1. ER-MODEL

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: Super markets, Firms.

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

⇒ Now a days the Databases are computerised 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 programers. (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 = (26, Raju, --) = Extension. ②

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

4. ATTRIBUTES

⇒ Attributes are useful inorder to describe the entities Better.

The Attributes are manily 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, Middle name, Last Name (Composite Attribute)

Age = Single valued Attribute

PNO = Multi valued Attribute

DOB = Stored Attribute

Age = Derived Attribute ← Derives.

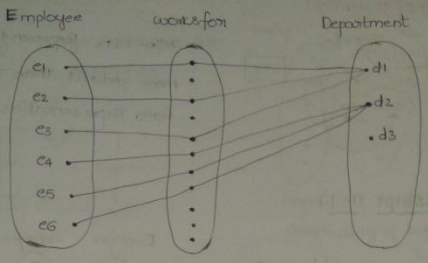
Address = Complex Attribute — Composite Attribute / Multi valued Attribute.

5. RELATIONSHIPS (1-M)

⇒ Relationship is nothing but Association among entities.

Requirement Analysis : 1. Every employee works for a Dep and a Dep can have many Employees (exactly)

2. New department need not have any Employee.



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 Existance : Min. no. of Relationships that an entity can participate

$\boxed{\text{participation} = \text{Min. cardinality}}$

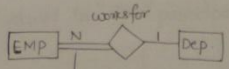
⇒ participation (employee) = 1

⇒ participation (Department) = 0. (Newly formed department cannot have employees).

cardinality Ratio
participation/Existance } Structural constraints.

Entities: ☐

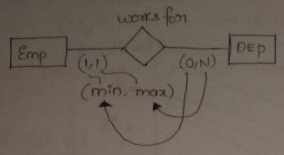
Relationships: ◇



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

CARDINALITY RATIO REPRESENTATION.

MIN MAX REPRESENTATION

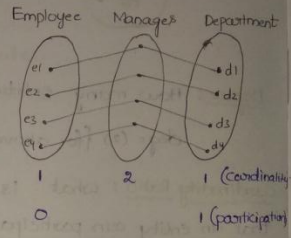


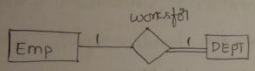
⇒ Min- Max Representation gives more details than Cardinality Ratio Representation.

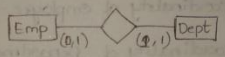
6. RELATIONSHIPS (1-1)

RA:

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

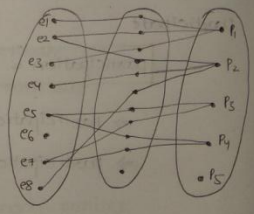






7. RELATIONSHIPS M-M EXAMPLES

RA: Every employee is supposed to work atleast on one project and he can work on many projects aswellas every project is supposed to have many employees and is supposed to have atleast one employee.



P1: e_1 e_2 e_3

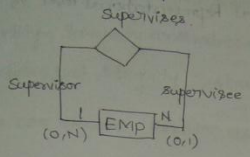
P2: e_2 e_4 e_8

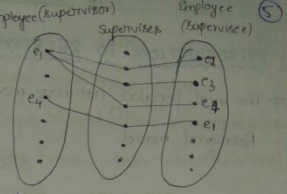
P3: e_5 e_7

P4: e_1 e_5 e_6

8. RECURSIVE RELATIONSHIPS

RA: Every employee is supposed to have exactly one supervisor





Degree = 2

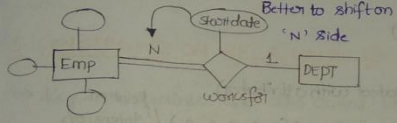
Cardinality of supervisor = N

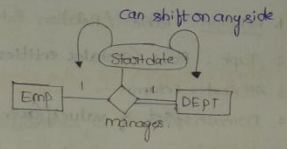
Supervisee = 1

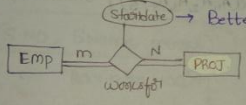
participation of supervisor = 0

of supervisee = 0

9. ATTRIBUTES TO RELATIONSHIPS







→ Better not to shift on any side.

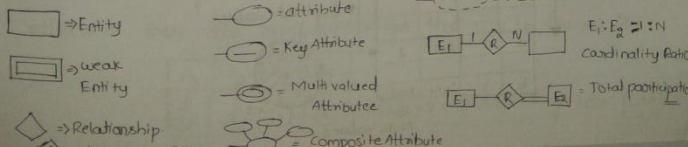
10. WEAK ENTITY

⇒ whenever 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".

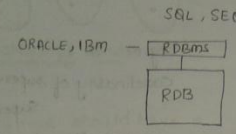
11. ER- DIAGRAM NOTATIONS



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 Intension
6. Degree of a Relation : The no of Attributes

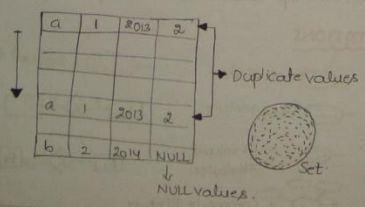
3. TUPLE, TUPLE VALUES AND NULL

⇒ whenever we store a Relation in a memory then it is 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 firstname, Middle Name, Lastname and now a person might not have middle name so Middlename == NULL



4. CONSTRAINTS ON RELATIONAL DB SCHEMA - DOMAIN CONSTRAINTS

1. Domain constraints ⇒ Entire schema should be Atomic.
2. Key constraints ⇒ No two Tuples should have same value
3. Entity- Integrity constraints ⇒ Entire tuple should follow some constraints.
4. Referential Integrity constraints ⇒ Applied between two tables (Relations)

⇒ We can view a Relation as "flat file structure".

✗ S.NO	Name ✗		
✓ S.NO	FN	MN	LN

Should be Atomic

Composite, Multivalued attributes are not allowed in Relations.

5. CONSTRAINTS ON RELATIONAL DB SCHEMA - KEY CONSTRAINTS

⇒ Key-constraints are also called "Uniqueness Constraints".

(S.NO, SName, marks) ⇒ SUPER KEY

(1, Ravindra, 100)

(1, Ravindra, 100)

	Student		Marks
t_1			100
t_2			100

$t_1 \neq t_2$

⇒ $\boxed{\text{SUPER KEY} \subseteq \text{Attributes}}$ for which no two tuples have the same values in all the Attributes.

⇒ Any Minimal superkey is a "Key". (SNO) $\boxed{\text{KEY} = \text{MINIMAL SUPERKEY}}$

⇒ Every Relation is going to have a superkey by default and that superkey is "set of all Attributes".

⇒ Any superset of a Key is a superkey.

⇒ If A1 is a Key, and the set/Relation/table contains $(A_1 A_2 A_3 A_4)$ attributes then the No. of superkeys that can be formed is

A_1	A_2	A_3	A_4
↓ 1	↓ 2	↓ 2	↓ 2

$= 1 \times 2 \times 2 = 8$ 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." (8)

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

$(\overset{\times}{A_1}\ A_2\ A_3\ A_4)$ - SK

$(\overset{\times}{A_2}\ A_3\ A_4)$ - SK

$(A_3\ A_4)$ - SK and Key

$(A_1\ A_2\ \overset{\times}{A_3}\ A_4)$

$(A_1\ A_2\ A_4)$ - SK and Key

$\therefore$ Keys = $(\underbrace{A_3 A_4\ ,\ A_1 A_2 A_4}_{\text{Mimal super Keys}})$ ⇒ candidate Keys.

⇒ "primary key" doesnot allow "NULL" values.

6. ENTITY AND REFERENTIAL INTEGRITY CONSTRAINS

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

Employee

ENO	ENa	DNO	PNO
1	a	1	4

DNO = FOREIGN KEY

Department

DNO	DNAME	
1		
2		
3		

PNO		
1		
2		
3		

"DNO" USUALLY WILL BE PRIMARY KEY IN DEPARTMENT RELATION.

⇒ foreign key can have "NULL" values unlike primary key.

7. ACTIONS UPON CONSTRAINT VIOLATIONS

⇒ The Actions that are performed on the database are

i) Insertion
ii) Deletion
iii) Update

} on performing these actions we should be sure that the constrains are not violated (Domain, key, Entity, Referential constrains).

⇒ If the above actions violate the constrains 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 referred by the above tuple & they should be deleted).
3. Set NULL or some other value

8. COUNTING THE NO OF SK'S POSSIBLE - EXAMPLE - 1

⇒ Given a Relation $R(A_1\ A_2\ A_3 \cdots A_n)$

Candidate key = $\{A_1\}$

Then the no of possible super keys are

$$\underbrace{A_1 \quad A_2 \quad A_3 \quad A_4 \cdots A_n}_{(n-1)\text{ elements}}$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 2 \quad 2 \quad 2 \quad 2$$

A1	A2	
1	a	
2	a	
3	b	
4	b	
5	c	

∴ Total no of keys = $1 \times 2 \times 2 \times 2 \times \cdots (n-1)$ times

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

$$\boxed{\text{Total no of SK's} = 2^{n-1}}$$

9. COUNTING THE NO. OF SK'S POSSIBLE - EXAMPLE 2

Relation $R = (A_1\ A_2\ A_3 \cdots A_n)$

Candidate Key $= \{A_1 A_2\}$ Candidate key is $A_1 A_2$ not A_1, A_2

A_1 → Single ck, A_2 → Two ck's

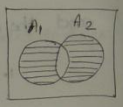
$$\underset{1}{A_1}\ \underset{1}{A_2}\ \underbrace{A_3\ A_4 \cdots A_{n-2}}_{(n-2)\text{ele}}$$

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

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

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

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



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

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

$$\boxed{SK = 2^n - 2^{n-2}}$$

10. COUNTING THE NO. OF SK'S POSSIBLE - EXAMPLE 3

Relation $R = (A_1\ A_2\ A_3 \cdots A_n)$

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

No. of SK's $= SK(A_1) + SK(A_2 A_3) - SK(A_1 A_2 A_3)$

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

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

No. of SK's $= SK(A_1 A_2) + SK(A_3 A_4) - SK(A_1 A_2 A_3 A_4)$

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

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

$$\boxed{SK\text{'s} = 2^{n-2} + 2^{n-2} - 2^{n-3}}$$

11. COUNTING THE NO. OF SK's POSSIBLE - EXAMPLE - 4

Relation $R = (A_1\ A_2\ A_3 \cdots A_n)$

$CK = (A_1, A_2, A_3)$

NO. of SK's $= SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1A_2) - SK(A_2A_3) - SK(A_1A_3)$
$+ SK(A_1A_2A_3)$

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

$\Rightarrow R = (A\ B\ C\ D)$
$CK = (A, BC)$

NO. of SK's $= SK(A) + SK(BC) - SK(ABC)$

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

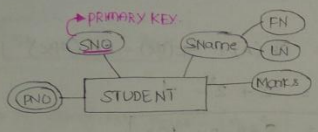
$$\boxed{SK = 8 + 4 - 2 = 10}$$

3. CONVERTION OF ER MODEL TO RELATIONAL MODEL

1. STEP 1

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

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



Now the Relation for this ER-diagram will be

Student

SNO	Marks	FN	LN

⇒ Every simple attribute is represented in the table.

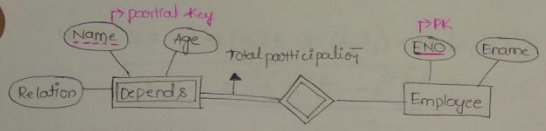
⇒ composite attribute is further divided and atomic parts are represented in the table

⇒ Multivalued 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.

Depends Relation

ENO	Name	Age	Relation

↳ pk of weak entity.

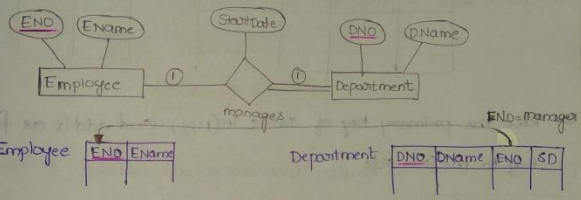
Employee Relation.

ENO	EName

⇒ 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.

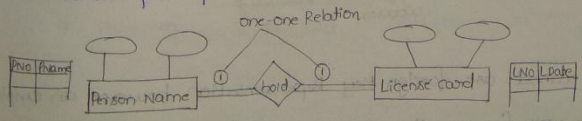
3 STEP-3

⇒ CONVERTION OF RELATIONSHIPS INTO RELATIONS (TABLES)



⇒ Now Add the primary key of one side (one table) as the foriegn key of another table, generally add the pk of (non-total participation) side as the foriegn 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 In that case there is no need of adding

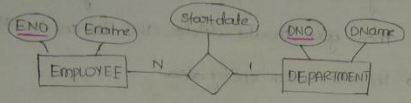
the primary key of one side onto the other side we can just combine the two relations into a single Relation.

PNO	PNAME	LNO	LDate

and it also takes less space.

4. STEP-4

⇒ CONVERSION OF 1:N RELATIONSHIP INTO A RELATION.



Employee

Department

ENO	Ename	DNO	Start Date

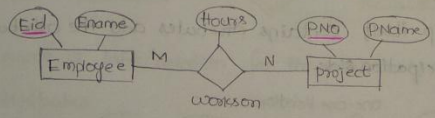
DNO	DName

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

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

5. Step-5

⇒ CONVERSION OF MANY TO MAY RELATIONSHIP



⇒ we cannot use foreign key Representation because an employee can work fore more than one project and a project can have more than one employee

⇒ Employee 1 works on 1, 2, 3 ✗

Eid	En	PNO
1		1,2,3

PNO	PNa

Eid	En

PNO	PN	Eid
1		1,2

Project 1 has more than one employee ✗

⇒ The solution to

attributes as

Eid

Eid + PNO=

⇒ Now the att

Newly formed

6. STEP-6

⇒ DEALING W

⇒ For the mult

⇒ Create a

key of the

PNO	SNO
P1	1
P2	1

∴ PK = SNO

⇒ The solution to the above problem is create a new table having attributes as the primary keys of the participating entities.

Eid	P.NO

Eid	E name

PNO	PName

(15)

Eid + P NO = PK of Newtable

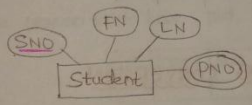
⇒ Now the attribute on the Relationship should be added to the Newly formed table.

Eid	PNO	Hours

6. STEP - 6

⇒ DEALING WITH MULTIVALUED ATTRIBUTES

⇒ For the multivalued attribute we are going to create a newtable



⇒ Create a Relation having the multivalued attribute and primary key of the entity (SNO, PNO)

SNO	PNO

⇒ SNO + PNO = PK for this table

PNO	SNO
P1	1
P2	1

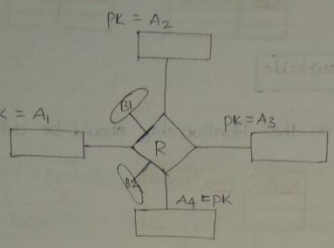
∴ PK = SNO + PNO

SNO	FN	LN
1	a	b

∴ PK = SNO

7. STEP-7

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



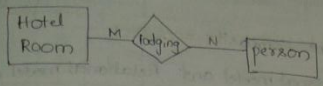
⇒ Now create a new table having the attributes as primary keys of all the entities.

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)
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 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 ✗

⇒ 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 it's own Rent, so we cannot put all the Rents here, ⇒ option C ✗

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

Pid	HNO	Rent
1	1	10,000
2	2	5,000

⇒ 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

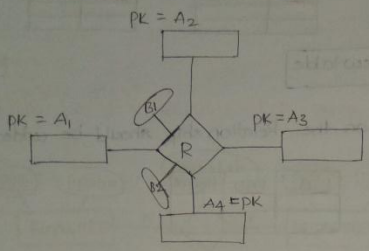
b) An attribute of an entity can be composite.

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 1 value or NULL

7. STEP-7

⇒ DEALING WITH N-ARY RELATIONSHIPS (MORE THAN 1 ENTITIES)



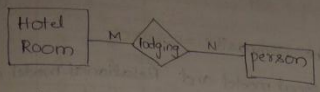
⇒ Now create a new table having the attributes as primary keys of all the entities.

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)
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 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 ✗

⇒ 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 C ✗

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

Pid	HNO	Rent
1	1	10,000
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 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 1 value or NULL

a) option a talks about ER-model and multivalued Attributes in ER-model. = CORRECT

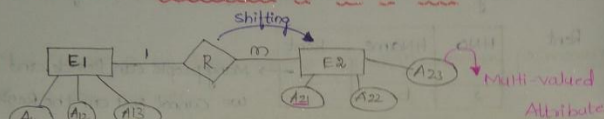
(18)

b) In ER-model Attributes can be composite. = CORRECT

c) option 'c' talks about Relational model and Relational model doesn't allow multivalued Attributes and Composite Attributes = INCORRECT

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

11. GATE IT 04 CONVERTION OF ER TO RDB



Min no of tables: ___

$\therefore$ $E_1(\underline{A_{11}}\ A_{12}\ A_{13})$

$E_2(\underline{A_{21}}\ R\ A_{22})$

$E_3(A_{23}, A_{11}, A_{21})$

} 3 Tables.

12. GATE 05 ON CASCADE DELETE IN CASE OF FOREIGN KEYS

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

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

Sol

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

14. GATE 08 QUES

$E_1 E_2$ - two entities

$R_1 R_2$ are two Rela

and R_2 is many t

own. what is the

Sol

PK = A_1 E

(1)

15. GATE 97 QU

Let R(a,b,c) and

that refers to t

'R' and 'S'.

a) Insert into R

which of the fo

a) None of a

b) All of a,b,c

c) Both a,b

d) Both b,c

Sol

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) (7,2) (9,5). (19)

⇒ (2,4) is deleted ⇒ delete the rows containing two(2)

⇒ Now on deleting we are deleting the rows (5,2) (7,2) because they contain '2'

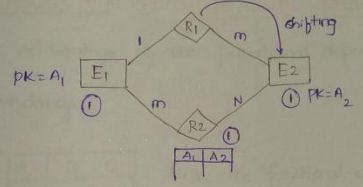
⇒ Now delete the rows containing (5)(7) = (9,5) deleted.

14. GATE 08 QUESTION ON CONVERTING ER TO RDB

E_1E_2 - two entities

R_1R_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 does not have any attributes of their own. What is the min no. of tables req used?

Sol



Min. no. of tables = 3

15. GATE 97 QUESTION ON REFERENTIAL INTEGRITY

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

a) Insert into R b) Insert into 'S' c) Delete from R d) Delete from 'S'.

Which of the following is true about the referential integrity constraint above?

a) None of a,b,c,d can cause its violation

b) All of a,b,c,d can cause violation

c) Both a,b can cause its violation

d) Both b,c can cause its violation.

Sol

R

a	b	c
1		
2		
3		
4		
5		
6		
10		

S

d	e	f
1		
1		
2		
2		
3		
7		
6		

V.V. Imp

⇒ 'd' is depending on 'a' not 'a' is depending on 'b'.

Violation (Inserting into 'S').

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

4. NORMALISATION

1. INTRODUCTION TO NORMALISATION

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

⇒ Less Redundancy

⇒ various Anamolies will occur
- Insert Anamolies
- Deletion Anamolies
- Update Anamolies.

⇒ splitting the tables into small tables such that our design will not contain all the above Anamolies and Redundancy is called "NORMALISATION".

⇒ In order to do Normalization we use the concept of Functional Dependencies (FD's). 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		
t_1 →	2	a	b
	3		
t_2 →	2	a	b

Here the functional dependency is $A \rightarrow BC$

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

$A \rightarrow BC$

$2 \rightarrow ab.$

(or) if $t_1(A) = t_2(A)$ then $t_1(BC) = t_2(BC)$ ⇒ if $(2=2)$ then $(ab)=(ab)$ ✓

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

⇒ Next 2nd NF

⇒ Next 3NF

⇒ Next BCNF

Now find $G_1 \supseteq F$ then,

$F: \{A \to C, AC \to D, E \to AD, E \to H\}$ check whether these are derivable from 'F'.

G_1:

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

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

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

$\therefore$ all the functional dependencies in F are covered by G_1

$\therefore$ Both the FD's 'F' and 'G' are Equivalent.

27. EQUIVALENCE OF FD's EXAMPLE-1

$F: \{A \to B, B \to C, C \to D\}$

$G_1: \{A \to BC, C \to D\}$

check whether these two FD's are Equalent or not?

Sol

$F \supseteq G_1 \Rightarrow$ Take Each FD of G_1 and check whether it is derivable from F:

$\Rightarrow A \to BC \Rightarrow$ Now Take A^+ from F and check BC is present in A^+

$A^+ = \{A, B, C, D\}$ $\therefore$ $A \to BC$ holds

$\Rightarrow C \to D \Rightarrow C^+ = \{D, C\} \to$ holds $\therefore F \supseteq G_1$

$G_1 \supseteq F \Rightarrow$ The FD's of 'F' are $A \to B, B \to C, C \to D$ check if they are covered by G_1 or not.

in G_1: $A^+ = \{A, B, C\}$ $\{A \to B$ holds$\}$

$B^+ = \{B\}$ $\{B \to C$ is not covered by $G_1\}$

$\therefore G_1 \not\supseteq F$

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

30. EQUIVALENCE OF TWO FD'S EXAMPLE - 2

(1)

$F: \{A \to B, AB \to C, D \to AC, D \to E\}$

$G: \{A \to BC, D \to AB\}$

(2)

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

$G: \{A \to BC, B \to A, C \to A\}$

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

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

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

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

$\Rightarrow D^+ = \{A, B, C, D\}$ $\quad D \to AC$ ✓ $\quad \therefore G \not\supseteq F$

$\Rightarrow D^+ = \{A, B, C, D\}$ $\quad D \to E$ ✗

$\therefore$ These two FD's are not Equivalent

(2)

$F \supseteq G$ $\Rightarrow A^+$ in $F = \{A, B, C\}$ $\quad A \to BC$ ✓

$\Rightarrow B^+$ in $F = \{B, C, A\}$ $\quad B \to A$ holds ✓ $\quad \therefore \boxed{F \supseteq G}$

$\Rightarrow C^+$ in $F = \{A, B, C\}$ $\quad C \to A$ holds

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

Now, B^+ in $G = \{B, A, C\}$ $\quad B \to C$ ✓ $\quad \therefore \boxed{G \supseteq F}$

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

$\therefore$ Bothe 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 FIND MINIMAL SET

1. split the FD's such that RHS contain single attribute.

   Ex: $A \to BC$, $A \to B$ & $A \to C$

2. Find the Redundant FD's and delete them from the set.

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

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

   Ex: $AB \to C$, A - can be deleted if B^+ contains 'A' (or) $\Rightarrow B \to C$
   B - can be deleted if A^+ contains 'B' $\Rightarrow A \to C$

1) Minimise $\{A \to C, AC \to D, E \to AD, E \to H\}$

① $A \to C$, $AC \to D$, $E \to A$, $E \to D$, $E \to H$.

This production is useless (Remove/delete this production and try to find E^+ in the Remaining if it contains 'D' then "$E \to D$" is Redundant FD.

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

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

This will be Redundant if C^+ contains A}

$C^+ = \{C\}$ ∴

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

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

∴ $A \to C$, $A \to D$, $E \to A$, $E \to H$, $E \to D$

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

32. MINIMAL COVER EXAMPLE - 1

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

① $A \to B$ $C \to B$ $D \to A$ ~~$D \to B$~~ $D \to C$ $AC \to D$.

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

Now, $(AC)^+ = \{ACB\}$ $AC \to D$ is not Redundant.

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

Now, $AC \to D$ can be deleted if A^+ contains C (or) C^+ contains A

Now, $A^+ = \{A, B\}$ $\therefore$ $AC \to D$ cannot be deleted.

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

33. GATE - 2013 ON MINIMAL COVER

Is $\{AB \to C, D \to E, E \to C\}$ is the minimal cover of $\{AB \to C, D \to E, AB \to E, E \to C\}$

Sol

Step-1: $AB \to C$ $\quad D \to E$ $\quad \underbrace{AB \to E}$ $\quad E \to C$

$AB^+ = \{A, B, C\}$ $\therefore$ This cannot be deleted and must be in the minimal set

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

34. LOSSLESS DECOMPOSITION

⇒ Mainly Normalisation is about splitting the tables i.e decomposing the tables, so while decomposing we should see some properties. One such property is "Lossless decomposition."

⇒ 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.

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

A | BC

⇒

A	BC
a_1	$b_1 c_1$
a_2	$b_2 c_2$

cross product

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

→ New Tuple.

→ Spurious tuple

Now, if i choose a common Alphabet Randomly.

ABC

AB AC

AB	
a_1	b_1
a_2	b_1
a_1	b_2

AC	
a_1	c_1
a_2	c_1
a_1	c_2

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

Spurious Tuples.

LOSSY DECOMPOSITION

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

ABC

AB BC

AB	
a_1	b_1
a_2	b_1
a_1	b_2

BC	
b_1	c_1
b_2	c_2

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

LOSLESS DECOMPOSITION

⇒ The decomposition is loseless when the common attribute is key in one of the tables. (AB or BC)

∴ For Loseless decomposition.

R_T

R_1 R_2

$(R_1 \cap R_2) \rightarrow R_1$ $(R_1 \cap R_2) \rightarrow R_1 - R_2$

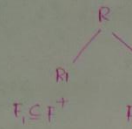
$(R_1 \cap R_2) \rightarrow R_2$ $(R_1 \cap R_2) \rightarrow R_2 - R_1$

The common attribute should be a key in any one of the relation.

35. FD PRESERVING

$R(A_1 A_2 A_3 \cdots A_n)$

FD — F^+ {set of all functional dependencies that are applicable on R}



Now, dependency preserving means

$$\boxed{(F_1 \cup F_2) = F^+}$$

⇒ The dependency preserving is not a mandatory thing

36. DECOMPOSITION - EXAMPLE 1

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

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

⇒ Now we need to check the lossless decomposition and dependency preservation.

⇒ $R_1(AB)$ $R_2(BC)$ ⇒ This decomposition is lossless because the common variable in $R_1 R_2 = B$ and 'B' is a key attribute in $R_2(BC)$ $\{\because B \to C\}$.

Now, $R_1(AB)$	$R_2(BC)$
✓ $A \to B$	✓ $B \to C$ } are the Non-trivial Dependencies.
$B \to A$ ✓	✓ $C \to B$
Non-Trivial Dependencies	$B^+ = \{BCA\}$ $(B \to C)$✓
	$C^+ = \{CAB\}$ $(C \to B)$✓

$F_1 \cup F_2 =$ $A \to B$ $B \to A$ $B \to C$ $C \to B$

(1) (2)

$=$ $A \to B$ $B \to A$ $A \to C$ $C \to B$ Redundant

$=$ $A \to B$ $B \to C$ $C \to A$

∴ The decomposition is lossless and Dependency preserving.

37. DECOMPOSITIO

R(ABCD)

F = (A→B, B→C, C

D = (AB, BC, CD).

Sol Given D =

LOSSLESS DECOMPOSITI

Now comming

R_1 (AB)	R(BC)
✓ A→B	B→C
✓ B→A	C→B
B^+ = {ABCD}	C^+ = {AB

Now

38. DECOMPO

R(ABCD)

F = {AB→CD,

D = {AD, BCD

Sol The com

the Relo

N

∴ The

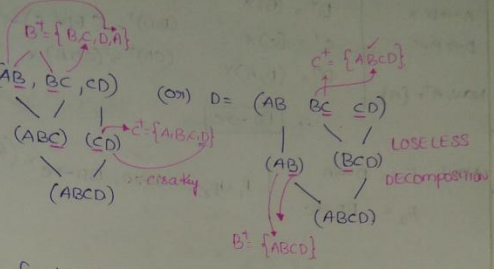
37. DECOMPOSITION Example 2

R(ABCD)

F = (A→B, B→C, C→D, D→A)

D = (AB, BC, CD)

Sol Given D = (AB, BC, CD)



Now comming to functional dependencies

R_1 (AB)	R(BC)	CD = R_3
✓ A→B	B→X ✓	C→D ✓ → Given in Question
✓ B→A	C→B ✓	D→C
B^+ = {ABCD}	C^+ = {ABCD}	D^+ = {ABCD}

Now, $F_1 \cup F_2 \cup F_3$ = A→B, B→C, C→D, D→A {indirectly covered}.

$\boxed{F_1 \cup F_2 \cup F_3 = F}$ F.D. is preserved. (Both)

38. DECOMPOSITION EXAMPLE 3

R(ABCD)

F = {AB→CD, D→A}

D = {AD, BCD}

Sol The common attribute is 'D' now 'D' should be a key in any one of the Relations R_1(AD) R_2(BCD)

Now, D^+ = {D,A} D→A ∴ 'D' is a key attribute in R_1.

∴ The decomposition is losless decomposition

Now,

R_1 (AD)

$A \to D$ ✗

$D \to A$ ✓

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

R_2 (BCD)

$B^+ = (B)$ ✗

$C^+ = (C)$ ✗

$D^+ = (D, A)$ ✗

$(BC)^+ = (BC)$ ✗

$(BD)^+ = (BDAC)$ ✓

$(CD)^+ = (CDA)$ ✗

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

Now. $F_1 = D \to A$

$F_2 = BD \to C$

$F_1 \cup F_2 = D \to A$ ✓, $BD \to C$ ✗ ($(AB)^+ = (AB) \Rightarrow \therefore$ The FD $AB \to CD$ is not preserved)

$\therefore$ The Decomposition is Lossless and non-FD preserving.

39. DEcomposition Example 4

R (ABCDEG)

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

D: (ABC, ABDE, EG)

D: ABC ABDE EG

ABCDE EG

ABCDEG

Now $AB^+ = \{A, B, C, D\}$ Now, $E^+ = \{E, G\}$

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

The decomposition is loseless

Now

R_1 (ABC)

$A \to A$ ✗

$B \to D$ ✗ (D not in table ABC)

$C \to C$ ✗

$(AB)^+ = ABCDEG \Rightarrow AB \to C$

$BC \to A$

$AC \to B$

R_2 (ABDE)

$B \to D$.

$AB \to DE$

$AD \to E$

$(BE) \to D$

$ABD \to E$

$ABE \to D$

R_3 (EG)

$E \to G$

$A^+ = A$

$B^+ = BD$

$C^+ = C$

$(AB)^+ = ABCDEG$

$(BC)^+ = BCDAEG$

$(AC)^+ = ACBDEG$

∴ The decomposition is lossless and Dependency preserving.

40. DECOMPOSITION EXAMPLE 5

R(ABCDE)

F: (A→BC, C→DE, D→E)

R: (ABCD) R_2: (DE).

Now, Given (ABCD) (DE)

∴ ABCD

A→BCD ✓

B→B ✗

C→D ✓

D→D ✗

(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.

Now, D^+ = {E, D}

R_2 (DE)

D→E ✓

E→D ✗ E^+ = {E}

∴ FD's = A→BCD, C→D, BC→D, D→E } = $F_1 \cup F_2$ = F

∴ The decomposition is lossless and dependency preserving.

41. DECOMPOSITION EXAMPLE -6

R(ABCDEG)

F :{ AB→C, AC→B , AD→E , B→D , BC→A, E→G }

D: (AB, BC, ABDE, EG)

D: (AB BC ABDE EG)

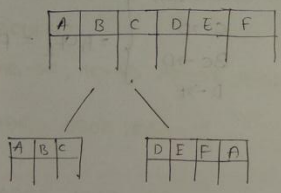
ABC

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

∴ The decomposition is lossy decomposition.

42. FIRST NORMAL FORM

⇒ The process of Removing the Redundancy is Normalisation

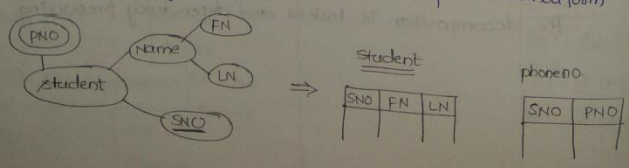


⇒ our ultimate aim is to Reduce a table to BCNF, it is known that by the time you convert the table into BCNF there wont be any Redundancy (Redundancy = 0%)

⇒ 1NF → 2NF → 3NF → BCNF

⇒ 1NF says that the table has to be flat

⇒ By defaut Every Relational DataBase is in First Normal form



⇒ No multivalued and composite values are allowed in 1NF

SECOND NORMAL FORM INTRODUCTION

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

⇒ Now, $AB \to C$, $B \to C$ } $A^+ = A$, $B^+ = BC$, $C^+ = C$ } with one attribute, key is not possible ∴ Try with 2 attributes.

$(AB)^+ = (ABC)$ ∴ (AB) has the capacity to become Candidate Key.

Now, $AB \to C$ → say that AB (combination) can determine 'C'
$B \to C$ → 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

⇒ 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.

⇒ 2NF is based on full functional Dependency

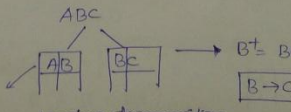
⇒ partial dependency is not allowed (A part of the key determines some thing).

↓

This table is Not in 2NF because there is a partial dependency ($B \to C$) in the candidate key ($\underbrace{AB}_{\text{partial key}} \to C$) so we should eliminate partial dependency.

Find $B^+ = \{BC\}$

$A^+ = \{A\}$



× $A^+ = A$
× $B^+ = BC$
× $AB = \{ABC\}$

Loseless decomposition.
Not FD preserving.

44. 2NF Example 1

R(ABCD)

FD: {AB → C, B → 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) is the candidate key.

$AB^+ = ABCD$

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

$(ACD)^+ = (ACD)$

$(BCD)^+ = (BCD)$

∴ (AB) → ABCD

B → D → partial Dependency

Now, find $A^+ = A$ $B^+ = BD$

R(ABCD)

R(ACB) R(BD)

Remaing we inserted so that we should have some common part

R(ABCD)

R(ACB) R(BD)

A	C	B

B	D

AB → C B → D

FD preserving too.

Lossless decomposition.

45. 2NF EXAMPLE 2

R(ABCDEFGHIJ)

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$H \rightarrow J$

Candidate Key = ABD.

$(ABD) \rightarrow (ABCDEFGHIJ)$

$$\left.\begin{array}{l} BD \rightarrow EF \\ AD \rightarrow GH \\ A \rightarrow I \\ AB \rightarrow C \end{array}\right\} \text{partial Dependencies}$$

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

$(AB)^+ = (ABC)$

R(A~~B~~~~C~~DEFGHIJ)

$AB \rightarrow C$

A	B	C

$(BD)^+ = (BDEF)$

R(A~~B~~~~C~~~~D~~~~E~~~~F~~GHIJ)

$BD \rightarrow EF$

B	D	E	F

$(AD)^+ = (A, D, G, H, J)$

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

A	D	G	H	J

$AD \rightarrow GH$

$H \rightarrow J$

$A^+ = (A, I)$

A	I

$A \rightarrow I$

ABD

A	B	D

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

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

$A \rightarrow B$

$B \rightarrow E$

A	B	E

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

R(A ~~B~~ C ~~D~~~~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.

47. THIRD NORMAL FORM INTRODUCTION

3NF: NO Transitive Dependencies

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

R(ABC)

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

Prime attribute: The attribute which is a part of some candidate key.

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

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

Transitive Dependency.

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

R (A B C)

Lossless Decomposition

FD preserving

A	B

$A \to B$

B	C

$B \to C$

48. 3NF Example 1

R(ABCDE)

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

CK = (AB)

Now, partial dependencies are $B \to D$

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

$B \to D$

$D \to E$

B	D	E

A	B	C

$AB \to C$ 2NF

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

$D \to E$ is

⇒ Now $D^+ = \{$

R (B

B

$B \to D$

49. 3NF EXAM

R(ABC)

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

V.V.V.V. Imp.

partial depend

Here $C \to A$

∴ $C \to A$ is

Ex

R(

CK

A →

A →

A →

B →

Here $D \to E$ is the Transitive dependency. ∴ The table (BDE) is not in 3NF.

(47)

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

⇒ Now, $D^+ = \{D, E\}$

R (B D ~~E~~)

B	D

$B \to D$

D	E

$D \to E$

A	B	C

$AB \to C$

3NF

49. 3NF EXAMPLE 2

R(ABC)

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

Now, Candidate Key = (B) = (~~A~~BC)

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

$(AB)^+ = \{ABC\}$ ✓

$(BC)^+ = \{BCA\}$ ✓

∴ Candidate Keys = (AB) (BC)

∴ All are prime attributes

V.V.V.V. Imp.

partial dependency = part of key → Non-prime attribute

⇒ prime attribute → Non-prime attribute

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

∴ $\boxed{C \to A}$ is not partial dependency.

Ex:

R(ABCDEF)

CK: ABC

$A \to D$ = partial dependency.

$A \to B$
$A \to C$
$B \to C$ } = Not partial dependencies

In the above table there are no partial dependencies ∴ The Relation is in 2NF

Transitive dependency (50)

Non prime attribute	→	Nonprime attributes
⇓		
Subset of Non-prime attributes	→	Subset of Non prime attributes

R(ABCDEF)
CK = ABC } ⇒ D→E, E→F } Transitive Dependencies.

B→C, D→C } Not Transitive dependency

∴ The above Relation doesnot contain Transitive dependency

⇒ ∴ The Relation is in 3NF

50. FORMAL DEFINITION OF 3NF

A Relational schema 'R' is in 3NF only if every Non-trivial FD X→Y

either 1> X is a superkey

or

2> Y is a prime attribute

Which of the following is allowed in 3NF?

a) proper subset of CK → Non prime (partial dependency)

b) Non-prime → Non-prime (Transitive dependency)

c) proper subset of CK + Nonprime → Nonprime (Transitive dependency)

d) proper subset of CK → proper subset of other CK (Not a PD, TD) ✓

51. 3NF EXAMPLE 3

R(ABCDEF) $F: \{A \to FC,\ C \to D,\ B \to E\}$

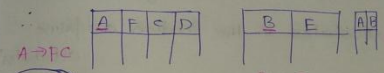
Now, candidate key = $(AB)^+ = (ABCDEF)$

Now, $(AB)^+ = (ABFCDE)$ ∴ (AB) is the candidate key

Now, $\left.\begin{array}{l} A \to FC \\ B \to E \end{array}\right\}$ are the partial dependencies.

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

∴



∴ The Relation is in 2NF

$A \to FC$

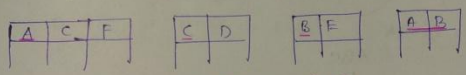
$C \to D$

⇓

Transitive Dependency.

$B \to E$

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



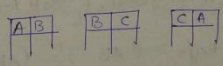
3NF

$A \to FC$ $C \to D$ $B \to E$

52. BCNF INTRODUCTION

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

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



⇒ candidate keys = $\{A, B, C\}$

BCNF

53. BCNF EXAMPLE 1

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

⇒ The candidate key = $(A)^+ = (ABC)$

$A^+ = \{A\}$

∴ $(AB)^+ = \{ABC\}$ ∴ (AB) (AC) are candidate Keys ⇒ A, B, C are prime attributes

$(AC)^+ = \{ACB\}$

∴ There are no partial dependencies in the table

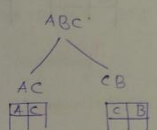
∴ The table is in 3NF because the candidate key contains all the attributes. (All the attributes in the Relation are prime attributes)

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

⇒ $(AB)^+ = \{ABC\}$

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

Now,



BCNF

⇒ Lossless Decomposition

⇒ NOT FD preserving ($AB \to C$ is lost)

54. BCNF EXAMPLE 2

R (ABCDEFGHIJ)

F: ($AB \to C$, $A \to DE$, $B \to F$, $F \to GH$, $D \to IJ$)

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

Now, $(AB)^+ = (ABCFGHDIJE)$ ⇒ A, B are only the prime attributes

Now, the partial dependencies are $\left\{ \begin{array}{l} A \to DE \\ B \to F \end{array} \right\}$

$A^+ = (ADEIJ)$ $B^+ = (BFGH)$

Now, the Relation

$D^+ = (DIJ)$ (ADE)

$F^+ = (FGH)$ $A \to DE$

⇒ To check BCN

on a table

∴ The Relation

R_1 (ADE)

$A \to DE$

$A^+ = \{ADE\}$

A = SK

55. BCNF Exa

R(ABCDEFGHIJ)

Candidate

Now, $(AB)^+$

⇒ Now B→

A→

Now,

Transitive

Now, the Relation R(ABCDEFGHIJ)

- R_1(ADEIJ): $A \to DE$, $D \to IJ$ (TD) → (ADE), (DIJ)
  - (ADE): $A \to DE$
  - (DIJ): $D \to IJ$
- R_2(BFGH): $B \to F$, $F \to GH$ (TD) → (BF), (FGH)
  - (BF): $B \to F$
  - (FGH): $F \to GH$
- R_3(ABC) ⇒ NO POIs ⇒ 2NF
  - $AB \to C$
  - → (ABC), $AB \to C$ } in 3NF

$D^+ = (DIJ)$

$F^+ = (FGH)$

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

∴ The Relations that we get are

R_1(ADE)	R_2(DIJ)	R_3(BF)	R_4(FGH)	R_5(ABC)
$A \to DE$	$D \to IJ$	$B \to F$	$F \to GH$	$AB \to C$
$A^+ = \{ADE\}$	$D^+ = \{DIJ\}$	$B^+ = \{FB\}$	$F^+ = \{FGH\}$	$(AB)^+ = \{ABC\}$
A = SK	D = SK	B = SK	F = SK	AB = SK

BCNF

55. BCNF Example 3

R(ABCDEFGHIJ) FD: { $AB \to C$, $B \to D$, $D \to EF$, $A \to GH$, $H \to IJ$ }

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

Now, $(AB)^+ = (ABCDEFGHIJ)$ (AB) is the candidate key

⇒ A, B are prime attributes. ⇒ $A^+ = \{AGHIJ\}$, $B^+ = \{BDEF\}$

⇒ Now $B \to D$, is the partial dependency ⇒ $B^+ = \{B, D, E, F\}$

$A \to GH$

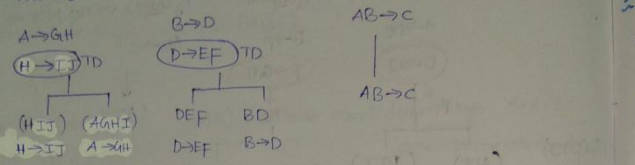
$B \to D$ is possible

Now, (ABCDEFGHIJ)

- R_1(BDEF): $B \to D$, $D \to EF$ — Transitive Dependency
- R_2(AGHIJ): $A \to GH$, $H \to IJ$ — TD.
- R_3(ABC): $AB \to C$

Both losless and FD preserving

Now,



∴ The tables are
1) HIJ
2) AGHI
3) DEF
4) BD
5) ABC

are in BCNF

56. BCNF Example 4

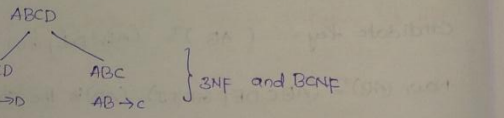
R(ABCD) FD = {AB→C, BC→D}

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

$(AB)^+ = \{(ABCD)\}$ (AB) is the candidate key

Now, There are no partial dependencies

Now, BC→D is the Transitive dependency, BC is not a superkey



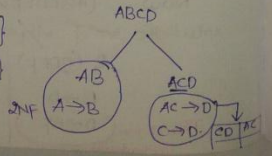
57. BCNF Example 5

R(ABCD) FD: {A→B, C→D} CK: {AC} A, C are prime attributes

partial dependency: A→B, C→D

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

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



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

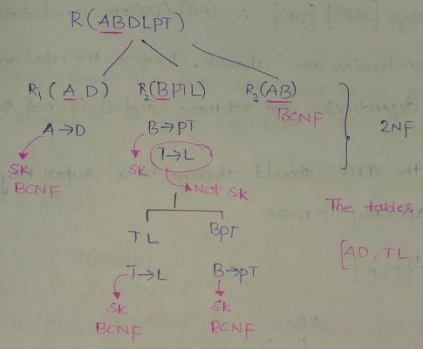
58. BCNF Example - 6

R=(ABDLPT) FD{B→PT, T→L, A→D}

Candidate Key = $(AB)^+ = (ABDLPT) \Rightarrow (AB)^+ = (ABDPTL)$ //

$\boxed{CK = AB}$

Now, partial Dependencies = B→PT, A→D } $B^+ = (B, P, T)$, $A^+ = (A, D)$



The tables are

[AD, TL, BPT, AB]

59. BCNF Example 7

R(ABCDE) FD{A→BC, CD→E, B→D, E→A} Candidate Key = {A, E, CD, BC}

All the attributes are prime

⇒ 3NF

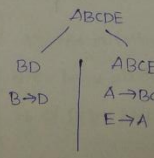
Now, To check whether the Relation is in BCNF check the functional dependencies and find whether the LHS is a super key or not

A→BC ⇒ $A^+ = (ABCDE)$ ⇒ A is SK

CD→E ⇒ $(CD)^+ = (CDEAB)$ ⇒ CD is SK

B→D ⇒ $(B)^+ = (B, D)$ ⇒ NOT SK

E→A ⇒ $(E)^+ = (E, A, B, C, D)$ = E is SK



ABCDE

BD	ABCE
$B \to D$	$A \to BCE$ $E \to ABC$ $BC \to AE$

} BCNF

61. BCNF Example 8 - Part 1 and 62 - Part 2

R(ABCD)

FD{$AB \to CD$, $D \to A$}

⇒ candidate key = {(AB)} {(DB)} ∴ (AB), (DB) are candidate keys.

⇒ partial dependencies are absent. Therefore the relation is in 2NF

⇒ Transitive dependencies are not there and 'D' is not superkey but A is prime attr.

⇒ For BCNF the LHS should always be a super key, and here 'D' is not super key in $D \to A$

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

Lossless Decomposition

ABCD

DA	BCD
✓ $\boxed{D \to A}$ BCNF (2 attributes)	$B^+ = (B)$ $C^+ = (C)$ $D^+ = (D, A)$

$(BC)^+ = (BC)$

$(BD)^+ = (BDAC) \Rightarrow BD \to C$ ✓

$(CD)^+ = (CDA)$

Now, find if $AB \to CD$ is preserved using $D \to A$ and $BD \to C$

⇒ $(AB)^+ = (AB)$ ∴ $AB \to CD$ is not preserved

∴ Dependency preserving failed.

63. BCNF EXAMPLE 9

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

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

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

$C^+ = \{C, D\}$ $(CD)^+ = (CD)$ ×

$D^+ = \{D\}$

Now, A, B are prime attributes ⇒ Now, partial dependencies are not prese
(only single attribute CK's).

Now for 3NF check the LHS, LHS must be a super key or the RHS should be a prime attribute in $C \to D$ ⇒ violating 3NF

Not SK — Not prime attribute

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

R(ABCD)

R_1(CD) — R_2(ABC)

$\boxed{C \to D}$ — 3NF and BCNF (2 attributes)

SK $A \to B$
SK $B \to A$
SK $B \to C$
} 3NF and BCNF

lossless and FD preserving

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

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

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

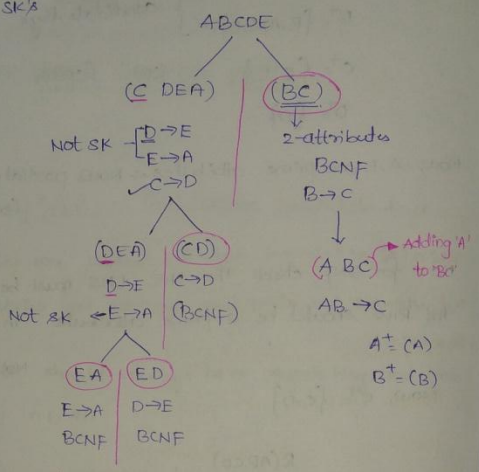
$(B)^+ = \{B\}$ Now $(AB)^+ = \{A, B, C, D, E\}$
$(CB)^+ = \{B, C, D, E, A\}$
$(DB)^+ = \{B, D, E, A, C\}$
$(EB)^+ = \{B, E, A, C, D\}$
} All the attributes are prime attributes ⇒ 'R' is in 3NF

For BCNF, every LHS of the functional Dependency should be a Super Key

$AB \to C \Rightarrow AB$ (SK)

$C \to D$
$D \to E$
$E \to A$ } C, D, E are not SK's

Now, $C^+ = (CDEA)$



But $(AB \to C)$ 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 LOSLESS 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 guarentee dependency preserving)

GATE-98

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

a) 2NF b) 5NF c) 4NF d) 3NF (Actual ans is BCNF)

GATE-99

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

$R(ABCDEF) \Rightarrow (EC)^+ = (ABCDEF) \Rightarrow (EC)^+ = \{E, C, A, B, F, D\}$

$\therefore$ EC is the key.

GATE-01

consider the schema R(ABCD) and functional dependencies $A \to B$ and $C \to D$. 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

$R(ABCD)$

$R_1(AB)$ $R_2(CD)$ $\Rightarrow$ No common attribute

$\therefore$ lossy decomposition.

Now, $R_1(AB)$	$R_2(CD)$
$A^+ = (A, B)$	$C^+ = (C, D)$.
$B^+ = (B)$	$D^+ = (D)$

$F_1 : (A \to B)$ $F_2 : (C \to D)$

$\boxed{F_1 \cup F_2 = F}$ $\therefore$ Dependency preserving.

GATE 02

Relation R with an associated set of FD's (F) is decomposed into BCNF. The Redundancy (arising 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 GUARENTE)

(d) Any Relation with 2 attributes is in BCNF. (TRUE) LHS = Key (SK) RHS =

GATE-II-05

A table has fields $F_1\ F_2\ F_3\ F_4\ F_5$ with the following functional dependencies $F_1 \rightarrow F_3$, $F_2 \rightarrow F_4$, $F_1F_2 \rightarrow F_5$ what is the NF of the Relation?

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

Now, Candidate Key = $(F_1F_2)^+ = (F_1\ F_2\ F_3\ F_4\ F_5)$

$(F_1F_2)^+ = (F_1F_2F_3F_4F_5)$ ∴ F_1F_2 is candidate key.

Now, partial dependencies are present ∴ 2NF × $\begin{pmatrix} F_1 \rightarrow F_3 \\ F_2 \rightarrow F_4 \end{pmatrix}$

Now, LHS should be super key for 3NF ⇒ $\left.\begin{matrix} F_1 \\ F_2 \\ F_1F_2 \end{matrix}\right\}$ are SK's = 3NF

GATE-12

which of th

a) Every

b) A Relation functiona

c) Every

d) NO Relat

68. GATE

GATE-14

A prime a

a) In all

b) In som

c) In a

d) Only in

GATE-95

Consider

∴ Now,

GATE-97

R(a,b,c,d)

FD: {a→c

Now

GATE-12

Which of the following is True?

a) Every relation in 3NF is also in BCNF (Not always)

b) A Relation 'R' is in 3NF if every non-prime attribute of 'R' is fully functionally dependent on every key of R → some key

c) Every relation in BCNF is in 3NF

d) NO Relation can be in both BCNF and 3NF (FALSE)

68. GATE QUESTION ON NORMALISATION 3

GATE-14

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

a) In all candidate keys of R

b) In some candidate key of R

c) In a foreign key of R

d) Only in the primary key of R.

GATE-95

Consider R(ABC) FD: {AB→C, C→A} show that R is in 3NF but not BCNF

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

$(AB)^+ = (ABC)$
$(CB)^+ = (ABC)$ } All are prime attributes ∴ the relation is in 3NF

∵ AB→C SK
C→A NOT SK } Not in BCNF

GATE-97

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

FD: {a→c b→d}

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

⇒ ab is the ck

⇒ a→c, b→d } are partial Dependency.

a) 1NF but not in 2NF

b) 2NF but not in 3NF

c) IN 3NF

d) None of the above

69. GATE QUESTIONS ON NORMALISATION 4

GATE-99

R(STUV); FD{S→T, T→U, U→V, V→S} and Let (R_1 and R_2) = R be a decomposition such that $R_1 \cap R_2 \neq \phi$. 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 = $(S)^+$ = (STUV)
$(T)^+$ = (TUVS)
U^+ = (UVST)
V^+ = (VSTU)

All are prime ⇒ NF = 3NF

70. 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→B, B→CD
b) A→B, B→C, C→D
c) AB→C, C→AD
d) A→BCD

optionA: A→B, B→CD R(ABCD)

CK = $(A)^+$ = (ABCD) No partial dependencies = 2NF

Transitive depency = B→CD

B^+ = (B,C,D)

ABCD
- BCD: B→CD, BCNF
- AB: A→B, BCNF

Lossless
FD preserved

option B: A→B, B→C, C→D → Transitive dependencies

CK = A $(A)^+$ = (ABCD) ⇒ No partial dependencies

ABCD
- BCD: B→C, C→D
  - CD: BCNF (C→D)
  - BC: BCNF (B→C)
- AB: (A→B) BCNF

Lossless and FD preserving

option C: $AB \to C$ $C \to AD$ CK = (AB), (CB) are candidate keys

$C \to AD$ is the partial Dependency

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

ABCD

CAD — $C \to AD$

BC — 2-

Now, $B^+ = B$

$C^+ = C$

The dependency $\boxed{AB \to C}$ is lost

71. 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 is which. To make guaranteed identification, which one of the following tests should be used on the decomposition?

a) Dependency - preservation ⇒ BCNF may not have dependency preserving some times

b) Lossless join ⇒ Both have lossless but we cannot differentiate them.

c) BCNF definition ⇒ Enough to prove BCNF table and other automatically will be 3NF

d) 3NF definition ⇒ Not Enough, BCNF cannot be identified.

72. GATE 04 QUESTION ON NORMALISATION

The Relation scheme student performance (name, courseNo, rollNo, grade) has the FD's

Name, courseNO → grade

RollNO, courseNO → grade

Name → rollNO

RollNO → name

The highest normal form of this Relation scheme is

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

candidate Key = $(CNO)^+$ = (name, rollNO, courseNO, Grade)

(64)

$(CNO)^+ = (CNO)$

Let us assume, name = A rollNO = C FD's

courseNO = B grade = D

$AB \to D$
$CB \to D$
$A \to C$
$C \to A$

R(ABCD)

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

$B^+ = B$

$(AB)^+ = (ABCD)$
$(CB)^+ = (ABCD)$ } are CK's. ⇒ (AB) (CB) ⇒ prime attributes are A, B, C

$(DB)^+ = (DB)$

Now, partial dependencies are Absent ∴ 2NF ✓

Now, 3NF check

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

Now, BCNF check

⇒ LHS should be Superkey

⇒ AB = SK
CB = SK
A → C
C → A
→ NOT SK
} BCNF ✗

∴ $\boxed{\text{Ans} = 3NF}$

73. GATE IT - 04 QUESTION ON NORMALISATION

A Relation Empdt1 is defined with attributes empcode (unique), name, Street, city, state and pincode. For any pincode there is only one city and state. Also, for any pincode, there is only one city and state. Also, for any given street, city and state, there is just one pincode. In Normalisation terms Empdt1 is a relation in.

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

Let empcode = A Street = C state = E R(ABCDEF)
name = B city = D pincode = F

Given (empcode) is the candidate key ⇒ $A \rightarrow BCDEF$

$F \rightarrow DE$

$CDE \rightarrow F$

Now, 'A' is the candidate key ⇒ No partial dependencies are present.

⇒ 2NF ✓

⇒ Now, 3NF check, [LHS = SK or RHS = prime attribute]

$F \rightarrow DE$

Not SK Not prime 3NF X

∴ $\boxed{2NF}$

74. GATE 07 QUESTION

Which one of the following statement is false?

a) Any relation with 2 attributes is in BCNF

b) Any relation with every key having only one attribute is in 2NF

c) A prime attribute can be transitively dependent on a key in 3NF

d) " " " " " " " " " BCNF

75. GATE 2008 QUESTION ON NORMALISATION

Consider the Relation schemas of Library DB

Book (Title, Author, catalog-no, publisher, year, price)
(A, B, C, D, E, F)

Collection (Title, Author, Catalog no) with the dependencies
(A, B, C)

I. Title Author → catalog-no

II. catalog-no → Title, Author, publisher, year

III. publisher, year title → price

Assume (Author, Title) is the key for both schemas. 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)

FDs: $AB \to C$, $C \to ABDE$, $DAE \to F$

Collection: Now, (ABC)

$AB \to C$

$C \to AB$

All are prime = 3NF

Book: (ABCDEF)

$AB \to C$

$C \to ABDE$

$DAE \to F$

Transitive Dependency → 2NF

Book = 2NF
Collection = 3NF

16. GATE IT-08 AND 2013 QUESTION ON NORMALISATION

Let R(ABCDEPG) 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 = $(AB)^+ = (ABCDEPG)$

Now, $(AB)^+ = (ABCDEPG)$ $\therefore$ (AB) is the candidate key

Now, the functional Dependencies are

$AB \rightarrow CD$

$DE \rightarrow P$
$C \rightarrow E$ } Transitive dependency.
$P \rightarrow C$

$B \rightarrow G$ 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 (Imp for GATE)
→ Relational calucculus

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

Relational calucculus 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 2 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 ($\sigma \times ( )$)

⇒ $/$: Division ($\pi, \times, -$)

2. SELECTION

⇒ This operation

⇒ Selection / Ho

Emp

ENO	Ename	DNO

From this it is that selection ope is commutativ

⇒ Select opera

⇒ Let us co in the

The syntax

2. SELECTION OPERATION (σ):

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

⇒ Selection / Horizontal partition

Emp

ENO	Ename	DNO	Salary

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

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

From this it is clear that selection operation is commutative

⇒ $\sigma_{DNO=4}(\sigma_{Sal>10,000}(Emp))$ ⇒ list of Employees whose sal > 10,000 and who belong to DNO=4.

⇓

⇒ $\sigma_{Sal>10,000}(\sigma_{DNO=4}(Emp))$ ⇒ " " " .

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

⇒ 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

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

max |R| tuples — min '0' tuples

The syntax of selection operation is

$\sigma_{<\text{selection condition}>}(\text{Relation name})$

↳ Can be Relational Algebra Expression also)

↓

Boolean Expression

(attribute name) <Comparision operator> (constant)

" " " (attribute name)

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, $\pi_A(R)$:

Degree = 1

A
1
2
3

$\pi_{AB}(R)$:

Deg = 2

A	B
1	a
2	c
3	e

⇒ General syntax of projection operation is

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

↳ can be 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
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,

$$|R| \geq |\pi_{\langle attrib \rangle}(R)|$$

$\boxed{\pi_{\langle attrib \rangle}(R) < R}$, when <attribute> is not superkey

$\pi_{\langle attrib \rangle}(R) = R$, when <attrib> is a superkey

⇒ The projection

Let R(ABC)

A	B	C

Let 'R' be a Rel

⇒ The "projectio

4. RENAME O

Now, consider a

R

A	B	C

Representation

⇒ sometimes t attributes of

⇒ sometimes y then the s

⇒ Renaming o

⇒ The projection operation is not "commutative".

Let R(ABC)

A	B	C

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

A	B

A

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

A

AB not possible

Let 'R' be a Relation, then $\boxed{\pi_{A_2}(\pi_{A_1}(R))}$ is valid only when $\boxed{A_2 \subseteq A_1}$

A_2: ✓A, ✓B, ✓AB

A_1: AB

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

A	B
1	a
2	c
3	e

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

4. RENAME OPERATION (ρ)

Now, consider a Relation R(ABC)

R

A	B	C

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 (Online Representation they represent in different table.

$$Temp \leftarrow \sigma_x(R)$$

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

$\pi_B(R) =$

B
a
b

⇒ 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) → 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".

5. GATE 98 QUESTION ON SELECTION AND PROJECTION

Which of the Query transformations (i.e Replacing the LHS 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

a) $\sigma_{C_1}(\sigma_{C_2}(R_1)) \rightarrow \sigma_{C_2}(\sigma_{C_2}(R_1))$ - False

b) $\sigma_{C_1}(\pi_{A_1}(R_1)) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1))$ = True

c) $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \cup \sigma_{C_2}(R_2) \Rightarrow$ TRUE

d) $\pi_{A_2}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_2}(R_1)) \Rightarrow$ FALSE

if C_1 condition is in such a way that it contains values in $A_1 A_2$ columns, we are extracting A_2

A_1 will be missied out

6. GATE 2012 QUESTION ON PROTECTION

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

a) $\pi_B(r_1) - \pi_C(r_2) = \phi$ (b) $\pi_C(r_2) - \pi_B(r_1) = \phi$ (c) $\pi_B(r_1) = \pi_C(r_2)$

d) $\pi_B(r_1) - \pi_C(r_2) \neq \phi$

Sol: 'B' is the foreign key of 'C' in R_2 ⇒ Every value of 'B' will be in 'C'.

A)

$\therefore \pi_B(r_1) - \pi_C(r_2) = \phi$ ✓

C) $\pi_B(r_1) = \pi_C(r_2)$ [In some cases but not always]

B)

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

D) $\pi_B(r_1) - \pi_C(r_2) \neq \phi$

{contradiction of option}

→ 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 OPERATIONS

⇒ The next set of operations are Union (∪), Intersection (∩), Minus (−)

⇒ when we apply the above operations on Relations then they should be "union compatable" or "Type compatability"

$R(A_1\ A_2 \cdots A_n)$
$S(B_1\ B_2 \cdots B_n)$] we can perform R∪S iff R, S have
1) have same degree
2) corresponding elements domain must be same.

Now, $(R \cup S) = R(A_1\ A_2 \cdots A_n)$
↳ we use the LHS set name of $R \cup S = R(A_1\ A_2 \cdots A_n, B_1\ B_2 \cdots B_n)$ }

∴ The Union operator is comutative = R∪S = S∪R [No duplicates]

∴ The Intersection operator is comutative = R∩S = S∩R [No duplicate values]

∴ The Difference operator is not commutative = (R−S) ≠ (S−R) [No duplicates]

⇒ Intersection is the derived operation $\boxed{R \cap S = ((R \cup S) - (R - S)) - (S - R)}$

$\boxed{R \cap S = R - (R - S)}$

8. CARTESIAN PRODUCT

(i) Binary operation

(ii) Relations need not be Union compatable

Emp

A	B	C

Dependent

D	E

⇒ Now Emp × dependent

A	B	C	D	E

⇒ If the table R_1 contains 'r' attributes and table R_2 contains 's' no of attributes then $\boxed{R_1 \times R_2}$ contain $\boxed{(r+s)}$ attributes (columns)

⇒ If R_1 has 'r' tuples and R_2 has 's' tuples then $\boxed{R_1 \times R_2}$ has $\boxed{(r \times s)}$ tuples

Employee

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

$(m)_1$ rows = 3

$(m)_2$ colums = 3

Dependent

D	E
1	a1
2	a2

$(r)_1$ rows = 2

$(r)_2$ columns = 2

Employee × 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

$(m_1 * r_1)$ rows

$(m_2 + r_2)$ columns

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

⇒ $\sigma_{A=D}$ (Employee × Dependent) ⇒

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

∴ A new operation called join ⋈ is developed to specify the cross product with a condition.

∴ Join is a combination of (× and σ).

9. JOIN AND NATURAL JOIN (⋈)

⇒ "Join" operation is used to combine related tuples from two operations / Relations into a single larger tuple.

Ex:

R

A	B	C

S

D	E

(R ⋈ S)

A	B	C	D	E

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

Ex:

R(ABC)

A	B
1	a
2	b
3	c

The syntax o

Now, some

operation ar

Automatical

called Nature

performed on

R

A	B	C

10. DIVISION O

R

A	B

E
a1
a2
a1
a2
a1
a2

$(m_1 * n_1)$ rows

$(m_2 + n_2)$ columns

Ex:

R(ABC)

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

S(DE)

D	E
1	a
2	b
2	c

(R ⋈ S) (A=D)

A	B	C	D	E
1	a	d	1	a
2	b	e	2	b
2	b	e	2	c

The syntax of Join is [R ⋈ <joincondition> S]

<condition> AND <condition>

↓

A_i AND B_j ⇒ This join is called "Theta" join (No comparision against with values, comparision only between attributes)

Now, some conditions are frequently used while performing join operation and so we dont 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.

R

A	B	C

S

A	B	D

R*S

A	B	C	D

[R ⋈ S <A=A> AND <B=B>]

'A' of 1st table of 2nd table

10. DIVISION OPERATION (÷)

R

A	B

S

A

R÷S(R−S)

B

R÷S ⇒ what ever attributes we have in 'S' you substract them from 'R' and write what is remaining in R:

⇒ The attributes of (R−S) will be in R÷S.

Ex 1:

R

A	B
a1	b1
a2	b1
a1	b2

S

A
a1
a2

R÷S

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 (a_1, a_2) of S so b1 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	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

R÷S

B
b1
b4

These are the values associated with $a_1 a_2 a_3$ = attributes of S.

Now, for the above example 1

$T_1 \leftarrow \pi_{(R-S)}(R) = B =$

B
b1
b2

$T_2 \leftarrow \pi_{(R-S)}\left[(S) \times T_1\right] - R) = \pi_B\left[(S \times B) - R\right] \Rightarrow$

S

A
a1
a2
a3

B

B
b1
b2

S×B − R

A	B
a1	b1
~~a2~~	~~b1~~
a1	b2
a2	b2

$T \leftarrow T_1 - T_2$

B
b1
~~b2~~

B
~~b2~~

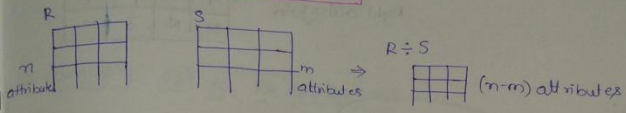
B
b1

T_2

B
b2

∴ The implementation of division using the basic operations is

$$T_1 \leftarrow \pi_{(R-S)}(R)$$
$$T_2 \leftarrow \pi_{(R-S)}\left[(S \times T_1) - R\right]$$
$$T \leftarrow T_1 - T_2$$



$\boxed{\text{Attributes in } S \subseteq \text{Attributes in } R}$

⇒ If R contains 'x' attributes, S contains y attributes then the Relation $Z = (R \div S)$ contains $x - y$ attributes

$$Z = (X) - (Y)$$
$$\Rightarrow \boxed{X = Z \cup Y}$$

11. AGGREGATE FUNCTIONS AND OUTER JOINS

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

⇒ Another additional function provided is outer join. outerjoin 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

outer join
- Left outer Join ($R ⟕ S$)
- Right outer Join ($R ⟖ S$)

⇒ 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
1	–
2	–

S

B	D
1	–
3	–

Now, $(R \bowtie_{A=B} S)$

A	C	B	D
1	–	1	–
2	–	N	N

Now, $(R \bowtie_{A=B} S)$

Right outer join

A	C	B	D
1	–	1	–
N	N	3	–

Ex:

Emp

ID	EN	ENO

Department

DNO	DN	MID

→ Manager ID.

Emp $\bowtie_{\langle ID=MID \rangle}$ Department =

ID	EN	ENO	DNO	DN	MID
			N	N	N

} Tuples where MID matches with EID

→ which does not match with MID but present in Emp table

Emp $\bowtie_{\langle MID=ID \rangle}$ Department =

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

} Tuples matching with EID

} Not matching and present in Department table

Emp $\bowtie$ Department = Full Outer Join.

Not Equal to Cross product of Employee and Department

ID	EN	ENO	DNO	DN	MID
2	Ravi	555	N	N	N
N	N	N	9	Pavan	509

} Tuples matching

} Not matching

Imp :

	Min	Max
$\times$	$(m \times n)$	$(m \times n)$
$\bowtie$	(0)	$(m \times n)$
$(R \bowtie S)$ $\bowtie$	n	$(m \times n)$
$\bowtie$	m	$(m \times n)$
$\bowtie$	$\max(m,n)$	$(m \times n)$

R - n tuples

S - m tuples

12. COMPLETE SET OF RA OPERATIONS

The operations $\{\sigma, \pi, \cup, e, -, \times\}$ = complete set

$\{\bowtie, \cap, \div, \bowtie, \bowtie, \bowtie\} \Rightarrow$ can be derived from complete set.

Now,

	Min	Max
$\cup$	$\max(m,n)$	$(m+n)$
$\cap$	0	$\min(m,n)$
$-$	0	m

$R \Rightarrow m$ tuples

$S \Rightarrow n$ tuples.

13. GATE QUESTIONS ON RA ...1

GATE-94

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

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

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

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

a) m+n and 0 (b) mn and 0

c) m+n and $|(m-n)|$ (d) mn and m+n

GATE -99

The Relational Algebra expression equalent to the following tuple calculus expression $\{t / t \in r \wedge (t[A] = 10 \wedge t[B] = 20)\}$ is:

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

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

c) $\sigma_{(A=10)}(r) \cap \sigma_{(B=20)}(r)$

d) $\sigma_{(A=10)}(r) - \sigma_{(B=20)}(r)$

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 ⇒ using join Emp ⋈ Dept <e.dno = d.duno> This will get address

b) Employees whose name is same as Dept. name ⇒ <e.ename = d.dname>

c) The sum of all employee salaries. = AGGREGATE FUNCTION

d) All employees of a given department = $\sigma_{e.ename}(\sigma_{d.duno})$ get dept name

select employee corresponding to that particular dept number

14. GATE 2003 QUESTION ON CONVERTING SQL TO RA

Consider the following SQL Query

select distinct $a_1 a_2 \cdots a_n$

From $r_1 r_2 \cdots r_m$

where P.

For an arbitrary predicate 'p' this query is equalent to which of the following Relational Algebra expressions

a) $\Pi_{a_1 a_2 \ldots a_n} \sigma_p (r_1 \times r_2 \times \ldots \times r_m)$

b) $\Pi_{a_1 a_2 \ldots a_n} \sigma_p (r_1 \bowtie r_2 \bowtie \ldots r_n)$

c) $\Pi_{a_1 a_2 \ldots a_n} \sigma_p (r_1 \cup r_2 \cup \ldots r_n)$

d) $\Pi_{a_1 a_2 \ldots a_n} \sigma_p (r_1 \cap r_2 \cap \ldots r_n)$.

select distinct = projection

where = Selection.

from = Cross product

① $\boxed{R1} \times \boxed{R2} \times \boxed{R3}$ From clause

$\Downarrow$ σ = where

$R_1 \times R_2 \times R_3 \times \ldots R_n$

$\Downarrow$ Π = Select distinct

$\Rightarrow \Pi_{a_1 a_2 \ldots a_n} \sigma_p (r_1 \times r_2 \times r_3 \ldots r_n)$

15. GATE 04 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 relation instances r_1 and r_2. which one of the following Relational Algebra expressions would necessarily produce and empty relation?

a) $\Pi_D(r_2) - \Pi_C(r_1)$ ✓ (B) $\Pi_C(r_1) - \Pi_D(r_2)$ (C) $\Pi_D(r_1 \bowtie_{C \neq D} r_2)$ (D) $\Pi_C(r_1 \bowtie_{C=D} r_2)$.

Sol

R_1(ABC) $\xrightarrow{FK}$ R_2 (DE) $\Longrightarrow$ The values present in the 'C' will definitely be present in 'D'.

A	B	C
		1
		1
		2
		2

D	E
1	
2	
3	
4	

$\{1, 2\} - \{1, 2, 3, 4\}$

$= \phi$.

$\therefore \Pi_C(r_1) - \Pi_D(r_2) = \phi$

16. GATE 05 QUESTION ON DECOMPOSITION AND JOIN

(82)

Let 'r' be a relation instance with schema R(ABCD). we define $r_1 = \Pi_{ABC}(r)$ and $r_2 = \Pi_{AID}(r)$. Let $S = r_1 * r_2$ where * 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) $r * S = S$.

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

∴ 'S' will have additional tuples (also called spurious tuples) that are not present in 'r'

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

17. GATE 2014 QUESTION ON CASCADING 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 $A_1 \subset A_2$ 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) $\Pi_{A_2}(\sigma_{F_1 \vee F_2}(r))$

⇒ selection operation is commutative and we can cascade it

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

⇒ Now projection operation ⇒ $\boxed{\Pi_A(\Pi_{(B)}(r)) \text{ if } A \subset B \text{ then we write } \Pi_A(r)}$

Here Given $A_1 \subset A_2$ $\Pi_{A_1}(\Pi_{A_2}(\sigma_{F_1 \wedge F_2}(r)))$

$= \Pi_{A_1}(\sigma_{F_1 \wedge F_2}(r))$

= Option a.

18. GATE 04 QUE

Consider the Relati
is shown underli
atleast one boy an
expression produ

$\Pi_{name}(\sigma_{Sex}$

a) Names of girls
b) Names of girls
c) Names of gir
d) Names of gir

Sol Let $A = \Pi_n$

$B = \Pi_n$

Name	Se

Now $A - B =$

⇒

18. GATE 04 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).

$$\pi_{name}\left(\sigma_{sex=female}(Student)\right) - \pi_{name}\left(Student \underset{\substack{sex=F \wedge x=male \wedge \\ marks \le m}}{\bowtie} \rho_{(n,x,m)}(student)\right)$$

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.

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

$B = \pi_{name}\left(Student \underset{\substack{sex=F \wedge \\ x=male \wedge \\ marks \le m}}{\bowtie} \rho_{(n,x,m)}(student)\right)$

Name	Sex	Marks

$\bowtie$

n	x	m

⇒ 1st gets the girls and then select the boys and display girls name whose marks are < boys.

Now A − B = {Names of all girls} − {Names of girls whose marks are < all boys}

⇒ {Names of all girls whose marks are more than all the boy students}

- OPTION D

19. GATE 07 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 (Stud Id, 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.

$$\pi_{courseid}\left(\left(\pi_{studid}\left(\sigma_{sex=\text{"female"}}(StudInfo)\right) \times \pi_{courseid}(enroll)\right) - enroll\right)$$

a) courses in which all the female students are enrolled

b) courses in which a proper subset of female students are enrolled

c) Courses in which only male students are enrolled

d) None of the above

Std Info

StdId	Name	Sex
508	Ammajan	F
507	Chavana	F
509	Pavan	M
504	Sujith	M

Enroll

StdId	CourseId
508	DBMS
507	CN
509	TOC
504	CD

Now, $\pi_{studid}\left(\sigma_{sex=female}(StudInfo)\right) = A =$

StdId
508
507

Now, $B = \pi_{courseid}(enroll) =$

CourseId
DBMS
CN
TOC
CD

Now, $A \times B =$

Std id	Course Id
508	DBMS
508	CN
508	TOC
508	CD
507	DBMS
507	CN
507	TOC
507	CD

Now, $\pi_{courseid} =$

CourseId	
DBMS	Subset of female
CN	
TOC	male Registered
CD	

∴ OPTION B

20. GATE 07 IT

Consider the fo

b- schema =

a- schema =

d- schema =

Let Branch, acc

schema. Assum

bigger than th

Consider the f

which one of th

above Query?

a) $\pi_{c\text{-}name}(\sigma_{Bal<c}$

b) $\pi_{c\text{-}name}(\sigma_{b\text{-}cit}$

c) $\pi_{c\text{-}name}((\sigma_{b\text{-}cit}$

d) $\pi_{c\text{-}name}(\sigma_{b\text{-}cit}$

⇒ If you perform

it wont be

no of Tuples

⇒ so first appl

product fo

∴ 1st selec

2nd sele

Now, Co

Now, c

$\pi_{c\text{-}name}(\sigma$ be

20. GATE QT 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. $\Pi_{c\text{-}name}(\sigma_{b\text{-}city=\text{"Agra"} \wedge bal<0}(\text{branch} \bowtie (\text{account} \bowtie \text{depositor}))$

Which one of the following Queries is the most efficient version of above Query?

a) $\Pi_{c\text{-}name}(\sigma_{bal<0}(\sigma_{b\text{-}city=\text{"Agra"}}(\text{branch}) \bowtie (\text{account})) \bowtie \text{depositor})$

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

c) $\Pi_{c\text{-}name}((\sigma_{b\text{-}city=\text{"Agra"}}(\text{branch}) \bowtie \sigma_{bcity=\text{"Agra"} \wedge bal<0}(\text{account}) \bowtie \text{depositor})$ → not present in acc table so x.

d) $\Pi_{c\text{-}name}(\sigma_{b\text{-}city=\text{"Agra"}}(\text{branch}) \bowtie \sigma_{bcity=\text{"Agra"} \wedge bal<0}(\text{account}) \bowtie \text{depositor})$

⇒ If you perform cross product first, and then the selection operation it wont be efficient because cross product generates large / more no of Tuples.

⇒ So first apply the selection condition and then apply the cross product from the obtained two smaller tables.

∴ 1st select Agra city from b-schema

2nd select the tuples with bal<0 from a-schema.

Now, cross product the account with depositor

Now, cross product the Result with b-schema

$\Pi_{c\text{-}name}(\sigma_{bcity=\text{"Agra"}}(\text{branch}) \bowtie (\sigma_{bal<0}(\text{account}) \bowtie \text{depositor}))$

= option B.

21. GATE 2008 QUESTION ON NATURAL JOIN

Let R and S be two Relations with the following schema

$R(P, Q, R_1, R_2, R_3)$ and $S(P, Q, S_1, S_2)$ where $\{P, Q\}$ is the key for both schemas. which of the following queries are equivalent?

PQ is the key not P, Q alone

1. $\Pi_P(R \bowtie S)$ 2. $\Pi_P(R) \bowtie \Pi_P(S)$ 3. $\Pi_P(\Pi_{P,Q}(R) \cap \Pi_{P,Q}(S))$

4. $\Pi_P(\Pi_{P,Q}(R) - \Pi_{P,Q}(R) - \Pi_{P,Q}(S))$

a) only I and II b) only I and III c) only I, II and III d) only I, III, and IV

I.

R

P	Q	R1	R2	R3
1	a	-	-	-
2	a	-	-	-
3	a	-	-	-

$\bowtie$

S

P	Q
1	b
2	a
3	a

$\Rightarrow$

R ⋈ S

P	Q
2	a
3	a

$\Rightarrow$

P
2
3

II.

R

P
1
2
3

$\bowtie$

S

P
1
2
3

=

R ⋈ S

P
1
2
3

III.

R

P	Q
1	a
2	a
3	a

S

P	Q
1	b
2	a
3	a

=

P	Q
2	a
3	a

$\Rightarrow$

R ∩ S

P
2
3

IV.

Now, we know that $A \cap B = A - (A - B)$

Let $A = \Pi_{P,Q}(R)$ $B = \Pi_{P,Q}(S)$

=

P
2
3

∴ option I, III, IV are equivalent.

for both

Q is the key

ot P, Q alone

I, III, and IV

P
2
3

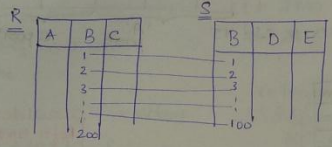
22. GATE 2010 QUESTION ON NATURAL JOIN ON PRIMARY KEY

- The following FD's hold for Relations R(ABC) S(BDE) $\{B \to A,\ A \to C\}$

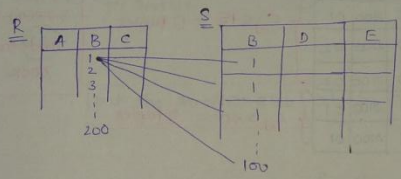
The relation 'R' contains 200 tuples and 'S' contains 100 tuples. what is the max no. of tuples possible in natural join of R, S i.e $R \bowtie S$?

a) 100 b) 200 d) 300 e) 2000

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



$\therefore$ The max. no. of tuples in $R \bowtie S = 100$



= 100 tuples.

23. GATE 12 QUESTION ON UNION AND NATURAL JOIN

consider the following relations A, B, C

A

ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

ID	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

ID	Phone	Area
10	2200	02
99	2100	01

How many tuples does the result of the following relational Algebra Expression contain? Assume that the schema of A∪B is the same as that of A. $(A \cup B) \bowtie_{A.id>40 \vee C.id<15} C$

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

$A \cup B =$

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

$\bowtie C$

C

ID	phone	Area
10	2200	02
99	2100	01

$\Rightarrow$ $(A \cup B) \bowtie C$

ID	Name	Age	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	Hari	40	10	2200	02
25	Hari	40	99	2100	01
98	Rohit	20	10	2200	02
98	Rohit	20	99	2100	01

'A' part | C part

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	Hari	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 tuples

$A_{id} > 40$ = 2 tuples

= 7 tuples

Refer 547 page 3.25(a) in made easy book

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) < size s(S)), the join will have fewer no of disk blocks access if

a) relation r(R) is in the outerloop

b) relation s(S) is in the outerloop

c) join selection factor between r(R) and s(S) is more than 0.5

d) join selection factor between r(R) and s(S) is less than 0.5.

The nested loop

we are going t

r(R)

↓

20 tuples

↓

2 Blocks.

↓

In this case the

No. of Block acces

are

$1 + 10 \times 10$ - for

$1 + 10 \times 10$ - for

202

∴

25. GATE 201

in the depe

depen dent (

relational

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. part

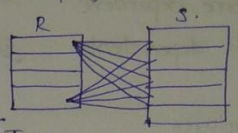
Refer 547 page 3.25(a) in 'made easy book

The nested loop method means for every tuple in one relation we are going to take other entire table and we are joining them.

r(R) → 20 tuples → 2 Blocks.

s(S) → 100 tuples → 10 Blocks

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



In this case the No. of Block acess are

1 + 10 × 10 — for 1st block
1 + 10 × 10 — for 2nd blk
202

1 + 2 * 10 — for 1st blk
1 + 2 * 10 — for 2nd blk
⋮
10 times
1 + 2 * 10 → for 10th blk
210 Block Acess

{Take every row of R and combine with 'S'}

∴ put the smaller size table in outer loop.

→ How did we write this is for the 1st block, we have to Acess it so (1+) and for every block we have to link with all the 10 blocks in 'S' (so 1 + 10 * 10)

1 → 1st block
10 → Acessing all the blocks in 'S' = 10 blocks and size of each block = 10 tuples = 10 * 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.

π_{empid} (employee) − π_{empid} (employee $\bowtie_{(empId = eId) \wedge (empAge \le depAge)}$ (dependent))

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

a) some dependent
b) All dependents.
c) some his/her 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, π_{empid} (employee $\bowtie_{(empId = eId) \wedge (EmpAge <= depAge)}$ (dependent)) ⇒ It returns the empids of employees whose Age is less than or equal to their dependents.

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

⇒ Empid's of Employees whose age is greater than his/her dependents.

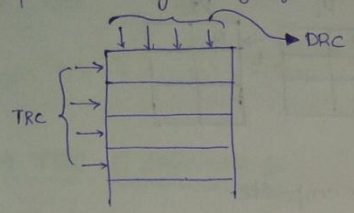
∴ Option D.

26. INTRODUCTION TO RELATIONAL CALUCULUS

On the Relational model two formal languages are proposed they are

1) Relational Algebra
2) Relational caluculus
   - Tuple Relation caluculus (TRC)
   - Domain Relational caluculus. (DRC)

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

$$\{t / \underbrace{cond(t)}_{\text{Condition}}\}$$

Tuple variable

Ex:-

Student

FN	LN	Marks

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

$$\{t / \underbrace{student(t)}_{\text{Name of the table that the tuple variable should Range}} \text{ AND } t.marks > 50\}$$

Tuple variable to range on table

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

↳ Gives FN of students whoose marks > 50.

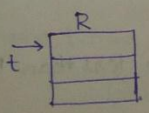
The general form is

$$\{t_1.A_j, t_2.A_k, \ldots t_n.A_m / cond(t_1 \ldots t_n \ldots t_{n+m})\}$$

28. FREE AND BOUNDED VARIABLES

Generally we use atomic expression while representing conditions in the TRC. The basic Atomic expressions will be

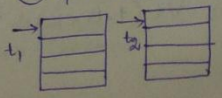
<1> R(t)



<2> $t.A \ \theta$ <constant>

$t.marks > 50$.

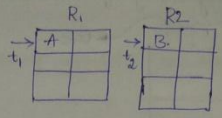
③ $t_1.A < t_2.B$



Now,

Composite Expressions

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



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 $\left\{ \begin{matrix} F_1 \\ F_2 \end{matrix} \right\}$ = 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

⇒ Free variables appear only on the leftside of bar. in TRC Representation

In the Right side Bounded variables occur.

TRC = { ____ / ____ }

Free variables → ____ ; Bounded variables → ____

Now,

R

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

R(t)

$\exists t\ (t.A > 50)$ ⇒ If i specify this condition then it returns false because the table doesnt contains the tupules satisfying the condition

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

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

Now, $\sim \exists t\ (t.$

$t.C < 10$

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

29. TRC EXA

List the nam

"Research" dep

Employee (Fnam

Department (Dn

Dept_location

Project (pna

works-on (Es

Dependent (

Sol

Emp

t →

⇒ Now, after c

variables f

should be

should be

variable po

⇒ The cross p

TRC

{t.Fname, t

Now, $\boxed{\sim \exists t (t.c < 10)}$ ⇒ There doesnot exist atleast one value of 't' such that t.c<10 ∴ TRUE

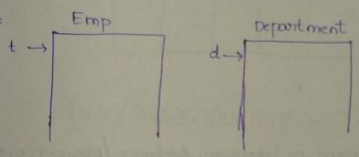
Now, $\forall t (t.c \geq 10)$ = TRUE (All values are greater than 10).

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, Dno)
Department (Dname, Dnumber, Mgr-ssn, Mgr-startdate)
Dept_locations (Dnumber, Dlocation)
Project (pname, pnumber, plocation)
works-on (Essn, pno, Hours)
Dependent (Essn, Dependent-name, sex, Bdate, Relationship)

Sol



⇒ The only way to combine two tables in TRC is by taking 2 attributes.

⇒ 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, [name, Address] are the values that should be printed and they are present in Emp table ⇒ free the variable pointing to Employee table ⇒ free the variable 't'.

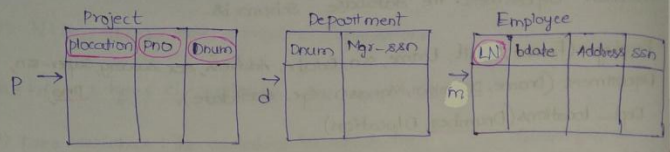
⇒ The cross product in Relational Algebra is equal to ∃ in the TRC

{t.Fname, t.minit, t.lastname, t.address / Employee (t) AND
(∃d) (Department (d) AND d.Dname = "Research"
AND d.Dnumber = t.DNO }

30. TRC EXAMPLE 2

For every project located in "stafford" list the project number, the controlling department number, and the department manager's 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".



In Relational Algebra

$$\sigma\ (p \times D \times E)$$

p. plocation = "stafford" ∧

p. Dnum = D. Dnum

D. mgrssn = E. ssn

In, TRC

{p.pnum, p.dnum, m.Lname, m.bdate, m.Address / PROJECT(p) AND

EMPLOYEE(m) AND p.location = "stafford" AND

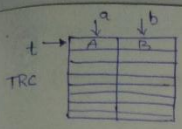
((∃d)(DEPARTMENT(d) AND p.Dnum

= d.Dnum AND d.mgrssn = m.ssn }

38. DOMAIN RELATIONAL CALUCULUS INTRODUCTION

The main difference between the TRC and DRC is the way in which we use the variables

⇒ In case of Domain Relational Calucules we need more variables than the TRC



t.A = 10 in TRC
a = 10 in DRC

⇒ SQL = based on (TRC and RA)

⇒ IBM = 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 Birthdate, 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	Super_ssn	Dno

$$\{u, v / (\exists q)(\exists r)(\exists s)(\exists t)(\exists w)(\exists x)(\exists y)(\exists z)$$

$$(\text{EMPLOYEE}(qrstuvwxyz) \text{ AND}$$

$$q = \text{'JOHN'} \text{ AND } r = \text{'B'} \text{ AND } s = \text{'SMITH'})\}$$

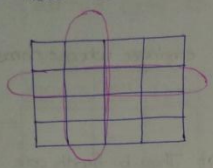
TRC- DRC- NOT much needed for GATE only
Simple Questions will be asked

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 'K'

2. CREATING A TABLE AND CONSTRAINS ON IT

```
CREATE TABLE Employee
( NAME    VARCHAR(15)  NOTNULL,
  SSN     CHAR(9)      NOT NULL,
  Bdate   DATE
  Super-ssn   CHAR(9)
   PRIMARY KEY(SSN)
   FOREIGN KEY (Super-ssn) REFERENCES EMPLOYEE (ssn);
);
```

primary key allows NOTNULL by default

Foreign key allows NULL by default.

The various datatypes that are used is NUMERIC, CHAR(n), VARCHAR(n)

INT SMALLINT FLOAT REAL

Concatenation operator abc || xyz = abcxyz

Now, DNO INT NOTNULL CHECK (DNO>0 AND DNO<20) ⇒ checking the variables within Range

write is

ee in result

the attribute

= 10.

(Employee)
<ssn=10>

select:

work for

5. ALIASING

for each employee, retreive 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, DNO)
Employee → Supervisor

Department (Dname, Dnumber, Mgr-ssn, Mgr-start date)

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

ESSN	FN	LN	S SSN
1	A	B	10

×

S

SSSN	FN	LN	ESSN
10	C	D	

(we are doing cross product on the same table so Rename it as S.

```
SELECT  E.FN, E.LN, S.Fname, S.Lname

FROM    EMPLOYEE AS E, EMPLOYEE AS S

WHERE   E.ESSSN = S.SSSN ;
```

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 tuples whose DNO=4/10
No Repetetions.

UNION - Eliminates duplicates ← SELECT DISTINCT

UNIONALL - Doesnot Eliminates duplicates ← SELECT ALL

R
a1
a2
a3

S
a1
a3
a5

R UNION S

a1
a2
a3
a5

R UNIONALL S

a1
a2
a3
a1
a3
a5

7. PATTERN MATCHING AND ARITHMETIC OPERATORS

The 'LIKE' is used for string comparing / comparing.

Now, if i want to find all the Employees whose name contains "Ravi"

⇒ The 2 operators used are

% = no.of '0' or more characters

'_' = Single character

```
SELECT Fname
FROM Employee
WHERE Fname LIKE '%Ravi%'
```

% Ravi % ⇒ Before Ravi there can be any no.of characters

→ After Ravi there can be any no.of characters.

⇒ Now if i want employees whose third letter in Fname is 'V' then

⇒ Fname LIKE '__V%' → After that there can be anything

↓ 1st charecter → 2nd charecter → 3rd charecter

⇒ Fname || Lname ⇒ Outputs Concatenated Names.

⇒ Now, if i want to increase the salary of all the employees by 10% then

```
{ SELECT  Fname, 1.1 * SALARY }
{ FROM    EMPLOYEE );         }
```

BETWEEN ⇒

```
SELECT *
FROM Employee
WHERE (SALARY BETWEEN 10,000 AND 20,000);
```

OR $10,000 \le SALARY \le 20,000$.

8. ORDER BY

Retreive 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 ⇒ should be

A	a	b
A	b	c
B	a	c
B	c	b
C	c	d

→ First order by dept. name, if they are same then order by Lastname and then by Firstname.

⇒ ORDER BY ⇒ used to order the output

⇒ "OrderBy" clause can be applied on the attributes appearing in Select Query.

SELECT D.Dname, E.Lname, E.Fname, P.project name

FROM Employee E, Works on W, project P, Department D

WHERE D.Dnumber = E.Dno AND E.ssn = W.ssn AND W.PNO = P.pnumber

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

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

⇒ select ABC FROM R ORDER BY A,B,C

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

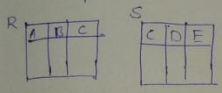
10. INSERT

The command used to insert a tuple in the relation is INSERT

→ INSERT INTO EMPLOYEE VALUES ('Ravi', 'Ravula', '1234', '4');

→ INSERT INTO EMPLOYEE ('Name', 'LN', 'SSN', 'DNO') VALUES ('Ravi', 'Ravula', '1234', '4');

⇒ Now, if i want to insert the values in a table in which are the values present in another table then



INSERT INTO R(A,B,C) SELECT C,D,E FROM S(C,D,E) WHERE

11. DELETE AND UPDATE

Now, if i want to delete all the employeenames whoose lastname is 'Ravi' then,

⇒ DELETE FROM EMPLOYEE WHERE LN = 'Ravi'.

⇒ DROP TABLE EMPLOYEE

⇒ UPDATE EMPLOYEE SET $\underbrace{\text{salary} = \text{salary} * 1.1}_{\text{updates the salaries of all Employee whoose DNO=5}}$, WHERE DNO=5;

12. DEALING WITH NULL VALUES

⇒ The problem with the NULL values is we dont 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 $(x \wedge y)$

AND	T	F	UK
T	T	F	UK
F	F	F	F
UK	UK	F	UK

OR	T	F	UK
T	T	T	T
F	T	F	UK
UK	T	UK	UK

NOT	A
T	F
F	T
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 Fnames, Addresses of employees who work for dept = {2,3,4,5}

```
SELECT   Fname, Address
FROM     EMPLOYEE
WHERE    Dno IN (1,2,3,4);
```

(or)

DNO= 1 V DNO=2 V DNO=3 V DNO=4

Find Fname, Address of employees who work for department location in "Stafford"

```
SELECT   Fname, Address
FROM     EMPLOYEE
WHERE    DNO IN ( SELECT DNO
                  FROM   DEPT_LOCATIONS
                  WHERE  DLOCATION='Stafford');
```

Fname	Add	DNO
A	B	1
C	D	2
E	F	3

DNO	DLocat
1	Stafford
2	Stafford
3	Stafford

14. ANY, ALL SOME

```
SELECT   DISTINCT Essn
FROM     WORKS-ON
WHERE    (Pno, Hours) IN (SELECT Pno, HOURS FROM WORKS-ON
                          WHERE Essn='10');
```

E	P	H
1	10	10
2	20	10
1	30	10
10	1	10
10	20	10
10	30	10

⇒ The Inner Query will give,

(1, 10)
(20, 10)
(30, 10)

⇒ The outer Query produces (2)

↳ 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.

ON SOME

ON ANY.

Find FName of all employee whoose 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 CORELATED QUERIES

RETRIEVE the fname of each employee who has a dependent with the same fname and is the same sex as the employee

```
SELECTED (E.Fname)
FROM EMPLOYEE AS E
WHERE  E.SSn IN (SELECT  FROM DEPENDET 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

Fname	SSN	Sex
A	501	F
B	502	m
C	503	m

Dependent

Dependent Name	Sex	Essn
A	F	501
B	m	502
F	m	509

Inner Query produces

Essn
501
502

Outer Query produces

A
B

16. EXISTS AND NOT EXISTS

```
SELECT F.name
FROM   EMPLOYEE AS E
WHERE  EXISTS (SELECT *
               FROM   DEPENDENT AS D
               WHERE  E.ssn = D.Essn AND
                      E.sex = D.sex AND
                      E.Fname = D.Dependent name);
```

=> If the Internal Query returns some value the EXISTS will return TRUE and if the internal Query does not return anything the Exists will return FALSE.

Retreive the Fnames of employees who have no dependents.

```
SELECT Fname
FROM   EMPLOYEE
WHERE  NOT EXISTS (SELECT * FROM DEPENDENT WHERE SSn = Essn);
```

→ If this Query returns something the NOT EXISTS Return FALSE

→ If this Query returns nothing then NOT EXIST Returns TRUE.

17. EXISTS EXAMPLE 1

List the Fnames of managers who have atleast one dependent

```
SELECT Fname
 FROM  Employee
  WHERE EXISTS (SELECT * FROM DEPARTMENT WHERE ssn=Essn).
   AND
    EXISTS (SELECT * FROM DEPARTMENT WHERE ssn=Mgr-ssn);
```

```
SELECT Fname
FROM EMPLOYEE
WHERE EXISTS (SELECT * FROM DEPENDENT WHERE ssn=Essn)
AND EXISTS (SELECT * FROM DEPARTMENT WHERE ssn=Mgr-ssn);
```

Emp

FN	SSN

Dependent

Essn	

Department

mssn	

```
for(i=1 to n)
{
  for (j=1 to n)
   for k=1 to n
}
```

18. EXISTS EXAMPLE 2

Retreive the fname of each employee who works on all the projects controlled by department number 5.

```
SELECT Fname
FROM Employee
WHERE NOT EXISTS (SELECT Pnumber
                  FROM PROJECT
                  WHERE Dnum=5) EXCEPT (SELECT Pno FROM
                  WORKS-ON WHERE ssn=Essn));
```

RETURNS TRUE

Emp

SSN	FN
1	Ravi
2	
3	
4	

Project

Pnum	Dnum
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

```
SELECT  Fname, LName, Address  FROM(EMPLOYEE JOIN DEPARTMENT
                                        ON DNO= DNumber)

WHERE  Dname = 'Research';


SELECT  Fname , Lname, Address  FROM (EMPLOYEE NATURAL JOIN DEPARTMENT
                AS DEPT (Dname, Dno, Msn, Msdate))
WHERE   Dname = 'Research');


SELECT  E.Lname  AS EMPLOYEE_name, S.LName AS Supervisor_name
        FROM (EMPLOYEE AS E  LEFT OUTER JOIN
                EMPLOYEE  AS ON E.super_ssn = S.ssn);
```

R

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S → for Natural join

A	D
1	g
2	h
3	i
7	j
8	K
9	l

Now, R⋈S (Join A and C) A=C

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i

R*S (Natural Join)

A	B	D
1	a	g
2	b	h
3	c	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

20. AGGRE

The Aggre

Employee

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

⇒ while c

3) SELECT (

4) SELECT

⇒ min, max

⇒

⇒ Select su

⇒ Avg, Sum

data t

⇒ SELECT

⇒ Select

⇒ Select c

⇒ select su

20. AGGREGATE FUNCTIONS 1

The Aggregate functions are Avg, min, max, sum, count

Employee

Eno	Ename	Salary	HA
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 considerd

3) SELECT (max(salary) − min(salary)) as diff from Employee :

Diff
3

4) SELECT COUNT (*) as Total FROM Employee :

Total
6

{No of rows}

⇒ min, max are applied on Numbers, strings, Dates

⇒ SELECT max(Name) FROM EMPLOYEE :

Name
F

⇒ Select sum (name) FRom Employee ✗ Not valid

⇒ Avg, Sum are applied only on Numerical Data not on any other data types.

⇒ SELECT COUNT (salary) FROM Employee

5

(1,2,4,4,4)

⇒ Select COUNT (salary) FROM Employee

3

(1,2,4)

distinct

⇒ Select COUNT (salary + HA) FROM Employee

4

(1,2) = 3
(2,4) = 6
(4,6) = 10
(4,8) = 12
(NULL) = NULL
(NULL) = NULL

⇒ select sum(salary + HA) FROM Employee

31

(3+6+10+12)

21. AGGREGATE FUNCTIONS 2

COUNT (salary) = 5

COUNT (DISTINCT salary) = 3

SUM(salary) = 15

SUM(DISTINCT SALARY) = 7

$$Avg(salary) = \frac{SUM(salary)}{COUNT(salary)} = \frac{15}{5} = 3$$

$$Avg(DISTINCT\ salary) = \frac{SUM(DISTINCT\ salary)}{COUNT(DISTINCT\ salary)} = 7/3$$

min (salary) = 1

min (DISTINCT SALARY) = 1 = min(1, 4, 4)

max(salary) = 4

max(DISTINCT SALARY) = 4

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	3

→ No of 1's

→ No. of 2's.

{whenever you have some attributes in Group By clause then they should always appear in select clause}

4> For each department, Retsetve the dno, the num of employees in the department and their average salary

SELECT DNO, count (*), Avg (salary),

FROM EMPLOYEE

GROUP BY DNO

1	100	10K
2	200	8K

⇒ Every NULL value will be treated as a seperate Group. (V.V. Imp)

1. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

Transaction:

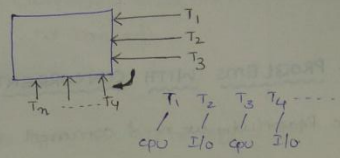
A Transaction is a collection of operations that forms a single logical unit of work.

Transfering money

R(A); — I/O
A = A - 50; — cpu
W(A); — I/O
R(B);
B = B + 50
W(B);

Sequence of steps

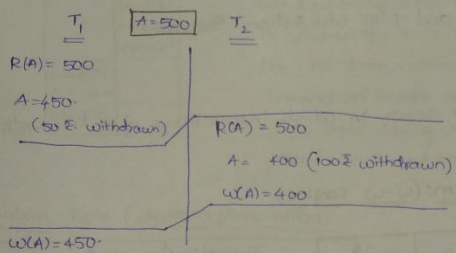
Bank Database System



Let us consider a problem of Transactions.

Let T_1 Represent transfering money from A to B

T_2 Represent withdrawing money from A



Transaction Properties (ACID PROPERTIES)

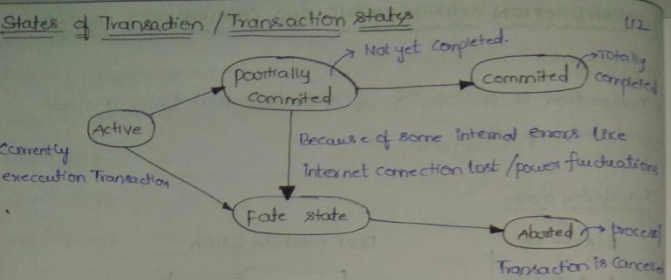
1) Atomicity : All or None should happen → Transaction Manager

2) Consistency: Correctness → user/Application programmer takes care

3) Isolation: Each Transaction must be executed without knowing what is happening with others - Concurrency Controll manager

4) Durability : All updations done by a Transaction must become permanent. - Recovery Manager takes care

States of Transaction / Transaction states



2. PROBLEMS WITH CONCURRENT EXECUTION

The Reason why we need concurrent execution is

1> For Avoiding long waiting time

2> If 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.

3> 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)

$A \xrightarrow{50} B$	$A \xrightarrow{4\%} A$
Read(A)	
A: A-50	
	Read(A) → A=100
	X → 0.04*A → A = 0.04*100
	A = A+X → A=104
write(A)	
	write(A) → A=104
Read(B)	
B = B+50	
write(B)	

A=100
A=50

A=104

Initial Value of A = 100

But the Actual ans should be

A = 100

A-50 = 50

$A = 50 \times 4\%_{100} = 2$

A = A+X = 50+2 = 52

A=52

Test: Two write operations of diff Transactions and in btn there is no Read then 2nd write operation overwrites 1st write operation

Dirty Read problem

T_1
Read(A)
A = A-50
write(A)
Read(B)
B = B+50
write(B)

A=100
A=50
A=50

Roll Back

Fail
↓
Because of Roll back
⇒ A=100 but the n
not considered

Un Repeatable Read

T_1
Read(A)
Read(A)
A = A-10,000
write(A)

Fail

Phantom Tuple (

T_1		
eno	ename	sal
1	A	50
3	C	40

Select * from Emp where salary > 3000

eno	ename	sal
1	A	5000
3	C	4000
4	D	3500

Select from Em where sal > 30

Dirty Read problem (W-R conflict) (Write-Read conflict)

	T_1	T_2	
A=100	Read(A)		
A=50	A=A-50		
A=50	write(A)		
		Read(A)	A=50
		x=0.04*A	x=2
		A=A+x	A=52
		write(A)	A=52 (Temporarily saved, this transaction is not comited)
Fail	Read(B)		
	B=B+50		
	write(B)		

Roll Back

↓ Because of Roll back

⇒ A=100 but the modifications done by T_2 will not be saved and not considered

Un Repeatable Read problem (or) Non-Repeatable Read Problem (R-W)

T_1	T_2
Read(A)	
	Read(A)
	A=A-15000
	write(A)
Read(A)	
A=A-10,000	
write(A)	

Fail

⇒ when a transaction tries to Read the value of dataitem twice and another transaction updates the same data item in between the two Read Operations of the 1st transaction, as a result the 1st Transaction Reads varied values of same data item during its execution, this is called un-repeatable Reads.

Phantom Tuple (phantom phenomenon)

T_1	T_2
eno ename salary 1 A 5000 3 C 4000 Select * from Emp where salary > 3000	
	Insert into Emp values (4, O, 3500);
sno ename sal 1 A 5000 3 C 4000 4 O 3500	select * from Emp where sal > 3000

Incorrect summary problem

T1	T2	
	Sum = 0	→ sum = 0
	Read(K)	→ Let K = 50
	sum = sum+K;	→ sum = 50
X = 100 ← Read(x)		
X = 600 ← X = X+500		
X = 600 ← write(x)		
	Read(x)	— x = 600
	sum = sum+x;	— 50+600 = 650
	Read(y)	— Let y = 200
	sum = sum+y;	— sum = 850 wrong summary.
Read(y)		
y = y+200		
write(y)		

→ Because of this operation 'y' will be changed and and as a result sum must be changed but if we retreive sum it gives 850.

Before T_1	After T_1
K = 50	50
X = 100	600
Y = 200	400
850	1050
=	=
→ Wrong	→ Actual sum

3. NO. OF SCHEDULES POSSIBLE

Let us say there are two operations associated with a particular Transaction and there are two transactions T_1 and T_2. Now, what are the diff ways to combine these two together.

T_1	T_2
a1	b1
a2	b2

S_1	S_2	S_3	S_4	S_5	S_6
a1	b1	a1	b1	a1	b1
a2	b2	b1	a1	b1	a1
b1	a1	b2	a2	a2	b2
b2	a2	a2	b2	b2	a2

⇒ In these schedules the order should be consistent which means a_1 should be executed only before a_2 and a_2 should not be executed before a_1 ....

⇒ All these schedules will not give us correct results (not consistent) so we need to find the correct schedule that Guarantee the consistency of database.

4. TYPES OF

Serial schedul

Transactions T_1

$T_1 \rightarrow T_2$

T1	T2
R(A)	
A=A+50	
w(A)	
	R(A)
	A=A+100
	w(A)

constan operatio

⇒ serial sched operations of

⇒ when Transac state

⇒ If there are

Complete sched

⇒ At the end present then

T1	T2
R(A)	
	R(A)
A=A-50	
w(A)	
Commit	
	A=A+50
	w(A)
	Abort

Recoverable sc

Non-

T1
R(A)
A=A-50
w(A)
Roll Back
Fail
R(B)
B=B+50
w(B)
Commit

4. TYPES OF SCHEDULES

Serial schedule : Serial schedule means one after the other. If i have two Transactions T_1 and T_2 then $T_1 T_2$ and $T_2 T_1$ 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+100
	w(A)
R(A)	
A=A+50	
w(A)	

constant states of operation

T_1	T_2
R(A)	
A=A+50	
	R(A)
	A=A+10
w(A)	
	w(A)

⇒ Not serial schedule

⇒ serial schedules are the ones that does not 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)	
	R(A)
A=A-50	
w(A)	
Commit	
	A=A+50
	w(A)
	Abort

Recoverable schedule

Non-

T_1	T_2
R(A)	
A=A-50	
w(A)	
	R(A)
	x=A*100
	A=A+x
	w(A)
	Commit
R(B)	
B=B+50	
w(B)	
Commit	

Roll Back

Roll back is not possible because it is committed

Fail

Fail

Non- Recoverable schedule.

Recoverable schedule

T_1	T_2
R(A)	
A=A-50	
w(A)	
	R(A)
	x=A*0.01A
	A=A+x
	w(A)
R(B)	
B=B+50	
w(B)	
Commit	
	Commit

Roll Back

Roll Back

Fail

CASCADING SCHEDULE

T1	T2	T3
R(A) W(A)		
	R(A) W(A)	
		R(A) W(A)
Commit	Commit	Commit

Roll Back

Roll Back

Fail

⇒ CASCADING ABORT

CASCADELESS SCHEDULE

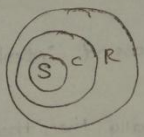
Every cascadeless is Recoverable

T1	T2	T3
R(A) W(A) Comit		
	R(A) W(A) commit	
		R(A) W(A) Commit

STRICT SCHEDULE

T1	T2
R(A) W(A) Comits	
	W(A)

↳ I wont Read/write untill the other one Commits



S= STRICT SCHEDULE

C= CASCADELESS SCHEDULE

R= RECOVERABLE SCHEDULE

of database.

Now, let us say there are Transactions $T_1\ T_2\ T_3 \cdots T_m$ and no. of operations in each transaction is $n_1, n_2\ n_3 \cdots n_m$ then the No. of schedules that are possible are $\dfrac{(n_1+n_2+n_3+\cdots+n_m)!}{(n_1!)(n_2!)(n_3!)\cdots(n_m!)}$

$$\boxed{\text{No. of schedules possible} = \frac{(n_1+n_2+n_3+\cdots+n_m)!}{(n_1!)(n_2!)(n_3!)\cdots(n_m!)}}$$

⇒ 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 $\boxed{m!}$

In the above Example the no. of serial schedules possible $= 2! = 2$

⇒ $\dfrac{T_1}{T_2}$ / $\dfrac{T_2}{T_1}$ ⇒ a_1 a_2 b_1 b_2 (T_2) ; b_1 b_2 a_1 a_2 (T_1)

⇒ All serial schedules are always consistent.

4. RESULT EQUIVALENT SCHEDULE

Two schedules are said to be Equivalent if they follow some 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	
A=100	R(A)	R(A)	A=100
A=110	A+10	A*1.1	A=110
A=110	W(A)	W(A)	A=110

⇒ check whether these two schedules are Equivalent / not

S1

T1	T2
R(X) X=X+5 W(X)	
	R(X) X=X*3 W(X)
R(Y) Y=Y+5 W(Y)	

X	Y	X	Y
2	5	2	5
21	10	6	10
		11	10
X=21		X=11	
Y=10		Y=10	

NOT Result Equivalent

S2

T1	T2
	R(X) X=X*3 W(X)
R(X) X=X+5 W(X)	
R(Y) Y=Y+5 W(Y)	

6. CONFLICT EQUIVALENT AND CONFLICT SERIALIZABLE

Conflict operations

T_i	T_j	
R(A)	W(A)	Conflict operations
W(A)	R(A)	
W(A)	W(A)	
R(A)	R(A)	Non-Conflict operations
Operating on diff data		

⇒ Both the transactions are accessing the same data item and one of the operation is write then it is a conflict operation

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(A)	
	R(A) W(A)
R(B) W(B)	

W-R, W-W

Conflicts. (R-W)

T1	T2
R(A) W(A)	
	R(A)
R(B)	
	W(A)
W(B)	

W-R, R-W, W-W

∴ These 2 are Conflict

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(A)	
	R(B)
w(A)	
	w(B)

No - conflicting actions

S2

T1	T2
	R(B)
R(A)	
	w(B)
w(A)	

No-conflicting actions

$\Rightarrow S_1, S_2$ are NOT conflicting

2)

S1: $R_1(A)$ $w_1(A)$ $R_2(B)$ $w_2(B)$ $R_1(B)$

S2: $R_1(B)$ $w_1(A)$ $R_1(B)$ $R_2(B)$ $w_2(B)$

T1	T2
R(A)	
w(A)	
	R(B)
	w(B)
R(B)	

Conflicting operations

$w_2(B) \rightarrow R_1(B)$

WR

$\neq$

T1	T2
R(A)	
w(A)	
R(B)	
	R(B)
	w(B)

Conflicting operations

$R_1(B) \rightarrow w_2(B)$

RW

NOT Conflict Equivalent.

CONFLICT SERIALIZABLE

A schedule is said to be conflict serializable if it is conflict Equivalent to a serial schedule

S_1: $T_1 T_2$ (not serializable)

S_2: $T_1 T_2$ (conflict Equ)

S_2: $T_2 T_1$ $\Rightarrow$ Conf serializable

7. CONFLICT SERIALIZABLE EXAMPLE

Now, check whether these two schedules are conflict serializable or not.

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

(2)

S1

T1	T2
R(A) w(A) R(B)	
	R(A) w(A) R(B)
w(B)	
	w(B)

$R_1(A) \rightarrow w_2(A)$

$w_1(A) \rightarrow R_2(A)$

$w_1(A) \rightarrow w_2(A)$

$R_1(B) \rightarrow w_2(B)$

$w_1(B) \rightarrow w_2(B) \rightarrow R_2(B) \rightarrow w_1(B).$

S2

T1	T2
R(A) w(A) R(B) w(B)	
	R(A) w(A) R(B) w(B)

T1	T2
	R(A) w(A) R(B) w(B)
R(A) w(A) R(B) w(B)	

$\Rightarrow$ NO $R_1 \rightarrow w_2$

$R_1(A) \rightarrow w_2(A) \rightarrow w_1(A) \rightarrow R_2(A)$

$w_1(A) \rightarrow w_2(A)$

$R_2(B) \rightarrow w_2(B)$

$w_1(B) \rightarrow w_2(B)$

$w_1(B) \rightarrow R_2(B)$ NO $\boxed{R_2(B) \rightarrow w_1(B)}$

Not Conflict Equivalent.

$\therefore$ The Relation S1 is not Equalent to any of the Seriali sable schedule

$\therefore$ S_1 is not conflict serializable.

PROCEDURE F

(1) Construct a transaction operation. (

(2) If the Dire is not confli

(3) If the graph conflict seri

Ex

T1	T2
R(A) w(A)	
	R(A) w(A)
R(B) w(B)	
	R(B) w(B)

Ex :

T1	T2
R(A) w(A) R(B)	
	R(A) w(A) R(B)
w(B)	
	w(B)

Ex :

T1	T2
	R(
w(y)	
	R(
	w(

PROCEDURE FOR CONFLICT SERIALIZABILITY USING PRECEDENCE GRAPH

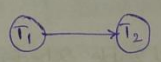
① Construct a directed graph where each vertex corresponds to a transaction and each directed edge represents a conflicting operation. (Read→write / write-write / write→Read)

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

T1	T2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

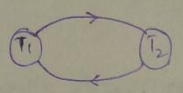


⇒ In the precedence Graph there is no cycle and therefore this schedule is conflict serializable

⇒ The serial schedule for which this schedule is equalent to is Topological order of the Graph = T_1T_2.

Ex:

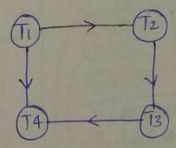
T1	T2
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(B)
w(B)	
	w(B)



⇒ In the precedence Graph there is a cycle and hence the schedule is not conflict serializable

Ex:

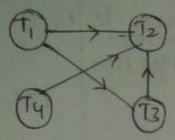
T1	T2	T3	T4
	R(X)		
		w(X)	
w(y)			
	R(y)		
	w(Z)		
			R(x)
			R(y)



There is no cycle and therefore the schedule is conflict serializable

⇒ The serial schedule to which the given schedule is $T_1T_2T_3T_4$ (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

⇒ No cycles in Graph ⇒ 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 conflict equalent 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 Equalent to the serial schedule $(T_1 \rightarrow T_2)$

Now, Among $R_1(A)$ $W_1(A)$ $R_1(B)$ $W_1(B)$
$R_2(A)$ $W_2(A)$ $R_2(B)$ $W_2(B)$ } $R_1(A)$ 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 $T_1 \rightarrow T_2$)

∴ $R_1(A)$ $W_1(A)$ $T_1 \rightarrow T_2$ and $R_2(B)$ $W_2(B)$ should come in end

∴

$R_1(A)$ $W_1(A)$	
	$W_1(B)$ $R_1(B)$ $R_2(A)$ $W_2(A)$ } 4! ways but R_1W_1 and R_2W_2 should be executed as group
	$R_2(B)$ $W_2(B)$

∴ No. of conflict serilazable schedules

$$= \frac{4!}{2!\,2!} = 6$$

Now $(T_2 \rightarrow T_1)$

	$R_2(A)$
	$W_2(A)$
	$R_2(B)$ $R_1(B)$
	$W_2(B)$ $W_1(B)$
$R_1(A)$	
$W_1(A)$	

$\Rightarrow \frac{4!}{2!\,2!}$ ways $= 6$

9. VIEW EQUIVALENT AND SERIALIZABLE

Two schedules 'S' and 'S'' are said to be view Equalent 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'.

⇒ 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'.

⇒ In summary All Read write sequences to be maintained in the same order in both S and S'

Ex:

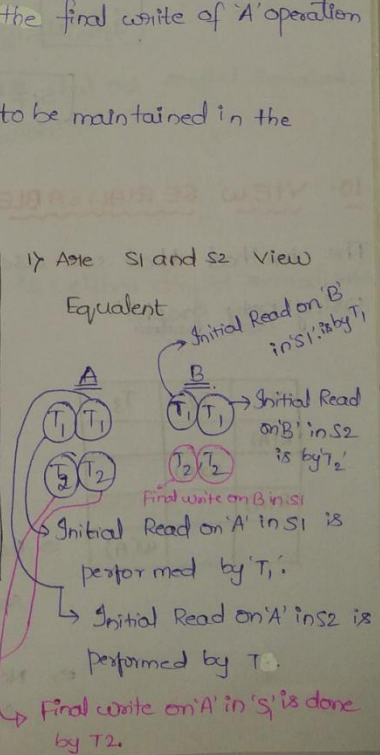
S1

T1	T2
R(A) W(A)	
	R(A) W(A)
R(B) W(B)	
	R(B) W(B)

S2

T1	T2
R(A) W(A) R(B) W(B)	
	R(A) W(A) R(B) W(B)

1) Are S1 and S2 View Equalent



Now, check for producers and consumers. (write and Read)

In S_1

S1

T1	T2
R(A) w(A)	
	R(A) w(A)
R(B) w(B)	
	R(B) w(B)

producer → R(B) w(B) → Consuming

In S_2

S2

T1	T2
R(A)	
w(A)	
R(B) w(B)	
	R(A) w(A) R(B) w(B)

producer → w(A) → R(A) w(A) R(B) w(B) → consume

∴ These two schedules is view Equivalent

Ex:

S1

T1	T2
R(A)	
	w(A)
w(A)	

Final write here is by T_1

S2

T1	T2
R(A) w(A)	
	w(A)

Final write here is by T_2

S3

T1	T2
	w(A)
R(A) w(A)	

producer

∴ $T_1 \neq T_2$ (Not View Equalent)

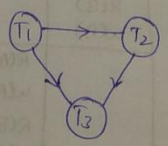
∴ $\boxed{S_1 \neq S_2}$ $\boxed{S_2 \neq S_3}$ (final write of 'A's are diff)

$\boxed{S_1 \neq S_3}$ (No producer consumer in S_1)

10. VIEW SERIALISABLE EXAMPLES

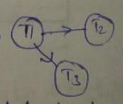
The method that we use to check view serializability is by using "poly graphs."

T1	T2	T3
R(A)		
	w(A)	
R(A)		
		w(A)

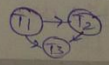


⇒ No cycle ⇒ The Schedule is View Serializable

⇒ Here T_1 should start first which means T_2 T_3 should be executed only after T_1 so



⇒ Now, comming to T_2 and T_3 T_2 should start first so



Ex

T1	T2
R(A) W(A)	
	R(A) W(A)
R(B) W(B)	
	R(B) W(B)

$T_1 \rightarrow T_2$

∴ The Graph contains No cycle and so the schedule is view serialisable and conflict serialisable

Ex

T1	T2
R(A)	
	W(A)
W(A)	

Says that T_1 should happen first and then T_2

This Says that T_1 should occur last which means T_2 and then T_1

∴ $T_1 \rightleftarrows T_2$ ⇒ Cycle ⇒ Not V.S.

⇒ If A schedule is Conflict serializable then it is view serializable but a view serializable schedule need not be conflict serialisable

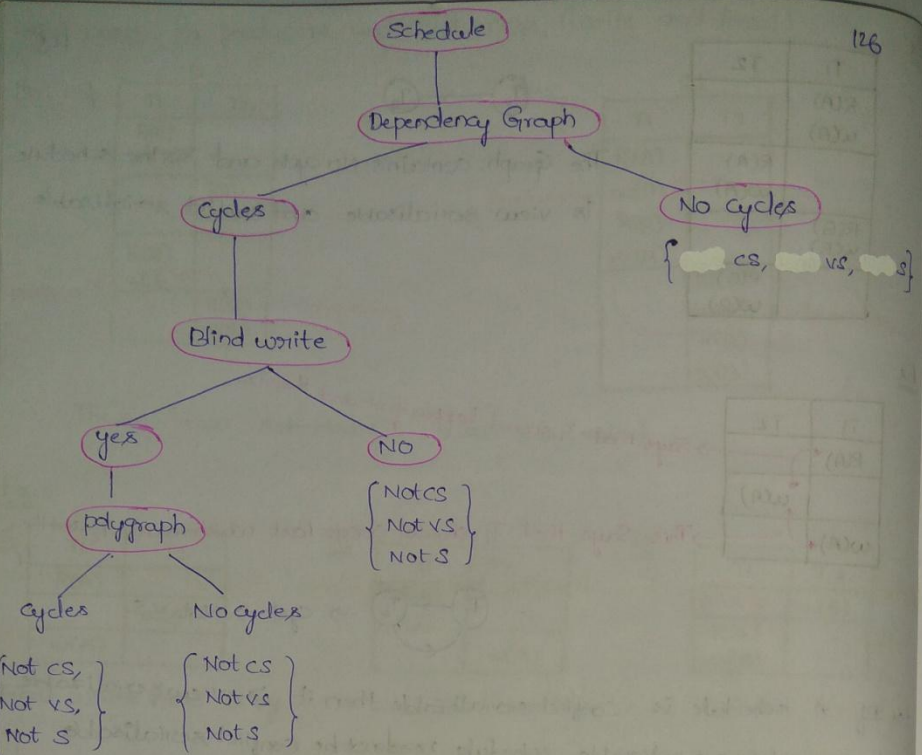
Blind write: Write without Read is called Blind write

⇒ If a schedule should be view serializable but not conflict Serialisable then there should be atleast one blind operation.

II. SERIALIZABLE SCHEDULES

⇒ A schedule is serializable if it is either conflict serializable or View serializable

⇒ Now, consider the following tree



How many concurrent schedules can be formed over two Transactions using two operations each?

T_1	T_2
$R_1(A)$	$R_2(A)$
$W_1(A)$	$W_2(A)$

No. of schedules possible $= \frac{(2+2)!}{2!\,2!}$ $\left[\frac{(n_1+n_2+\dots+n_m)!}{n_1!\,n_2!\dots n_m!}\right]$

$= \underline{\underline{6}}$

Non serial schedules = Total schedules − No. of serial schedules

$= 6 - (2!)$ (No. of Transaction)

$= 4$

⇒ The no. of concurrent schedules that can be formed over two transactions having n_1 and n_2 operations Respectively are

$$= \left[\frac{(n_1+n_2)!}{n_1!\,n_2!}\right]$$

The no. of serial schedules possible are $\left[\frac{(n_1+n_2)!}{n_1!\, n_2!}\right]-2$

⇒ The no. of concurrent schedules that can be formed over m-transactions having $n_1\, n_2 \cdots n_m$ operations respectively are

$$\left[\frac{(n_1+n_2+n_3+\cdots+n_m)!}{n_1!\, n_2!\, n_3! \cdots n_m!}\right]$$

⇒ The no. of serial schedules are $\left[\frac{(n_1+n_2+\cdots+n_m)!}{n_1!\, n_2!\, n_3! \cdots n_m!}\right]-(m!)$

Determine whether the following schedules are conflict serializable or not?

S_1: $R_1(A)\ W_1(A)\ R_2(B)\ W_2(B)\ R_1(B)\ W_1(B)$

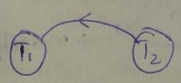
S_2: $R_1(A)\ R_1(B)\ W_2(A)\ R_3(A)\ W_1(B)\ W_3(A)\ R_2(B)\ W_2(B)$

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

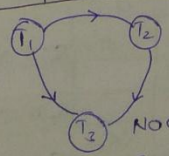
T1	T2
R(A)	
W(A)	
	R(B)
	W(B)
R(B)	
W(B)	



∴ conflict serializable

S_2

T1	T2	T3
R(A)		
R(B)		
	W(A)	
		R(A)
W(B)		
		W(A)
	R(B)	
	W2(B)	

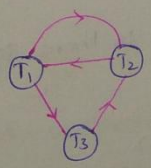


No cycles

∴ conflict serializable

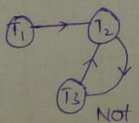
S_3

T1	T2	T3
R(A)		
	R(A)	
R1(B)		
	R(B)	
		R3(B)
W(A)		
	W(B)	



Not conflict serializable

S_4



Not conflict serialisable

12. EXAMPLES ON TYPES OF SCHEDULES

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?

⇒ No. of serial schedules = $m! = 2! = 2$ [m = No. of Transactions]
$(T_1 \to T_2)\ (T_1 \to T_2)$

⇒

Consider the following schedules

S_1: $R_2(x)\ R_2(y)\ R_1(x)\ R_1(y)\ W_1(x)\ R_2(x)$ ✓

S_2: $R_2(x)\ R_2(y)\ W_2(x)\ R_1(x)\ R_1(y)$

S_3: $R_2(x)\ R_2(x)\ R_3(x)\ R_2(y)\ W_2(x)\ W_2(x)$

which of the above schedules are having un-repeatable Read problem?

⇒ un Repeatable Read problem means if a transaction tries to read the same data two times and in btn this two reads if some other transaction changes the value then it is called un-Repeatable Read

⇒ Now S_1: $R_2(x)$ ----- $W_1(x)\ R_2(x)$

$x = 100$ $x =$ NOT 100

→ modified the value of x

which of the above schedules is having lost update problem.

⇒ Lost update problem means one transaction will write a value and immediately another transaction writes a value without Reading it (one writes ov

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 ∴ lost update problem

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_2: $R_2(x)$ ... $w_2(x)(R_1(x))$

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: $w_1(x)$ $R_2(x)$ = Dirty Read problem.

Which of the following schedules are Cascadeless, Recoverable and strict

S_1: $R_1(x)$ $R_2(x)$ $w_1(x)$ $w_2(x)$ Commit(C_2) Commit(C_1)

T1	T2
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 some one writes a value you should not Read/write it untill the previous one commits ∴ Not strict schedule

S_2: $R_1(x)$ $w_2(x)$ $w_1(x)$ $R_1(x)$ Commit(2) Commit 1

T1	T2
R(x)	
	w(x)
w(x)	
R1(x)	
	Comit
comit	

⇒ No producer consumer ⇒ cascadeless, Recoverable

⇒ Not strict schedule

S_3: $R_3(x)$ $R_1(x)$ $R_2(x)$ $w_1(y)$ $R_2(y)$ $w_2(\ )$ C_3 C_1 C_2

T1	T2	T3
		$R_3(x)$
$R_1(x)$		
	$R_2(x)$	
$w_1(y)$	Producer - consumer	
	$R_2(y)$	
	$R_2(x)$	
		Commit
comit		
	commit	

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

13. EXAMPLES ON CONFLICT SERIALIZABLES

· which of the following schedules is conflict serializable? For each serializable schedule, determine the equivalent serial schedules?

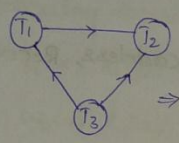
i) $r_1(x), r_3(x), w_1(x), r_2(x), w_2(x)$

ii) $r_1(x), r_3(x), w_3(x), w_1(x), r_2(x)$

iii) $r_3(x), r_2(x), w_3(x), r_1(x), w_1(x)$

iv) $r_3(x), r_2(x), r_1(x), w_3(x), w_1(x)$

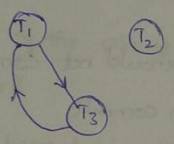
Now, i)



∴ The schedule is conflict serializable

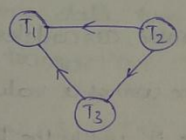
⇒ Start with the node that doesnt have any incomming edge = $T_3\ T_1\ T_2$

ii)



⇒ Not conflict serializable

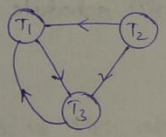
iii)



⇒ ∴ conflict serializable

⇒ serial schedule is = $T_2\ T_3\ T_1$

iv)



⇒ 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

T1	T2
R(x)	
	R(x) R(y) w(x) w(y)
R(x) R(y) w(x)	

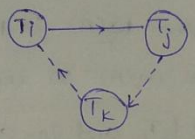
WW

T1	T2
R(x) R(y) w(x)	
	w(x) w(y) R(x) R(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 an before T_j so the polygraph will be



S : $r_1(A)$ $w_2(A)$ $w_1(A)$ $w_3(A)$

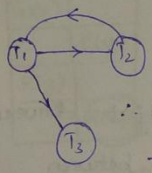
NOTE :⇒ If a schedule is conflict serializable then the schedule is view serialisable, so the 1st check for VS should be CS.

⇒ 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 ✗	CS ✗	CS ✓
Bw ✗	Bw ✓	VS ✓
VS ✗	VS ✓	

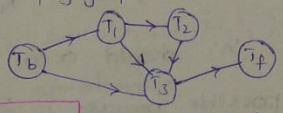
CS = Conflict serializable
Bw = Blind writes
VS = View serializables.



∴ Not CS ✗ ∴ Now check for Blind writed (write without Read)

Transaction 2 is writeing without Reading (∵ Bw ✓)

↳ Now draw poly graph



Now, Insert two dummy transactions T_b and T_f

T_b	T_1	T_2	T_3	T_f
$w(A)$				
	$R(A)$			
		$w(A)$		
	$w(A)$			
			$w(A)$	
				$R(A)$

∴ CS ✗
Bw ✓
$\boxed{\text{VS ✓}}$

No cycles in poly graph
The schedule is view serializable

$\cancel{T_b}$ T_1 T_2 T_3 $\cancel{T_f}$

$\boxed{T_1\ T_2\ T_3}$

we Assume

T_b = writes data items initially

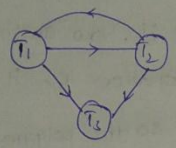
T_f = Read all data items

T_b, T_f are dummy transactions

16. POLY GRAPHS FOR VIEW SERIALISABLE EXAMPLE 2

$S = r_1(A)\ w_2(A)\ r_3(A)\ w_1(A)\ w_3(A)$ ⇒ find if it is view serialisable / not

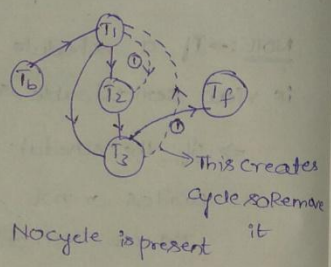
⇒ First check whether it is conflict serialisable / not



⇒ Not conflict serialisable (Cycle is present)

⇒ 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 ⇒ Now check for VS by Adding (T_b and T_f as dummy Transactions)

T_b	T_1	T_2	T_3	T_f
$w(A)$				
	$R(A)$			
		$w(A)$		
			$R(A)$	
	$w(A)$			
			$w(A)$	
				$R(A)$



∴ The schedule is view serializable and the serial schedule is

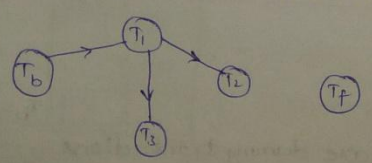
$\cancel{T_b}\ T_1\ T_2\ T_3\ \cancel{T_f} = T_1 T_2 T_3$

⇒ Procedure to draw the above poly graph

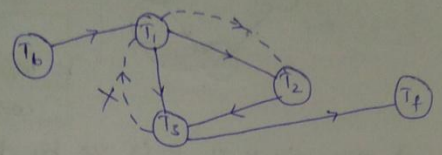
1) Check for write and Read's of the same object between two different transactions. The first WR is between

⇒ $w_b(A)$ $R_1(A)$ 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 ∴ 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

∴



⇒ Now again check for write Reads we find WR in $w_2(A), R_3(A)$ so all the writes of 'A' should occur before T_2 and after T_3

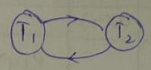


⇒ Now, one write is before T_2 i.e $w_1(A)$ so there exists an edge from T_1 to T_2 and $w_1(A)$ occurs between T_3 ∴ there exists an edge from T_3 to T_1. Since it forms cycle Remove it

⇒ Next WR is between T_3 and T_f so there exists an edge from T_3-T_f

17. EXAMPLES ON VERIFYING THE SCHEDULES - 1

T1	T2
R(x)	
	R(x)
w(x)	
	w(x)

⇒ 

⇒ Not conflict serializable

⇒ No Blind write (Not VS)

⇒ Not serializable

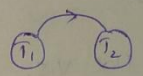
⇒ Recoverable ✓ → No (producer - consumer) (Read after writes)

⇒ Cascadeless

⇒ Not strict → $w_2(x)$ should be done after $w_1(x)$ commits

T1	T2
w(x)	
	R(y)
R(y)	
	R(x)

→ Dirty Read.

⇒ 

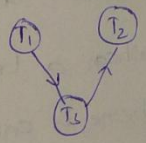
⇒ Conflict serializable, View serializable, Strict

⇒ Recoverable cannot be said (No commits) ⇒

⇒ cascading schedule

T1	T2	T3
R(x)		
	R(y)	
		w(x)
	R(x)	

→ producer - consumer



⇒ conflict serializable ($T_1T_3T_2$)

⇒ View serializable

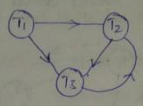
⇒ Recoverable cannot be decided

⇒ Cascading schedule

⇒ Not Strict.

18. EXAMPLES ON VERIFYING THE SCHEDULES - 2

T_1	T_2	T_3
R(x)		
R(y)		
w(x)		
	R(y)	
		w(y)
w(x)		
	R(y)	

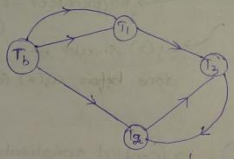


∴ Not conflict serializable

⇒ $w_3(y)$ is the Blind write

⇒ Now check if it is vs by drawing polygraph

T_b	T_1	T_2	T_3	T_f
w(x)				
w(y)				
	R(x)			
	R(y)			
	w(x)			
		R(y)		
			w(y)	
	w(x)			
		R(y)		
				R(x)
				R(y)



↳ Not V.S.

→ Not CS

→ Not VS

→ Recoverability Not decidable

⇒ Cascading

⇒ NOT Strict

19. EXAMPLES ON VERIFYING THE SCHEDULES - 3

T_1	T_2
R(x)	
	w(x)
w(x)	
	Abort
Commit	

Roll Back

⇒ when a Transaction says Abort then that means i should Roll back

⇒ when a transaction says Abort then there is only one transaction and hence it will be CS.

⇒ Recoverable (No commit on T_2)

⇒ Cascadeless (Independent

⇒ Not Strict $w_2(x)$ has not Read x.

T1	T2
w(x)	
	R(x)
w(x)	
	Abort
Commit	

pc problem

Rollback

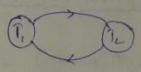
Reading before w(x) commited

⇒ CS ✓
⇒ VS ✓
⇒ S ✓

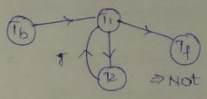
⇒ Producer - Consumer
⇒ Recoverable (T_2 is aborting)
⇒ Cascading schedule

135

T_b	T1	T2	T_f
w(x)	R(x)		
		w(x)	
	w(x)		
		Commit	
	Commit		
			R(x)



⇒ Not CS.
⇒ w(x) is the Blindwrite
⇒ Recoverable



⇒ Not VS, NOT STRICT

20. EXAMPLES ON VERIFYING SCHEDULES 4

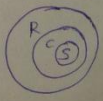
T_1	T_2
w(x)	
	w(x)
	R(x)
w(x)	
	Commit
Commit	

should occur before T_1 another
∴ $T_2 \to T_1$ edge will be present

⇒ Not CS
⇒ Bw present $w_2(x)$.
⇒ Not VS.
⇒ Not Recoverable

Now,

T_b	T_1	T_2	T_f
w(x)			
	w(x)		
		w(x)	
		R(x)	
	w(x)		
		Commit	
	Commit		
			R(x)



T_1	T_2
w(x)	
	R(x)
w(x)	
	commit
Abort	

Rollback

⇒ when T_1 aborts there will be Roll back and there will be only one transaction left

∴ The schedule is conflict serializable

⇒ VS

⇒ SV

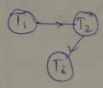
Recoverable – ✗

∴ cascading – ✓ (NOT Cascadeless)

⇒ Not strict

21. EXAMPLES ON VERIFYING SCHEDULE - 5

T1	T2	T3
	R(x)	
		w(x) commit
w(y) commit		
	R(y) w(z) commit	



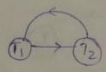
= CS, VS, S

= Recoverable

= Cascadeless schedule

= Strict

T1	T2	T3
R(x)		
	w(x)	
w(x) commit		
		R(x) commit

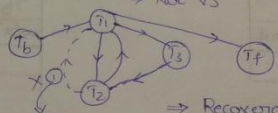


⇒ Not CS

⇒ BWV

⇒ Now draw poly graph

⇒ Not VS



⇒ Recoverable

⇒ Cascadeless

⇒ ↑ Strict

Not $w_2(x)$ and $w(x)$

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.

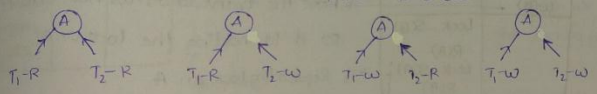
Lock - Based protocol

⇒ This protocol says that all the data items should be accessed in a mutually exclusive manner, to acheive this we use two locks called 1> shared lock (only Read) (lock-S)

2> Exclusive lock (Both Read and write) (lock-X)

	S	L
S	✓	✗
L	✗	✗

⇒ 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:

T_1 : $A \xrightarrow{50} B$

Read(A)
A = A - 50
write(A)
Unlock(A)
Lock - X(B)
Read(B)
B = B + 50
Write(B)
Unlock(B)

T_2 : Display(B+A)

Lock - S(B)
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 Transactions are interleaved (Concurrent execution) then there occur some conflicts.

T_1	T_2
Lock-X(A) Read(A) A = A-50 write(A) unlock(A)	
	Lock-S(B) Read(B) unlock(B) Lock-S(A) Read(A) unlock(A) Display(B+A)
Lock-X(B) Read(B) B = B+50 write(B) unlock(B)	

A B

100 200 = (A+B) = 300

50 + 200 = 250 = Incorrect it should be 300.

→ Because of concurrent execution we get 250.

⇒ Therefore Simple locking protocol doesn't give appropriate Results.

T_1	T_2
Lock-X(A) R(A) A = A-50 W(A)	
	Lock-S(B) R(B) Lock-S(A) R(A) Display(B+A) Commit unlock(B) unlock(A)
Lock-X(B) R(B) B = B+50 W(B) Commit unlock(B) unlock(A)	

→ Here the transaction has not unlocked 'A' so it is holding the lock on 'A'.

→ T_2 Requesting lock on A

← T_1 req lock on B

∴ There is a cycle, one transaction waits for the other to unlock

∴ DEADLOCK

∴ Simple locking might produce Deadlock

Simple locking might not Guarantee Serializability

25. TWO PHASE LOCKING PROTCOL

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
⇒ Shrinking phase - Release all the locks.
} No one can come in middle 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 T_2

→ locking point for T_3 (point where the final lock appears)

← locking point for T_1

∴ Serial schedule = $T_2\ T_3\ T_1$ = order in which we get locking points

⇒ The 2-phase locking has [sometimes 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 untill 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.

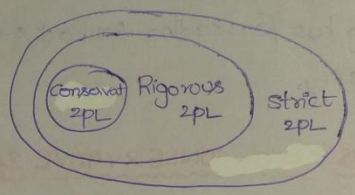
27. EXAMPLES ON 2PL

Which of the following schedules (Transactions) are in Strict 2PL?

a) Lock-S(A)
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' commited ∴ NOT STRICT 2PL



b) Lock-S(A)
R(A)
Lock-X(B)
unlock(A)
R(B)
W(B)
Commit
unlock(B)

unlock done before Commit ⇒ NOT Rigorous

⇒ There is Growth and shrink ⇒ 2PL ✓

⇒ Consider Exclusive locks lock X-(B) it should be unlocked only after 'B' commits

∴ $\boxed{\text{Strict 2PL ✓}}$

⇒ 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

∴ $\boxed{\text{NOT RIGOROUS 2PL}}$

c)
Lock-S(A)
R(A)
lock-X(B)
R(B)
W(B)
commit
unlock(A)
unlock(B)

d) lock-S(A)
lock-X(B)
R(B)
W(B)
R(A)
commit
unlock(A)
unlock(B)

e) Lock-S(A)
R(A)
unlock(A)
lock-X(B)
R(B)
W(B)
unlock(B)
unlock(A)
Commit

⇒ In the

⇒ In the

c)

lock-S(A)
R(A)
lock-X(B)
R(B)
w(B)
commit
unlock(A)
unlock(B)

⇒ Growth and shrink ⇒ 2PL

⇒ unlock (B) is done after commit ∴ $\boxed{\text{Strict 2PL}}$

⇒ unlock (A) is done after commit ∴ $\boxed{\text{Rigorous 2PL}}$

⇒ Conservative 2PL says that you have to acquire the locks before starting any operation

∴ The given schedule started R(A) before acquiring the lock on B

∴ $\boxed{\text{NOT conservative 2PL}}$

d) lock-S(A)
lock-X(B)
R(B)
w(B)
R(A)
commit
unlock(A)
unlock(B)

⇒ Growth and shrink ⇒ 2PL

⇒ unlock (B) done after commit = Strict 2PL

⇒ Both A & B are unlocked after commit = Rigorous 2PL

⇒ All the locks are acquired before proceeding the operations ⇒ conservative 2PL

e) lock-S(A)
R(A)
unlock(A)
lock-X(B)
R(B)
w(B)
unlock(B)
unlock(A)
commit

NOT 2PL

NOT 2PL

⇒ 2PL ✗

⇒ ~~Not strict 2PL~~ ∵ ~~unlock (B) done before commit~~

⇒ In the Growing phase locks are acquired and they can be upgraded

⇒ shared lock → upgraded to Xclusive lock

⇒ In the Shrinking phase locks are Released and locks can be degraded

→ Exclusive lock → 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 and each transaction T_i can lock a data item atmost once and must observe the following rules.

i> The 1st lock by T_i may be on any data item.

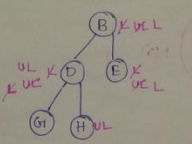
ii> Subsequently a data item can be locked by T_i only if the parent of the data item is currently locked by T_i

iii> Data items may be unlocked at any time

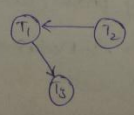
iv) A Data item that has been locked and unlocked by T_i cannot subsequently relocked by T_i

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(D)		

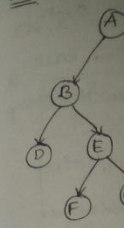
⇒ Now consider the Transactions one-by-one T_1 and then T_2 and then



{Now if you want to see serializability order, replace all the locks with write statements}



Ex



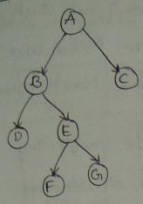
∴ The give
schedul

Advantages
⇒ Easily unl
⇒ Ensures
⇒ Deadlock

Drawbacks
⇒ you shoo
⇒ unneces

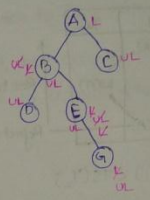
29. TIME S
⇒ Time stam
concurrency
⇒ It Requ
in advan
⇒ Each Tran
⇒ If any
this time

Ex

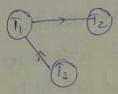


T_1	T_2	T_3
L(A) L(B) L(D) U(B)		
	L(B)	
L(C)		
		L(E)
U(D)		
		L(G) U(G) U(E)
	L(E) U(B) U(E)	
U(C)		

check whether this schedule is Acc to that Graph or not?



∴ The given schedule is according to the Graph given and serial schedule is



$T_3 - T_1 - T_2$

$T_1 \rightarrow T_3 \rightarrow T_2$

Advantages

⇒ Easily unlocks causes increase in concurrency.

⇒ Ensures Conflict serialisability

⇒ 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 Serialisable

⇒ 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 $T_S(T_i)$

→ If any transaction T_j that enters after T_i the relation between their time stamps be $TS(T_i) < TS(T_j)$ which means that the producing schedule must be equivalent to a serial schedule $T_i \rightarrow T_j$

1) 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.

T_1	T_2
$R(A)$	
	$W(A)$
$W(A)$	

Roll back

Rejected

$TS(T_1) < TS(T_2)$

$\therefore$ Conflict serialisable to $T_1 \rightarrow T_2$

$\Rightarrow$ Here $R_1(A)\ W_2(A)$ is the conflicting action which means T_1 should occur first and then T_2.

$\Rightarrow$ Again $W_2(A)$ and $W_1(A)$ is the conflicting action but T_1 should occur first and then T_2 so Reject the action & Rollback

30. THOMAS WRITE RULE

Thomas write Rule says that obselete Writes can be ignored. obselete write means if the writes are late and before that some other write has happend which is supposed to happen after it then you can clearly ignore the write and make the transaction stay there before rolling back.

Reject or Rollback

	Not Allowed	Allowed
Time Stamp ordering protocol ← TOP	$R_1(A)\ W_2(A)$ $W_2(A)\ R_2(A)$ $W_1(A)\ W_2(A)$	$R_2(A)\ W_1(A)$ $R_1(A)\ R_2(A)$ $W_2(A)\ R_1(A)$ $W_2(A)\ W_1(A)$
Thomas write rule ← TWR	$R_1(A)\ W_2(A)$ $W_2(A)\ R_2(A)$	$R_1(A)\ R_2(A)$ $W_1(A)\ W_2(A)$

Assuming $TS(T_2) < TS(T_1)$

$\boxed{T_2 \rightarrow T_1}$

If $T_2 \rightarrow T_1$ then $R_2(A)$ must be present

If $T_1 \rightarrow T_2$ is the Time stamp order and there is $R_1(A)$ is present.

31. EXAMPLES

check which c

1)

T_1
$R(B)$
$R(A)$
Display(A+B)

2)

T_1
$R(A)$
$W(A)$

3)

T_1	T_2
$W(x)$	
	$W(x)$
	$R(x)$

$\Rightarrow$ In the abo

$T_2\ T_3\ T_4$.

$1-2-3-4$.

$\Rightarrow$ Here $W_3(x)$

$\therefore$ This is

$\Rightarrow$ Now, Thom

here the con

31. EXAMPLES ON TIME STAMP ORDERING PROTOCOL

Check which of the following schedules can appear under TOP.

1)

T1	T2
R(B)	
	R(B)
	B = B - 50
	w(B)
R(A)	
	R(A)
Display(A+B)	
	A = A + 50
	w(A)
	Display(A+B)

⇒ Here T_1 is starting first ⇒ $TS(T_1) < TS(T_2)$

⇒ There are 2 conflicts ⇒ T_1 should come first acc to Time stamps.

⇒ In Both the conflicts T_1 has occured first before T_2 ∴ The schedule is according to TOP

2)

T_1	T_2
R(A)	⇒ valid
	w(A)
w(A)	(T_2 first) T_1 next

⇒ $TS(T_1) < TS(T_2)$

⇒ ∴ Not according to TOP.

3)

T_1	T_2	T_3	T_4
w(x)			
	w(x)		
		w(x)	
	R(x)		
			w(x)

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?

⇒ In the above schedule T_1 arrived first and then followed by T_2 T_3 T_4. ∴ The order in which we should serialise them is. 1-2-3-4.

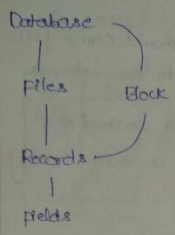
⇒ Here $w_3(x)$ $R_2(x)$ ⇒ '3' should occur first and then 'T_2' (Conflict acc to 1-2-3-4)

∴ 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 TWP

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-datablocks to access a record

$$\boxed{\text{Avg no. of block access} = \lceil \log_2(B) \rceil \text{ blocks}}$$

Adv : Searching is efficient

Dis : Insertion is expensive

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

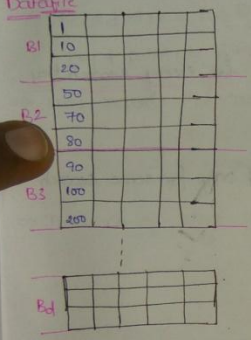
$\boxed{\text{Avg. no. of block access} = B/2 \text{ Blocks}}$

Adv: Insertion is easy

DisAdv: Searching is inefficient.

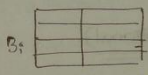
2. INDEXING

Datafile



Index file

1	B1
10	B1
20	B1
50	B2
70	B2
90	B3
100	B3



⇒ Avg No. of Block Access $= (\log_2 B_d)$

(Binary search)

Avg. Block Access $= (\log_2 B_i + 1)$

For searching in index file

for search in block

B_i = size of Index file

⇒ For every record in the datafile if we have a record/entry in the index file then that index is called Dense index

⇒ For every block in the datafile 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 $= (\log_2 B_i + 1)$

4. TYPES OF INDEXING

- Index
  - Single level Index
    - primary index (PK + ordered)
    - Clustered Index (Non Key + ordered)
    - secondary Index (Non key + unordered) or CK
  - Multilevel Index
    - B-Trees
    - B^+-Trees

  } Grow Breadth wise

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 keys values then it is called sparse index

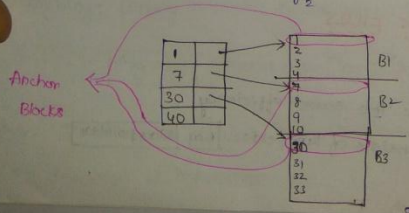
Primary Index (PK + optional)

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.

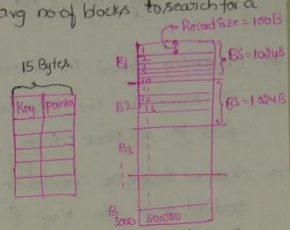
⇒ Avg. no. of block access $= \log_2(B_i) + 1$ (B_i = Index blocks)



1) I want to search 9 then I check in index file I go to '1' and now (9>1) so I goto 7 so now (9>7) so I goto 30, now as 9<30 I will say that 9 is in B2? and I check in the block (7-30)

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 un-spanned 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 = Block factor = $\frac{1024}{100} = \lfloor 10.24 \rfloor$

$= \underline{10}$

∴ 10 records can be placed in one block

$$\boxed{BF = \frac{\text{Block Size}}{\text{Size of each Record}}}$$

⇒ Total no. of Blocks =

1 Block — 10 Records

x Blocks — 30,000 Records

$$\Rightarrow \frac{x}{1} = \frac{30,000}{10} = \boxed{x = 3000 \text{ blocks}}$$

⇒ 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.

$$\boxed{\Rightarrow \text{No. of Block Access} = \lceil \log_2(45) + 1 \rceil = \lceil \log_2(2^{5.5}) + 1 \rceil = \lceil 6.5 \rceil = 7}$$

Total No of index Record = No. of Data blocks = 3000

1 Block (Index file) ⟶ 68 Records

x ⟶ 3000 Records

$$\Rightarrow \frac{x}{1} = \left\lceil \frac{3000}{68} \right\rceil = 45$$

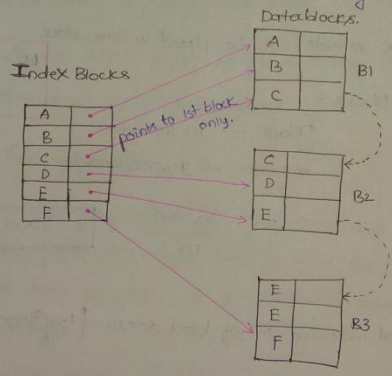
7. CLUSTURED INDEXING (NK + ordered)

⇒ clustering index is a combination of both Dense and sparse Index.

⇒ 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 datafile whose records are physically ordered on a non-Key field which doesnt have a distinct value for each record that field is called clustering field.



Searching: Binary search

Avg. no. of block Access = $\log_2(B_i) + 1$

⇒ Index entry is created for each distinct value of a clustering field

⇒ The block pointer points to first block in which the key is available

⇒ Type of Index is Dense / sparse

8. SECONDARY INDEXING (NK/UK + unordered)

⇒ Index file should always be an ordered file

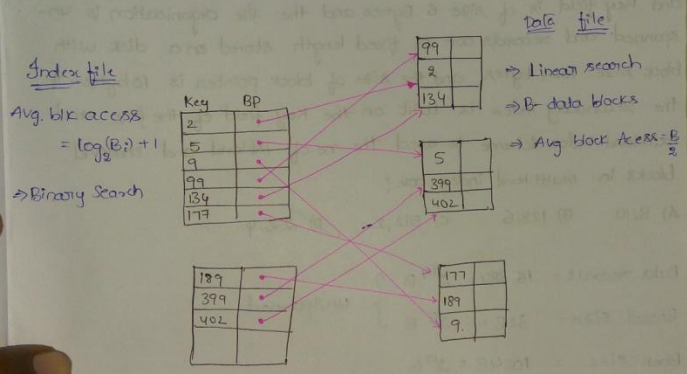
⇒ Secondary Index provides a secondary means of accessing a file for which some primary access already exists.

⇒ Secondary Index maybe a key or non key

⇒ Index entry is created for every record of datafile

⇒ No. of index entries = No. of records.

⇒ Type of secondary index is dense



⇒ Linear search

⇒ B- data blocks

⇒ Avg block Access = $\frac{B}{2}$

Avg. blk access $= \log_2(B_i) + 1$

⇒ Binary Search

Consider a secondary index is built on the key field of the file of previous Question. the find. no. avg no of block Access to access a record using with and without index

⇒ R = 30,000 (unordered)

⇒ No. of data Blocks = 3000 blocks

⇒ No. of block access required without indexing = $B/2$ = 1,500

⇒ 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 ⟶ 68 Records

x — 30,000 Records

⇒ $x = 441$ blocks

∴ $BA = \lceil \log_2(441) \rceil + 1$

$BA = 10$

9. EXAMPLE ON MULTILEVEL INDEXING

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 access to search for a record = n+1 (one blk at each level and 1 data blk finally)

Consider a file of 16,384 records, each record is of size 32-bytes and key field is of size 6 Bytes and the file organisation 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 multi level index are?

A) 8,10 B) 128,6 C) 512,2 D) 256,4

Data records = 16,384 = 2^{14}

Record size = 32B = 2^5 B } unspanned

Block size = 1024B = 2^{10} B.

Key-field = 6B, Bp = 10B

∴ Index Record size = Key + Block pointer = 16B.

⇒ 1st level Index

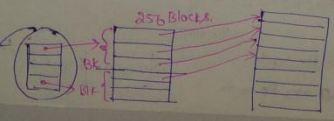
No. of Index Records = No. of Data Records / No. of index Records per block = $\frac{2^{10}}{2^4} = 2^6$

$= 2^{14}$

∴ No of blocks in Index $= \frac{2^{14}}{2^6} = 2^8 = 256$

⇒ At 2nd level Index

No. of blocks $= \frac{256}{2^6} = 4$

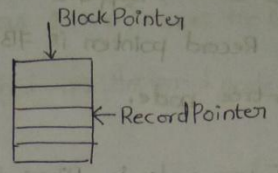


10. INTRODUCTION TO BTREES AND B^+ TREES

⇒ BTree and B^+ Trees are Dynamic

⇒ Node pointer and Block pointer

⇒ Record pointer



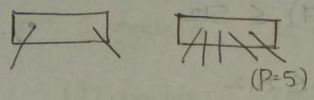
⇒ All the databases today use B^+ Trees.

⇒ The advantage of B, B^+ Trees is they grow vertically

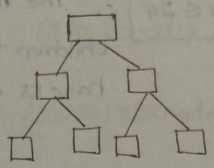
11. PROPERTIES OF BTREES

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.



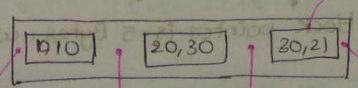
2) Internal nodes:



⇒ Internal nodes can store upto (p−1) keys.

⇒ Have between $\lceil p/2 \rceil$ and p children

The general Representation is $\langle P_1 \langle K_1, Pr_1 \rangle P_2 \langle K_2, {}_2 \rangle, \ldots P_q \langle K_{q-1}, Pr_{q-1} \rangle$



If you want the record whose key value = 30 then it's present at block no. 21.

contains the node whose key value < 10

Here 4 children are there ∴ p = 4
No. of keys = p − 1 = 3
= (10,10) (20,30) (30,21)

3) Leaf node : - stores between $\lceil (p-1)/2 \rceil$ and p−1 keys

- All leaf nodes have same depth.

12. EXAMPLE ON BTREE -- 1

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

$$\boxed{P(K_1 P_r)\ P(K_1 P_r) \text{ - - - - - - - - - - - - - } P}$$

⇒ The formula that every node should satisfy is

$$= \boxed{n * p + (n-1)(k + P_r) \leq \text{Block size}}$$

$$= n*6 + (n-1)(9+7) \leq 512$$

$$= 6n + 16n - 16 \leq 512$$

$$= 22n \leq 528 \Rightarrow \boxed{n \leq 24}$$

n = order of the tree

p = Block pointer size

P_r = Record pointer

∴ The max amount of children that this tree can have is 24.

13. EXAMPLE ON BTREE -- 2

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:

$$= n * p + (n-1)(k + P_r) \leq \text{Block size}$$

$$= n*5 + (n-1)(10+8) \leq 512$$

$$= 5n + 18n - 18 \leq 512$$

$$= 23n \leq 540$$

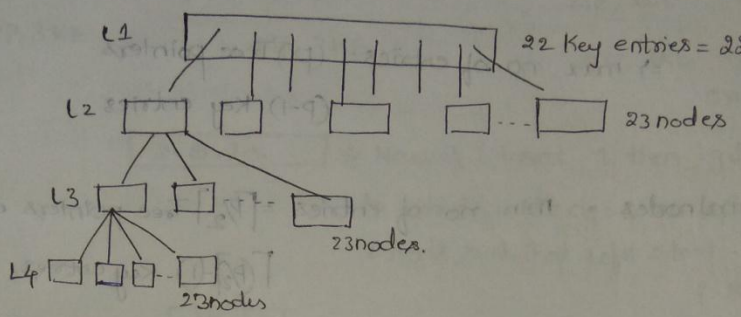
$$= n \leq \lfloor 23.478 \rfloor$$

$$= n = 23$$

14. EXAMPLE IN BTREE-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.

⇒ $n = 23$ (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	23x22 Index Records	23x23 node ptrs.
At level 3 :	(23)x 23nodes	(23x23)x 22 IR	(23x23) x23 NP
At level 4 :	(23x23) x23nodes	(23x23x23)x22 IR	X (leaf nodes)

No. of IR = 23x 23x 23x 22

$\boxed{\text{IR's} = 267674}$

15. UNDERFLOW AND OVERFLOW IN THE B-TREES

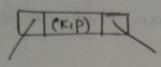
⇒ while we do Insertion and deletion in B-Trees we should check 2 conditions they are overflow and underflow

Insert - overflow

Delete - Underflow.

Order of B-Tree = p then,

i) Root node => min no of entries = 2 Tree pointers
1 Key entry



=> max no. of entries = (p) Tree pointers
(p-1) Key entries

ii) Internal nodes = min no. of entries = $\lceil p/2 \rceil$ Tree pointers and
$\lceil (p/2) - (1) \rceil$ Key entries.

= max no. of entries = 'p' Tree entries
(p-1) Key entries

iii) Leaf nodes = min no. of Key entries = $\lceil p/2 \rceil - 1$ Key entries

max no of key entries = (p-1).

p = 5

i) Root = min = 2TP, 1KE max = 5TP, 4KE

ii) Internal nodes = min = $\lceil p/2 \rceil$ TP = $\lceil 5/2 \rceil$ = 3TP, 2KE max = 5TP, 4KE

iii) Leaf nodes = min = $\lceil p/2 \rceil - 1$ = 2KE max = 4 K

16. INSERTION INTO BTREES -- EXAMPLE - 1

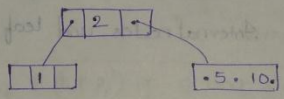
Insert the following Keys in B-Tree of order - 4.

Keys: 2, 5, 10, 1, 6, 9, 4, 3, 12, 18, 20, 25.

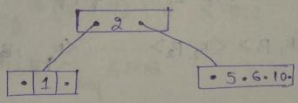
	Root	Intermediate nodes	leafs
Min:	2TP, 1KE	2TP, 1KE	2TP, 1KE
Max:	4TP, 3KE	4TP, 3KE	0TP, 3KE.

Overflow (so split)

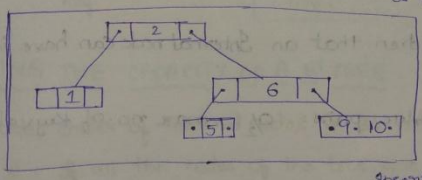
[• 2 • 5 • 10 •] ⇒ Now, if i insert 1 then i get (1 2 5 10) in which middle ele = 2 so split it into 2 nodes such that Left side = 1 Right = 5 10 & Root = 2



⇒ Now inserting '6' (6 > 2) so move in RST

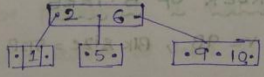


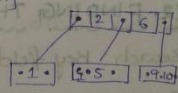
⇒ If 9 is inserted it becomes (5 6 9 10) overflow. split at mid point = 6.



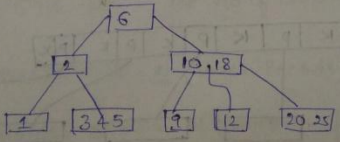
X

Inserting '4' ⇒

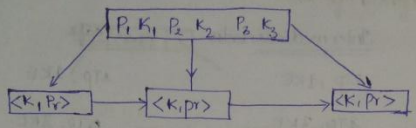




∴ Final tree =



21. B^+-TREES STRUCTURE OF LEAF AND INTERNAL NODES



⇒ The information needed to access the data is present in the leaves

⇒ The intermediate nodes donot contain the pointers that points to data insted they guide to reach the leaves for which it points to the Required data block

⇒ The structure of of the Internal nodes and leaf nodes are different

Internal node : $\langle P_1\ K_1\ P_2\ K_2 \cdots K_{q-1}\ P_q \rangle$

leaf node : $\langle \langle K_1\ P_1 \rangle \langle K_2\ P_2 \rangle \cdots \langle K_{q-1}\ P_{q-1} \rangle\ P_{next} \rangle$

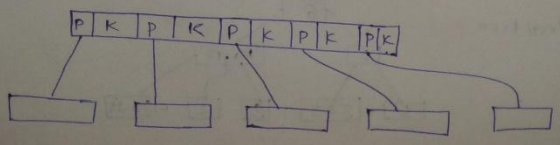
let, O_{leaf} be the order of leaf node

⇒ The min. no. of children that an Internal node can have is $\lceil O/2 \rceil$ to 'O' pointers.

⇒ min. no. of Key value pairs = $O/2$, max no. of Key value pairs = O.

22. FINDING THE ORDER OF B^+ TREE

Search Key field is V = 9B, Blk size = 512B, Record ptr = P_r = 7B, Blk ptr = P = 6B. what is the order of internal node and leaf node?



$\Rightarrow \boxed{O(p) + (O-1)k \leq \text{Block size}}$

O = Order

P = Block pointer

K = Key.

$= O(6) + (O-1)9 \leq 512$

$= 6(O) + 9(O) - 9 \leq 512$

$= 15(O) - 9 \leq 512 \Rightarrow 15(O) \leq 521$

$\Rightarrow O \leq \lfloor 34.3 \rfloor \Rightarrow \boxed{O = 34}$

$\therefore \boxed{\text{Order} = 34}$

leaf node: $\boxed{\boxed{K|P}\ \boxed{K|P}\ \boxed{K|P} \cdots \boxed{K|P}\ P_{next}} \rightarrow$

Let the order of leaf node = O_{leaf} = (max no of $\boxed{\text{Key, ptr}}$ pairs we can have in the leaf node)

$\Rightarrow \boxed{O_{leaf} * (K + P_r) + P \leq \text{Block size}}$

$\Rightarrow O_{leaf} * (7 + 9) + 6 \leq 512$

$\Rightarrow 16 * O_{leaf} \leq 506$

$\Rightarrow O_{leaf} \leq 31.2 \Rightarrow \boxed{O_{leaf} = 31}$ $\therefore$ order of leaf node = 31

23. 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 leavel '0' and leaves at level-3.

$\Rightarrow$ Inter node = 69% full = The no of children = $\frac{69}{100} \times 34 = 23$ ptrs
= 22 Keys
= (ptrs − 1)

$\Rightarrow$ leaf nodes = $0.69 \times 31 = 21.39 = 21$ (index Records) (Key, data ptr pairs)

$\Rightarrow$ At Root 1 nodes = 22 KE = 23 pointers.

$\Rightarrow$ At level 1 $23 \times 22 = 506$ KE | $23 \times 23 = 529$ ptrs.

At L2: 529 nodes
(529×22) KE = 11,638 KE
No of ptrs = $529 \times 23 = 12,167$

13: 12,167 nodes 12,167 × 21 = 255,507 (index records that the tree is able to store)

Imp points

1> Secondary index is built on non-ordering field. Hence if a file consists 'n' attributes then the no. of secondary indices is $2^n - 1$

2> Clustered Index = Non key + ordered ⇒ clustering index is used when we have clusters of similar database entries for particular attribute.

So 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.

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) $XY \to Z$ and $Z \to Y$ (b) $YZ \to X$ and $Y \to Z$

(c) $YZ \to X$ and $X \to Z$ (d) $XZ \to Y$ and $Y \to X$

a) $XY \to Z$ and $Z \to Y$
↓ Holds
↓ Not hold $\{3 \to 5, 3 \to 6\}$

b) $YZ \to X$ and $Y \to Z$
↓ Holds
↓ Holds

c) $YZ \to X$ and $X \to Z$
Holds
not hold $\{1-2, 1-3\}$

d) $XZ \to Y$ and $Y \to X$
↓ Not hold $\{1,3 \to 5, 1,3 \to 6\}$
↓ Holds.

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

b) $A \to B$ and $B \not\to C$

c) $B \not\to C$

d) $A \not\to B$ and $B \not\to C$

(a) $A \to B$ ↓ holds; $B \to C$ ↓ Not holds.

(b) $A \to B$ ↓ holds; $B \not\to C$ ↓ holds. for this instance but fail when insertion ⇒ NOT HOLDS.

(c) $B \not\to C$ holds

(d) $A \not\to B$ ↓ Not holds; $B \not\to C$ ↓ holds

6. FORMAL DEFINITION OF FUNCTIONAL DEPENDENCY

	A	B
t_1		
t_2		
	If t_1 and t_2 agree here	then they must agree here
	If t_1 and t_2 disagree here	then they may agree or 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 Equalences 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 : $A \to B$ if $B \subseteq A$

b) Transitive : $A \to B$ and $B \to C$ then $A \to C$

c) Decomposition: $A \to BC$ then $A \to B$ $A \to C$

d) Augmentation: $A \to B$ then $AC \to BC$

e) Union : $A \to B$ and $A \to C$, then $A \to BC$

d) Composition: $A \to B$ and $C \to D$, then $AC \to BD$.

Agree

Agree or may not Agree.

thods

8. CLOSURE SET OF ATTRIBUTES

Functional Dependencies : $A \to B$
$B \to D$
$C \to DE$
$CD \to AB$

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 CLOSUE SET OF ATTRIBUTES

The following functional dependencies are given:

$AB \to CD$
$AF \to D$
$DE \to F$
$C \to G$
$F \to E$
$G \to A$

which one of the following options is false ?

A) $(CF)^+ = \{A, C, D, E, F, G\}$ ✓ $(AF)^+ = \{A, C, D, E, F, G\}$

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

Sol) A) $\{CF\}^+ = \{C, F, G, A, E, D\} = \{A, C, D, E, F, G\}$ = TRUE

B) $\{BG\}^+ = \{B, G, A, C, D\} = \{A, B, C, D, G\}$ = TRUE

C) $\{AF\}^+ = \{A, F, E, D, \}$ = FALSE

D) $\{AB\}^+ = \{A, B, C, D, G\}$ = TRUE

10. DETERMINING CANDIDATE KEYS

Given Relation $R(A, B, C, D)$ and the FD's are $A \to B, B \to C, C \to D, D \to A$

The no of candidate Keys that can be formed $= 2^n - 1$ (n = no of attributes in the Relation)

$\therefore$ $\boxed{\text{No. of CK's} = 2^n - 1}$

$\Rightarrow$ 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

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, lets check if 'A' is candidate Key ⇒ $A^+ = \{A,B,C,D\}$

∴ A is candidate Key

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

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

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

⇒ $\{A,B,C,D\}$ are candidate keys

∴ $\boxed{\text{No. of CK's} = 4}$

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→F, E→A, EC→D, A→B) then CK = ?

A) CD B) EC c) AE d) AC

Sol $(CE)^+ = (A,B,C,D,E,F)$

∴ $(CE)^+ = (C,E,F,A,D,B)$

$\boxed{CK = EC}$

{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 $(CE)^+ = \{C,E\}$

∴ CE

or check by opvm

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 $\{E, F\} \rightarrow \{G\}$, $\{F\} \rightarrow \{I, J\}$, $\{E, H\} \rightarrow \{K, L\} \rightarrow \{M\}$, $\{K\} \rightarrow \{M\}$, $\{L\} \rightarrow \{N\}$ on R. what is the key for R?

(A) $\{E, F\}$ (B) $\{E, F, H\}$ (C) $\{E, F, H, K, L\}$ (D) $\{E\}$

Sol Given FD's = $\{E, F\} \rightarrow \{G\}$ $\Rightarrow (EF) \rightarrow (G)$

$\{F\} \rightarrow \{I, J\}$ $\Rightarrow (F) \rightarrow (IJ)$

$\{E, H\} \rightarrow \{K, L\} \rightarrow \{M\}$ $\Rightarrow (EH) \rightarrow (KL) \rightarrow (M)$

$\{K\} \rightarrow \{M\}$ $\Rightarrow (K) \rightarrow (M)$

$\{L\} \rightarrow \{N\}$ $\Rightarrow (L) \rightarrow (N)$

Now, $(EFH)^+ = (EFGHIJKLMN)$ ← The attributes that can be derived from LHS are (GIJKLMN) and there is no way; could get (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?

(a) AE, BE (b) AE, BE, DE (c) AEH, BEH, BCH (d) AEH, BEH, DEH

Sol The functional dependencies = $A \rightarrow B$, $BC \rightarrow D$, $E \rightarrow C$, $D \rightarrow A$ $\therefore$ Candidate Keys should contain EH.

$\Rightarrow (EH)^+ = (A, B, C, D, E, H)$ Now, $(EH)^+ = (E, H, C)$

Now, $(EH)^+ = \{E, H, C\}$

$\therefore (AEH)^+ = \{A, B, C, D, E, H\}$ ✓ → 8 SK'S [Total 6 attributes already included (AEH). ∴ Total tuples = $2^3 = 8$]

$(BEH)^+ = \{A, B, C, D, E, H\}$ ✓ → 8 SK's ∴ AEH, BEH, DEH are CK's

$(CEH)^+ = \{C, E, H\}$

$(DEH)^+ = \{A, B, C, D, E, H\}$ ✓ → 8 SK's.

14. GATE 2013 QUESTION ON CANDIDATE KEY

Relation R has 8 attributes ABCDEFGH. Fields of 'R' contain only atomic values. $F = \{CH \to G, A \to BC, B \to CFH, E \to A, F \to EG\}$ is a set of functional dependencies so that F^+ is exactly the set of FD's that hold for R; How many CK's does R have?

Sol

$CH \to G$

$A \to BC$ $\quad (\ D\ )^+ = (\cancel{A}\cancel{B}\cancel{C}\cancel{D}\cancel{E}\cancel{F}\cancel{G}\cancel{H})^+$

$B \to CFH$

$E \to A$

$F \to EG$

⇒ Now, $D^+ = \{D\}$

⇒ $(AD)^+ = (ABCDEFGH)$ ✓

⇒ $(BD)^+ = (ABCDEFGH)$ ✓

⇒ $(CD)^+ = (CD)$ ✗

⇒ $(ED)^+ = (ABCDEFGH)$ ✓

⇒ $(FD)^+ = (ABCDEFGH)$ ✓

⇒ $(GD)^+ = (GD)$ ✗

⇒ $(HD)^+ = (HD)$ ✗

∴ No. of CK's possible are = 4's = (AD)(BD)(ED)(FD)

15. EXAMPLES ON CANDIDATE KEYS - 1

1) R = (ABCDE)

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

$(\ )^+ = ABCDE$

⇒ $(AB\ )^+ = (ABCDE)$

Now, $(AB)^+ = \{A, B, C, D, E\}$ ∴ "AB" is the candidate key
⇒ 1 SK

Jes already Sou
∴ Total keys = 2^3 = 8]
EH are CK's

only atomic
set of
D's that had

D)

② R(ABCDE) FD = {AB→C, C→D, B→EA}

⇒ $(B)^+ = (ABCDE)$

Now $(B)^+ = \{B, E, A, C, D\}$ ∴ 'B' is the candidate key

∴ No. of super keys = $2^4 = 16$

③ R(ABCDEF) FD {A→B, C→D, E→F}

⇒ $(ACE)^+ = (ABCDEF)$

⇒ $(ACE)^+ = \{A, C, E, B, D, F\}$ ⇒ "ACE" is the CK and $2^3 = 8$ SK are possible

④ R = ABCD, FD = {AB→CD, D→A}

⇒ $(B)^+ = (ABCD)$ ⇒ 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 = {AB→CD, C→A, D→B}

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

$B^+ = \{B\}$

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

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

Two attribute keys = $(AB)^+ = (ABCD)$ ✓

$(AC)^+ = (AC)$

$(AD)^+ = (ADBC)$ ✓

$(BC)^+ = (BCAD)$ ✓

$(BD)^+ = (BD)$

$(CD)^+ = (CDBA)$ ✓

= AB, AD, BC, CD are CK's.

∴ CK's are: AB, AD, BC, CD.

EXAMPLES ON CANDIDATE KEYS - 3

R = (ABCDEF)

FD = {AB→C, C→D, D→E, E→F, F→A}

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

⇒ $(AB)^+ = (ABCDEF)$

⇒ $(CB)^+ = (ABCDEF)$

⇒ $(DB)^+ = (ABCDEF)$

⇒ $(EB)^+ = (ABCDEF)$

⇒ $(FB)^+ = (ABCDEF)$

∴ CKs = (AB)(CB)(DB)(EB)(FB

= 5CK's

18. CANDIDATE KEY - Example - 4

R = (ABCDEF)

FD = {AB→C, C→D, D→EB, E→F, F→A}

Now,

$A^+ = A$

$B^+ = B$

$C^+ = (CDEBFA)$ ✓

$D^+ = (DEBFAC)$ ✓

$E^+ = EFA$

$F^+ = FA$

Now,

$(AB)^+ = (ABCDEF)$

$(AE)^+ = (AEF)$

$(AF)^+ = (AF)$

$(EF)^+ = EFA$

$(BE)^+ = (BEFACD)$ ✓

$(BF)^+ = (BFACDE)$ ✓

Now,

(BEF) ✗

$(AEF)^+ = (AEF)$ ✗

(ABF) ✗

(ABE) ✗

Now,

(ABEF) ✗

∴ 'C' and 'D' are candidate keys

AB and BE, BF are candidate keys.

19. EXAMPLE

R (ABCDEF)

A → BCDEF

BC → ADEF

DEF → ABC

Now,

$A^+ = (ABCDE$

$B^+ = (B)$ ✗

$C^+ = ✗$

$D^+ = ✗$

$E^+ = ✗$

$F^+ = ✗$

20. EXAM

R = (ABCDE

FD = {A→B

Now,

$(E)^+ = \{E\}$

21. EXAM

R(ABCDE

A→BC

CD→E

B→D

E→A

19. EXAMPLES ON CANDIDATE KEYS - 5

R(ABCDEF)

A → BCDEF

BC → ADEF

DEF → ABC

Now,

$A^+ = (ABCDEF)$ ✓

$B^+ = (B)$ ×

$C^+ = \times$

$D^+ = \times$

$E^+ = \times$

$F^+ = \times$

Now,

$(BC)^+ = ABCDEF$

Now,

$(DEF)^+ = (DEFABC)$

∴ $\boxed{\text{Candidate keys} = A, BC, DEF}$

20. EXAMPLE ON CANDIDATE KEY - 6

R = (ABCDE)

FD = {A → B, B → C, C → D, D → A}

Now,

(ABEF) ×

Now,

$(E)^+ = \{E\}$

Now,

$(AE)^+ = (AEBCD)$ ✓

$(BE)^+ = (BECDA)$ ✓

$(CE)^+ = (CEDAB)$ ✓

$(DE)^+ = (DEABC)$ ✓

∴ $\boxed{\text{The CK's are : AE, BE, CE, DE}}$

No. of CK's = 4

21. EXAMPLE ON CANDIDATE KEY - 7

R(ABCDE)

A → BC

CD → E

B → D

E → A

Now,

$A^+ = (ABCDE)$ ✓

$B^+ = BD$

$C^+ = C$

$D^+ = D$

$E^+ = (EABCD)$ ✓

$(BC)^+ = (ABCDE)$ ✓

$(BD)^+ = (BD)$

$(CD)^+ = (ABCDE)$ ✓

∴ CK's are A, E, BC, CD

22. CANDIDATE KEY FOR SUB RELATION - EXAMPLE - 1

R(ABCD)

{AB→CD, D→A} what are the candidate keys of subrelation R_1(BCD)?

Now, Given Relation R_1(BCD)

$B^+ = B$ $\quad (BC)^+ = BC$

$C^+ = C$ $\quad (BD)^+ = CDAB$ $\quad \therefore$ candidate Key = BD

$D^+ = DA$ $\quad (CD)^+ = CDA$

23. CANDIDATE KEY FOR SUB RELATION EXAMPLE 2

R(ABCDEF)

FD's : {AB→C, B→D, AD→F, C→D, D→E, E→F, E→D}

Find candidate keys for R_1(DEF).

Given Sub Relation R_1(DEF)

$D^+ = \{DEF\}$ ✓

$E^+ = \{EDF\}$ ✓ $\quad \therefore$ The candidate keys are D, E

$F^+ = \{F\}$

24. CANDIDATE KEYS FOR SUB RELATION -- EXAMPLE 3

R (ABCDE)

FD's {A→BC, CD→E, B→D, E→A}

what are the candidate keys of R_1(ABCE)

Given sub Relation R_1(ABCE)

$A^+ = ABCDE$ $\quad$ Now,

$B^+ = BD$ $\quad BC = BCDEA$ ✓

$C^+ = C$ $\quad BD$ × (D not intable) $\quad \therefore$ Candidate keys are,

$E^+ = EABCD$ $\quad CD$ × (D not intable) $\quad$ A, E, BC

25. CHECKING

⇒ Given the se

functional Depe

Inference Rule

R(AB) and the

X	→ Y
ϕ	ϕ
A	A
B	B
AB	AB

⇒

The no. of Add

✓ϕ→ϕ ✓A

×ϕ→A ✓A

×ϕ→B ✓A

×ϕ→AB ✓

The no. of FD's

26 CHECKING

R(ABC)

FD's: {A→B, B→

X → Y

↓ ↓

2^3 × 2^3 = 8×

ϕ→ϕ (only

A → {ABC} =

B → {BC} =

C → {C} =

AB → $(AB)^+$ = {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 $\{A \to B, B \to A\}$

X	Y
ϕ	ϕ
A	A
B	B
AB	AB

⇒ No. of functional dependencies are $4 \times 4 = \underline{16}$

The no. of Additional FD's possible are (along with given FD's $A \to B$, $B \to A$)

✓ $\phi \to \phi$	✓ $A \to \phi$	✓ $B \to \phi$	✓ $AB \to \phi$
× $\phi \to A$	✓ $A \to A$	✓ $B \to A$	✓ $AB \to A$
× $\phi \to B$	✓ $A \to B$	✓ $B \to B$	✓ $AB \to B$
× $\phi \to AB$	✓ $A \to AB$	✓ $B \to AB$	✓ $AB \to AB$

→ $B^+ = \{BA\}$: possible

→ $A^+ = \{AB\}$ = possible.

The no. of FD's possible on the table are (✓) 13

26 CHECKING ADDITIONAL FD'S - EXAMPLE 2

R(ABC)

FD's: $\{A \to B, B \to C\}$

$\underline{X} \to \underline{Y}$

$2^3 \times 2^3 = 8 \times 8 = 64$ FD's

$\phi \to \phi$ (only one FD possible with ϕ) = 1 FD

$A \to \{ABC\}$ = 8 FD's are possible with LHS = 'a' because $A^+ = \{ABC\} = 2^3$

$B \to \{BC\}$ = 4 FD's are possible with LHS = 'B' because $B^+ = \{B, C\} = 2^2$

$C \to \{C\}$ = 2 FD are possible with LHS = 'C' because $C^+ = \{C\} = 2^1 = 2$ ($C \to \phi$, $C \to C$)

$AB \to (AB)^+ = \{ABC\} = 8$ FD

$(BC)^+ = (BC)^+ = \{BC\} = 4$ FD

$AC \to (AC)^+ = \{ABC\} = 8$ FD's

$(ABC)^+ = \{ABC\} = 8$ FD's

⇒ Total valid fd's $= 1+8+4+2+8+4+8+8 = 43$

∴ In General the no. of functional dependencies that are possible over a Relation R containing 'n' attributes is 2^{2n} // $X \rightarrow Y$ 2^n $2^n \Rightarrow 2^n \times 2^n$ $= 2^{2n}$ //

(34)

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, $(CD)^+ = \{CDEABD\} \Rightarrow (CD^+) = (AC)$ ✓ possible

$(BD)^+ = \{BD\}$ = NOT POSSIBLE

$(BC)^+ = \{BCDEA\} = (CD)$ is possible

$(AC)^+ = \{ACBDE\} = (BC)$ is possible

28 EQUALENCE OF FD'S

$F: \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$

$G: \{A \rightarrow CD, E \rightarrow AH\}$ are both the functional dependencies Equivalent

Sol First check if 'F' covers 'G' i.e $\boxed{F \supseteq G}$ how to check is take each FD in the set 'G' 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.

Now, $F \supseteq G$ Now: $G = \{A \rightarrow CD, E \rightarrow AH\}$

F: ↓ covered by 'F' ↓ covered by "F"

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

$E^+ = \{EADCH\}$

$F \supseteq G$ is TRUE