Date:
Page:

Collection of fields → Record or tuple
Collection of records → file
Collection of all files → database

* Database is collection of interrelated data
* DBMS → Database + set of program etc to [access] (retrival, insertion, deletion, updation) those database.

Characteristics of DBMS

1. Real World Object
2. Relation based table means structured data
3. Isolation of data & Application → more file can run without affecting other files.
4. Less redundancy
5. Consistency - data must be in original state.
6. Query language → SQL
7. ACID properties
   ↳ Atomicity (either all or none), consistency, Isolation, Durability
8. Multiple user & concurrent execution
9. Multiple views
10. Security → backup & recovery mechanism

User of DBMS

1- Administrator ⇒ DBA (by Sys Analyst) → overall design of database (schema designing), granting permissions to user, Backup & Recovery Procedure, conflict resolution
2- Designer / Developer
3- End-users

- Team leader → good motivator, psychologist

Advantages of Database BMS

- Control redundancy
- Data Sharing
- Reduced Time / Increased Throughput ↑ work done in per time unit
- Multiuser — only in read mode can access at same time
- Backup
- Security

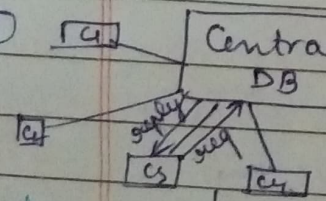
Disadvantages

- Costly hardware & Software — see
- Higher impact of failure — complexity

Classification of DBMS / Types

① - Centralized DB
② - Distributed DB

Dumb Terminal
↓
without CPU

① 

It comforts users to access stored data from diff loc. through www app.

Disadv
① Not easy to update with an entire DB
→ dec risk of data manu
→ data consistency maintained as it manage data in central repository

② Every terminal is intelligent terminal
Difficult to implement distributed DB because of complex structure

Advan :- Modular dev possible
→ server fail will not affect entire data rel

Difference b/w File management & DBMS

FMS	DBMS
⇒ Small sized programs - Written in C, C++ & Java	Large sized system e.g. Oracle, Sybase etc.
⇒ Relatively cheaper	Relatively expensive.
⇒ Structure is simple	⇒ Complex structure.
⇒ Single user	⇒ Multiuser
⇒ Requires less preliminary knowledge of ~~~~ design	Requires more preliminary knowledge of design
⇒ less secure	More secure
⇒ Simple backup & recovery process	Strong backup & recovery procedure.

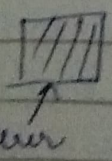
Database System Architecture

- 1 tier Architecture
- 2-tier arc
- 3-tier arc

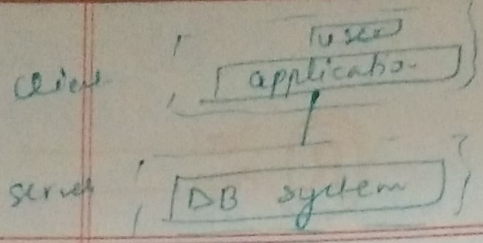
⇒ based on client/server

2-tier arc: application client ↔ server

1-tier



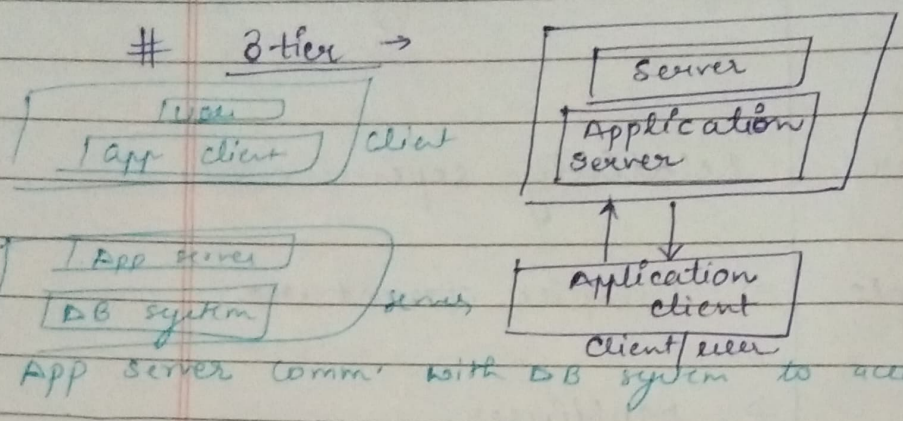
one client/server



2 tier → No of clients connected to one server

⇒ Interface → API, ODBC, JDBC

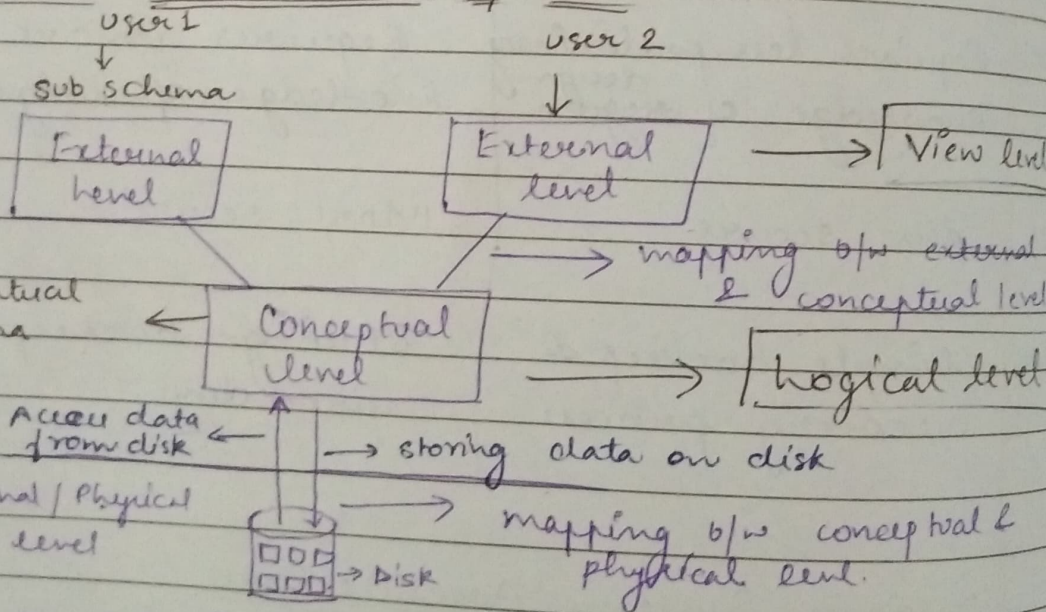
3-tier →



- To reduce load of server, introduce app server layer.

App server comm' with DB system to access data

3 Level Architecture of DBMS



Internal schema

↳ to tell how data is to be stored in disk & what data str. you are using.

⇒ Only 1 physical & 1 conceptual but many of external level

Allocating part of conceptual schema to sub schema is advantageous. → multi user can access portion of conceptual as per requirement

Date:
Page:

Data independence

Whenever you are modifying certain things which doesn't affect DB architecture.

- Physical Data independence
  - ⇒ Modification at physical level & does not affect higher level (conc. or logical level)
- Logical Data independence
  - ⇒ Modification at conc. or logical level, it doesn't effect phy. or higher level.

Database Language

- Data Definition Lange (DDL) → to create DB/ alter the structure or to drop the table
  - → specifies DB schema
- Data Manipulation Lang (DML) → Retrieval (Access)
  - ↓ Updation
  - ↓ Deletion
  - ↓ Insertion
- SQL → supports both DDL & DML
- TCL → Transcation Control Lang → Grant & evoke
- Data Control Lang.
- DSL
- SDL
- VDL → View Definition Lang.
- HL → Host Lang.
- 4G.L. → 4 Gen Lang.
- TCL

char varchar
↓ ↓
fixed size

SQL

Roll	Name	Add	Branch
1	Amit	Lko	IT
2			
3			
4			
5			

⇒ Create a Table

```
SQL > CREATE Table student
      Roll Number(2),
      Name varchar(30),
      Address varchar(40),
      Branch varchar(20),
      Mob. Number number(10);
// Table Created
```

⇒ To insert the data

```
SQL > INSERT INTO
        student;
      VALUES 1, 'Amit', 'Lko', 'IT', 998...;
// 1 row updated
```

⇒ Display all content

```
SQL > SELECT * FROM student;

SQL > SELECT Roll, Name FROM student;

SQL > SELECT Roll, Name FROM student
      WHERE Address = 'Lko';
```

⇒

Employee (Emp id, Emp name, Emp designation, Mob, Salary, address)

Get emp details who are getting more than 5000 salary

```
SQL > SELECT *F
```

Date:
Page:

① SQL > SELECT * FROM Employee
WHERE Salary > 50,000;

② SELECT * FROM Employee
WHERE Salary > 50000 AND salary < 100000;
WHERE Salary BETWEEN (50,000, 100000);

③ SELECT * FROM Employee
WHERE Salary IN (20,000, 50,000, 70,000, 80,000);

④ SELECT Emp-id, Emp-name FROM employee
WHERE Emp designation = "Manager" AND
Address = "LKO";

// in sorted order of employee name
⑤ SELECT Emp-id, Emp-name FROM employee
ORDER BY emp-name asc;

// disp name of emp whose name starts with 'c'
⑥ SELECT Emp-name FROM employee
WHERE emp-name LIKE "C%";
"_UV";

// Add coleumn age
⑦ ALTER Table Employee
Add Age number(2);

// By giving 10% hike in salary
⑧ SELECT * FROM employee
WHERE SALARY += 0.10 * SALARY;

UPDATE Table Employee
SET Salary = (Salary + Salary * 10) / 100;

SQL query to get salary 10% hike who are working as manager

(9) UPDATE Table Employee
~~block~~ SET Salary = Salary + 0.10 * Salary
where emp_designation = "Manager";

(10) DELETE * FROM employee
where emp_id = "E10";

(11) RENAME Employee To Emp;

Other fn conversion Date func. } Functions in SQL
- Number fn - Aggregate fn - character fn

Library → Dual

absolute (number fn)

⇒ SQL > SELECT ABS(-200) Forom Dual;
= 200

⇒ SQL> SELECT CEIL (74.23) Forom Dual;
= 74

⇒ SQL> SELECT POWER (9,2) Forom Dual;
= 81

⇒ SQL> SELECT SQRT (81) Forom Dual;
= 9

⇒ SELECT FLOOR (74.5) Forom Dual;
= 75

Date:
Page:

⇒ SELECT MOD (81,6) From Dual;
= 3

⇒ SQL > SELECT ROUND (125.5,0) (125.55,1)
= 124 125.6

⇒ SQL > SELECT TRUNC (123.55, 1)
Truncate
= 123.5

⇒ SQL > SELECT SIN(x), Cos, Tan

⇒ SQL > SELECT SINH(x)

⇒ LN(data)

⇒ LOG(b, data)

Employee

Emp ID	Emp Name	Salary	Dept
1		NULL	
20		—	

Aggregate function

⇒ SQL > SELECT AVG (salary) FROM Employee;

⇒ SQL > SELECT COUNT (Salary) FROM Employee;
COUNT * (Salary) → All rows.
COUNT DISTINCT (Salary)

⇒ SQL > MIN (SALARY) / MAX (Salary)

SELECT * FROM Employee;
WHERE Salary = MAX (Salary);

SELECT Emp_Id, Emp_Name, Desig. MAX(Salary)

⇒ SQL> SUM (SALARY)

⇒ SQL > STDDEV (SALARY)

⇒ SQL > VARIANCE (SALARY)

Character Function

INITCAP → convert first character to capital & remaining char. lower case

⇒ SQL > SELECT INITCAP ("RAJIV") FROM Dual;
⇒ Rajiv

⇒ SQL> SELECT LENGTH ("RAJIV") From Dual;
= 5

SUBSTR → to find loc of particular char.

⇒ SELECT SUBSTR ("Delhi_University" 5,3)
= i-U

⇒ INSTR ("GGSIPU, Delhi", IPU) FROM Dual;
= 4

⇒ GREATEST (85, 95.5, 40.2)
⇒ 95.5

⇒ LEAST (85, 95.5, 40.2)
= 40.2

Non Procedural → focus on what

Date: Page:

Relational Algebra

Set - oriented operation
- Union
- Intersection
- Cartesian Product
- Set - difference

Relation oriented operation
- Selection (σ)
- Projection (π) → vertical value
- Join ($\bowtie$)

① R

Id	Name
101	Amit
103	Akash

S

Id	Name
101	Amit
102	Amy

R ∪ S

Id	Name
101	Amit
103	Akash
102	Amy

R ∩ S

Id	Name
101	Amit

R - S

Id	Name
103	Akash

R - S ≠ S - R

R × S ⇒ 4 tuples

Id	Name	S.ID	S.Name

② Employee

E.Id	E.Name	Des.	Salary

Employee 2

E.Id	Qualification

⇒ SQL> SELECT * From Employee
WHERE Salary > 40000;

⇒ RA > σ (Employee)
Salary > 40000

⇒ RA > π (~~Eid~~, Employee)
Eid, salary > 40000

Display Emp.Id & Name & Sallry whose qualification is 'MBA'

②
SELECT *
Emp. AS
SELECT *
FROM Emp2,

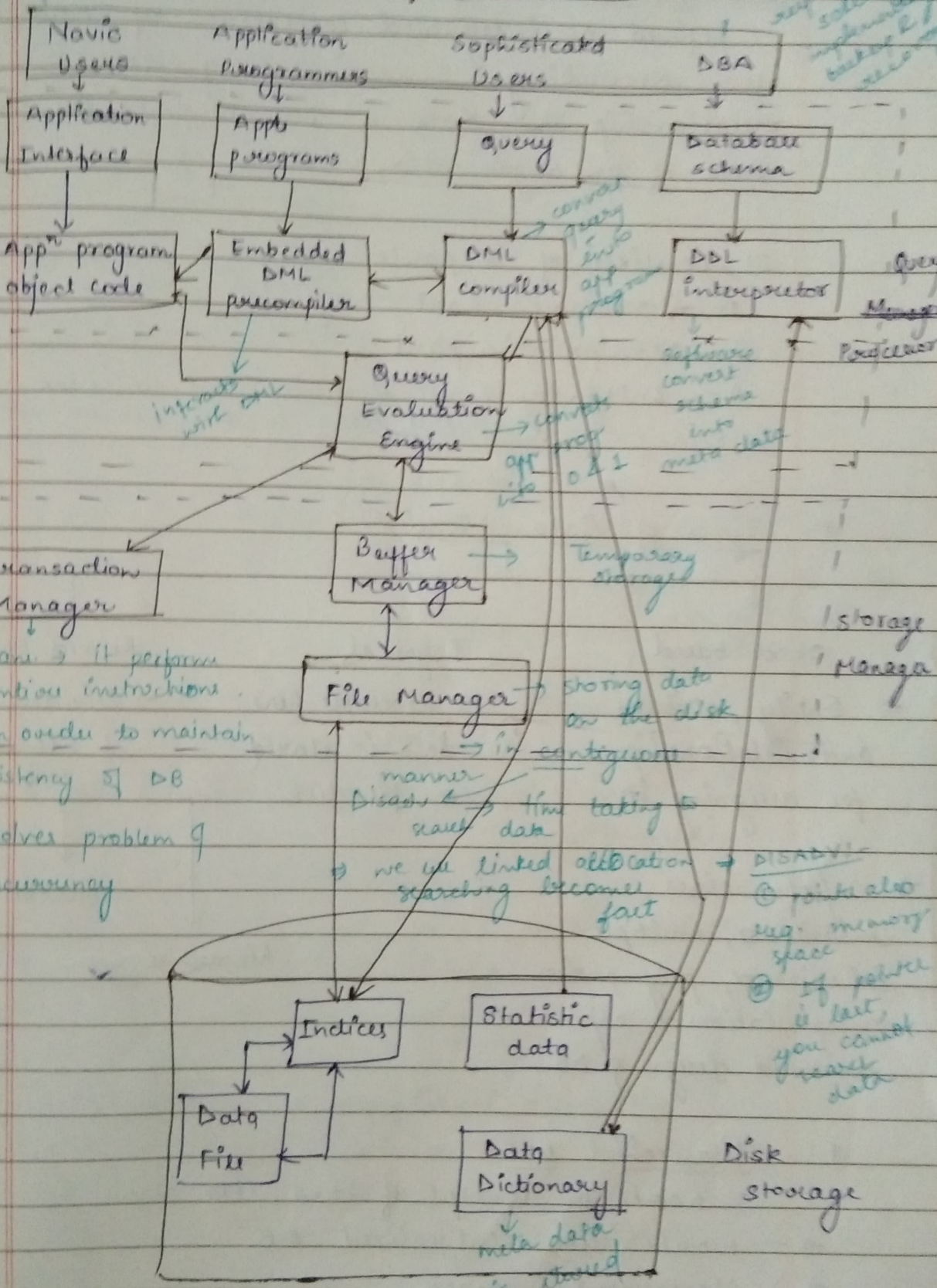
①
SELECT E.Id, E.Name, Salary From Employee
SELECT E.Id, Qualification
WHERE Qualification = "MBA"

RA> σ (Employee, ⋈ Employee 2)
Qualification = "MBA"

SELECT * From Employee ∪ Employee 2
WHERE Qualification = "MBA"

WHERE E.Id . Employee = E.Id . Employee 2

Overall Structure of Database



Data Model

Defined as integrated collection of concepts for describing & manipulating data, relationship b/w data & constraints on the data of an organisation.

It consists of 3 parts
1- Structure chart → set of rules that can be used to design DB
2- Manipulative part → Retrival & updation
3- Integrity → Set of rules used to check whether data is accurate or not

Types of Data Model

- Object-based
  - ⇒ Entity, attributes and relationship for designing it
  - ⇒ Objects, class, inheritence, multiple inheritence, polymarphism encapsulation → obj. oriented design of DB
- Physical
  - → describe how data is stored on the disk
- Record based
  - → Divided into 3 categories
    - Network
    - hierarchial
    - Relati

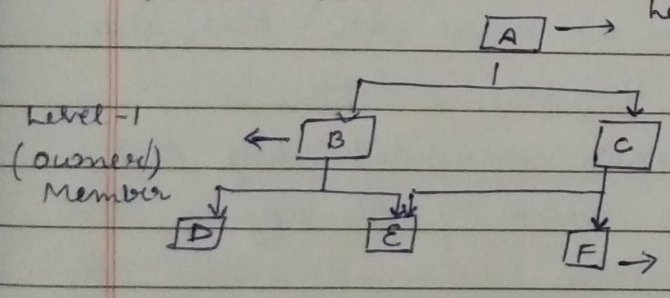
1. Hierarchial DM
   - works on concept of tree like str.
   - → works like centralized DB
   - → Management is easy
   - → If root → overloaded, system will fail.

→ for recovery, redundancy is advantageous.

Advantage → Simplicity
→ Data sharing
→ Data integrity
→ Data security
→ Efficiency

Disadvantages → It can't support many-to many relationship.
→ Implementation is difficult
→ No standard
→ operational anamilies → inconsistency in data resulting from operation like update, insertion or deletion.

2. Network DM



Many-to-Many relationship.

Advantage → Simplicity
→ 1:M, M:1, M:M relationships
→ Data sharing
→ Data Independency

Disadvantages → operational anomilies
→ Implementation is difficult upto some extent

3. **Relational DM**

no operational anomalies.

RDBMS → oracle, MySQL

→ Tuple → One complete row

→ Cardinality → No of rows

→ Degree → No of attributes / columns

Adv :- Can be easily queried.

→ provide consistent DB design

Disadv :- When it comes to high volume transaction on large amount of data storage, slows of query can occur.

4. Key :-

Superkey :- set of one or more att that allow us to identify uniquely a tuple.

Primary key :- To denote candidate key that is chosen as principle means of identifying tuples within relation.

UNIT-3 Normalisation

→ process of removing / minimizing the anamolies from a table / relation

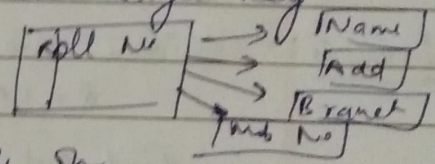
Anamolies :-
- Redundancy - by decomposing relation
- Insertion anomalies
- Deletion "
- Updation

Functional Dependency :-
association of two attributes - determinant & determined.

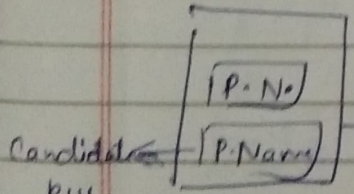
Eg: $R \rightarrow N$ ⇒ Nam is functionally dependent on R

key attr. { R-N, R-A, R-B, R-M } Non-key attr.

⇒ When more non-key attr. depend on single key → fully func dependency.



→ When non key attr. depends on more than one key



→ Partial func dependency

Rules :-

Armstrong axioms

1. Reflexive rule
2. Augmentation rule
3. Transitive Dependency :-

1.) If Two attr. A, B & $B \subseteq A \Rightarrow A \to B$.

2) If $A \to B \Rightarrow CA \to CB$

3) If $A \to B$ & $B \to C \Rightarrow A \to C$

- Union Rule

$$\left.\begin{array}{l} A \to B \\ A \to C \end{array}\right\} \Rightarrow A \to BC$$

- Decomposition Rule

If $A \to BC \Rightarrow A \to B$ & $A \to C$

- Pseudo transitivity Rule

If $A \to B$ & $BC \to D$

$\Rightarrow AC \to D$

Example: Given $A \to B$, $B \to C$, $BC \to D$

$R = (A, B, C, D)$

Find out closure of FDs (F^+)

$A \to B$ is given

$\Rightarrow CA \to CB$

$A \to B$ & $B \to C$ is given

$\Rightarrow A \to C$. (Transitive Dependency)

$A \to B$ & $BC \to D$

$\Rightarrow AC \to D$ (Pseudo Trans. Dep.)

Date:
Page:

Qw: Let R be relation ; R = (A, B, C, G, H, I)
and FDs are given $A \to B$, $A \to C$, $CG \to H$,
$CG \to I$, $B \to H$

Find out F^+

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

$\Rightarrow A \to BC$ (Union)

$CG \to H$ & $CG \to I$

$\Rightarrow CG \to HI$ (Union)

$A \to B$ & $B \to H$

$A \to H$ (Transv.).

$A \to B$ & $CG \to H$ $\Rightarrow$ ~~A~~. $AG \to I$ (PS)

$AG \to H$ (PS)

Qw R = (A, B, C, D, E, F)

FDs : $C \to F$, $E \to A$, $EC \to F$, $A \to B$, $F \to D$

Find Key.

E & C are the Keys.

$\{EC\}^+ = \{E, C, F\}$

$\{EC\}^+ = \{E, C, F, A\}$

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

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

Qw R = (A, B, C, D, E, F, G)

FDs : $A \to B$, $BC \to E$, $BC \to D$, $AEF \to G$

$B \to G$, $B \to C$

ACF = {A, C, F, E, B, G, D} Find Key. (Candidate)

$A \to E$

$AC \to D$

$A \to G$

$\{BC\}^+ = \{B, C, E, D, G\}$

$\{AC\}^+ = \{A, C, D, E, G\}$

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