 = 43$

∴ In General the no. of functional dependencies that are possible over a Relation R containing 'n' attributes is 2^n

$$X \rightarrow Y \\ 2^n \quad 2^n \Rightarrow 2^n \times 2^n$$

Now find G_1

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

G_1 :

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

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

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

27. GATE IT 05 QUESTION ON ADDITIONAL FD'S

In a schema with attributes A, B, C, D and E, following set of FD's are given $A \rightarrow B, A \rightarrow C, CD \rightarrow E, B \rightarrow D, E \rightarrow A$ which of the following FD's is NOT implied by above set?

- a) $CD \rightarrow AC$ b) $BD \rightarrow CD$
c) $BC \rightarrow CD$ d) $AC \rightarrow BC$

Now, $(CD)^+ = \{C, D, E, A, B, D, E\} \Rightarrow (CD)^+ = (AC) \checkmark$ possible

$$(BD)^+ = \{B, D\} = \text{NOT POSSIBLE}$$

$$(BC)^+ = \{B, C, D, E, A\} = (CD) \text{ is possible}$$

$$(AC)^+ = \{A, C, B, D, E\} = (BC) \text{ is possible}$$

28. EQUIVALENCE OF FD'S

$$F: \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$$

$G_1: \{A \rightarrow CD, E \rightarrow AH\}$ are both the functional dependencies Equivalents

Sol First check if 'F' covers 'G₁' i.e. $F \supseteq G_1$ how to check is take

each FD in the set 'G₁' and find whether it is derivable from 'F' or not

And now, check $G \supseteq F$, take each FD of 'F' and check, if it is already implied in 'G' or not.

$$\text{Now, } F \supseteq G_1 \quad \text{Now, } G_1: \{A \rightarrow CD, E \rightarrow AH\}$$

$$F:$$

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

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

covered by 'F'

covered by 'F'

$F \supseteq G_1$ is TRUE

$$G_1 \supseteq F \Rightarrow \text{TRUE}$$

In G_1 :