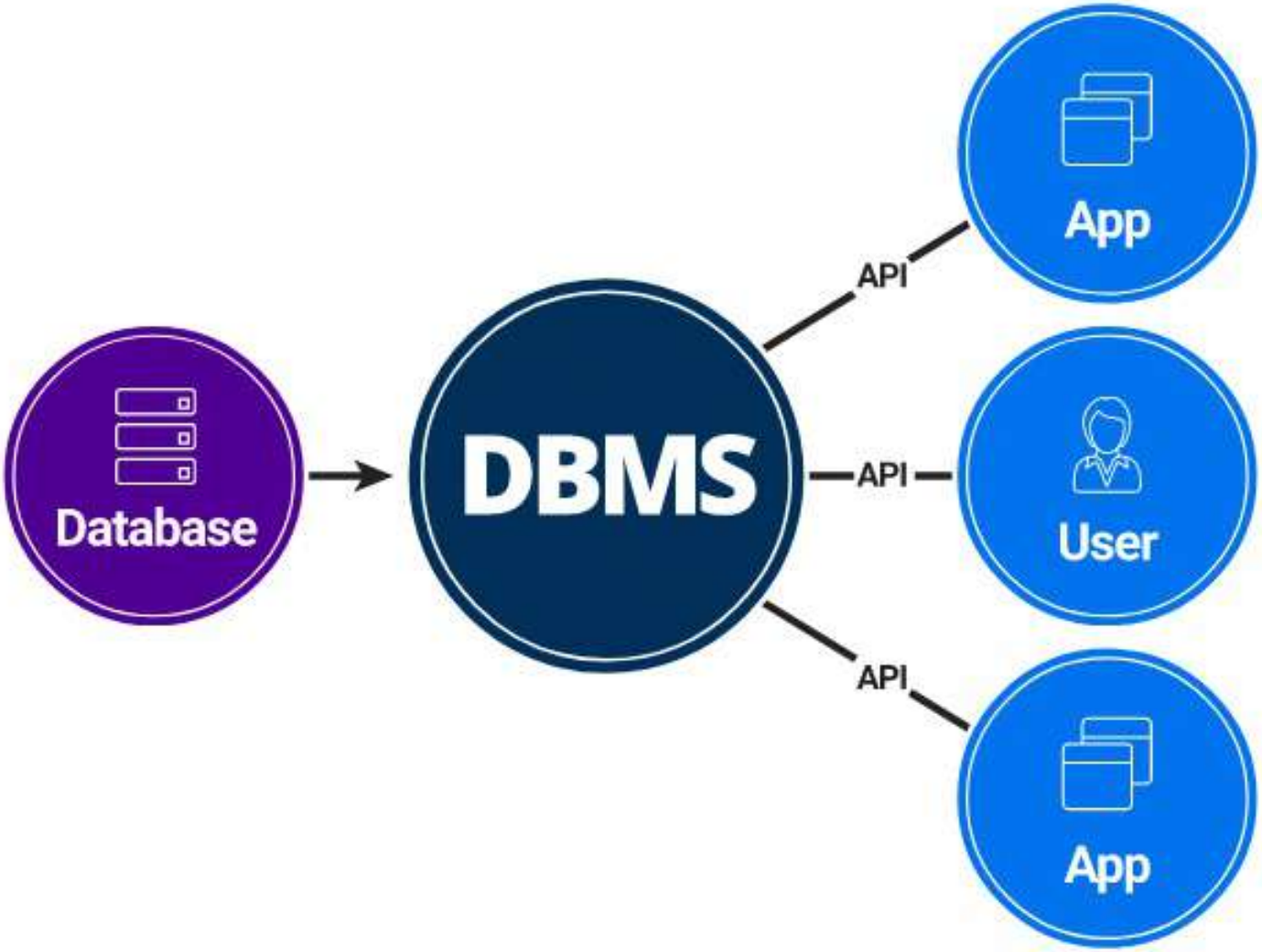
DBMS



Database Management System(8 Marks)

- ER Model & Relational Model
    - Models in DBMS
    - ER Model and notations
    - Relational Model
    - Conversion of ER to Relational model
- Database design and Normalization
    - Inference Rules & Integrity constraints
    - Closure set of FDs and Discovery of Candidate Key
    - Lossless decomposition & Dependency preserving
    - Equivalence set of FD & Minimal cover
    - Normalization
- Relational Algebra & SQL
    - Operations of Relational Algebra
    - Relational calculus
        - Tuple Relational Calculus
        - Domain Relational Calculus
    - SQL
- Transaction
    - ACID Property
    - Problems due to concurrency
    - Recoverability of Transactions
    - Concurrent Execution and its problems
    - Classification of Schedules
        - Based on Recoverability
            - Irrecoverable schedule
            - Recoverable schedule
                - Cascadeless rollback schedule
                - Strict recoverable schedule
        - Based on Serializability
            - Conflict and View Serializability
- Concurrency Control Techniques
    - Introduction
    - Shared & Exclusive locks
    - 2 Phase locking protocols
        - Strict 2PL
        - Rigorous 2PL
    - Timestamp based
- File Organization & Indexing
    - File Organization
    - Index Structure, Multilevel Index
    - B and B+ tree

Top Notorious Topics (Recent/Frequent)

★ ★

★ ★

★ ★

★ ★

★ ★

What is DBMS ?

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.

Levels of Viewing a Database

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, Middlename, LastName (Composite Attribute)

Age = Single valued Attribute

PNO = Multi valued Attribute

DOB = Stored Attribute — Derives. → Age = Derived Attribute

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. (works → exactly)

2. New department need not have any Employee.

Degree:

Cardinality

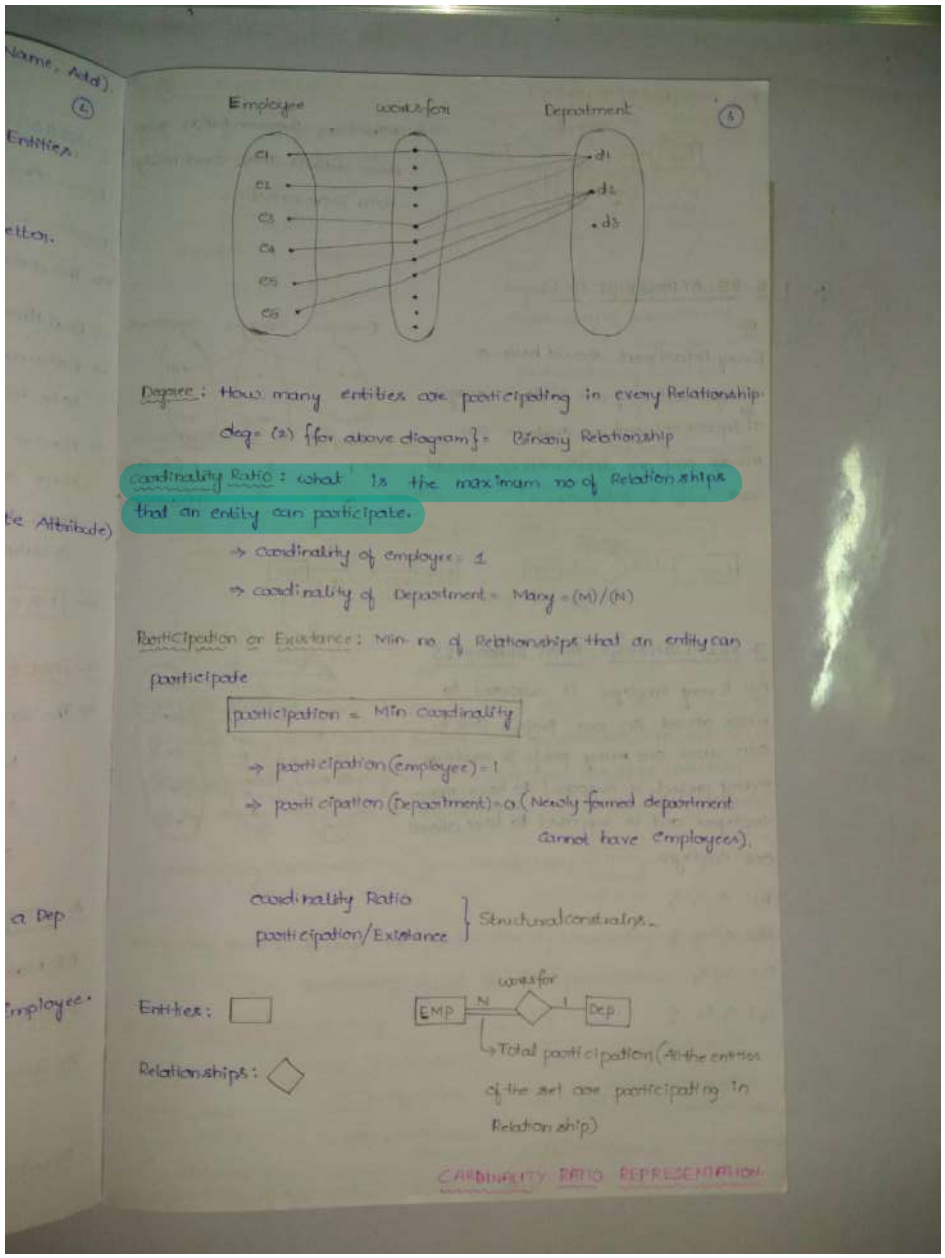
that an e

Participat

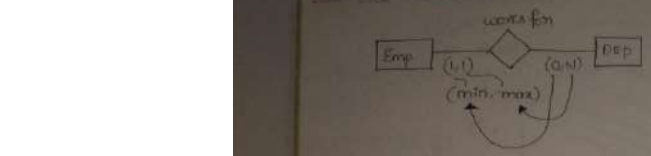
partici

Entities

Relation


Employee
works for
Department
e1
e2
e3
e4
e5
e6
d1
d2
d3
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
participation = Min Cardinality
⇒ participation (employee) = 1
⇒ participation (Department) = 0 (Newly formed department cannot have employees).
Cardinality Ratio
participation/Existance
} Structural constraints
Entities:
Relationships:
works for
EMP
N
1
Dep
Total participation (All the entities of the set are participating in Relationship)
CARDINALITY RATIO REPRESENTATION

MIN MAX REPRESENTATION

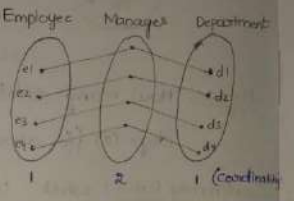


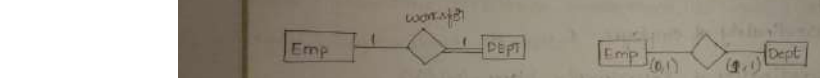
⇒ Min- Max Representation gives more details than Cardinality Ratio Representation.

6. RELATIONSHIPS (1-1)

RA:

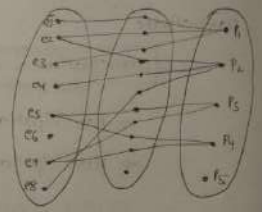
Every department should have a manager and only one manager manages a department and an employee can manage only one department. (Emp should work only in 1 department)





7. RELATIONSHIPS' M-M EXAMPLES

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



P1: e_1 e_2 e_3

P2: e_2 e_4 e_8

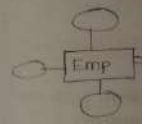
P3: e_5 e_7

P4: e_1 e_5 e_6

8. RECURSIVE

RA: Every e
have exactly

9. ATTRIBUT



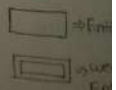
WEAK

⇒ whenever
entity is
to the

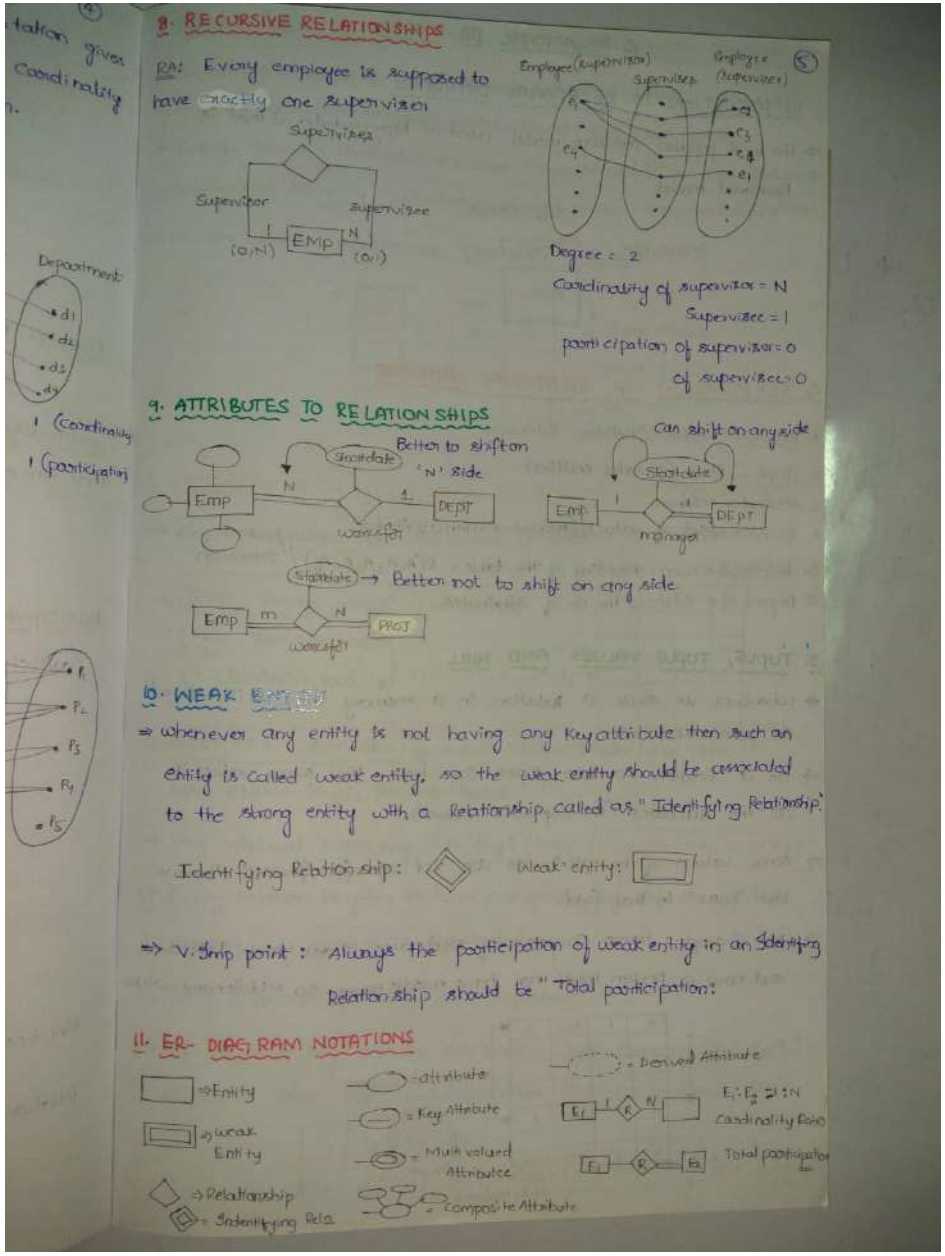
Identif

⇒ V. Strip

11. ER- DIA



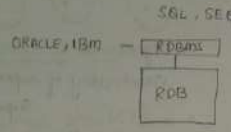
Total participation is necessary for weak entity. However total participation doesn't mean that the entity is weak



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

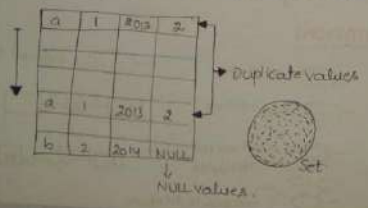
3. TUPLE, TUPLE VALUES AND NULL

⇒ 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. CONSTRAI

1. Domain con
2. Key const
3. Entity- Int
4. Referential

⇒ We can vie

5. CONSTRAI

⇒ Key - cons

(S.NO, S
(1, Ra
(1, Ravi

⇒ SUPER KE

same valu

⇒ Any Mini

⇒ Every Rel

Super Key

⇒ any sup

⇒ If A1 is

attributes

Conditions

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 ✗		
✓	SNO	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

$t_1 \neq t_2$

⇒ SUPER KEY ⊆ Attributes for which no two tuples have the same values in all the Attributes

⇒ Any Minimal superkey is a "key". (SNO) KEY = 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)

$= 1 \times 2 \times 2 \times 2 = 8$ 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

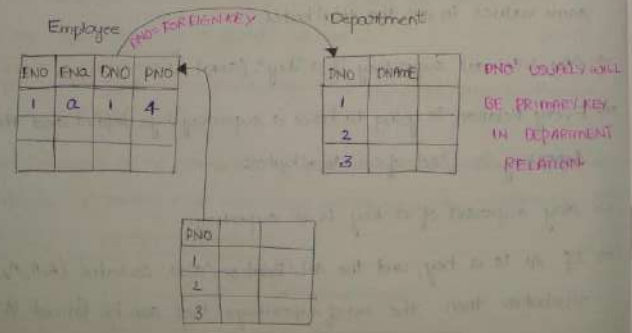
∴ Keys = $(A_3A_4,\ A_1A_2A_4)$ ⇒ candidate keys.

Minimal super keys

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

6. ENTITY AND REFERENTIAL INTEGRITY CONSTRAINTS

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



⇒ Foreign Key

7. ACTIONS UPO

⇒ The Actions

⇒ If the abo
to reject

⇒ while "delet
and the

1.

2.

3

2. COUNTIN

⇒ Given a

Then the

⇒ Foreign key can have "NULL" values unlike primary key

(9)

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

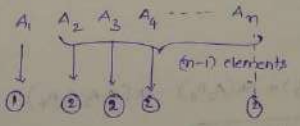
1. Ignore it (Reject the action)
2. Cascade (Delete the tuple and also delete the tuples which are being referred by the above tuple & they should be deleted).
3. Set NULL or some other value

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



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

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

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_1A_2 not A_1, A_2 (Single ck / Two ck's)

$A_1\ A_2\ A_3\ A_4 \cdots A_{n-2}$

$\downarrow\ \downarrow$ $(n-2)$ ele

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

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

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

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

The no. of super keys = $Sk(A_1) + Sk(A_2) + SK(A_1A_2)$

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

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

10. COUNTING THE NO. OF SK's POSSIBLE – EXAMPLE 3

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

$CK = \{A_1, A_2A_3\}$

No. of SK's = $Sk(A_1) + SK(A_2A_3) - SK(A_1A_2A_3)$

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

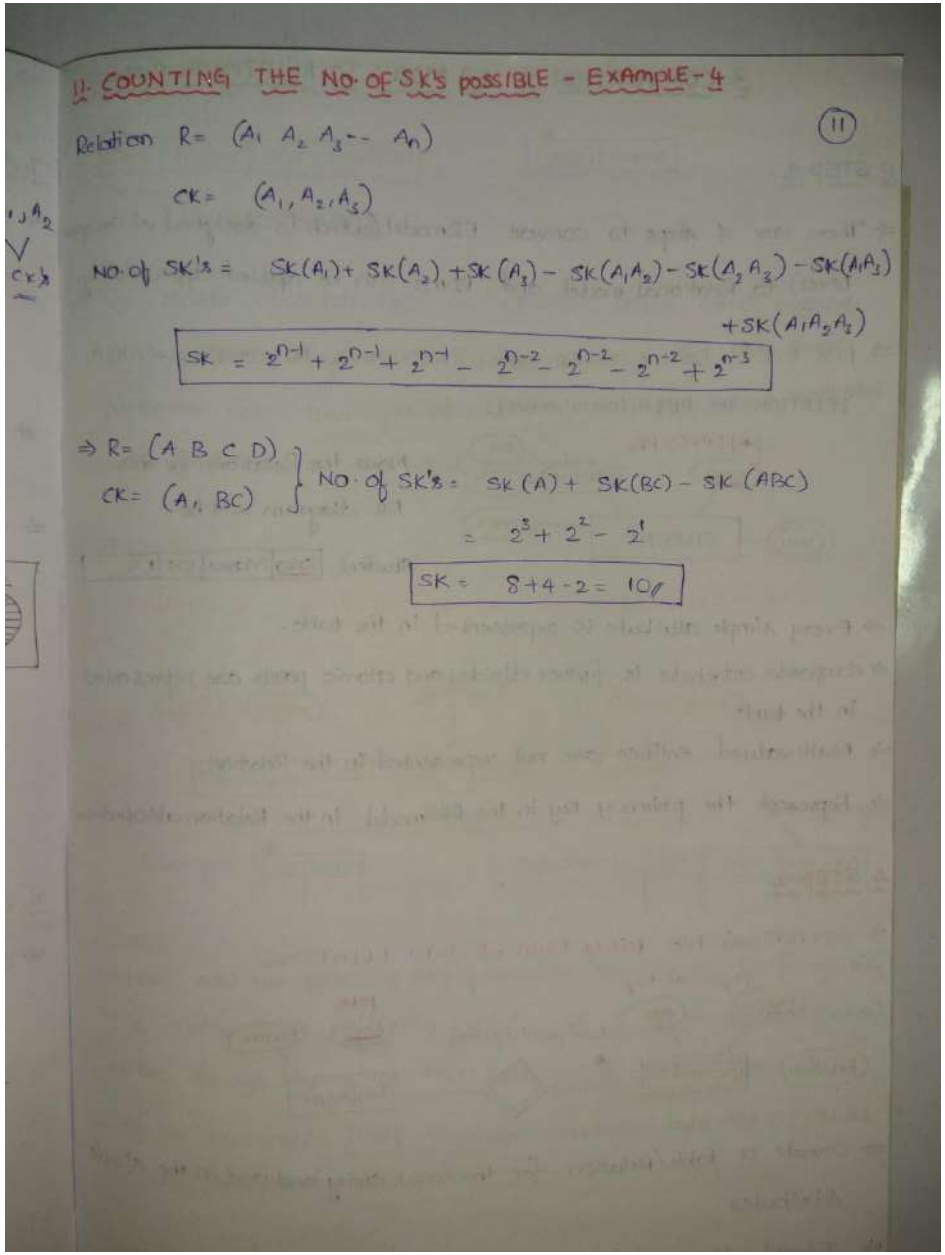
$CK = \{A_1A_2, A_3A_4\}$

No. of SK's = $SK(A_1A_2) + SK(A_3A_4) - SK(A_1A_2A_3A_4)$

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

$CK = \{A_1A_2, A_1A_3\} \Rightarrow$ No. of SK's = $SK(A_1A_2) + SK(A_1A_3) - SK(A_1A_2A_3)$

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



No. Of Candidate keys

Q.61 NAT Subject: DBMS Max Marks: 2

Consider table R with attributes A,B,C,D, and E. What is the largest number of candidate keys that R could have at the same time ?

Time taken to answer this question 00:00:24 hrs

Hide Answer | Add Notes | View Notes | Show Comments

Correct Answer

Solution: (10)

We can get max no of candidate keys by combining attributes such the no one should be the subset of others
So we have 5 possibilities here
taking 1 attribute as candidate key : ie 5C1 (A,B,C,D,E)
Similarly: for 2 attributes and 3 attributes 5C2=5C3=10
4 attributes 5C4 =5
5 attributes = 1
So maximum no of candidate keys that R could have at the same time is 10

Your Answer is wrong (31)

Q. 32 Solution Video | Have any Doubt ?

Consider database table R with 8 attributes A, B, C, D, E, F, G, H. What is the largest number of candidate keys that R could simultaneously have, given that ABC is already known to be a candidate key?

66 Correct Option

Solution :
66
8 attributes. Maximum number of candidate keys possible = $^{8}C_{4}$.
But ABC is already a key, so we need to remove those keys which have ABC, which is 5.
So, Total = $^{8}C_{4} - {}^{5}C_{1} + 1 = 66$

+1 for ABC

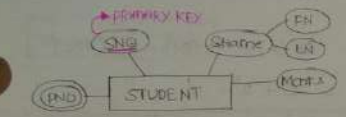
Your Answer is 21

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.

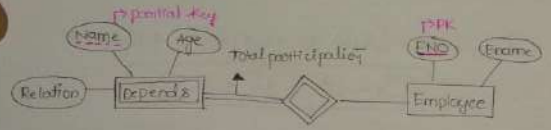
⇒ composite attribute is further divided and atomic parts are represented in the table

⇒ 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.

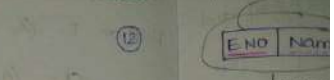


⇒ Create a table/Relation for the weak entity and add all the simple Attributes

⇒ 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



⇒ The delete

if you de

particular

with that

3. STEP-3

⇒ CONVERT

Employe

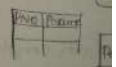
⇒ Now A

of anot

as the fo

⇒ If the

on the t



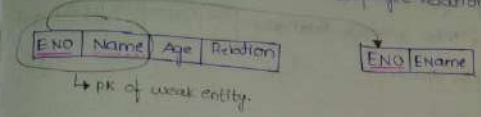
⇒ when y

Relation

of tupl

Depends Relation

Employee Relation



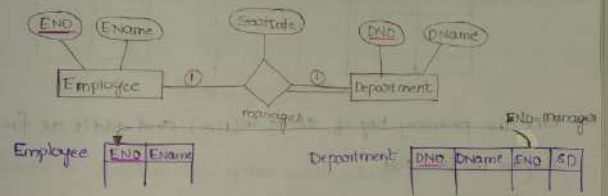
↳ PK of weak entity.

(13)

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

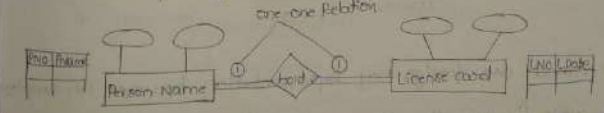
STEP-3

* CONVERTION OF RELATIONSHIPS INTO RELATIONS (TABLES)

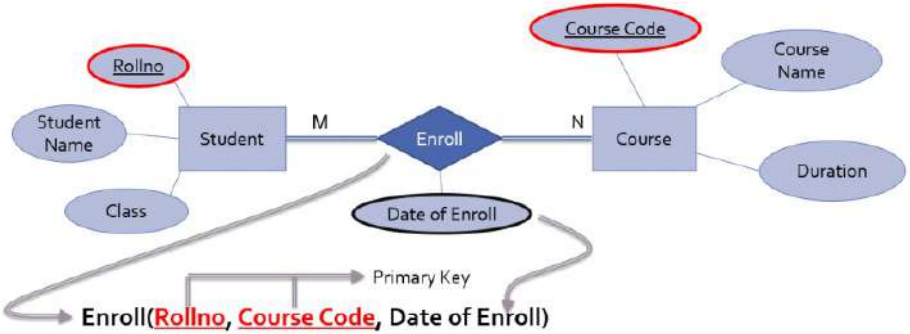


⇒ Now Add the primary key of one side (One table) as the foreign key of another table, generally add the PK of (non-total participation) side as the foriegn key of the total participation table.

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



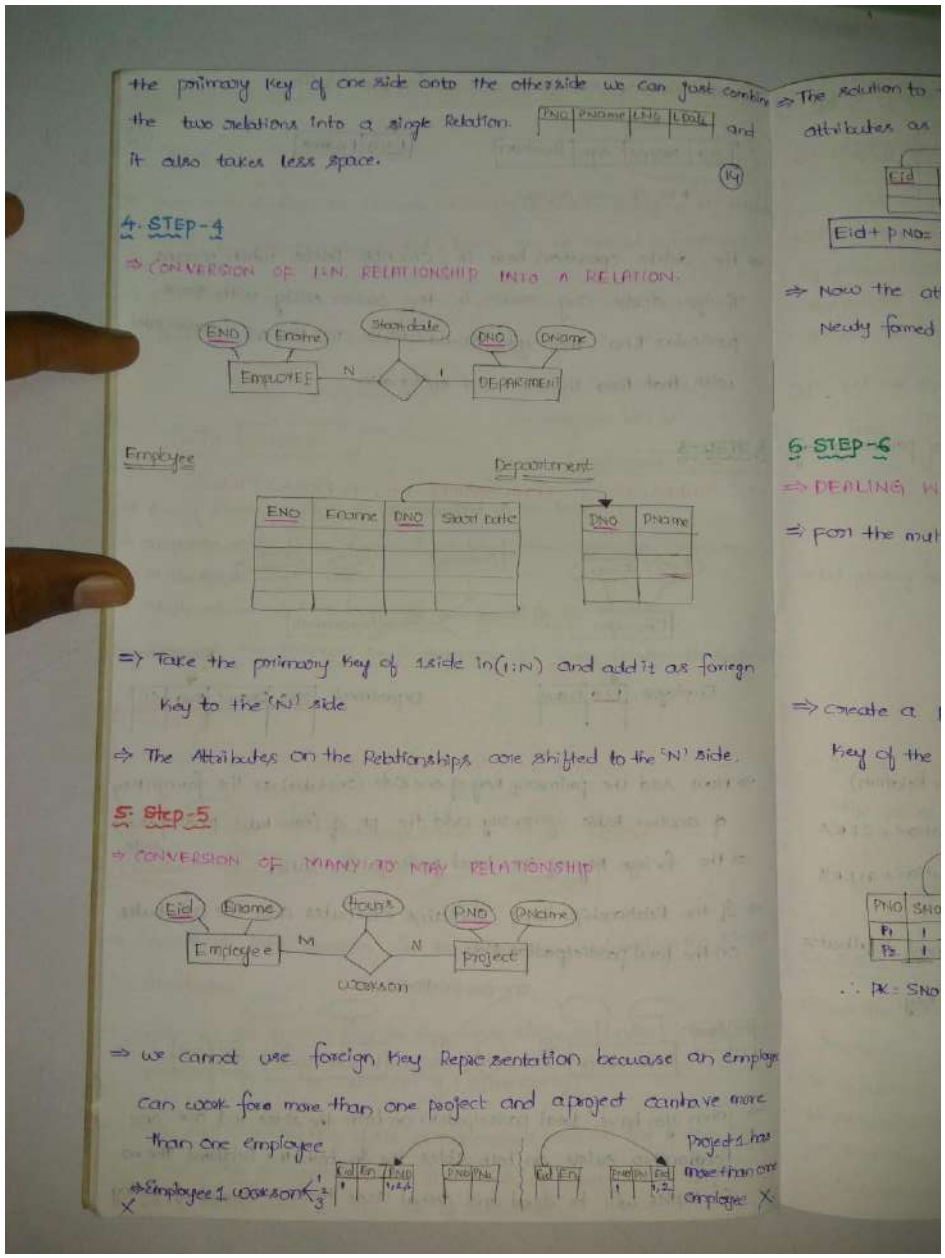
⇒ when you have total participation on both the sides and one-one Relationship exists on both sides the in both the Relations the no. of tuples will be equal and hence because there is no need of adding



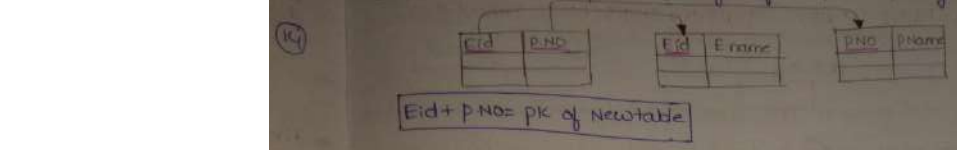
Enroll(Rollno, Course Code, Date of Enroll)

Student(Rollno, Student Name, Class)

Course(Course Code, Course Name, Duration)



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



Eid + PNO = PK of Newtable

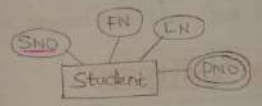
⇒ Now the attribute on the Relationship should be added to the Newly formed table.

Eid	PNO	Hours

6. STEP-5

⇒ DEALING WITH MULTIVALUED ATTRIBUTES

⇒ For the multivalued attribute we are going to create a new table



⇒ 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

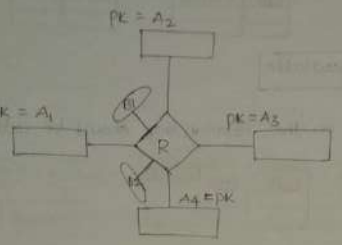
∴ PK = SNO + PNO

SNO	FN	LN
1	a	b

∴ PK = SNO

7. STEP-7 (6)

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



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

A1	A2	A3	A4	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 PK's
N-ary Relationship type	Relationship "relation" + n PK's
Simple Attribute	Attribute
Composite Attribute	set of simple component Attributes
Multi valued Attribute	Relation and Foreign Key.
value set	Domain
Key Attribute	primary Key.

9. GATE IT 2

Hotel Room

Lodging is ...
by person(s)
attribute to
a) Hotel b) ...

Sol: Let us ...

Rent

⇒ Let us say ...

⇒ In case of ...
the PK ...

∴ The ...

10. GATE 12

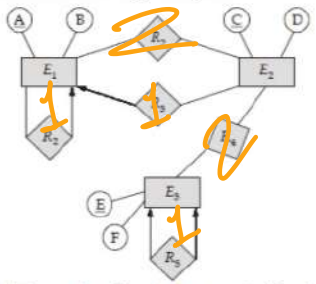
Given the ...

Incorrect

a) An attribute ...
b) An attribute ...
c) In a ...

Q. 53

Consider the following ER diagram:



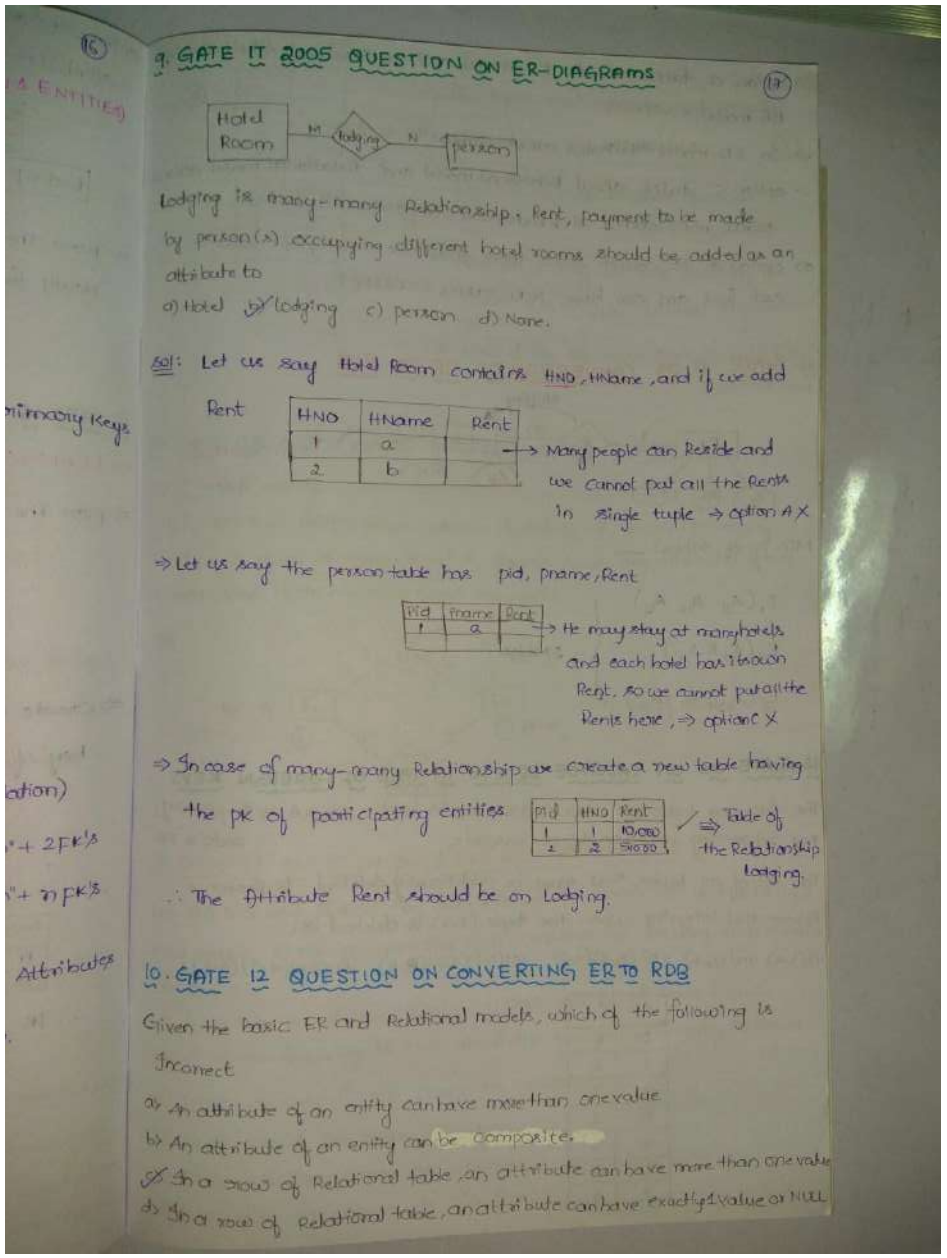
Find the number of foreign keys required for above ER diagram _____.

7

Solution :
7

- For relation E_1 – R_1, A is foreign key so, only 1 foreign key for this.
- For relation E_1 – E_2, R_2 is many to many relationship so, 2 foreign keys required i.e. A and C.
- For relation E_1 – E_2, R_3 is one to many so, only 1 foreign key required.
- For E_1 - E_2, R_4 is many to many therefore 2 foreign keys required.
- For E_3 - R_5 only 1 foreign key is required.

Therefore total foreign keys = 1 + 2 + 1 + 2 + 1 = 7



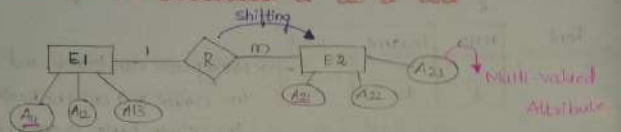
a) option a talks about ER-model and multivalued Attributes in ER-model - CORRECT

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: 3

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

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

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

} 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 and C is FK referencing with A on-delete cascade.

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

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

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

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

14. GATE OB QUES

E_1E_2 - two entities

R_1R_2 are two Rel

and R_2 is many t

own. What is the

Sol

PK: A_1 E

15. GATE 97 QU

Let R(a,b,c) and

that refers to

R and S:

a) Insert into R

which of the f

a) None of

b) All of a,b,c

c) Both a,b

d) Both b,c

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) (1)

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

⇒ Now on deleting we are deleting the rows (5,2) (9,2) because they contain '2'

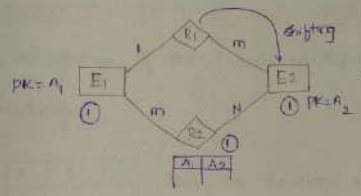
⇒ Now delete the rows containing (5)(7) = (9,5) deleted.

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 doesnot have any attributes of their own. what is the min no of tables required?

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:

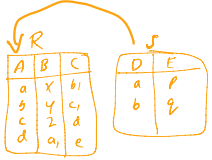
Which of the following is true about the referential integrity constraint above?

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.



Q. 8

? FAQ ▶ Solution Video Have any Doubt ?

For two relations $R(A, B, C)$ and $S(D, E)$, relation S maintain a foreign key for D on attribute A of relation R. Consider the following statements:
(a) Each record of R is related to 0 or more record of S.
(b) Each record of S is related to 0 or more record of R.
(c) Each record of S is related to 0 or 1 record of R.
(d) Each record of R is related to 0 or 1 record of S.
Which of the following is/are true?

A a — Your option is Correct

B b — Your answer is IN-CORRECT

C c — Correct Option

D d

YOUR ANSWER - a,b | CORRECT ANSWER - a,c | STATUS - ✗

Solution :

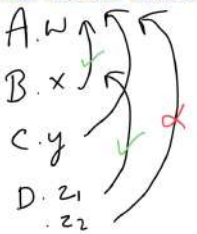
(a, c)
Option (a) is correct because relation S is allowed to take only those value which are part of relation R. Hence, R is related to 0 or more record of S. Option (c) is correct, since relation S is going to have only those values which are part of R. Hence, it is related to only 1 value of R. But the field with NULL is related to 0 value of R.
Hence, option (b) and (d) are false.

QUESTION ANALYTICS +

Understood

Q.18) Negative Marks: 0.66 MCQ Subject: DBMS Max Marks: 2

Suppose we have the following table declarations:
Create table A(w int PRIMARY KEY);
Create table B(x int PRIMARY KEY
References A(w) on delete set NULL);
Create table C(y int references A(w));
Create table D(z1 int references B(x) on delete set NULL,
Z2 int references A(w) on update cascade);
Consider the following scripts:
I. Delete from C; Delete from B; Delete from A; Delete from D;
II. Delete from C; Delete from D; Delete from A; Delete from B;
III. Delete from B; Delete from C; Delete from D; Delete from A;



Which of the above scripts will empty all four tables, without error?

Hide Answer | Add Notes | View Notes | Show Comments

A III only

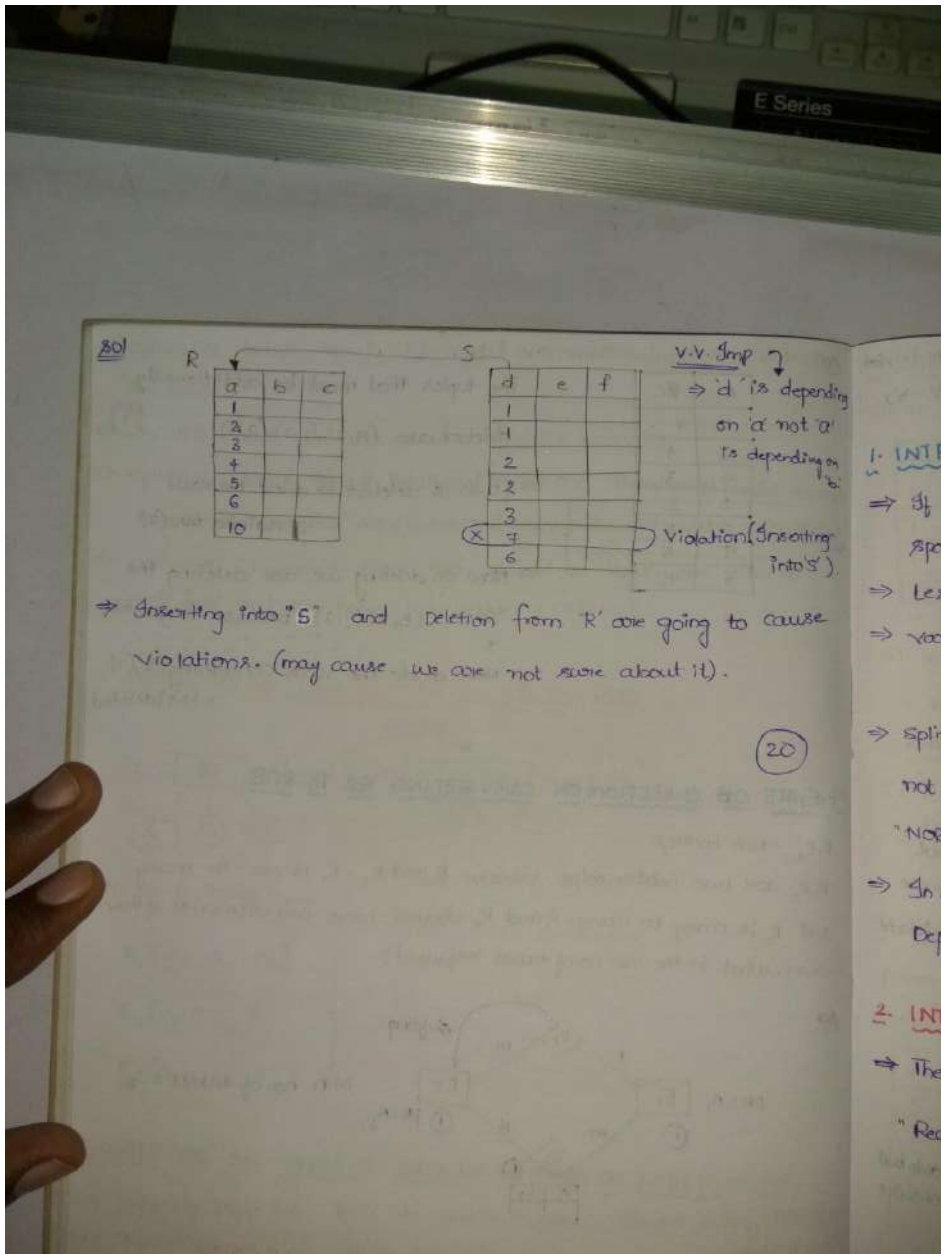
B I only

C II and III only — Correct Option

Solution: (C)

The foreign-key constraint from B to A doesn't cause problems if we delete from A first, since there is a clause that tells how to handle dangling tuples in B. However, the foreign-key constraint from C to A is a problem, since the default policy is to reject a deletion from A that causes a dangling tuple in C. Thus, in a valid sequence, C must precede A. The foreign-key constraint from D to B is not a problem, but the one from D to A is, again because the default policy will cause a rejection of certain deletions from A. Thus, D must also come before A. Of the three sequences, only I script has D after A, therefore, I is not the valid script os operations. While II and III are valid.

D I and III only



Chapter 1: ER Diagrams GATE Questions

QUESTION

1. ER Model
2. Relational model
3. Conversion from ER Model to Relational model
4. No. Of tables, and its normalisation from ER model
5. On Delete/Insert cascade
6. Attributes shifting in M:N / 1:M / 1:1 Relationship
7. Find Index/Candidate Key

ANSWER

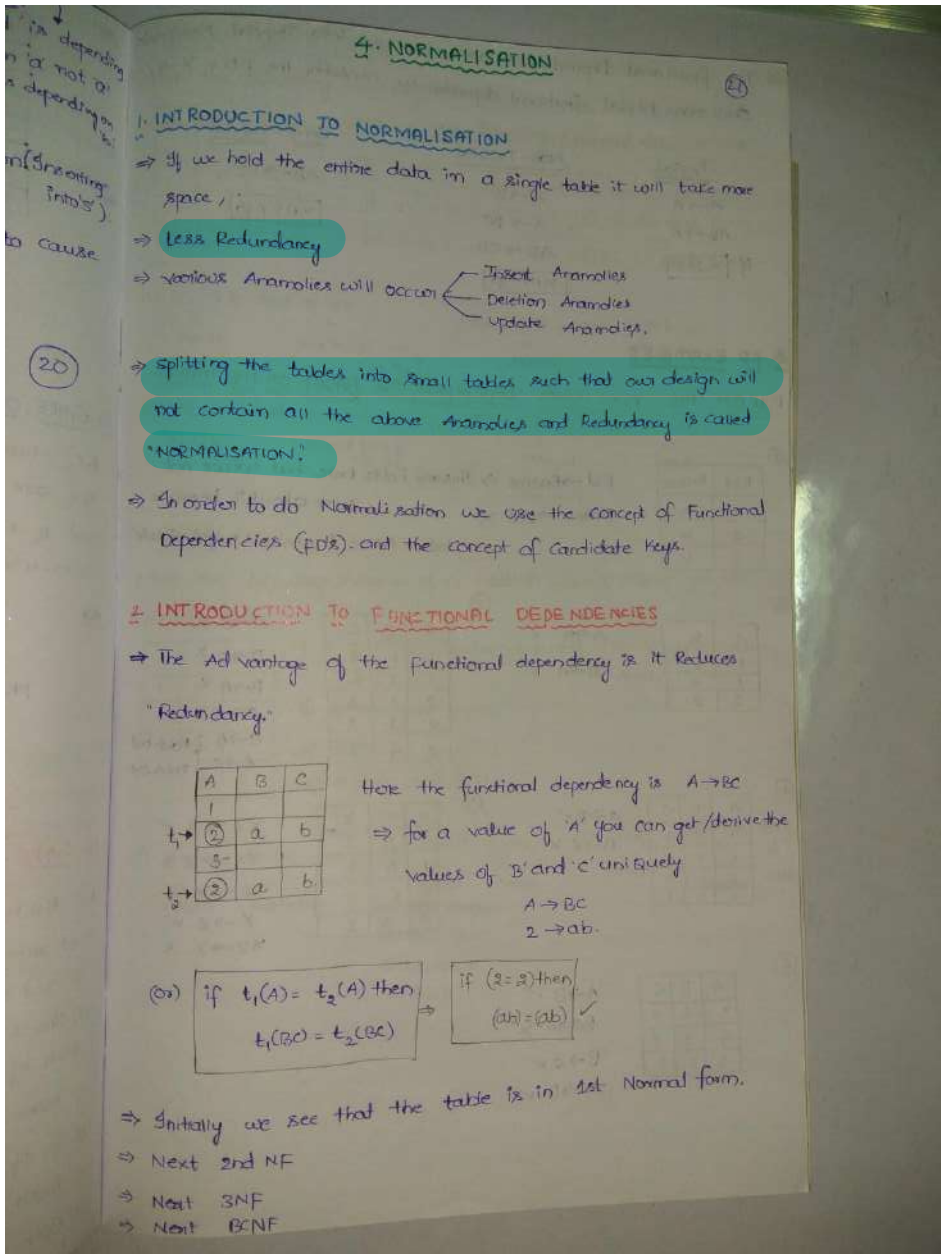
Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

Normalization

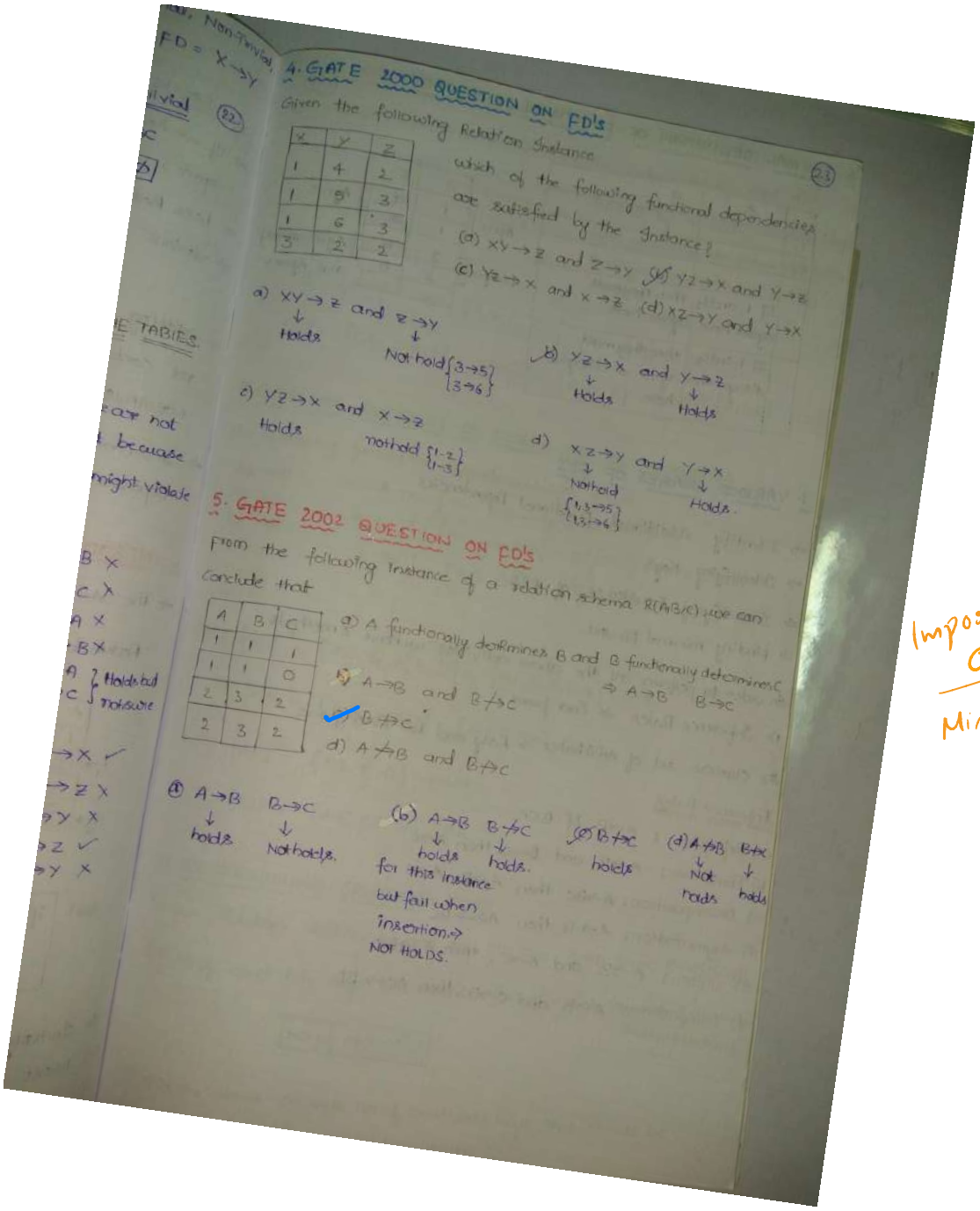
If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- **Update anomalies** – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** – We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** – We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.



The normalisation is done to increase the redundancy of data due to non-key dependencies of the attributes in relation



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 → Z and Z → Y (b) YZ → X and Y → Z

(c) YZ → X and X → Z (d) XZ → Y and Y → X

a) XY → Z and Z → Y
↓ Holds ↓ Not hold {3→5, 3→6}

b) YZ → X and Y → Z
↓ Holds ↓ Holds

c) YZ → X and X → Z
Holds not hold {1→2, 1→3}

d) XZ → Y and Y → X
↓ Not hold {1,3→5, 1,3→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 ⇒ A → B B → C

b) A → B and B → C

c) B → C

d) A → B and B → C

(a) A → B (B → C ↓ holds) ↓ holds. B → C ↓ Not holds.

(b) A → B (B → C ↓ holds) ↓ holds. for this instance but fail when insertion → NOT HOLDS.

(c) B → C holds

(d) A → B ↓ Not holds. B → C ↓ holds

Important Concept

Mistake in ACE Book Mcq.

6. FORMAL DEFINITION OF FUNCTIONAL DEPENDENCY

(24)

	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 $A \to BC$

f) Composition: $A \to B$ and $C \to D$, then $AC \to BD$.

8. CLOSU

Functional

Here, A^+

B^+

$(CD)^+$

9. GATE

The follo

$AB \to CD$

$AF \to D$

$DE \to F$

$C \to G_1$

$F \to E$

$G_1 \to A$

Sol) A) fo

B)

c)

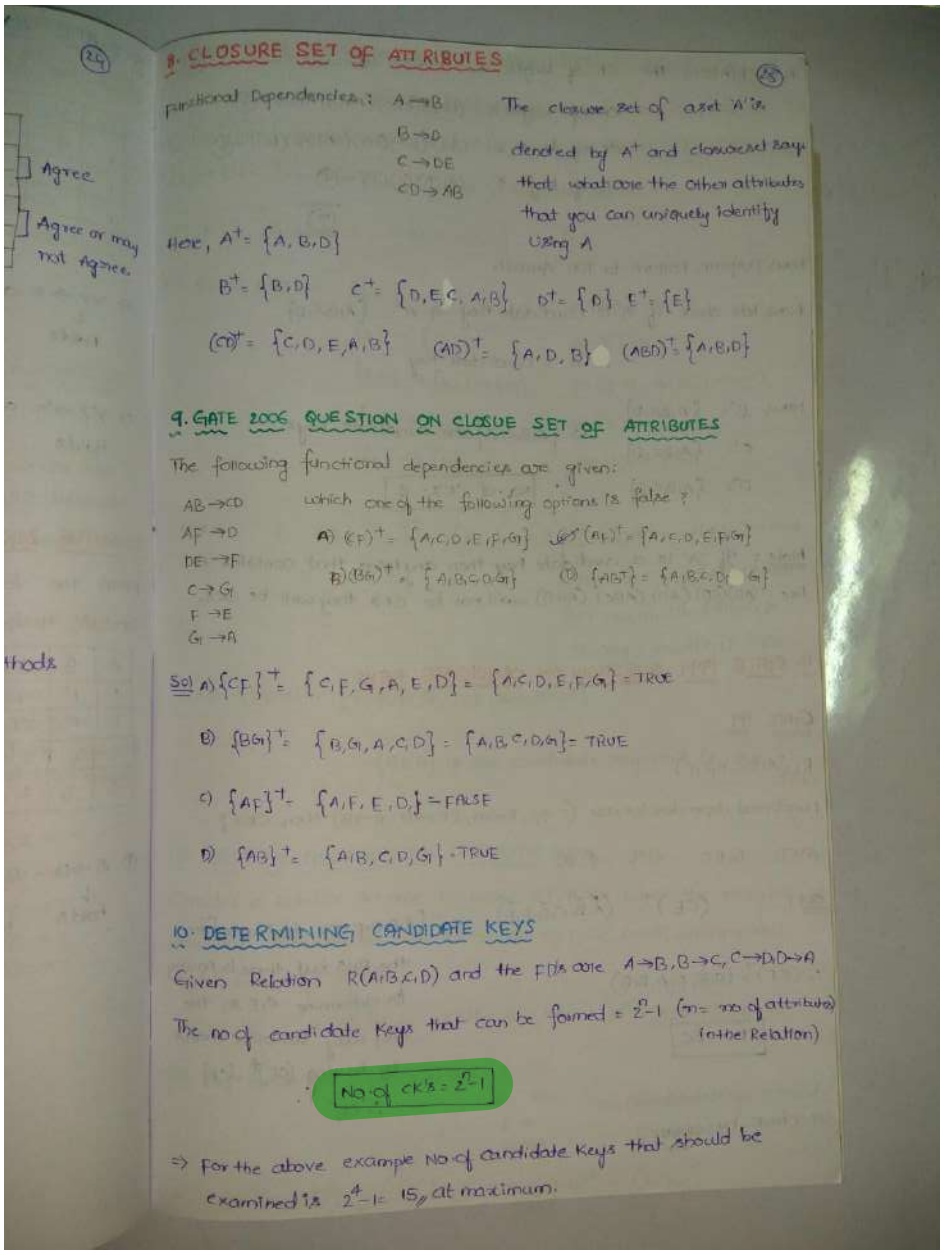
D)

10. DE

Given

The no

=> Fo



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 CLOSURE 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) $(CF)^+ = \{A, C, D, E, F, G\}$ B) $(BG)^+ = \{A, B, C, D, G\}$ C) $(AF)^+ = \{A, C, D, E, F, G\}$ 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 FDs 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)

No. of CK's = $2^n - 1$

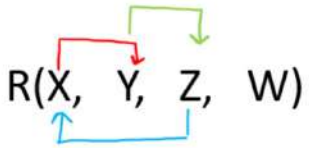
⇒ For the above example No. of candidate keys that should be examined is $2^4 - 1 = 15$, at maximum.

javatpoint.com

SHORTCUT TO FIND CANDIDATE KEY

Example 1: Give R(X, Y, Z, W) and Set of Functional Dependency FD = { X → Y, Y → Z, Z → X}. The question is to calculate the candidate key and no. of candidate key in above relation R using a given set of FDs.

Let us construct an arrow diagram on R using FD



From the above arrow diagram on R, we can see that an attribute W is not determined by any of the given FD, hence W will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have W compulsory attribute.

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} ⇒ {A,B,C,D} are candidate Keys

D^+ = {A,B,C,D} ∴ 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)

{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

∴ $(CF)^+$ = (C,E,F,A,D,B)

∴ CK = EC

or check by opvm

12. GATE 20

Consider th

of function

{K} → {M},

A) {E,F}

Sol) Given F

Now,

Now, (E

13. GATE

Consider a

functional

candidate

a) AE, BE

Sol) The fo

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}→{G}, {F}→{I,J}, {E,H}→{K,L}→{M}, {K}→{M}, {L}→{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} → {G} ⇒ (EF) → (G)

{F} → {I,J} ⇒ (F) → (IJ)

{E,H} → {K,L} → {M} ⇒ (EH) → (KL) → (M)

{K} → {M} ⇒ (K) → (M)

{L} → {N} ⇒ (L) → (N)

Now, (EFH)$^+$ = (EFGHIJKLMN) * The attributes that can be derived from this are (GIJKLMN) and there is no way I could get (EFH)

Now, (EFH)$^+$ = {E,F,H,G,I,J,K,L,M,N}

∴ {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→B, BC→D, E→C, D→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→B, BC→D, E→C, D→A. Candidate Keys should contain EH.

⇒ (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 Given Included (AEH) ∴ Total keys = 2^3]

$(BEH)^+ = \{A,B,C,D,E,H\}$ ✓ → 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

$CH \to G$

$A \to BC$

$B \to CFH$

$E \to A$

$F \to EG$

$(\because D \to \_)^+ = (ABCDEFGH)^+$

⇒ 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

④ 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 superkeys = 2^4 = 16

⑤ 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

⑥ 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,

$A^+ = (ABCDE$

$B^+ = (B)$ ×

$C^+ = X$

$D^+ = X$

$E^+ = X$

$F^+ = X$

20. EXAM

R = (ABCDE

FD = {A→B

Now,

$(E)^+ = \{E\}$

21. EXA

R(ABCD

A → BC

CD → E

B → D

E → A

19. EXAMPLES ON CANDIDATE KEYS -5

R (ABCDEF)

$A \rightarrow BCDEF$

$BC \rightarrow ADEF$

$DEF \rightarrow ABC$

Now,

$A^+ = (ABCDEF)$ ✓

$B^+ = (B)$ ×

$C^+ = X$

$D^+ = X$

$E^+ = X$

$F^+ = X$

Now,

$(BC)^+ = ABCDEF$

Now,

$(DEF)^+ = (DEFABC)$

∴ Candidate keys = A, BC, DEF

20. EXAMPLE ON CANDIDATE KEY - 6

R = (ABCDE)

FD = {A → B, B → C, C → D, D → A}

Now,

$(E)^+ = \{E\}$

Now,

$(AE)^+ = (AEBCD)$ ✓

$(BE)^+ = (BECDA)$ ✓

$(CE)^+ = (CEDAB)$ ✓

$(DE)^+ = (DEABC)$ ✓

∴ The CK's are : AE, BE, CE, DE

No. of CK's = 4

Now,

(ABEF) X

21. EXAMPLE ON CANDIDATE KEY-7

R(ABCDE)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

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 $R_1(BCD)$

$B^+ = B$

$C^+ = C$

$D^+ = DA$

$(BC)^+ = BC$

$(BD)^+ = CDAB$

$(CD)^+ = CDA$

∴ candidate Key = BD

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 $R_1(DEF)$

$D^+ = \{DEF\}$ ✓

$E^+ = \{EDF\}$ ✓

$F^+ = \{F\}$

∴ The candidate keys are DE

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 $R_1(ABCE)$

$A^+ = ABCDE$

$B^+ = BD$

$C^+ = C$

$E^+ = EABCD$

Now,

BC = BCDEA ✓

BD ✗ (D not intable)

CD ✗ (D not intable)

candidate keys are, A, E, BC

25. CHECKING

⇒ Given the se

functional Dep

Inference Rule

R(AB) and the

X	Y
ϕ	ϕ
A	A
B	B
AB	AB

⇒

The no of Add

ϕ→ϕ ✓

×ϕ→A ✓

×ϕ→B ✓

×ϕ→AB ✓

The no of FD's

26 CHECKING

R(ABC)

FD's: {A→B, B→

X → Y

$2^3 \times 2^3 = 8\times$

ϕ→ϕ (only

A → {ABC} =

B → {BC} =

C → {C} =

AB → $(AB)^+$ = {A

Refer Applied Video + formula

(52)

ation $R_1(BCD)$?

25. CHECKING ADDITIONAL FD's - EXAMPLE 1

(53)

⇒ 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 $4 \times 4 = 16$

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

X	Y
↓	↓
2^3	2^3

$\times = 8 \times 8 = 64$ 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 $B^+ = \{BC\} = 2^2$

$C \to \{C\}$ = 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

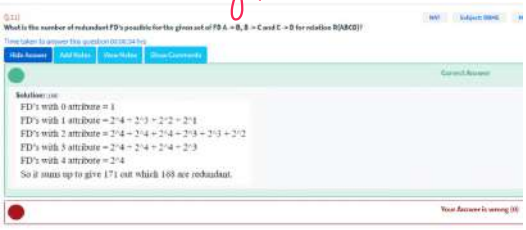
$(BC)^+ = (BC)^+ = \{BC\} = 4$ FD

$AC \to (AC)^+ = \{ABC\} = 8$ FD's

$(ABC)^+ = \{ABC\} = 8$ FD's

⇒ Total valid fd's $= 1+8+4+2+8+4+8+8 = 43$

How?



∴ In General the no of functional dependencies that are possible over a Relation R containing 'n' attributes is 2^{2n} // $X \to Y$ $2^n \quad 2^n \Rightarrow 2^n \times 2^n = 4^n$

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 $A \to B$, $A \to C$, $CD \to E$, $B \to D$, $E \to A$ which of the following FD's is NOT implied by above set?

a) $CD \to AC$ b) $BD \to CD$

c) $BC \to CD$ d) $AC \to BC$

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

$F: \{A \to C, AC \to D, E \to AD, E \to H\}$

$G_1: \{A \to CD, E \to AH\}$ are both the functional dependencies Equivalent

Sol First check if 'F' covers 'G_1' i.e $\boxed{F \supseteq G_1}$ 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 $G_1: \{A \to CD, E \to AH\}$

F: ↓ covered by 'F' ↓ covered by 'F'

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

$E^+ = \{EADCH\}$

$F \supseteq G_1$ is TRUE

Now find G_1

F: $\{A \to C$ ✓

G_1:

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

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

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

29. EQUIVALE

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

G_1: $\{A \to BC,$

Sol

$F \supseteq G_1 \Rightarrow$ Ta

$\Rightarrow A$

$\Rightarrow$

$G_1 \supseteq F \Rightarrow$ T

in G_1:

Bo

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:

$A^+ = \{A, C, D\}$ ∴ all the functional dependencies in F are covered by G_1

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

$E^+ = \{E, A, H, C, D\}$ ∴ Both the FD's F and G_1 are Equivalent.

EQUIVALENCE OF FD's EXAMPLE-1

$F: \{A \to B, B \to C, C \to D\}$ check whether these two FD's are Equivalent or not?

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

Sol

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

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

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

⇒ $C \to D$ ⇒ $C^+ = \{D, C\}$ holds ∴ $F \supseteq G_1$

$G_1 \supseteq F$ ⇒ 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 $G_1\}$

∴ $G_1 \not\supseteq F$

∴ Both the functional dependencies are not Equivalent.

30. EQUIVALENCE OF TWO FD'S EXAMPLE-2

① $F: \{A \to B, AB \to C, D \to AC, D \to E\}$
$G_1: \{A \to BC, D \to AB\}$

② $F: \{A \to B, B \to C, C \to A\}$
$G_1: \{A \to BC, B \to A, C \to A\}$

Sol: $F \supseteq G_1 \Rightarrow A^+$ in $F = \{A, B, C\}$ ✓
$\Rightarrow D^+$ in $F = \{D, E, A, C, B\}$ ✓ $\quad F \supseteq G_1$

$G_1 \supseteq F \Rightarrow A^+$ in $G_1 = \{A, B, C\}$ $\quad A \to B$ ✓
$\Rightarrow (AB)^+ = \{A, B, C\}$ $\quad AB \to C$ ✓
$\Rightarrow D^+ = \{A, B, C, D\}$ $\quad D \to AC$ ✓ $\quad \therefore G_1 \not\supseteq F$
$\Rightarrow D^+ = \{A, B, C, D\}$ $\quad D \to E$ ✗

∴ These two FD's are not Equivalent

②
$F \supseteq G_1 \Rightarrow A^+$ in $F = \{A, B, C\}$ $\quad A \to BC$ ✓
$\Rightarrow B^+$ in $F = \{B, C, A\}$ $\quad B \to A$ holds ✓ $\quad \therefore \boxed{F \supseteq G_1}$
$\Rightarrow C^+$ in $F = \{A, B, C\}$ $\quad C \to A$ holds

$G_1 \supseteq F \Rightarrow$ Now, A^+ in $G_1 = \{A, B, C\}$ $\quad A \to B$ ✓
Now, B^+ in $G_1 = \{B, A, C\}$ $\quad B \to C$ ✓ $\quad \therefore \boxed{G_1 \supseteq F}$
Now, C^+ in $G_1 = \{C, A, B\}$ $\quad C \to A$ holds ✓

∴ 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_1 such that 'G_1' covers 'F' and 'F' covers 'G_1' and 'G_1' is minimal, then 'G_1' is called Minimal cover of 'F'.

PROCEDURE TO

1. split the F
Ex: $A \to BC$

2. Find the Re
Ex: $\{A \to B,$

3. Find the Red
Ex: $AB \to C$

i) Minimise $\{A \to$
① $A \to C$, A

② $A \to C$, A

∴

⇒

32. MINIMA

Minimise $\{A$
① $A \to B$

Now, (A

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' $\Rightarrow A \to C$

i) Minimize $\{A \to C, AC \to D, E \to AD, E \to H\}$

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.

$E^+ = \{A, E, H, C, D\}$ ⇒ "$E \to D$" 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,

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

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

⇒ $A \to CD$, $E \to AH$

32. MINIMAL COVER EXAMPLE-1

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

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

Redundant ⇒ $D^+ = \{D, A, B\}$ $D \to B$ 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, $A^+ = \{A, B\}$ ∴ $AC \to D$ 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: $AB \to C$ $D \to E$ $AB \to E$ $E \to C$

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

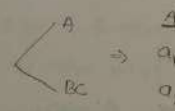
∴ $\{AB \to C, D \to E, E \to C\}$ is not the minimal cover, The minal cover should be $\{D \to E, AB \to E, E \to C\}$

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_1	b_1c_1
a_2	b_2c_2

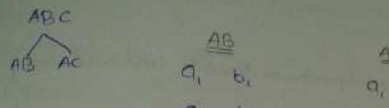
↓ cross product

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_3

→ New Tuple.

→ Spurious tuple

Now if I choose a common Alphabet Randomly:



AB

a_1	b_1
a_2	b_1
a_1	b_2

AC

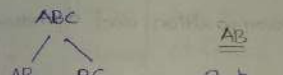
a_1	c_1
a_2	c_1
a_1	c_2

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 likewise not present in the given table.



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

LOSSLESS DECOMPOSITION

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

∴ For Loseless decomposition.

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

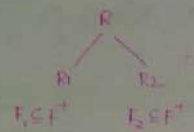
$(R_1 \cap R_2) \rightarrow R_2$ | $(R_1 \cap R_2) \rightarrow R_2 - R_1$

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

35. FD PRESERVING

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

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



Now, dependency preserving means

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

⇒ The dependency preserving is not a mandatory thing

36. DECOMPOSITION - EXAMPLE 1

R(ABC), FD: {A→B, B→C, C→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) {∵ B→C}.

Now, R_1(AB)

✓ A→B

B→A ✓

Non-Trivial Dependencies

R_2(BC)

✓ B→C

✓ C→B

} are the Non-trivial Dependencies.

$B^+ = \{BCA\}$ (B→C)✓

$C^+ = \{CAB\}$ (C→B)✓

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

$= A \to B,\ B \to A,\ A \to C,\ C \to B$ (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,

D = (AB, BC, CD)

Sol Given D=

LOSELESS

DECOMPOSITI

Now comming

R_1(AB) | (BC)

✓ A→B | B→

✓ B→A | C→B

$B^+ = \{ABCD\}$ | $C^+ = \{A$

Now

38. DECOMPO

R(ABCD)

F = {AB→CD,

D = {AD, BCD

Sol the com

the Relo

∴ The

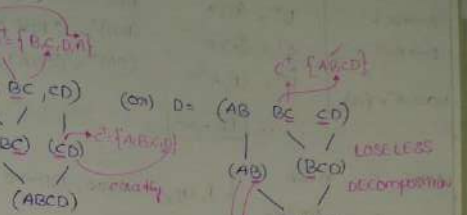
32. DECOMPOSITION EXAMPLE 2

R(ABCD)

F = {A→B, B→C, C→D, D→A}

D = {AB, BC, CD}

Sol: Given D = (AB, BC, CD)



(OR) D = (AB, BC, CD)

Now comming to functional dependencies

$R_1(AB)$	$R_2(BC)$	$CD = R_3$
✓ A→B	B→C ✓	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}

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

33. 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) | R_2 (BCD)

$A \to D$ ✗
$D \to A$ ✓

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

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

$(BC)^+ = (BC)$ ✗
$(BD)^+ = (BDAC)$ ✓
$(CD)^+ = (CDA)$ ✗

$\therefore$ $BD \to C$

Now. $F_1 = D \to A$
$F_2 = BD \to C$

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

∴ 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, $E^+ = \{E, G\}$

∴ $AB \to C$ is possible ⇒

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_1: (ABCD) R_2: (DE).

Now Given (ABCD) (DE)

∴ ABCD

A→BCD ✓

B→BX

C→D ✓

D→DX

(BC)→D.

Now A^+ = ABCDE

B^+ = B

C^+ = CDE

D^+ = D, E

$(BC)^+$ = BCDE

$(BD)^+$ = BDE

$(CD)^+$ = CDE

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

Now, D^+ = {E, D}

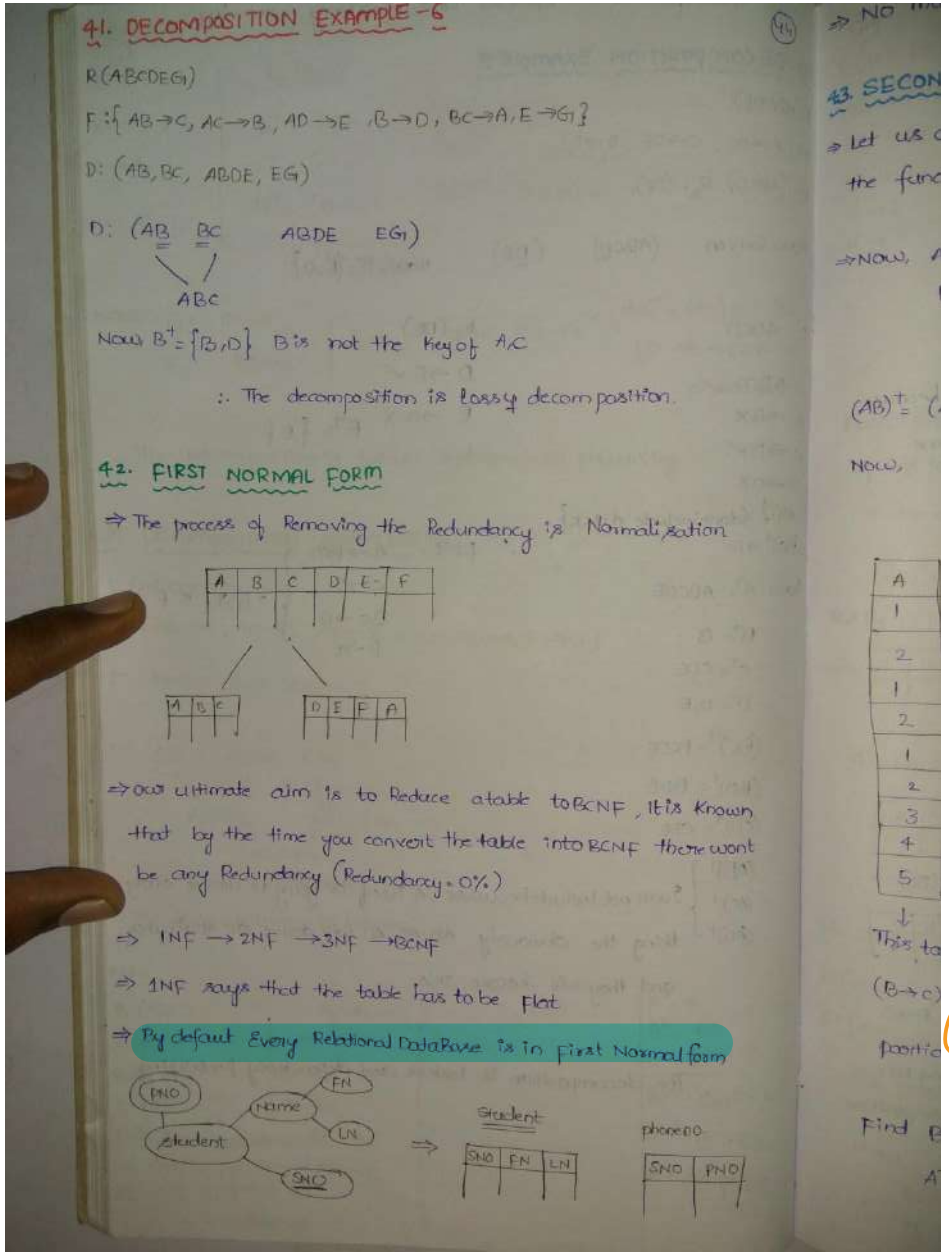
R_2 (DE)

D→E ✓

E→D X 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 AC

∴ The decomposition is lossy decomposition.

42. FIRST NORMAL FORM

⇒ The process of Removing the Redundancy is Normalization

A	B	C	D	E	F

A	B	C

D	E	F	A

⇒ Our ultimate aim is to Reduce a table to BCNF, It is known that by the time you convert the table into BCNF there won't be any Redundancy (Redundancy = 0%)

⇒ 1NF → 2NF → 3NF → BCNF

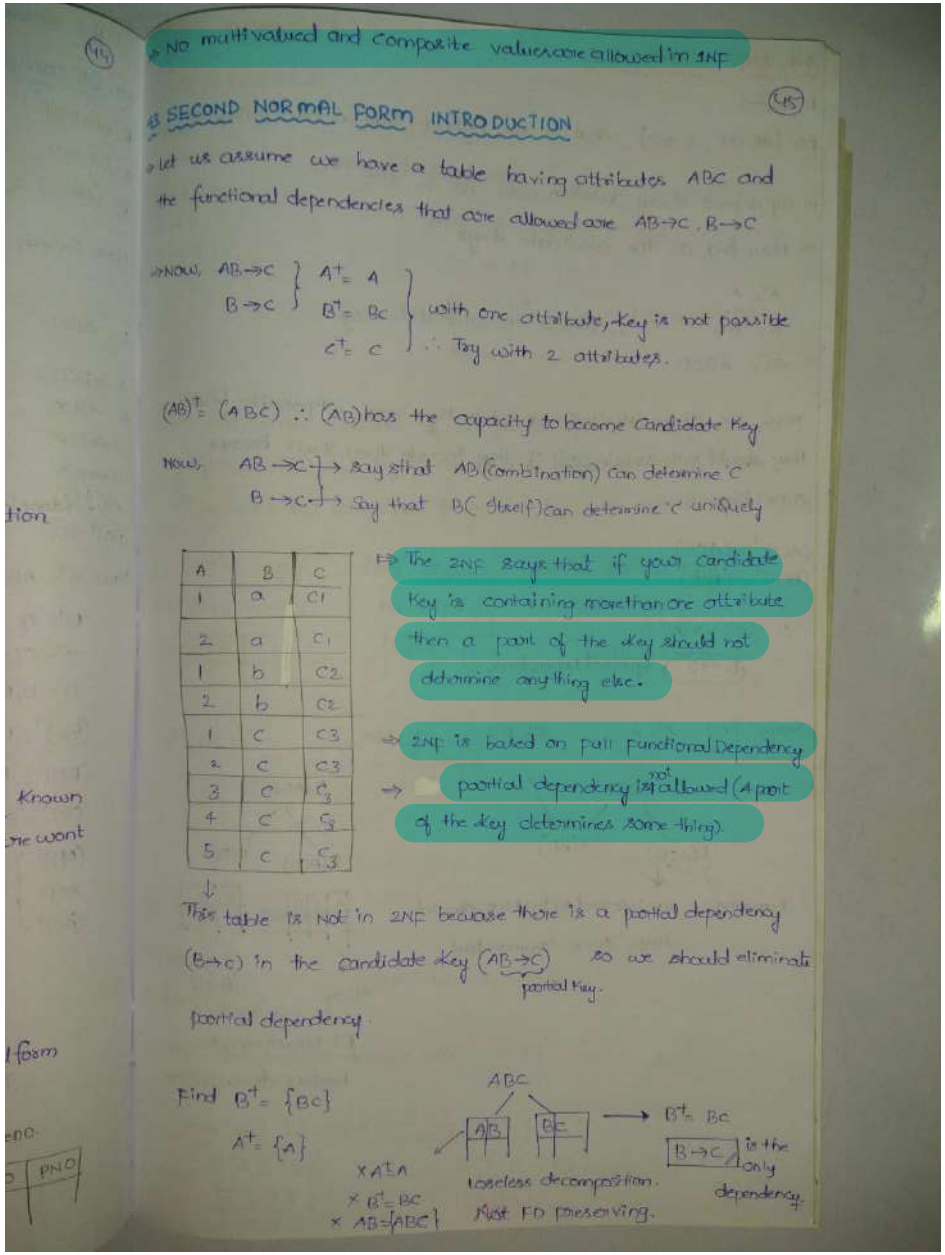
⇒ 1NF says that the table has to be Flat

⇒ By default Every Relational DataBase is in First Normal form

1 NF

Flat table

Atomic Value



- 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→C, B→C

⇒ Now, AB→C, B→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→C → say that AB (combination) can determine C
B→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→C) in the candidate key (AB→C) (partial key) so we should eliminate partial dependency.

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

2NF

No Partial dependency alloweth

Royal People should not meet citizen (non-royal) alone

- Only family visit allowed
- Royal Meeting royal is fine

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.

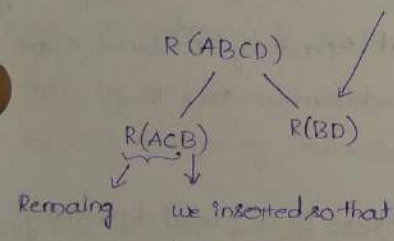
$(ACD)^+ = (ACD)$

$(BCD)^+ = (BCD)$

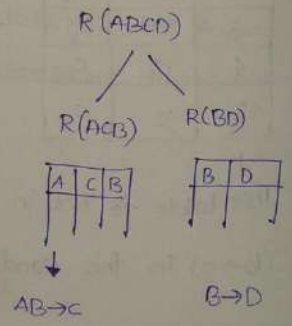
∴ (AB) → ABCD

B → D → partial Dependency

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



Remaing we inserted, so that we should have some common part



FD preserving too.

Lossless decomposition.

45. 2NF

R(A B C D E

AB → C

BD → EF

AD → GH

A → I

H → J

Now, find

of left h

all the c

$(AB)^+ =$

R(A B C D E F

AB→C | A | B

$A^+ = (A, I$

A | I

A → I

46. 2NF

R(ABCDE)

F: {A → B, B

The par

Now,

R(

fied?

and

ome

R(BD)

B	D

B→D

ving too.

decomposition

45. 2NF EXAMPLE 2

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

AB → C
BD → EF
AD → G H
A → I
H → J

Candidate Key = ABD.

(ABD) → (A B C D E F G H I J)

- BD → EF
- AD → G H
- A → I
- AB → C

} 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 D E F G H I J)

A	B	C

AB → C

$(BD)^+ = (BDEF)$

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

B	D	E	F

BD → EF

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

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

A	D	G	H	I

AD → G H
H → J

$A^+ = (A, I)$

A	I

A → I

ABD

A	B	D

Required as ABD is a Key

46. 2NF EXAMPLE 3

R(A B C D E)

Candidate Key = (AC)

F: {A → B, B → E, C → D}

AC is the Key.

The partial dependencies are A → B
C → D

AC

A	C

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

R(A B C D E)

A → B
B → E

A	B	E

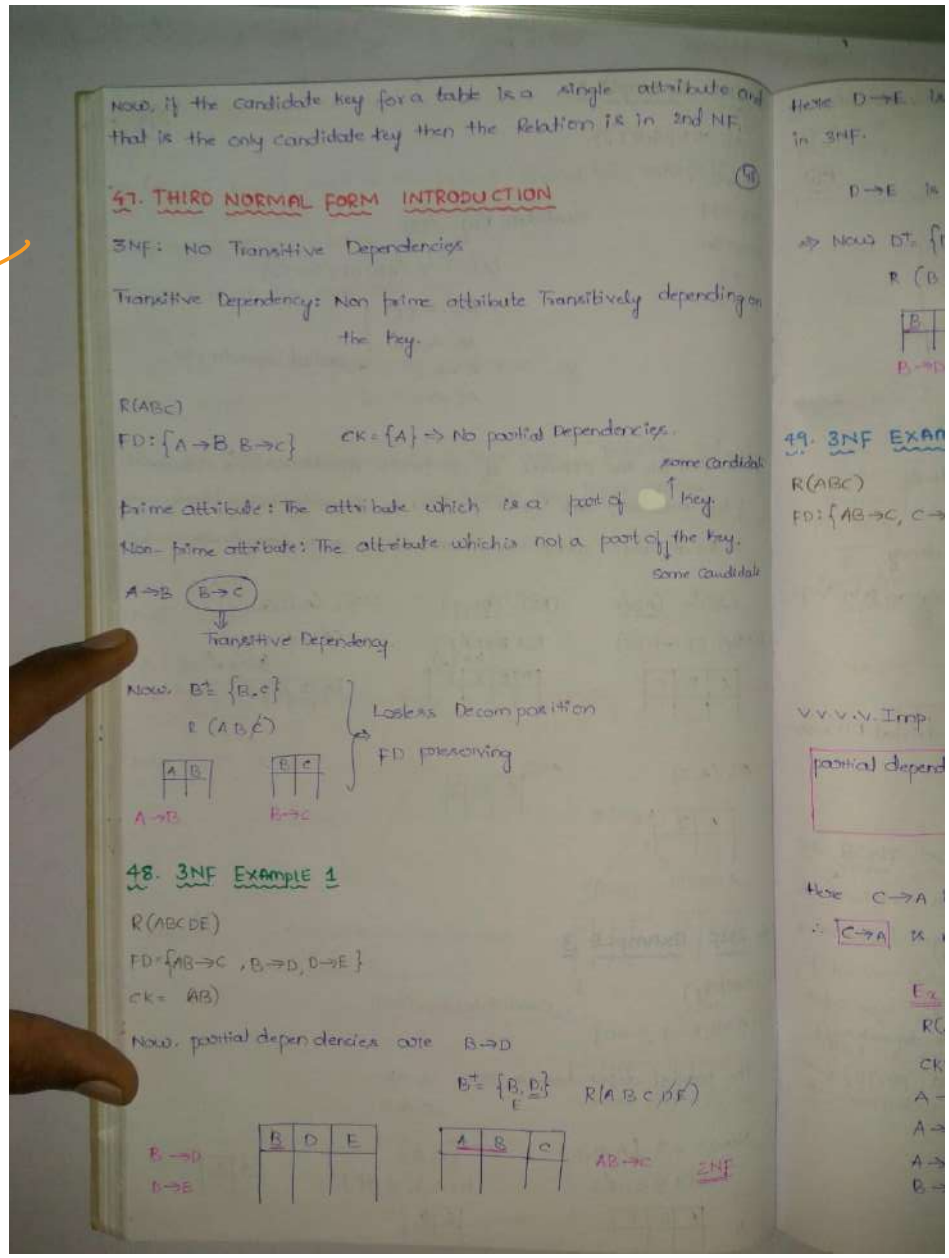
$C^+ = \{C, D\}$

R(A B C D E)

C → D

C	D

- Lossless, dependency preservating decomposition into 3NF is always possible



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

D→E is the Transitive Dependency present in table BDE

⇒ Now $D^+ = \{D, E\}$

R (B D F)

B	D
B→D	

D	E
D→E	

A	B	C
AB→C		

3NF

49. 3NF EXAMPLE 2

R(ABC)

FD: {AB→C, C→A}

Now, Candidate key = (B) = (ABC)

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

$(AB)^+ = \{ABC\}$ ✓ ∴ Candidate keys

$(BC)^+ = \{BCA\}$ ✓ = (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→A is not partial dependency because C→ prime attribute (A)

∴ C→A is not partial dependency.

Ex:

R(ABCDEF)

CK: ABC

A→D = partial dependency.

A→B, A→C, B→C = Not partial dependencies

D→E

E→F

D→C

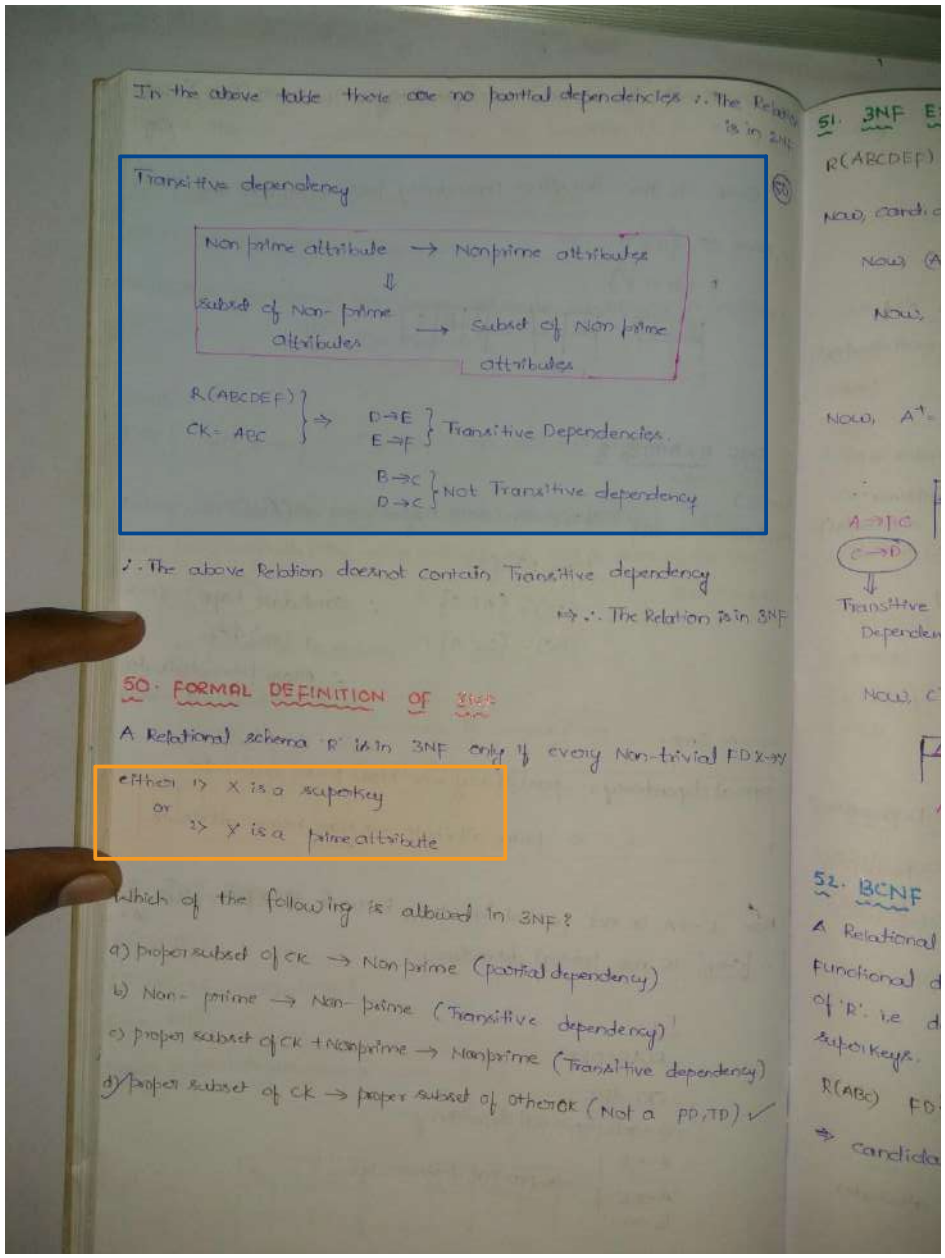
Something wrong in the question

3 NF

Non-Royal$_1$ → Non-Royal$_2$

Non-Royal$_2$ → Non-Royal$_3$

Excess talking leads to back bitching

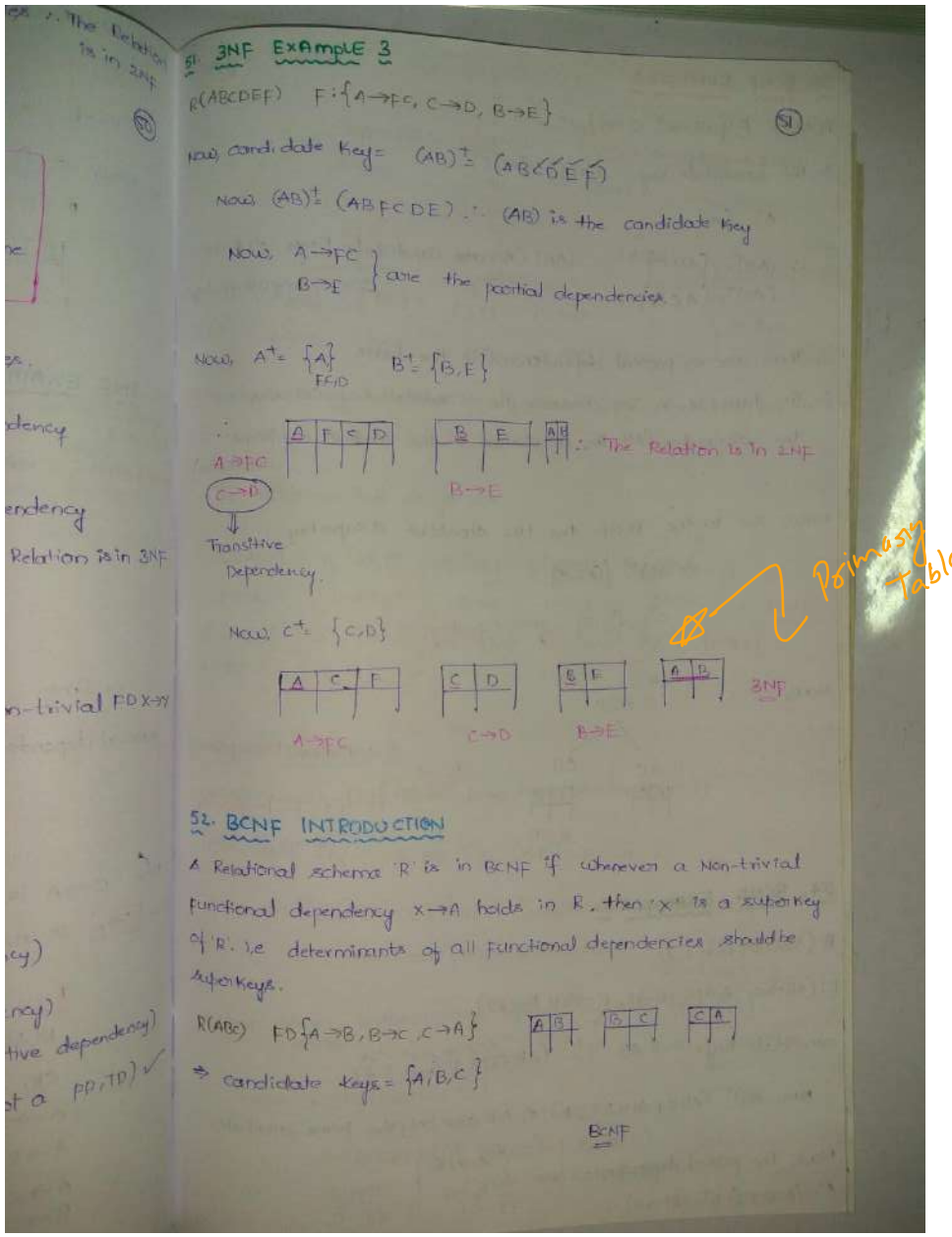


BCNF

Super Key ⟶ talks to ⟶ Royal Member

only this is allowed

NOTHING ELSE



51. 3NF Example 3

R(ABCDEF) F:{A→FC, C→D, B→E}

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

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

Now, A→FC, B→E } are the partial dependencies

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

A→FC | B→E | AB : The Relation is in 2NF

C→D

Transitive Dependency.

Now, C+ = {C, D}

Primary Table

A→FC | C→D | B→E | AB | 3NF

52. BCNF INTRODUCTION

A Relational schema 'R' is in BCNF if whenever a Non-trivial functional dependency X→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→B, B→C, C→A} | AB | BC | CA

⇒ Candidate keys = {A, B, C}

BCNF

53. BCNF EXAMPLE 1

(53)

R(ABC) F: {AB → C, C → B}

⇒ The candidate key = $(A)^+ = (ABC)$

$A^+ = \{A\}$

$\therefore (AB)^+ = \{ABC\}$ $\quad$ $(AC)^+ = \{ACB\}$ $\quad$ (AB) (AC) are candidate keys ⇒ A, B, C are prime attributes

∴ There are no partial dependencies in the table

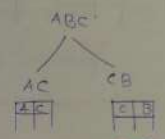
∴ The table is in 3NF because the candidate key contains all the attributes (All the attributes in the Relation are prime attributes)

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 → C is lost)

54. BCNF EXAMPLE 2

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

F: (AB → C, A → DE, B → F, F → GH, D → IJ)

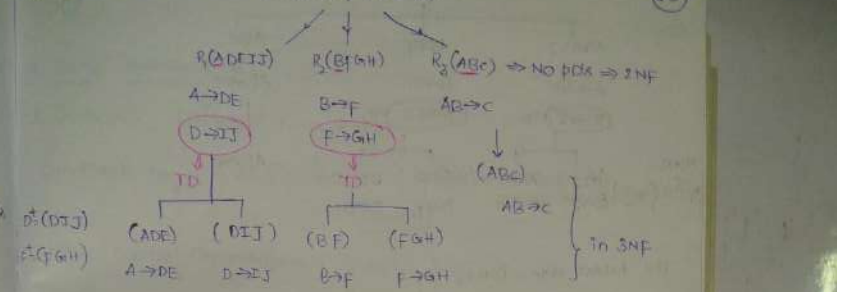
candidate key = $(AB)^+ = (ABCDEFGHIJ)$

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

Now, the partial dependencies are $\{A \to DE, B \to F\}$

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

Now the Relation R(ABCDEFGHIJ)



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

∴ The Relations that we get are

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

55. BCNF Example 1

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

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

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

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

$B^+ = \{BDEF\}$

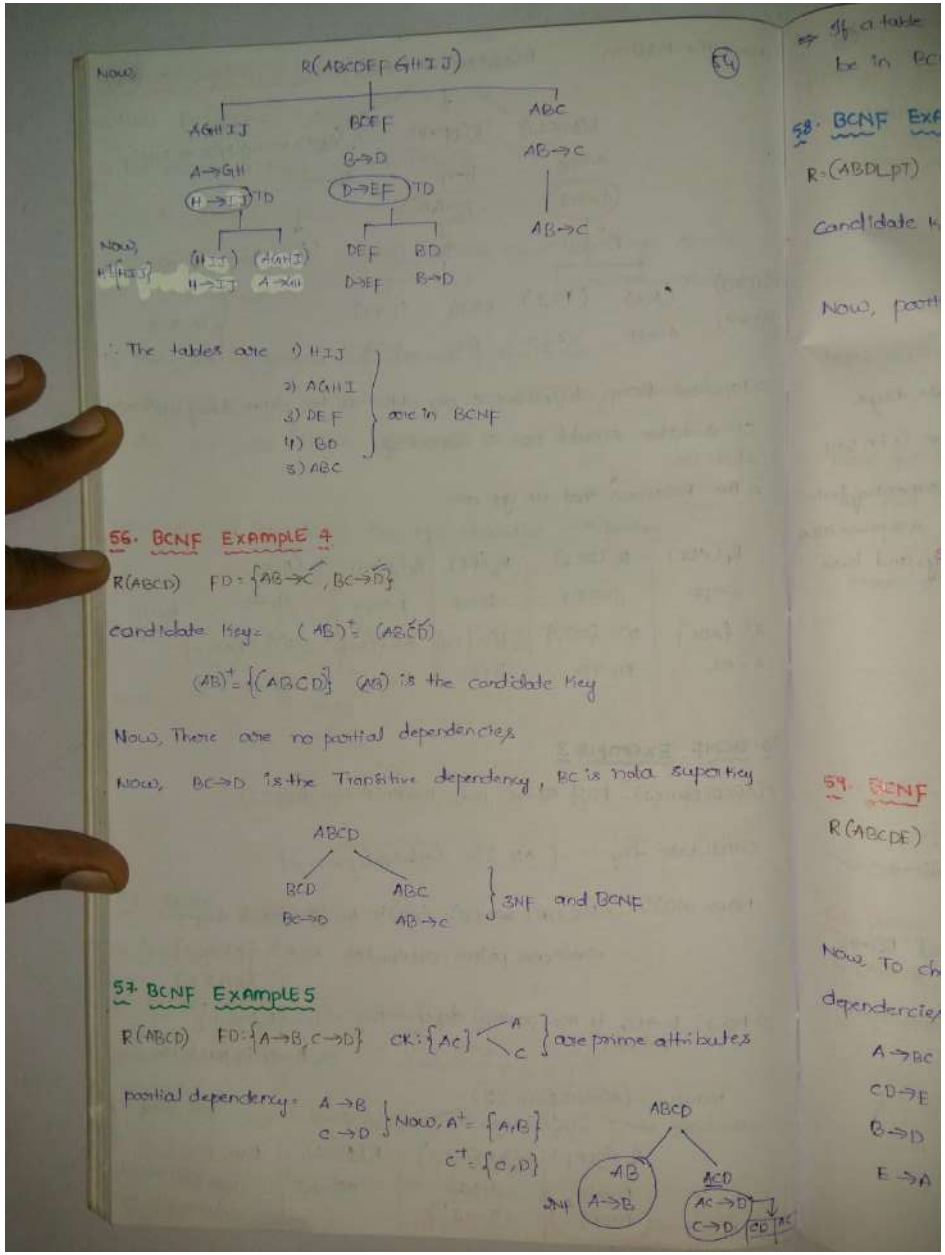
⇒ Now B→D, is the partial dependency ⇒ $B^+ = \{B,D,E,F\}$

A→GH B→D is possible

Now, (ABCDEFGHIJ)

$R_1(BDEF)$	$R_2(AGHIJ)$	$R_3(ABC)$	
B→D	A→GH	AB→C	Both lossless and FD preserving
D→EF	H→IJ TD		

Transitive Dependency


Now,
R(ABCDEFGHIJ)
AGHIJ
BDEF
ABC
A→GH
H→IJ TD
B→D
D→EF TD
AB→C
AB→C
Now,
(HIJ)
(AGHI)
DEF
BD
H→IJ
A→GH
D→EF
B→D
∴ 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
ABCD
BCD
BC→D
ABC
AB→C
3NF and BCNF
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}
ABCD
AB
A→B
ACD
AC→D
C→D
CD
AC
24

* 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}

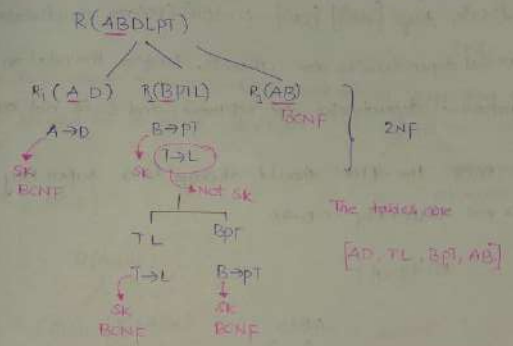
Candidate key = $(AB)^+$ = (ABDLPT) ⇒ $(AB)^+$ = (ABDPTL) //

CK = AB

Now, partial Dependencies = B→PT, A→D

B^+ = (B, P, T, L)

A^+ = (A, D)



59. BCNF Example 3

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

All the attributes are prime

⇒ 3NF

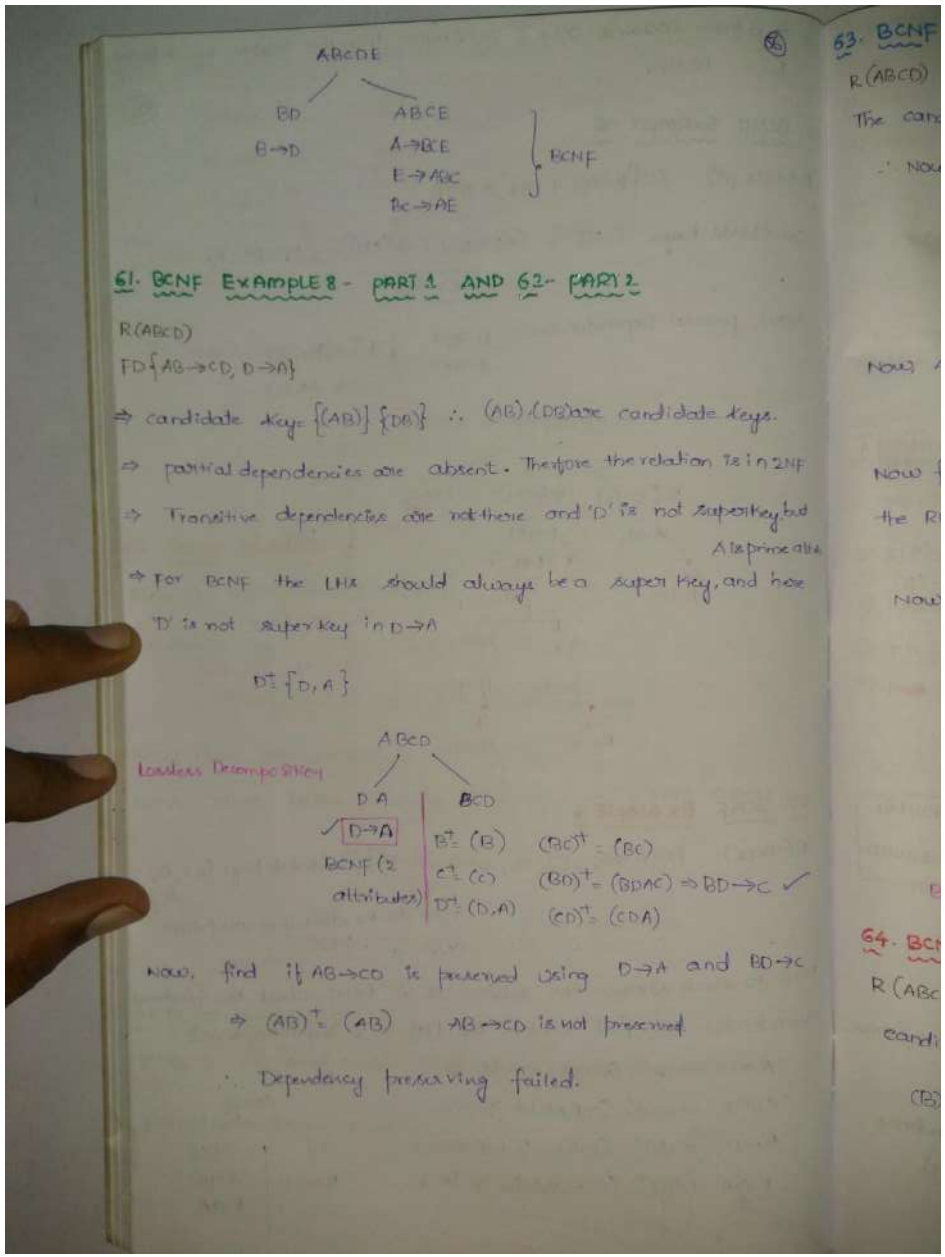
Now, To check whether the Relation is in BCNF check the functional dependencies and find whether the LHS is a super key or not

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, BC, D) = E is SK



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

R(ABCD)

FD{AB→CD, D→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 attribute

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

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

Lossless Decomposition

ABCD

DA	BCD
✓ D→A	$B^+ = (B)$
BCNF (2 attributes)	$C^+ = (C)$
	$D^+ = (D, A)$

$(BC)^+ = (BC)$

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

$(CD)^+ = (CDA)$

Now, find if AB→CD is preserved using D→A and BD→C

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

∴ Dependency preserving failed.

63. BCNF EXAMPLE 9

R(ABCD) {A→B, B→A, B→C, C→D}

The candidate key = ( )⁺ = (ABCD)

∴ Now, $A^+ = \{ABCD\}$ ✓
$B^+ = \{B,A,C,D\}$ ✓ } candidate keys
$C^+ = \{C,D\}$ $(CD)^+ = (CD)$ ×
$D^+ = \{D\}$

Now A, B are prime attributes ⇒ Now partial dependencies are not present (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→D ⇒ violating 3NF
Not SK, Not prime attribute

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

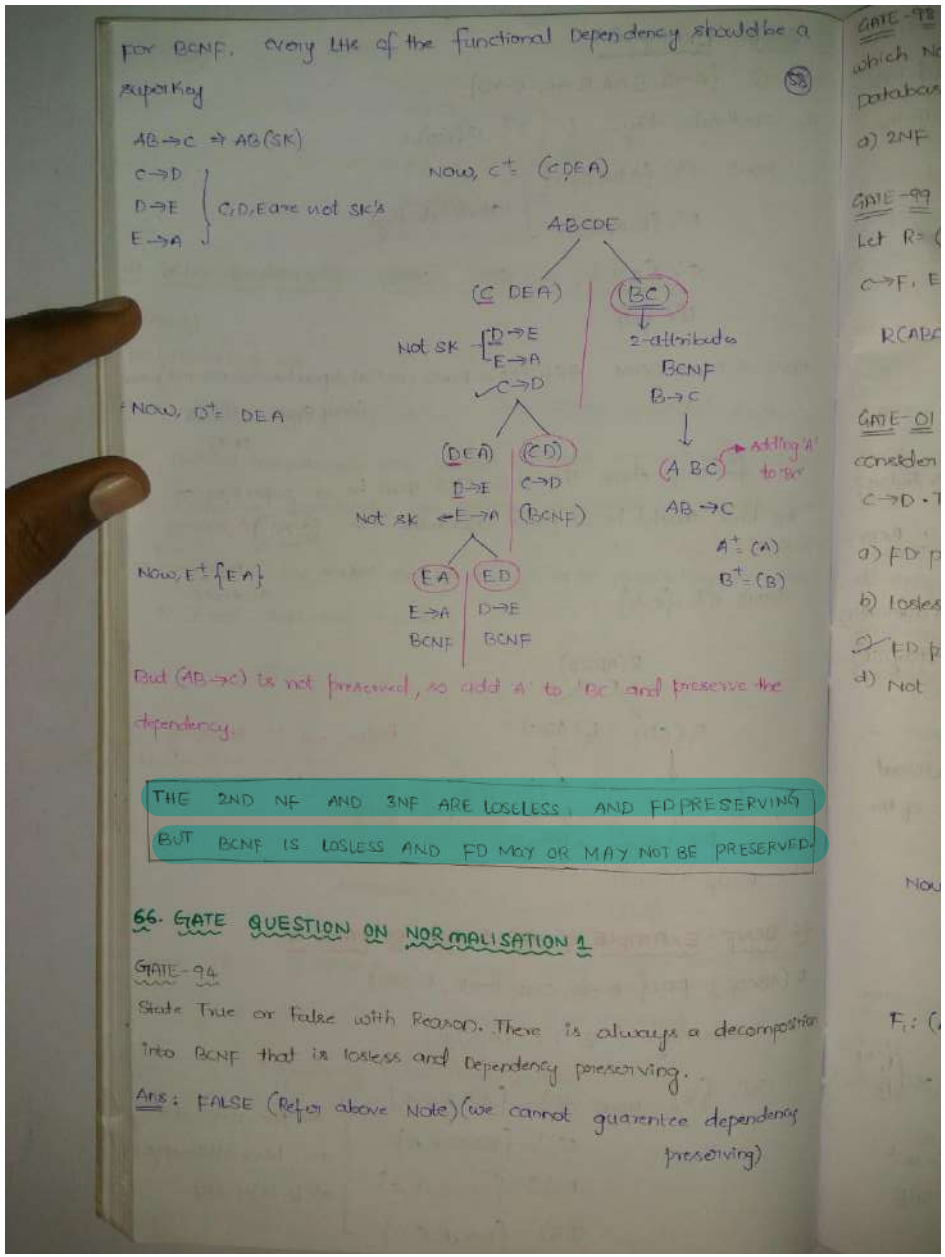
R(ABCD)
$R_1(CD)$ $R_2(ABC)$ lossless and FD preserving
C→D
3NF (attribute rule) and BCNF
A→B
B→A
B→C
3NF and BCNF

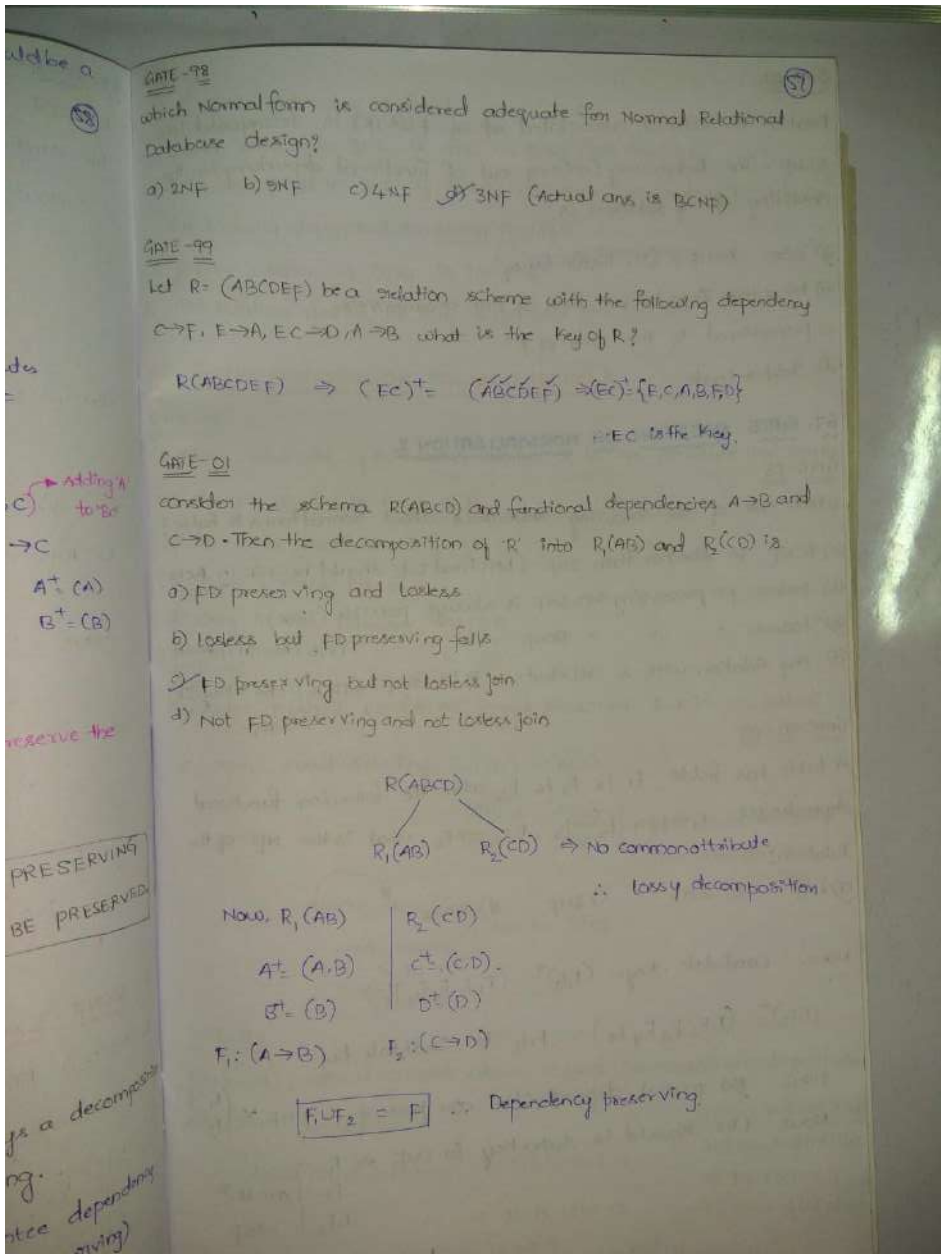
64. BCNF - EXAMPLE 10 PART 1 AND 65. PART-2

R(ABCDE) FD: {AB→C, C→D, D→E, E→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 superKey
AB→C ⇒ AB(SK)
C→D
D→E
E→A
C,D,E are not SK's
Now, C+ = (CDEA)
ABCDE
(C DEA)
(BC)
Not SK
D→E
E→A
C→D
2-attributes BCNF
B→C
Now, D+ = DEA
(DEA)
(CD)
(A BC)
Adding 'A' to 'BC'
D→E
C→D
Not SK
E→A
(BCNF)
AB→C
A+ = (A)
B+ = (B)
Now, E+ = {EA}
EA
ED
E→A
D→E
BCNF
BCNF
But (AB→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 guarantee dependency preserving)



GATE-98 (57)

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→F, E→A, EC→D, A→B. what is the key of R?

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

∴ EC is the key.

GATE-01

consider the schema R(ABCD) and functional dependencies A→B and C→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) ⇒ No common attribute

∴ lossy decomposition

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

$F_1 \cup F_2 = F$ ∴ Dependency preserving

GATE 02 (20)

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 GUARANTEE)

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

GATE-IT-05

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

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

Now, candidate Keys $(F_1F_2)^+ = (F_1 F_2 F_3 F_4 F_5)$

$(F_1F_2)^+ = (F_1F_2F_3F_4F_5)$ $\therefore F_1F_2$ is candidate Key.

Now, partial dependencies are present $\therefore$ 2NF ✗ $(F_1 \to F_3, F_2 \to F_4)$

Now, LHS should be super Key for 3NF $\Rightarrow$ F_1, F_2, F_1F_2 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

Nov

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?

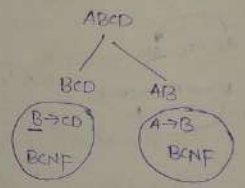
a) A→B, B→CD
b) A→B, B→C, C→D
c) AB→C, C→AD
d) A→BCD

Option A: A→B, B→CD R(ABCD)

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

Transitive depency = B→CD

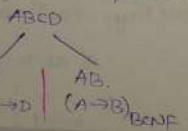
B$^+$ = (BCD)



Lossless FD preserved

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

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



Lossless and FD preserving

Option C: AB→
C→AD is
C$^+$ = {C, A, D

71. GATE 200

Relation 'R' is
is decompose
is in BCNF
To make quasi
should be

a) Dependency
b) Lossless join
c) BCNF defi
d) 3NF defini

72. GATE 04

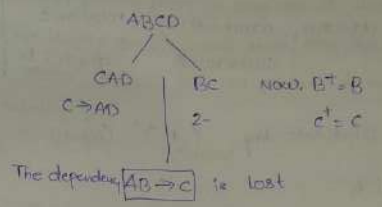
The Relation
has the FD

The highest n
a) 2NF

Option C: AB→C C→AD CK= (AB),(CB) are candidate keys

C→AD is the partial Dependency

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



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 decompositions?

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

b) Lossless join ⇒ Both have lossless but we cannot differentiate them (some times)

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, courseNO = B, grade = D

FD's: $AB \to D$, $CB \to D$, $A \to C$, $C \to A$ } R(ABCD)

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

$B^+ = B$

$(AB)^+ = (ABCD)$

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

$(DB)^+ = (DB)$

Now, partial dependencies are Absent $\therefore$ 2NF ✓

Now, 3NF check

AB = SK

CB = SK

$A \to C \Rightarrow$ LHS is not SK but RHS is prime attribute

$C \to A \Rightarrow$ LHS is not SK but RHS is prime attribute } 3NF ✓

Now, BCNF check

$\Rightarrow$ LHS should be Super Key

$\Rightarrow$ AB = SK

CB = SK

A → C

C → A

NOT SK } BCNF ✗

$\therefore$ Ans = 3NF

→D
→D
→C
→A
TRANSCED)

xutes are
A, B, C

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 ⇒ BCDEF
F → DE
CDE → F

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

⇒ 2NF ✓

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

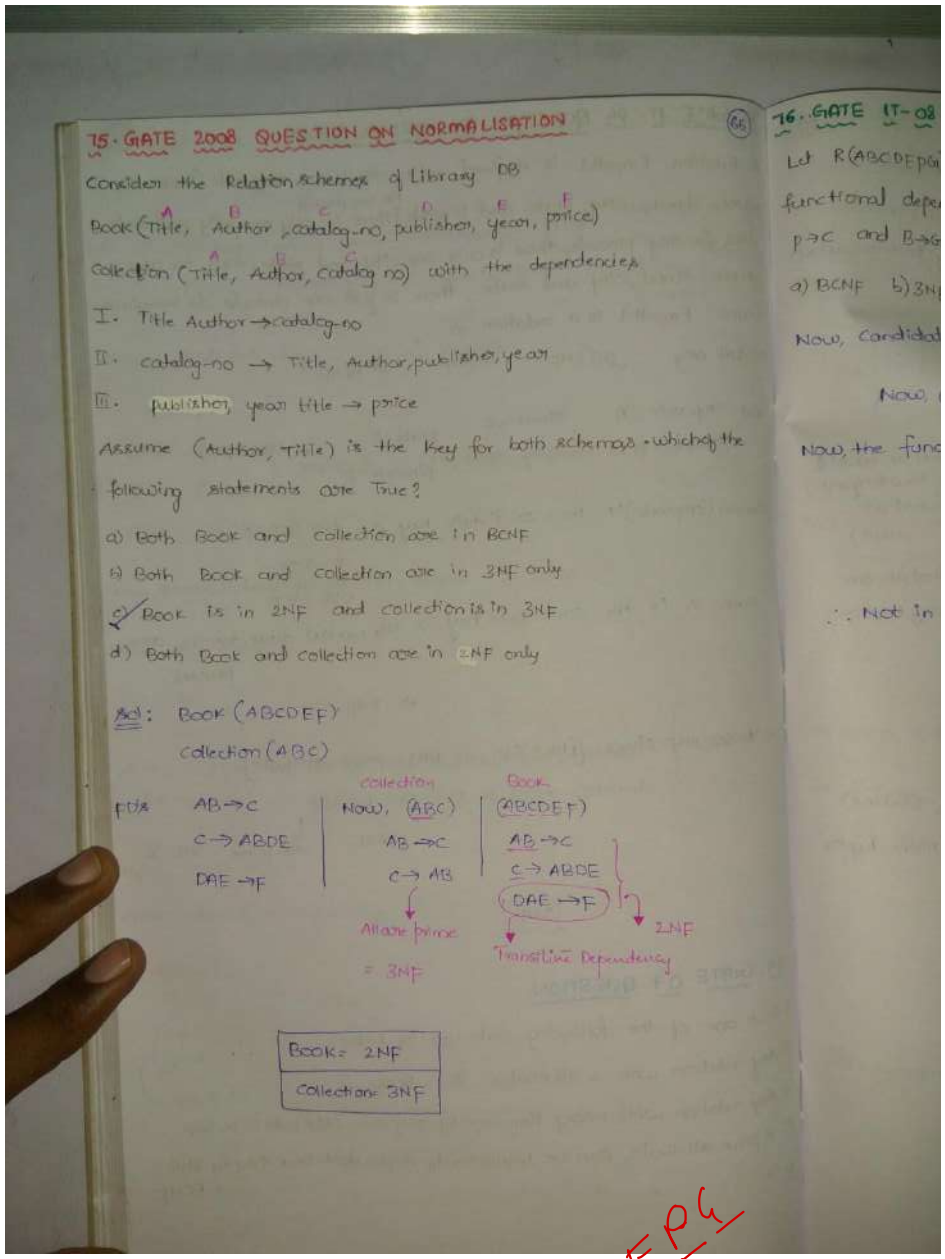
F → DE
Not SK, Not prime ⇒ 3NF ✗

∴ 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)

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

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 → C, C → ABDE, DAE → F

collection: Now, (ABC); AB → C, C → AB; All are prime = 3NF

Book: (ABCDEF); AB → C, C → ABDE, DAE → F; DAE → F: Transitive Dependency; 2NF

Book: 2NF
Collection: 3NF

76. GATE IT-08

Let R(ABCDEFG

functional depe

P → C and B →

a) BCNF b) 3N

Now, Candidat

Now,

Now, the func

∴ Not in

ABCDEFG

javatpoint.com

QUESTIONS TO IDENTIFY NORMAL FORM

← Prev Next →

To solve the question to identify normal form, we must understand its definitions of BCNF, 3 NF, and 2NF:

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key. i.e. there should not be a partial dependency from X → Y.

Definition of 3NF: First, it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e. Y is not a subset of X) then

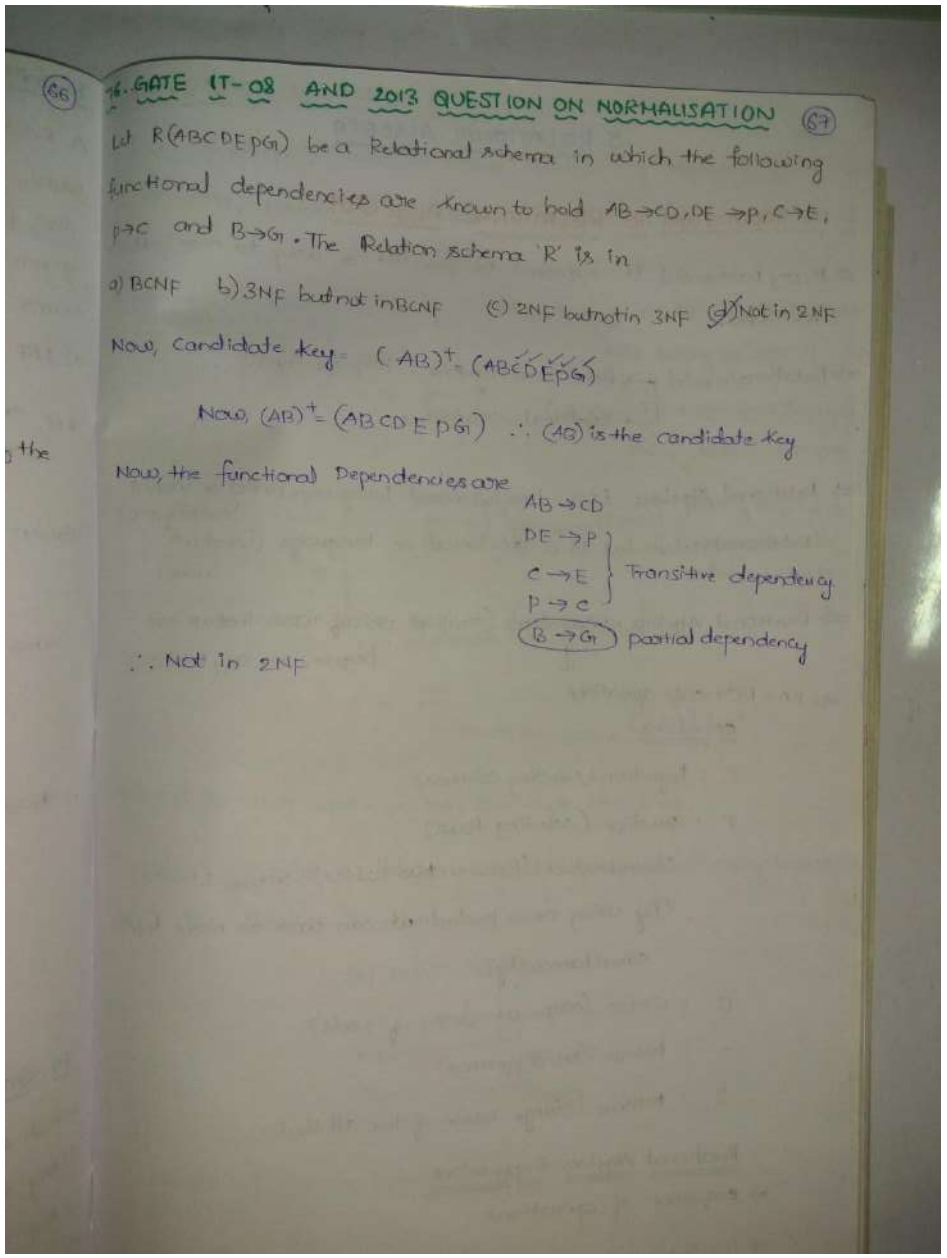
a. Either X is Super Key

b. Or Y is a prime attribute.

Definition of BCNF: First, it should be in 3NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then

a. X is Super Key

NOTE: If a table is in BCNF then it is in 3NF, 2NF, and 1NF, similarly if the table is in 3NF Then it is in 2NF and 1NF. Hence, we can say that if a table is in the higher normal form then by default it is in lower normal form.



Following are the types of Normalization:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. Fourth Normal Form
5. Fifth Normal Form
6. BCNF (Boyce – Codd Normal Form)
7. DKNF (Domain Key Normal Form)

1. First Normal Form (1NF)

- First Normal Form (1NF) is a simple form of Normalization.
- It simplifies each attribute in a relation.
- In 1NF, there should not be any repeating group of data.
- Each set of column must have a unique value.
- It contains atomic values because the table cannot hold multiple values.

Example: Employee Table

ECode	Employee_Name	Department_Name
1	ABC	Sales, Production
2	PQR	Human Resource
3	XYZ	Quality Assurance, Marketing

Employee Table using 1NF

ECode	Employee_Name	Department_Name
1	ABC	Sales
1	ABC	Production
2	PQR	Human Resource
3	XYZ	Quality Assurance
3	XYZ	Marketing

2. Second Normal Form (2NF)

- In 2NF, the table is required in 1NF.

An attribute which is not part of candidate key is known as non-prime attribute.

Example : Employee Table using 1NF

ECode	Employee_Name	Employee_Age
1	ABC	38
1	ABC	38
2	PQR	38
3	XYZ	40
3	XYZ	40

Candidate Key: ECode, Employee_Name
Non prime attribute: Employee_Age

- The above table is in 1NF. Each attribute has atomic values. However, it is not in 2NF because non prime attribute Employee_Age is dependent on ECode alone, which is a proper subset of candidate key. This violates the rule for 2NF as the rule says 'No non-prime attribute is dependent on the proper subset of any candidate key of the table'.

2NF (Second Normal Form) : Employee1 Table

ECode	Employee_Age
1	38
2	38
3	40

Employee2 Table

ECode	Employee_Name
1	ABC
1	ABC
2	PQR
3	XYZ

3. Third Normal Form (3NF)

- Third Normal Form (3NF) is used to minimize the transitive redundancy.
- In 3NF, the table is required in 2NF.
- While using the 2NF table, there should not be any transitive partial dependency.
- 3NF reduces the duplication of data and also achieves the data integrity.

Example : <Employee> Table

EId	Ename	DOB	City	State	Zip
001	ABC	10/05/1990	Pune	Maharashtra	411038
002	XYZ	11/05/1988	Mumbai	Maharashtra	400007

- In the above <Employee> table, EId is a primary key but City, State depends upon Zip code.
- The dependency between Zip and other fields is called Transitive Dependency.
- Therefore we apply 3NF. So, we need to move the city and state to the new <Employee_Table2> table, with Zip as a Primary key.

<Employee_Table1> Table

EId	Ename	DOB	Zip
001	ABC	10/05/1990	411038
002	XYZ	11/05/1988	400007

<Employee_Table2> Table

City	State	Zip
Pune	Maharashtra	411038
Mumbai	Maharashtra	400007

redundancy of data while inserting the new records.

- The City, State and Zip code will be stored in the separate table. And therefore the updation becomes more easier because of no data redundancy.

4. BCNF (Boyce – Code Normal Form)

- BCNF which stands for Boyce – Code Normal From is developed by Raymond F. Boyce and E. F. Codd in 1974.
- BCNF is a higher version of 3NF.
- It deals with the certain type of anomaly which is not handled by 3NF.
- A table complies with BCNF if it is in 3NF and any attribute is fully functionally dependent that is A → B. (Attribute 'A' is determinant).
- If every determinant is a candidate key, then it is said to be BCNF.
- Candidate key has the ability to become a primary key. It is a column in a table.

Example : <EmployeeMain> Table

Empid	Ename	DeptName	DepType
E001	ABC	Production	D001
E002	XYZ	Sales	D002

The functional dependencies are:
Empid → EmpName
DeptName → DeptType

Candidate Key:
Empid
DeptName

- The above table is not in BCNF as neither Empid nor DeptName alone are keys.
- We can break the table in three tables to make it comply with BCNF.

Empid	EmpName
E001	ABC
E002	XYZ

<Department> Table

DeptName	DeptType
Production	D001
Sales	D002

<Emp_Dept> Table

Empid	DeptName
E001	Production
E002	Sales

Now, the functional dependencies are:

Empid → EmpName

DeptName → DeptType

Candidate Key:

<Employee> Table : Empid

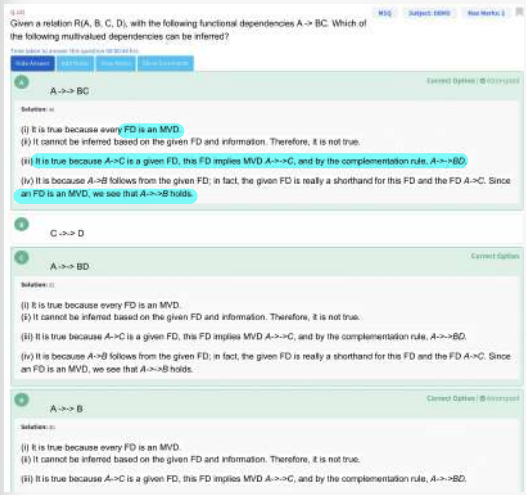
<Department> Table : DeptType

<Emp_Dept> Table : Empid, DeptType

- So, now both the functional dependencies left side part is a key, so it is in the BCNF.

5. Fourth Normal Form (4NF)

- Fourth Normal Form (4NF) does not have non-trivial multivalued dependencies other than a candidate key.
- 4NF builds on the first three normal forms (1NF, 2NF and 3NF) and the BCNF.
- It does not contain more than one multivalued dependency.
- This normal form is rarely used outside of academic circles.



Chapter 2: Functional Dependency & Normalization GATE Questions

QUESTION

1. Normalisation Identification and decomposition of Table
2. Identification of Lossless and Dependency preserving
3. Identification of candidate key
4. Normalisation from ER model
5. Minimal cover of Functional dependence
6. Numerical on number of super key

ANSWER

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 calculus

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

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

⇒ RA = RC + safe operations

Operations

π : Projection (selecting columns)

σ : Selection (selecting Rows)

× : Cross product (Between two tables) (combine 2 tables)
(By using cross product we can work on more tuples simultaneously)

∪ : Union (same as union of sets)

− : Minus (Set difference)

ρ : Rename (change name of the attributes)

Relational Algebra Expressions

⇒ Sequence of operations

⇒ ∩ : Intersection A∩B = A − (A − B)

⇒ ⋈ : Join (σ × ( ))

⇒ / : Division (π, ×, −)

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

SELECTION OPERATION (σ):

- This operation is used to choose a subset of Tuples (Rows).
- Selection / Horizontal partition

Emp

ENO	Ename	DNO	Salary

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

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

From this it is clear that selection operation is commutative

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

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

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

$\Rightarrow$ Select operation works on only one tuple at a time.

$\Rightarrow$ 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

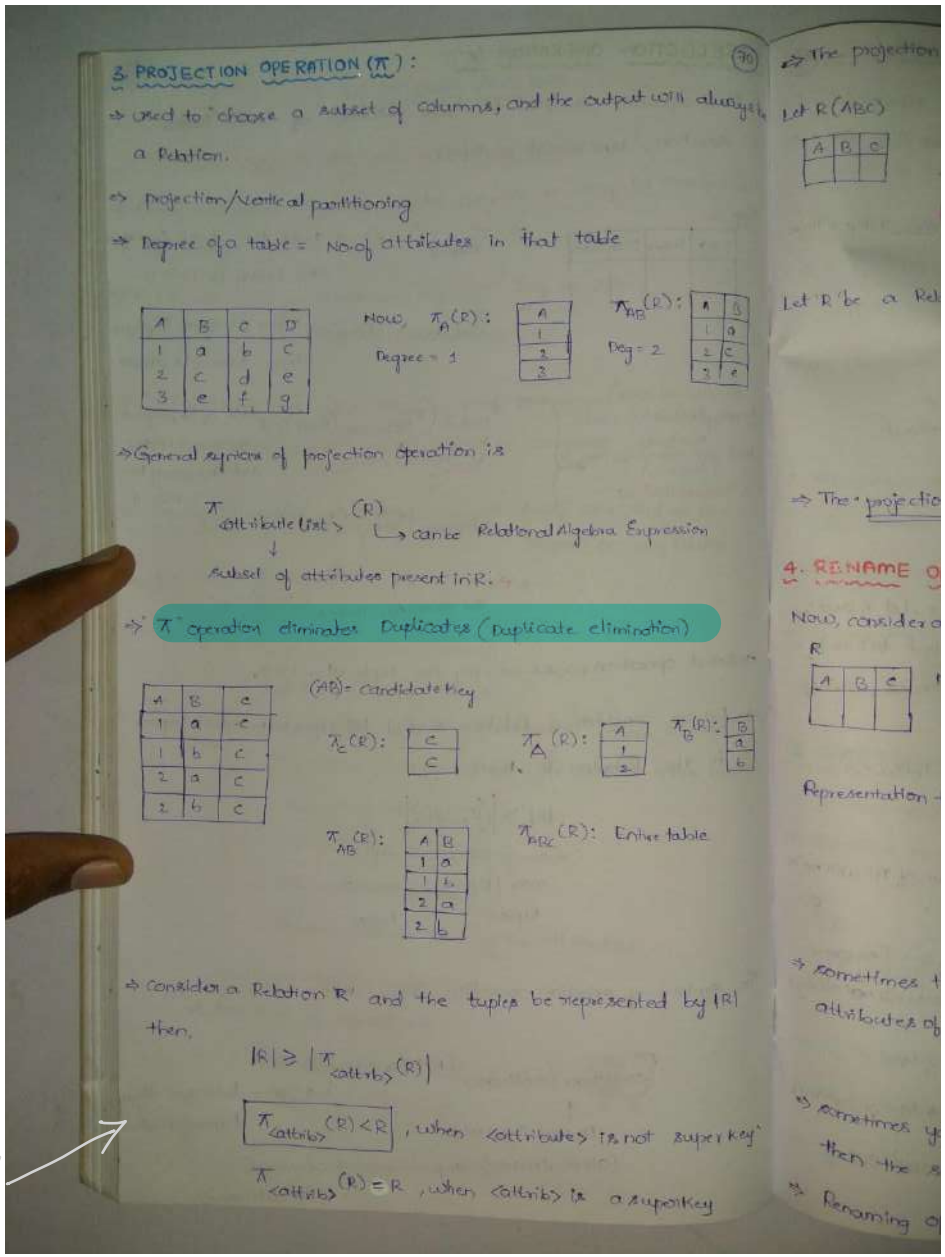
$\sigma_{\text{<selection condition>}}(\text{Relation name})$ → can be Relational Algebra Expression also)

↓

Boolean Expression

(attribute name) <Comparison 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

$\pi_{\text{<attribute list>}}(R)$ → 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

$\pi_A(R)$:

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,

$$|R| \geq |\pi_{\text{<attrib>}}(R)|$$

$\pi_{\text{<attrib>}}(R) < R$, when <attributes> is not superkey

$\pi_{\text{<attrib>}}(R) = R$, 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

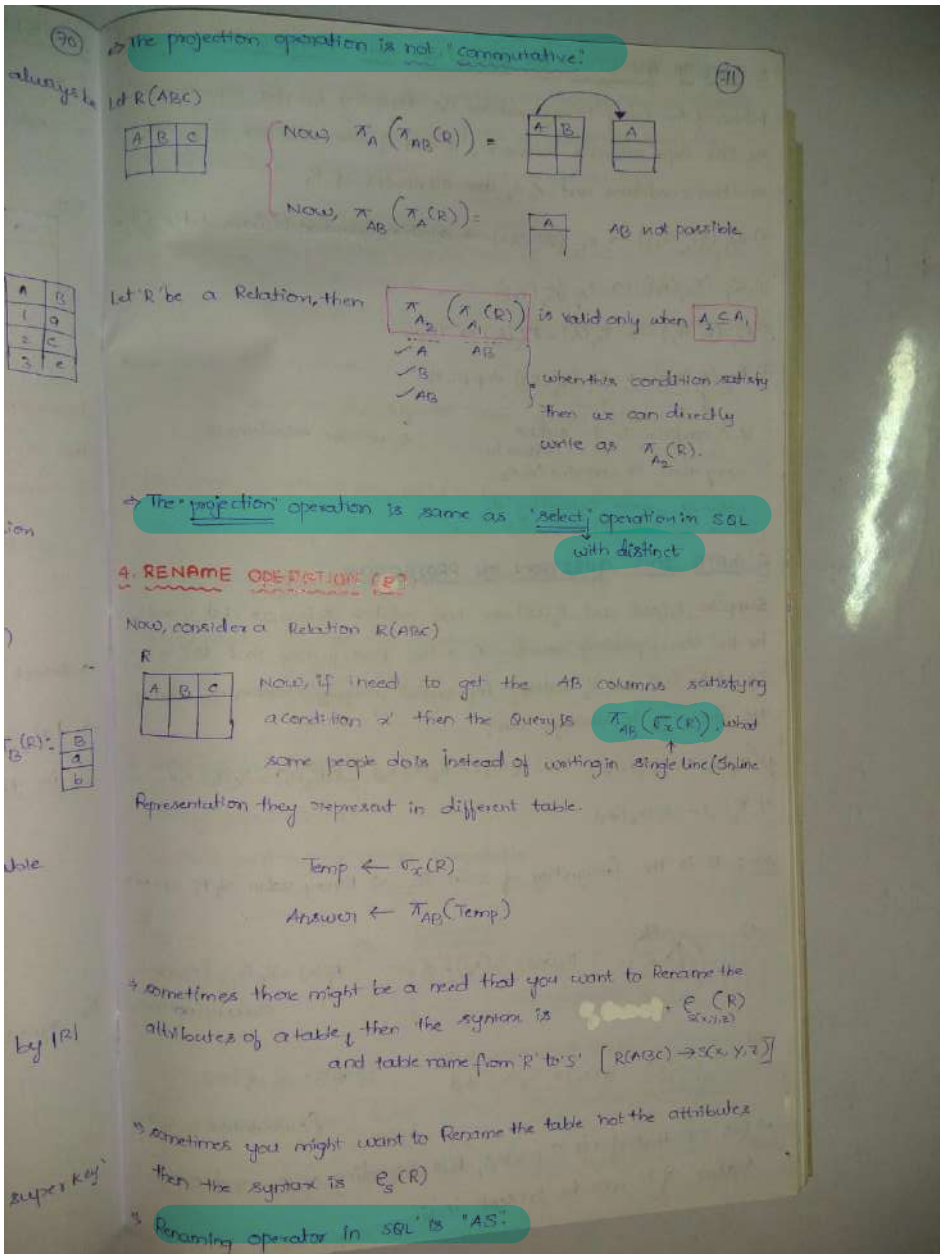
Representation -

- sometimes t attributes of

- sometimes y then the s

- Renaming o

always be



→ The projection operation is not "commutative".

Let R(ABC)

A	B	C

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

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

AB not possible

A	B
1	a
2	c
3	e

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

∴ A ✓, AB ✓, B ✓

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

$\pi_B(R)$:

B
a
b

→ 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 does instead of writing in single line (Online Representation) they represent in different table.

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

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

* sometimes there might be a need that you want to Rename the attributes of a table, then the syntax is $\rho_{S(X,Y,Z)}(R)$ and table name from 'R' to 'S' [R(ABC) → 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) $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \cup \sigma_{C_2}(R_2)$ ⇒ TRUE

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

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

6. GATE 2012 QUESTION ON PROJECTION

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 'C' in R_2. If data in R_1 and R_2 satisfy Referential integrity constraints, 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) $\pi_B(r_1) = \pi_C(r_2)$ [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"

$R(A_1, A_2, \ldots, A_n)$
$S(B_1, B_2, \ldots, B_n)$] we can perform R ∪ S iff R, S have

i) have same degree
2) corresponding elements domain must be same

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

↳ we use the 1st's set name of $R \cup S = R(A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_n)$

∴ The union operator is commutative - $R \cup S = S \cup R$ [No duplicates]

∴ The Intersection operator is commutative - $R \cap S = S \cap R$ [No duplicate values]

∴ The Difference operator is not commutative - $(R-S) \neq (S-R)$ [No duplicates]

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

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

8. CARTESIAN

(i) Binary operation

(ii) Relations need not be Union compatable

Emp

A	B	C

Dependent

D	E

⇒ Now Emp x dependent

A	B	C	D	E

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

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

Q.19 Consider two relations P and Q have x and y tuples respectively. Match the following expression with maximum and minimum number of tuples:

Expression	Tuples
A. $\sigma_A(P) \times Q$	1. max = x * y, min = 0
B. $\pi_A(P) - Q$	2. max = y * x, min = 0
C. $P \cup Q$	3. max = x, min = 0
D. $P \bowtie Q$	4. max = x + y, min = max (x, y)

	A	B	C	D
(a)	3	2	4	1
(b)	1	2	3	4
(c)	3	2	1	4
(d)	2	3	4	1

Employee

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

(m) rows = 3
(n) columns = 3

Dependent

D	E
1	a1
2	a2

(x) rows = 2
(y) 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*x) rows
(n+y) 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 two operations / Relations into a single longer tuple. Related tuples from

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

Now, some operation ar

Automatical

called Natura

performed on

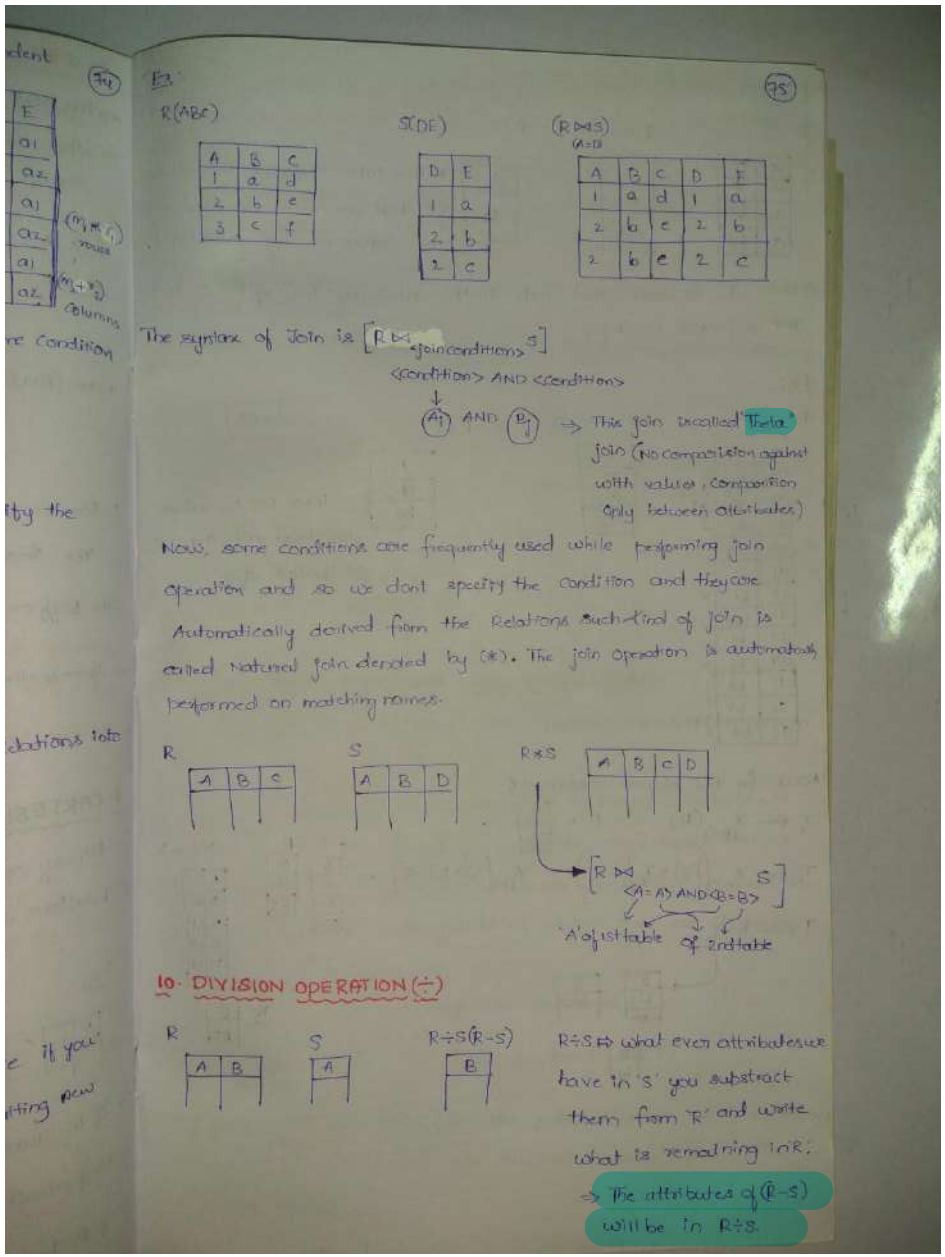
R

A	B	C

10. DIVISION

R

A	B



dent

74

E
a1
a2
a1
a2
a1
a2

(m * n) rows
(n+2) columns

re condition

ity the

dations into

e if you'

iting new

75

E3:

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 ⋈ <joinconditions> 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 ⋈ <A=A AND B=B> S]

'A' of 1st table, 'B' of 2nd table

10. DIVISION OPERATION (÷)

R

A	B

S

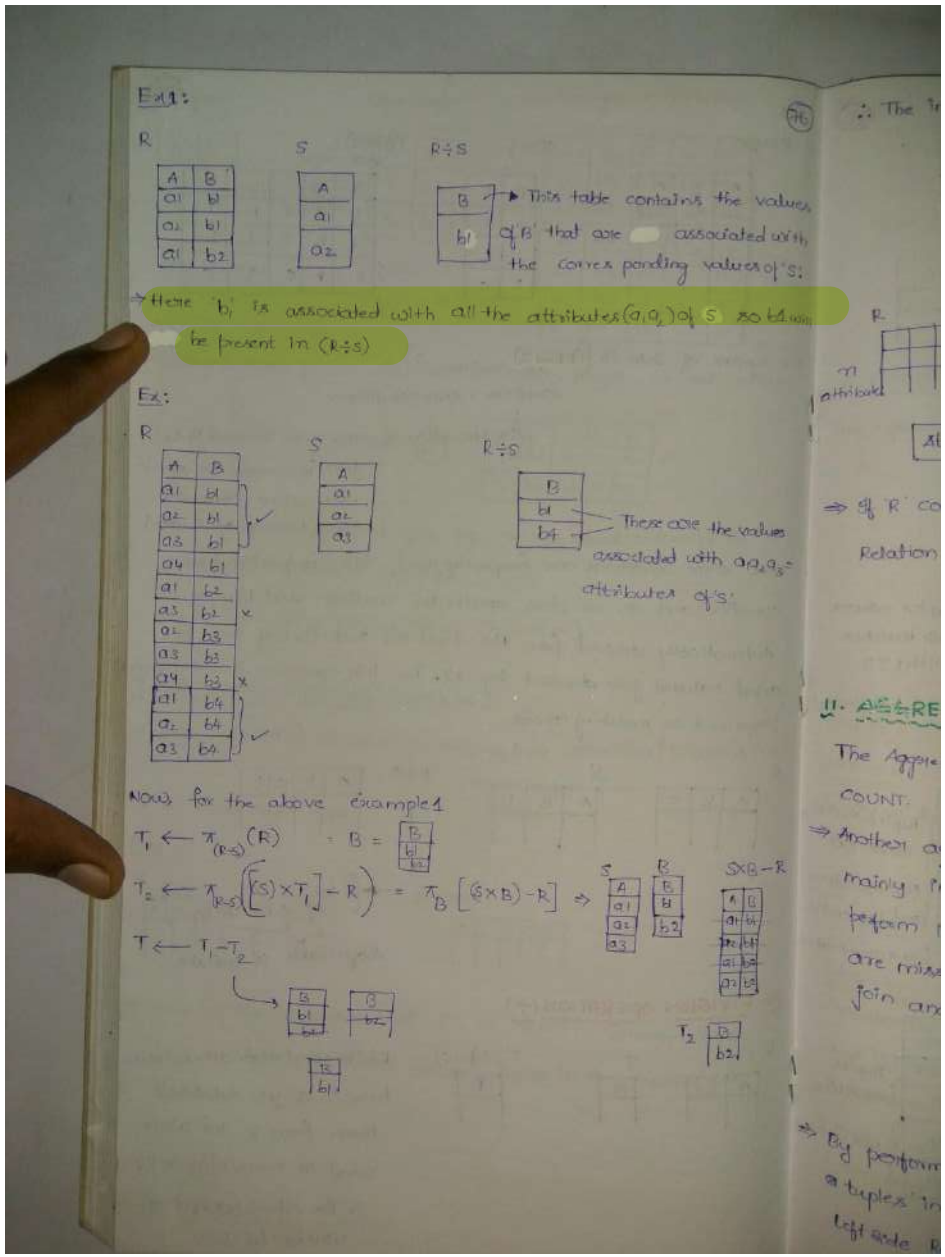
A

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.



Ex1:

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 (a1, a2) of S so that 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

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

S

A
a1
a2
a3

R

B
b1
b2

S×B−R

A	B
a1	b1
a2	b1
a1	b2
a2	b2

T_2

B
b2

$T \leftarrow T_1 - T_2$

B
b1

B
b1

ins the values
associated with
g values of 's':

of S so that will

are the values
d with a_1, a_2, a_3
of 's':

$S \times B - R$

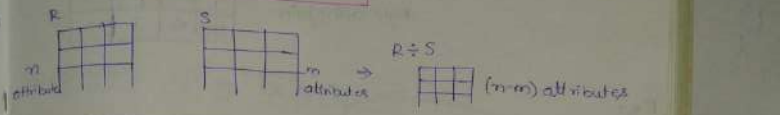
A	B
a_1	b_1
a_2	b_1
a_1	b_2
a_2	b_2

T_2

B
b_2

∴ The implementation of division using the basic operations is

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



Attributes in S ⊆ Attributes in R

⇒ If R contains 'x' attributes, S contains 'y' attributes then the Relation Z=(R÷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.

Q. 18 ? FAQ ▶ Solution Video Have any Doubt ?

Consider the following relational schema:
Student (Sid : integers, Sname : string, address : string)
Course (Cid : integer, Cname : string, branch: string)
Enroll (Sid : integer, Cid : integer, employee : integer)
Which of the following queries are equivalent to this query in English?
"Find the Sid of students who enrolled in some courses of 'AI' branch and some courses of 'ML' branch".

S_1 : Select Sid from course P, Enroll E where P.branch = "AI" and P.cid = E.cid and EXISTS (select Sid from course P_1, enroll E_1 where P_1.branch = "ML" and E_1.Sid = E.Sid and P_1.Cid = E_1.Cid);

S_2 : $\rho(T_1, \pi_{sid}(\pi_{cid}(\sigma_{branch = 'AI'}(course)) \bowtie Enroll))$
$\rho(T_2, \pi_{sid}(\pi_{cid}(\sigma_{branch = 'ML'}(course)) \bowtie Enroll))$
$T_1 \cap T_2$

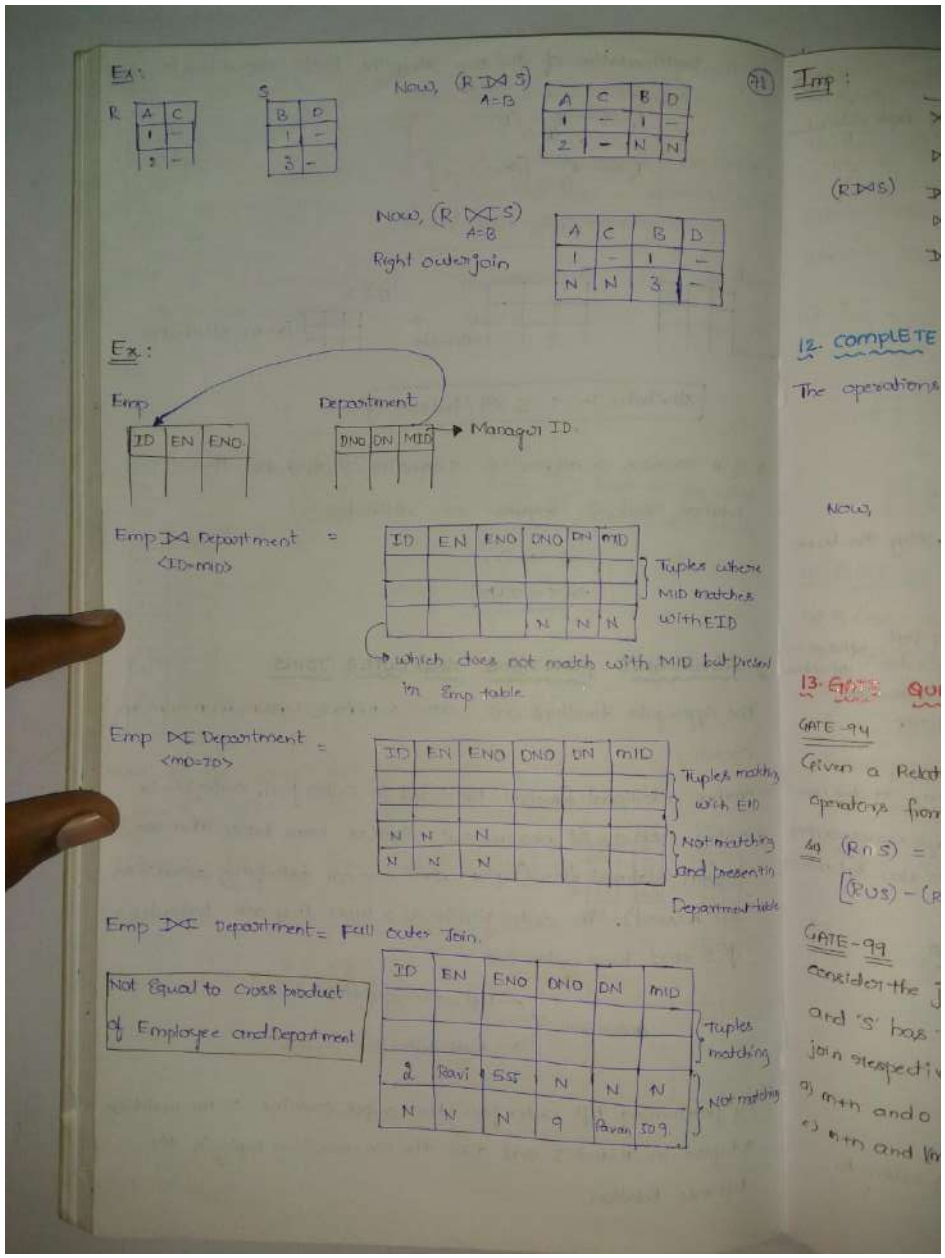
A Only S_1 — Your answer is IN-CORRECT

B Only S_2

C Both S_1 and S_2 — Correct Option

Solution :
(c)
Both S_1 and S_2 are equivalent and finds "The sid of students who enrolled in some courses of 'AI' and 'ML' branch".

D None of these



Imp :

	Min	Max
×	(m×n)	(m×n)
⋈	(0)	(m×n)
(R⋈S) ⟕	n	(m×n)
⟖	m	(m×n)
⟗	max(m,n)	(m×n)

R - m tuples
S - n tuples

12. COMPLETE SET OF RA OPERATIONS

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

$\{\bowtie, \cap, \div, ⟕, ⟖, ⟗\}$ ⇒ can be derived from complete set.

Now,

	Min	Max
∪	max(m,n)	(m+n)
∩	0	min(m,n)
−	0	m

R ⇒ m tuples
S ⇒ 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∩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

es where
matches
h EID
but present

tuples matching
with EID
ot matching
nd presence in
partment table

-tuples
matching
Not matching

GATE -99

The Relational Algebra expression equivalent 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.dno) → 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: $\sigma_{e.ename}(\sigma_{d.dno})$ 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 equivalent to which of the following Relational Algebra expressions

ple calculus

g the basic

This will get address

name>

N

get dept name

corresponding

an dept numbe

lent to

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 from = Cross product

where = Selection.

① $R_1 \times R_2 \times R_3$ From clause

⇓ σ = where

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

⇓ 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) $\Pi_D(r_1 \bowtie_{C=D} r_2)$ d) $\Pi_C(r_1 \bowtie_{C=D} r_2)$.

Ans

$R(ABC)$ — FK ⟹ $R_2(DE)$

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

Let 'r' be a relation instance with schema R(ABCD). We define $r_1 = \Pi_{ABC}(r)$ and $r_2 = \Pi_{A,D}(r)$. 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'

∴ $r \subset s$ or $s \supset r$

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 ⇒ $\Pi_A(\Pi_B(r))$ if $A \subset B$ then we write $\Pi_A(r)$

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

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

∴ Option a.

18. GATE 04 QUE

Consider the Relati
is shown unclearly
atleast one tag an
expression produ

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

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

Sol Let $A = \Pi$

$B = \Pi$

Name	Se

Now $A - B =$

⇒

. we define
tes natural
s lossy, which

which means
join them
as tuples) that

AND JOIN
expression
attributes
based on the

$\pi_{A_2}(\sigma_{F_1 \vee F_2}(r))$

ade it

then we
$\pi_A(r)$

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).

$$\pi_{name}\left(\sigma_{Sex=female}(Student)\right) - \pi_{name}\left(Student \bowtie_{Sex=F \wedge x=male \wedge m \geq m} \rho_{(x,m)}(Student)\right)$$

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)

$B = \pi_{name}\left(Student \bowtie_{Sex=F \wedge x=male \wedge m \geq m} \rho_{(x,m)}(Student)\right)$

Name	Sex	Marks

$\bowtie$

m	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 (StudId, CourseId) gives which student has enrolled for what course(s). Assume that every course is taken by one male and atleast one female student. what does the following Algebraic Expression Represent.

$$\Pi_{courseId}\left(\left(\Pi_{studId}\left(\sigma_{Sex='female'}(StudInfo)\right) \times \Pi_{courseId}(enroll)\right) - enroll\right)$$

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	Armaan	F
507	Bhavana	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, $A \times B =$

StdId	CourseId
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 17

Consider the fo

b- schema =

a- schema =

d- schema =

Let Branch, acc

schema. Assum

bigger than th

Consider the

which one of t

above Query?

a) $\Pi_{cname}(\sigma_{bu}$

b) $\Pi_{cname}(\sigma_{b-u}$

c) $\Pi_{cname}((\sigma_{b}$

d) $\Pi_{cname}(\sigma_{b-}$

⇒ If you perfor

it wont be

no of Tuples

⇒ So first appl

product fo

· 1st selec

and sele

Now, Cor

Now, c

$\Pi_{cname}(\sigma_{bu}$

20. GATE 07 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. $\Pi_{c\text{-name}}(\sigma_{b\text{-city}=\text{"Agra"} \wedge \text{bal}<0}(\text{branch} \bowtie (\text{account} \bowtie \text{depositor}))$

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

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

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

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

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

⇒ 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
- and select the tuples with bal<0 from a-schema.
- Now cross product the account with depositor
- Now, cross product the Result with b-schema

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

= 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?

P, Q is the key not P, Q alone

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

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

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	R_1	R_2	R_3
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 $\bowtie$ S = R ⋈ S

P
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.

22. GATE 20

The following

The relation "R

is the max

a) 100 b) 20

Now, $B \to A$

contains distin

R

A	B
1	
2	
3	
⋮	
200	

R

A	B
1	
2	
3	
⋮	
200	

23. GATE 12

consider the

A

ID	Name
12	Arun
15	Shreya
99	Rohit

How many tup

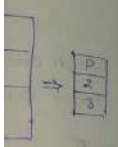
expression co

as that of A.

a) 7 b) 4

for both,

I, III, and IV



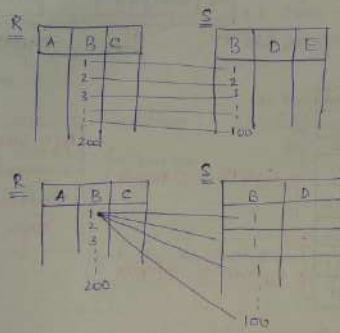
22. GATE 2010 QUESTION ON NATURAL JOIN ON PRIMARY KEY

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 ⋈ 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)



∴ The max no. of tuples in R ⋈ 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 ∪ B =

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

⋈ C

C

ID	phone	Area
10	2200	02
99	2100	01

⇒ (A ∪ B) ⋈ 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 ∨ 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

$A_d > 40$ = 2 tuples

= 7 tuples

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

The nested loop
we are going t

r(R)
↓
20 tuples
↓
2 Blocks.
↓
In this case the
No. of Block acces
are

1 + 10 × 10 - for
1 + 10 × 10 - for
202

∴ Put

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.

25. GATE 201

in the

Q. 26

Ask any Doubt ?

Consider the relation R has 1000 tuples and relation S has 5000 tuples. The block size is 200 tuples/ block. These 2 relations have to be joined as per a specified join condition that needs to be evaluated for every pair of records from these 2 relations. The memory buffer space available can hold exactly one block of records for R and one block of records of S simultaneously at any point of time. No index is available on either relation. The optimal number of block transfer required in case of Nested loop join is _____

25005 — Correct Option

Solution :
25025
Number of tuples in R (T_1) = 1000
Number of tuples in S (T_2) = 5000
Block size = 200 tuples/block
Number of block in R (B_1) = $\left\lceil \frac{1000}{200} \right\rceil = 5$
Number of block in S (B_2) = $\left\lceil \frac{5000}{200} \right\rceil = 25$
Number of block transfer in nested join = 1000 × 25 + 5 = 25005

QUESTION ANALYTICS

Level : Difficult — Accuracy — Topper's Time

Correct Marks : 1.0 — Average Time — Your Time

Negative Marks : 0

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. post

Refer 547 page 3.25(a) in made easy book

IN

nested loop

block size,

ults. Assuming

sk blocks

n 0.5

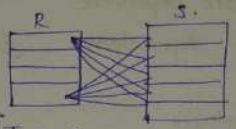
an 0.5.

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 access are

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

202

1 + 2*10 — for 1st blk { Take every row of R and combine with 'S' }
1 + 2*10 — for 2nd blk
:
10 times → for 10th blk
1 + 2*10

210 Block Access

∴ put the smaller size table in outer loop.

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

- 1 → 1st block
- 10*10 → Accessing all the blocks in 'S' = 10 blocks and size of each block = 10 tuples = 10 * 10 = 100 tuples.

15. 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 ⋈ (dependent))

$\langle empId = eId) \wedge (empAge \leq depAge)$

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 ⋈ (dependent)) ⇒ It returns the empids of employees whose Age is less than or equal to their dependents

$\langle empId = eId) \wedge (empAge <= depAge)$

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

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

∴ Option D.

26. INTRODUCTION TO RELATIONAL CALCULUS

On the Relational model two formal languages are proposed they are

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

A. languag
express

TRC

27. TRC

The most

{t/c

Tuple variable

Ex:-

student

FN

{t.FN/stu

The gener

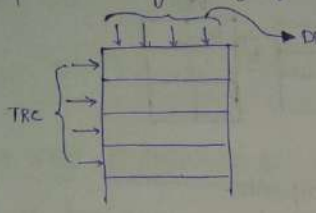
{t_i . A_j

28. FREE

General
in the
(1) R(t)

Relational calculus is weaker than relational algebra and it is based on predicate calculus of formal logic

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

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

Tuple variable — Condition

Ex:-

Student

FN	LN	Marks

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

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

Tuple variable to range on table — Name of the table that the tuple variable should Range

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

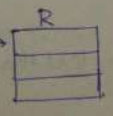
↳ Gives FN of students whoose marks > 50.

The general form is

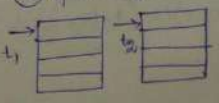
$$\{t_1.A_j, t_2.A_k, \ldots t_n.A_m / cond(t_1 \ldots t_n, t_{n+1} \ldots t_{n+m})\}$$

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> $t.A \ \theta \ \langle constant \rangle$ ($t.marks > 50$)

③ $t_1.A < t_2.B$ 

Q. 21

Solution Video | Have any Doubt ?

Consider a database that has a relation schema:
student (rollNo, name, degree, year, sex, deptNo, advisor)
course (courseId, cname, credits, deptNo)
enrollment (rollNo, courseId, sem, year, grade)
Assume that every student is enrolled in at least one course, then which of the following query
"Retrieve the names of students who have scored 'A' in all subjects they have enrolled"?

A. $\{s.name \mid student(s) \wedge (\forall e)\ [(enrollment(e) \vee e.rollNo = s.rollNo) \rightarrow e.grade = 'A']\}$

B. $\{s.name \mid student(s) \wedge (\forall e)\ [(enrollment(e) \wedge e.rollNo = s.rollNo) \rightarrow e.grade = 'A']\}$ — Correct Option

Solution :
(b)
In option (b)
When Student e with all A grade, for enrollment tuples not having her roll number, LHS is false. For enrollment tuples having his/her roll number, LHS is true, RHS also true so the implication is true for all e tuples.
When Student e with some non-S grades, for enrollment tuples not having her roll number, LHS is false.
For enrollment tuples having his/her roll number, LHS is true, but RHS is false for at least one tuple.

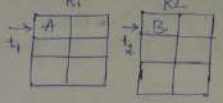
C. $\{s.name \mid student(s) \wedge (\exists e)\ [(enrollment(e) \wedge e.rollNo = s.rollNo) \rightarrow e.grade = 'A']\}$ — Your answer is IN-CORRECT

D. $\{s.name \mid student(s) \wedge (\exists e)\ [(enrollment(e) \wedge e.rollNo = s.rollNo) \rightarrow e.grade = 'A']\}$

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 $\{F_1, F_2\}$ = Atomic Expressions

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

we have 2 types of variables they are:

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

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

=> Free variables appear only on the leftside of the bar in TRC Representation

& 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 doesn't contains the tuples 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.C < 10

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

29. TRC EXA...

List the nam...

"Research" dep...

Employee (Fna...

Department (Dn...

Dept_location...

Project (Pna...

Works_on (E...

Dependent (...

Sol

Emp...

t →

=> Now, after ...

variables f...

should be ...

should be ...

variable p...

=> The cross p...

TRC

{t. Fname, t...

Now, $\sim \exists t (t.C < 10)$ ⇒ There doesn't exist atleast one value of t.C such that t.C < 10 ∴ TRUE

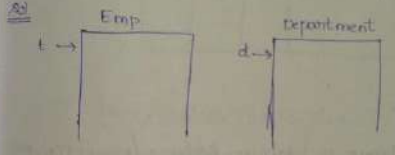
Now, $\forall t\ (t.C \geq 10)$ = TRUE (All values are greater than 10).

29. TRC EXAMPLE 1

List the names and Addresses 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 having 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'.

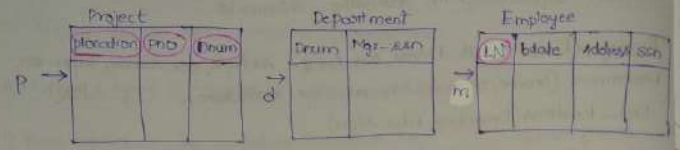
⇒ The cross product in Relational Algebra is equal to ∃ in the TRC

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

30. TRC EXAMPLE 2

For every project located in "stafford" list the project numbers, 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

$$\{p.pnum, p.dnum, m.Lname, m.bdate, m.Address \mid PROJECT(p)\ AND$$

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

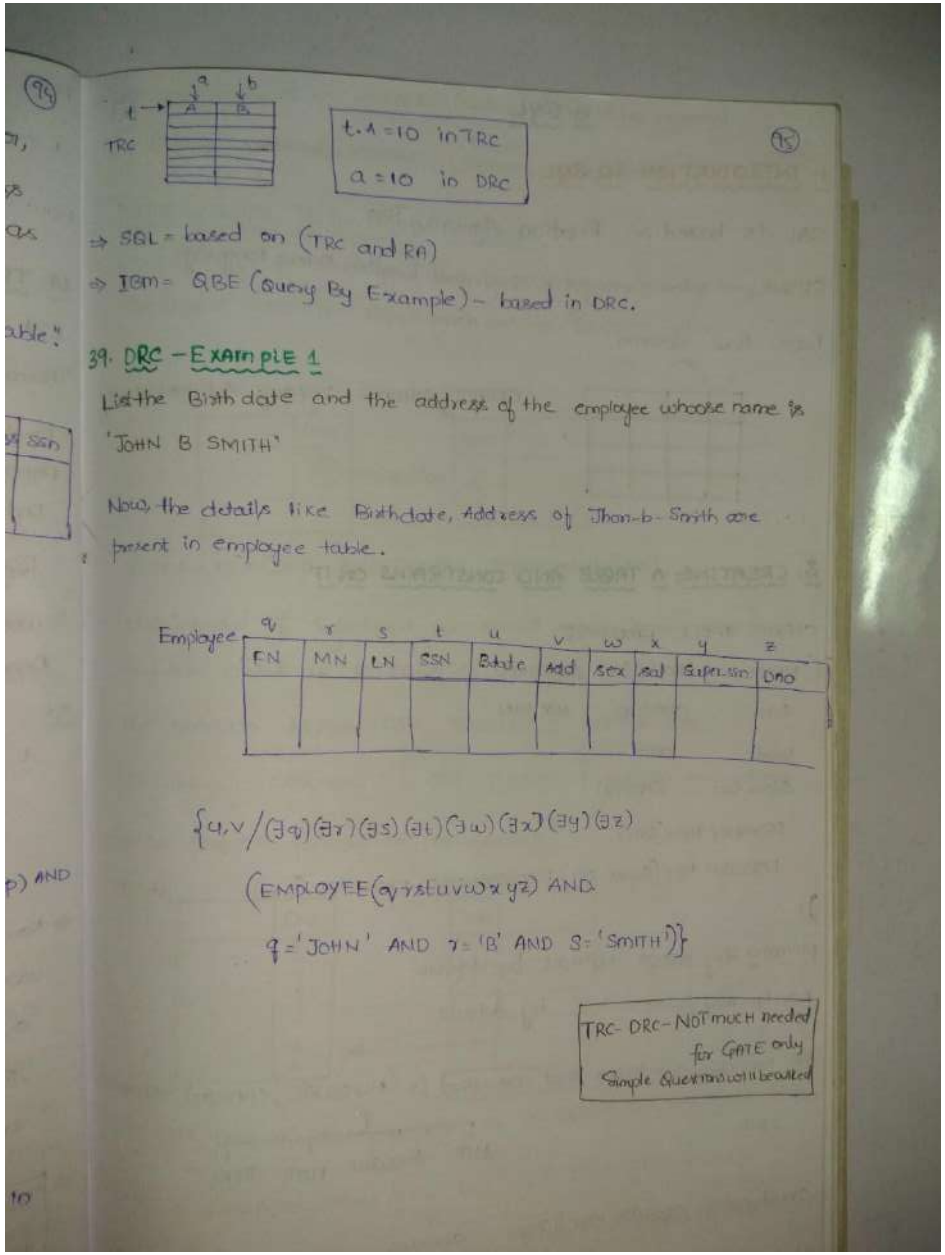
$$((\exists d)(DEPARTMENT(d)\ AND\ p.Dnum$$

$$= d.Dnum\ AND\ d.mgrssn = m.ssn)\}$$

38. DOMAIN RELATIONAL CALCULUS INTRODUCTION

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

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



TRC

A	B

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

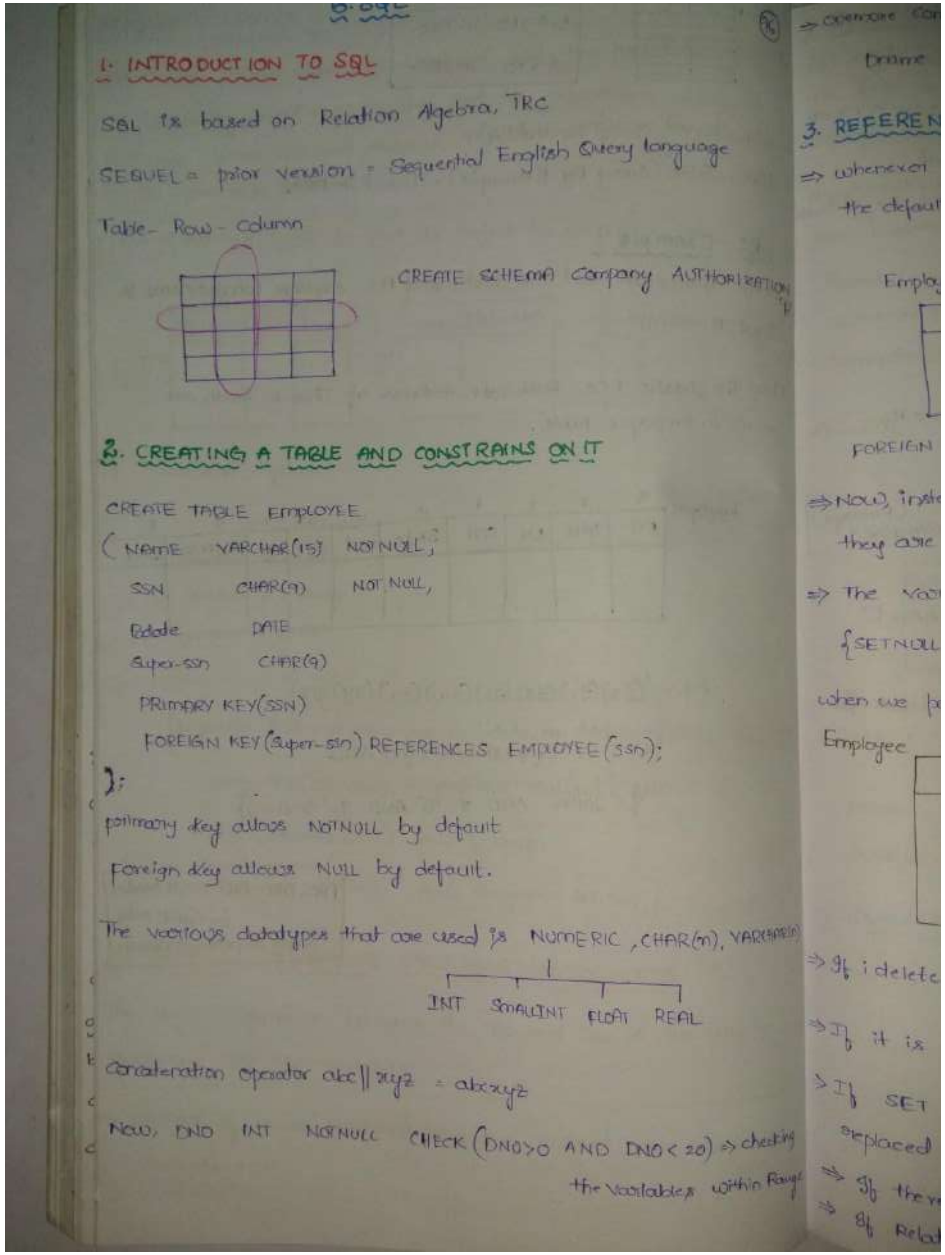
q	r	s	t	u	v	w	x	y	z
FN	MN	LN	SSN	Bdate	Add	Sex	Sal	Super_ssn	DNO

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

(EMPLOYEE(qrstuvwxyz) AND

q = 'JOHN' AND r = 'B' AND s = 'SMITH')}

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



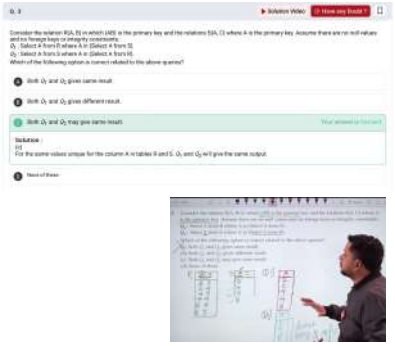
Predicate evaluation order

If you are familiar with the logical query processing order, then you may expect that a query is executed in the following order:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT

The sequence above outlines the logical order for executing query. Logically the **FROM** clause is processed first defining the source data set, next the **WHERE** predicates are applied, followed by **GROUP BY**, and so on.

However, physically, the query is processed differently and the query optimizer is free to move expressions in the query plan in order to produce the most cost efficient plan for retrieving the data. This leads to a common misunderstanding that a filter in the **WHERE** clause is applied before the next phases are processed. In fact, a predicate can be applied much later in the physical execution plan. Also, there is no left to right order for execution of predicates. For example, if you have a **WHERE** clause containing "**WHERE x=1 AND y=2**", there is no guarantee that "**x=1**" will be evaluated f can be executed in any order.



write it's

see in result

the attributes

= 10.

(Employee)
<ssn=10>

select:

work for

5. ALIASING is recommended

for each employee, retrieve the employee's first and lastname and the first and lastname of his immediate supervisor.

Employee (Fname, Minit, Lname, SSN, Bdate, Address, sex, salary, super-ssn, 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.ESSN = S.SSN ;
```

6. DUPLICATE TUPLES AND SET OPERATIONS

⇒ we can use the keyword DISTINCT to eliminate the duplicates.

```
SELECT  DISTINCT  FName
FROM    Employee
WHERE   DNO = '4' ;
```

⇒ All the fnames of employee who work for dept no '4' (No duplications)

Q.100 MSQ Subject: DBMS Mark: 2

Consider a relation R has the schema R(a, b). Which of the following SQL statement is not legal?

Time taken to answer this question 00:00:36 hrs

Hide Answer | Add Notes | View Notes | Show Community

A. Your answer is Wrong

Select a, null from R where ISNULL(b);

B. Correct Option | Attempted

Select a, b from R where b like '%NULL%'

Solution:

(i) The query: Select a, null from R where ISNULL(b); is valid as well as legal. It will return 2 columns in the resultant set (a, null) and the tuples in the resultant set will be the ones which have attribute b is null.

(ii) The query: Select a, b from R where b like '%NULL' is valid but it is not legal as it has a missing semicolon at the end. Thus the given query will throw an error.

The query is valid as it does not involve nulls at all. It is a condition involving a pattern that happens to have the characters NULL as part of it. This pattern will not even match a NULL value.

(iii) The query: Select a, b from R where b = null; is not legal as we cannot compare or check for the null values using the equating operator.

Hence, the correct option is (ii), and (iii).

C. Correct Option | Attempted

Select a, b from R where b = null;

Solution:

(i) The query: Select a, null from R where ISNULL(b); is valid as well as legal. It will return 2 columns in the resultant set (a, null) and the tuples in the resultant set will be the ones which have attribute b is null.

(ii) The query: Select a, b from R where b like '%NULL' is valid but it is not legal as it has a missing semicolon at the end. Thus the given query will throw an error.

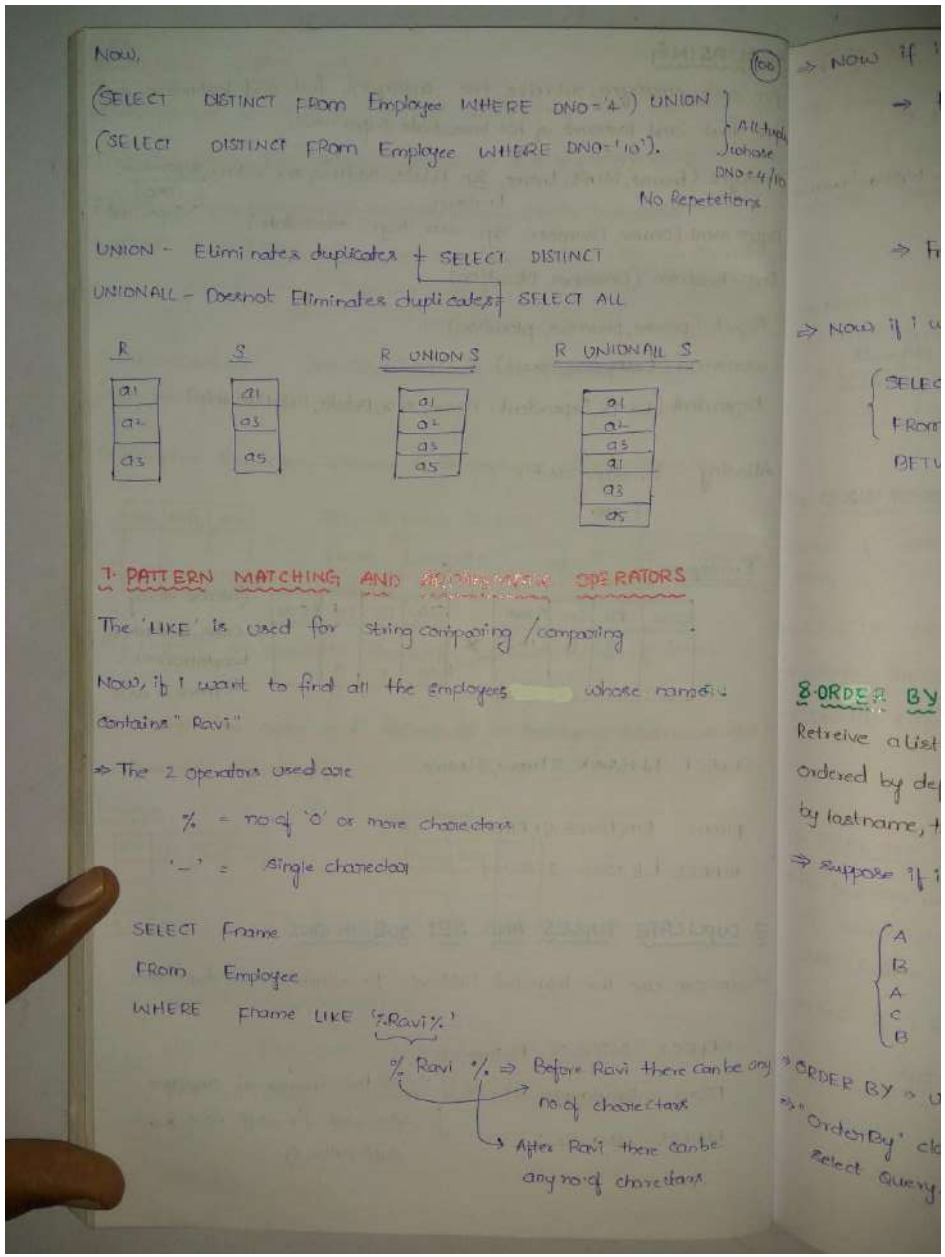
The query is valid as it does not involve nulls at all. It is a condition involving a pattern that happens to have the characters NULL as part of it. This pattern will not even match a NULL value.

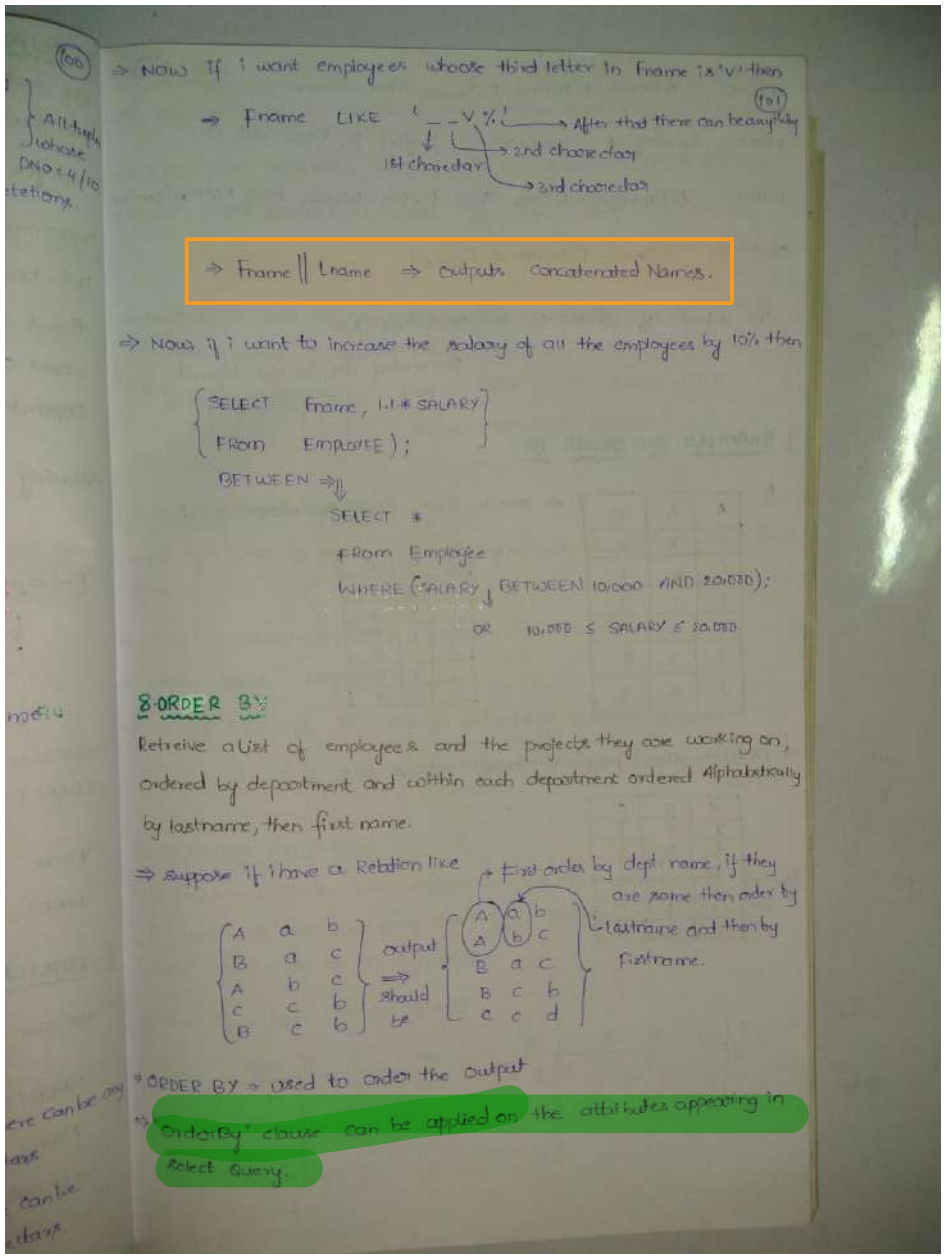
(iii) The query: Select a, b from R where b = null; is not legal as we cannot compare or check for the null values using the equating operator.

Hence, the correct option is (ii), and (iii).

D. Your answer is Wrong

All the above queries have errors.




⇒ Now if I want employees whose third letter in Fname is 'v' then
⇒ Fname LIKE '__v%'
After that there can be anything
1st character
2nd character
3rd character
⇒ 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 ≤ SALARY ≤ 20,000
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
output ⇒ should be
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.Projectname
FROM     EmployeeE, WorksonW, ProjectP, DepartmentD
WHERE    D.Dnumber= E.Dno AND E.Ssn= W.Essn AND W.Pno=P.Pnum
ORDERBY  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 ORDERBY 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. INSER

The comm

→INSERT

→INSERT

⇒ Now, if

values

R

11. DELET

Now, if

then,

⇒ DELETE

⇒ DROP

⇒ UPDATE

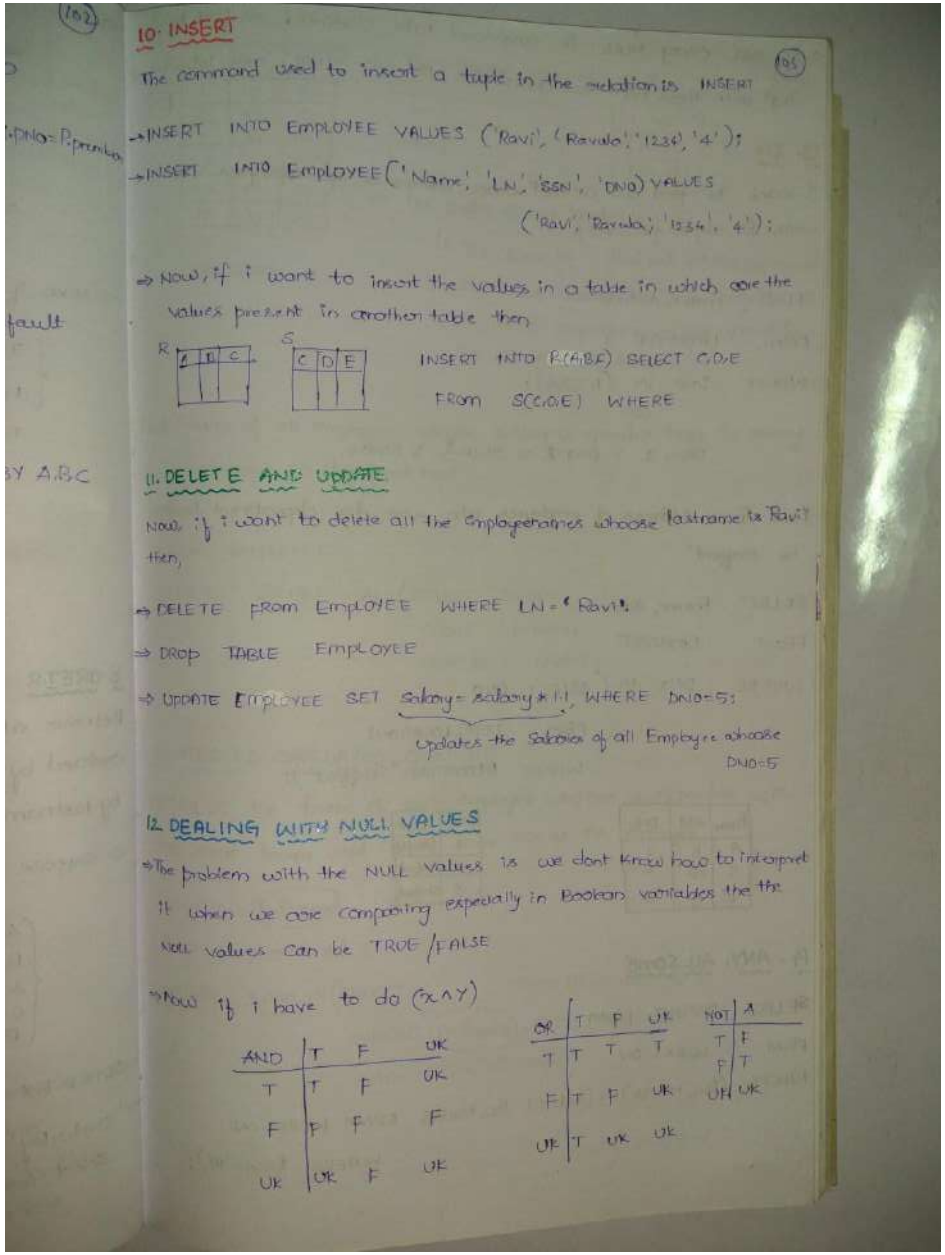
12. DEAL

⇒ The probl

If wher

NULL va

⇒ Now



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

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

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 Salary = salary * 1.1 WHERE DNO=5;

Updates the Salaries of all Employee whoose 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 ∧ 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

Q. 17

Solution Video | Have any Doubt ?

Which of the following is/are integrity constraint?

A. Not null — Your option is Correct

B. Unique — Your option is Correct

C. Identical — Your answer is IN-CORRECT

D. Check — Your option is Correct

YOUR ANSWER - a,b,c,d | CORRECT ANSWER - a,b,d | STATUS -

Solution :

(a, b, d)
Identical is not an allowed integrity constraint in SQL. Not null prevents null values and unique only allows unique values to be entered. Check checks for a given condition.

Q. 4

Solution Video | Have any Doubt ?

Consider the given boolean expression:
Exp. : a ≥ 98 AND NOT (b ≤ 30)
For what values of a and b including NULL the Exp have truth value UNKNOWN in SQL. Consider these statements regarding above conditions:
I. b is NULL and a ≥ 98
II. a is NULL and b > 30
Which of the above statements is/are correct?

Comparison with NULL is Unknown.

Unknown && T = Unknown

Unknown && T = Unknown

A. Only I — Your answer is IN-CORRECT

B. Only II

C. Both I and II — Correct Option

Solution :
(c)
- The AND evaluates to UNKNOWN (x_1 AND x_2) when atleast one of the x_1 or x_2 is NULL and neither x_1 nor x_2 is FALSE.
- So, both the statement results in UNKNOWN.

D. None of these

Q.11)

MCQ | Subject: DBMS | Max Marks: 1

Given a database with multiple tables, which of the following constraints can be used in a way to ensure, or will by definition not allow NULL values to be inserted ?

I. UNIQUE
II. NOT NULL
III. FOREIGN KEY
IV. PRIMARY KEY
V. CHECK

Time taken to answer this question 00:01:41 hrs

Hide Answer | Add Notes | View Notes | Show Comments

A. I , II, and IV

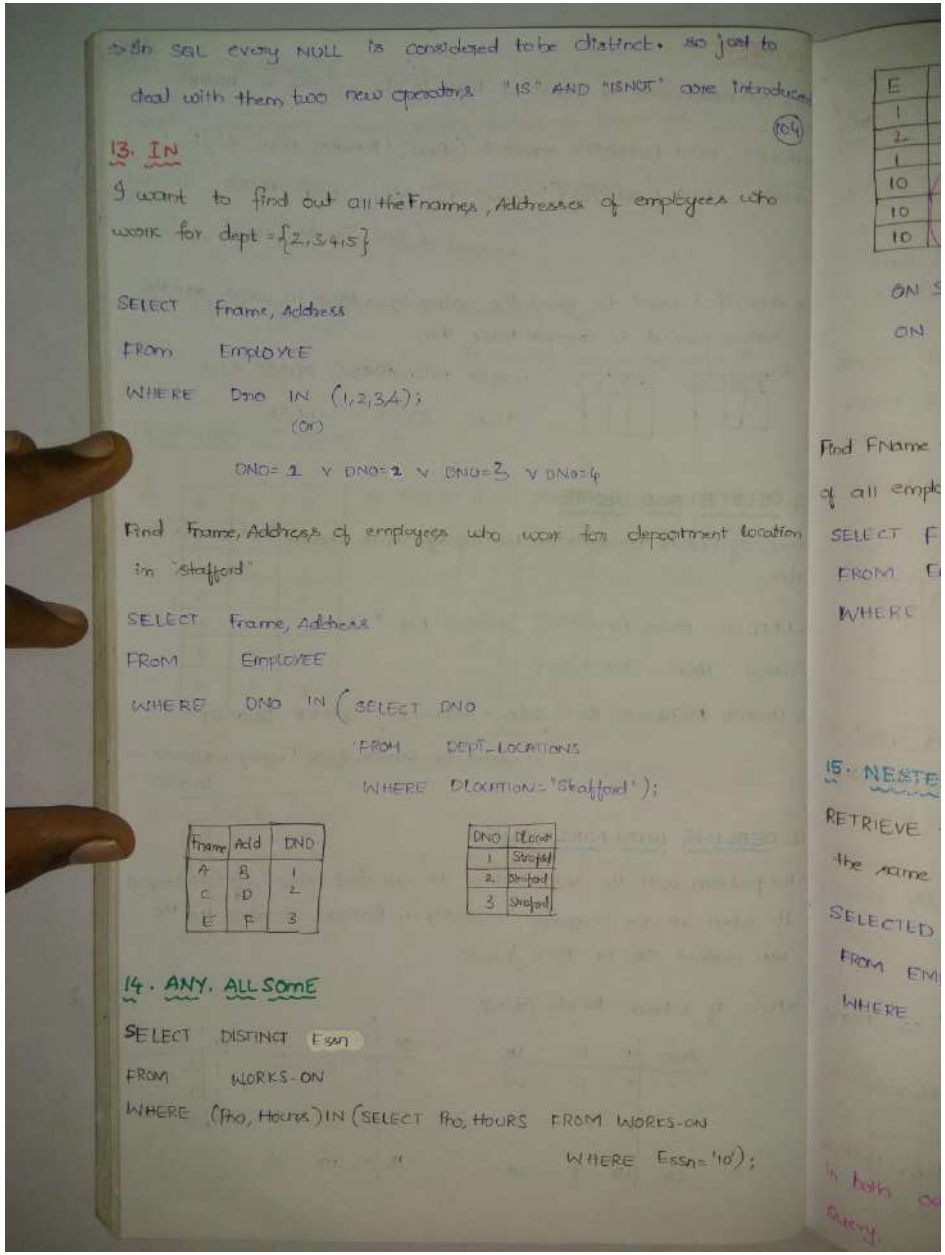
B. I , II, IV and V

C. II ,IV and V — Correct Option | Attempted

Solution: (C)

UNIQUE : By itself, this does not enforce values not to be NULL.
NOT NULL: The purpose of this is preventing NULL values.
FOREIGN KEY: Foreign keys are allowed to be NULL.
PRIMARY KEY: Since this is the identifying value, it has to have a value other than NULL.
CHECK: It can be used to check if a value is NULL:CHECK(X IS NOT NULL).

D. I, II , III , IV and V



=> In SQL every NULL is considered to be distinct. so just to deal with them two new operators "IS" AND "ISNOT" are introduced

104

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
1
2
1
10
10
10

ON

ON

Find FName

of all empl

SELECT F

FROM E

WHERE

15. NESTE

RETRIEVE

the name

SELECTED

FROM EM

WHERE

In both

Query.

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, (105)

(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 depaartment no 5

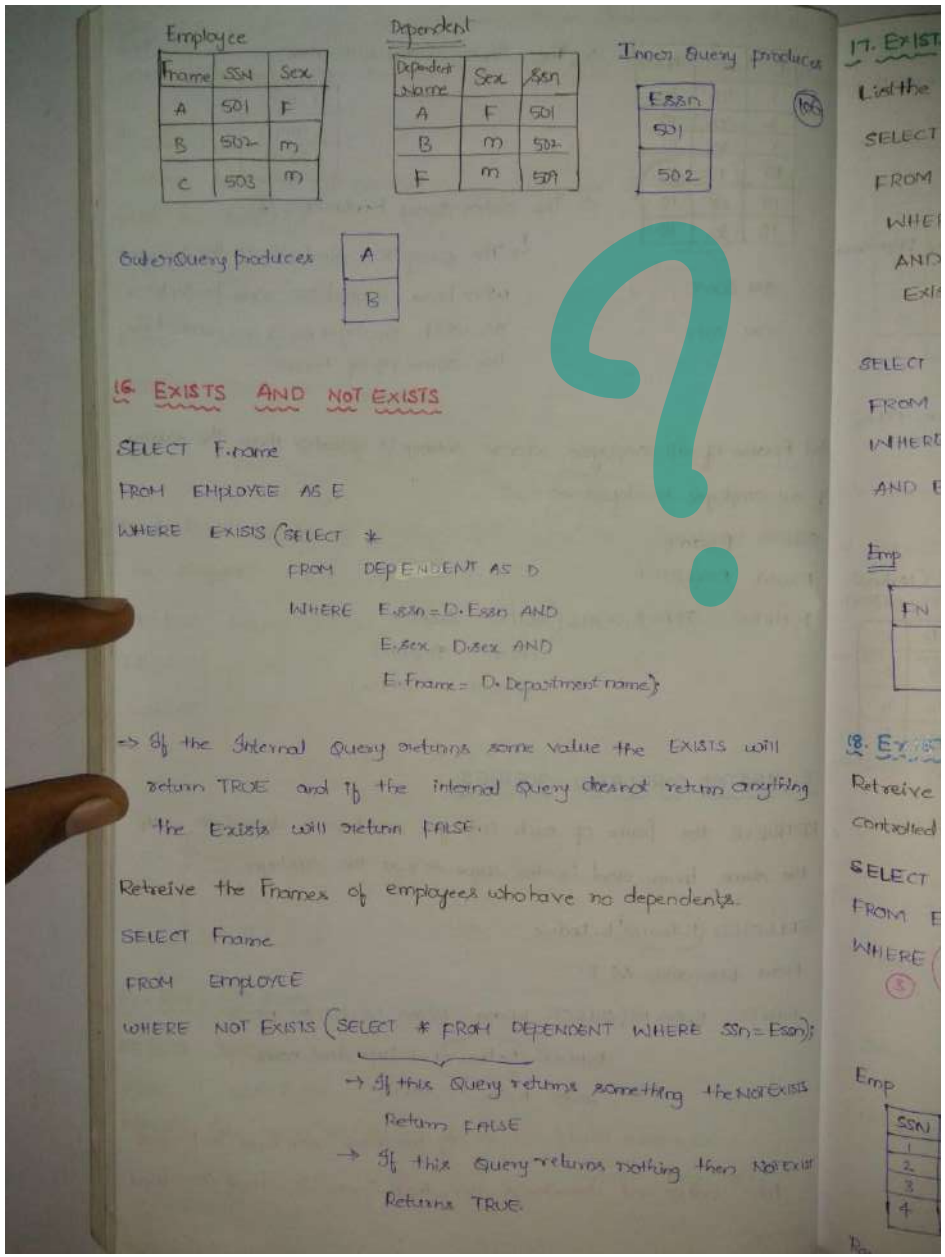
```
SELECT  FName
FROM    EMPLOYEE E
WHERE   SALARY > ALL (SELECT  SALARY
                      FROM    EMPLOYEE
                      WHERE   DNO=5);
```

15. NESTED CORELATED

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  [illegible] 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	Ssn
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.Department 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. EXIST

List the

SELECT

FROM

WHE

AND

EXI

SELECT

FROM

WHERE

AND E

Emp

FN

18. EX

Retreive

controlled

SELECT

FROM E

WHERE

(S)

Emp

SSN

1

2

3

4

Q. 4 ▶ Solutic

Consider the following relational schema:

STUDENT(SNO, SNAME, DEPT)

ENROLL(CNO, SNO, GRADE)

COURSE(CNO, DEPT)

PREREQ(CNO, PNO)

Keys are underlined. The column ENROLL(SNO) is a foreign key referencing STUDENT(SNO). All the occurrences of the columns CNO and PNO, except for the one in COURSE, are foreign keys referencing COURSE(CNO).

```
SELECT E1.CNO, S.SNAME
FROM ENROLL E1, STUDENT S
WHERE E1.SNO = S.SNO
      AND NOT EXISTS (SELECT * FROM ENROLL E2
                      WHERE E2.CNO = E1.CNO
                      AND E2.GRADE > E1.GRADE )
```

What is the output of the Query?

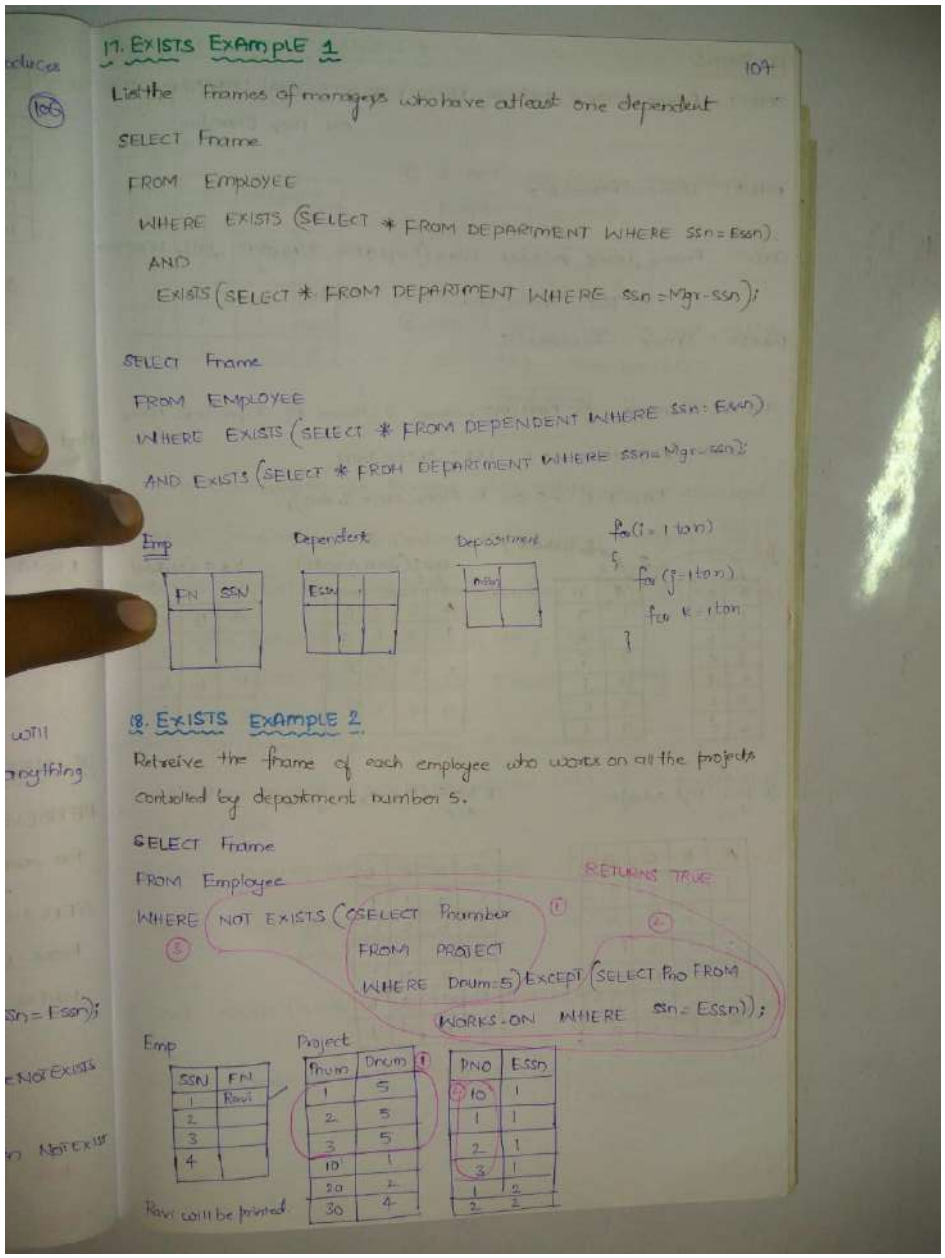
A. For every course, return the names of the highest-scoring students.

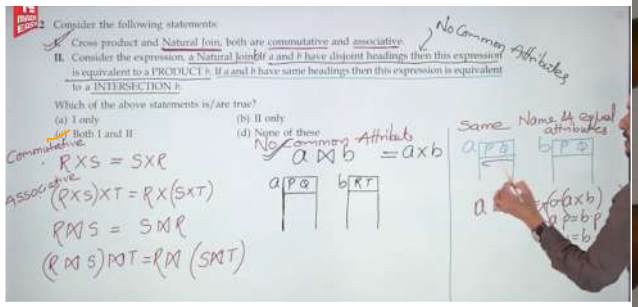
Solution :

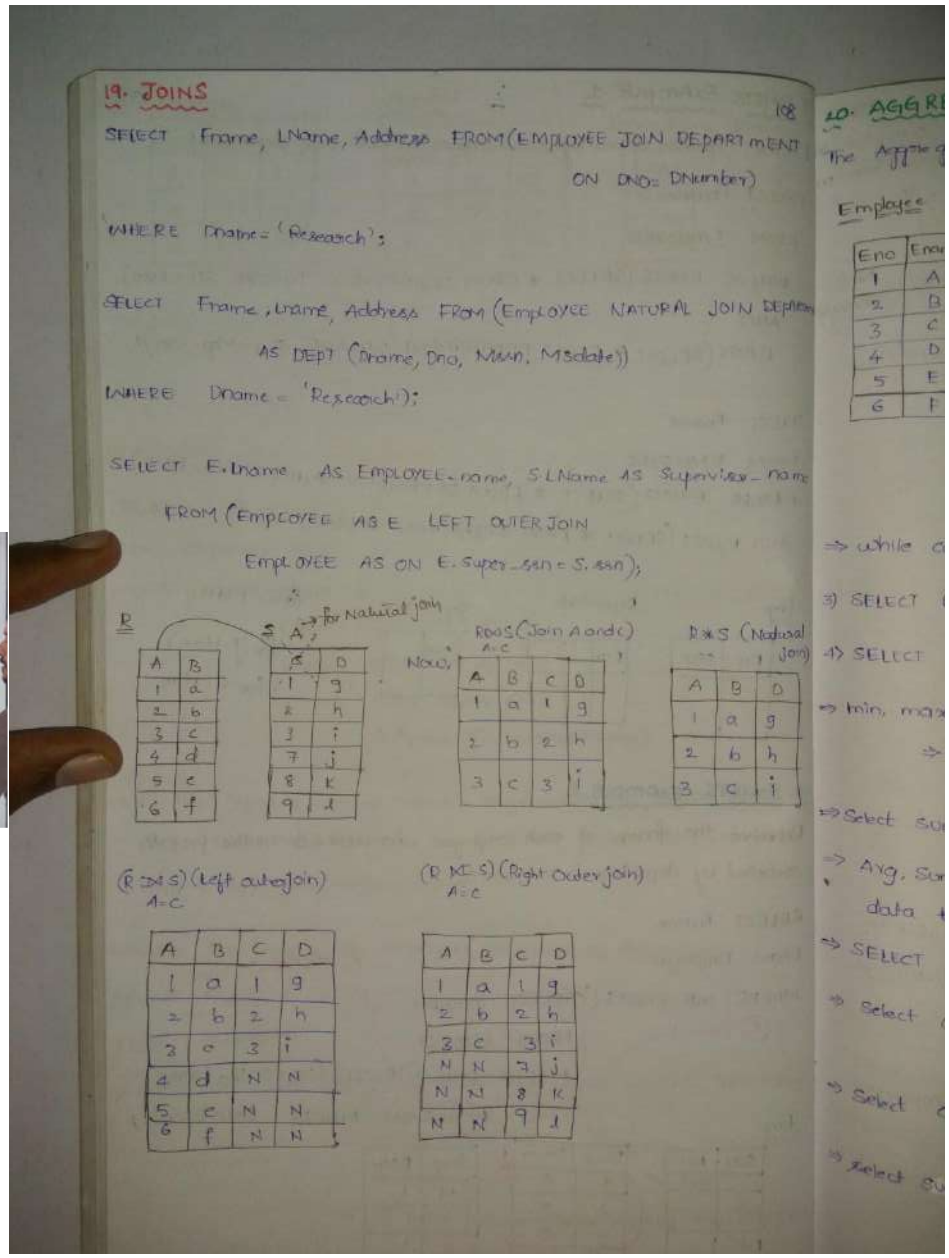
(a)

Given SQL query,

```
SELECT E1.CNO, S.SNAME
FROM ENROLL E1, STUDENT S
WHERE E1.SNO = S.SNO
      AND NOT EXISTS ( SELECT * FROM ENROLL E2
```




Illinois Exam
2 Consider the following statements:
I. Cross product and Natural Join, both are commutative and associative.
II. Consider the expression, a Natural Join b. If a and b have disjoint headings then this expression is equivalent to a PRODUCT b. If a and b have same headings then this expression is equivalent to a INTERSECTION b.
Which of the above statements is/are true?
(a) I only
(b) II only
(c) Both I and II
(d) None of these
No Common Attributes
Same Name & equal attributes
Commutative
R X S = S X R
Associative
(R X S) X T = R X (S X T)
R ⋈ S = S ⋈ R
(R ⋈ S) ⋈ T = R ⋈ (S ⋈ T)
No common Attributes
a ⋈ b = a x b
for Natural join
for (a x b)
p = b p
= b


19. JOINS
108
SELECT Fname, LName, Address FROM (EMPLOYEE JOIN DEPARTMENT
ON DNO = DNumber)
WHERE Dname = 'Research';
SELECT Fname, Lname, Address FROM (Employee NATURAL JOIN Depart
AS DEPT (Dname, Dno, Mssn, Msdate))
WHERE Dname = 'Research';
SELECT E.Lname AS Employee_name, S.LName AS Supervisor_name
FROM (Employee AS E LEFT OUTER JOIN
Employee AS S ON E.Super_ssn = S.ssn);
R
S
for Natural join
Now,
R ⋈ S (Join A and C)
A=C
R ⋈ S (Natural Join)
(R ⟕ S) (Left outer join)
A=C
(R ⟖ S) (Right Outer join)
A=C
20. AGGR
The Agg
Employee
Eno
=> while
3) SELECT
4) SELECT
=> min, max
=>
=> Select
=> Avg, Sum
data
=> SELECT
=> Select
=> Select
=> Select

10. 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 considered

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)

Q. 8

Solution Video | Have any Doubt ?

Which of the following statements is/are False about an SQL query?

A. An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause. — Your option is Correct

B. An SQL query can contain a HAVING clause only if it has a GROUP BY clause

C. All attributes used in the GROUP BY clause must appear in the SELECT clause

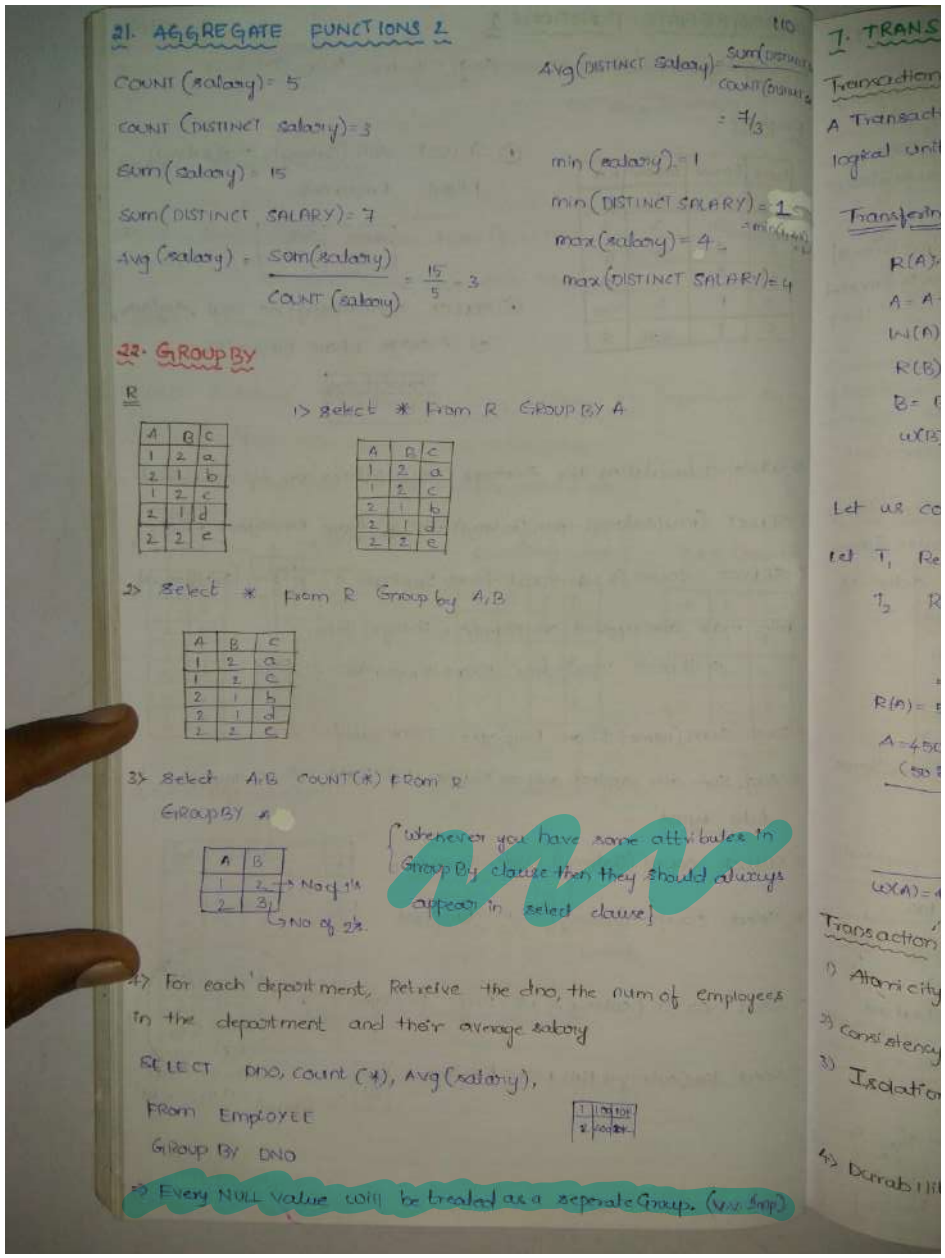
D. Not all attributes used in the GROUP BY clause need to appear in the SELECT clause. — Your option is Correct

YOUR ANSWER - a,d | CORRECT ANSWER - a,d | STATUS - ✓

Solution :
(a, d)

The conclusion is that averages are only calculated using non-NULL values. The general rule is **NULL values are not considered in any aggregate function like** `SUM()`, `AVG()`, `COUNT()`, `MAX()` **and** `MIN()`. The exception to this rule is the `COUNT(*)` function, which counts all rows, even those rows with NULL values. Here's an example:

https://learnsql.com/blog/null-values-group-clause/



Database and Table Manipulation

Command	Description
CREATE DATABASE database_name	Create a database
DROP DATABASE database_name	Delete a database
CREATE TABLE "table_name" ("column_1" "column_1_data_type", "column_2" "column_2_data_type", ...)	Create a table in a database.
ALTER TABLE table_name ADD column_name column_datatype	Add columns in an existing table.
ALTER TABLE table_name DDROP column_name column_datatype	Delete columns in an existing table.
DROP TABLE table_name	Delete a table.

Insert, Update and Delete:

Command	Description
INSERT INTO table_name VALUES (value_1, value_2,....) INSERT INTO table_name (column1, column2,...) VALUES (value_1, value_2,....)	Insert new rows into a table.
UPDATE table_name SET column_name_1 = new_value_1, column_name_2 = new_value_2 WHERE column_name = some_value	Update one or several columns in rows.
DELETE FROM table_name WHERE column_name = some_value	Delete rows in a table.

Functions:

SQL functions	Description
Aggregate Function	This function can produce a single value for an entire group or table. Some Aggregate functions are - • SQL Count function • SQL Sum function • SQL Avg function • SQL Max function • SQL Min function
Arithmetic Function	A mathematical function executes a mathematical operation usually based on input values that are provided as arguments, and return a numeric value as the result of the operation. Some Arithmetic functions are - • abs() • ceil() • floor() • exp() • ln()
Character Function	A character or string function is a function which takes one or more characters or numbers as parameters and returns a character value. Some Character functions are - • lower() • upper() • trim() • translate()

Select:

Command	Description
SELECT column_name(s) FROM table_name	Select data from a table.
SELECT * FROM table_name	Select all data from a table.
SELECT DISTINCT column_name(s) FROM table_name	Select only distinct (different) data from a table.
SELECT column_name(s) FROM table_name WHERE column operator value AND column operator value OR column operator value AND (... OR ...) ...	Select only certain data from a table.
SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...)	The IN operator may be used if you know the exact value you want to return for at least one of the columns.
SELECT column_name(s) FROM table_name ORDER BY row_1, row_2 DESC, row_3 ASC, ...	Select data from a table with sort the rows.
SELECT column_1, ..., SUM(group_column_name) FROM table_name GROUP BY group_column_name	The GROUP BY clause is used with the SELECT statement to make a group of rows based on the values of a specific column or expression. The SQL AGGREGATE function can be used to get summary information for every group and these are applied to individual group.
SELECT column_name(s) INTO new_table_name FROM source_table_name WHERE query	Select data from table(S) and insert it into another table.
SELECT column_name(s) IN external_database_name FROM source_table_name WHERE query	Select data from table(S) and insert it in another database.

Writing something with query

Write a query to display a string "This is SQL Exercise, Practice and Solution".

Sample Solution:

```
1 | SELECT "This is SQL Exercise, Practice and Solution";
```

Write a query to display three numbers in three columns.

Sample Solution:

```
1 | SELECT 5, 10, 15;
```

Write a SQL query to give the name of the 'Physics' winners since the year 1950.

Sample table: nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist

Sample Solution :

```
1 | SELECT winner
2 | FROM nobel_win
3 | WHERE year>=1950
4 | AND subject='Physics';
```

SQL Basic Select Statement: Exercise-18 with Solution.

Write a SQL query to show all the details of the winners with first name Louis.

Sample table: nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist

Sample Solution:

```
1 | SELECT *
2 |  FROM nobel_win
3 |    WHERE winner LIKE 'Louis %';
```

Write a SQL query to show all the winners in Physics for 1970 together with the winner of Economics for 1971.

Sample table: nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist
1971	Peace	Willy Brandt	Germany	Chancellor

Sample Solution:

```
1 | SELECT * FROM nobel_win  WHERE (subject ='Physics' AND year=1970) UNION (SELECT * FROM nobel_win  WHERE (subject ='
```

Write a SQL query to show all the winners of nobel prize in the year 1970 except the subject Physiology and Economics

Sample table: nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist
1971	Peace	Willy Brandt	Germany	Chancellor

Sample Solution:

```
1 | SELECT *
2 |  FROM nobel_win
3 |    WHERE year=1970
4 |     AND subject NOT IN('Physiology','Economics');
```

Write a SQL query to find all the details of the nobel winners for the subject not started with the letter 'P' and arranged the list as the most recent comes first, then by name in order.

Sample table : nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist
1971	Peace	Willy Brandt	Germany	Chancellor

Sample Solution :

```
1 | SELECT *
2 |  FROM nobel_win
3 |   WHERE subject NOT LIKE 'P%'
4 |    ORDER BY year DESC, winner;
```

Important and different

Write a SQL query to find all the details of 1970 winners by the ordered to subject and winner name; but the list contain the subject Economics and Chemistry at last.

Sample table: nobel_win

YEAR	SUBJECT	WINNER	COUNTRY	CATEGORY
1970	Physics	Hannes Alfven	Sweden	Scientist
1970	Physics	Louis Neel	France	Scientist
1970	Chemistry	Luis Federico Leloir	France	Scientist
1970	Physiology	Ulf von Euler	Sweden	Scientist
1970	Physiology	Bernard Katz	Germany	Scientist
1970	Literature	Aleksandr Solzhenitsyn	Russia	Linguist
1970	Economics	Paul Samuelson	USA	Economist
1970	Physiology	Julius Axelrod	USA	Scientist
1971	Physics	Dennis Gabor	Hungary	Scientist
1971	Chemistry	Gerhard Herzberg	Germany	Scientist
1971	Peace	Willy Brandt	Germany	Chancellor

Not for GATE

Sample Solution :

```
SELECT *
FROM nobel_win
WHERE year=1970
ORDER BY
 CASE
    WHEN subject IN ('Economics','Chemistry') THEN 1
    ELSE 0
 END ASC,
 subject,
 winner;
```

Write a SQL statement to make a list with order no, purchase amount, customer name and their cities for those orders which order amount between 500 and 2000.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT  a.ord_no,a.purch_amt,
b.cust_name,b.city
FROM orders a,customer b
WHERE a.customer_id=b.customer_id
AND a.purch_amt BETWEEN 500 AND 2000;
```

Output of the Query:

ord_no	purch_amt	cust_name	city
70007	948.50	Graham Zusi	California
70010	1983.43	Fabian Johnson	Paris

Write a SQL query to find all the products with a price between Rs.200 and Rs.600.

Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution :

```
SELECT * FROM item_mast
   WHERE pro_price BETWEEN 200 AND 600;
```

Write a SQL statement to display all customers, who are either belongs to the city New York or had a grade above 100.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution :

```
SELECT *
FROM customer
WHERE city = 'New York' OR grade>100;
```

Write a SQL query to calculate the average price of all products of the manufacturer which code is 16.

Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution:

```
SELECT AVG(pro_price) FROM item_mast
  WHERE pro_com=16;
```

Write a SQL query to display those customers who are neither belongs to the city New York nor grade value is more than 100.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution :

```
SELECT *
FROM customer
WHERE NOT (city = 'New York' OR grade>100);
```

Write a SQL query to display the name and price of all the items with a price is equal or more than Rs.250, and the list contain the larger price first and then by name in ascending order.

Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution:

```
SELECT pro_name, pro_price
     FROM item_mast
    WHERE pro_price >= 250
 ORDER BY pro_price DESC, pro_name;
```

Write a SQL statement to display either those orders which are not issued on date 2012-09-10 and issued by the salesman whose ID is 5005 and below or those orders which purchase amount is 1000.00 and below.

Sample table : orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution :

```
SELECT *
FROM  orders
WHERE NOT ((ord_date ='2012-09-10'
AND salesman_id>5005)
OR purch_amt>1000.00);
```

Write a SQL query to display the average price of the items for each company, showing only the company code.

Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution:

```
SELECT AVG(pro_price), pro_com
    FROM item_mast
GROUP BY pro_com;
```

Write a SQL query to display order number, purchase amount, the achieved and unachieved percentage (%) for those order which exceeds the 50% of the target value of 6000.

Sample table: orders

different

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution :

```
SELECT ord_no,purch_amt,
(100*purch_amt)/6000 AS "Achieved %",
(100*(6000-purch_amt)/6000) AS "Unachieved %"
FROM  orders
WHERE (100*purch_amt)/6000>50;
```

Write a SQL query to find the name and price of the cheapest item(s).

Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution:

```
SELECT pro_name, pro_price
   FROM item_mast
   WHERE pro_price =
    (SELECT MIN(pro_price) FROM item_mast);
```

I did Mistake here

I only wrote this

Output of the Query:

pro_name	pro_price
ZIP drive	250.00
Mouse	250.00

Will not work

Practice Online

```
SELECT pro_name, pro_price FROM item_mast ORDER BY pro_price limit 1;
```

Escape Character

Write a SQL statement to find those rows from the table testtable which contain the escape character underscore (_) in its column 'col1'.

Sample table: testtable

```
col1
--------------------
A001/D3-402\44_/100/2015
A001_\D3-402\44_/100/2015
A001_D3-402-2014-2015
A002_D3-401-2014-2015
A001/D3_401
A001/D3_402\44
A001/D3_402\44\2015
A001/D3-402\45\2015/200
A001/D3_402\45\2015\100
A001/D3_402\45\2015/300
A001/D3-402\44
```

Sample Solution:

```
SELECT *
FROM testtable
WHERE col1 LIKE '%/_%' ESCAPE '/';
```

Write a SQL statement to find those rows from the table testtable which contain the escape character (/) in its column 'col1'.

Sample table: testtable

```
col1
--------------------
A001/D3-402\44_/100/2015
A001_\D3-402\44_/100/2015
A001_D3-402-2014-2015
A002_D3-401-2014-2015
A001/D3_401
A001/D3_402\44
A001/D3_402\44\2015
A001/D3-402\45\2015/200
A001/D3_402\45\2015\100
A001/D3_402\45\2015/300
A001/D3-402\44
```

Sample Solution:

```
SELECT *
FROM testtable
WHERE col1 LIKE '%//%' ESCAPE '/';
```

Write a SQL statement to find those rows from the table testtable which contain the character (%) in its column 'col1'.

Sample table: testtable

```
col1
--------------------
A001/D3-402\44_/100/2015
A001_\D3-402\44_/100/2015
A001_D3-402-2014-2015
A002_D3-401-2014-2015
A001/D3_401
A001/D3_402\44
A001/D3_402\44\2015
A001/D3-402\45\2015/200
A001/D3_402\45\2015\100
A001/D3_402\45\2015/300
A001/D3-402\44
```

Sample Solution:

```
SELECT *
FROM testtable
WHERE col1 LIKE '%/%%' ESCAPE'/';
```

Write a SQL statement to find that customer with all information who does not get any grade except NULL.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT *
FROM customer
WHERE grade IS NULL;
```

SQL Aggregate Functions: Exercise-4 with Solution

Write a SQL statement to know how many customers have listed their names.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT COUNT(*)
FROM customer;
```

Write a SQL statement to find the number of salesman currently listing for all of their customers.

Sample table: orders

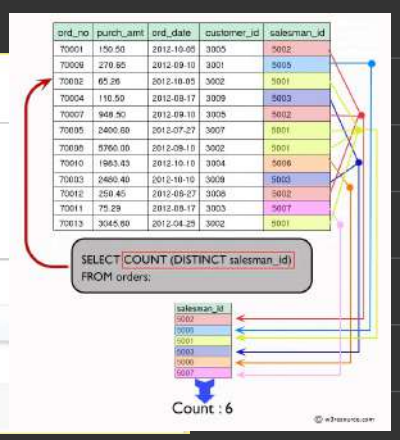
ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

Sample Solution:

```
SELECT COUNT (DISTINCT salesman_id)
FROM orders;
```

Output of the Query:

```
count
6
```



Write a SQL statement find the number of customers who gets at least a gradation for his/her performance.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT COUNT (ALL grade)
FROM customer;
```

Output of the Query:

```
count
7
```

Write a SQL statement which selects the highest grade for each of the cities of the customers.

Sample table: customer

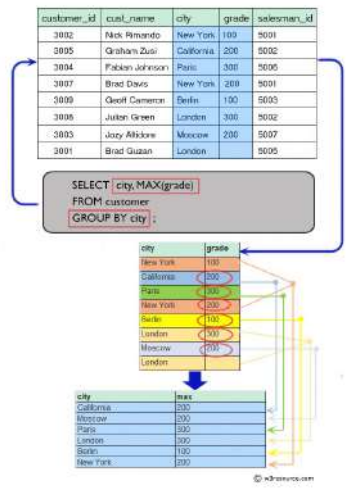
customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT city,MAX(grade)
FROM customer
GROUP BY city;
```

Output of the Query:

city	max
London	300
Paris	300
New York	200
California	200
Berlin	100
Moscow	200



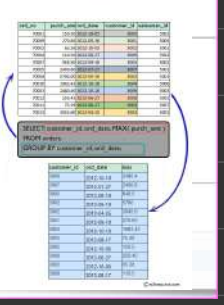
Write a SQL statement to find the highest purchase amount ordered by the each customer with their ID and highest purchase amount.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution:

```
SELECT customer_id,MAX(purch_amt)
FROM orders
GROUP BY customer_id;
```



Write a SQL statement to find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution:

```
SELECT customer_id,ord_date,MAX(purch_amt)
FROM orders
GROUP BY customer_id,ord_date;
```

Write a SQL statement to find the highest purchase amount with their ID and order date, for only those customers who have highest purchase amount in a day is more than 2000.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution:

```
SELECT customer_id,ord_date,MAX(purch_amt)
FROM orders
GROUP BY customer_id,ord_date
HAVING MAX(purch_amt)>2000.00;
```

Write a SQL statement to find the highest purchase amount with their ID and order date, for only those customers who have a higher purchase amount in a day is within the list 2000, 3000, 5760 and 6000.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003

Sample Solution:

```
SELECT customer_id,ord_date,MAX(purch_amt)
FROM orders
GROUP BY customer_id,ord_date
HAVING MAX(purch_amt) IN(2000 ,3000,5760, 6000);
```

Write a SQL statement that count the number of salesmen for whom a city is specified. Note that there may be spaces or no spaces in the city column if no city is specified.

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample Solution :

```
SELECT COUNT(*)
FROM salesman
WHERE city IS NOT NULL;
```

Write a SQL query to display the average price of each company's products, along with their code.

Sample table: item_mast

pro_id	pro_name	pro_price	pro_com
101	Mother Board	3200	11
102	Key Board	450	14
103	ZIP Drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD Drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

Sample Solution:

```
SELECT AVG(pro_price) AS "Average Price",
       pro_com AS "Company ID"
    FROM item_mast
GROUP BY pro_com;
```

Write a SQL statement to find out the number of orders booked for each day and display it in such a format like "For 2001-10-10 there are 15 orders".

Sample table: orders

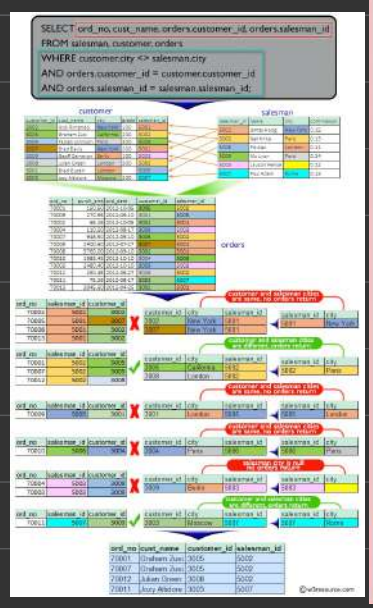
ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT ' For',ord_date,',there are',
COUNT (DISTINCT ord_no),'orders.'
FROM orders
GROUP BY ord_date;
```

Output of the Query:

?column?	ord_date	?column?	count	?column?
For	2012-04-25	,there are	1	orders.
For	2012-06-27	,there are	1	orders.
For	2012-07-27	,there are	1	orders.



Write a SQL statement to display all those orders by the customers not located in the same cities where their salesmen live.

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT ord_no, cust_name, orders.customer_id, orders.salesman_id
FROM salesman, customer, orders
WHERE customer.city <> salesman.city
AND orders.customer_id = customer.customer_id
AND orders.salesman_id = salesman.salesman_id;
```

Write a SQL statement to display the orders with all information in such a manner that, the older order date will come first and the highest purchase amount of same day will come first.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT *
FROM orders
ORDER BY ord_date, purch_amt DESC;
```

Output of the Query:

ord_no	purch_amt	ord_date	customer_id	salesman_id
70011	3045.60	2012-04-25	3002	5001
70012	250.45	2012-06-27	3008	5002
70005	2400.60	2012-07-27	3007	5001
70004	110.50	2012-08-17	3009	5003
70001	75.29	2012-08-17	3003	5007

Write a query that produces all customers with their name, city, salesman and commission, who served by a salesman and the salesman works at a rate of the commission within 12% to 14%.

Sample table: salesman

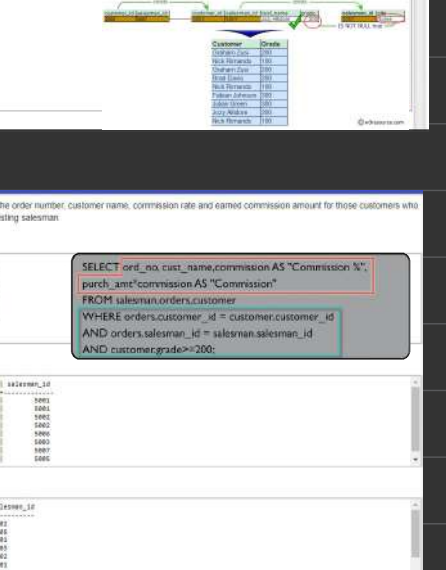
salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT customer.cust_name AS "Customer",
customer.city AS "City",
salesman.name AS "Salesman",
salesman.commission
FROM customer,salesman
WHERE customer.salesman_id = salesman.salesman_id
AND salesman.commission
BETWEEN .12 AND .14;
```



Write a SQL statement to make a report with salesman ID, order date and highest purchase amount in such an arrangement that, the smallest salesman ID will come first along with their smallest order date.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution :

```
SELECT salesman_id,ord_date,MAX(purch_amt)
FROM orders
GROUP BY salesman_id,ord_date
ORDER BY salesman_id,ord_date;
```

Output of the Query:

salesman_id	ord_date	max
5001	2012-04-25	3045.60
5001	2012-07-27	2400.60
5001	2012-09-10	5760.00
5002	2012-10-05	65.26

Write a SQL statement that shorts out the customer and their grade who made an order. Each of the customers must have a grade and served by at least a salesman, who belongs to a city.

Sample table: salesman

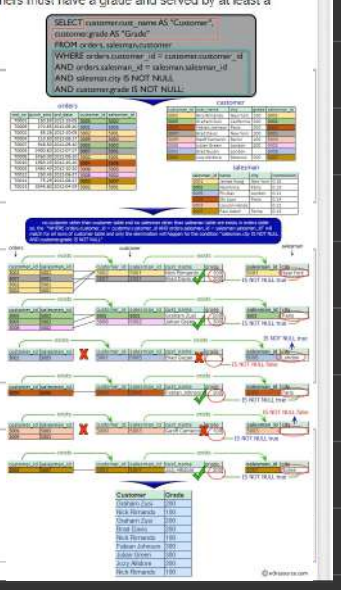
salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006



Write a SQL statement to display customer name, city and grade in such a manner that, the customer holding highest grade will come first.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Unique

Write a SQL statement to make a report with customer ID in such a manner that, the largest number of orders booked by the customer will come first along with their highest purchase amount.

Sample table: orders

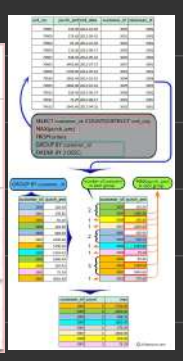
ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT customer_id, COUNT(DISTINCT ord_no),
MAX(purch_amt)
FROM orders
GROUP BY customer_id
ORDER BY 2 DESC;
```

Output of the Query:

customer_id	count	max
3002	3	5760.00
3005	2	2400.60
3001	2	948.50



Write a SQL statement make a report with order date in such a manner that, the latest order date will come last along with the total purchase amount and total commission (15% for all salesmen) for that date.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT ord_date, SUM(purch_amt),
SUM(purch_amt)*.15
FROM orders
GROUP BY ord_date
ORDER BY ord_date;
```

Output of the Query:

ord_date	sum	fcolumn?
2012-04-25	3045.60	456.8400
2012-06-27	250.45	37.5675
2012-07-27	2400.60	360.0900

Write a SQL statement that produces all orders with the order number, customer name, commission rate and earned commission amount for those customers who carry their grade is 200 or more and served by an existing salesman.

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

```
SELECT ord_no, cust_name,commission AS "Commission %",
purch_amt*commission AS "Commission"
FROM salesman,orders,customer
WHERE orders.customer_id = customer.customer_id
AND orders.salesman_id = salesman.salesman_id
AND customer.grade>=200;
```

Write a query to find those customers with their name and those salesmen with their name and city who lives in the same city.

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample Solution:

```
SELECT customer.cust_name,
salesman.name, salesman.city
FROM salesman, customer
WHERE salesman.city = customer.city;
```

35. Write a query in SQL to display job Title, the difference between minimum and maximum salaries for those jobs which max salary within the range 12000 to 18000.

Sample table : jobs

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
FI_ACCOUNT	Accountant	4200	9000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
PU_MAN	Purchasing Manager	8000	15000
PU_CLERK	Purchasing Clerk	2500	5500

Sample Solution:

```
SELECT job_title, max_salary-min_salary AS salary_differences
    FROM jobs
        WHERE max_salary BETWEEN 12000 AND 18000;
```

Sample Output:

job_title	salary_differences
Finance Manager	7000
Accounting Manager	7000
Sales Representative	6000
Purchasing Manager	7000
Marketing Manager	6000

(5 rows)

8. Write a query to display all customers with orders on October 5, 2012.

Sample table: Customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

Sample table: Orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006

Sample Solution:

```
SELECT *
FROM customer a,orders  b
WHERE a.customer_id=b.customer_id
AND b.ord_date='2012-10-05';
```

Output of the Query:

customer_id	cust_name	city	grade	salesman_id	ord_no	purch_amt	ord_date	customer_id	salesm
3002	Nick Rimando	New York	100	5001	70001	65.26	2012-10-05	3002	5001
3005	Graham Zusi	California	200	5002	70001	150.50	2012-10-05	3005	5002

SQL SORTING and FILTERING on HR Database: Exercise-1 with Solution

1. Write a query in SQL to display the full name (first and last name), and salary for those employees who earn below 6000.

Sample table: employees

Sample Solution:

```
SELECT first_name ||' '||last_name AS Full_Name, salary
  FROM employees
    WHERE salary < 6000;
```

For Concatenation, || is used

Sample Output:

full_name	salary
David Austin	4800.00
Valli Pataballa	4800.00
Diana Lorentz	4200.00
Alexander Khoo	3100.00
Shelli Baida	2900.00
Sigal Tobias	2800.00
Guy Himuro	2600.00

w3resource

22. Write a query in SQL to display the ID for those employees who did two or more jobs in the past.

Sample table : job_history

Sample Solution:

```
SELECT employee_id
  FROM job_history
    GROUP BY employee_id
      HAVING COUNT(*) >=2;
```

Sample Output:

employee_id
101
176
200

(3 rows)

Practice Online

```
SELECT employee_id FROM job_history GROUP BY employee_id
HAVING count(employee_id)>=2;
```

23. Write a query in SQL to display job ID, number of employees, sum of salary, and difference between highest salary and lowest salary for a job.

Sample table: employees

Sample Solution:

```
SELECT job_id, COUNT(*), SUM(salary),
    MAX(salary)-MIN(salary) AS salary_difference
      FROM employees
        GROUP BY job_id;
```

Sample Output:

job_id	count	sum	salary_difference
AC_ACCOUNT	1	8300.00	0.00
ST_MAN	5	36400.00	2400.00
IT_PROG	5	28800.00	4800.00
SA_MAN	5	61000.00	3500.00
AD_PRES	1	24000.00	0.00
AC_MGR	1	12008.00	0.00
FI_MGR	1	12008.00	0.00

4. Write a query in SQL to display all the information for all employees without any department number.

Sample table: employees

Sample Solution:

```
SELECT *
  FROM employees
    WHERE department_id IS NULL;
```

Sample Output:

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id

(0 rows)

SQL SORTING and FILTERING on HR Database: Exercise-24 with Solution

24. Write a query in SQL to display job ID for those jobs that were done by two or more for more than 300 days.

Sample table : job_history

Sample Solution:

```
SELECT job_id
    FROM job_history
      WHERE end_date-start_date >300
        GROUP BY job_id
          HAVING COUNT(*)>=2;
```

Sample Output:

job_id
AC_ACCOUNT
ST_CLERK

(2 rows)

25. Write a query in SQL to display the country ID and number of cities in that country we have.

Sample table : locations

Sample Solution:

```
SELECT country_id, COUNT(*)
    FROM locations
      GROUP BY country_id;
```

Sample Output:

country_id	count
CH	2
DE	1
US	4
AU	1
IT	2
CA	1
JP	2

6. Write a query in SQL to display the full name (first and last), hire date, salary, and department number for those employees whose first name does not contain the letter M. The result set must be in ascending order by department number.

Sample table : employees

Sample Solution:

```
SELECT first_name || ' ' || last_name as Full_Name, hire_date,
  salary, department_id
   FROM employees
     WHERE first_name NOT LIKE '%M%'
       ORDER BY department_id;
```

Sample Output:

Full_name	hire_date	salary	department_id
Kimberely Grant	2007-05-24	7000.00	0
JenniFer Whalen	2003-09-17	4400.00	10
Pat Fay	2005-08-17	6000.00	20
Guy Himuro	2006-11-15	2600.00	30
Alexander Khoo	2003-05-18	3100.00	30
Den Raphaely	2002-12-07	11000.00	30

26. Write a query in SQL to display the manager ID and number of employees managed by the manager.

Sample table: employees

Sample Solution:

```
SELECT manager_id, COUNT(*)
    FROM employees
      GROUP BY manager_id;
```

Sample Output:

manager_id	count
205	1
122	8
120	8
101	5
103	4
108	5
145	0
100	14
201	1
124	8

29. Write a query in SQL to display the average salary of employees for each department who gets a commission percentage.

Sample table: employees

Sample Solution:

```
SELECT department_id, AVG(salary)
    FROM employees
      WHERE commission_pct IS NOT NULL
        GROUP BY department_id;
```

Sample Output:

department_id	avg
90	19333.333333333333
20	9500.000000000000000
100	8600.000000000000000
40	6500.000000000000000
110	10150.000000000000000
80	8955.882352941176470
70	10000.000000000000000

7. Write a query in SQL to display all the information of employees whose salary is in the range of 8000 and 12000 and commission is not null or department number is except the number 40, 120 and 70 and they have been hired before June 5th, 1987.

Sample table employees

Sample Solution:

```
SELECT *
 FROM employees
  WHERE salary BETWEEN 8000 AND 12000
    AND commission_pct IS NOT NULL
      OR  department_id NOT IN (40 , 120 , 70)
        AND  hire_date < '1987-06-05'
```

SQL SORTING and FILTERING on HR Database: Exercise-30 with Solution

30. Write a query in SQL to display those departments where any manager is managing 4 or more employees.

Sample table : employees

Sample Solution:

```
SELECT DISTINCT department_id
    FROM employees
      GROUP BY department_id, manager_id
        HAVING COUNT(employee_id) >=4;
```

Sample Output:

department_id
80
50

Hiếu Nguyễn Trung - 9 months ago - edited

SELECT DISTINCT department_id FROM employees
WHERE manager_id IN (SELECT manager_id FROM employees GROUP BY manager_id HAVING COUNT(*) >=4);

Reply · Share

9. Write a query in SQL to display the full name (first and last), the phone number and email separated by hyphen, and salary, for those employees whose salary is within the range of 9000 and 17000. The column headings assign with Full_Name, Contact_Details and Remuneration respectively

Sample table : employees

Sample Solution:

```
SELECT first_name ||' '||last_name AS Full_Name,
       phone_number ||' - '|| email AS Contact_Details,
       salary AS Remuneration
         FROM employees
           WHERE salary BETWEEN 9000 AND 17000;
```

Sample Output:

Full_name	Contact_details	remuneration
Neena Kochhar	515.123.4568 - NKOCHHAR	17000.00
Lex De Haan	515.123.4569 - LDEHAAN	17000.00
Alexander Hunold	590.423.4567 - AHUNOLD	9000.00
Nancy Greenberg	515.124.4569 - NGREENBE	12008.00

11. Write a query in SQL to display the full name (first and last) name, and salary, for all employees whose salary is out of the range 7000 and 15000 and make the result set in ascending order by the full name.

Sample table : employees

Sample Solution:

```
SELECT first_name || ' ' || last_name as Name, salary
 FROM employees
   WHERE salary NOT BETWEEN 7000 AND 15000
     ORDER BY first_name || ' ' || last_name;
```

Sample Output:

name	salary
Alana Walsh	3100.00
Alexander Khoo	3100.00
Alexis Bull	4100.00
Amit Banda	6200.00

31. Write a query in SQL to display those departments where more than ten employees work who got a commission percentage.

Sample table : employees

Sample Solution:

```
SELECT department_id
    FROM employees
      WHERE commission_pct IS NOT NULL
        GROUP BY department_id
          HAVING COUNT(commission_pct)>10;
```

Sample Output:

department_id
80
50

(2 rows)

14. Write a query in SQL to display the full name (first and last name), salary, and manager number for those employees who is working under a manager.

Sample table: employees

Sample Solution:

```
SELECT first_name ||' '||last_name AS Full_Name, salary, manager_id
    FROM employees
      WHERE manager_id IS NOT NULL;
```

```
SELECT department_id FROM employees WHERE commission_pct IS NOT NULL
GROUP BY department_id
HAVING COUNT(employee_id)>10;
```

w3resource

33. Write a query in SQL to display the employee ID and the date on which he ended his previous job.

Sample table : job_history

Sample Solution:

```
SELECT employee_id , MAX(end_date)
FROM job_history
WHERE employee_id IN (SELECT employee_id
FROM job_history
GROUP BY 1
HAVING COUNT(employee_id) > 1)
GROUP BY 1
```

Sample Output:

employee_id	max
101	2005-03-15
200	2006-12-31
176	2007-12-31

(7 rows)

34. Write a query in SQL to display the job ID for those jobs which average salary is above 8000.

Sample table : employees

Sample Solution:

```
SELECT job_id, AVG(salary)
    FROM employees
      GROUP BY job_id
        HAVING AVG(salary)>8000;
```

Sample Output:

job_id	avg
AC_ACCOUNT	8300.000000000000000
SA_MAN	12200.000000000000000
AD_PRES	24000.000000000000000
AC_MGR	12008.000000000000000
FI_MGR	12008.000000000000000
MK_MAN	13000.000000000000000
PR_REP	10000.000000000000000
AD_VP	17000.000000000000000
SA_REP	8350.000000000000000

17. Write a query in SQL to display all the information for all employees who have the letters D, S, or N in their first name and also arrange the result in descending order by salary.

Sample table : employees

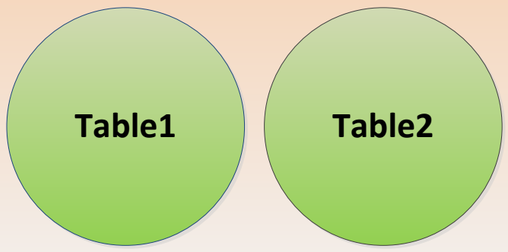
Sample Solution:

```
SELECT *
 FROM employees
  WHERE first_name LIKE '%D%'
    OR first_name LIKE '%S%'
    OR first_name LIKE '%N%'
     ORDER BY salary DESC;
```

https://www.w3resource.com/sql-exercises/sql-subqueries-exercises.php

TSQL JOIN TYPES

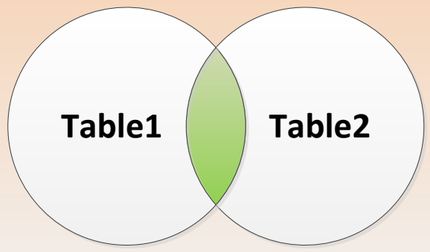
Created by Steve Stedman



SELECT from two tables

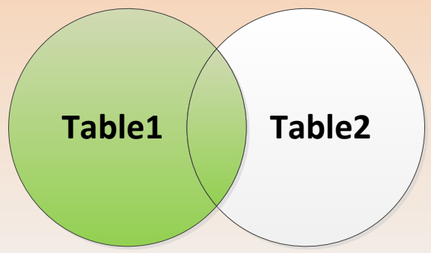
```
SELECT *
  FROM Table1;

SELECT *
  FROM Table2;
```



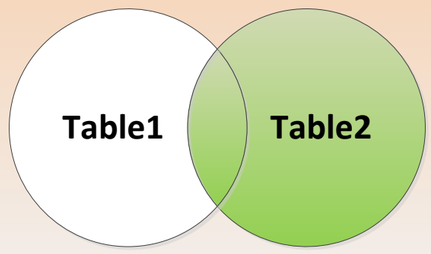
INNER JOIN

```
SELECT *
  FROM Table1 t1
 INNER JOIN Table2 t2
    ON t1.fk = t2.id;
```



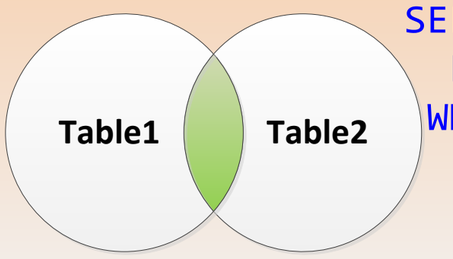
LEFT OUTER JOIN

```
SELECT *
  FROM Table1 t1
 LEFT OUTER JOIN Table2 t2
   ON t1.fk = t2.id;
```



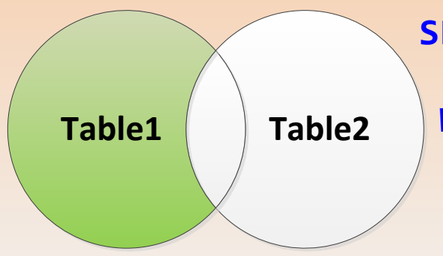
RIGHT OUTER JOIN

```
SELECT *
  FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
   ON t1.fk = t2.id;
```



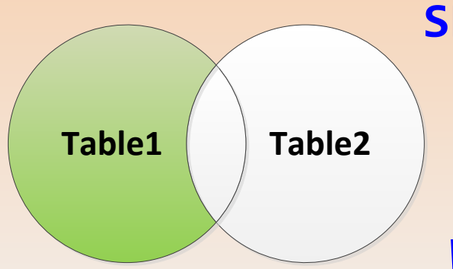
SEMI JOIN

```
SELECT *
  FROM Table1 t1
WHERE EXISTS (SELECT 1
                FROM Table2 t2
               WHERE t1.fk = t2.id
              );
```



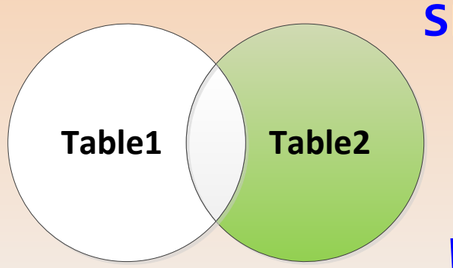
ANTI SEMI JOIN

```
SELECT *
  FROM Table1 t1
WHERE NOT EXISTS (SELECT 1
                    FROM Table2 t2
                   WHERE t1.fk = t2.id
                  );
```



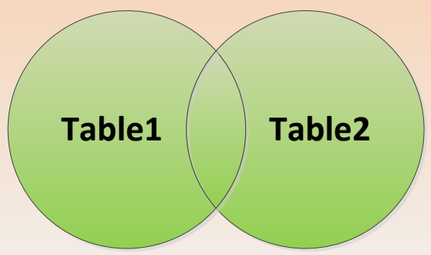
LEFT OUTER JOIN with exclusion
– replacement for a NOT IN

```
SELECT *
  FROM Table1 t1
  LEFT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t2.id IS NULL;
```



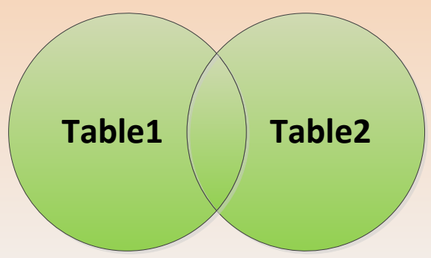
RIGHT OUTER JOIN with exclusion
– replacement for a NOT IN

```
SELECT *
  FROM Table1 t1
  RIGHT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t1.fk IS NULL;
```



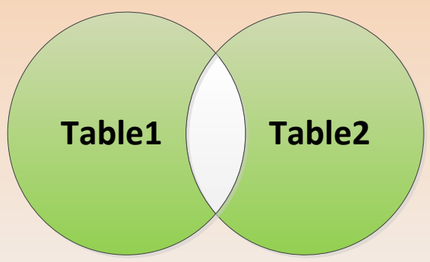
FULL OUTER JOIN

```
SELECT *
  FROM Table1 t1
  FULL OUTER JOIN Table2 t2
    ON t1.fk = t2.id;
```



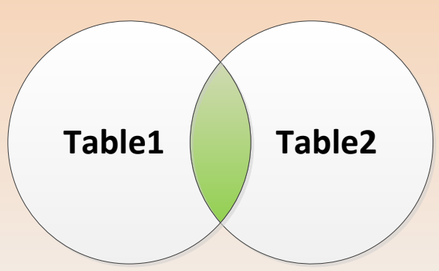
CROSS JOIN, the Cartesian product

```
SELECT *
  FROM Table1 t1
  CROSS JOIN Table2 t2;
```



FULL OUTER JOIN with exclusion

```
SELECT *
  FROM Table1 t1
  FULL OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t1.fk IS NULL
    OR t2.id IS NULL;
```



NON-EQUI INNER JOIN

```
SELECT *
  FROM Table1 t1
 INNER JOIN Table2 t2
    ON t1.fk >= t2.id;
```

Created By Steve Stedman http://SteveStedman.com
Twitter @SqlEmt http://linkedin.com/in/stevestedman

7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

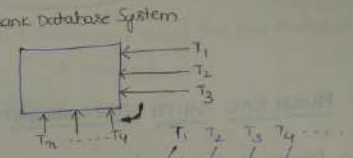
Transaction:

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);

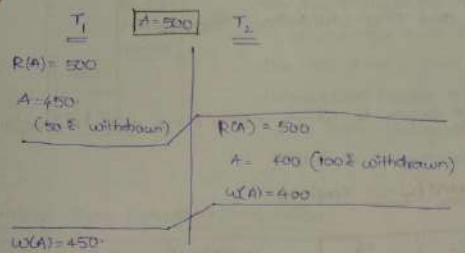
Sequence of steps



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)

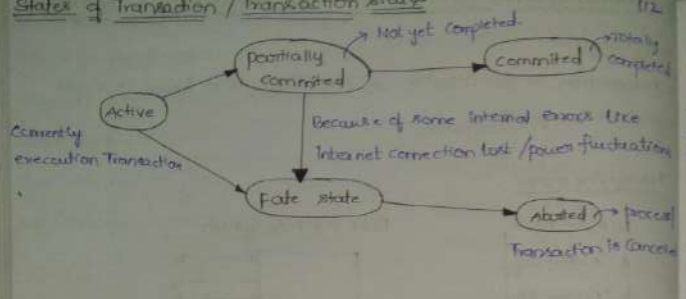
1) Atomicity : All or None should happen → Transaction Manager

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

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

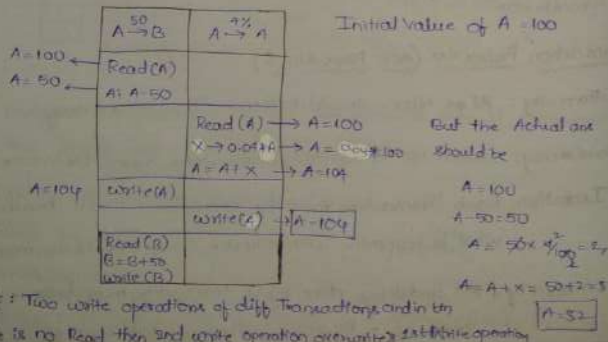
2) If Transaction Consists of multiple steps. Some Involve I/O Activities and other involve cpu activities. In a Computer System cpu and I/o operations can be done in parallel. Therefore I/o activities can be done in parallel with processing at the cpu.

3) The processor and Disk utilisation Increases

Schedule:

It represents the order in which instructions of a transactions are executed.

Loss update problem: (w-w Conflict)



Initial value of A = 100

But the Actual one should be

A = 100

A = 50 = 50

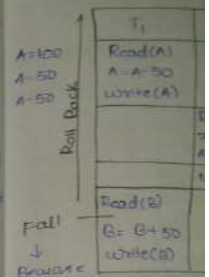
A = 50 × 4/100 = 2

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



fail

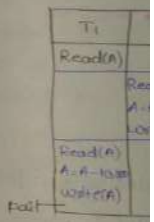
↓

Because of Roll back

⇒ A=100 but the

not considered

Un Repeatable Read



fail

Phantom Tuple

T1

eno	ename	sal
1	A	5
3	C	4

Select * from Employ

salary > 3000

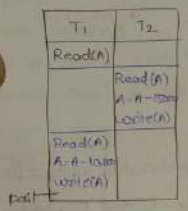
eno	ename	sal	select
1	A	5000	from Em
3	C	4000	where
4	D	3500	sal >

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

T_1	T_2
Read(A)	
A = A - 50	
write(A)	
	Read(A) — A = 50
	x = 0.01 * A — x = 1
	A = A + x — A = 52
	write(A) — A = 52 (Temporarily saved, this transaction is not comited)
Read(B)	
B = B + 50	
write(B)	

$\Rightarrow$ 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 - 1000
	write(A)
Read(A)	
A = A - 1000	
write(A)	

$\Rightarrow$ when a transaction tries to Read the value of data item 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, D, 3500);
eno ename sal 1 A 5000 3 C 4000 4 D 3500	select * from Emp where sal > 3000

Incorrect summary problem

T_1	T_2	
	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)		

Before T_1 / After T_1

K = 50	50
X = 100	600
Y = 200	400
850	1050
Wrong	Actual one

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

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
a_1	b_1
a_2	b_2

S_1	S_2	S_3	S_4	S_5	S_6
a_1	b_1	a_1	b_1	a_1	b_1
a_2	b_2	b_1	a_1	b_1	a_1
b_1	a_1	b_2	a_2	a_2	b_2
b_2	a_2	a_2	b_2	b_2	a_2

⇒ 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 (and consistent) so we need to find the correct schedule that guarantee the consistency

of database.

4 TYPES OF

Serial schedul

Transactions T_1

$T_1 \rightarrow T_2$

T_1	T_2
R(A) A=A+50 W(A)	
	R(A) A=A+50 W(A)

⇒ serial schedule operations of

⇒ when Transac state

⇒ If there are

Complete sched

⇒ At the end present then

T_1	T_2
R(A)	
	R(A)
A=A-50 W(A) Commit	
	A=A+50 W(A) Abort

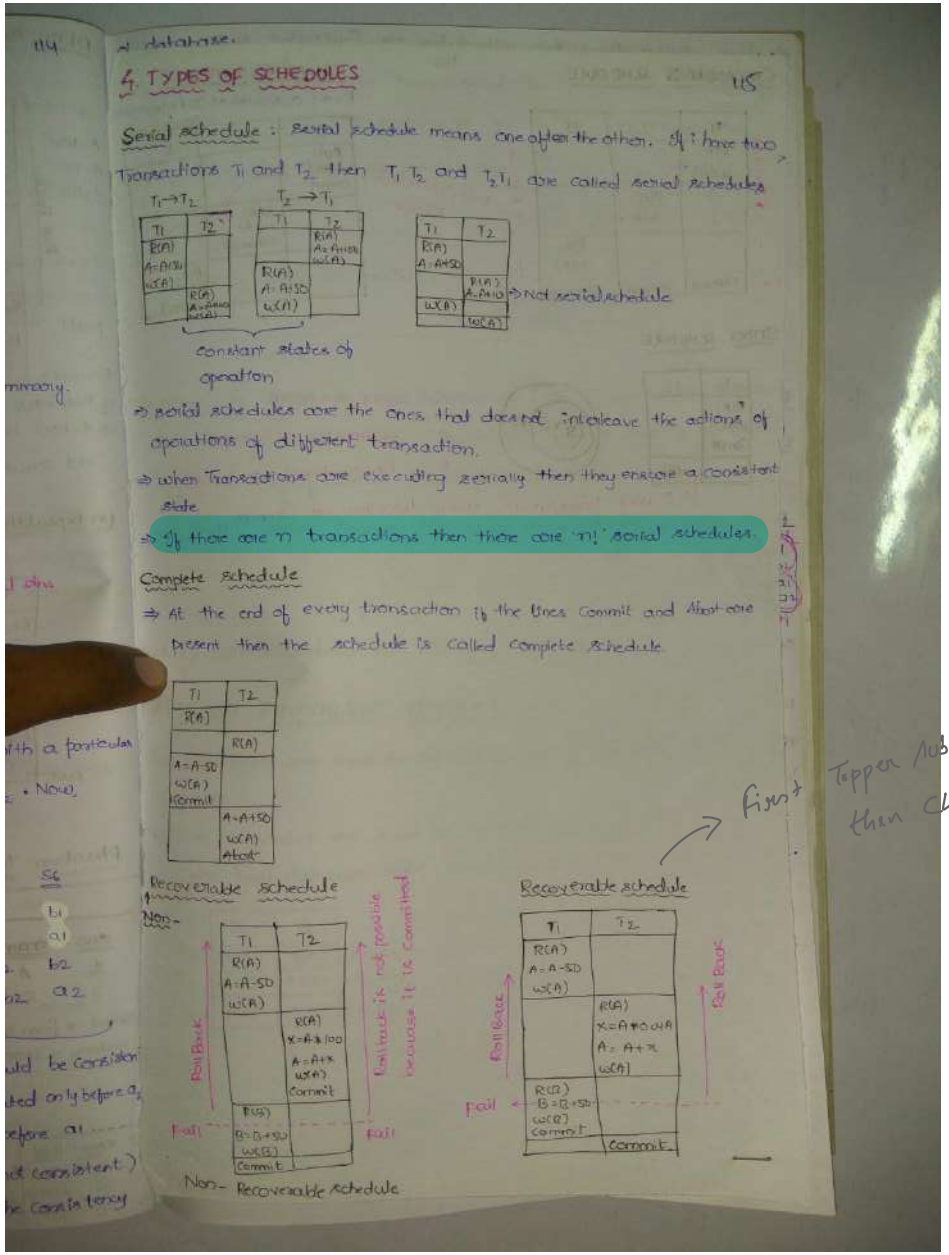
Recoverable

Non-

T_1
R(A)
A=A-50
W(A)
R(B)
B=B+50
W(B)
Commit

Roll Back

Fail



database.

4. TYPES OF SCHEDULES

115

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

$T_1 \rightarrow T_2$ | $T_2 \rightarrow T_1$

T_1	T_2
R(A)	
A=A-SD	
W(A)	
	R(A)
	A=A+SD
	W(A)

constant states of operation

T_1	T_2
R(A)	
A=A+SD	
	R(A)
	A=A+SD
W(A)	
	W(A)

⇒ Not serial schedule

⇒ serial schedules are the ones that doesn't interleave the actions of operations of different transaction.

⇒ when Transactions are executing serially then they ensure a consistent state

⇒ If there are n transactions then there are 'n!' serial schedules.

Complete schedule

⇒ At the end of every transaction if the lines commit and Abort are present then the schedule is called complete schedule

T_1	T_2
R(A)	
	R(A)
A=A-SD	
W(A)	
Commit	
	A=A+SD
	W(A)
	Abort

Recoverable schedule

Non-

T_1	T_2
R(A)	
A=A-SD	
W(A)	
	R(A)
	X=A+100
	A=A+X
	W(A)
	Commit
R(A)	
B=B+SD	
W(B)	
Commit	

Roll Back — fail — Undo/rollback not possible because it is committed

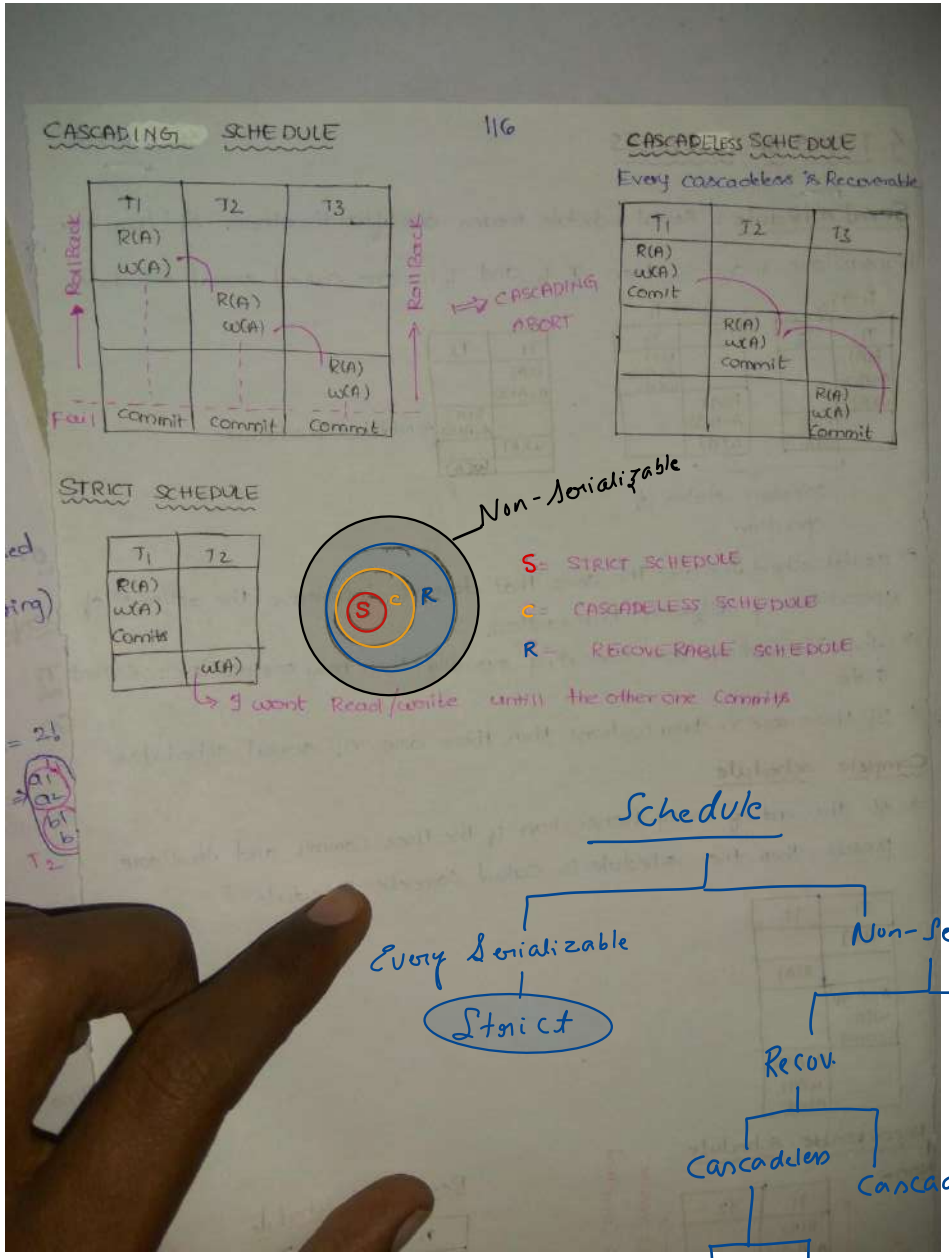
Non- Recoverable schedule

Recoverable schedule

T_1	T_2
R(A)	
A=A-SD	
W(A)	
	R(A)
	X=A+10 or A
	A=A+X
	W(A)
R(B)	
B=B+SD	
W(B)	
Commit	
	Commit

Roll Back — fail — Roll Back

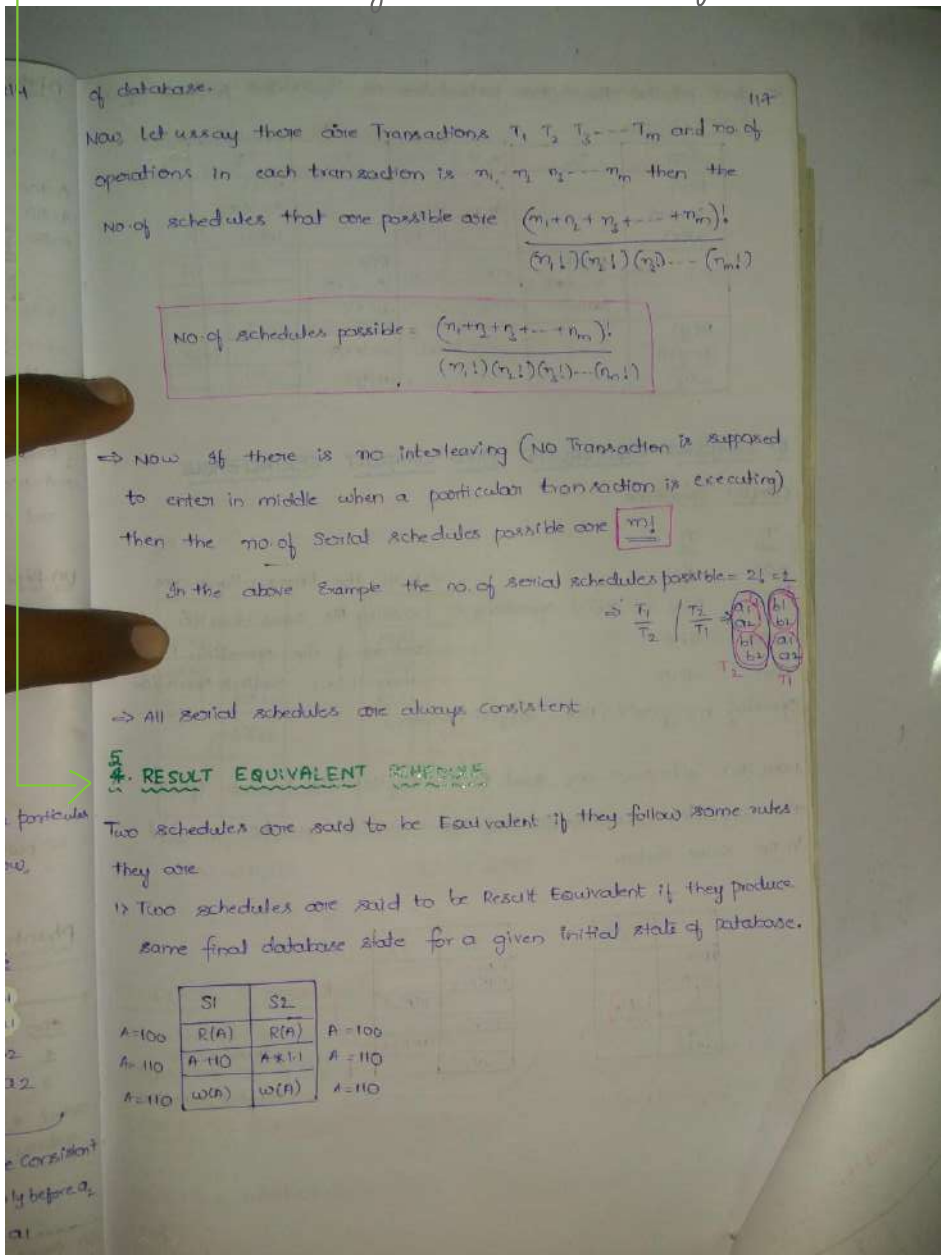
First Topper submit, then Cheater will submit the paper.


CASCADING SCHEDULE
116
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)
Comit
W(A)
I want Read/write until the other one Commits
Non-Serializable
S
C
R
S = STRICT SCHEDULE
C = CASCADELESS SCHEDULE
R = RECOVERABLE SCHEDULE

Schedule
Every Serializable
Non-Serializable
Strict
Recov.
Non-Recov.
Cascadeless
Cascading
Strict
Non-Strict

Serializable Schedule :

A Transaction Schedule is Serializable if its outcome is equal to the outcome of its transaction executed serially. i.e. Sequentially without overlapping in time.



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 $\frac{(n_1+n_2+n_3+\cdots+n_m)!}{(n_1!)(n_2!)(n_3!)\cdots(n_m!)}$

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 $m!$

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

$\Rightarrow \frac{T_1}{T_2} \,/\, \frac{T_2}{T_1}$ → T_2: $a_1\ a_2\ b_1\ b_2$; T_1: $b_1\ b_2\ a_1\ a_2$

⇒ All Serial schedules are always consistent

4. RESULT EQUIVALENT SCHEDULE

Two schedules are said to be Equivalent if they follow same rules they are

1) Two schedules are said to be Result Equivalent if they produce same final database state for a given initial state of database.

	S1	S2	
A=100	R(A)	R(A)	A=100
A=110	A=110	A*1.1	A=110
A=110	W(A)	W(A)	A=110

⇒ check whether these two schedules are Equivalent / not

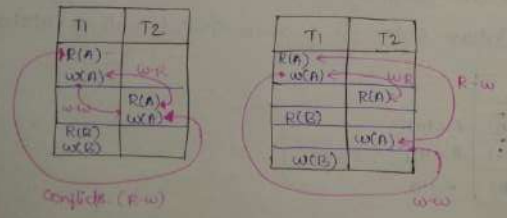
6. CONFLICT EQUIVALENT AND CONFLICT SERIALIZABLE

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 operations 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.



∴ These 2 are Conflict equivalent

CONFLICT
A schedule
Equivalent

7. CONFLICT

Now, check
not.

118

1) Test which of the following are conflict serializable

S1: $R_1(A)$ $R_2(B)$ $W_1(A)$ $W_2(B)$

S2: $R_2(B)$ $R_1(A)$ $W_2(B)$ $W_1(A)$

119

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)	

⇒ S1, S2 are NOT conflicting

No conflicting actions

2)

S1: $R_1(A)$ $W_1(A)$ $R_2(B)$ $W_2(B)$ $R_3(B)$

S2: $R_2(B)$ $W_1(A)$ $R_1(B)$ $R_2(B)$ $W_2(B)$

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

Conflicting operations

$W_2(B) \rightarrow R_1(B)$

WR

≠

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

S1: $T_1 T_2$ (not serializable)

S2: $T_1 T_2$ / $T_2 T_1$ (Conflict Equivalent)

S2 ⇒ Conflict 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)	

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

} Not Conflict Equivalent

∴ The Relation S1 is not Equalent to any of the seriali sable schedule

∴ S_1 is not conflict serializable.

PROCEDURE

① Construct a transaction operation.

② If the Dire is not confli

③ If the grap conflict ser

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)
2. If the Directed Graph contains cycles then the concurrent schedule is not conflict serializable
3. If the graph contains no cycles then the schedule is called conflict serializable

Ex

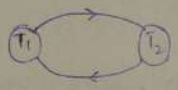
T_1	T_2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(B)	
	R(B)
	w(B)

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

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

Ex:

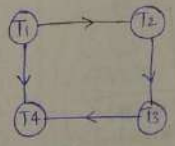
T_1	T_2
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(B)
w(B)	
	w(B)



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

Ex:

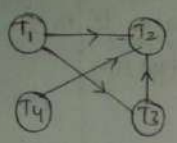
T_1	T_2	T_3	T_4
	R(x)		
		w(x)	
w(y)			
	R(y)		
	w(z)		
			R(x)
			R(y)



There is no cycle and therefore the schedule is conflict-serializable

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

Ex



(a) Not conflict serializable

(b) $T_3\ T_4\ T_1\ T_2$ ⇒ No cycles in Graph ⇒ Conflict serializable

(c) $T_1\ T_4\ T_3\ T_2$

(d) $T_2\ T_3\ T_1\ T_4$

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 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
$R_2(B)$ $w_2(B)$	

but $R_1 w_1$ and $R_2 w_2$ should be executed as group

∴ No. of conflict serializable schedules

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

Now $(T_2 \rightarrow T_1$

$R_1(A)$

$w_1(A)$

9. VIEW E

Two sched

following 3

1) For each

in sche

read th

⇒ If Trans

produced

S' also

3) For each

of 'A' o

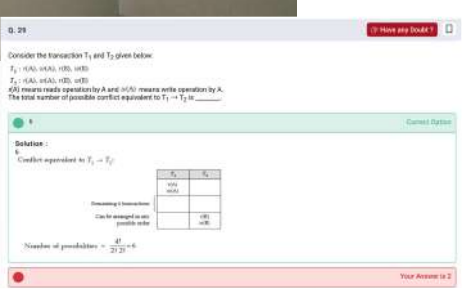
in S'.

⇒ In summar

same order

Ex:

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



⇒ conflict
serializable

not have

$(T_1 \to T_2)$ or
lled conflict

ne first. 9
(A) 1st because
$T_2 \to T_1$ (but my

uted as group

lizable schedules

Now $(T_2 \to T_1)$

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

$\Rightarrow \frac{4!}{2!2!}$ 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 inital 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:

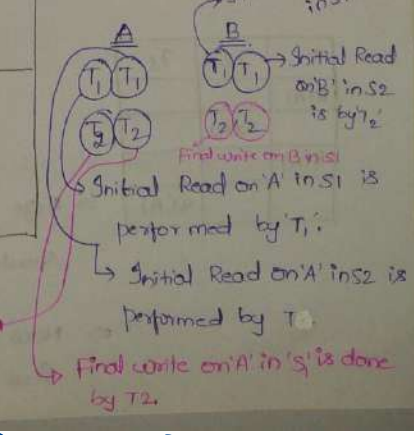
S1

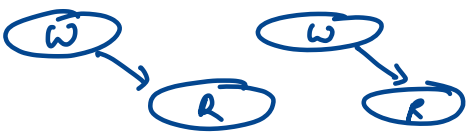
T1	T2
R(A) w(A)	
	R(A) w(A)
R(B) w(B)	
	R(B) w(B)

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





	S_1	S_2
A	T_1, T_2	T_1, T_2
B	T_1, T_2	T_1, T_2

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 → Consuming

In S_2

	S2	
	T1	T2
	R(A)	
producer →	W(A)	
	R(B)	
	W(B)	
		R(A)
		W(A)
		R(B)
		W(B)

→ consume

∴ These two schedules is view Equivalent

Ex:

S_1

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

Final write here is by T_1

S_2

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

Final write here is by T_2

S_3

T1	T2
produce →	W(A)
R(A) W(A)	

$\therefore T_1 \neq T_2$ (Not View Equalent)

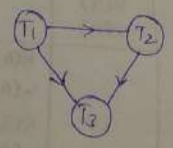
$\therefore$ $S_1 \neq S_2$ $S_2 \neq S_3$ (final write of 'A's are diff)

$S_1 \neq S_3$ (No producer consumer in S1)

10. VIEW SERIALISABLE EXAMPLES

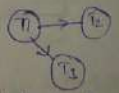
The method that we use to check view serializability is by using "poly graphs"

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 coming to T_2 and T_3 T_2 should start first so

Ex

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

Ex

T1	
R(A)	
W(A)	

⇒ If

but a

Blind write:

⇒ If a sch

then there sh

11. SERIALIZ

⇒ A schedule

or view se

⇒ Now, consi

Ex

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

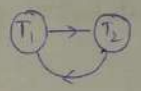
∴ The Graph contains No cycle and so the schedule is view serializable and conflict serializable

Ex

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



∴ ⇒ Cycle ⇒ Not V.S.

⇒ If a schedule is Conflict serializable then it is view serializable but a view serializable schedule need not be Conflict serializable

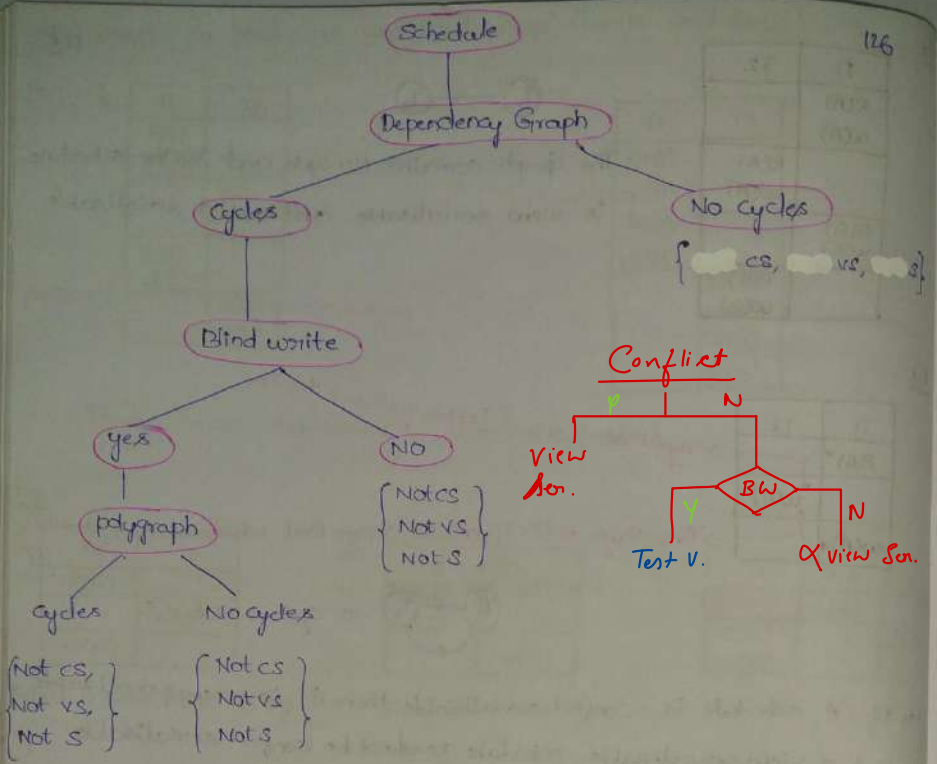
Blind write: Write without Read is called Blind write

⇒ If a schedule should be view serializable but not conflict Serializable then there should be atleast one blind operation.

11. 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 = $\frac{(2+2)!}{2!\,2!}$ $\frac{(n_1+n_2+\dots+n_m)!}{n_1!\,n_2!\dots n_m!}$

$= 6$

Non serial schedules = Total schedules − No. of serial schedules

$= 6 - (2!)$ (No. of Transactions)

$= 4$

⇒ The no. of concurrent schedules that can be formed over two transactions having n_1 and n_2 operations Respectively are

$= \left[\frac{(n_1+n_2)!}{n_1!\,n_2!}\right]$

The no of

⇒ The no. of

having n_1 n_2

⇒ The no. of

Determine

or not?

S_1: $R_1(A)$ W_1(

S_2: $R_1(A)$ R_1(

S_3: $R_1(A)$ R_2(

S_4: $W_3(A)$ R_2

S_1

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

S_3

T1	T2
R(A)	
	R(A)
R1(B)	
	R(B)
W(A)	
	W(B)

The no of serial schedules possible are $\left[\frac{(n_1+n_2)!}{n_1!\ n_2!}\right]-2$

⇒ 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 $\left[\frac{(n_1+n_2+\cdots+n_m)!}{n_1!\ n_2!\ n_3!\cdots n_m!}\right]-(m!)$

Now

Determine whether the following schedules are conflict serializable or not?

S_1: $R_1(A)\ w_1(A)\ R_2(B)\ w_2(B)\ R_1(B)\ w_1(B)$

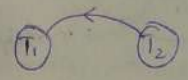
S_2: $R_1(A)\ R_1(B)\ w_2(A)\ R_3(A)\ w_1(B)\ w_3(A)\ R_2(B)\ w_2(B)$

S_3: $R_1(A)\ R_2(A)\ R_1(B)\ R_2(B)\ R_3(B)\ w_1(A)\ w_2(B)$

S_4: $w_3(A)\ R_2(A)\ w_1(B)\ R_2(B)\ w_2(C)\ R_3(C)$

S_1

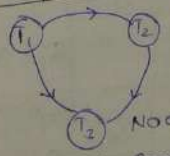
T1	T2
R(A)	
w(A)	
	R(B)
	w(B)
R(B)	
w(B)	



∴ conflict serializable

S_2

T1	T2	T3
R(A)		
R(B)		
	w(A)	
		R(A)
w(B)		
		w(A)
	R(B)	
	$w_2(B)$	

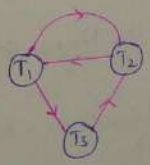


No cycles

∴ conflict serializable

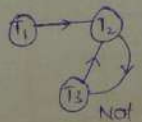
S_3

T1	T2	T3
R(A)		
	R(A)	
$R_1(B)$		
	R(B)	
		$R_3(B)$
w(A)		
	w(B)	



Not conflict serializable

S_4



Not conflict serializable

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?

$\Rightarrow$ No. of serial schedules $= m! = 2! = 2$ [m = No. of Transactions] $(T_1 \to T_2)\ (T_2 \to T_1)$

$\Rightarrow$

Consider the following schedules

S_1: $R_2(x)\ R_2(y)\ R_1(x)\ R_1(y)\ W_1(x)\ R_2(x)$ ✓

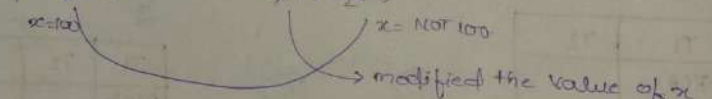
S_2: $R_2(x)\ R_2(y)\ W_2(x)\ R_1(x)\ R_1(y)$

S_3: $R_2(x)\ R_3(x)\ R_3(x)\ R_2(y)\ W_2(x)\ W_2(x)$

Which of the above schedules are having un-repeatable Read problem?

$\Rightarrow$ Un Repeatable Read problem means if a transaction tries to read the same data two times and in b/t this two reads if some other transaction changes the value then it is called un-Repeatable Read

$\Rightarrow$ Now S_1: $R_2(x)$ ----- $W_1(x)\ R_2(x)$



Which of the above schedules is having lost update problem.

$\Rightarrow$ Lost update problem means one transaction will write a value and immediately another transaction writes a value without Reading it (one writes ov

S_1: only one write is there so no lost update

S_2: " " " " " " " "

S_3: $W_1(x)\ W_2(x)$

one write (W_2) overwrites the other write $\therefore$ lost update problem

which of the

$\Rightarrow$ Dirty Read

someone el

S_2: $R_2(x)$ ·

S_1: $W_1(x)$

which of the

strict

S_1: $R_1(x)\ R_2(x)$

T
R(
W(
com

S_2: $R_1(x)\ W_2($

T
R
W
R
Co

S_3: $R_3(x)\ R_1($

Recoverable:

Cascadeless

which of the above schedules is having dirty Read problem. 129

⇒ Dirty Read problem is if someone will write it and before writing someone else will read it this is called Dirty Read problem.

S_1: $R_2(x) \quad \cdots \quad w_2(x)(R_1(x))$

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)

T_1	T_2
R(x)	
	R(x)
w(x)	
	w(x)
	commit
commit	

⇒ Recoverable : consumer should not commit before the producer commit

⇒ There are no producer and consumer . The schedule is Recoverable, cascadeless

⇒ Strict says if some one writes a value you should not Read/write it until the previous one commits :. Not strict schedule

S_2: $R_1(x)\ w_2(x)\ w_1(x)\ R_1(x)$ commit(2) commit 1

T_1	T_2
R(x)	
	w(x)
w(x)	
R(x)	
	commit
commit	

⇒ No producer consumer ⇒ cascadeless, Recoverable

⇒ Not strict schedule

S_3: $R_3(x)\ R_1(x)\ R_2(x)\ w_1(y)\ R_2(y)\ w_4(\ )\ C_3 C_1 C_2$

T_1	T_2	T_3
		$R_3(x)$
$R_1(x)$		
	$R_2(x)$	
$w_1(y)$	Producer - Consumer	
	$R_2(y)$	
	$R_2(x)$	
		Commit
commit		
	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?

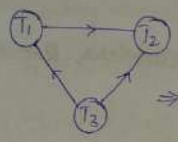
i) $r_1(x), r_3(x), w_1(x), r_2(x), w_2(x)$

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) $r_3(x), r_2(x), r_1(x), w_3(x), w_1(x)$

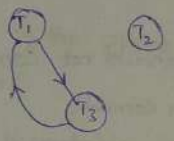
Now, i)



∴ The schedule is conflict serializable

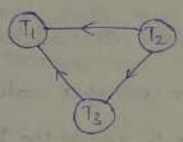
⇒ Start with the node that doesn't have any incoming edge = $T_3\ T_1\ T_2$

ii)



⇒ Not conflict serializable

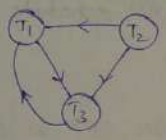
iii)



⇒ ∴ 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

T_1	T_2
R(x)	
	R(x)
	R(y)
	w(x)
	w(y)
R(x)	
R(y)	
w(x)	

WW

T_1	T_2
R(x)	
R(y)	
w(x)	
	w(x)
	w(y)
	R(x)
	R(y)

15. POLYGR

T_K	T_i	
	⋮	
	w(A)	
		R(A

S : $r_i(A)$ w_j(

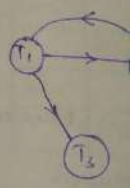
NOTE :⇒ If

is view s

⇒ If

writ

Not



Now, Insert two

trans actions T_k

T_b	T_1	T_2	T_3	T
w(A)				
	R(A)			
		w(A)		
	w(A)			
			w(A)	
				R

each

les?

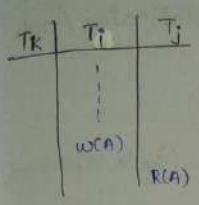
lizable

e any

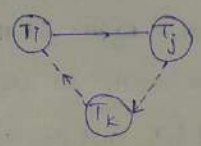
t result in

-conflict

15. POLYGRAPHS FOR VIEW SERIALIZABLE -- EXAMPLE 1



⇒ 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 : $r_1(A)$ $w_2(A)$ $w_1(A)$ $w_3(A)$

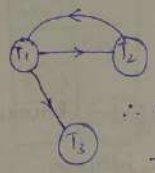
NOTE: ⇒ If a schedule is conflict serializable then the schedule is view serializable, so the 1st check for VS should be CS.

⇒ If the schedule is not CS, now check if there are Blind writes or not, in case if there are Blind writes and it is not CS then only we should check for VS.

∴

CS ✗	CS ✗	CS ✓
Bw ✗	Bw ✓	VS ✓
VS ✗	VS ✓	

CS = Conflict serializable
Bw = Blind writes
VS = View serializables



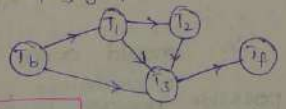
∴ Not CS ✗ ∴ Now check for Blind writes (write without Read)

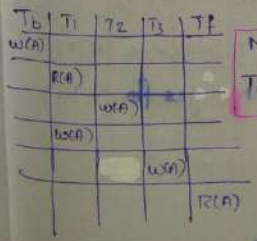
Transaction 2 is writing without Reading (∴ Bw ✓)

Now, Insert two dummy transactions T_b and T_f

CS ✗		
Bw ✓		
VS ✓		

↳ Now draw poly graph





No cycles in poly graph
The schedule is view serializable

~~T_b~~ T_1 T_2 T_3 ~~T_f~~

T_1 T_2 T_3

We Assume

T_b = writes data items initially

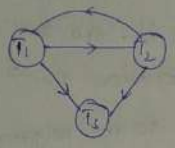
T_f = Read all data items

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

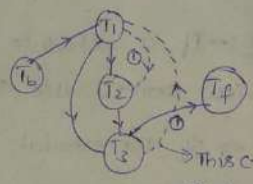
⇒ First check whether it is conflict serialisable / not



⇒ 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 $w_3(A)$ 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)



No cycle is present

∴ The schedule is view serializable and the serial schedule is

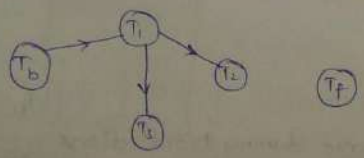
$T_b\ T_1\ T_2\ T_3\ T_f = T_1\ T_2\ T_3$

⇒ Procedure to draw the above poly graph

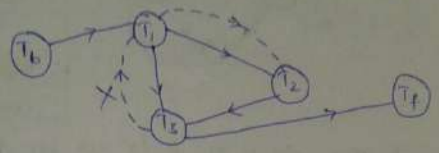
1) Check for write and Read is of the same object between two different transactions. The first WR is between

⇒ $w_b(A)$ $R_1(A)$ 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 $T_3 \to T_f$

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)

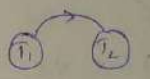
⇒ Not serializable

⇒ 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)

⇒ 

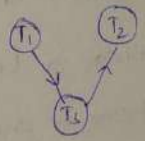
⇒ Conflict serializable, View serializable, Strict

⇒ Recoverable cannot be said (No commit) ⇒

⇒ Cascading schedule

→ Dirty Read.

T1	T2	T3
R(x)		
	R(y)	
		w(x)
	R(x)	



⇒ conflict serializable $(T_1 T_3 T_2)$

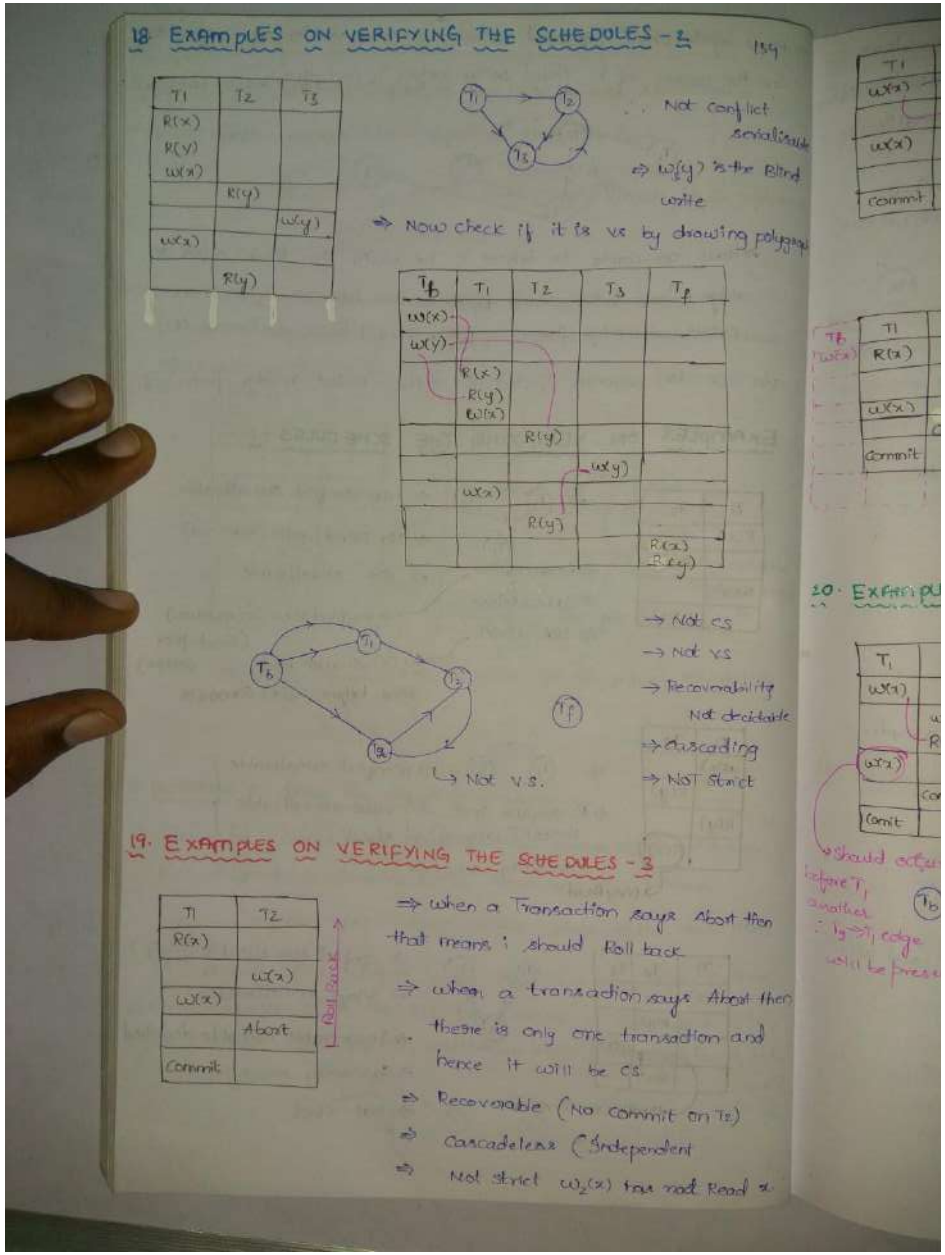
⇒ view serializable

⇒ Recoverable cannot be decided

⇒ Cascading schedule

⇒ Not Strict.

→ producer - consumer



Repeated Mintake

Q. 14

FAQ | Solution Video | Have any Doubt ?

Consider the following schedule S:

$S : w_1(x),\ r_2(x),\ w_3(x),\ r_4(x),\ w_5(x),\ r_6(x),\ w_7(x),\ r_8(x),\ w_9(x),\ r_{10}(x)$

The number of serial schedules which are view equal to schedule to S but not conflict equal to S are _____.

23 — Correct Option

Solution :

23

① $w_1(x), r_2(x)$ ② $w_3(x), r_4(x)$ ③ $w_5(x), r_6(x)$ ④ $w_7(x), r_8(x)$ it can execute in any order.

Total 4! = 24 view equal serial schedule for S and only one conflict equal serial schedule for S.

Total 24 - 1 = 23 which are view equal but not conflict equal.

Your Answer is 0

Read carefully

Q.21

Which of the following are true

A. All view serializable schedules are conflict serializable as well.

B. All conflict serializable schedules are view serializable as well.

C. If a schedule is view serializable then it is serializable

D. All Serial schedules are conflict serializable, view serializable.

Time taken to answer this question 00:00:34 hrs

Hide Answer | Add Notes | View Notes | Show Comments

A A,B,D

B B only — Your answer is Wrong

C D only

D None of the above — Correct Option

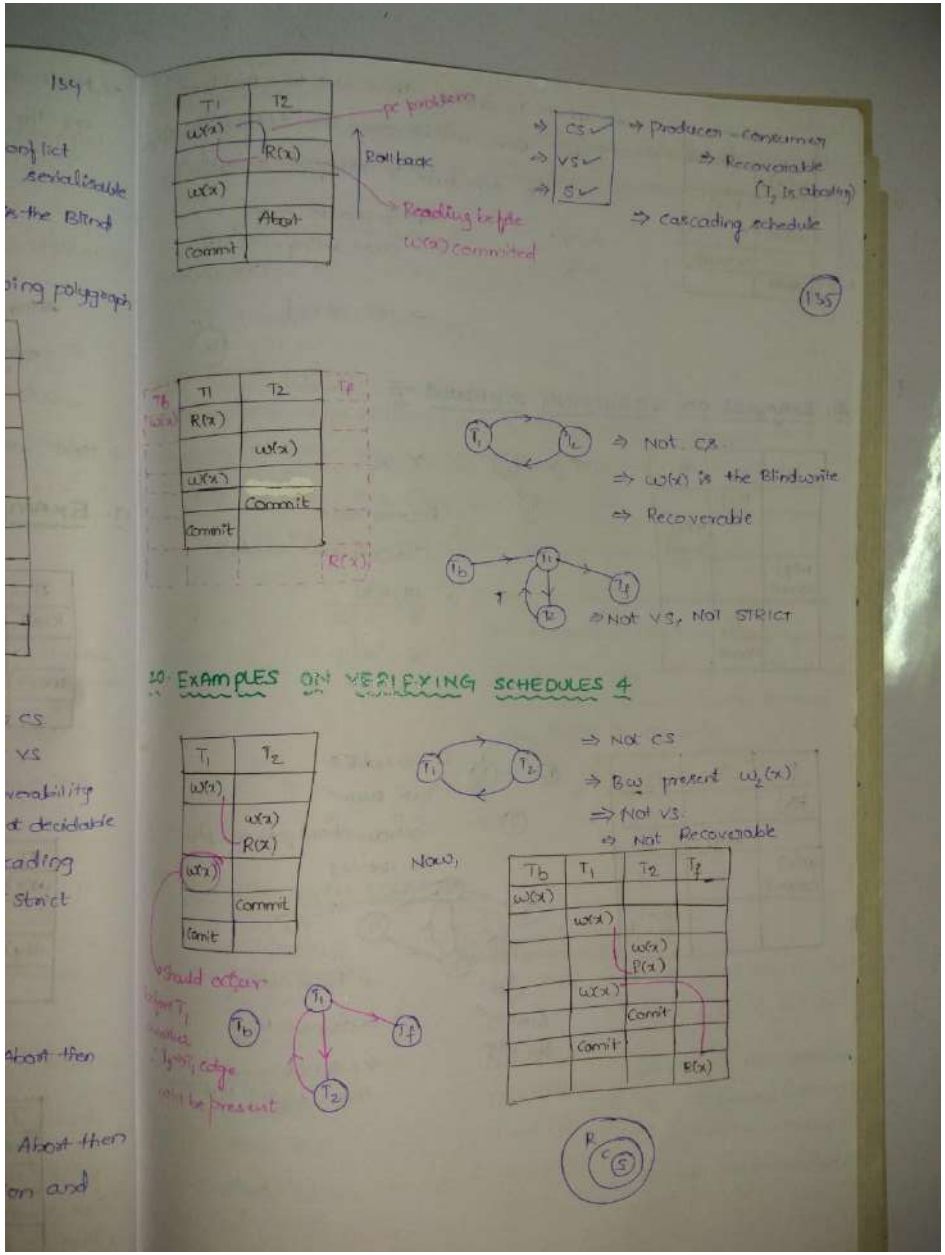
Solution: (D)

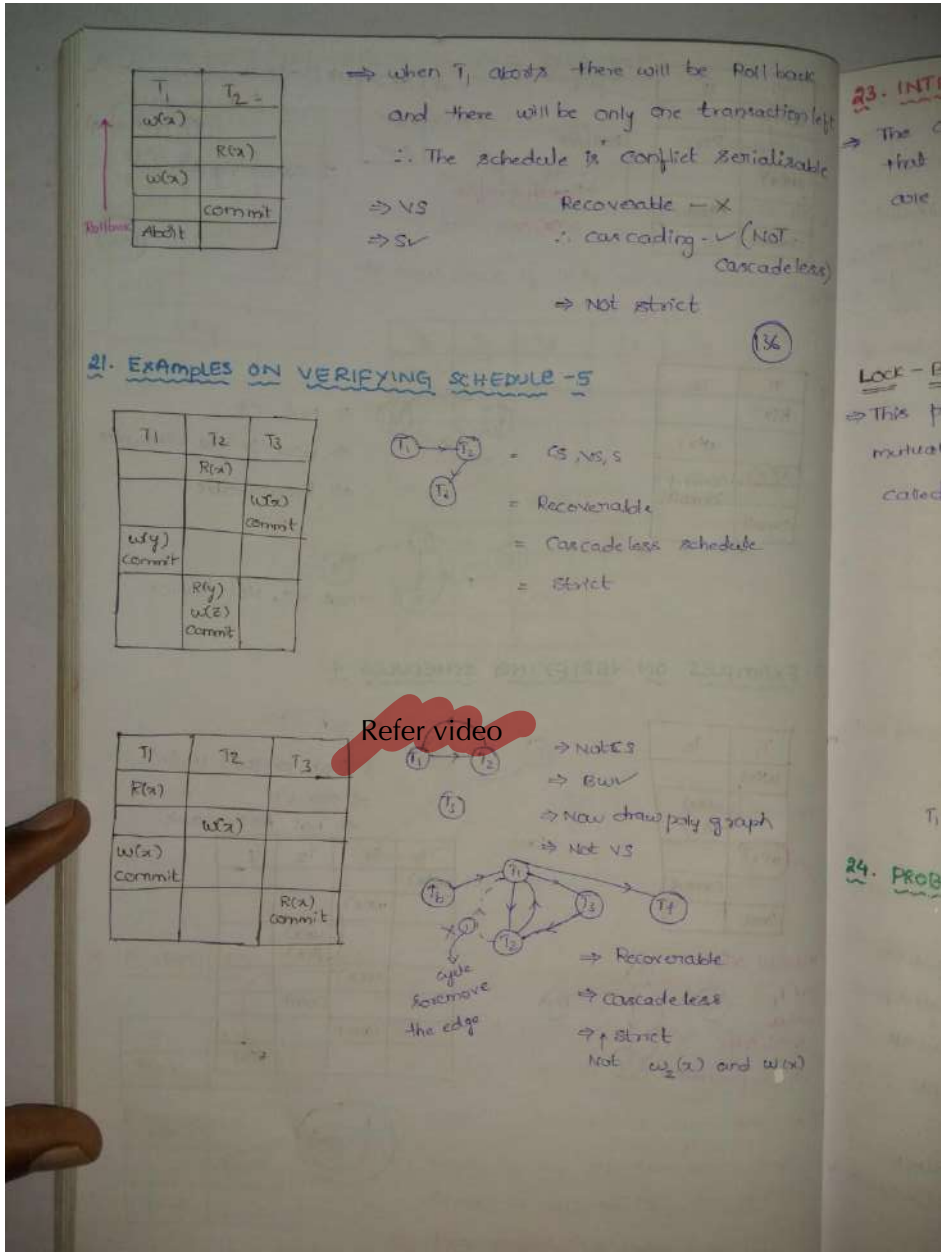
All serializable schedules are conflict and view serializable.

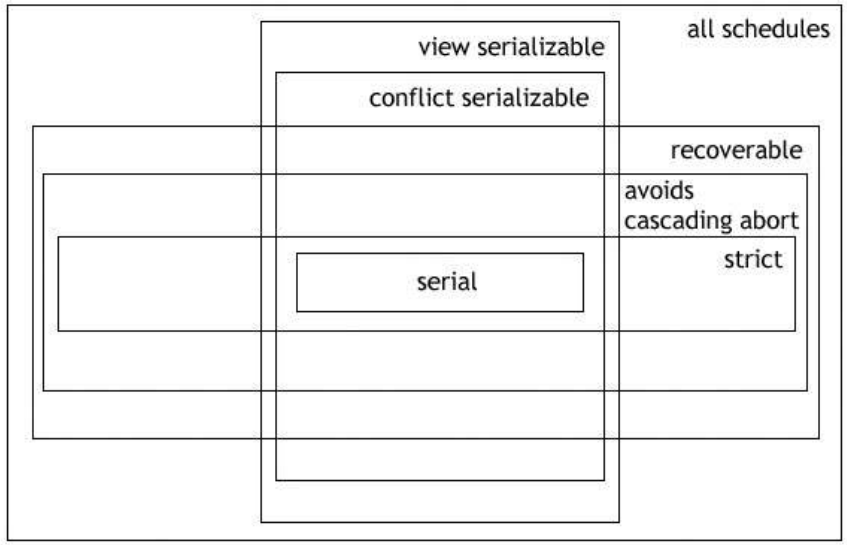
And all conflict serializable schedules are view serializable but the reverse need not be true.

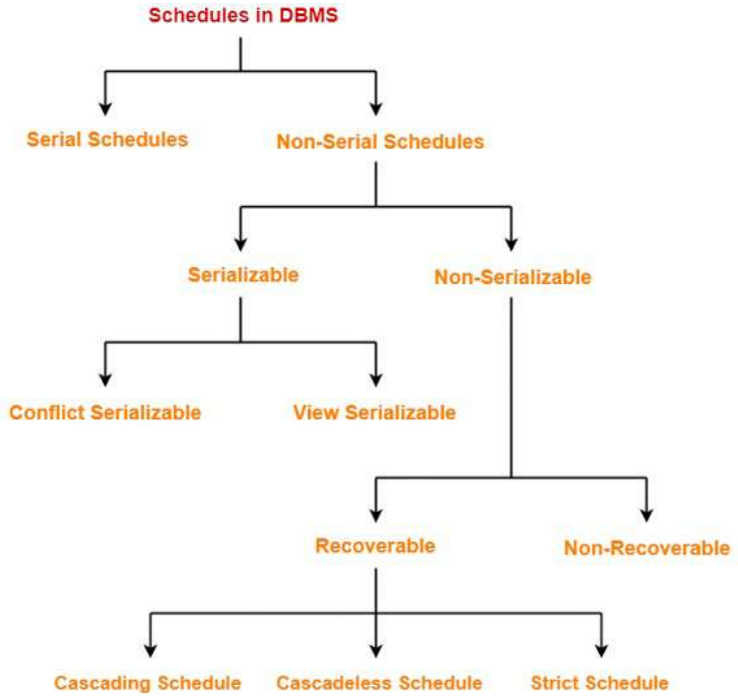
If a schedule is view serializable then it is serializable

B and C and D are true.


T1 T2
W(x)
R(x)
W(x)
Abort
Commit
rc problem
Rollback
Reading before W(x) committed
⇒ CS ✓
⇒ VS ✓
⇒ S ✓
⇒ Producer - Consumer
⇒ Recoverable
(T2 is aborting)
⇒ cascading schedule
135
T1 T2
R(x)
W(x)
W(x)
Commit
Commit
Tb
Tf
R(x)
⇒ Not CS
⇒ W(x) is the Blindwrite
⇒ Recoverable
⇒ Not VS, NOT STRICT
10. EXAMPLES ON VERIFYING SCHEDULES 4
T1 T2
W(x)
W(x)
R(x)
W(x)
Commit
Commit
should occur
⇒ Not CS
⇒ Bw present W2(x)
⇒ Not VS
⇒ Not Recoverable
Now,
Tb T1 T2 Tf
W(x)
W(x)
W(x)
R(x)
W(x)
Commit
Commit
R(x)
R
C
S


⇒ when T1 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
= CS, VS, S
= Recoverable
= Cascadeless schedule
= Strict
Refer video
⇒ Notes
⇒ BWR
⇒ Now draw poly graph
⇒ Not VS
Cycle so remove the edge
⇒ Recoverable
⇒ Cascadeless
⇒ Strict
Not W2(x) and W(x)


all schedules
view serializable
conflict serializable
recoverable
avoids cascading abort
strict
serial


Schedules in DBMS
Serial Schedules
Non-Serial Schedules
Serializable
Non-Serializable
Conflict Serializable
View Serializable
Recoverable
Non-Recoverable
Cascading Schedule
Cascadeless Schedule
Strict Schedule

Concurrency Control Protocols

23. INTRODUCTION TO LOCKING

(137)

⇒ The concurrency manager present in the os make sure that the transactions that are being executed are serializable, by using some rules they are

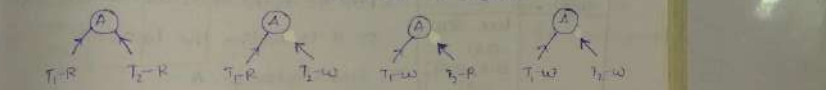
- ⇒ Lock-based protocol
- ⇒ Time stamp based protocols
- ⇒ Graph based protocols.

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	✓	X
L	X	X

⇒ If a transaction has shared lock on particular data item then another transaction may also acquire the shared lock on the same data item so shared 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)

Note if it is only Reading then go for shared lock

⇒ Now if the Transaction 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)	

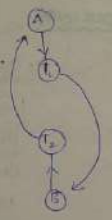
A B
100 200 = (A+B) = 300

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) → Here the transaction has not unlocked 'A' so it is holding the lock on 'A'.	
	Lock-S(B) R(B) Lock-S(A) → T_2 Requesting lock on A R(A) Display(B+A) commit unlock(B) unlock(A)
← Lock-X(B) (T_1 req lock on B) R(B) B = B + 50 W(B) commit unlock(B) unlock(A)	



∴ There is cycle, one transaction waits for the other to unlock

∴ DEADLOCK

∴ Simple locking might produce Deadlock

Simple locking might not Guarantee Serializable

25. Two PHASE

Since the

2-phase lock

Growing phase,

⇒ Growing phas

⇒ Shrinking p

⇒ The point w

⇒ with the hel

T_1	
	L
	L U
	U
L-S(B)	
L-X(A) ← locking point for T_1 U(A) U(B)	

⇒ The 2-phas

obtained

26. STRICT

Some modifica

cascading Ro

Strict 2PL

Which Requir

mode locks

commits. (U

L-X

WCA

Com

U(X

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 [cascading Rollbacks and Deadlocks] sometimes 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 inaddition to locking being 2pL all exclusive mode locks taken by a transaction must be held until the transaction commits. (unlock only after a particular transaction commits).

Lx-(A)
W(A)
commit
U(x) → unlock only after transaction commits.

Strict 2pL are cascadeless, Recoverable, Deadlock possible

Rigorous 2PL:
which requires that in addition to locking being 2-phase all locks must be held until the transaction commits.

⇒ Can Avoid cascading Roll backs.
⇒ Deadlock cannot be avoided (Deadlocks are possible because of hold and wait)

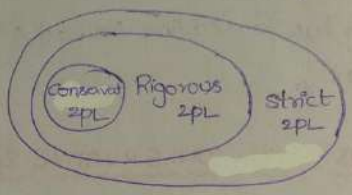
Conservative 2PL
⇒ which Requires the transaction to update all the locks before it starts and release all the locks after it commits.
⇒ Avoids cascading Rollbacks and Deadlocks.

27. EXAMPLES ON 2PL

Which of the following schedules (Transactions) are in Strict 2PL?

a) Lock-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
∴ Strict 2PL ✓
⇒ Rigorous says that either it is shared lock or exclusive lock they should be unlocked only after commit, here shared lock is released before commit
∴ NOT RIGOROUS 2PL

c)
Lock-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

Q. 5

Solution Video | Have any Doubt ?

Which of the following is true?

A. Wound wait and wait die algorithms are pessimistic deadlock prevention algorithm and can cause more transaction aborts than needed. — Your answer is Correct

Solution :
(a)
- Option (a) is correct.
- 2PL guarantees schedule will be conflict schedule will be conflict serializable but reverse is false.
- Strict 2PL schedule guarantee to prevent cascading aborts but not basic 2PL. So it is correct.

B. Schedules that are conflict serializable have to be produced by two phase locking.

C. Schedules produced by two phase locking are guaranteed to prevent cascading aborts.

D. None of these

140

all locks

ecause of hold

efore H

ict 2pL ?

phase

lock - X(B)

B' commits

ted :. NOT

STRICT 2PL

pL ✓

-(B) it

B' commits

it is shared

hould be

shared lock

c)
lock-S(A)
R(A)
lock-X(B)
R(B)
w(B)
commit
unlock(A)
unlock(B)

⇒ Growth and shrink ⇒ 2pL

141

⇒ unlock (B) is done after commit ∴ Strict 2pL

⇒ unlock (A) is done after commit ∴ 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

∴ 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)
NOT 2pL

lock-X(B)
R(B)
w(B)
unlock(B)
NOT 2pL

unlock(A)
commit

⇒ 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

i) In tree protocol the only lock instruction allowed is lock-X on 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.

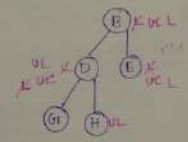
ii) Subsequently a data item can be locked by T_i only if the parent of the data item is currently locked by T_i

iii) Data items may be unlocked at any time

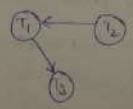
iv) A Data item that has been locked and unlocked by T_i cannot subsequently relocked by T_i

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)		

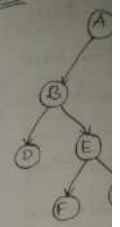
⇒ Now consider the Transactions one-by-one T_1 and then T_2 and then



{Now if you want to see serializability order, replace all the locks with write statements}



Ex



∴ The give
schedul

Advantages

⇒ Early un
⇒ Ensures
⇒ Deadlock

Drawbacks

⇒ you shou
⇒ Unneces

29. TIME S

⇒ Time stam
Concurrency
⇒ It Requ
in advan
⇒ Each Tran
⇒ If any
the time

Wait-Die scheme

It is a **non-preemptive** technique for deadlock prevention. When transaction T_n requests a data item currently held by T_k, T_n is allowed to wait only if it has a timestamp smaller than that of T_k (That is T_n is *older* than T_k), otherwise T_n is killed ("die").

In this scheme, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur:

1. **Timestamp(T_n) < Timestamp(T_k)** – that is T_n, which is requesting a conflicting lock, is *older* than T_k – then T_n is allowed to "wait" until the data-item is available.
2. **Timestamp(T_n) > Timestamp(T_k)** – that is T_n is *younger* than T_k – then T_n is killed ("dies"). *T_n is restarted later with a random delay but with the same timestamp(n).*

This scheme allows the older transaction to "wait" but kills the younger one ("die").

Example

Suppose that transaction T_5, T_{10}, T_{15} have time-stamps 5, 10 and 15 respectively.

If T_5 requests a data item held by T_{10} then T_5 will "wait".

If T_{15} requests a data item held by T_{10}, then T_{15} will be killed ("die").

Wound-Wait scheme

It is a **preemptive** technique for deadlock prevention. It is a counterpart to the wait-die scheme. When Transaction T_n requests a data item currently held by T_k, T_n is allowed to wait only if it has a timestamp larger than that of T_k, otherwise T_k is killed (i.e. T_k is wounded by T_n).

In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur:

1. **Timestamp(T_n) < Timestamp(T_k)**, then T_n forces T_k to be killed – that is T_n "wounds" T_k. *T_k is restarted later with a random delay but with the same timestamp(k).*
2. **Timestamp(T_n) > Timestamp(T_k)**, then T_n is forced to "wait" until the resource is available.

This scheme allows the younger transaction requesting a lock to "wait" if the older transaction already holds a lock, but forces the younger one to be suspended ("wound") if the older transaction requests a lock on an item already held by the younger one.

Example

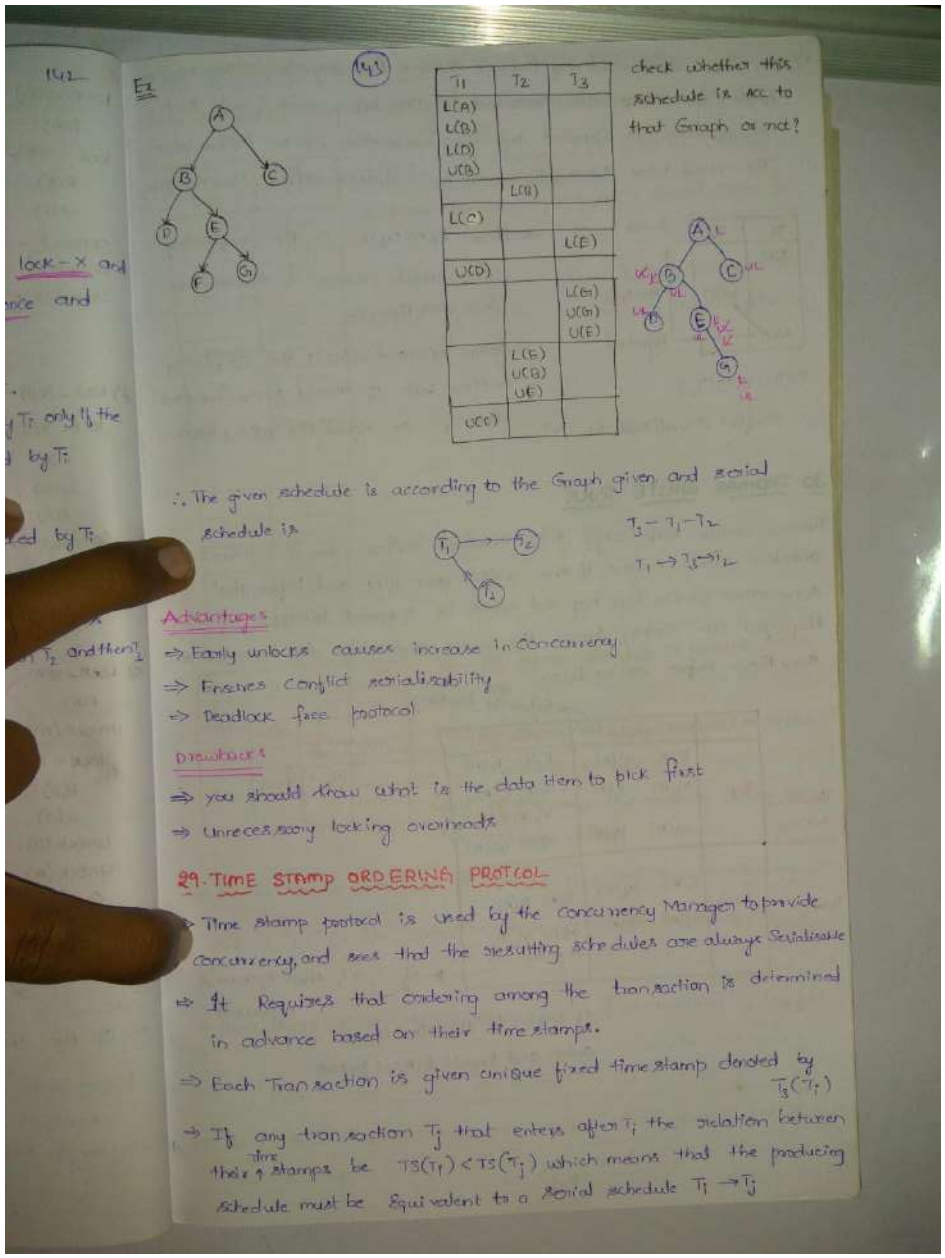
Again, suppose that Transactions T_5, T_{10}, T_{15} have time-stamps 5, 10 and 15 respectively.

If T_5 requests a data item held by T_{10}, then data item will be preempted from T_{10} and T_{10} will be suspended. ("wounded")

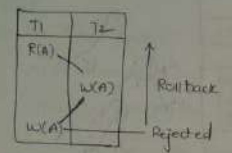
If T_{15} requests a data item held by T_{10}, then T_{15} will "wait".

Summary

In both the cases, only the transaction that enters the system at a *later* timestamp (i.e. the younger transaction) might be killed and restarted.



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



$TS(T_1) < TS(T_2)$

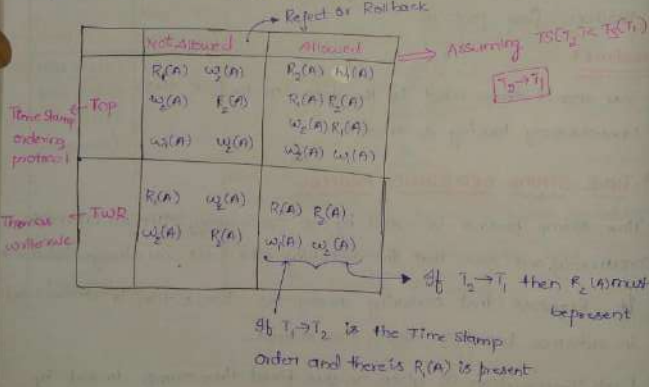
∴ Conflict serializable to $T_1 \rightarrow T_2$

⇒ Here $R_1(A)\ W_2(A)$ is the conflicting action which means T_1 should occur first and then T_2

⇒ 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

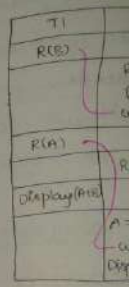
Thomas write Rule says that obsolete Writes can be ignored obsolete write means if the writes are late and before that some other write has happend which is supposed to happen after it then you can clearly ignore the write and make the transaction stay there before rolling back.



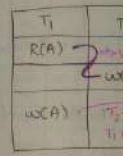
31. EXAMPLES

check which c

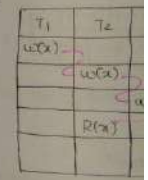
1)



2)



3)



⇒ In the abo

$T_2\ T_3\ T_4$

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

⇒ Here $W_3(x)$

∴ This is

⇒ Now, Thom

here the co

31. EXAMPLES ON TIME STAMP ORDERING PROTOCOL

check which of the following schedules can appear under Top.

1)

T_1	T_2
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)	
	W(A)
W(A)	

⇒ valid

⇒ $TS(T_1) < TS(T_2)$

⇒ ∴ 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 '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

Chapter 5: Transaction and concurrency GATE Questions

QUESTION

1. ACID Property Questions(Theory)
2. Concurrency Control Protocols
3. Serialisability(Conflict/View)
4. No. Of serialisable Conflicts possible
5. Recoverable and Irrecoverable Transactions
6. Checkpointing protocol questions🧐

ANSWER

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

i) ordered file organisation

All files of Records are ordered based on some search key value

Searching : Binary search, B-datablocks to access a record

$$\text{Avg no of block acess} = \lceil \log_2(B) \rceil \text{ blocks}$$

Adv : Searching is efficient

DisAdv : Insertion is expensive, because we have to shift all the data to make space for new data.

Unordered file

All the file

usually at

Searching:

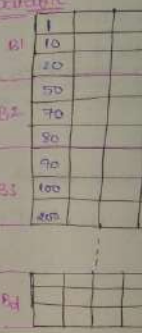
Avg. no of

Adv: Insertio

DisAdv: Sear

2. INDEXING

Datafile



⇒ Avg No of Block

⇒ For every

Index file th

⇒ For every

Index file

3. STRUCTURE

Index:

⇒ Indexes are us

⇒ Index is a sea

⇒ Index is an

Searching:

Unordered file organisation

All the file records are inserted at where ever the place is available usually at the end of the file

$\left(\frac{1+B}{2}\right)$ Avg.

Searching: Linear search

Avg. no of block access = $B/_2$ blocks

Adv: Insertion is easy

DisAdv: Searching is inefficient.

2. INDEXING

Datafile

B1	1
	10
	20
B2	50
	70
	80
	90
B3	100
	200
Bd	

Index file

1	B1
10	B1
20	B1
50	B2
70	B2
90	B3
100	B3
Bi	

⇒ Avg No of Block Access: $(\log_2 B_d)$ (Binary search)

For Searching in index file

Avg. Block Access = $(\log_2 B_i + 1)$ (+1 for accessing 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

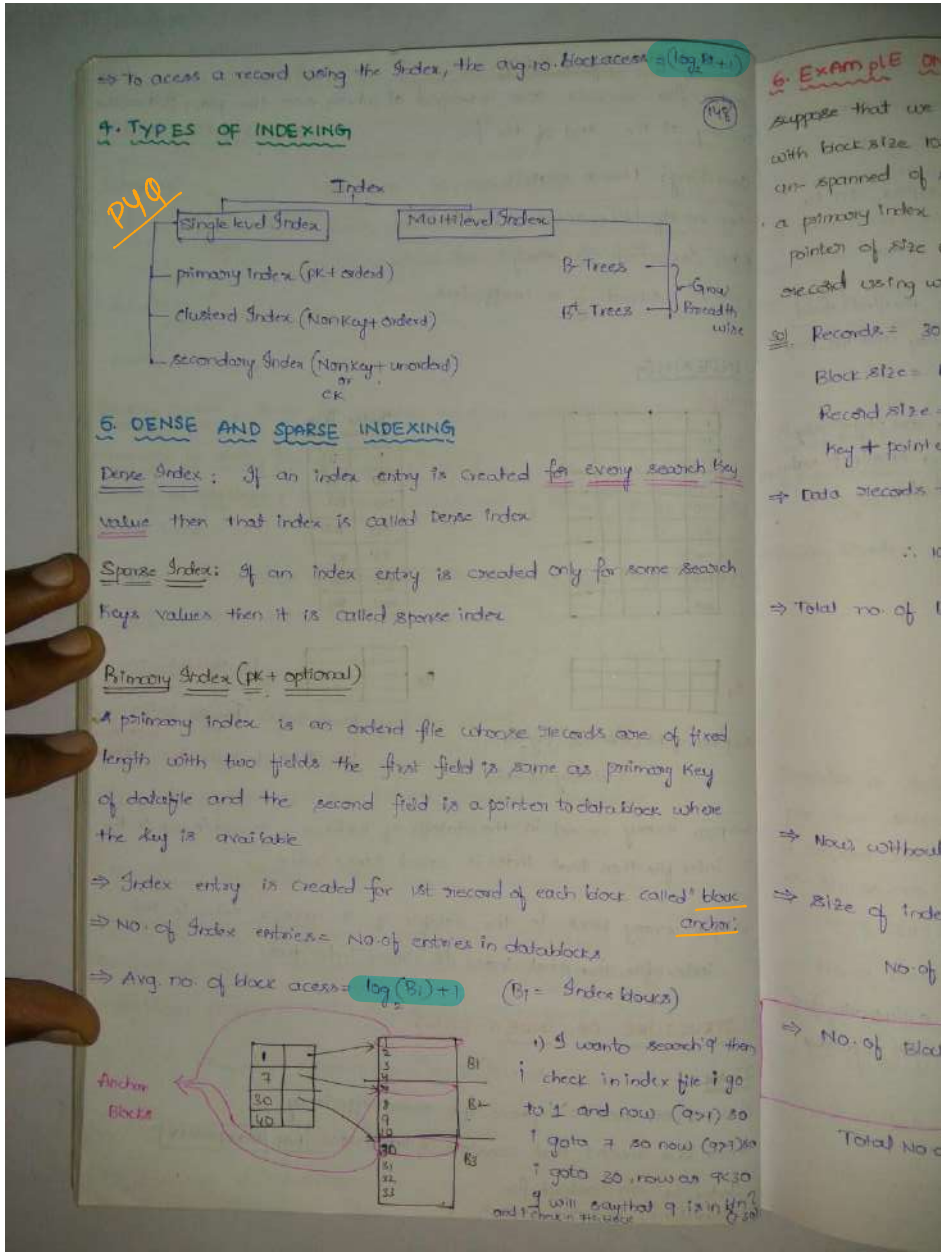
Index:

⇒ 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



Q.7) Negative Marks: 0.33 MCQ Subject: DBMS Max Marks: 1

Which of the following match is correct?

Type of Index	Number of first-level index entries
A. Primary	I. Number of records in data file
B. Clustering	II. Number of blocks in data file
C. Secondary (key)	III. Number of record or Number of distinct index field values
D. Secondary(non-key)	IV. Number of distinct index field values

Time taken to answer this question 00:01:33 hrs

Hide Answer | Add Notes | View Notes | Show Comments

A. A-I, B-IV, C-II, D-III

B. A-II, B-IV, C-III, D-I — Your answer is Wrong

C. A-II, B-IV, C-I, D-III — Correct Option

Solution:

The correct match will be:

Primary	II. Number of blocks in data file
Clustering	IV. Number of distinct index field values
Secondary (key)	I. Number of records in data file
Secondary (nonkey)	III. Number of records or Number of distinct index field values.

Q.15 Consider the following statements:

I. Primary index may or may not be sparse.

II. Secondary index may or may not be dense.

III. Unspanned strategy is suitable for fixed length records.

The number of the correct statements is/are _____.

⇒ PI can be dense/sparse (possible)

Sparse PI is preferred

⇒ SI over key always Dense

SI over Non key is sparse/dense

Copyright © MADE EASY EDUCATION Pvt. Ltd.

Q.12 Consider the following instance of the relation Images:

File_name	Folder_name	Created_by	Created_on	Theme
Scenery_1	Flowers	Isha	12-01-2021	Landscape
Scenery_2	Mountains	Rahul	17-02-2021	Landscape
Scenery_3	Flowers	Isha	12-01-2021	Landscape
Scenery_4	Mountains	Rahul	19-01-2021	Landscape

If File_name is selected as the primary key of the relation Images, X as the secondary key, and Y as the super key. The appropriate set of attributes for X and Y will be:

Time taken to answer this question 00:01:41 hrs

Hide Answer | Add Notes | View Notes | Show Comments

A. X = {Folder_name, Created_by}, Y = {Created_by, Theme}

B. X = {Created_on, Created_by}, Y = {Folder_name, Theme}

C. X = {Folder_name, Created_on}, Y = {File_name, Theme} — Your answer is Wrong

D. None of the above — Correct Option

Solution:

For X:

1. Folder_name, Created_by cannot be the secondary key or candidate key because, for 2 same tuples, it has different values File_name.
2. Similarly for Created_by, Created_on as well as for Folder_name, Created_on.

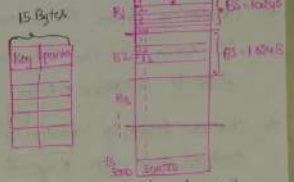
Therefore, none of the X values given in the options are correct. The correct value of Y could be anything that includes the attribute File_name. Therefore, the correct value of Y amongst given options are {File_name, Theme}

Hence, none of the options are correct.

6. Example on Primary Indexing

149

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 = $\frac{1024}{100} = \lfloor 10.24 \rfloor$

$= 10$

∴ 10 records can be placed in one block

$BF = \frac{\text{Block Size}}{\text{Size of each Record}}$

⇒ Total no of Blocks =

1 Block — 10 Records

x Blocks — 30,000 Records

$\Rightarrow \frac{x}{1} = \frac{30,000}{10} = x = 3000$ 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.

⇒ No of Block Access = $\lceil \log_2(45) + 1 \rceil = \lceil \log_2(2^{5.5}) + 1 \rceil$

$= \lceil 5.5 \rceil = 7$

Total No of Index Record = No of Data blocks = 3000

1 Block (Index file) → 68 Records

x → 3000 Records

$\Rightarrow \frac{x}{1} = \lceil \frac{3000}{68} \rceil = 45$

Draw Diagram

1. CLUSTURED INDEXING (NK+ ordered)

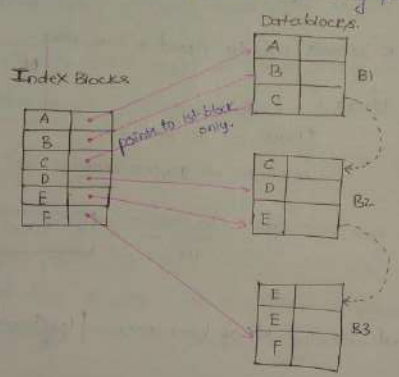
150

⇒ 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 doesn't have a distinct value for each record that field is called clustering field.



searching: Binary search

Avg. no of block Accss = $\log_2(B_i) + 1$

⇒ 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

2. SECOND

⇒ Index file

⇒ secondary

for which

⇒ Secondary

⇒ Index e

⇒ No. of ind

⇒ Type of

Index file

Avg. blk access

$= \log_2(B_i) +$

⇒ Binary Search

Consider a

previous Q

a record u

⇒ R = 30,00

⇒ No of dat

⇒ No of blo

⇒ size of In

⇒ No of Reco

⇒ No of Inde

1. Blo

8. SECONDARY INDEXING (NK/UK + unordered)

151

⇒ Index file should always be an ordered file

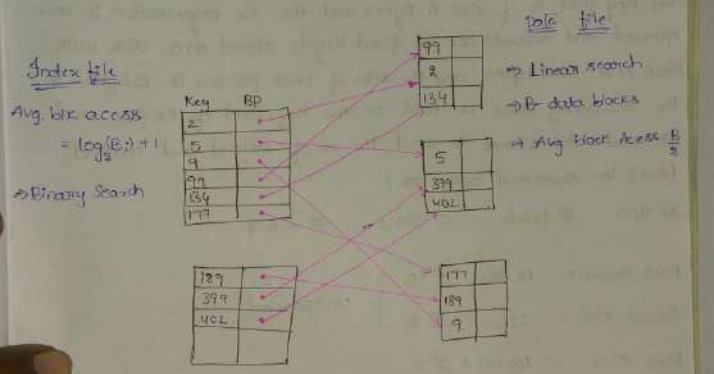
⇒ Secondary Index provides a secondary means of accessing a file for which some primary access already exists.

⇒ Secondary Index may be 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



Avg. blk access = $(\log_2(B_i)) + 1$

⇒ Binary Search

⇒ Linear search

⇒ B- data blocks

⇒ Avg block Access $\frac{B}{2}$

Consider a secondary index is built on the key field of the file of previous question. the find. no. avg no. of block Acess to access a record using with and without index

⇒ R = 30,000 (unordered)

⇒ No of data Blocks = 3000blocks

⇒ No. of block acess requisted without indexing = $B/_2$ = 1500

⇒ Size of Index record = 15 Bytes

⇒ No. of Records we can place in 1 block = $\left\lfloor \frac{1024}{15} \right\rfloor$ = 68

⇒ No of Index Records = No. of Data Records

= 30,000

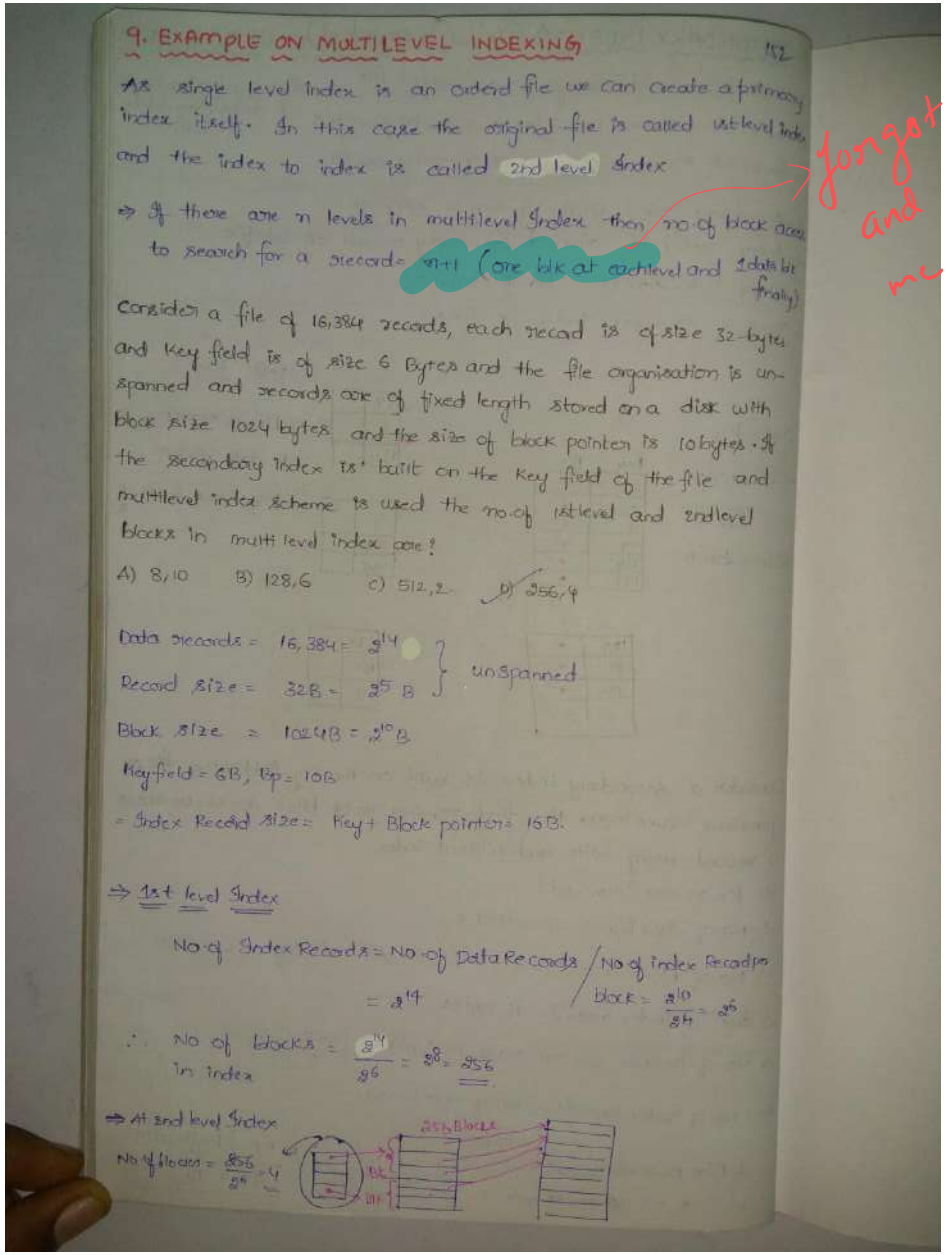
1 Block ⟶ 68 Records

x ⟶ 30,000 Records

$x = 441$ blocks

$\therefore$ 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 as 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 = 2^{14} / 2^4 = 2^{10}

∴ No of blocks in index = 2^{14} / 2^6 = 2^8 = 256

⇒ At 2nd level Index

No of blocks = 256 / 2^6 = 4

256 Blocks

forgot this and it cost me marks

Q. 27

Have any Doubt ?

Consider a disk with block size 1 KB, pointer size P = 9 bytes long. A file has 15000 employee records of fixed length. Each record has the following fields:

Fields	Size (in bytes)
NAME	20
ID	12
DEPTT	30
SALARY	15

Assume the file is ordered by the key field ID and primary index is based on ID. The number of levels needed to make it into a multi-level index is

2 Correct Option

Solution :

2

Record length = 20 + 12 + 30 + 15 = 77 bytes

Blocking factor (Bf) = $\left\lfloor \frac{1024}{77} \right\rfloor = 13$

Number of blocks needed for file = $\left\lceil \frac{15000}{13} \right\rceil = 1154$

Index entry size = (ID + P) = (12 + 9) = 21 bytes

Index blocking factor = $\left\lfloor \frac{1024}{21} \right\rfloor = 48$

Number of 1st level index blocks = $\left\lceil \frac{1154}{48} \right\rceil = 25$

Number of 2nd level index blocks = $\left\lceil \frac{25}{48} \right\rceil = 1$

Since the 2nd level has only one block it is the top index level.
So only 2 levels are required.

Q.14) Negative Marks: 0.66 MCQ Subject: DBMS Max Marks: 2

Consider a file of 8192 records. Each record is 16 bytes long and its keys field is of size 6 bytes. The file is ordered on a key field, and the records of data file and index file are not split across disc blocks. The file is stored in a file system with a block size of 512 bytes, and the size of a block pointer is 10 bytes. If the primary index is built on the key field of the file, and a multivalued index scheme is used to store the primary index, the number of first level and second level blocks in the multilevel index are respectively

Time taken to answer this question 00:03:47 hrs

Mock Test-6

A 16 and 1

B 8 and 1

C 16 and 2

D 8 and 2

Show Answer Add Notes View Notes Show Comments

Block size = 2^m * 2^n, where m,n is 1st level and 2nd level index

4

No of records in one block=512/16=32

No of data blocks=8192/32=256

No of entries in one index block=512/16=32

No of first level index blocks=256/32=8

Best answer

No of second level index block=1(8 entries would fit in single block)

answered Dec 2, 2015 · selected Dec 2, 2015 by Umang Raman

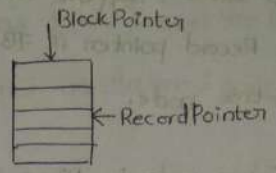
https://www.cpp.edu/~ftang/courses/CS241/notes/b-tree.htm

10. INTRODUCTION TO BTREES AND B+ TREES

⇒ BTree and B+ Trees are Dynamic

⇒ Node pointer and Block pointer

⇒ Record pointer



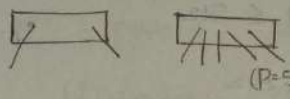
⇒ All the databases today use B+ Trees.

⇒ The advantage of B & B+ Trees is they grow vertically

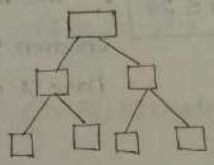
11. PROPERTIES OF BTREES

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.



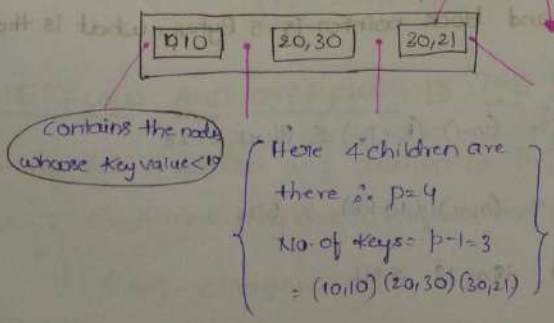
2) Internal nodes:



⇒ Internal nodes can store upto (p-1) keys.

⇒ Have between $\lceil p/2 \rceil$ and p children

The general Representation is $\langle P_1, \langle K_1, P_{r_1} \rangle P_2 \langle K_2, \_ \rangle \cdots P_q \langle K_{q-1}, P_{q-1} \rangle$



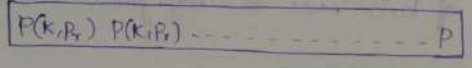
3) Leaf node : - stores between $\lceil (p-1)/2 \rceil$ and p-1 keys

- 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



⇒ The formula that every node should satisfy is

$= n * p + (n-1)(k + P_r) \leq \text{Block size}$

$= n*6 + (n-1)(9+7) \leq 512$

$= 6n + 16n - 16 \leq 512$

$= 22n \leq 528 \Rightarrow n \leq 24$ ∴ The max amount of children that this tree can have is 24.

n = order of the tree
p = Block pointer size
P_r = Record pointer

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:

$= n * p + (n-1)(k + P_r) \leq \text{Block size}$

$= n*5 + (n-1)(10+8) \leq 512$

$= 5n + 18n - 18 \leq 512$

$= 23n \leq 540$

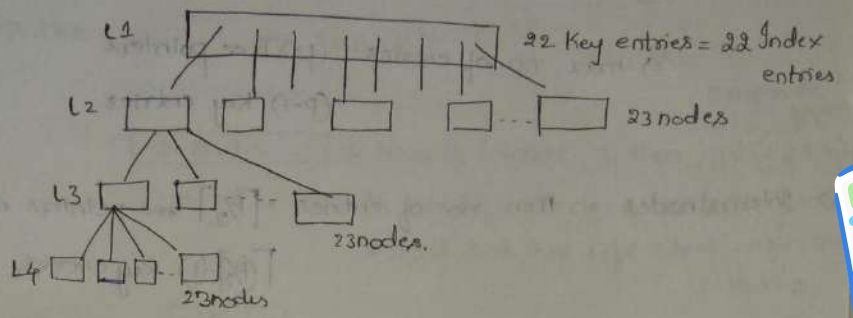
$= n \leq \lfloor 23.478 \rfloor$

$= n = 23$

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

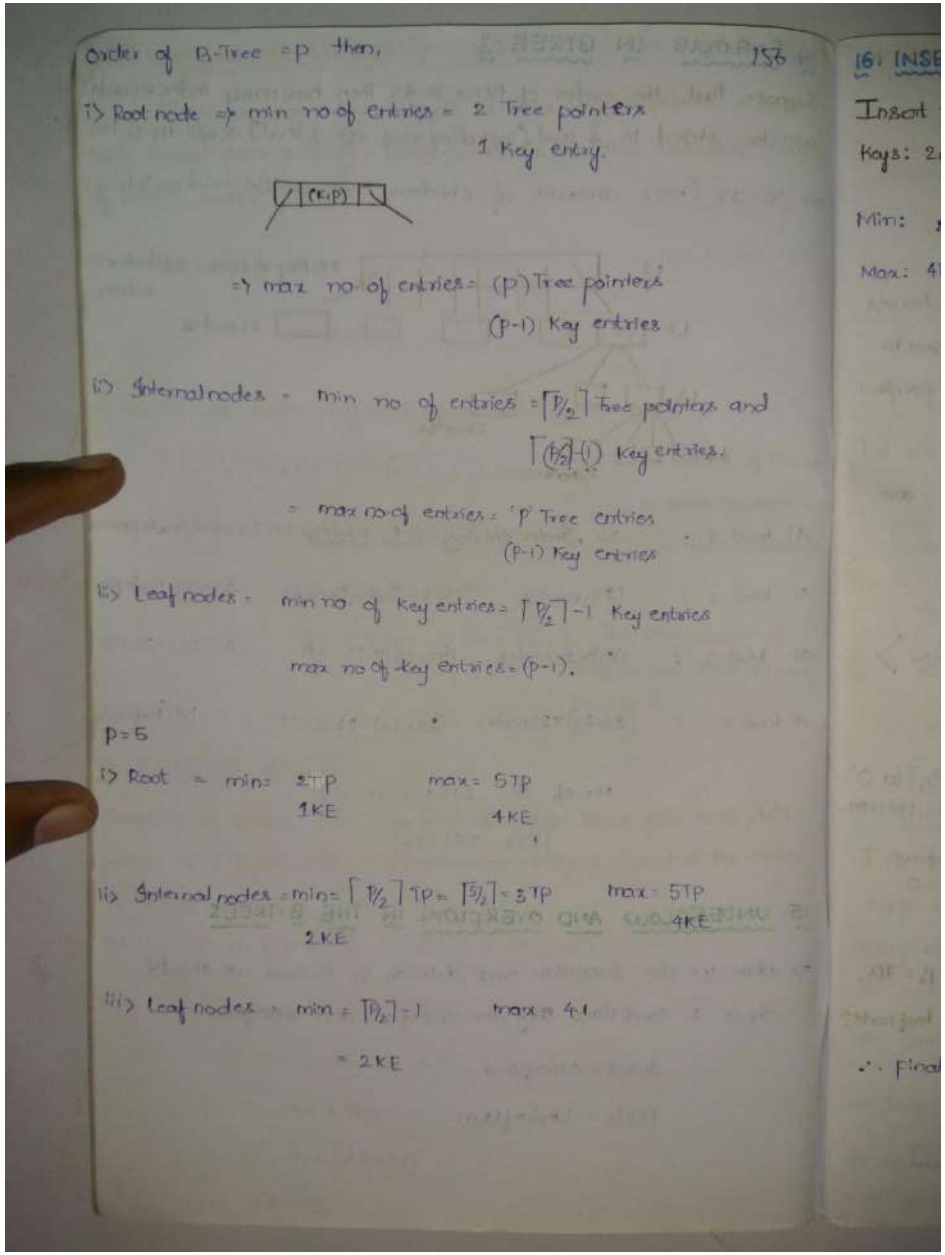
IRs = 267674

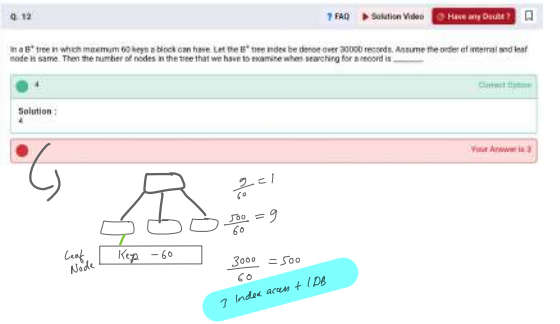
15. UNDERFLOW AND OVERFLOW IN THE B-TREES

⇒ while we do Insertion and deletion in B-Trees we should check 2 conditions they are overflow and underflow

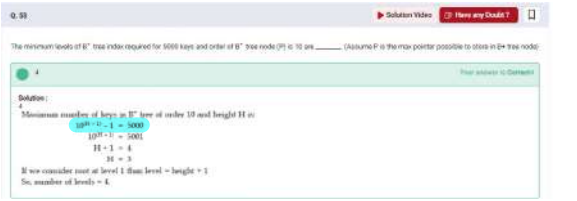
Insert - overflow

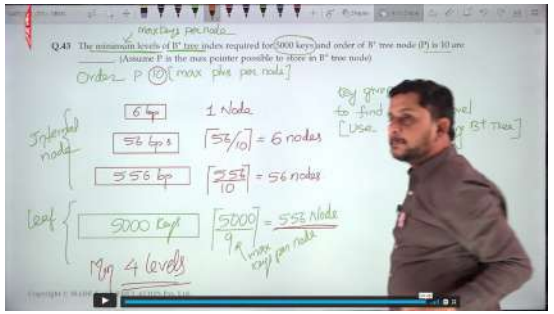
Delete - Underflow.


Order of B-Tree = p then,
156
i) Root node => min no of entries = 2 Tree pointers
1 Key entry.
(K|P)
=> max no of entries = (p) Tree pointers
(p-1) Key entries
ii) Internal nodes - min no of entries = ⌈p/2⌉ Tree pointers and
⌈(p/2)-1⌉ Key entries.
= max no of entries = 'p' Tree entries
(p-1) Key entries
iii) Leaf nodes - min no of Key entries = ⌈p/2⌉-1 Key entries
max no of key entries = (p-1).
p=5
i) Root = min: 2TP
1KE
max: 5TP
4KE
ii) Internal nodes = min: ⌈5/2⌉ TP = ⌈5/2⌉ = 3TP
2KE
max: 5TP
4KE
iii) Leaf nodes = min = ⌈5/2⌉-1
= 2KE
max = 4.1
16. INSE
Insert
Keys: 2
Min:
Max: 4


Q. 12
FAQ
Solution Video
Have any Doubt ?
In a B+ tree in which maximum 60 keys a block can have. Let the B+ tree index be dense over 30000 records. Assume the order of internal and leaf node is same. Then the number of nodes in the tree that we have to examine when searching for a record is _____
4
Correct Option
Solution :
4
Your Answer is 3
Leaf Node
Keys - 60
3000/60 = 500
1 Index access + 1DB


Q. 52
Solution Video
Have any Doubt ?
For order M, the root node will have keys between 1 to 2 M and for other nodes keys will be in between M to 2 M, where the size of the disk block is given as 2048 bytes while the block pointer and record pointer are of 30 and 25 bytes respectively. Which of the following is correct regarding the order of the B+ tree if the size of the search key is given as 20 bytes
A
Your answer is INCORRECT
B
Correct Option
Solution :
(b)
The definition of order given in the question is based on number of keys. Since, order will be based on maximum occupancy which is given as 2 M keys.
Order for non-leaf node of B+ tree:
Order of leaf node of B+ tree:


Q. 53
Solution Video
Have any Doubt ?
The minimum levels of B+ tree index required for 5000 keys and order of B+ tree node (P) is 10 are _____ (Assume P is the max pointer possible to store in B+ tree node)
Solution :
4
Minimum number of keys in B+ tree of order 10 and height H is:
10^H - 1 = 5000
10^H = 5001
H = 4
H = 3
If we consider root at level 1 than level = height + 1
So, number of levels = 4.


Q.43 The minimum levels of B+ tree index required for 5000 keys and order of B+ tree node (P) is 10 are _____ (Assume P is the max pointer possible to store in B+ tree node)
Order = P (10) (max ptrs per node)
6 bp
1 Node
56 bp s
⌈56/10⌉ = 6 nodes
556 bp
⌈556/10⌉ = 56 nodes
5000 bp
⌈5000/9⌉ = 556 Nodes
Min 4 levels

16. INSERTION INTO BTREES -- EXAMPLE -1

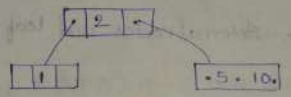
Insert the following keys in B-Tree of order - 4.

Keys: 2, 5, 10, 1, 6, 9, 4, 3, 12, 18, 20, 25.

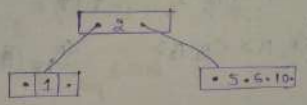
	Root	Intermediate nodes	Leafs
Min:	1TP, 1KE	2TP, 1KE	2TP, 1KE
Max:	4TP, 3KE	4TP, 3KE	0TP, 3KE

⇒ Now, if I insert 1 then I get (1 2 5 10) Overflow (so split)

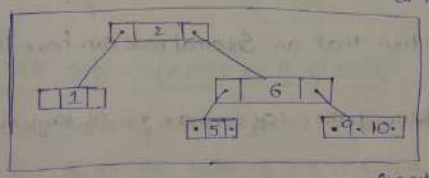
In which middle ele = 2 so split it into 2 nodes such that Left side = 1 Right = 5 10 & Root = 2



⇒ Now inserting 6 (6 > 2) so move in RST

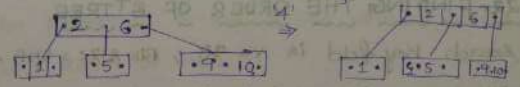


⇒ If 9 is inserted it becomes (5 6 9 10) overflow. split at midpoint = 6.

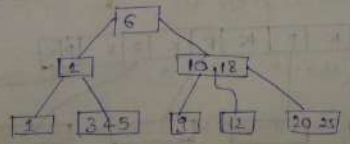


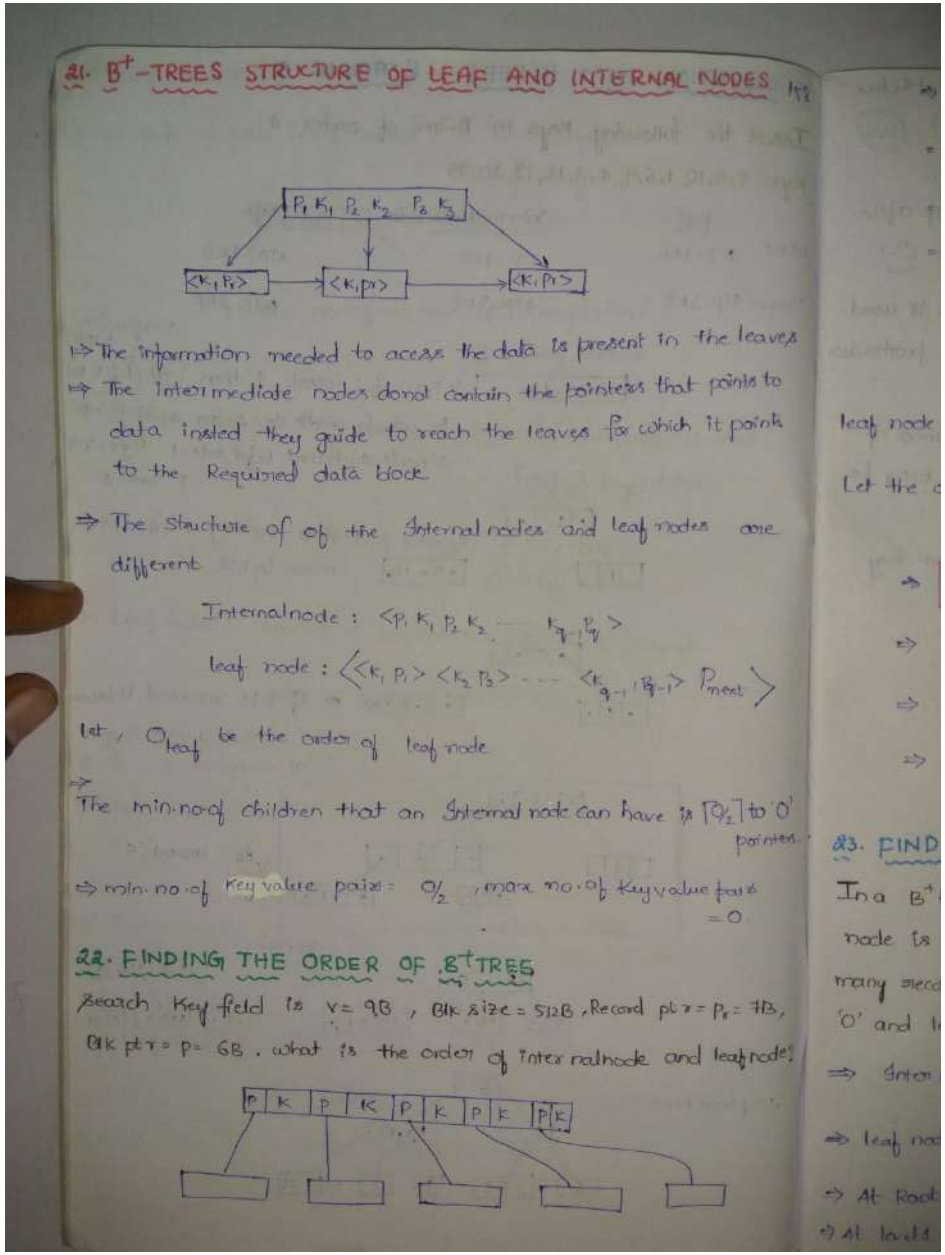
X

Inserting 4 ⇒



∴ Final tree =





Steps for insertion in B+ Tree

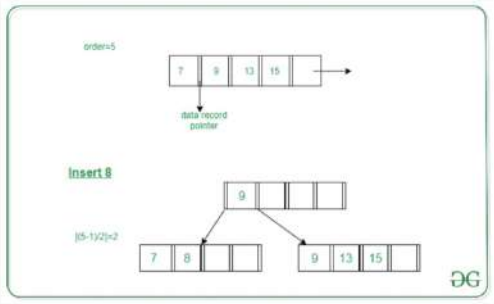
1. Every element is inserted into a leaf node. So, go to the appropriate leaf node.
2. Insert the key into the leaf node in increasing order only if there is no overflow. If there is an overflow go ahead with the following steps mentioned below to deal with overflow while maintaining the B+ Tree properties.

Properties for insertion B+ Tree

Case 1: Overflow in leaf node

1. Split the leaf node into two nodes.
2. First node contains **ceil((m-1)/2)** values.
3. Second node contains the remaining values.
4. Copy the smallest search key value from second node to the parent node.(Right biased)

Below is the illustration of inserting 8 into B+ Tree of order of 5:



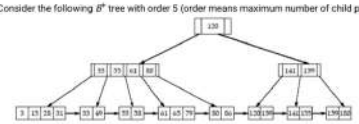
Case 2: Overflow in non-leaf node

1. Split the non leaf node into two nodes.
2. First node contains ceil(m/2)-1 values.
3. Move the smallest among remaining to the parent.
4. Second node contains the remaining ke

Q. 13

? FAQ ▶ Solution Video @ Have any Doubt ?

Consider the following B+ tree with order 5 (order means maximum number of child pointers a node can have)



How many addition records could be added in the above B+ tree without altering the height?

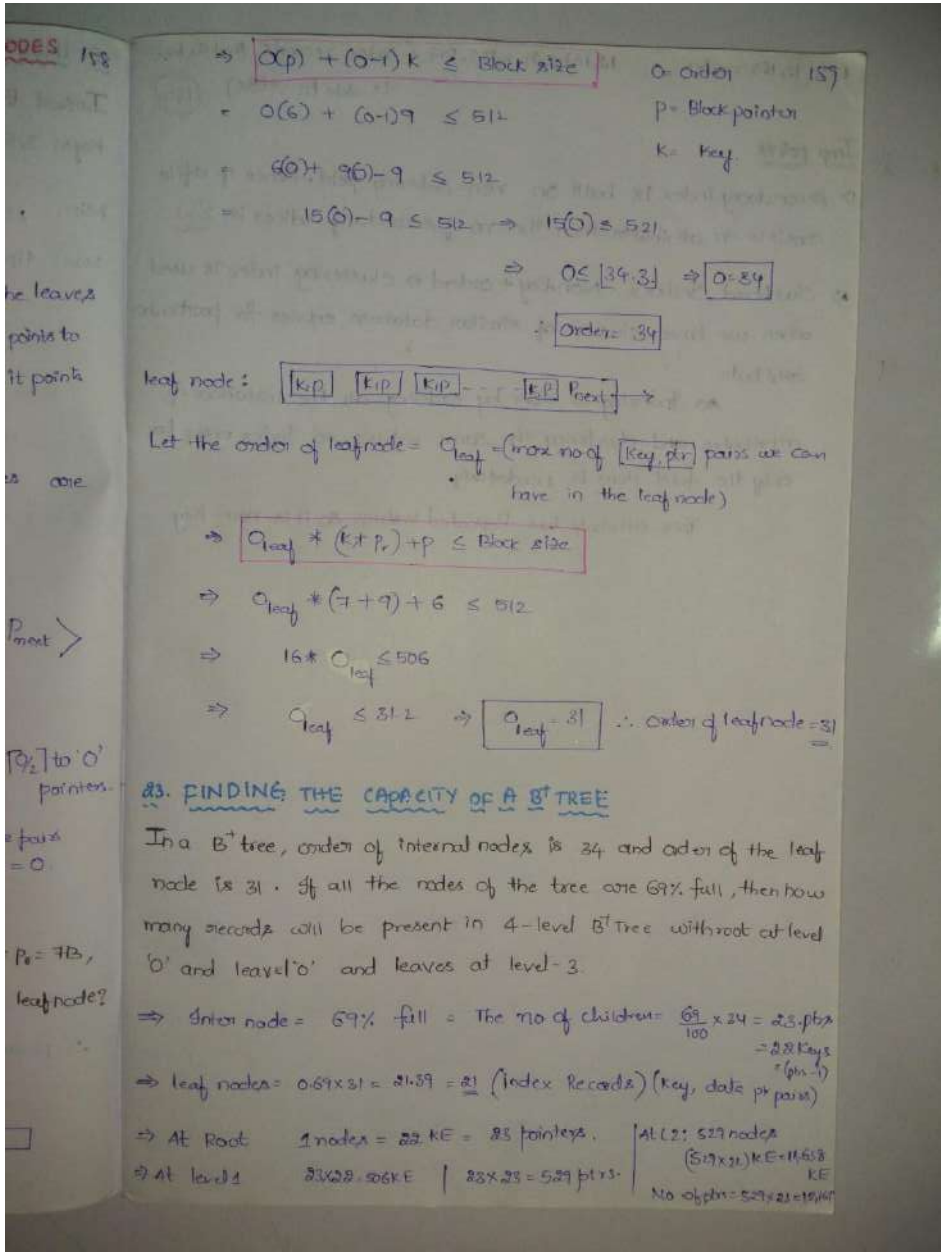
A 100

B 75

C 13

D 81 Correct Option

Solution :
(d)
- Root node can point 5 internal nodes.
- Total maximum number of records this B+ tree can have = 5 × 5 × 4 = 100
- Total additional records can be added
  = 100 - 19 (available) = 81



⇒ $\boxed{\alpha(p) + (\alpha-1)k \leq \text{Block size}}$ — 157

α = order, p = Block pointer, k = Key

$= 6(6) + (6-1)9 \leq 512$

$= 6(6) + 9(6) - 9 \leq 512$

$= 15(6) - 9 \leq 512 \Rightarrow 15(\alpha) \leq 521$

$\Rightarrow \alpha \leq 34.3 \Rightarrow \boxed{\alpha = 34}$

$\therefore$ Order = 34

leaf node: [k|p] [k|p] [k|p] ... [k|p] [P_{next}] →

Let the order of leaf node = α_{leaf} (max no of [Key, ptr] pairs we can have in the leaf node)

$\Rightarrow \boxed{\alpha_{leaf} * (k + p) + p \leq \text{Block size}}$

$\Rightarrow \alpha_{leaf} * (7 + 9) + 6 \leq 512$

$\Rightarrow 16 * \alpha_{leaf} \leq 506$

$\Rightarrow \alpha_{leaf} \leq 31.2 \Rightarrow \boxed{\alpha_{leaf} = 31}$ $\therefore$ order of leaf node = 31

83. 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 leaves at level-3

⇒ Inter node = 69% full = The no of children = $\frac{69}{100} \times 34 = 23.46$ ptrs = 22 Keys (ptrs − 1)

⇒ leaf nodes = $0.69 \times 31 = 21.39 = 21$ (Index Records) (Key, data ptr pairs)

⇒ At Root 1 node = 22 KE = 23 pointers.

⇒ At level 1 23 × 22 = 506 KE | 23 × 23 = 529 ptrs. | ALL: 529 nodes, (529 × 21) KE = 11,109 KE, No of ptrs = 529 × 21 = 11,109

Q. 49

Solution Video | Have any Doubt ?

What is the maximum number of nodes in B+ tree possible with order 3 and height 3 is _____

40 — Correct Option

Solution :

40

B^+ tree with order 3 means that at maximum we can have 3 block pointers in a node while the minimum is 2. For getting the maximum number of nodes, we need to utilize the maximum possible block pointer as a node in next levels. So the tree will be like:

Level	Maximum nodes	Maximum block pointers
0	1//root node	3
1	3	3 × 3 = 9
2	9	9 × 3 = 27
3	27	27 × 3 = 81
Total nodes	40	

The height of the node is the number of edges on the longest path from the node to a leaf. So we have taken upto 4th level, so this way root to leaf, number of edges will be 4 and maximum number of nodes will be 40.

Your Answer is 162

13: 12,167 nodes 12,167 × 21 = 255,507 (index records that the index 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.

8 Marks weightage in the last 3 years GATE Paper

2021(Set-1) : Marks: 8
Relational Algebra: 3 marks(easy to medium level question), FD and normalisation: 2 marks (easy), Transaction and concurrency control:3 marks (easy and majority questions are from conflict serialisability).

2021(Set-2) : Marks: 8
Relational constraints:1 mark(easy), SQL:2 marks(easy), FD and Normalization:2 marks(easy), Transaction and concurrency control: 2 marks (easy), Indexing:1 marks(easy)

2020 : Marks : 8
2 marks question is asked from SQL and Transaction concurrency control. Important topics are: ER diagram, SQL, Transaction concurrency control

2019 : Marks : 8
Two 2 marks questions from querying(RA, SQL). Rest of the questions are from FD, Concurrency and B,B+ trees. NOTES: Queries asked are bit easy to answer

Analysis by Gate Applied Course

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Difficulty	Topic Name	2019	2018	2017 Set-1	2017 set-2	2016 Set-1	2016 Set-2	2015 Set-1	2015 Set-2	2015 Set-3	2014 Set-1	2014 Set-2	2014 Set-
moderate	ER-Diagrams		1		1			2					
easy	Relational Schema- Key Constraints				1	1					1	2	
moderate	Relational Algebra	2	2	2					2				
moderate	Tuple Relational Calculus			2									
moderate	SQL	2	2	1	2		2	3		1	2	4	
easy	Functional Dependencies and Normalization	2	2	1		1				1	3		
very easy	Transactions and Concurrency control	1		2	2	3	3		3	2	2	2	
easy	Indexing and File Structures (B-Trees and B+ Trees)	1			2		1		1	2			
	Total Marks	8	7	8	8	5	6	5	6	6	8	8	